Introduction to δ-artithmetic : computation of integer approximate gcd and factorization.

Christian Fotso

Projet - ***ULAS***

INTRODUCTION A LA δ-ARITHMETIQUE

CALCUL DU PGCD, DES PG-CD APPROXIMATIFS, FACTORISATION …

PROJET DE THESE

FOTSO TALLA Christian

Christian.fotso@yahoo.fr

Tel :

A la mémoire de NGOUNOU KAMTO Reine Pascale

A l'adorable petite fille, à l'enfant aimé, à cette brillante intelligence et cette magnifique personnalité dont tu as incessamment ébloui tous ceux qui ont eu le privilège de te connaitre durant ta vie trop brève.

Et avec toi, aux milliers d'intelligence que l'Afrique subsaharienne enterre chaque jours et si jeunes. A Ces fleurs non écloses dont du parfum nous n'aurons connu que la promesse. A Ces trésors inestimables disparus dans d'absurdes conflits, des famines qu'on aurait pu éviter, des maladies qu'on aurait su traiter…

A travers ces quelques pages j'ai voulu élever pour vous un petit monument au cœur des mathématiques, pour qu'au pied des ce petit amas de formules le monde se souvienne de ce qu'il a perdu en ne sachant pas vous accueillir, et qu'ici vos intelligences qui n'ont pas eu le temps de s'épanouir trouvent enfin le repos.

ABSTRACT

Cette thèse constitue une introduction à la δ-arithmétique (delta arithmétique). Cette dernière propose une extension de l'arithmétique classique en y introduisant une notion d'incertitude (ou de flexibilité) sur les données utilisées pour résoudre un problème donné. Dans ce cadre, la solution à un problème, n'est plus soumis à l'immutabilité des données initiales, mais les données tout comme les solutions sont deux paramètres d'un δ-problème et dans ce cadre, peuvent être ajustés afin d'arriver à une solution optimal du δ-problème, l'optimalité de celui-ci résidant dans le compromis entre l'ajustement des données initiales et les propriétés de la solution induite.

In introduisant cette flexibilité, la δ-arithmétique se veut plus proche des problématiques réelles des ingénieurs. En effets, quand l'ingénieur fait face à un problème d'arithmétique, celui-ci constitue rarement une fin en soi. La solution du problème d'arithmétique à vocation à être réutilisée pour résoudre le problème réel auquel l'ingénieur est confronté, et pour résoudre ce problème il est souvent crucial que la solution du problème d'arithmétique possédé un certain nombre de propriétés. Par ailleurs les données sur lesquels il s'appuie ne sont pas toujours aussi exact qu'en théorie, en réalité, les données sont le plus souvent affectées d'une erreur plus ou moins importante, soumis à l'imprécision des instruments de mesures utilisés pour les récolter, voir volontairement affecté d'erreur pour empêcher leur utilisation par des personnes qui n'en sont pas destinataire (cryptographie). La nature des données couplées aux propriétés souhaitées des solutions offre donc un certain degré de liberté dans la manipulation de ces quantités et la résolution du problème initiale.

Dans ce cadre, la reformulation du problème initiale en un δ-problème, permet de remettre les données dans l'ensemble des variables ajustable en vue de trouver une solution optimale tout en limitant leur ajustement à un seuil δ, qui est en quelque sorte la marge d'erreur admissible, et qui dépends du type de problème. Lorsque celle-ci est nulle, les solutions du δ-problème se résument à celles du problème d'arithmétique classique, et comme nous le verrons il peut parfois être plus simple de résoudre le problème d'arithmétique classique dans l'univers de la δ-arithmétique, en le considérant tout simplement comme un δ-problème de δ nul.

Pour illustrer notre propos, nous nous attaquerons à deux grands problèmes d'arithmétique classique, celui du calcul du PGCD et celui de la factorisation, nous en proposerons une reformulation et terme de δ-problème, ce qui nous mènera au problème de PGCD approximatif de Howgrave-Graham [1] et au problème de factorisation approximative, nous verrons alors comment la δ-arithmétique peu proposé une formulation rigoureuse de ces problèmes et comment elle peut permettre de les résoudre.

Mot clés : δ-arithmétique, Pgcd, pgcd approximatif, Pgcd d'ordre α, δ – pgcd, Pgcd de série d'intervalle, factorisation.

Table des matières

Chapitre 0

INTRODUCTION GENERALE

EXEMPLE INTRODUCTIF

Nous introduisons ce travail par une brève présentation du cas pratique, qui m'a été soumis en Mai 2011 et qui m'a conduit à m'intéresser à ce sujet. En tant qu'ingénieur financier je suis souvent amener à trouver des solutions de contournement pour corriger les problèmes rencontrés quotidiennement dans la mise en place des algorithmes de valorisation de produits financiers. Le cas qui va vous intéresser ici est celui de la valorisation d'un actif appelé CDO dont le nom a été vulgariser par la dernière crise car pointé du doigt comme étant une des sources de celles-ci. Nous ne présentons bien sur ici que ce qui dans la valorisation du produit à un lien avec la notion de PGCD approximatif, cet exemple nous permettra de bien comprendre les raisons profondes qui ont guidé la construction de cette théorie de la δ-arithmétique.

Les CDO sont des produits financiers construits à partir de panier d'actif. Leur valorisation dépends entre autre du nominal de chacun des actifs composant le panier. Une approche classique de valorisation de ces actifs est la méthode Monte-Carlo. Pour la mettre en place, les nominaux doivent être découpé en multiple d'une valeur fondamentale qui représentera l'unité de perte minimale (la Loss Unit « LU »), c'est en tant que multiple de cette LU que les nominaux initiaux seront manipulé dans l'algorithme.

L'unité de perte idéale serait alors le PGCD des différents nominaux, car il permettrait de réécrire de façon exacte, chacun de nominaux initiaux comme un multiple de ce PGCD. Par contre nous devons noter qu'un algorithme de type Monte-Carlo est en générale un algorithme à durée d'exécution longue. Dans le cas de notre CDO le temps d'exécution de l'algorithme dépendra directement de l'ordre de grandeur de la LU : plus la LU sera grande, plus l'algorithme sera rapide. D'un autre coté la précision du résultat dépendra de la capacité des multiple de la LU choisit à reproduire les nominaux initiaux (idéalement leur PGCD). Nous avons donc un arbitrage entre la taille de la LU qui nous permet d'aller plus rapidement, et l'exactitude du PGCD qui nous permet d'obtenir un résultat plus précis, d'où le dilemme. Notre but était de donner aux ingénieurs la possibilité de choisir entre la taille du PGCD et l'exactitude de la division, et surtout de maitriser le niveau d'erreur qu'impliquait leur arbitrage.

Considérons par exemple un panier de 3 actifs sur lequel on constate les pertes suivantes : $X = (50, 125, 250)$. Le PGCD de cette série est 25. Nous pouvons donc considérer une LU=25 et utiliser pour la simulation le panier X = (2*LU, 5*LU, 10*LU), c'est en quelque sorte un cas idéal, l'unité de perte retenue nous permet de reconstituer de façon exact le panier initial, et est suffisamment grand (par rapport à nos données initiales : le plus petit des nominal n'est divisé que par 2) pour que l'algorithme puisse se terminer rapidement.

Supposons que le lendemain on constate des nouvelles valeurs : $X = (50, 124, 251)$ le nouveau PGCD est égal à 1, cependant, retenir une LU=1 nous forcerait à simuler un panier du type (50*LU, 124*LU, 251*LU), ce qui pourrait s'avérer fatal pour notre algorithme Monte-Carlo en augmentant de façon drastique le temps de simulation nécessaire pour obtenir un prix. Or cette série est pratiquement identique à celle de la veille et sa valorisation n'est pas vraiment différente de la précédente. En réalité ça ne nous gênerait pas d'utiliser pour la valorisation le panier (2*LU, 5*LU, 10*LU) de la veille. Mais comment formaliser cette possibilité ?

Si le PGCD semblait de prime abord répondre de façon parfaite à notre besoin, sa sensibilité extrême aux faibles variations des données, et le fait que cette qu'il ne soit pas toujours en rapport (en termes d'ordre de grandeur) avec les données utilisées pour la calculer la rendent inadaptée à notre contexte. En fait le PGCD est un problème mal posé au sens de Hadamard, i.e. que un faible changement sur les entrée entraine une modification importantes des sorties, ces changement ne sont pas monotones et leur amplitude est sans rapport réel avec l'amplitude de variation des données initiales.

Nous avions donc besoin d'une unité de pertes ayant les caractéristiques suivantes :

- *moins sensible à ces faibles variations (pour que l'algorithme de simulation soit plus stable).*
- *Plus en rapport en termes d'ordre de grandeur avec les données initiales.*

Par ailleurs nous étions prêts à concéder de légères modifications de nos données initiales.

C'est pour nous adapter à ce contexte que nous avons développé de ce cette nouvelle notion de pgcd approximatif et ce après avoir essayé plusieurs approches heuristiques ne répondant que trop partiellement à nos besoins initiaux.

Les chapitres 1 et 2 présentent les base de cette nouvelle arithmétique, et offre les outils permettant de peser t de manipuler les valeurs en prenant en compte leur inscription dans un voisinage.

A PROPOS DU PGCD APPROXIMATIF

Dans l'approche traditionnelle du PGCD, une place prépondérante est donnée à la notion de Diviseur commun (CD) au détriment de l'ordre de grandeur du diviseur retenu. Ce qui dans bien des cas conduit à un PGCD dont l'ordre de grandeur est sans rapport avec les données initiales (c'est par exemple le cas des nombres premiers entre eux). Or si on s'autorisait une recherche du PGCD non plus uniquement parmi les diviseurs stricts mais aussi parmi les diviseurs approchés, on augmenterait de façon considérable nos chances de trouver des « PGCD » dont l'ordre de grandeur est plus en ligne avec nos données initiales, c'est ce constat qui nous a menés au problème du PGCD approximatif.

Le problème du PGCD approximatif introduit par Howgrave-Graham [1] en 2001. Problème réputé difficile il a depuis son introduction fait l'objet de nombreuses publications. Cependant les algorithmes et heuristiques (en général basé sur les treillis) proposés pour le résoudre sont le plus souvent complexe, difficile à mettre en place et ne résolvent que de façon très partielle la question. Par ailleurs malgré le fait que ce problème soit devenu aujourd'hui une des problèmes majeur d'algèbre computationnel, et trouve de nombreuse application en cryptographie par exemple, il manque encore un cadre théorique permettant de l'énoncer clairement, d'où la multiplicité des définitions. C'est pour combler ce manque que nous proposons cette nouvelle arithmétique des bulles (δ-arithmétique) qui généralise l'arithmétique classique en prenant en compte de façon naturelle l'incertitude ou la possibilité d'ajustement des données initiales.

Le problème du PGCD approximatif a été posé par Howgrave-Graham dans les termes suivant : étant donné deux entiers x_0 et x1, et trois bornes E_0, E_1 et D. le problème consiste en la recherche des entiers d vérifiant : $d| x_0 + e_0$ et $d| x_1 + e_1$ avec $d > D$ et $|e_0| < E_0$ et $|e_1| < E_1$. Si $e_0 = 0$, alors x_0 est connu de façon exact le problème est alors celui de l'approximation partielle du PGCD (PADP) sinon, le problème est désigné par approximation généralisé du PGCD (ACDP).

Howgrave-Graham propose alors deux solutions pour résoudre ces problèmes, la première est basée sur les fractions continues et la seconde est une approche par treillis (lattice) basé sur la méthode introduite par coppersmith. Cette dernière méthode est encore à la base de la plus part des solutions proposées à ce problème aujourd'hui. Dans leur article daté d'aout dernier Henry cohn et Nadia Heninger [2] proposent une généralisation de ce problème au cas de n entiers et une solution reposant sur les treillis.

La réputation difficile de ce problème à conduit les cryptographes à le mettre au cœur de certain système de cryptographie. En 2009 par exemple Marten van Dijk, Craig Gentry, Shai Halevi, et Vinod Vaikuntanathan [3] proposent un système de cryptographie dont la sécurité repose sur la difficulté du problème de l'approximation du PGCD.

Cet article utilise la δ-arithmétique pour construire un cadre théorique permettant de poser de façon clair et de répondre efficacement à cette question du Pgcd approximatif. Nous y construisons un algorithme de calcul du pgcd, en analysons le cout, et proposons en annexe un code en C++ implémentant cet algorithme.

Une approche à laquelle certain pourraient penser pour résoudre ce type de problème est l'utilisation de l'arithmétique des intervalles. En effet avec l'arithmétique des

intervalles il est possible de tenir compte de l'incertitude sur les données et de construire ce faisant un intervalle contenant de façon certaine le résultat recherché. Pour ce faire, l'arithmétique des intervalles remplace tous nombres par un intervalle le contenant et effectue ensuite le calcul sur les bornes de l'intervalle de façon à obtenir un encadrement du résultat exact. Les limites d'une telle approche apparaissent lorsqu'il faut résoudre des problèmes mal posés au sens d'Hadamard i.e. ceux ou une légère altération des données initiales est susceptible de provoquer une forte variation du résultat finale, c'est clairement le cas du problème de calcul du PGCD d'un ensemble X. il n'est donc pas possible d'encadrer le PGCD d'un ensemble X entre les PGCD de deux ensembles contenant ce dernier. C'est pour répondre à ce type de problème (mal posé) que nous nous proposons de construire cette nouvelle approche, contrairement à l'arithmétique des intervalles nous n'essayons pas de produire un encadrement de la solution finale à partir d'un encadrement des données initiales, mais de construire pour un sous ensemble de solutions possibles, la meilleurs réponse au problème posé.

Le calcul du Pgcd approximatif est abordé au chapitre 4 et l'algorithme de calcul est analysé au chapitre 9.

A PROPOS DE LA FACTORISATION

Le problème de la factorisation des entiers est aujourd'hui au cœur de la sécurité dans nos sociétés numérisées, il y a plus d'un siècle déjà Gauss écrivait à ce propos

« La reconnaissance des nombres premiers et des nombres composés avec leur décomposition en facteurs premiers est connue pour être des plus importants et utiles en arithmétique. Il a tant impliqué le zèle et la sagesse des géomètres anciens comme modernes qu'il serait superflu d'en discuter plus avant... En plus, la dignité des sciences mêmes semble exiger que tous les moyens possibles soient explorés pour trouver la solution d'un problème si élégant et si célébré »

Plus d'un siècle le chalenge n'est toujours pas totalement relevé et la complexité du sujet en a fait le pilier même sur lequel repose certain de nos plus importants systèmes de cryptographie.

Cependant A. Lenstrass qui est un des grands spécialiste contemporain de la question rappel que la seule preuve que nous ayons de la difficulté de la factorisation est le fait que malgré une recherche très actives sur la question, nous ne disposons toujours pas d'algorithme efficace permettant de résoudre ce problème.

La question de la factorisation d'un entier N peut-être envisagé sous deux angles différents :

- Soit N un entier, trouver U, V entier tel que N = U*V

- Soit N un entier et M un entier vérifiant $1 \leq M \leq N$ existe-t-il un diviseur d de N vérifiant $1 \leq d \leq M$.

Notre étude de la factorisation n'a pas été motivée directement par l'un ou l'autre de ces problèmes, mais par notre intérêt pour le pgcd approximatif, aussi le problème auquel nous nous intéressons ici est celui de la factorisation approximative. Il peut être formulé ainsi :

Soit N un entier, trouver U et V tel que :

$U \leq \lceil\sqrt{N}\rceil$ et $V \geq \lceil\sqrt{N}\rceil$, $N = U * V + E, E \leq \frac{P}{2}$ et ou E est le plus petit possible pour P supérieur à une borne fixé (ou encore E inférieur à un seuil donné).

Ce problème est équivalent à celui de la factorisation lorsqu'on impose 0 comme seuil maximale pour E.

Nous traitons ici du problème de factorisation approximative, mais nous construisons aussi une nouvelle approche permettant de traiter le problème plus spécifique de la factorisation. L'algorithme que nous en déduisons pour la factorisation d'un nombre illustre bien le fait que difficulté de la factorisation d'un nombre N dépends surtout du rapport (U/V) des facteurs recherchés.

Nous proposons une nouvelle approche permettant de factoriser efficacement un nombre N lorsque l'un de facteur du nombre à factoriser est « proche » de la racine du nombre en question. Puis nous construisons un procédé permettant de lever cette contrainte de proximité du facteur recherché et de la racine du nombre à factoriser.

L'approche que nous proposons permet de retrouver U et V en un nombre d'opération approximativement égale à $\left(\frac{L}{\sqrt{\sqrt{N}}}\right)^2$, ou $L := U - V$ est la distance qui sépare les deux facteurs. Elle permet aussi de démontrer en k opérations qu'il n'existe pas de diviseur D de N vérifiant $\sqrt{k\sqrt{N}} \leq D \leq N$

Une fois établi, nous construirons un processus dit de rééquilibrage des facteurs permettant de réduire la distance L et donc de réduire le nombre d'opération nécessaires à la factorisation de N. ceci débouche sur la construction d'un indice k, fonction des deux facteurs, et permettant de mesurer le cout de la recherche de ces deux facteurs. Nous montrons que cet indice est d'autant plus faible que le rapport U/V est proche d'une fraction m/n : aussi la factorisation de N = U*V est d'autant moins couteuse que $\varepsilon = \left(\frac{U}{V} - \frac{m}{n}\right)$ est petit et que m et n sont petits.

Un des principaux intérêts de cette nouvelle approche de la factorisation est sa grande simplicité. En s'appuyant comme pour le calcul du pgcd approximatif sur la recherche et l'exploitation des structures stables cet algorithme confirme l'intérêt de cette nouvelle approche de résolution des problèmes d'arithmétique.

Le problème de la factorisation est traité au chapitre 5.

EXEMPLE DE CALCUL DU PGCD-APPROXIMATIF

Afin de mieux illustrer nos propos, nous vous proposons à travers un exemple de suivre pas à pas la construction des PGCD qui seront introduit et détaillé plus loin. Bien sur tous les concepts introduits dans cet exemple seront ensuite explicité dans la suite de l'article. Il s'agit pour nous à travers un exemple de rendre plus concret les concepts que nous allons introduire avant de nous embarquer dans la construction mathématique détaillé.

Comme premier exemple je vous propose la première série qui a été porté à mon attention lorsque le problème m'a été soumis, il démontre bien la capacité de la δ-arithmétique à répondre efficacement à ce type de question.

X = {45 000 000 ; 50 000 000 ; 66 666 667}
Bien évidement son PGCD est 1.

Nous allons maintenant utiliser l'approche de recherche du pgcd par division approximative pour déterminer un pgcd approximatif de cette série.

Pour un taux d'erreur admis de 1% on a par exemple :

A l'ordre 6

Générateur Initial : P = Arrondi (45 000 000 /6) = 6 666 667

Structure Générée : Q = {6, 7, 10}

Intervalle des générateurs pour D = 0.01 : P = Vide (P-min = 7 135 714 > P-max = 6 673 334)
Générateur Idéal : P =
Ensemble des Erreurs : E =
Taux d'erreur :

A l'ordre 12

Générateur Initial : P = Arrondi (45 000 000 /12) = 3 336 333

Structure Générée : Q = {12, 15, 20}

Intervalle des générateurs pour D = 0.01 : P = [3 330 000 – 3 333 333]
Générateur Idéal : P = 3 333 333
Ensemble des Erreurs : E = {4, 5, 7}
Taux d'erreur :

A l'ordre 24

Générateur Initial : P = Arrondi (45 000 000 /24) = 1 666 667

Structure Générée : Q = {24, 30, 40}

Intervalle des générateurs pour D = 0.01 : P = [1 665 000 – 1 668 334]
Générateur Idéal : P = 1 666 667
Ensemble des Erreurs : E = {8, 10, 13}
Taux d'erreur :

Cet exemple nous montre qu'en choisissant par exemple le générateur idéal d'ordre 3, on réussit à obtenir un pgcd 3 333 333, en ne concédant que de très petits ajustement sur les donnée initiale, pour notre problème de pricing par exemple, c'est ajustement n'aurais absolument aucun impact sur la valorisation du CDO, alors qu'utiliser un pgcd de 3 333 333 améliorerait infiniment la rapidité du calcul. Le but de cette thèse est donc d'établir un procédé rigoureux de construction de ces pgcd approximatif.

Chapitre 1

INTRODUCTION A LA δ-ARITHMETIQUE

"Nature is a good approximation of Mathematics"
--Zvi Artstein (Rutgers Univ. Colloquium, Dec. 6, 2002).

Cette citation de Zvi Artstein illustre bien la difficulté qu'il peut y avoir à passer de la perfection du monde mathématique aux approximations monde réel. En effet, contrairement au monde idéal – et idéalisé - des mathématiques, l'environnement réel est fait d'imperfection, d'inexactitude et d'incertitude. Et si l'ingénieur au cours de sa formation est rompu à la manipulation rigoureuse d'entités mathématiques parfaites, il se voit moins confronté une fois sur le terrain à une réalité fort différente, ou il se doit de conjuguer avec des contraintes techniques et des enjeux d'efficacité d'une part, et d'autres parts des imperfections et des inexactitudes des données sur lesquels il doit s'appuyer. Pour moi le rôle de l'ingénieur est de construire des ponts entre la perfection des théories et les contraintes et imperfections de la réalité. La δ-arithmétique se veut pour les problèmes d'arithmétique, le pont permettant d'adapter de façon optimale les concepts théorique aux réalités et contraintes de l'ingénieur qui s'appuie sur ces derniers pour résoudre des problèmes plus vastes.

Pour bien comprendre d'où vient le divorce entre mathématiques et réalité, considérons par exemple l'arithmétique et son théorème dit fondamentale. Celui-ci stipule que « tout nombre peut s'écrire de façon unique comme produit de facteurs premiers», la désignation de ce théorème comme fondamentale illustre bien le fait qu'il est au cœur de la construction de tout l'édifice arithmétique. Pourtant, si cette vision des nombres offre aux mathématiciens une approche parfaite pour leurs manipulations, elle n'en est pas moins loin de la réalité de l'ingénieur pour qui un nombre représente le plus souvent une quantité réelle. Regarder les nombres comme produit de facteurs premiers modifie totalement la nature de ces derniers, et le rapport qu'ils peuvent entretenir. La succession des nombres –et donc la proximité des quantités qu'ils représentent – disparaît au profit d'un rapport plus abstrait sur leur décomposition, ce qui ne favorise pas toujours la confrontation des concepts qui en découle à la réalité des problèmes que doivent résoudre les ingénieurs. Si par exemple pour l'ingénieur financier les sommes 1000000 et 999983 sont quasiment interchangeables, leur décomposition en facteur premier ne leur prête aucun rapport 999983 étant d'ailleurs premier.

La δ-arithmétique propose de réinscrire les nombres dans leur environnement, et de conserver les rapports de proximité qu'ils entretiennent et qui sont cruciaux pour offrir aux ingénieurs la flexibilité dont ils ont besoin dans leur manipulation. Pour ce faire nous introduisant le concept de δ-voisinage, qui permet de regrouper ensemble des quantités proches et interchangeable dans la résolution d'un problème. Ces δ-voisinages, sont à la δ-arithmétique, ce que les nombres sont à l'arithmétique.

Une fois les nombres sortie de leur isolement et inscrit dans leur δ-voisinage, embarquant ainsi avec eux à chaque instant un ensemble d'autre nombre jugé suffisamment proche pour lui être substitué sans nuire à la résolution du problème qui nous intéresse. Une fois inscrit dans ce voisinage, l'entité de départ devient une sorte d'idéal que la δ-arithmétique s'efforcera au mieux de conserver, et qui lorsqu'il devra absolument être substitué à un de ces voisins, le sera avec une volonté de lui rester le plus « proche » possible. Le nombre étant remplacé par l'ensemble de ses δ-voisins, le problème est alors de redéfinir sur ces nouvelles entités, les concepts fondamentaux de l'arithmétique classique (égalité, divisibilité, facteur, diviseur commun …), et d'utiliser ces concepts pour résoudre les δ-problèmes issue de la reformulation des problèmes d'arithmétique classique (factorisation, calcul du PGCD …).

L'ensemble de ses δ-concept nous permet de reformuler les problèmes de façon à y introduire le degré de flexibilité dont l'ingénieur à besoins pour apporter au problème une réponse optimale, ils nous permettent de formuler de façon rigoureuse le δ-problème dont la solution constitue une réponse optimale au problème réel, prenant en compte la possibilité d'ajustement – plus ou moins important - des données initiales et les contraintes sur les solutions recherchées.

Une fois le δ-problèmes formulé, il faut bien sur le résoudre. Une approche naïve du problème pourrais consister tout simplement à teste successivement l'ensemble des solutions possible du problème pour trouver la solution optimale, cette approche sera bien sur limité rapidement par le nombre de tests à effectuer. Aussi la δ-arithmétique propose d'identifier et d'exploiter des objets dites structures stables, celles si regroupe des sous-ensembles de solutions et permet de les traiter directement plutôt qu'individuellement, aussi les algorithmes utilisé serons d'autant plus efficace que les structures stable regrouperons des sous ensemble important de solutions possibles. La résolution des δ-problème se résume donc à l'identification et à l'exploitation de ces structures stables, cette recherche est donc au cœur de la δ-arithmétique.

L'ensemble des données initiales ainsi que l'ensemble des solutions possibles constituent la donnée initiale des δ-problèmes. Il faut à la fois trouver les données et la solution P qui va avec. Leur résolution consiste à mettre face à face le problème et les solutions et de les liés via une structure stable, puis de rechercher sur cette structure le point qui permet d'ajuster de façon optimale les données et les solutions.

Ce chapitre propose une introduction à la δ-arithmétique, elle fournit les outils de base qui nous permettrons de comprendre et de manipuler avec aisance et rigueur cette flexibilité si cher à l'ingénieur.

Nous commençons - à travers la notion δ-voisinage - par une formalisation de la notion d'incertitude et de possibilité de substitution des données, ce qui nous permet de construire l'ensemble δ-Z dans lequel se font les opérations de δ-arithmétique. Nous abordons ensuite les questions de divisibilité dans δ-Z et reformulons le théorème de division pour l'adapter au problème traité en δ-arithmétique. Le chapitre se termine par une introduction du concept de diviseur idéal qui est une sorte de pont entre la δ-arithmétique et l'arithmétique classique.

NOTION D'INCERTITUDE SUR LES DONNEES

Nous allons ici commencer notre construction de la δ-arithmétique, par une réinscription des nombres dans leur δ-voisinage. Ces δ-voisinages sont en effet les éléments atomiques sur lesquels reposera la construction de la δ-arithmétique. L'objectif du voisinage est de sortir les nombres de leur isolement et de les inscrire dans un ensemble englobant les éléments qui lui sont substituables.

La notion de δ-voisinage, essaie de formaliser les concepts d'incertitude et de flexibilité des données utilisées pour résoudre un problème donné. Cette nécessité de traiter les nombres en termes de δ-voisinage se justifie dans les cas où les données sont soumises à l'imprécision des appareils de mesure utilisés pour les déterminés, à l'introduction volontaire d'erreur pour les brouiller, mais surtout à la nécessité que les quantités résultant possèdent un certain nombre de propriétés.

A travers la notion de δ-voisinage, la δ-arithmétique nous propose de traiter une valeur x, comme un simple représentant de l'ensemble des valeurs possibles qui sont situés dans son voisinage immédiat. la taille de ce voisinage dépendant de la confiance qu'on a dans le système nous ayant permis d'arriver à cette mesure x (notre instrument de mesure par exemple) ou de notre seuil de tolérance à l'erreur (si on veut remplacer x par une valeur qui est mieux adaptée à la résolution du problème sur lequel on travaille). On dira alors de tous les points de ce voisinage de x, qu'ils sont presque égaux à x dans le sens ou dans les conditions du problème qu'on souhaite résoudre, ils peuvent être substitués à x sans biaiser totalement les solutions finales.

Pour quantifier et traiter cette notion d''incertitude sur les données nous introduisons 3 concepts : celui de la « presque égalité » qui nous dit quand est ce que un nombre est substituable à un autre, celui du « voisinage » qui nous donne pour une valeur x observé l'ensemble des valeurs qui lui sont substituable, et enfin la « relation d'ordre dans le voisinage » qui nous dit entre deux valeurs substituable à x, laquelle est la plus adapté étant donné le problème qu'on essaie de résoudre.

PRESQUE EGALITE ET VOISINAGE

Supposons que nous observions une valeur x sur laquelle pèse une incertitude d'ampleur δ (inadéquation de l'instrument de mesure, seuil de tolérance à l'erreur, cryptage….). Alors la valeur réelle que cache la valeur observée x peut être une quelconque valeur y qui lui est « suffisamment proche »… on dira alors que la valeur y est presque égale à la valeur x, dans le sens ou plutôt que traiter la valeur x observée, nous pouvons lui substituer la valeur y sans modifier de façon inacceptable (pour le problème traité) les solutions finales. En effet nous pouvons substituer à l'observation x toutes valeurs qui lui est suffisamment proche, d'où les définitions suivantes.

Def 1 (presque égalité dans Z) :
Soit $(x, y) \in Z * Z$ ***on dit que y est presque égal à x et on note*** $y =_\delta x$ ***si*** $\exists \delta' \in Q, | x = y + \delta' * x$ ***et*** $|\delta'| < |\delta|$. ***avec*** $|\delta| \leq \frac{1}{2}$, en d'autre terme, ***y est δ-égal à x.***
<=> y δ-égal à x ssi $y \in [x - \delta * x, x + \delta * x]$ *ie* $\exists e \in [-\delta * x, \delta * x] | x = y + e$. e est **l'erreur d'approximation** de x par y et $\xi = \frac{e}{x}$ est le taux d'erreur d'approximation de y par x.

- ➔ ***Dire que y est δ-égal à x, c'est traduire le fait que lorsqu'on est prêt à tolérer un taux d'erreur δ on peut remplacer y par x sans altérer de façon inacceptable les résultats de nos calcul.***
- ➔ ***Les points δ-égaux à x, sont ceux situés à une distance de x inférieur à δx de x.***

L'ensemble $[x]_\delta = [x - \delta x, x + \delta x]$ ***qui regroupe tous les points δ-égaux à x est alors appelé δ-voisinage de x.***

- ➔ Le ***δ-voisinage de x regroupe donc tous les points qui lui sont δ-égaux, et donc lui sont substituables.***
- ➔ Les ***δ-voisinage sont en quelque sorte les atomes de la δ-arithmétique. Ici plutôt que de raisonner en termes de points individuels on pensera en permanence en termes de voisinage.***

Aussi, nous pouvons étendre l'ensemble Z dont on traite en arithmétique classique à un δ-Z regroupant tous les voisinages des points de Z. $\delta - Z = \{ [x]_\delta \; x \in Z \}$.

Au concept d'égalité exigeant en l'arithmétique classique l'identité parfaite des objets comparés, nous avons substitué celui de δ-égalité, se contentant d'une substituabilité à conséquence acceptable.
Au point individuel de l'arithmétique classique, nous avons adjoint les valeurs proches pour raisonner en termes de δ-voisinage.

La δ-arithmétique ne s'intéresse donc plus au traitement de points isolés de Z, mais envisage chaque point comme une bulle contenant l'ensemble des points qui lui sont proche et s'autorise donc à remplacer l'observation x par n'importe lequel des points contenu dans la bulle dont il est le centre, l'ensemble de ces bulle est l'ensemble δ-Z sur lequel porte la suite de cette étude.

RELATION D'ORDRE DANS LE VOISINAGE D'UN POINT

Dans le paragraphe précédent nous avons introduit la notion de voisinage d'un point x, délimitant l'ensemble des valeurs substituables à x, cet ensemble peut contenir un nombre plus ou moins important de valeur selon que x est grand et/ou que δ est élevé. La question qui se pose alors est celle du critère de choix, qui nous permettra de décider entre deux valeurs laquelle est la plus « adapté » au remplacement de x.

Nous introduisons donc pour résoudre ce problème de priorité dans le remplacement de x, une relation d'ordre dans son voisinage. Cette relation régit la succession de x par ses voisins, en stipulant premièrement que x est le mieux placé pour se remplacer et deuxièmement, que parmi les éléments de son voisinage, ceux qui lui sont le plus proches sont prioritaires pour sa succession.

Notons cependant qu'il est tout à fait possible d'envisager d'autre type de relation d'ordre en fonction du problème que l'on souhaite résoudre, celle formulé ici a été choisi en vue du calcul du PGCD parce que nous souhaitons une modification aussi faible que possible des données initiales.

Def 2 (relation d'ordre) :
Soit x un entier et a, b deux entiers δ-égaux à x (i.e. $x = a + e$ et $x = b = e'$ avec $|e| <= \delta x$ et $|e'| <= \delta x$). On dit que a est plus proche de x que b et on note $a <_x b$ ssi $|e| < |e'|$. En d'autre terme a est une meilleurs estimation de x que b.

- ***Nous établissons ainsi que plus on est proche du centre, plus haut on est placé dans l'ordre de succession.***
- ***Notons aussi que ce critère maintien x comme meilleures approximation de x (***δ=0).

Propriétés

Nous allons maintenant étudiés ce que deviennent les propriétés des opérations arithmétique classique dans ce contexte. Signalons au passage que la définition de ces opérations (adition, soustraction, multiplication, division) n'est pas modifiée, c'est l'interprétation des résultats qui change.

P1 : $Card([x]_\delta) = 2\delta x <= x \rightarrow x < y -> card([x]_\delta) < card([y]_\delta)$

- *Plus un point est grand, plus il a de voisins.*
- *Le nombre maximal de voisins d'un point est égal à sa valeur (car* δ < ½).

P2: $\delta < \delta' -> card([x]_\delta) < card([x]_{\delta'})$

- *Plus la tolérance à l'erreur est importante plus un point à de voisins.*

P3: $a =_\delta x$ *et* $\delta' > \delta$ *alors* $a =_{\delta'} x$.

- *On ne perd pas de voisins en augmentant la tolérance à l'erreur, de nouveaux voisins viennent simplement s'ajouter aux précédents. En d'autre terme si on augmente le seuil de tolérance à l'erreur alors les approximations précédentes de x restent bonnes. Evidement si on réduit ce seuil, il peut arriver qu'une estimation précédemment acceptable ne le soit plus.*

P4 : $a =_\delta b$ et $b =_\delta c$ n'implique pas $a =_\delta c$: la Presque égalité contrairement à l'égalité n'est pas transitive.

➔ *Eh oui, les voisins de mes voisins ne sont pas tous mes voisins, sinon tout le monde serait voisin de tout le monde, ce qui n'est pas vrais dans la réalité, alors ce n'est pas vrai en δ-arithmétique non plus.*

P5: $a =_\delta x$ n'entraine pas forcément $x =_\delta a$. La relation n'est pas non plus réflexive.

C'est aussi une conséquence directe de P1, en effet si x < y alors le voisinage de y contient plus de point que celui de x, et on pourra donc y trouver des points qui ne sont pas dans celui de x.

C'est donc un abus de langage que de dire que deux nombres sont δ-égaux, il y'en a en réalité qu'un qui est δ-égal à l'autre.

CL : Nous avons à travers la notion de « presque égalité » établi sous qu'elles conditions une valeur y était substituable à une valeur observée x. la notion de voisinage nous a ensuite permis de réunir dans un ensemble, les valeurs pouvant correspondre à l' observation d'une valeur x. ceci nous a permis de définir l'objet d'étude de la δ-arithmétique comme l'ensemble des voisinages des points de Z plutôt que de nous restreindre à des points isolés comme c'est le cas en arithmétique classique. Nous avons ensuite établi une relation d'ordre nous permettant de choisir entre deux éléments du voisinage d'un point x, lequel était le mieux adapté au remplacement de x.

Pour finir nous avons établi quelque propriétés des voisinages, notamment que le nombre d'élément présent dans le voisinage d'un point x était proportionnel à x et mais aussi au seuil de tolérance à l'erreur et que la relation de presque égalité n'était ni transitive ni réflexive.

VOISINAGE ET RELATION D'ORDRE DANS LE VOISINAGE D'UN ENSEMBLE

Nous commençons par une généralisation de la notion de voisinage, celui-ci dans le cas d'un ensemble réunira toutes les combinaisons de points qu'on peut former en prenant un point du voisinage de chacun des éléments de l'ensemble.

Def 2.1 (presque égalité d'ensemble):
Soient $X := \{x_0, x_1, x_2 \dots . x_n\}$ et $X' := \{x'_0, x'_1, x'_2 \dots . x'_n\}$ deux ensemble de même cardinal on dit que l'ensemble X' est δ-égal à X et on note X'= δ X ssi $\forall (x_i, x'_i) \in (X, X')$ on a $x'_i =_\delta x_i$

Le premier constat que l'on fait une fois définit le concept de voisinage d'ensemble est que l'ensemble à beaucoup plus de voisins que les points de plus chacune des combinaisons s'ajuste à sa manière par rapport à l'ensemble initiale. C'est un quasi chao, et mettre de l'ordre autour d'un ensemble est une tache bien ardue. De nombreux critères peuvent être retenus pour définir la proximité d'un ensemble par rapport au centre l'important est que quel que soit le critère retenu l'ensemble initial préfère toujours autant que possible garder sa place, et s'il est contraint de la céder il doit la

céder à un ensemble qui joue au mieux le rôle que lui jouait. La relation d'ordre à choisir dépend alors du problème que l'on essaie de résoudre.

Def 2.2 (relation d'ordre):
Soient $X := \{x_0, x_1, \dots . x_n\}$ *et* $X' := \{x'_0, x'_1, \dots . x'_n\}$, $X'' := \{x''_0, x''_1, \dots . x''_n\}$ *deux ensembles δ-égaux X'. On dit que X' est plus proche de X que X" et on note* $X' <_X X''$ *ssi* $\exists \delta' < \delta$ *tel que* $X' =_{\delta'} X$ *et* $X'' \neq_{\delta''} X$ *en d'autre terme en réduisant la tolérance à l'erreur, arrivera un moment ou X" ne pourra plus être considéré comme une approximation de X alors que X' le restera.*

➔ La plus importante erreur d'estimation de X par un point de X'' est supérieur à la plus importante erreur d'estimation par les point de X'.

Le choix de la relation d'ordre dépend du problème que l'on souhaite résoudre. Le choix d'une relation d'ordre a pour but de permettre que l'ensemble X' qui est le plus proche de X soit aussi le mieux adapté à la résolution du problème qui nous intéresse. On note $E := X - X' := \{e_i = x_i - x'_i\}$ l'ensemble des erreurs qu'induiraient le remplacement de X par X'. considérant l'ensemble E, nous voulons mesurer la distance entre X et X', la mesure ξ(X,X') qui mesure cette distance doit alors être construite de façon à ce qu'un élément soit d'autant plus adapté au remplacement de X que cette distance est faible.

Def 2.4 (relation d'ordre 3) :
Une relation d'ordre sur un voisinage de X est donné par $X' < X''$ *ssi* $E(X') < E(X'')$
Avec $X' = P'Q'$ *et* $E(X) = f(q_0)E'(X)$ *ou* $E'(X)$ *est donné par l'une ou l'autre des fonctions d'erreur précédentes, et f es une fonction décroissante en* q_0, q_0, *désignant le plus petit élément de Q (son ordre).*

Ainsi en choisissant une fonction à décroissance plus ou moins rapide (par exemple $f(a) = \frac{1}{a}, ou\ f(a) = \sqrt{a}$ nous pouvons ajuster la recherche de PGCD en marquant notre préférence pour la minimisation de l'erreur ou pour l'ordre de grandeur, plus f sera décroissance, plus on exigera d'une structure d'ordre inférieur d'améliorer de façon importante le seuil d'erreur d'une structure s'ordre supérieure pour que l'approximation générée soit meilleurs que la précédente.

➔ cette troisième famille de relations est surtout utile pour les problématiques liées au calcul du PGCD elle intègre en plus de la proximité tel que définit précédemment, l'ordre de grandeur du générateur de l'ensemble ; nous verrons son intérêt un peu plus tard.

3.1 MINIMISATION DE L'ERREUR MAXIMALE.

Un bon indicateur de la proximité de deux ensemble est la distance qui sépare leur points les plus éloignés, dans cette première approche du classement des éléments du voisinage de X c'est sur cet élément que nous allons concentrer notre évaluation des voisins de X.

Def 2.5(l'individuel) :
Une relation d'ordre sur un voisinage de X est donné par $X' < X''$ ssi $f(X') < f(X'')$
Avec $X' = P'Q'$ et $f(X') = Max\left(\left|1 - \frac{x'_i}{x_i}\right|\right) i = 1..n$

Cette première approche de la construction de ξ peut est construite autour de la minimisation de la plus importante des erreurs d'approximation des éléments de X par ceux de X'.

Soit $X = \{x_0, x_1, \dots, x_n\}$ et $X' = \{x'_0, x'_1, \dots, x'_n\}$ on défit ξ (X, X') par :

$\xi^I(X, X') = max\left\{\delta_i = \frac{|x_i - x'_i|}{x_i}\right\}$ ***Ξ (X, X') est donc le maximum des taux d'erreur individuel des éléments de X'.***

Le contrôle de ξ^I, permet à l'utilisateur de maitriser la moins bonne approximation et donc par conséquent les autres qui sont par définition meilleurs.

3.2 MINIMISATION DE L'ERREUR MOYENNE

La mesure précédente accorde un poids excessif au élément de X les moins bien repris dans l'approximation X', aussi pourra-t-on voir des cas ou pour rapprocher même de façon très marginale un élément de X' de sa valeur réel dans X, on introduira une erreur potentiellement importante sur l'ensemble des autres éléments. Aussi pour repartir de façon plus équitable les erreurs d'approximation et redonner plus de poids au consensus par rapport aux éléments isolé, on définit :

Def 2.5(la moyenne) :
Une relation d'ordre sur un voisinage de X est donné par $X' < X''$ ssi $f(X') < f(X'')$
Avec $X' = P'Q'$ et $f(X') = \sum_{i=0}^{n} \left|1 - \frac{x'_i}{x_i}\right|$

Avec cette approche, plus il y a d'élément de X' qui sont proches de X (la mesure de cette proximité prenant en compte la distance individuel à l'élément approximé) plus X' est proche de X.

Une mesure de distance à X au sein de son voisinage est alors fourni par :

$\xi^M(X, X') = \frac{\sum_{i=0}^{n} \delta_i}{n}$ **avec** $\delta_i = \frac{|x_i - x'_i|}{x_i} = \left|1 - \frac{x'_i}{x_i}\right|$ **C'est la moyenne des taux d'erreur.**

Avec la mesure ξ^M, contrairement à la mesure ξ^I précédente, ce n'est plus la maitrise de l'erreur individuelle qui est privilégié mais celle de l'erreur moyenne, on essaie de répliquer au mieux l'ensemble des éléments de X.

3.3 MINIMISATION DU NIVEAU D'ERREUR GLOBAL

On considéré au voisinage de X, la relation d'ordre donné par :
Def 2.5(le Globale) :

Une relation d'ordre sur un voisinage de X est donné par $X' < X''$ *ssi* $f(X') < f(X'')$
Avec $X' = P'Q'$ *et* $f(X) = \sum_{i=0}^{n} |x_i|$

Celle-ci stipule en quelque sorte que X' est d'autant plus proche de X que la somme de ses élément est proche de la somme des éléments de X. cette relation convient au traitement de problème ou il est nécessaire de conserver aussi exact que possible la taille total des données qu'on manipule (c'est souvent le cas en finance par exemple)

Lorsque maintien de la taille totale de l'ensemble X peut être plus important que les enjeux d'approximation individuelle, i.e de maintenir à un niveau minimal la différence entre la somme des éléments de X, $\sum_{i=0}^{n} x_i$ et la somme de leur approximation $\sum_{i=0}^{n} x_i'$, la mesure d'erreur suivante peut être mieux adapté pour la mesure de la distance à X de ses voisins.

On définit ξ *par :* $\xi^G(X, X') = \frac{|\sum_{i=0}^{n} x_i - \sum_{i=0}^{n} x_i'|}{\sum_{i=0}^{n} x_i}$

Contrairement aux deux approches précédentes, cette mesure ignore totalement les approximations individuelles pour se concentrer sur la conservation globale de X.

3.4 MINIMISATION SIMULTANEE DES NIVEAU D'ERREUR GLOBAUX ET INDIVIDUEL

Pour prendre en compte les contraintes qui pèses à la fois sur la maitrise nécessaire de la qualité d'approximation des individuels des éléments de X et la nécessité de conservation globale de la taille de X, nous proposons une mesure d'erreur combinant les deux type d'erreur précédente nous proposons la définition suivante de ξ.

$$\xi^{IG}(X, X') = \xi^G(X, X') + \xi^I(X, X')$$

RQ :

- Une alternative aurait pu être de définir ξ comme $\xi(X, X') = \xi^G(X, X') * \xi^I(X, X')$ le problème d'une telle définition est qu'une très bonne approximation d'un des aspects (globaux ou individuels) peut masquer totalement une grosse erreur sur le second aspect. Nous avons pu vérifier très rapidement dans nos tests qu'il était très facile moyennant de grosse erreur d'approximation individuelle d'obtenir des approximations globales quasi exempte d'erreur, ce qui ne correspond absolument pas à ce que nous recherchons.
- Par ailleurs plutôt que de minimiser directement la somme $\xi^{IG}(X, X') = \xi^G(X, X') + \xi^I(X, X')$ on peut associé un poids à chacune des erreur prise en compte pour marquer le fait qu'on est par exemple plus sensible à l'erreur globale qu'à l'erreur individuel, on définit alors le niveau d'erreur à minimiser par somme $\xi^{IG}(X, X') = \beta * \xi^G(X, X') + \xi^I(X, X')$ ou β est le poids de l'erreur globale : par exemple dire que $\beta = 10$ c'est dire que dans l'erreur finale, la part associé au niveau d'erreur globale doit être égale au plus au 10ieme de l'apport de l'erreur individuel. Dans cette configuration on essaie bien sûr de minimiser

ξ^{IG} mais dans la détermination du générateur idéal on fait porter β plus d'erreur sur le niveau individuel que sur le niveau globale.

- Une autre approche pourrait être de combiner erreur moyenne et erreur globale, ou erreur moyenne et individuel.

PROBLEME EN δ-ARITHMETIQUE

Problème d'arithmétique classique

On considère un problème d'arithmétique Pb consistant à partir d'une donnée X (X, entier, ou ensemble d'entiers) à trouver dans un ensemble S une quantité Ps (solution du problème) ayant avec X une relation Pr.

Notons que l'ensemble des solutions n'est pas toujours une donnée du problème, mais le plus souvent il se déduit aisément de celui-ci, alors allons-nous le considérer quasiment comme une donnée.

Par ailleurs la formulation classique de certain problème peu différé quelque peu de celle adopté ici, le choix de la formulation ici est fait de façon à dégager une reformulation en δ-arithmétique qui en est proche.

EXP :

Le problème de la factorisation peut être formulé ainsi :

Soit X un entier non premier, trouver dans l'ensemble $S =]1, \sqrt{X}]$, un entier P_s, tel que Ps soit un facteur de X (i.e. qu'il existe Q entier tel que $X = Q*P_s$)

Le problème de la recherche de diviseur commun peut se formuler ainsi :

Soit X un ensemble fini d'entiers et x0 sont plus petit, trouver dans l'ensemble $S =]2, x_0]$ un entier Ps tel que Ps soit un diviseur commun aux éléments de X (i.e. $\forall xi \in X, Ps|xi$).

Le δ-problème

Les problèmes auxquels on est confronté en δ-arithmétique sont quasiment les même que ceux présentés précédemment, à ceci près que dans leur résolutions, on s'autorise un léger ajustement sur X afin d'obtenir l'élément Ps vérifiant avec X la relation Pr.

Soit X un entier (ou un ensemble d'entiers), on considère le problème Pb consistant à rechercher dans un ensemble S un entier Ps, vérifiant avec X la relation Pr. Toutefois, étant donné l'usage que nous aurons de Ps, on s'autorise pour obtenir un léger ajustement de X le transformant en X' à condition de maintenir la distance entre X et X' inférieur à un seuil δ, i.e. $\xi(X-X') < \delta$ ou δ est un seuil fixé en fonction de notre tolérance à l'ajustement des données et ξ une pseudo distance garantissant que les quantité X' les

plus proche de X soient les mieux adaptées pour remplacer X dans la résolution des problèmes qui nous intéressent.

Le δ-problème peut alors se reformuler ainsi, étant donné un entier (ou ensemble d'entiers) X, trouver dans un ensemble S, un entier Ps vérifiant la relation Pr avec un entier X' appartenant au δ-voisinage de X.

La seule différence entre le problème et le δ-problème réside donc dans le fait que ce dernier ne restreint pas la relation Pr que Ps doit entretenir avec X au seul élément X, mais l'ouvre à l'ensemble de ses δ-voisins. Ainsi si une solution est trouvée pour un δ-voisin X' de X, la suite du traitement se fera en remplaçant X par X' et comme un ajustement de δ avait été jugé d'ampleur raisonnable pour ne pas affecter de façon importante nos résultats, le remplacement de X par X' sera sans grande incidence dans la résolution de notre problème initiale.

La résolution d'un δ-problème exige donc que soit fixé de façon préalable le niveau maximal d'ajustement tolérable δ, ainsi qu'une pseudo mesure de distance ξ fixant l'ordre de priorité du remplacement de X par les éléments de son voisinage.

Notons par ailleurs que le problème d'arithmétique classique n'est rien d'autre qu'un δ-problème ou on a réduit δ à 0, et comme nous le verrons il peut souvent être plus efficace de résoudre un δ-problème avec $\delta = 0$ que de résoudre le problème d'arithmétique classique.

EXP :

δ-problème de recherche du plus grand facteur commun

Étant donné un entier X, trouver dans l'ensemble $S = [1, \sqrt{X}]$ l'élément Ps tel que Ps soit un facteur d'un δ-voisin de X et tel que Ps est « supérieur » à tout autre élément vérifiant cette propriété (i.e. Ps le plus proche possible de $\sqrt{X}$).

δ-problème de recherche du plus grand diviseur commun

Étant donné un ensemble d'entiers X et x_0 son plus petit élément, trouver dans l'ensemble $S = [1, x0]$ l'élément Ps tel que Ps soit un diviseur commun à l'ensemble des éléments d'un δ-voisin X' de X et tel que Ps est « supérieur » à tout autre élément vérifiant cette propriété.

Le δ-problème est souvent requalifier d'approximatif, ainsi le δ-problème de la recherche de PGCD est appelé recherche de PGCD approximatif et celui de la factorisation, factorisation approximative.

EXPLORATION DE S ET RESOLUTION DE δ-PROBLEME

Notre approche de la résolution des δ-problème est simple, nous partons d'un algorithme trivial (naïf) permettant de résoudre le problème Pb et nous en améliorons

progressivement la complexité temporelle jusqu'à obtenir un cout raisonnable voir meilleurs que ceux traitant du problème plus simple d'arithmétique classique.

Toute la subtilité de cette approche repose sur un art de la navigation entre la manipulation des données réel (en tant que nombres réels) et le passage aux entiers (tel que l'exige l'arithmétique).

APPROCHE NAIVE

Soit $[X]_\delta$ l'ensemble des données initiales utilisable pour résoudre le problème Pb, et soit S l'ensemble des solutions potentielles du problème.

L'approche naïve de la résolution du problème Pb consiste en une itération simple de l'ensemble des éléments S et en une vérification du respect ou non des propriétés Pr par l'élément de S.

Pour la résolution du δ-problème l'approche naïve exige un petit effort supplémentaire, consistant à construire une méthode M associant à un élément P de S l'élément X' le plus proche de X pour la distance ξ et tel que P vérifie avec X' la propriété Pr, P est alors une solution du δ-problème ssi X' est un δ-voisin de X, et P sera une solution du problème si X' est égal à X.

La méthode M est généralement très simple elle prend en paramètre la donnée X et la solution potentielle P et associe au couple une donnée X' : M(P,X) = X' et X' e [] -> P solution du δ-problème, et X'=X -> P solution du problème.

Exp :

Dans le problème de la factorisation approximative, M(P,X) : = M_x(P)= X' := P*[X/P] ceci traduit le fait que l'élément le plus proche de X dont P est un facteur donné par P*[X/P].

APPROCHE PAR EXPLORATION DES STRUCTURES STABLES

Le cœur de la δ-arithmétique réside dans la recherche et l'exploitation des objets appelée structures stables.

Les structures stables sont tout simplement des entités qui, lorsqu'on évalue un par un les éléments de l'ensemble des solutions possibles à un problème, varient moins rapidement que ces solutions. Une structure stable est donc une sorte d'invariant reliant entre eux un sous ensemble de solutions potentielles à un problème donné, et suffisant pour reconstruire l'approximation correspondante des données initiales à partir d'une solution correspondante.

Considérons notre problème Pb, et S l'ensemble des solutions possibles de Pb. On dit que Q est une structure stable compatible avec X ssi il existe un e sous ensemble S_Q de S tel que pour tout P ∈ S_Q, M(P,X) :=X' = f(P,Q) i.e. l'approximation X' induite par P se déduit

du couple (P,Q) par application d'une méthode f, et non plus de (P,X) et ceci pour tous les éléments du sous ensemble S_Q.

L'intérêt de ses structures est que lorsque l'on restreint notre recherche de solution à Pb dans le sous ensemble SQ, il est possible de trouver directement la solution P qui offre la meilleur approximation de X en résolvant f(P,Q) = X (en général on résout alors le problème dans R et on arrondi pour trouver la meilleure solution entière) la solution à ce problème et appelé générateur idéal de Q et est la meilleurs réponse que l'on puisse donné au problème si on se restreint au sous ensemble S_Q.

EXP : recherche de grands diviseurs

Si on explore la suite des presque- diviseurs d'un entier X : $Xn = \frac{X}{n} \mid n = X, \dots, \sqrt{X}$, on constate que deux diviseurs proche partagent le même quotient i.e. $P \approx P' \rightarrow \left[\frac{X}{P}\right] = \left[\frac{X}{P'}\right] := Q$. et pour un quotient Q fixé, si P est un diviseur associé à ce quotient, alors l'approximation de X induit par P est donné par X' = P*Q. Q est donc une structure stable pour la division approximative de X.

Cette structure est utilisée plus en détail dans le premier chapitre de la première partie de cette thèse.

EXP 2 : structure stable pour la factorisation.

Soit X un entier, et P et P' deux facteurs potentiel de X tous les deux proche de la racine de X, si P≈P' $alors\ k = P - \left[\frac{X}{P}\right] = P' - \left[\frac{X}{P'}\right]$, et pour k fixé, la meilleur approximation de X induite par un facteur d'ordre k est donné par : $X' = (R - a - k)(R + a + k)\ ou\ R = \left[\sqrt{X}\right]\ et\ a = (R - P)$. k est donc une structure stable pour la factorisation approximative de X.

Nous étudierons cette structure dans le premier chapitre de la seconde partie de cette thèse.

INTERET DES STRUCTURES STABLES

L'arithmétique est le monde des entiers, il permet d'apporter des réponses à des problèmes exigeant la manipulation des quantités entières, or l'opération de division conduit directement à des données réels, pour satisfaire à la contrainte de travail sur Z, la division arithmétique transfert donc une partie de l'information dans un objet appelé erreur. Cette mise de côté d'une partie de l'information est le coup exigé par l'arithmétique pour maintenir son ensemble de travail, aussi exigera-t-elle pour validé une propriété que la quantité d'information ainsi mise de côté soit nul (la δ-arithmétique se contentera d'exiger qu'elle soit minimale). Cependant si nous analysons l'évolution de l'erreur et de la partie retenue, nous remarquerons qu'elles ne sont pas

aléatoires, une analyse minutieuse permet de mettre en évidence la continuité de leur distribution (lorsque l'on choisit le bon angle de vu). En effet si on déroule nos opérations suivant un certain ordre, on verra émerger une certaine continuité dans l'évolution de la partie de l'information retenue par l'arithmétique, c'est en quelque sorte cette trajectoire continue suivi par les informations que nous appelons structure stable. Tout l'enjeu de la résolution des problèmes d'arithmétique impliquant une opération de division, est de retrouver ces structures stables suivant lesquels l'information se partage de façon continue entre partie réellement utilisée et partie mise à l'écart dans le niveau d'erreur.

Le principal intérêt des structures stable est qu'il permettent de regrouper dans des sorte de classe d'équivalence des sous ensemble entier de solutions possible à un δ-problème, et surtout de traiter comme un tous ces sous-ensembles, en offrant la possibilité de trouver directement la meilleurs réponse P au problème Pb parmi les éléments du sous ensemble S_Q.

Par ailleurs les structures permettent de prendre en compte sans effort supplémentaire la nécessité d'explorer les voisinages données initiales pour rechercher la solution optimale. Solution finale et donnée initiales se trouvent regroupées en une seule information, la structure stable.

Aussi une approche de la résolution des δ-problèmes basé sur les structures stables, sera d'autant plus efficace que celle-ci couvrirons des sous ensemble de taille importantes.

Les algorithmes de résolution de δ-problème remplacerons donc l'itération sur l'ensemble S en une itération sur les structures stables, en construirons le générateur idéal et vérifierons si celui-ci réponds ou non aux exigences en terme de proximité de X et de son approximation induite X' (bien évidement si l'idéal ne répond pas à cette exigence, aucun autre générateur de la structure n'y répondra).

L'ordre d'une structure caractérise la position de ses générateurs dans le découpage de l'ensemble S des solutions potentielle au problème Pb. Cet ordre qualifie aussi chacun de ses générateurs et donc chacun des éléments du sous ensemble en question.

La couverture d'une structure Q qu'on note C(Q) est égale au pourcentage de l'ensemble de générateur représenté par l'ensemble de ses générateurs. $C(Q) = \frac{Card(Gen(Q))}{Card(S)}$.

La couverture d'ordre n est égale à la somme des couvertures des structures d'ordre inférieur ou égale à n. $C(1) = C(Q)$ ou Q est la structure d'ordre 1 ; $C(n) = C(n-1) + C(Qn)$ ou Qn est la structure d'ordre n.

- ➔ C(n) croit, mais de façon générale (pour les structures que nous avons étudiées) l'apport marginal de chaque structure d'ordre supérieur à la couverture décroit de façon quadratique.

PROCEDE DE RESOLUTION DE PROBLEME EN δ-ARITHMETIQUE

Construire l'ensemble des solutions possibles du problème. (On se restreint au minimum possible pour minimiser le coup de l'algorithme de base)

- ➔ Pour le PGCD c'est l'ensemble des entiers inférieur ou égale au plus petit des éléments de X.
- ➔ Pour la factorisation, c'est l'ensemble des entiers inférieur ou égale à la racine de x.

Clarifier l'ensemble des propriétés que doit vérifier l'élément recherché.

- ➔ Pour la factorisation : P doit diviser X.
- ➔ Pour la PGCD, P doit diviser chacun des éléments de X et être supérieur à toutes autres valeurs vérifiant cette propriété.

Notons que pour le delta problème, ces mêmes propriétés doivent être respecté par P, vis-à-vis d'un ensemble ou d'une donnée X' qui est suffisamment proche de X. la distance que l'on accepte entre X et X' dépends du problème que l'on souhaite résoudre à l'aide de P.

Construire la méthode M qui donne X' à partir de P.

- ➔ Pour la factorisation X' := M(P) := P*[X/P]
- ➔ Pour le PGCD de X, X'= {X'i} Xi' := M(Pi) := Pi*[Xi/Pi]

Construire la méthode C qui permettant de déduire de X et P, une structure stable Q.

- ➔ Non trivial, c'est la recherche de cette méthode qui est au cœur de la δ-arithmétique.

Etudier la structure et sa stabilité.

- ➔ Il est impératif que la structure stable ait un nombre important de générateurs…
- ➔ La fonction C induit un découpage de l'ensemble $\left[P_m, P_M\right]$ en sous intervalle de taille non nécessairement égale. $\left[P_m, P_M\right] = \left[P_{m0}, P_{M0}\right] U \left[P_{m1}, P_{M1}\right] U \dots U \left[P_{mn}, P_{Mn}\right]$, on en déduit donc une notion de niveau ou d'ordre de la structure en fonction de la position de l'intervalle de générateur qui permettent de le construire. Ainsi, la structure Q de niveau 1 sera générée par l'ensemble des Pi appartenant à l'intervalle [P_m1 , P_M1]
- ➔ La conséquence de ce regroupement des solutions potentiel du problème en intervalles caractérisés par leur niveau et le fait qu'ils partagent la même

structure, est que plutôt que parcourir chacun de élément de P , notre algorithme fera une itération sur les niveaux des structures et ira directement chercher le diviseur idéal.

Construire la méthode $A^{-1}_Q(X)$ permettant d'obtenir le générateur idéal de Q.

- ➔ Le générateur idéal $P = A^{-1}_Q(X)$ est celui qui permet d'obtenir l'approximation $X' = A(P,Q)$ tel que X' est le plus proche de X au sens du critère de proximité retenu.
- ➔ La construction de l'algorithme inverse A^{-1}_Q dépend du problème et peut présenter quelque subtilité que nous verrons pour chacun des problèmes qui nous intéresse.
- ➔ une fois l'approximation X' de X obtenu on vérifie qu'il respecte les critères de proximité cible, et on décide alors d'arrêter ou de poursuivre la recherche.

PLAN

Soit X l'ensemble dont on cherche le PGCD et x_0 sont plus petit élément. Le PGCD est un élément de l'intervalle $[1, x_0]$

Division Approximative	**Factorisation Approximative**	**Division Equivalente**
1	R(x0)	xo

Nous avons donc déterminée trois zone de recherche pour le PGCD, la première correspond au cas où le PGCD recherché est « très inférieur » à la racine du plus petit élément, le second au cas où celui-ci se situe autour de cette racine, et le dernier au cas où il est « très supérieur » à cette racine.

Notre présentation du sujet se divise en 4 Parties.

La première partie construit la notion de division approximative et s'appuie sur cette dernière nous développer un algorithme de recherche du pgcd lorsque celui-ci est relativement grand par rapport à la racine du plus petit élément de l'ensemble des données.

La seconde partie introduit quant à elle la notion de factorisation approximative et en déduit une seconde approche pour la recherche du pgcd approximatif. Cette approche est quant à elle d'autant plus efficace que le pgcd recherché est proche de la racine du plus petit élément.

La troisième partie se concentre sur le périmètre restant, i.e. aux cas ou le PGCD recherché est supérieur à la racine du plus petit élément. *(Recherche de structure stable en cours)*

La quatrième et dernière partie présente l'implémentation des algorithmes et en analyse la complexité.

Cette étude se répartie sur 6 Chapitres.

Les deux premiers chapitres sont des chapitres introductifs.

Le premier propose une présentation des problématiques qui nous ont amené étudier cette question du Pgcd approximatif, nous y proposons aussi une brève présentation des problèmes de calcul de PGCD approximatif et de factorisation qui seront traité dans cet article.

Le second chapitre propose une introduction à la δ-arithmétique qui constitue le framework que nous utiliserons pour résoudre les problèmes présentés au chapitre précédent. Ce chapitre commence par une formalisation de la notion d'incertitude sur laquelle est construite la δ-arithmétique, en particulier nous introduisons les notions de voisinage, de δ-voisinage et de relation d'ordre dans un voisinage. Nous y présentons ensuite la démarche de résolution des problèmes en δ-arithmétique, cette démarche s'articule autour de la recherche et de l'exploitation des structures stables. On trouve donc dans ce chapitre les principales notions de δ-arithmétique qui serons ensuite illustrées et exploitées dans les chapitres suivant pour résoudre les problèmes qui nous occupent dans cette thèse.

La Première partie de la thèse qui traite de la recherche su PGCD approximatif lorsqu'il est plutôt grand comparé à la racine du plus petit élément de l'ensemble X. cette partie comporte 3 chapitres.

Le premier chapitre introduit la notion de δ-divisibilité qui étend la notion de divisibilité de l'arithmétique classique. Nous essayons ici d'envisager ici non plus la divisibilité de x par un entier P mais de traiter simultanément x et son δ-voisinage. Nous y proposons une méthode de division qui, contrairement à l'approche euclidienne ou on se fixe un diviseur et lui associe un couple quotient-reste unique, nous fixons ici un quotient et essayons de déterminer l'ensemble des couple diviseur-reste qui lui sont attaché. Cette division dite inversée est comme nous le verrons mieux adapté à la résolution de problème en δ-arithmétique, car elle nous permet de mettre en évidence une structure stable et celle-ci est l'édifice sur lequel se construit tout l'édifice de la δ- arithmétique.

Dans le second chapitre nous nous penchons sur le cas des sous-ensembles finis de δ-Z et abordons la question des diviseurs communs. Nous commençons par généraliser au cas des sous-ensembles les notions introduites au chapitre précédent. Cette étude nous

mènera naturellement des diviseurs communs dans δ-Z et à la définition des Pgcd dans δ-Z (ceux-ci ne sont rien d'autre que les pgcd approximatifs qui nous intéressent).

Le troisième chapitre et dernier chapitre, nous analysons en profondeur la question du pgcd en s'appuyant sur les résultats des deux chapitres précédents. Nous en déduisons une méthode de construction du pgcd…

La seconde partie s'intéresse à la recherche du pgcd autour de la racine du pivot. Et se décompose en deux chapitres.

Le quatrième chapitre et premier chapitre de cette seconde partie traite des questions de factorisation et de factorisation approximative. Nous y développons et analysons une nouvelle méthode de factorisation, celle-ci nous permet dans le cadre de la factorisation approximative d'exhiber une nouvelles structure stable à partir de laquelle nous pourrons refaire de la δ-arithmétique.

Le cinquième chapitre s'appuie sur la notion de factorisation approximative introduite au chapitre précédent, construit les concepts de δ-facteur commun et en déduit une nouvelle approche de la recherche du Pgcd approximatif.

La troisième partie de cette thèse s'intéresse à la question du pgcd approximatif lorsque celui-ci est très petit par rapport à la racine du pivot.

La dernière partie se penche sur les questions pratiques d'implémentation de cet algorithme et en analyse le cout. Une version développée en C++ afin de bien mettre en évidence les différentes notions, est disponible en annexe.

PARTIE1

DIVISION APPROXIMATIVE ET PGCD APPROXIMATIF

Chapitre 2

DIVISIBILITE ET DIVISION EN δ-ARITHMETIQUE

Contrairement aux autres opérations dans l'ensemble Z des entiers, l'opération de division, ne conduit pas toujours à un élément de Z, mais le plus souvent à un réel, d'où la nécessité de se restreindre à des solutions approximatives, lorsque l'on travaille sur des données entières et qu'une opération de division ou une inversion de multiplication est impliquée dans notre processus.

Nous nous intéressons ici au δ-problème de la division, il peut se formuler ainsi, étant donné un entier X, trouvé dans intervalle donné, un entier P divisant un δ-voisin de X, aussi proche que possible de celui-ci.

En d'autre terme, supposons que l'on dispose d'une donnée X, et que nous devons résoudre un problème Pb, impliquant un entier P auquel on impose les propriétés, P dois diviser X et P doit être aussi grand que possible et compris entre deux valeurs minimale et maximale a et b (on peut exprimer [a, b] = [n√X, m√X] 1< n < m < √X), le δ-problème consiste alors à choisir dans [a, b] la valeur P divisant un voisin X' de X, X' devant être aussi proche que possible de X. pour résoudre le problème initiale, on aura alors plus qu'à remplacer X par X' et à utiliser le diviseur P.

Cette question du choix de P et du δ-voisin X' de X associé, pose la question de la répartition des erreurs d'approximation de X autour de ses diviseurs potentielles, c'est cette répartition des erreurs et le choix du diviseur idéal minimisant l'erreur dans une zone donnée qui nous intéresse dans ce chapitre.

Rappelons que le traitement des δ-problème passe par l'identification et l'exploitation des structures stables, la question est donc la suivante : quel sont les invariants lorsqu'on explore l'ensemble des diviseurs potentiel d'un entier x. la réponse comme nous le verrons dans ce cas est plutôt simple, ce sont les quotients.

Pour le comprendre, Considérons un entier X. l'ensemble de ses diviseurs potentiels est l'intervalle [1, X].

On considère la suite des rapports de X à ses diviseurs (potentiel de X : δ-diviseur en effet) de X :

$X_n = \frac{X}{n}, n = X$ … 1. : le δ-diviseur n de X induit l'approximation $X' = n * [X_n]$

La suite des erreurs d'approximation est donné par :

$E_n = X - X' = X - n * [X_n] = n *]X_n[$ ou [X] est la partie entière de X (on considèrera souvent l'arrondi à l'entier le plus proche) et]X[sa partie décimal (le résidu si on à considérer l'arrondi)

RQ 1 : $quand\ \ n : X \to \sqrt{X}\ , [X_n] : 1 \to \sqrt{X}$ La quantité [Xn] est donc une structure stable pour les diviseurs de X compris entre X et √X, en effet pour les (X - √X) diviseur de cet intervalle il n'ay que √X quotient possible, soit en moyenne 1 quotient pour 1/(√X-1) diviseurs. C'est cette simple structure stable que nous allons étudier ici.

RQ2 : la suite $D_n = X_n - X_{n-1} = \frac{X}{n(n-1)}$ qui induit l'évolution du niveau d'erreur de l'approximation X' de X est croissante et $quand\ \ n : X \to \sqrt{X},\ \ ]X_n[: 0 \to 1$.

DIVISIBILITE DANS δ-Z

Nous allons maintenant formaliser la notion de δ-divisibilité correspondant à la divisibilité dans l'ensemble δ-Z. rappelons que contrairement à Z, les opérations n'engagent pas des éléments isolés mais leur voisinage tout entier. Aussi, pour vérifier une propriété de l'arithmétique classique, -étant donné la possibilité qui est offerte de remplacer un point par l'un quelconque de ses δ-voisins – il faut dans le cadre de la δ-arithmétique que la propriété en question soit vérifiée pour l'un quelconque de ses δ-voisins.

Soit X un entier, en arithmétique, on dit que P divise X ssi, il existe n entier tel que X = n*P. en δ-arithmétique, on se laisse la possibilité de remplacer X par n'importe lequel de ses δ-voisin, on peut donc déduire naturellement que P est δ-diviseur de X ssi P divise au moins un de ses δ-voisins.

Def 3(presque divisibilité) :
Soit $(x, d) \in Z * Z$ ***on dit que d est un presque-diviseur de x et on note*** $d|_\delta\ x$ ***ssi*** $\exists\ \delta \in R$ ***et*** $q \in N\ , | x = d * q + \delta * x$ ***avec*** $|\delta| \leq \frac{1}{2}$***, plus précisément, on dit que d est un δ-diviseur de x. (ou que x est δ-divisible par d).***
Dit autrement $d\ |_\delta\ x \leftrightarrow \exists x' \in [x]_\delta$ ***tel que*** $d|x'$ ***(i.e. d divise un élément du δ -voisinage de x)***

Si $\delta = 0$***, on dit que d divise x. (arithmétique classique)***

Rq : pour être δ-diviseur d'un nombre en δ-arithmétique il suffit d'être diviseur de l'un quelconque de ses voisins. Or nous avons vu que dans le voisinage tous les points ne se valaient pas, la relation d'ordre dans le voisinage va donc induire une relation d'ordre dans l'ensemble des diviseurs en fonction de la proximité du voisin que ce dernier

divisent. ***d est donc un meilleur diviseur de x que d' si jamais le voisin a qu'il divise est plus proche de x que le voisin b que divise x.***

Afin de mieux comprendre les concepts introduit intéressons-nous maintenant à leur propriétés.et à la façon dont ils diffèrent ou se rapprochent de celles de l'arithmétique classique (cas δ =0).

Il serait intéressant d'étudié les structures algébriques construit à partir de δ-Z et des concepts introduits jusqu'ici.

Propriétés :

P6 : $p \mid x \rightarrow p|_\delta x$ ***pour tout*** δ **On en déduit que** $x|_\delta x$.

➔ Un diviseur est aussi un δ-diviseur (quel que soit δ), x divisant x, x est donc un δ-diviseur de x.

P7a : $p|_\delta x$ ***et*** $p'|p \rightarrow p'|_\delta x$.

➔ Les diviseurs d'un δ-diviseur de x sont donc des δ-diviseurs de x.

P7b : $p|_\delta x$ ***et*** $p'|_\delta p$ $n'implique\ pas\ p'|_\delta x$ **mais** $p'|_{\delta\delta} x$ **(l'erreur se multiplie)** la δ-divisibilité n'est pas transitive contrairement à la divisibilité.

En d'autres termes un δ-diviseur d'un δ-diviseur de x n'est pas forcément un δ-diviseur de x.

➔ Il n'est donc pas possible de faire transiter les δ-diviseur dans une chaine de division comme c'était le cas avec les diviseurs. Pour faire transiter un δ-diviseur il faut s'assurer que le seuil δ reste tolérable pour le plus petit élément. Ainsi le diviseur strict (0-diviseur) eux peuvent continuer à se transmettre.

P8 : $p \mid x$ ***et*** $x' \in [x]_\delta \rightarrow p|_\delta x'$ *Si p est un diviseur de x alors p est un δ-diviseur des éléments de son δ-voisinage.*

➔ *Cette transitivité restreinte permet aux voisins de se refiler au moins leur vrais diviseurs.*

P9 : $p|_\delta x$ ***et*** $p|_\delta y \rightarrow p|_\delta (x+y)$ ***et*** $p|_\delta (x-y)$ plus généralement, pour des entiers u et v quelconque, on aura $p|_\delta (u*x+v*y)$

P10 :

Démonstration.

DIVISION DANS δ-Z

Nous avons établi la notion de divisibilité et en avons étudié les propriétés fondamentales. Or la notion de divisibilité est intrinsèquement liée à l'opération de division. Pour pouvoir établir la δ-divisibilité d'un nombre x par P il nous faut traiter la question de la δ-division de x par P i.e. envisager la division de chacun des éléments du voisinage de x par P, or comme nous l'avons vu le voisinage d'un point peut contenir un nombre de point assez élevé, ce qui rend l'opération difficile voire impossible. Nous

devons donc envisager autrement le processus de division, de manière à obtenir sans traiter de façon exhaustive les points du voisinage, les réponses aux questions concernant la divisibilité d'un point x par un point P. pour cela nous envisagerons la division de façon quelque peu différente de la façon dont elle est traitée en arithmétique classique.

Commençons par revenir sur le théorème de division de façon à prendre en compte le fait que le reste d'une division peut désormais être positive ou négative, ceci tien à la définition que nous avons établi du voisinage d'un point x et au fait qu'une approximation d'un point x peut lui être inférieur ou supérieur. Nous parlerons alors d'erreur d'approximation plutôt que de reste de division. Le théorème de division peut donc être reformulé de la façon suivante.

Théorème1 (théorème de division) :
Soit x une entier, pour tout entier p, il existe une unique décomposition de x sous la forme $x = pq + e$ *avec* $-\frac{p}{2} \leq e < \frac{p}{2}$.

P est le diviseur, q est le quotient, a = p*q est l'approximation de x et e l'erreur d'approximation de x par a, Δ = x/e est le taux d'erreur d'approximation de x par a.

Si Δ < δ alors a est un δ-voisin de x.

➔ *Contrairement à la division classique le reste (ici noté e pour erreur) peut être positif ou négatif et est limité en valeur absolu à la moitié de la valeur du diviseur.*

Démonstration : *identique à celle du théorème de division classique, il faut tout simplement lorsque le reste r est supérieur à p/2 remplacer q par (q+1) et le reste r par (r-p).*
Je vous propose maintenant d'analyser de façon plus précise ce qui se passe pendant le processus de division de x par P.

Remarquons premièrement la dissymétrie des rôles joués par le quotient et le diviseur dans la factorisation $x = pq + e$***. En effet tandis que le quotient q est neutre, le diviseur p influence directement l'ensemble des niveaux d'erreur possible*** $\left[-\frac{p}{2}, \frac{p}{2}\right]$.
Ceci entraine plusieurs conséquences sur la dynamique du triplet (P,q,e)

- ***La modification de P conduira à un ajustement progressif du niveau d'erreur jusqu'à ce que ce dernier dépasse le seuil*** $\pm\frac{P}{2}$ ***, ce n'est qu'a ce moment qu'il sera nécessaire d'ajuster le quotient q.***
- ***Le nombre de diviseur associé à un quotient sera supérieur à 1 tan que q sera supérieur à P (*** $P > \sqrt{x}$ ***et*** $q < \sqrt{x}$ ***).***
- ***lorsque P deviendra inférieur à q les ajustements sur P ne pourrons plus être compensé par un ajustement uniquement sur le niveau d'erreur mais nécessiterons en plus un ajustement du quotient.***

➔ ***si dans la factorisation de x, on se fixe un diviseur p (procédé courant de l'arithmétique classique) alors on peut lui associé un unique couple quotient – erreur (q,e). Par contre si on fixe plutôt le quotient q, alors on peut généralement***

lui associé (tan que sera inférieur à √x) plusieurs couples diviseur – erreur (P,e) compatible avec le théorème de division.

➔ *Cette stabilité du quotient lorsque le diviseur évolue dans l'intervalle [x, √x] lui confère toutes les caractéristiques des structures stables sur lesquels s'appuie la résolution de δ-problèmes. Le quotient q sera donc au cœur de la résolution du δ-problème de la recherche de diviseur d'un entier x.*

Ce procédé présente un avantage majeur dans la recherche de diviseurs approximatifs de x, c'est qu'il réduit de façon très importante le nombre de possibilités. En effet pour un nombre x dont nous cherchons les diviseurs approximatifs, nous avons exactement x choix possible pour le diviseur P et donc x possibilités à envisager, par contre si au lieu d'analyser directement l'ensemble des diviseurs potentiels, nous partons de l'ensemble des quotients auxquels ils sont rattachés, nous avons infiniment moins de possibilités (nous avons exactement √x possibilité pour q , si P est compris entre x et √x).

EXP : Supposons par exemple qu'on veuille diviser 10 et avoir un quotient de 2, alors on peut avoir comme factorisation 10 =5*2 ; 10 = 4*2 +2 ; 10 =6*2 -2 on a donc 3 factorisations possibles pour 10 lorsque le quotient est 2.

Ce procédé de division qui fixe le quotient et en déduit l'ensemble des diviseurs (plutôt que fixé le diviseur pour en déduire le quotient unique), nous l'appellerons **divisions inversé**. Et c'est cette façon de procéder que nous adopterons dans cette étude, nous en étudions les conséquences dans le paragraphe suivant. (Notons qu'on à une inversion de cette propriété lorsque le quotient devient supérieur à la racine du nombre en question, et qu'il est plus intéressant alors de retravailler avec l'approche classique. Nous le verrons dans le chapitre consacré à la factorisation)

Le lemme suivant montre comment déterminer les bornes de cette plage en partant d'une factorisation quelconque du point initiale.

Lemme 1 (point de rupture) :
Soit x un entier q un quotient de x fixé et $p = \frac{x}{q}$.

L'ajustement maximale à la hausse est donné par $A^+ = \frac{P}{2q-1}$
L'ajustement maximale à la baisse est donné par $A^- = \frac{P}{2q+1}$

$P_{min} := P^- := P - [A^-]$ *et* $P_{max} := P^+ := P - [A^+]$

L'ensemble des diviseurs compatible avec un quotient q donné est donc $[P^-, P^+]$ *et son cardinal* $Card([P^-, P^+]) \approx \frac{x}{q^2}$ *pour* $q > 1$

➔ *Plus le quotient q est élevé, moins grand sont les points de ruptures et donc moins grand est cardinal de l'ensemble des diviseurs.*
➔ *Plus le diviseur P est grand plus les points de ruptures sont éloignés et plus il y a de diviseur compatible avec q.*

➔ *Le niveau d'erreur qui est lié au diviseur arbitrairement choisi n'a comme on pouvait s'y attendre aucune incidence sur le nombre de diviseur compatible avec x pour un quotient donné.*
➔ *Le nombre de diviseur de x est donc inversement proportionnel au quotient carré du q.*

Démonstration :
Si $x = pq + e$ avec q fixé, en faisant varier p on va faire varier de façon continue le niveau d'erreur, ceci sera possible tant que le niveau d'erreur sera compatible avec les conditions fixé par le théorème de division.

$x = pq + e \rightarrow x = p'q + e' = (p + a)q + e'$ avec $e' = e - aq$. p' sera acceptable tan que $|e'| < \frac{p'}{2}$. Ceci ne sera plus possible donc lorsque a sera tel que $e' = \frac{p'}{2}$ ou $e' = -\frac{p'}{2}$.

$$e - aq = \frac{p + a}{2} \rightarrow 2(e - aq) = p + a \rightarrow 2e - 2aq - a = p \rightarrow a(2q + 1) = -(p - 2e)$$

➔ $a^- = -\frac{p-2e}{2q+1}$ et pour $p = x/q$, $e = 0$ et on à $a^- = -\frac{p}{2q+1}$

De même :

$$e - aq = -\frac{(p + a)}{2} \rightarrow 2(e - aq) = -p - a \rightarrow 2e - 2aq + a = -p$$
$$\rightarrow a(2q - 1) = (p + 2e)$$

➔ $a^+ = \frac{p+2e}{2q-1}$ et pour $P = x/q$, $e = 0$ et on a $a^+ = \frac{p}{2q-1}$

Le cardinal de l'ensemble des diviseurs est donné par :

$$Card([P^-, P^+]) = P^+ - P^- = A^+ + A^- = \frac{p}{2q + 1} + \frac{p}{2q - 1}$$

$$Card([P^-, P^+]) = \frac{4x}{4q^2-1} \approx \frac{x}{q^2}$$

Et si $q > 1$ alors $Card([P^-, P^+]) \approx \frac{x}{q^2}$

➔ Si je fixe un quotient, je peux m'amuser à bouger le diviseur, je ne tomberais plus exactement sur mon point initiale mais sur ces voisins, et vu qu'il peuvent le remplacer, ce n'est pas grave, mais viendra un moment ou le résultat obtenu sortira totalement du voisinage, c'est cet instant ou on franchit la limite que caractérise le point de rupture lemme précédent. Le lemme est construit en envisageant l'extension maximale du voisinage ($\delta=1/2$) mais rien ne nous empêche de le restreindre à notre convenance.
➔ le nombre de diviseurs compatible avec q dépend donc de la taille relative du quotient et du diviseur.
➔ Le diviseur devient unique quand $p = 2q^2 + 2qr - 1/2$ (en deçà on a plus de point qui soit compatible…)

Corollaire 1 (efficacité de la recherche par division approximative)

$$\boldsymbol{Card([P^-, P^+]) > m \rightarrow \frac{x}{q^2} > m \rightarrow q < \frac{\sqrt{x}}{m}}.$$

➔ ***L'approche est d'autant plus efficace que le rapport q est faible.***

Corolaire 2 (point de δ-rupture) :
Soit x un entier q un quotient de x fixé et p un entier quelconque permettant d'obtenir une factorisation de x de la forme $x = pq + e$ *avec* $-\frac{p}{2} < e <= \frac{p}{2}$ *et* $|e| < \delta x$ *et* $\delta < \frac{1}{2}$.
Alors les diviseurs minimaux et maximaux associé à q sont donné par :
$P_{min} = min\left(\left\lceil\frac{x-\delta x}{q}\right\rceil, p + A^{-}\right)$ *et* $P_{max} = max\left(\left\lfloor\frac{x+\delta x}{q}\right\rfloor, p + A^{+}\right)$.
L'ensemble des diviseurs δ-compatible avec un quotient q donné est donc [Pmin, Pmax]

DIVISEUR IDEAL

Le processus de division inversé que nous avons introduit dans la partie précédente nous a permis de construire pour un quotient fixé tout un ensemble de diviseurs compatible avec ces derniers. Chacun de ces diviseurs pj permet de générer une approximation xj de x. cette approximation est situé dans le voisinage de x. de la relation d'ordre définit dans le voisinage de x, nous avons alors déduit une relation d'ordre implicite entre les diviseurs reliés à un quotient donné. Celle stipule en quelque sorte que, le diviseur sera alors d'autant plus intéressant qu'il permettra de générer une bonne approximation de x. nous désignerons alors par diviseur idéal celui qui permettra pour q fixé d'obtenir la meilleure (au sens de la relation d'ordre dans le voisinage) approximation de x, i.e. le xj le plus proche de x.

Def 9 (diviseur idéal):
Soit $(xi, qi) \in X * Q$ *on appel diviseur idéal de xi, l'unique nombre pi tel que* $xi = p * qi + ei$ *tel que l'approximation* $x'_i = p * q_i$ *de x soit la plus proche de x.*

➔ Le diviseur idéal P pour un quotient fixé q_i, est celui qui permet de minimiser l'erreur d'estimation de x par $p*q_i$.

Le lemme suivant nous donne la méthode de construction de l'idéal pour un quotient fixé.

Lemme 1.2 (construction du diviseur idéal)
Soit x un entier et q un quotient fixé, le diviseur idéal de x pour le quotient q est donné par $\overline{P} = [P]$ *et* $P = \frac{x}{q}$ *, ou le [] désigne l'arrondi à l'entier le plus proche (ou la partie entière).*

Démonstration :
$X = pq + e$, on considère une factorisation alternative de x sous la forme $x = p'q + e'$, on doit montrer que |e'|>=|e|.
$x = pq + e \rightarrow ps = x - e \rightarrow p = \frac{x-e}{q}$ De même on établi $x = p'q + e' \rightarrow p' = \frac{x-e'}{q}$ or p est défini par $p = \left[\frac{x}{q}\right]$ donc e est la plus petite quantité tel que $\frac{x-e}{q}$ soit un entier, or p' est entier donc $e' >= e$.

CQFD

Notons que plus le diviseur d choisit est éloigné du diviseur idéal P, plus l'erreur d'approximation serra importante. En d'autre terme :

Lemme 1.3 (Evolution du niveau d'erreur en fonction du diviseur)
Soit X un entier et Q un quotient fixé, soit P le diviseur idéal de X pour Q et E le niveau d'erreur associé à P : $X = P * Q + E$.
soit P_1 *et* P_2 *tel que* $X = Q * P_1 + E_1$ *et* $X = Q * P_2 + E_2$.
Si $P < P_1 < P_2$ *alors* $E_1 < E_2$

Démonstration :

Soit X et q deux entiers, ont considéré une factorisation de x par q donné par $x = pq + e$.

Et soit P le diviseur idéal de x par q , on à $x = Pq + \varepsilon$ avec donc $\varepsilon \leq e$

On en déduit $\Delta P := P - p = \frac{(e-\varepsilon)}{q}$ si donc on suppose que pour le diviseur idéal l'erreur est proche de 0, on obtient $\Delta P : \left[\frac{e}{q}\right] \rightarrow P = p + \left[\frac{e}{q}\right]$ et $x = Pq + \varepsilon \rightarrow q = \frac{(x-\varepsilon)}{P} \rightarrow x = p\left(\frac{(x-\varepsilon)}{P}\right) + e$

- ➔ $x - e = \frac{P}{P}(x - \varepsilon) \rightarrow \frac{p}{P} = \frac{x-e}{x-\varepsilon}$
- ➔ On note $\delta := \frac{e}{x}$ le taux d'erreur sur x, lorsque $\varepsilon \rightarrow 0$ on à $\frac{p}{P} \approx 1 - \frac{e}{x}$ et donc $P_x = P/(1- \delta_x)$
- ➔ $x = Pq + \varepsilon$ et $x = pq + e \rightarrow e - \varepsilon = pq - Pq = q(p - P)$
  $p - P = \frac{e-\varepsilon}{q}$ *et si* $\varepsilon \rightarrow 0$ on à $\mathbf{\Delta P = \frac{e}{q}}$

La distance qui sépare un diviseur quelconque de l'idéal est égale au rapport de l'erreur sur le quotient. Diminuer l'erreur ou augmenter le quotient permet donc de rapprocher le diviseur initial du diviseur idéal.

Le diviseur idéal correspond donc à un ajustement du diviseur initial afin de réduire au minimum le niveau d'erreur. L'ajustement sera donc d'autant plus important que ce niveau d'erreur sera grand. Par ailleurs à niveau d'erreur égale on aura un ajustement inversement proportionnel au quotient, et à la valeur de x.

Par ailleurs nous avons montré que si δx est le taux d'erreur sur x pour le générateur P, alors le diviseur idéal de x est donné par $Px = P/(1- \delta x)$.

ANALYSE DE LA STRUCTURE

Nous allons maintenant évaluer l'intérêt de cette structure pour la recherche de δ-diviseurs d'un entier X.

Soit X un entier, rechercher les δ-diviseurs de X peut se faire de façon basique, en explorant un par un les éléments de [X, ..,1].

La structure que nous avons mis en évidence, découpe cet ensemble en une série de sous intervalle D(n), contenant l'ensemble des diviseurs d'ordre n de X.

$\boldsymbol{D(X,n) = [P_m(n), P_M(n)] = \left[\frac{P}{2n+1}, \frac{P}{2n-1}\right]}$ $\boldsymbol{ou\ P = \frac{X}{n}}$. Est le diviseur idéal d'ordre n de X

Et le taux couverture au niveau n est donné par :

$$\boldsymbol{C(X,n) = \frac{card(D(X,n))}{card([1,X])} = \frac{\frac{4X}{4n^2-1}}{X-1} \approx \frac{1}{n^2}\ pou\ n > 1\ et\ C(X,1) \approx \frac{1}{3}}$$

➔ On a donc une décroissance quadratique de la contribution de chaque nouvelle exploration. Plus l'ordre est grand moins important est le taux de couverture supplémentaire ajouté par le test.

Ainsi au bout de N itérations on aura une couverture totale de :

$$C(X,1,n) = \sum_{j=1}^{n} \frac{1}{n^2} = \frac{1}{3} + \sum_{j=2}^{n} \frac{1}{n^2} \quad \text{or } \frac{n(2n-1)}{3(2n+1)^2}\pi^2 < \sum_{j=2}^{n} \frac{1}{n^2} < \frac{n(2n+1)}{3(2n+1)^2}\pi^2$$

On en déduit $\boldsymbol{\frac{1}{3} + \frac{n(2n-1)}{3(2n+1)^2}\pi^2 < C(X,1,n) < \frac{1}{3} + \frac{n(2n+1)}{3(2n+1)^2}\pi^2}$

EXP : les 50 premiers tests permettent de couvrir 92-96% des δ-diviseurs potentiel.

➔ Cette structure disparait progressivement quand on se rapproche de √X (la couverture diminue jusqu'à se réduire à un unique élément quand n = √X marquant la fin de la structure) on voit alors naitre une autre structure qui sera étudié dans la second partie de cette thèse.

➔ Cette structure est donc d'une grande efficacité pour couvrir les δ-diviseur d'ordre faible. (Et sera utiliser pour la recherche du diviseur commun dans cette zone)

Ensemble des δ–diviseurs de x

~~Soit x ∈ X, on considère la suite de P donné $par\ x = nP^n + \varepsilon^n\ avec\ P0 = x\ et\ |\varepsilon^n| < \frac{p^n}{2}$.~~

~~On s'intéresse à l'évolution du couple (n, ε^n), en particulier on veut savoir à partir d'une observation de ε^n, $si\ \varepsilon^m\ (m = n + a)$ sera inférieur à un niveau E(p)= donné. i.e. on~~

~~étudie les moments ou le diviseur idéal à un niveau d'erreur inférieur que le seuil de tolérance fixé. Et en d'autre terme on veut construire l'ensemble de δ − diviseur de x.~~

~~On considère la décomposition d'ordre n de x donné par $x = nP^n + \varepsilon^n$. n sera un δ – diviseur de x ssi $|\varepsilon^n| < \delta * P^n$.~~

On considéré la factorisation de x sous la forme $x = P * q + e$ avec $p < \sqrt{x}$, alors si on fixe q, il est possible de modifier légèrement P tout en restant conforme au théorème de division (i.e. sans que le niveau d'erreur ne devienne supérieur à P/2 ou au seuil de compatibilité),

CONCLUSION

Nous avons dans cette première partie explorer les concepts fondamentaux de l'arithmétique classique dans le cadre de la δ-arithmétique.

La première étape a été la construction de l'ensemble δ-Z sur lequel on travaille dans ce nouveau cadre. Cet ensemble embarque avec chacun de ses points, les éléments de son voisinages proches ayant la possibilité de se substitué à ce dernier non pas dans les opérations arithmétiques mais dans la vérification des propriétés de l'arithmétique pour le point en question. Sur ce dernier principe nous avons donc étendue la notion de divisibilité classique d'un entier x par un entier p, à celle de δ-divisibilité autorisant le point x à se faire représenter dans la division par p, par n'importe lequel de ses voisins.

Une fois construit l'ensemble faisant l'objet de notre étude ainsi que le principe de substituabilité permettant de retrouver les propriétés de l'arithmétique classique et en particulier la divisibilité, nous avons construit sur le voisinage d'un point x, une relation d'ordre permettant de l'ordre de priorité dans le remplacement de x par un de ses voisins.

La seconde grande notion explorer dans ce chapitre est celui de la division. D'abord nous avons légèrement modifié le théorème de division classique en remplaçant la notion de « reste » de la division par celle « d'erreur » de division prenant en compte le fait qu'une erreur d'approximation pouvait être positive ou négative. Mais surtout nous avons proposé une nouvelle approche de la division d'un entier x. à l'approche classique consistant à dire : pour un entier x, si je fixe un diviseur p, alors je peux lui associer un unique couple quotient-reste (q,r) tel que $x = pq + r$. au cœur de cette approche de la division se trouve donc le diviseur p. dans la nouvelle approche que nous avons introduit nous proposons plutôt l'angle suivant : si je considère un entier x et un quotient q, alors il existe tout un intervalle $[Pmin, Pmax]$, de diviseurs et un ensemble $E = \{e1, \dots en\}$ d'erreur d'approximation tel que l'on puisse écrire $x = pq + e$ avec p dans [Pmin, Pmax], et e l'erreur de division qui lui est associé. Cette approche comme

nous le verrons par la suite est beaucoup plus riche que la précédente parce qu'elle permet d'explorer directement des intervalles entier de diviseurs.

Pour finir, ayant établi cette méthode de division nous permettant d'analyser les diviseurs non plus un par un, mais par intervalle, nous avons introduit la notion de diviseur idéal qui nous permet de dire pour un quotient donné quel est le diviseur qui nous permet d'atteindre le niveau minimal d'erreur, il représente aussi le « meilleurs » des diviseurs pour l'intervalle des diviseurs associé à q. cette notion est cruciale pour la notion de PGCD approximatif qui est l'un des objectif de cette construction, en effet elle permet d'offrir un substitut optimal à la notion de diviseur exact sur laquelle repose le PGCD classique et que nous avons identifié comme étant la principal raison du fait que le problème de calcul du PGCD soit mal posé.

Le chapitre suivant est consacré à la notion de diviseur commun et à celle de PGCD.

Chapitre 3

DIVISEUR COMMUN ET PGCD EN δ-ARITHMETIQUE

Dans le chapitre précédent, nous avons introduit et étudier le concept de δ-divisibilité, prolongeant au δ-voisinage d'un entier x, les notions de division et de divisibilité de x. en exploitant la stabilité du quotient dans [x, √x] nous avons proposé une solution au problème de la recherche de δ-diviseurs dans cet intervalle. A travers la notion de diviseur idéal permettant de minimiser l'erreur d'approximation de x par l'entier x' induit par ce diviseur, la δ-arithmétique propose donc un procéder de résolution du problème de recherche de diviseurs d'un entier ajustable dans un ensemble délimité.
Dans ce chapitre nous nous penchons sur le traitement en δ-arithmétique de la notion de diviseur commun. Nous proposons donc une extension des concepts et méthodes construites dans le chapitre précédent, pour traiter le problème similaire dans le cas ou X n'est plus un entier, mais un ensemble fini d'entiers, ce qui nous mènera naturellement à la notion de PGCD approximatif.

Soit $X := \{x^0, x^1, x^2 \ldots . x_n\}$ *un ensemble de n entiers, et soit P un entiers, on dit que P divise X ssi P divise chacun des éléments de X i.e. :* $P \mid X \leftrightarrow \exists Q$ *tel que* $X = P * Q$

$i.e.\, Q := \{q_0, q_1, q_2 \ldots . q_n\}$ *tel que* $\forall x_i \in X, x_i = P * q_i$

Le δ-problème de la recherche de diviseur commun de X, consiste non plus à confronter l'ensemble S des diviseurs potentiel à X mais plutôt à $[X]_\delta$.

Soit P le plus petit élément de x (P = min(xi, xi ∈ X) , P est appelé pivot de X), les diviseurs commun au élément de X appartiennent à l'ensemble S = [1, P].
Rappelons par ailleurs que le δ-voisinage d'un ensemble est composé de toutes les combinaisons possibles des δ-voisins de ses éléments. Il peut donc y avoir (même pour des δ très faible) un nombre important d'élément dans le δ-voisinage d'un ensemble.

En se basant sur les concepts établis au chapitre précédent on peut naturellement définir un diviseur commun comme suit :
Soit $X := \{x_0, x_1, x_2 \ldots . x_n\}$ *un ensemble de n entiers, et soit P un entiers, on dit que P est un δ-diviseur de X ssi P est un δ-diviseur de chacun des éléments de X i.e. :*
$P|_\delta X \leftrightarrow \exists Q$ *de même cardinal que X et* $X' \in [X]_\delta$ *tel que* $X' = P * Q \;\; i.e.\, Q :=$
$\{q_0, q_1, q_2 \ldots . q_n\}$ *tel que* $\forall x'_i \in X', x'_i = P * q_i$

STRUCTURE QUOTIENT ET STRUCTURES COMPATIBLE AVEC X

Nous allons maintenant définir le concept de structure compatible avec X qui se substituera à celui de quotient dans la division classique. La structure compatible, n'est rien d'autre qu'un ensemble de quotients de X partageant au moins un diviseur P.

Soit maintenant X un sous ensemble fini d'entiers ; $X := \{x_0, x_1, x_2 \ldots . x_n\}$ ~~*avec* $x_0 < x_1 < x_2 < \ldots < x_n$~~.
Et soit $Q := \{q_0, q_1, q_2 \ldots . q_n\}$ un sous ensemble d'entiers de même cardinal que X.

Def 2.5 (Quotient exact):
On dit que Q est une structure quotient de X ssi
$\exists\, \boldsymbol{P \in N} \mid \forall(\boldsymbol{xi, qi}) \in (\boldsymbol{X, Q}), \boldsymbol{xi = qi * P}.$
P est alors dit générateur de Q et diviseur de X.

Def 2.6 (Structure):
On dit que Q est une structure δ-Compatible avec X ssi
$\exists\, \boldsymbol{P \in N} \mid \forall(\boldsymbol{xi, qi}) \in (\boldsymbol{X, Q})\ \boldsymbol{et\ qi} = \left[\frac{xi}{P}\right]$.
En d'autre terme $\exists\, P \in N \mid \forall(xi, qi)\ \ xi = qi * p + ei$ avec $|ei| \leq\ \delta * xi$. et $\delta \leq \frac{1}{2}$

P est alors dit δ-diviseur ou presque diviseur de X, et q_0 ordre de la structure Q et est noté α.

Lorsqu'on ne dispose pas d'information particulière sur la valeur de δ, on parle tout simplement de structure compatible avec X.
La structure quotient est en quelque sorte la structure 0-compatible.

Lemme 2.1 :
Soit Q1 une structure d'ordre α1 compatible avec S, et Q2 une structure d'ordre α2 compatible elle aussi avec X, si $\boldsymbol{\alpha 1 < \alpha 2 \rightarrow \forall g1\ generateur\ de\ Q2, g2\ générateur\ de\ Q2, on\ à\ g1 < g2}$

Remarque :

- ➔ Une structure Q compatible avec X, associe à chacun des éléments x_i de X un quotient q_i tout en assurant la compatible avec les quotients affectés aux autres éléments.
- ➔ Pour qu'une structure soit compatible avec X, elle doit générer pour chacun des éléments de x une erreur d'estimation conforme au théorème de division ($e < p/2$), il y'a donc « très peu » d'ensemble de même cardinal que X qui soient des structures compatibles avec X.
- ➔ L'ordre d'une structure Q compatible avec X, renseigne sur le niveau de fractionnement des éléments de X par les générateurs de Q. plus l'ordre de grandeur du diviseur choisi sera proche de celui du plus petit élément de X, moins le fractionnement sera important.
- ➔ Notre recherche de PG-CD plutôt que de porter sur des diviseurs portera sur de δ-diviseur i.e. des diviseurs admettant une erreur d'estimation de δ fois l'entier

estimé, aussi serra t'elle équivalente à une recherche traditionnelle de PGCD lorsque δ sera nul.

➔ L'ordre α d'une structure fixe un ordre de grandeur pour ses générateurs, en effet tous les générateurs d'une structure d'ordre α sont proches de x_0 / α (un peu plus grand ou un peu plus petit).

Le principal intérêt des structures δ-compatibles avec X est, qu'elles nous permettent d'obtenir directement des ensembles X' δ-égaux à X et dont le PGCD est du même ordre de grandeur que leur générateur. Aussi la recherche de PGCD se résume comme nous le verrons à la construction des structures δ-compatible avec X et à l'exploration de leur générateurs.

Pré-Analyse des structures

Dans la construction d'une structure, nous confrontons un générateur P, élément de l'ensemble de diviseurs commun potentiels, afin d'extraire l'information complémentaire nécessaire pour reconstruire X à partir de P ; c'est la structure Q. étant donné que nous travaillons sur des entiers une partie de l'information est mise à l'écart en remplaçant le résultat initial de la division par [Q] (arrondi ou partie entière). Ainsi la structure réel serait composé de deux parties $Q_E = Q + Q_R$ ou $Q = \{xi/P\}$, $Q = [Q_E]$ et $Q_R = Q_E - Q$. Ainsi tous les éléments de X ne sont pas égaux devant la structure Q, utilisé dans notre recherche de diviseur commun (nous le verrons par la suite) certain vois une partie plus importante de leur information rejeté dans le résidu Q_R. C'est donc ce résidu qui va induire l'ordre des points de ruptures (les éléments les moins bien pris en compte par Q aurons plus rapidement tendance à modifier leur quotient) la différence des diviseurs idéaux et la distance entre ces idéaux et le générateur initial P (les points ayant vu une part importante de leur structure rejeté dans Q_R aurons un idéal ($Pi = xi/qi$) d'autant plus éloigné de P.
Par ailleurs étant donné que la sensibilité des points à l'erreur dépends de leur ordre de grandeur, on va ramener l'ensemble Q_R à $\xi_Q := Q_R / Q$, les éléments de X pourrons donc directement être comparé pour la structure Q en fonction de leur niveau d'erreur ξ_Q.

Nous verrons dans la suite ce qu'implique ces remarques, il faut seulement nous rappeler que chaque fois que nous utilisons la structure Q, nous négligeons une quantité d'information Q_R, qui n'est pas la même pour tous les éléments de X.

Construction de l'ensemble Q compatible avec X, généré par P

Dans le chapitre précédent nous avons vu que pour résoudre le δ-problème de la division, il était plus efficace de commencé par fixer le quotient pour ensuite analyser l'ensemble des diviseurs qui lui étaient associés. Si dans le cas d'un entier isolé il était facile d'identifier les quotients (tous les éléments inférieurs à $\sqrt{x}$) ceci l'est un peu moins dans le cas des ensembles. Le quotient étant un ensemble de quotient liés par le partage d'au moins un diviseur commun, il est nécessaire pour le construire de disposer d'un générateur P ($P \in S := [1, P_0]$, $P_0 = Min\{x_i, x_i \in X\}$). Une fois construit nous verrons que cette structure tout comme le quotient présente le caractère de stabilité que nous recherchons, et que plusieurs générateurs peuvent lui être associé (ils sont alors dit compatible avec Q)

Etant donné un entier P∈S , le lemme suivant donne la méthode de construction de la structure générée par P et compatible avec X. l'ordre et le seuil de compatibilité de la structure générée s'en déduisent.

Lemme 2.2. Génération de structure compatible avec X.

Soit X sous ensemble de N et $P \in N$: l'ensemble Q généré par P pour X est donné par: l'ensemble des quotients de la « division » des x_i par p tel que :

$$xi = P * qi + ei,\ avec\ ei\ \in Z\ et\ |ei| < P/2$$

$$Q := \left\{ qi\ \in N |\ qi = \left[\frac{xi}{P}\right] \right\}$$

Cette structure est par construction d'ordre $\alpha = q0$, et est δ-compatible avec X pour tout $\delta <= max\left(\frac{(ei)}{xi}\right)$.

- δ est le taux d'erreur maximale constaté.
- On Note $E = \{e_0, e_1, e_2\ \ldots . en\}$ l'ensemble des erreurs d'approximation de X par P*Q.
- $|E| = \sum_{i=0}^{n} |ei|$ est l'erreur absolue cumulée.

Ensemble des générateur d'une structure

Comme nous l'avons déjà signalé le diviseur (générateur) et le quotient (structure compatible) ne jouent pas des rôles symétriques (d'ailleurs dans le cadre d'un ensemble de points on voit bien que tandis que l'un est un point l'autre est justement un ensemble), de plus à une structure peut correspondre un nombre important de générateurs, nous construisons ici pour une structure données l'ensemble de ses générateurs.

Soit X un sous ensemble de N et Q une structure compatible avec X, et P un générateur quelconque de Q, nous allons maintenant construire l'ensemble [P-min, P-max] des générateurs de Q.

Lemme 2. 4 : l'ensemble [P-min, P-max] des générateurs d'une structure Q est donné par :

P-min = P - A^- et P-max = P + A^+ avec :

$$A^+ = min\left(\left(\frac{P}{2q_i - 1}\right)\right)_{i=1..n} \quad et\ A^- = max\left(\left(\frac{P}{2q_i + 1}\right)\right)_{i=1..n}$$

Ou $x_i = Pq_i \forall (x_i, q_i)\ \in X, Q$ et P un générateur quelconque de Q.

Démonstration :
On prend tout simplement les bornes min et max de l'ensemble des presque-diviseurs de chacun de xi (construit au lemme1) afin d'avoir des presque-diviseurs qui sont acceptable pour tous.

Lemme 2.5 : l'ensemble de générateur d'une structure δ-compatible est donnée par l'intersection des intervalles de δ-diviseurs des éléments de X (voir corolaire du lemme 1).

Def 2.7 (structure δ-compatible) :
Une structure est dite δ-compatible avec X si elle a au moins un générateur lui permettant d'atteindre le seuil de compatibilité δ. En d'autre terme (lemme 5) si l'intersection de ses δ-diviseur n'est pas vide.

Diviseurs Idéaux et générateur Idéal.

Soit Q une structure d'ordre o compatible avec X et soit G l'ensemble des générateurs de cette structure, chaque élément gi de G induit un ensemble X' := gi*Q situé dans le voisinage de X. de la relation d'ordre entre les éléments du voisinage de X nous pouvons donc déduire une relation d'ordre entre les générateurs de la structure Q. cette relation traduirait alors le fait qu'un générateur est d'autant meilleure que l'ensemble X' qu'il génère est proche de X.

Pour chacun des éléments de G nous pouvons construire l'ensemble X' générée puis classé ces ensembles en fonction de leur proximité à X. nous désignerons alors comme générateur idéal de Q, l'élément g de G qui nous aura permis d'obtenir l'ensemble X', le plus proche de X. la difficulté est viendra du coup d'une telle démarche, en effet nous avons vu que l'ensemble de générateur pouvait être assez vaste, il serait donc totalement déraisonnable de les évaluer un par un, nous devons donc trouver comme nous l'avons fait pour le diviseur idéal, un procédé nous permettant de déduire le générateur idéal des information de base sur la structure Q.

Rappelons que le diviseur idéal p de x par q était caractérisé par le fait qu'il minimisait l'erreur d'estimation de x par p*q.
Si p_i est le diviseur idéal de x_i par qi, alors $ei = |xi - qi * pi|$ *est l'erreur minimal d'estimation de x_i pour le quotient* $\delta i = \frac{ei}{xi}$ *le taux d'erreur minimale d'estimation de xi.*
Le diviseur idéal est donc pour un q fixé le candidat le mieux placé pour remplacer x.

Nous souhaitons maintenant étendre cette définition au cas où X est un ensemble et Q une structure compatible avec X. le déterminer parmi les générateurs possible d'une structure celui qui permet avec la structure d'engendrer l'approximation X' (X' =PQ) la plus proche possible de X.

Def 2.8 : (générateur idéal 1) :
Soit Q une structure δ-compatible avec X. on appel générateur idéal de Q celui qui permet de construire l'ensemble X'=PQ la plus proche de X en fonction de la relation d'ordre choisit.

Lemme 2.6 (générateur idéal 2):
Si on se fixe comme relation d'ordre dans le voisinage de X, la relation définit précédemment (un ensemble X' voisin de X est d'autant plus proche de X que son taux d'erreur maximale est faible), alors le générateur idéal d'une structure δ-compatible avec

X est donné par $G = \left(\frac{2\,(P_{max} * P_{min})}{P_{max} + P_{min}}\right)$ *ou* P_{max} *et* P_{min} *désignent respectivement les diviseurs idéaux minimaux et maximaux.*

➔ *Ici est mis en évidence le fait que ce sont les deux extrémités de l'ensemble des idéaux individuel, qu'il faut réconcilier.*

De $G = \left(\frac{2\,(P_{max} * P_{min})}{P_{max} + P_{min}}\right)$ *on déduit* $G = \frac{2xy}{(x+y)-(x*\delta y + y*\delta x)}$ *ou x et y sont respectivement les éléments de taux d'erreur minimaux et maximaux et* δ_x , δ_y *leur niveau d'erreur respectif.*

➔ *Cette expression montre que se sont les éléments dont les taux d'erreurs sont les plus importants (positif et négatif) qui sont à ajuster, ce qui correspond bien à l'idée intuitive de la mesure d'erreur individuel.*

On peut aussi exprimer G comme : $G = P_{min} + \left(\frac{\lambda_{min}}{\lambda_{max} + \lambda_{min}}\right) * D$ *ou* $D = P_{max} - P_{min}$ *est la distance qui sépare les deux extrémités de l'ensemble des idéaux individuels et* $\lambda_i = \frac{q_i}{x_i}$

➔ *Ici on voit bien qu'une fois de plus les deux extrémités sont mises à contribution et que pour trouver le consensus, l'effort demandé à chacun est proportionnel à une sorte d'intensité égale au rapport de son quotient q sur la valeur du point x.*

Une autre expression intéressante de G est celle en fonction du déplacement induit sur le point fixe de la structure, le diviseur idéal de X0, qui étend donné un ordre n est commun à toutes les structures d'ordre n, c'est aussi générateur naturelle (celui utilisé pour la construction de la structure Pn = P/n)

De $G = \left(\frac{2\,(P_{max} * P_{min})}{P_{max} + P_{min}}\right)$ *on déduit* $G = P_n * \left(1 + \frac{1}{2P_n + (B-A)}\left((B-A) - \frac{2*A*B}{P_n}\right)\right)$ *ou A et B sont respectivement les distance entre le point fixe Pn et les extrémités gauche et droite de l'ensemble de idéaux (*$A = P_n - P_{min}$ *et* $B = P_{max} - P_n$ *et* $P_n = \frac{x_0}{n}$*).*

➔ Cette relation est très importante car elle relie l'idéal à un point fixe partagé par toutes les structures de même ordre, et que l'ajustement apporté par chaque structure dépend uniquement des distances à ce point fixe des deux points qui lui sont le plus éloigné (à gauche et à droite)

➔ Une approximation de cet ajustement lorsque la distance entre A et B est faible devant Pn (ce qui est souvent le cas, et sinon induit un taux d'erreur important) est donné par

$$G \approx P_n * \left(1 + \frac{(B-A)}{2P_n} - \frac{A*B}{2P_n^2}\right) = P_n * \left(1 + \left(\frac{1}{2}(B_n - A_n) - (B_n * A_n)\right)\right)$$

$$et\ B_n = \frac{B}{P_n}\ et\ A_n = \frac{A}{P_n}$$

Le niveau d'erreur du point xi est alors donné par :

$$\delta_i = \frac{x_i - G*q_i}{x_i}\ 1 - \ G * \frac{q_i}{x_i} = \frac{\varepsilon_i}{q_i}\left(1 + \frac{1}{2P_n + (B-A)}\left((B-A) - \frac{2*A*B}{P_n}\right)\right)$$

$$\delta_i \approx \frac{\varepsilon_i}{q_i}\left(1 + \frac{1}{2}(B_n - A_n) - \frac{1}{2}(B_n * A_n)\right) \approx \frac{\varepsilon_i}{q_i}$$

Démonstration :
Il suffit de remarquer que pour tout autres choix de générateur le taux d'erreur sur l'un au moins des éléments de taux d'erreur min ou max sera supérieur à celui construit ici. En effet en situant le générateur idéal de façon à ajuster au mieux le point d'erreur maximale par rapport au point d'erreur minimale, on arrive à la réduction maximale de l'erreur pour l'élément dont le taux d'erreur est max.

On considère un intervalle [x,y] on cherche le point z de cet intervalle tel que le niveau d'erreur induit sur les points x et y en les ramenant à ce point z soit égal. i.e. $\frac{z-x}{x} = \frac{y-z}{y} \rightarrow y(z-x) = x(y-z) \rightarrow z(y+x) = 2xy \rightarrow z = \frac{2xy}{x+y}$. $CQFD$

Dem 2:

~~$\frac{(P+a)*q_m - x_m}{x_m} = \frac{x_M - P*q_M}{x_M} = P * \frac{q_m}{x_m} - 1 = 1 - P * \frac{q_M}{x_M}$~~
~~$\rightarrow P * \lambda_m + P * \lambda_M = 2 \rightarrow P(\lambda_m + \lambda_M) = 2$~~

Dem 3 :
$$G = \left(\frac{2\,(P_{max} * P_{min})}{P_{max} + P_{min}}\right) = \left(\frac{2*(P_n - a) * (P_n + b)}{(P_n - a) + (P_n + b)}\right) = P_n * \left(\frac{2*P_n + (b-a) - 2ab}{2*P_n + (b-a)}\right)$$
$$G = P_n * \left(1 + \frac{(b-a)}{2*P_n + (b-a)} - \frac{\frac{2ab}{P_n}}{2*P_n + (b-a)}\right) = P_n * \left(1 + \frac{1}{2*P_n + (b-a)}\left((b-a) - \frac{2ab}{P_n}\right)\right)$$

Dem : soit [x,y] l'intervalle contenant l'ensemble de diviseur idéaux. Nous voulons montrer que le niveau d'erreur induit en choisissant $z = \frac{2xy}{x+y}$ comme générateur idéal de la structure est le niveau d'erreur minimale que l'on puisse atteindre pour cet ensemble de diviseur.

Supposons dans un premier temps que les diviseurs idéaux sont des diviseurs stricts (niveau d'erreur implicite nul pour chacun d'eux).

Soit d, un élément quelconque de l'ensemble D des diviseurs idéaux. On à : $d \in]x,z[$ ***ou*** $d \in]z,y[$ ***ou*** $d \in \{x,z,y\}$

Si on choisit un point z'> z, alors x qui est déjà le point le plus petit (donc celui qui supporte le moins bien le déplacement de son idéal) va subir un choc encore plus important et donc une trop importante augmentation de son niveau d'erreur. Si donc tous les idéaux étaient à des niveaux d'erreur nul (ou égaux) le taux d'erreur sur le générateur serais celui égal au taux d'erreur sur le déplacement de x (qui ici à été construit pour être égal a celui de y)

Cas 1 : $z' \in \{x, z, y\}$ alors x et y présentent le taux d'erreur liè au déplacement de leur idéal vers z
Cas 2 : $z' \in]x, z[$ si z' est inférieur à z, alors y subira un déplacement plus important que précédemment ce qui induira pour z un niveau d'erreur supérieur a δ et donc un niveau d'erreur global du générateur plus grand.
De même, si z' est supérieur à z alors x subira un déplacement plus important que précédemment ce qui induira un niveau d'erreur plus grand pour x et pour le générateur.

Dans le cas plus général ou les taux d'erreur initiaux des générateurs idéaux ne sont pas nul ou égaux, il faudrait commencer par ajuster l'ensemble des générateurs pour les ramener à un niveau d'erreur initial égal à celui du point présentant le plus fort taux d'erreur (éventuellement un ajustement avec les deux seules bornes de l'intervalle suffirait).

NB : si x et y sont proches la quantité $\frac{y-x}{2}$ peut être utilisée pour approximer ce point. Aussi chacun des points extrêmes sera affecté d'un taux d'erreur correspondant à la moitié de la distance qui le sépare de l'autre extrême.

Lemme 2.7 (générateur idéal 3):
Si on se fixe plutôt comme relation d'ordre sur le voisinage de X celle stipulant qu'un ensemble X' est d'autant plus proche de X que son erreur cumulée est faible. Alors le générateur idéal de Q est égal à la moyenne de ses diviseurs idéaux ajustés.

L'ajustement des diviseurs idéaux consiste à les ramener tous au taux d'erreur de celui dont l'erreur est maximale.

Démonstration :
La proximité étant construit ici sur la capacité à être en moyenne proche du maximum des points, en choisissant comme générateur idéal, la moyenne des diviseurs idéaux ramener à un niveau d'erreur équivalent, on atteint le niveau d'erreur absolu moyen le plus faible.

Démonstration :

Lemme 2.8 (générateur idéal 4):
Si on se fixe plutôt comme relation d'ordre sur le voisinage de X celle stipulant qu'un ensemble X' est d'autant plus proche de X que $\sum_{i=0}^{n} x_i = \sum_{i=0}^{n} x'_i$ *(i.e. celle qui conserve globalement la taille de X) . Alors le générateur idéal P de Q est donné par* $P = \left[\frac{\sum_{i=0}^{n} x_i}{\sum_{i=0}^{n} n_i}\right]$

Démonstration :
Soit P un diviseur commun des xi et générateur de Q, alors $x'_i = P * q_i \rightarrow \sum_{i=0}^{n} x'_i = \sum_{i=0}^{n} P * q_i = P \sum_{i=0}^{n} q_i$ et $\sum_{i=0}^{n} x_i = \sum_{i=0}^{n} x'_i \rightarrow P = \frac{\sum_{i=0}^{n} x_i}{\sum_{i=0}^{n} n_i}$ on prend l'arrondi si cette quantité n'est pas entière.

Lemme 2.9 (borne inférieur du taux d'erreur):
Le taux d'erreur globale est supérieur aux taux d'erreur de chacun diviseurs idéaux.

Rq : plus les diviseurs idéaux sont proche, plus le générateur idéal se rapproche de son niveau d'erreur minimale (qui est égale au niveau d'erreur maximale de diviseurs idéaux individuel). Dans le cas contraire, le taux d'erreur finale sera d'autant plus éloigné que les diviseurs idéaux d'erreur minimale et maximale seront éloignés.

Qualité du générateur idéal

La qualité (sa capacité à atteindre un niveau d'erreur faible) du générateur idéal est intrinsèquement liée à celle de diviseurs idéaux de xi. En particulier au consensus entre les diviseurs idéaux (leur capacité à être proche les uns des autres).
On aura donc le générateur idéal le mieux adapté lorsqu'on a un consensus entre les diviseurs idéaux, en effet plus les diviseurs idéaux sont proches plus faible sera l'erreur final (pour le générateur idéal).

La qualité d'une structure est liée à la quantité d'information Q_R mise de côté dans sa construction et à la façon dont cet ajustement est répartie entre les différents éléments de Q. l'équité de cette répartition à une incidence direct sur la capacité de Q à produire un consensus entre les diviseurs idéaux. Par ailleurs plus négligeable sera cette information, plus adapté sera l'idéal choisit.

Lemme 2.9 (consensus entre diviseurs idéaux)
Soit X un sous ensemble de N, et Q la structure compatible avec X générée par P. pour chacun des points de X on a donc la factorisation $x_i = pq_i + e_i$ *et* $\delta_i = \frac{e_i}{x_i}$ *. Alors les diviseurs idéaux sont identiques ssi les taux d'erreur δi sont identiques. Et plus les δi sont proches plus on se rapproche de consensus entre les diviseurs idéaux.*

Propriétés des structures compatibles avec X

~~Lemme 2.3 :~~
~~Soit Q est une structure compatible avec X.~~

~~*1- Si P1 et P2 sont des générateurs de Q, alors tout P tel que* $P1 < P < P2$ *est aussi un générateur de Q.*~~

~~*Si P1 et P2 ne sont pas des générateurs de Q, alors tout P tel que* $P1 < P < P2$ *n'est pas un générateur de Q.*~~

➔ ~~L'ensemble des générateurs d'une structure constitue un intervalle borné [P-min, P-max] de N.~~

~~2- Si $\delta = 0$ (cas des structures Quotient : arithmétique classique) alors le générateur de P est unique.~~

~~Démonstration :~~

ENSEMBLE DES STRUCTURES COMPATIBLE AVEC X

Dans le cas de la division approximative d'un entier unique, l'ensemble des quotients sur lequel s'appuyait la recherche était relativement simple à identifier, il était égal à l'intervalle $[1, \sqrt{x}]$. Aussi pour rechercher un δ-diviseur de x, il suffisait de parcourir successivement l'ensemble de ses quotients et d'analyser l'ensemble des diviseurs qui lui était associé, pour trouver celui respectant les contraintes de proximité qu'ont imposé sur l'approximation de x. le parcourt des quotients compris dans $[1, \sqrt{x}]$ permettait bien d couvrir, l'ensemble $S = [1, x]$ des solutions possibles au problème de δ-division.
Nous avons vu dans le paragraphe précédent comment construire et exploiter un équivalent de ce quotient dans le cas ou X était un ensemble. Mais pour résoudre le δ-problème de la division dans le cas ou X est un ensemble, il nous faut construire l'ensemble des structures qui sont compatible avec X, ce qui est moins trivial que dans le cas ou x est un entier et qui fera l'objet de cette étude.

L'ensemble S des solutions potentielle au problème de δ-division d'un ensemble X est donné par $S = [1, P_0]$ il s'agit ici de construire un ensemble de structures tel que l'ensemble des générateurs qui leur sont associés recouvre S.

Soit $X := \{x_0, x_1, x_2 \ldots . x_n\}$ un ensemble de n entiers et $P_0 = \text{Min}(X)$ son pivot.
Soit $Q := \{q_0, q_1, q_2 \ldots . q_n\}$ une structure compatible avec X. on dit que Q est d'ordre α si $q_0 = \alpha$.
L'ordre α d'une structure définit donc le niveau de fractionnement du pivot P_0 de l'ensemble X auquel il est rattaché.

Si donc on regroupe les structures compatibles avec X en « classe d'équivalence » contenant les structures induisant le même niveau de fractionnement du pivot (i.e. les structures de même ordre), alors la réunion des classes d'équivalence des structures d'ordre 1 à P0 suffira à couvrir l'ensemble $S=[1,P_0]$ (car les quotients possibles de P_0 sont les entiers de $[1,P_0]$)

Lemme :
L'ensemble des générateurs de structure d'ordre n est donné par Gn(X)
$[\mathbf{1}, \boldsymbol{P0}] = \cup_{n=1}^{Po} \boldsymbol{G_n(X)}$ ***ou*** $\boldsymbol{G_n(X) = [P_n^m, P_n^M]}$

$$et\ P_n^m = p_n - \frac{p_n}{2n-1}, P_n^M = p_n + \frac{p_n}{2n+1}\ \ et\ p_n = \frac{P_0}{n}$$

➔ Toutes structures à un ordre unique, compris dans [1,P_0], aussi parcourir l'ensemble des structures de chacun de ces ordres, permet de couvrir la totalité des solutions possibles au δ-problème du diviseur commun.

Ensembles des structures de même ordre

Soit X un ensemble et n<P0, nous voulons construire l'ensemble des structures d'ordre n compatible avec X.

Nous avons établi au chapitre précédent que l'ensemble des diviseurs d'ordre n de P0 était donné par $D(P_0, n) = [P_n^m, P_n^M]$ $et\ P_n^m = p_n - \frac{p_n}{2n-1}, P_n^M = p_n + \frac{p_n}{2n+1}\ \ et\ p_n = \frac{P_0}{n}$.

Lemme : toute structure d'ordre n est générée par un élément de $D(P_0,n)$ et réciproquement tout élément n'appartenant pas à $D(P_0,n)$ ne génère pas une structure d'ordre n.
Démonstration : évident par construction

L'ensemble des structures d'ordre n donné n'est pas vide, il contient au moins une structure, la structure naturelle.

Def : On appelle structure naturelle d'ordre n ($n<P_0$) la structure générée par $P_n = \frac{P_0}{n}$.
P_n est encore appelé pivot d'ordre n, ou générateur naturelle d'ordre n.

Il est évident que cette structure est d'ordre n, cependant elle n'est pas la seule structure de cet ordre.

La structure naturelle n'est pas l'unique structure d'ordre n, il existe d'autres structure dite secondaire, elles sont obtenu en choisissant un générateur suffisamment à gauche ou à droite de Pn.

Soit $Q = \left\{q_i,\ q_i = \left[\frac{x_i}{P_n}\right]\ i = 1..n\right\}$ la structure naturelle d'ordre n, on définit l'ensemble des points de ruptures à droite de cette structure par $R_n^+ = \left\{P_i^+,\ P_i^+ = \frac{x_i}{q_i} + \frac{x_i}{q_i(2n-1)}\right\}$ on définit de même l'ensemble des points de rupture à gauche par : $R_n^- = \left\{P_i^-,\ P_i^- = \frac{x_i}{q_i} - \frac{x_i}{q_i(2n+1)}\right\}$

Si $P_i^- > P_n^m$ ***alors*** P_i^- génère une structure secondaire d'ordre n compatible avec X.
De même si $P_i^+ < P_n^M$ ***alors*** P_i^+ génère une structure secondaire d'ordre n compatible avec X.

Les structures secondaire d'ordre n sont les générées par les éléments de $G_n^+\ et\ G_n^-$.

~~On montre que l'ensemble des structures d'ordre n compatible avec X.~~

Origine de la distance des idéaux au générateur naturelle

Nous nous intéressons ici à facteur déterminant la distance des idéaux des points de la structure au point fixe de l'ordre (n) qui est l'idéal du pivot d'ordre n.

Lemme : Soit Pn , le générateur de la structure naturelle d'ordre n, A et B l'ensemble de distance à Pn des idéaux de X pour la structure naturelle, alors les éléments de A et B sont les distance minimale à Pn pour toutes structure d'ordre n.

Démonstration :
Pn générateur de la structure naturelle -> Pn = P0/n = xo/n
On considère la structure naturelle non arrondi Q_n = {xi/Pn}
Pour la structure Q_n les idéaux des éléments de X seraient données par :
$P_i = \frac{xi}{qi} = \frac{xi}{\frac{xi}{P_n}} = P_n$. Tous les idéaux pour cette structure non arrondi seraient identique au générateur d'ordre n. il en découle que la distance entre les idéaux des point de X et ce point fixe est induit par l'erreur introduite en remplaçant qi par [qi] (arrondi ou partie entière)

On pose $q\_i = qi + \varepsilon i$ (avec Ɛi < 1 si [] est la partie entière, et Ɛi < ½ si [] est l'arrondi à l'entier le plus proche)

La distance entre Pi l'idéal du ieme point de X et Pn le point fixe et idéal du pivot est donné par :

$$ai := Pn - Pi = \frac{xi}{q_i} - \frac{xi}{q\_i} = \frac{xi}{qi}\left(1 - \frac{xi}{q\_i} * \frac{qi}{xi}\right) = \frac{xi}{qi}\left(1 - \frac{qi}{q\_i}\right) = Pn\left(1 - \frac{q_i + +\varepsilon i}{q_i}\right) =$$

$$Pn * \left(\frac{\varepsilon i}{qi+\varepsilon i}\right) = Pn * \left(\frac{\varepsilon i}{q\_i}\right) =$$

- ➔ ***Plus l'erreur introduit par l'arrondi est important plus on est éloigné du point fixe.***
- ➔ ***Plus le quotient qi est important moins on est sensible à Ɛi.***

Supposons maintenant que l'on passe de qi a qi +/-1

$$a'i := Pn - Pi = \frac{xi}{q_i} - \frac{xi}{q'\_i} = Pn\left(1 - \frac{qi}{qi+\overline{-1}}\right) = Pn\left(1 - \frac{-/+1}{qi+/-1}\right)$$

A'>a
*Si ei > 1*2 alors a'*

Lemme : soit P^n le pivot d'ordre n, Qn la structure naturelle et Pn l'ensemble des diviseurs idéaux associé à Qn. Soit G un générateur de structure d'ordre n (i.e. $R_0^- < G < R_0^+$) et P'n l'ensemble des diviseurs idéaux associé à la structure induite par G.
Si $G > P$ alors $P'_i \geq P_i \; \forall (P'_i, P_i) \in P'_n * P_n$
Si $G < P$ alors $P'_i \leq P_i \; \forall (P'_i, P_i) \in P'_n * P_n$

Démonstration :

$G > P_n \to q' \coloneqq \left[\frac{x}{G}\right] < \left[\frac{x}{P}\right] \coloneqq q$ Et $q' < q \to P' \coloneqq \left[\frac{x}{q'}\right] > \left[\frac{x}{q}\right] \coloneqq P$ on en déduit $G > P_n \to P' \geq P$ CQFD.

- ➔ Choisir un générateur dans R_n^+ entraine une réduction ou une conservation de idéaux initiaux.
- ➔ Choisir un générateur dans R_n^- entraine une augmentation ou une conservation de idéaux initiaux.

PGCD (X), δ-PGCD(X), PGCD$_\alpha$(X), PG-CD([X])

Dans les paragraphes précédent, nous avons introduit la notion de δ-diviseur commun et nous avons vu comment celui-ci pouvait être construit via l'exploitation des structure stables, nous avons ensuite vu comment construire l'ensemble des structures stables, nous disposons donc désormais de tout le matériel nécessaire pour introduire les notions de PGCD approximatifs. Pour ce faire nous allons procédé comme nous l'avons fait jusqu'ici en reposant le problème tel qu'il est formulé en arithmétique classique et en le généralisant pour prendre en compte l'impact du remplacement de X par son δ-voisinage.

Soit $X := \{x_0, x_1, x_2 \ldots . xn\}$,

Commençons par redéfinir le PGCD classique dans le cadre de la δ-arithmétique.

Lemme 2.10 (PGCD classique):
Le PGCD (au sens classique du terme) de X est le générateur idéal P de la structure quotient de X qui est d'ordre maximale.

Démonstration :

Soit P le Pgcd (classique) de X, on pose $\alpha = \frac{x_0}{p}$, soit Q la structure compatible avec X générée par P, il est évident que le générateur idéal de cette structure est P (aucune erreur d'évaluation des xi). P sera donc le PGCD si il n'existe pas de P' > P générateur d'une structure Quotient. Supposons que P' généré une structure quotient alors P' serait un diviseur commun aux xi et serait supérieur à P, ceci contredirait l'hypothèse P est le PGCD. Nous avons donc démontré que la structure quotient générée par le PGCD était la structure quotient d'ordre maximale.
Par ailleurs si Q est une structure quotient alors il existe un générateur de Q qui soit un diviseur commun aux xi notons le P' ce générateur. De plus si Q est d'ordre maximale alors P' alors tout diviseur commun aux xi génère une structure d'ordre inférieur et donc tout autre diviseur commun est inférieur à P', P' est donc le PGCD. CQFD

Etant donné que la δ-arithmétique relaxe la condition de divisibilité stricte qu'implique les structures quotient (0-compatible) pour intégrer la division à δ prêt qu'offre les structure δ-compatible, on peut aisément généraliser la notion de PGCD précédent pour obtenir la notion de PGCD approximatif suivante.

Def 11 : δ-PGCD de X
*Le δ-PGCD de X qu'on note δ-P(X) est le générateur idéal, de la structure δ-compatible qui est d'ordre maximale. (*Ou dit autrement, c'est le plus grand des générateurs idéaux des structures δ -compatible avec X)*.*

RQ :

- ➔ Le δ-PGCD est le plus grand des δ-diviseur commun.
- ➔ Dans le calcul du δ-PGCD, on fixe un niveau d'erreur maximale admissible (δ), et on choisit comme PGCD le plus grand des générateurs de structures respectant cette δ-compatibilité avec X.
- ➔ Il s'agit ici de maximiser l'ordre de grandeur en respectant le critère de δ-compatibilité de la structure.

NB :
Il s'agit en effet de trouver parmi les générateurs de la première structure δ-compatible avec X, celui qui minimise l'erreur. L'idée étant que même si on a atteint le seuil de tolérance qu'on c'était fixé, on peut encore à moindre coup réduire le niveau d'erreur.

L'une des raisons qui a motivé notre recherche du PGCD approximatif était le fait que le PGCD classique même l'lorsqu'il existait était souvent d'un ordre de grandeur sans commune mesure avec nos données initiale, or dans la pratique on a souvent un seuil en deçà duquel le PGCD ne serait plus adapté pour le problème qu'on essaie de résoudre, c'est souvent le cas dans les problématiques de simulation. Le concept de PGCD d'ordre α que nous allons maintenant introduire va nous permettre de répondre en ce besoin, en spécifiant un niveau maximale de fractionnement tolérable et sous cette contrainte de minimiser le taux d'erreur admissible, il se définit comme suit.

Def 2.9 (PGCD d'ordre α) :
Le qu'on note $P_\alpha(X)$ est le générateur idéal de la structure d'ordre inférieure ou égal à a qui est la plus proche (au sens de la relation d'ordre choisit). C'est des générateurs idéaux supérieur ou égal à α, celui dont la structure donne offre le meilleur seuil de compatibilité δ*.*

- ➔ Ici on commence par fixer un ordre de grandeur pour le PGCD puis on minimise l'erreur.

Def 2.10 : PGCD ([X])
Soit $[X] := \{[x0 - \varepsilon 0, x0 + \varepsilon 0], [x1 - \varepsilon 1, x1 + \varepsilon 1], \ldots . [xn - \varepsilon n, xn + \varepsilon n]\}$ *une série d'intervalle centré autour des points de X, on définit le PGCD de [X] qu'on note PGCD([X]), le PGCD d'un ensemble X' tel que P est grand et X' proche de X.*

Ici un ajustement de l'exigence sur le taux d'erreur δ est fait au moyen d'une fonction f(δ, α) décroissante en α. La fonction donnant l'erreur ajustée sera choisi en fonction de la préférence pour la minimisation de l'erreur ou pour la maximisation de l'ordre de grandeur. (Exp : $f(\delta, \alpha) = \sqrt{\alpha} * \delta$) ou $f(\delta, \alpha) = \delta/\alpha$ …

- ➔ Le PGCD de [X] recherche un compromis entre la taille du PGCD et son ordre de grandeur. Tous en ajustant au mieux chacun des points par rapport au

centre de l'intervalle (contrairement au δ -PGCD qui privilégie systématiquement l'ajustement du taux global d'erreur i.e. du point le plus éloigné)

CONCLUSION

Dans ce chapitre nous avons généralisé à cas de sous ensemble fini de Z les notions établies dans le chapitre précédent.

Premièrement, le voisinage d'un ensemble a été déduit comme étant composé de l'ensemble des voisinages des points qui constituaient cet ensemble. Ainsi, notre ensemble X peut être représenté par une quelconque des combinaisons d'éléments pris dans les voisinages de ses éléments. Nous avons ensuite proposé plusieurs relations d'ordre sur le voisinage de X. ce choix d'une relation d'ordre dépendant du problème que l'on souhaite résoudre, l'ensemble le plus proche de l'ensemble de départ pouvant alors être celui dont le point le plus éloigné l'est moins que celui des autres, ou celui dont la moyenne des distance des points à ceux qu'ils représente est la plus faible.

La seconde étape a consisté en la généralisation de l'approche de la division déjà établi au chapitre précédent pour un nombre x donné. Pour un ensemble X nous avons proposé une méthode de génération de structure Q compatible avec cet ensemble (contrairement au cas des points isolés ou il suffisait de choisir les quotients successifs, 1, 2,3… il faut ici construire des ensembles de quotients qui conviennent simultanément pour tous les éléments de X). Une fois cette structure construite nous avons comme dans le chapitre précédent déduit l'ensemble de ses générateurs, ces derniers sont en quelque sorte des « diviseurs communs » (δ-diviseur en réalité) à tous les éléments de X. pour cet intervalle de diviseur commun associés à la structure Q, nous avons proposé une méthode permettant de déduire le diviseur commun idéal i.e. celui permettant d'obtenir l'approximation X' de X la meilleur au sens de la relation d'ordre choisit.

De la notion de « diviseur commun idéal » associé à un quotient Q ainsi construit, nous avons déduit aisément une définition du PGCD, et effet plusieurs définitions parce que en plus de la taille du diviseur, la qualité de l'approximation qui s'en déduit rentrait en jeux.

La question qui se pose maintenant est celui d'un algorithme de construction, de ce PGCD, nous verrons que, s'il est possible de déduire de nos résultats déjà établi un algorithme de construction des différent PGCD, de nombreuses optimisation peuvent être apportées et sont l'objet du chapitre suivant.

Chapitre 3

CALCUL DES PG-CD PAR DIVISION APPROXIMATIVE

Dans les chapitres précédents, nous avons introduit successivement les notions de δ-voisinage, δ-divisibilité, et de δ-diviseur commun, ce qui nous a conduits naturellement, a de nouveau concept de PGCD (δ-PGCD, PGCD d'ordre α, …) formalisant le concept traditionnel de PGCD approximatif. Ces chapitres nous ont permis de nous doter de tout le matériel nécessaire pour calculer les PGCD approximatif. Dans ce chapitre nous utilisons ce matériel pour présenter un algorithme de calcul du PGCD approximatif. Nous en profitons pour analyser en détail la méthode de construction et introduisons des optimisations.

ALGORITHME DE CONSTRUCTION DU PGCD APPROXIMATIF

Dans ce paragraphe nous regroupons le matériel accumulé dans les chapitres précédents pour construire un premier algorithme de calcul du PGCD approximatif, celui-ci sera ensuite analyser en détail et des optimisations seront introduites.

Soit $X = \{x_0, x_1, \dots, x_n\}$ l'ensemble des données dont on recherche le PGCD approximatif. On suppose que tous les éléments de X sont différents (sinon on supprime les doublons)

On note P_0 le pivot de X ($P_0 = Min\{x_i, x_i \in X\}$) Alors le PGCD approximatif que nous recherchons est un élément de l'ensemble $S = [1, P_0]$

1. *Ensemble des solutions et Pivot*

La fonction GetPivot(X) retourne le pivot qui sera utilisé pour la construction du PGCD, ici elle est tout simplement égale au plus petit élément de X.

Entier P0 = GetPivot(Vecteur X)

{Retourne P0 = Min(X) }

2. *Construction de la structure naturelle d'ordre n*

Construction du pivot d'ordre n : ***Reel $P_n := P_0/n$.***

Construction de la structure naturelle d'ordre n :

Vecteur Qn = BuildStructure(Vecteur X_input, Reel P_gen)

{Pour chacun des éléments de X calculer

Entier qi := Arrondi(xi/P_gen)

Retourne Qn := {qi} }

3. Déterminer l'ensemble des points de rupture

Construction des points de rupture à droite du pivot d'ordre n

Vecteur G+ = ComputeRigthBreakPoints(Vecteur X_input, Vecteur Q_struc)

{Pour chacun des éléments de X calculer

Réel Pi = xi/qi

*Réel bi := Pi *(1-1/(2* qi -1))*

Retourne G+ := {bi} }

Construction des points de rupture à gauche du pivot d'ordre n

Vecteur G-= ComputeLeftBreakPoints(Vecteur X_input, Vecteur Q_struc)

{Pour chacun des éléments de X calculer

Réel Pi = xi/qi

*Réel bi := Pi *(1-1/(2* qi +1))*

Retourne G- := {bi} }

Fixation des limites de l'ordre et de la structure naturelle

Reel OrderGenMin = $(x_0/q_0)(1-1/(2* q_0-1))$*

Reel OrderGenMax = $(x_0/q_0)(1-1/(2* q_0+1))$*

Reel StrucGenMin = Min{G+}

Reel StrucGenMax = Max{G-}

4. Calculer le générateur idéal de la structure

Construire l'ensemble des diviseurs idéaux de la structure

Vecteur Pn = BuildStructureIdeals(Vecteur X_input, Vecteur Q_structure)

{Pour chacun des éléments de X calculer

Réel Pi := xi/qi

Retourne Pn := {Pi} }

Calculer le générateur idéal.

Entier Gn = ComputeIdeal(*Vecteur P_ideal)*

{Reel Pmin = Min{ P_ideal}

Reel Pmax = Max{ P_ideal}

Reel Gn = (2 Pmax* Pmin)/ (Pmax+ Pmin)*

Reel AjustGn = Gn

Si (Gn > StrucGenMax) Alors AjustGn = StrucGenMax

Si (Gn < StrucGenMin) Alors AjustGn = StrucGenMin

Retourne Arrondi(AjustGn)}

Calculer le taux d'erreur induit par Gn

Réel δn = ComputeErrorRate(Vecteur X_input, Entier G_gene)

{ Pour chacun des éléments de X calculer

*, δi = 1 – (G_gene/x)*Arrondi(x/G_gene)*

Δ : = Vecteur(|δi |)

Retourne Max(Δ)}

5. Construire les structures secondaires de droite/gauche

Déterminer les générateurs des structures secondaires d'ordre n

Vecteur G = GetSndStrucGen(Vecteur G-_Breaks, Vecteur G-_Breaks)

{Pour chacun des éléments de G+ et G- faire

Si (G+(i) < OrderGenMax) Ajouter G+(i) G.

Si (G-(i) < OrderGenMin) Ajouter G-(i) G.

Retourne G }

Déterminer les idéaux et niveau d'erreur des structures secondaire

Pour chacun des éléments de G reprendre l'étape 4.

Synthèse : algorithme de construction du PGCD

ANALYSE ET OPTIMISATION

TYPE DE PGCD

Le type de pgcd va nous permettre de fixer les conditions d'arrêt de l'algorithme.

Pour un calcul de PGCD d'ordre α, les itérations se font sur les structure d'ordre 1 à α, lorsque l'ordre α est atteint, l'algorithme s'arrête et le PGCD retourner est celui des générateurs idéaux des α structures explorées ayant conduit au plus petit niveau d'erreur.

Pour le δ-PGCD on explore les structures d'ordres successifs en partant de l'ordre 1 et on s' arrête dès qu'on trouve une structure dont le générateur idéal conduit à un taux d'erreur inférieur à δ.

Pour le PGCD classique, on procède comme pour le δ-PGCD avec $\delta = 0$.

ENSEMBLE DES SOLUTIONS ET CHOIX DU PIVOT

Nous avons établi que le PGCD de X appartenait à l'ensemble $S = [1, P_0]$, P_0 étant le plus petit élément de X. il y a donc un lien direct entre le coup de recherche du PGCD de X et la taille de son plus petit élément. le but de ce paragraphe est de réduire au minimum possible cet ensemble, pour cela nous allons nous basé sur les propriétés des différent PGCD.

Cas du pgcd classique

Propriété : Si P divise X (i.e. P divise chacun de ses éléments) alors P divise toutes combinaison linéaire d'éléments de X.

Lemme 3.6 (construction du pivot optimal)
Soit $E = \{e_0, e_1, \dots, e_n | \ e_i = x_i - \left[\frac{x_i}{x_0}\right]\}$ ***alors la plus petite des combinaisons linéaires d'éléments de E est donné par*** $\varepsilon = Min\{t = (mn+1)x - my$ ***ou*** $n = \left[\frac{y}{x}\right], m = \left[\frac{x}{y'}\right]$ ***et*** $y' = y - nx, \forall (x,y) \in E\}$
Prendre $P_0 := \varepsilon$ ***comme pivot***

Démonstration :
On considère un couple$(x,y) \in N$ *avec* $x < y$ *et* $x \neq 0, y \neq 0$. On veut construire t tel que $\forall (\alpha, \beta) \in Z$ *on ai* $t < \alpha x + \beta y$.
Soit n le diviseur idéal de y par x ($n = \left[\frac{y}{x}\right]$) alors on à $y = nx + y'$, *on* à $|y'| < x/2$. Et y' minimale pour toute factorisation de y par x.
De même on soit m le diviseur idéal de x par y' ($m = \left[\frac{x}{y'}\right]$) alors on à la factorisation de x par y', $x = my' + x'$ *et on a* $|x'| < |y'|/2$ et x' est minimal.

De ces deux égalités on déduit $x = my' + x' \rightarrow\ x = m(y - nx) + x' \rightarrow x' = x - m(y - nx) \rightarrow x' = (mn + 1)x - my$

On pose $\boldsymbol{t = (mn + 1)x - my\ \ ou\ \ n = \left[\frac{y}{x}\right]}$, $\boldsymbol{m = \left[\frac{x}{y'}\right]\ \ et\ \ y' = y - nx}$

➔ Le procéder consiste simplement à supprimer de y tous les morceaux de taille x, le morceau y' restant (dont la taille est inférieur à x/2) est ensuite utiliser pour réduire à son tour x.

Y

X	X	X	X	Y'

X

Y'	Y'	Y'	Y'	**ε**

Reste à montrer que |t| est minimal.

Remarque :
A postériori, nous constatons que cette approche à elle seule suffit pour effectuer un calcul du PGCD, il suffit pour cela de partir de l'ensemble initial, de calculer ε comme si dessus, de le rajouter à l'ensemble initial auquel on retranche tous les multiples de ε et de recommencer le processus, jusqu'à ce que tous les nombres restant soient multiples de ε.

Corolaire 3.5.1 : (Test de co-primalité)
Soit $\boldsymbol{\varepsilon = Min\{t = (mn + 1)x - my\ \ ou\ \ n = \left[\frac{y}{x}\right]}$, $\boldsymbol{m = \left[\frac{x}{y'}\right]\ \ et\ \ y' = y - nx}$, $\forall (x, y) \in E\}$ ***construit à partir des niveaux d'erreur de X pour la structure naturelle d'ordre 1.***
Si ε est premier alors PGCD(X) = ε. en particulier si un des ε vaut 1 alors les xi sont premiers entre eux.

Démonstration :
$Pm|e\ \ et\ e\ premier \rightarrow Pm = e.$ CQFD

➔ Ce résultat nous fournis un critère de co-primalité dont le cout est très faible.

Cas du pgcd approximatif partiel

Dans le cas du pgcd approximatif partiel nous savons que le pgcd est un diviseur exact du plus petit des éléments de l'ensemble.

Propriété : Soit d, un δ-diviseur de x si d est un diviseur (exacte) de P alors d est un δ-diviseur de e = .

~~On en déduit que pour le Pgcd approximatif partiel le plus petit des niveaux d'erreur par la structure d'ordre 1 pourra servir de pivot~~

Cas du pgcd approximatif

Dans le cas générale du pgcd approximatif il est difficile (du fit de la non transitivité de la δ-divisibilité) de construire un tel élément, aussi nous contenterons nous d'utiliser le pivot basique p = min(xi).

DIVISEURS IDEAUX ET GENERATEUR IDEAL

Dans ce paragraphe, nous nous intéressons à la relation qui lie le taux d'erreur finale (issue des approximations par les multiples du générateur idéal) et l'ensemble des diviseurs idéaux.

Pour rappel si on considère l'ensemble $\{x_0, x_1, \dots, x_n\}$et une structure $Q = \{q_0, q_1, \dots, q_n\}$ compatible avec cet ensemble. Les diviseurs idéaux sont donnés par
$D = \{ d_0, d_1, \dots, d_n \mid d_i = \frac{x_i}{q_i} \}$ et les niveaux d'erreur associé sont donnés par
$E = \{e_0, e_1, \dots e_n \mid e_i = (x_i - d_i * q_i)\}$ et les taux d'erreur associé par
$\Delta = \left\{\delta_0, \delta_1, \dots \delta_n \mid \delta_i = \frac{e_i}{x_i} \right\}$.

Condition d'arrêt sur les taux d'erreur des diviseurs idéaux

Théorème 3.1 : ***Soit P le générateur idéal construit à partir de l'ensemble D des diviseurs idéaux, et δ le son taux d'erreur, alors δ >= δi pour tout δi pris dans Δ.***

Le taux d'erreur final induit par le générateur idéal de la structure Q est toujours supérieur ou égale à chacun des taux d'erreur individuel des diviseurs idéaux.

Démonstration :

Soit dj le diviseur idéal ayant le taux d'erreur maximale parmi les diviseurs idéaux. Alors deux cas possibles : soit le générateur idéal est égale à di et on à $\delta = \delta j$ et donc δ supérieur ou égal à tous les δi. Soit le générateur est différent de dj et dans ce cas le niveau d'erreur sur xj sera supérieur à δi (lemme 1.2) et donc supérieure à tous les autres. CQFD.

- ⇨ **Règle 1 : si le niveau d'erreur sur un diviseur idéal est supérieur au seuil de tolérance, il est inutile de poursuivre la construction pour la structure considérée, car son taux d'erreur final sera lui aussi supérieur au taux d'erreur maximal ciblé.**
- ⇨ **Par exemple dans le cadre de la recherche du pgcd classique, lorsque le générateur (qui est par construction le diviseur idéal du pivot) n'est pas un diviseur strict, l'erreur introduite sur le plus petit élément est déjà supérieur au seuil de tolérance qui est nul dans ce cadre.**

Niveau d'erreur et borne de l'ensemble des diviseurs idéaux

On rappelle que le générateur idéal médian (celui qui permet de minimiser le taux maximal d'erreur subit par les éléments de X : voir lemme 2.6) est donné par $\boldsymbol{G = \left(\frac{2\,(P_{max} * P_{min})}{P_{max} + P_{min}}\right)}$ ***ou*** $\boldsymbol{P_{max}}$ ***et*** $\boldsymbol{P_{min}}$ ***désignent respectivement les diviseurs idéaux minimaux et maximaux.***

Lorsque les points P_{mim} et P_{max} ne sont pas trop éloignés G se situe à peu près au milieu de l'intervalle [P_{mim} - P_{max}]. Les éléments de X associé à ces deux diviseurs idéaux sont ceux qui subissent l'ajustement le plus important de leur idéaux. Le taux d'erreur du générateur idéal final sera alors à peu près égale à $P_{mim}/(P_{min} - G)$ (ou du moins supérieur à $\frac{Pmin}{Pmin + \frac{Pmin - Pmax}{2}}$.)

➔ On constate donc que le niveau d'erreur final sera donc d'autant plus important que la distance qui sépare P_{mim} et P_{max} sera grande.

~~Théorème 3.2 : le générateur idéal final induit un taux d'erreur d'autant plus faible que les diviseurs idéaux maximal et minimal P_{min} et P_{max} sont proches.~~

~~Plus précisément si il existe p1, p2 tel que p1-p2 > d*… alors le seuil de compatibilité du générateur idéal est supérieur ou égal à d/2…~~

~~Démonstration :~~

CL :

➔ La capacité d'une structure à produire un générateur idéal de niveau d'erreur faible, est directement liée à la proximité des diviseurs idéaux maximaux et minimaux qui lui sont rattaché et donc à la concentration de l'ensemble des diviseurs idéaux.

➔ Seuls les changements affectant les niveaux min et max de l'intervalle idéal sont susceptibles de réduire le niveau d'erreur. Cette réduction est alors proportionnelle à la compression de l'intervalle qui en résulte.

➔ Le niveau d'erreur est minimale lorsque tous les idéaux sont identiques. Si en plus ces diviseurs idéaux sont des diviseurs au sens stricte du terme, alors le générateur qui en résulte et un diviseur commun. (on à un Pgcd classique)

STRUCTURE ET DIVISEURS IDEAUX

Dans la partie précédente, nous avons étudié le rapport entre l'ensemble des diviseurs idéaux et le générateur idéal qui en résulte. Cette étude nous a permis de dégagé les

principale caractéristique de cet ensemble et la façon dont ils sont liées à la qualité du générateur idéal qui en résulte.

Dans la recherche du diviseur ou du générateur idéal, seul compte le couple (ensemble – structure) X-Q qui associe à chacune des données initiales, un quotient. Le diviseur qui en résulte est indépendant de toutes les autres considérations, et même dû procéder ayant mené à cette construction (y compris du générateur initial de la structure).

Pour comprendre l'émergence du générateur idéal, et donc du pgcd, il est donc indispensable de comprendre la dynamique de la structure Q, le rapport qu'elle entretient avec l'ensemble des diviseurs et la façon dont elle évolue et change au cours du processus de construction du pgcd.

➔ ***Il en résulte donc que tout changement dans le procéder de construction n'impactant pas la structure quotient Q, n'aura aucun impact sur le générateur idéal qui en résulte.***

Commençons par nous intéresser aux altérations partielles d'une structure et à ses conséquences sur l'ensemble des diviseurs idéaux et sur le générateur idéal qui en résulte.

Corolaire 3.1 (Union de structure)

Considérons un ensemble X $= \{x_0, x_1, \dots, x_n\}$ ***et un ensemble X'*** $=$ ***X U*** $\{x_{n+1}\}$***.***

Soit Q $= \{q_0, q_1, \dots, q_n\}$ ***une structure compatible avec X, et Q'*** $= \{q_0, q_1, \dots, q_n, q_{n+1}\}$ ***une structure compatible avec X' vérifiant Q'*** $=$ ***Q U*** $\{ q_{n+1} \}$

Alors le taux d'erreur δ du générateur idéal de Q est inférieur ou égal au taux d'erreur δ' du générateur idéal de Q'.

En effet deux cas de figure se présentent :

Soit le diviseur idéal associé à l'élément rajouté appartient aux à l'intervalle [P_{mim} - P_{max}] des diviseurs associés à la structure Q initiale. Auquel cas le taux d'erreur n'est pas affecté (le générateur idéal qui ne dépendent que des bornes de l'ensemble des diviseurs idéaux reste d'ailleurs inchangé). Se sera par exemple le cas l'lorsqu'on rajoute à l'ensemble précédent un diviseur d'un de ses éléments.

Soit le diviseur idéal associé au quotient rajouté se situe à l'extérieur de cet ensemble auquel cas le taux d'erreur associé au nouveau générateur est supérieure à celui du générateur précédent car les bornes de l'ensemble des générateurs sont désormais plus éloigné.

Corolaire 3.2 (partie stable d'une structure)

On considéré une structure Q $= \{q_0, q_1, \dots, q_n\}$***, ainsi qu'une structure Q'*** $= \{q_0, q_1, \dots q'_i, \dots, q_n\}$***, construit en modifiant un (ou plusieurs) éléments de la structure***

initiale. On définit l'extrait $Qs = \{q_0, q_1, \dots q_{i-1}, q_{i+1} \dots q_n\}$, composées des éléments de Q qui n'ont pas été modifiés (Qs est dite partie stable de Q). Alors le taux d'erreur de Q' est supérieure au taux d'erreur de la partie stable.

Dem : la démonstration se déduit aisément du fait que, la non modification du quotient entraine un maintien du diviseur idéal associé.

- ➔ Une modification partielle de structure est porteuse d'un taux d'erreur implicite non réductible par ladite modification, c'est le taux d'erreur associé à la partie stable de la structure.
- ➔ Toute évolution de la structure n'impactant pas les quotients associés aux diviseurs idéaux maximaux et minimaux, ne peut entrainer une réduction du niveau d'erreur.

La dynamique de la structure elle dirigé par le triplet (X-G-E)

EXPLORATION HORIZONTALE DES STRUCTURES

Dans ce paragraphe nous analysons le rapport entre la structure naturelle et les structure secondaires de même ordre. Le but est d'anticiper les taux d'erreurs qu'induirons les structures secondaires, et donc d'éviter de les construire quand on sait qu'elles ne permettront pas de réduire le niveau d'erreur de la structure naturelles.

Ce qui est important pour cette analyse c'est le fait que les structures secondaires ne constituent que des altérations partielles de la structure naturelles.

Soit $X = \{x_0, x_1, \dots, x_n\}$ et $Q = \{q_0, q_1, \dots, q_n\}$ la structure naturelle d'ordre n, $P = \{P_0, P_1, \dots, P_n\}$ l'ensemble des idéaux et $R^+ = \{R_0^+, R_1^+, \dots, R_N^+\}$ et $R^- = \{R_0^-, R_1^-, \dots, R_N^-\}$ les points de rupture à droite et à gauche.

Restructuration des idéaux maximal et minimal

On rappelle que le générateur idéal de la structure est donné par :

$\boldsymbol{G} = \left(\frac{\boldsymbol{2\,(P_{max} * P_{min})}}{\boldsymbol{P_{max} + P_{min}}}\right)$ Ou Pmin = Min(P), Pmax = Max(P).

Ou $\boldsymbol{G} = \boldsymbol{P_{min}} + \boldsymbol{D} * \left(\frac{\boldsymbol{\lambda_{min}}}{\boldsymbol{\lambda_{max} + \lambda_{min}}}\right)$ D = |Pmax –Pmin| λmin = Min{λi = qi/xi} et λmax = Max{λi = qi/xi}

Et le niveau d'erreur induit est $\delta = 1 - \left(\left[\frac{x}{G}\right] * \left(\frac{x}{G}\right)\right)$ ou x est l'élément dont Pmin est le diviseur idéal.

RQ1 : Seul les quantités Pmin et Pmax sont pris en compte dans le calcul du générateur idéal d'une structure.

- ➔ Toutes structures secondaires ne modifiant pas au moins un des quotients associés aux diviseurs Pmin ou Pmax aura le même idéal que la structure secondaire et donc induira le même niveau d'erreur.
- ➔ Soient $R_m^+, R_M^+, R_m^-, R_M^-$ les points de ruptures associés aux quotients des diviseurs Pmin et Pmax. Seul les éléments de G+ inférieur à $min(R_m^+, R_M^+)$ et ceux de G- supérieur à $max(R_m^-, R_M^-)$ peuvent conduire à des générateurs induisant des niveaux d'erreurs inférieur à celui de la structure naturelle.

Conservation de l'ordre de la structure

On réordonne les ensembles R^+et R^- $tel\ que\ R_1^+ < \ldots < R_N^+\ et\ R_1^- < \ldots < R_N^-$:

RQ1 :

Nous avons déjà montré que si le point de rupture d'un quotient se situait au-delà de celui du pivot (i.e. de R_0) alors la structure induite n'était pas une structure de même ordre que la structure naturelle.

- ➔ Il n'est pas nécessaire d'analyser les structures générée par des éléments se situant à l'extérieur des borne de la structure (i.e. n'appartenant pas à $[R_0^-, R_0^+]$), les point dont la rupture intervient après celle de Q0 peuvent donc être considéré comme des points fixes des structures d'ordre n.

Ordre point fixe et seuil d'erreur minimale

Le fait que tous les points de rupture se situant au-delà du point de rupture du pivot (qui garantit la conservation de l'ordre n pour la structure) entraine l'existence d'idéaux fixes partagés par toutes les structures d'ordre n.

Soit $P^n = \{\}$ l'ensemble des diviseurs idéaux des points de X pour une structure d'ordre N. soit P^{fn} l'ensemble des diviseurs idéaux fixe partagé par toutes les structure d'ordre n.

1. P^{fn} est non vide : Pn = xo/n le diviseur idéal du pivot (et générateur de la structure) appartient à P^{fn}. par définition le quotient du pivot est fixe et égale à l'ordre pour toutes les structures de même ordre, il s'en suit que le diviseur idéal est fixe lui aussi.
2. Soit Pi le diviseur idéal du ieme élément de X : $P_i \in P^{fm} \leftrightarrow P_i - a_i^- < P_0 - a_0^-\ ou\ P_i - a_i^+ > P_0 - a_0^+$ en d'autre terme si la restructuration d'un point ne survient qu'après celle du Pivot alors le point est fixe pour l'ordre et son diviseur idéal aussi.
3. On en déduit la distance minimale au point fixe Pn : $A_{fn} = Max(P_n - P_i \mid P_i \in P^{fm}\ et\ P_i < P_n)$ et $B_{fn} = Max(P_i - P_n \mid P_i \in P^{fm}\ et\ P_i > P_n)$

On déduit le niveau d'erreur minimale sur le pivot pour toutes structures d'ordre n. $\delta_{\mathrm{fn}} = \frac{\varepsilon_0}{\mathrm{n}}\left(\mathbf{1} + \frac{\mathbf{1}}{\mathbf{2P_n}+(B_{fn}-A_{fn})}\left((B_{fn}-A_{fn}) - \frac{\mathbf{2}*A_{fn}*B_{fn}}{\boldsymbol{P_n}}\right)\right)$

➔ Si ce seuil est supérieur au seuil cible, il n'est pas utile de poursuivre l'exploration des structures de cet ordre.

4. Par ailleurs étant donné que la restructuration entraine soit un déplacement vers la gauche soit un déplacement vers la droite des points restructurés (tous dans le même sens) la restructuration ne pourra améliorer qu'au plus une des valeurs A ou B (les distance maximale à l'idéale) on en déduit les seuils d'erreur minimaux des structure secondaire d'ordre n

$$\delta_{fn}^{+} = \frac{\varepsilon_0}{\mathrm{n}}\left(1 + \frac{1}{2\mathrm{P_n}+(B_{fn}-A)}\left((B_{fn}-A) - \frac{2*A*B_{fn}}{\mathrm{P_n}}\right)\right)$$

$$\delta_{fn}^{-} = \frac{\varepsilon_0}{\mathrm{n}}\left(1 + \frac{1}{2\mathrm{P_n}+(B-A_{fn})}\left((B-A_{fn}) - \frac{2*A_{fn}*B}{\mathrm{P_n}}\right)\right)$$

➔ Si ce seuil est supérieur au seuil cible, il n'est pas utile de poursuivre l'exploration des structures secondaire correspondantes.

Distance maximale aux points fixes

Toutes les structures d'un ordre n donné partagent au moins un point commun, le quotient associé au pivot, c'est donc un point fixe. Aussi dans l'ensemble des diviseurs idéaux, figurera toujours le diviseur idéal $P^n := x_0/n$. Notons que la distance qui sépare les idéaux les plus éloigné peut alors être ré-exprimer en fonction de leur distance respective à ce point fixe : $D := |Pmax-Pmin| = |Pmax - P^n| + |P^n - Pmin|$

De plus nous savons que choisir un générateur dans R_n^+ entraine une réduction ou une conservation de idéaux initiaux, et qu'on à l'effet contraire en en choisissant un dans R_n^-.

Si donc on choisit un générateur $G < P^n$ (par exemple G pris dans R^-)

- Alors tous les idéaux inférieurs a P^n seront conserver ou dégrader (dans le sens ou la distance qui les sépare de P sera augmenter),
- la distance $A = |P^n - Pmin|$ sera conserver au augmentera.
- La distance B ne sera réduite que si on restructure le point Pmax.

De même si on choisit un générateur $G > P^n$ (par exemple G pris dans R^+)

- Alors tous les idéaux inférieurs à P^n seront conserver ou dégrader (dans le sens ou la distance qui les sépare de P sera augmenter).

- La distance B= |Pmax - P^n| sera conserver au augmentera.

Algorithme d'Exploration des structures secondaires

Exclusions des cas triviaux

Initialisation

A chacun des éléments de X, et pour le générateur Pn, associé les quantités

- La structure : $q_i := \frac{x_i}{P_n}$; $\bar{Q} := \left[\frac{x_i}{P_n}\right]$; $\varepsilon_i = q_i - \overline{q_i}$ ou [] est l'arrondi à l'entier le plus proche.
- En déduire l'ordre de restructuration $o_i = \frac{\varepsilon_i}{q_i}$ o_i peut être positif ou négatif, et plus on est proche de l'extrémité pus vite se fera la restructuration(Q -> Q+1 / Q-1) à droite ($\varepsilon > 0$) ou à gauche ($\varepsilon < 0$) .
- Ordonner les éléments de X suivant oi (croissant, puis décroissant dans un second temps)

…

~~EXPLORATION VERTICALE DES STRUCTURES~~

~~Dans cette partie nous ne nous intéressons plus aux structures de même ordre mais à la limitation du nombre de structure d'ordre successifs à explorerez. Le but étant à partir de données disponible pour un ordre n de comprendre dans qu'elle mesure il peut être inutile d'explorer un certain nombre de structures d'ordre suivant. Ici c'est le nombre d'ordre à explorer qu'on veut réduire au minimum, et transformer l'itération sur les structures d'ordre successif (n, puis n+1) en une itération qui passe de l'ordre n à l'ordre n + a (a<=1) , nous devons ici calculer a.~~

~~Soit $X = \{x_0, x_1, \dots, x_n\}$ et la suite des générateurs donnés par $Pn = \{[x0/i], i = 1 \dots n\}$, on veut comprendre l'évolution des différentes décompositions $x = pq + e$ lorsque l'ordre et donc le quotient q augmente.~~

~~Pour ce faire nous commençons par la décomposition initial (ordre 1, générateur x0. Elle est donné par.~~

~~$\forall x \in X, \exists \alpha i \in N \; et \; x'i \in Z \mid xi = \alpha i * x0 + x''i \; avec \; x''i < \frac{x0}{2}$ on note A l'ensemble des αi~~

~~$x'i = \alpha i * x0$ Est appelé partie stable de x. Et l'ensemble X(1) = $\{x'i, \mid x'i = \alpha i * x0\}$ est appelé partie stable d'ordre 1 de X. de même $ri = xi - x'i$ este appelé partie résiduelle~~

de X et $R(1) = \{r'i | r'i = xi - x'i\}$ est appelé partie résiduelle d'ordre 1 de X. Noter que les résidus ne sont pas forcément positif.

De la même façon, nous pouvons décomposer les structures Q = { αi } en partie stable et instable. Nous nous intéressons à la façon dont l'augmentation de l'ordre de fractionnement impacte la partie stable et instable de X ainsi que sa structure. De façon très intuitive on constate que la partie stable de la structure qui provient de la décomposition de la partie stable de X aura une évolution linéaire. On passera tout simplement de Q(1) = { αi } à Q(n) = { $n * \alpha i$ } . tandis que les éléments de la partie instables ne serons affectés à la structure finale qu'à partir du moment où le fractionnement sera suffisamment important pour que le générateur soit du même ordre de grandeur que le résidu en question. Tan que l'ordre ne sera pas suffisamment grand pour que ces résidus soient intégrés à la structure finale, ils constitueront une erreur irréductible d'approximation. Si le taux d'erreur implicite pour ces erreur irréductible est supérieur au taux d'erreur cible, il sera inutile d'étudier plus en détail les ordre de fractionnement ne permettant pas la prise en compte de ces résidus dans la structure Q. il faudra donc passer directement à l'ordre de fractionnement permettant l'intégration du plus petit résidu dont le taux d'erreur implicite est supérieur au niveau cible.

D'autre part l'intégration de ces résidus dans la structure stable, ne sera pas totale (à moins que leur valeur soit exactement égale à une fraction du pivot) aussi provoquera-t-elle une nouvelle séparation de la structure en partie stable et en partie instable à partir du pivot ayant permis l'intégration partielle du résidu (et non plus à partir de x0), cette nouvelle dissection sera à son tour traité comme la précédente.

Ce procédé permettant d'éviter une itération sur des ordres de fractionnement non pertinent sera utilisé jusqu'à ce que les résidus représentent un niveau d'erreur implicite inférieur au niveau cible.

Nous allons maintenant présenter de façon détaillée cette idée intuitive.

Considérons maintenant le résidu x'' de cette factorisation initiale.

On considère la décomposition d'ordre 1 de X :

$\forall x \in X, \exists \alpha i \in N \; et \; x'i \in Z \mid xi = \alpha i * x0 + x''i \; avec \; x''i < \frac{x0}{2}$ on note Qs(1) l'ensemble des αi

$\forall \; x''i < \frac{x0}{2} \; \exists mi \in N \mid x''i = Pm + ri \; avec \; |ri| < \frac{pm}{2}$, m est appelé seuil de resorbtion initial de xi.

En effet on montre aisément que tant que n > m la meilleurs approximation de x par Pn est n*Pn

Si on considère un ordre n < mi alors on a :

La structure Q de X se décompose alors à l'ordre n en deux sous structures. Q = Qs + Qi

Qs la structure stable est issue de la décomposition de la partie $x'i = \alpha i * x0$ des xi et on à pour un générateur $\forall Pn = \left[\frac{x0}{n}\right]$ $Qs = \{qi = \alpha i * n,\}$ tandis que le résidu $x''i$ complètera la structure en $\forall Pn = \left[\frac{x0}{n}\right]$ $Qi = \left\{qi = \lambda i * n,\ \lambda i = \frac{x''i}{x0}\right\}$

Etudions maintenant le coefficient λ.i

Ainsi on à $qi = \lambda i * Pn = n * \frac{x''i}{x0} \rightarrow$ qi < 1 si $n * \frac{x''i}{x0} < 1$

Or $n * \frac{x''i}{x0} = \frac{n*Pm+nRi}{nPn+\mathcal{E}n}$ $et\ |\mathcal{E}n| < n \rightarrow qin < \frac{Pm+ri}{Pn+1} < \frac{Pm}{Pn}$ donc si m < n -> qin = 0.

$qin = \frac{n*Pm+nRi}{nPn+\mathcal{E}n}$ $\min\{|\mathcal{E}n|\} = \frac{n}{2}$ $et \max\{|ri|\} = \frac{Pm}{2} \rightarrow qin > \frac{3Pm}{2(Pn-1)}$

$\boldsymbol{qin > 1 \rightarrow 3Pm > 2(Pn-1) \rightarrow Pn < \frac{3Pm}{2} + 1 \rightarrow \approx > n > \frac{2x0}{3Pm+1}}$

$x0 = nPn + \mathcal{E}n$ $et\ x = \lambda x0 +$

Cas du Pgcd Exact.

Commençons par nous intéresser au cas du PGCD exact.
On considéré notre ensemble $X := \{xi, i = 1,,n\}$. Soit P son pivot (P := Min {xi}). On définit P^n := [P/n] le n^{ieme} diviseur de P (arrondi à l'entier le plus proche) i.e. qu'il existe a entier, tel que $P = n\ P^n + a$.

Soit m l'ordre du PGCD(X).

Lemme 3.3 :
PGCD(X) := P^m divise xi pour tout xi dans X, en particulier P^m |pivot -> m| pivot.
➔ ***Tout ordre qui n'est pas un diviseur du pivot peut être exclu de la recherche de PGCD.***

La démonstration est évidente.

Lemme 3.4:
m < Racine(Pivot) sinon PGCD = 1.
➔ ***La recherche du PGCD n'a pas besoin d'aller jusqu'à un ordre égal au pivot, si elle franchit la racine du pivot, on peut conclure que le PGCD est égal à 1.***

La démonstration découle directement de la décomposition en facteur premier du Pivot, le plus grand de ces facteurs devant être inférieur à sa racine.

Cas des PGCD approchés.

Fractionnement

Dans cette partie, nous nous penchons sur l'analyse de l'évolution d'une structure lorsqu'on en fractionne le générateur initial. Aussi pouvons-nous déduire de l'étude d'une structure générée par P, les informations sur toutes les structures donc les générateurs sont de la forme P/n. (notons que ceci réduira notre recherche du PGCD à la construction et à l'étude d'un nombre très faible de structure).

Notre but est ici d'analyser les structures dont l'ordre est un multiple de l'ordre de la structure initiale. Si donc le structure initiale à comme générateur P, la structure d'ordre n'aura comme générateur Pn := [P/n].

Comme nous l'avons déjà montré dans la première partie de ce chapitre, les modifications qui feront évoluer notre générateur idéal, sont les modifications affectant la structure (seul une modification de structure entraine une modification de l'idéal).

Comment donc est ce que le fractionnement du générateur initiale affect il la structure générée ?

Lorsqu'on passe de P à Pn, contrairement au cas du PGCD on ne s'intéresse pas uniquement au P, tel que P = nPn, mais plus généralement aux factorisations P = nPn +b,

Le passage du générateur P au générateur P, va donc en plus du fractionnement de P en n morceaux, entrainer un ajustement des niveaux d'erreur d'une quantité a* qi proportionnel à leur quotient respectif. Ce qui va avoir deux conséquences au niveau de la structure d'abord les quotients passerons de q à n*q, puis il seront ajusté d'une quantité a = [(e +aq)/P^n], et donc q^n := n*q + a. ce qui provoquera donc en plus du fractionnement du diviseur idéal, un léger ajustement de ce dernier. C'est ce double effet que nous allons maintenant analyser.

X

P	P	P	P	E

P'	A	P'	A	P'	A	P'	A	E

P'	P'	P'	P'	P'	A	A	A	E

P'	P'	P'	P'	P'	

Théorème 2 (de fractionnement optimal) :

Soit $x = pq + e$ et $x = p'q' + e'$ Si $p' = \left[\frac{p}{n}\right] \rightarrow p = np' + a$, alors :

Pour résorber une erreur égale à e il faut fixer $n = \left[\frac{p}{2e}\right]$, l'erreur pourra alors être totalement résorbée si 2e divise p et n divise p. dans le cas où on ne réussit pas à obtenir un n vérifiant ces propriétés, on a :

1. **Si $n \mid p$ $(a = 0)$ et $n < \frac{p}{e} \rightarrow e' = e$**
   i.e. le passage d'un diviseur p à une fraction de ce diviseur n'a aucun impact sur le niveau d'erreur si le fractionnement est inférieur au rapport du diviseur et de l'erreur (p/e).

2. **$n \mid p$ $(a = 0)$ et $n > \frac{p}{e} \rightarrow e' = e - \frac{p}{n}$**
   i.e. le passage d'un diviseur p à une fraction de ce diviseur provoque une réduction du niveau d'erreur de p/n.

3. **Si n *ne divise pas* p $(a \neq 0)$ $n < \frac{p}{e} \rightarrow e' = e + aq$**
   i.e. le simple fait d'utiliser des diviseurs inexacts de p provoque un ajustement proportionnel au quotient et au résidu de la division, elle est positive ou négative suivant le sens du résidu de division de p par n.
   Si donc l'erreur est un multiple du quotient alors un diviseur bien choisi peu permettre d'éliminer l'erreur ou de le réduire en proportion du quotient (choisir n tel que $n \equiv a\ Mod(P)$).

4. **Si n *ne divise pas* p $(a \neq 0)$ $n > \frac{p}{e} \rightarrow e' = e - \left[\frac{p}{n}\right] + aq$**
   i.e. on a un double effet sur le niveau d'erreur, d'une par elle est réduite de p/n, d'autre part elle est ajusté de aq, suivant le sens du reste de la division de p par n. Le résultat est de ce fait un peu plus compliqué à analyser.

Si on note X l'ensemble de points, Q la structure compatible avec X et générée par P, alors pour chacun de xi dans X, on a la factorisation **$x_i = Pq_i + e_i$ et $\delta_i = \frac{e_i}{x_i}$** :

Le théorème de fractionnement illustre l'évolution du niveau d'erreur lorsque le générateur initial est remplacé par ses fractions successives. Etant donné que dans le cadre des ensembles, le diviseur est commun (le générateur initial) l'impact choisi affectera tous les e_i, il est donc crucial de bien choisir le fractionnement si on veut une réduction globale du niveau d'erreur. Comme nous l'avons signalé précédemment il est important de ramener les taux d'erreur à des niveaux comparable (plus ils sont proche meilleurs sera notre générateur idéal).

Nous avons montré que la résorption de l'erreur dépend de l'indice de résorption sonné par $n_t = \frac{p}{2e_t}$ et avons étudié le comportement de l'erreur en fonction du rapport entre n et cet indice.
En plus de la résorption nous savons par le même théorème que nous pouvons ajuster l'erreur en proportion du quotient. On s'intéressera donc aux rapports $a_t = \left[\frac{e_t}{q_t}\right]$. En effet en choisissant n tel que $p \equiv a[n]$ on pourra ajuster toutes les erreurs e_i d'une quantité aq_i. et pour ceux dont l'indice de résorption est inférieur à n on aura un double effet : d'abord une résorption partielle de l'erreur d'une quantité P/n puis un ajustement de aq_i. Par ailleurs notons que si n est choisi comme un diviseur de P, alors l'ajustement est nul et l'effet de la division par n du générateur initial est nul sauf pour les éléments dont l'indice de résorption est inférieur à n ; ces derniers voient leur erreur réduite de p/n.

Le choix du n optimale doit donc se faire en fonction de tous ces éléments en s'appuyant sur le théorème de fractionnement.
La méthode d'action consiste en : identifier les éléments qui sont loin du consensus et choisir le moyen d'action qui permet à ces dernier de se rapprocher du consensus sans trop impacter les autres éléments.

Comme dans le cas de l'exploration horizontale, nous allons nous intéresser aux points fixes (qu'on peut exclure à priori de notre optimisation car nous savons que leur taux d'erreur ne variera pas)

Soit x = pq+e, nous dirons que x est fixe à l'ordre n si le quotient q' qui lui est associé à l'ordre n est q' = nq.

X est fixe pour tout n tel que |e|< p/2n.

Si x est un point fixe à l'ordre n, alors en vérifie e - (n/2)*q <en< e + (n/2)*q.

Pour les points fixe, le rapport e/q est donc très important il nous dit pour quel ajustement de p (résidu de division par n) l'impact sur la structure sera capable de consommer l'erreur.

Rq : Etant donné que les quotients sont positifs et que l'ajustement est le même pour tous les xi, on ne pourra réduire que des erreurs qui sont de signe opposé à l'ajustement, ceux de même signe serons accentués….

CONCLUSION

PARTIE2

FACTORISATION APPROXIMATIVE ET PGCD APPROXIMATIF

Chapitre 5

Factorisation Approximative

``The obvious mathematical breakthrough would be development of an easy way to factor large prime numbers."
-Bill Gates, The Road Ahead

Dans ce chapitre, nous nous intéressons à la question millénaire de la factorisation. Nous présentons une nouvelle approche basée sur l'observation du processus dit de transfert optimal d'information entre les facteurs. Nous présentons une série d'algorithme pour répondre à cette question. Le premier est basé sur la quantité a d'information à transférer du premier vers le second facteur et son cout est fonction de la distance qui sépare le facteur le plus proche de la racine du nombre à factoriser de cette racine en question. Puis nous présentons une amélioration de cette approche qui plutôt qu'envisager la factorisation en terme de quantité d'information à transférer du premier vers le second facteur considère le cout k de ce transfert d'information, or ce cout est le même pour des plages entière d'information ce qui permet de réduire de façon notable la complexité de la factorisation. Ces deux algorithmes se situent dans la classe des « spécial purpose algorithm ». Nous proposons ensuite un mécanisme dit de rééquilibrage des facteurs qui nous permet d'étendre ces algorithmes pour en faire des « general purpose », l'idée ici est de créer artificiellement un facteur proche de la racine afin de profiter de l'efficacité des algorithmes précédent dans la recherche des facteurs présentant cette caractéristique. Nous terminons par la construction d'un indice dit de Kamto qui nous permet de mesurer de façon assez précise la difficulté de factoriser un nombre et donc pourra guider le choix des nombres à utiliser dans le cadre de la cryptographie afin d'éviter une factorisation rapide.

FACTORISATION

Dans ce premier paragraphe nous présentons différentes formulations du problème de factorisation.

Probleme1 (formulation fonctionnel) :

Soit N un entier, trouver un entier D vérifiant $1 < D < N$ tel que D divise N.

Problème 2 (Version décisionnelle) :

On considéré un entier N, soit M un entier vérifiant $1 \leq B \leq N$, Existe-t-il un entier D vérifiant $1 < D < B$ tel que D divise N.

Dans ce chapitre nous traiterons d'une généralisation de cette formulation du problème sous la forme : étant donné deux entier B1, B2 compris entre 1 et N, existe-t-il un diviseur D de N vérifiant $B_1 < D < B_2$.

Problème 3 (cryptographie) :

Soit N un entier, trouver U et V entier tel que N = U*V.

Factoriser un entier N, c'est trouver deux entiers U et V tel que N = U*V.

La factorisation de N sous la forme U*V correspond aussi au problème de cryptographie dans lesquels N est souvent construit en multipliant deux grands nombres premiers.

Notons que si N n'est pas un produit de deux nombres premiers, alors cette factorisation de N sous la forme U*V n'est pas unique. Dans cet article nous nous intéresserons uniquement à la recherche du couple U,V pour lequel la distance entre U et V est la plus faible.

Problème 4 (factorisation approximative pour calcul de pgcd approximatif):

Dans le cadre de la δ-arithmétique, la question de la factorisation se pose quelque peu différemment. En effet plutôt que de rechercher les couples (U,V) tel que N = U*V, on s'intéresse ou couple (U,V) qui minimisent l'erreur d'approximation de N par le produit U*V. en d'autre terme on recherche(U, V) tel que $E = N - U * V$ soit minimal, ($N = U * V + E$). C'est cette formulation du problème qui a motivé notre recherche initiale.

Ainsi le problème de la factorisation approximative peut être formulé comme suit :

Soit N un entier, et $r := \lceil \sqrt{N} \rceil$ ***trouver U et V tel que :***

$$U \leq r \text{ et } V \geq r \ , \ N = U * V + E, E \leq \frac{P}{2}$$

Minimiser E sous la contrainte P > M donné ou Maximiser P sous la contrainte E inférieur à Ɛ donné.

Le principal avantage de cette formulation est de présenter la factorisation comme un processus continue, ou le niveau d'erreur de factorisation est réduit progressivement et qui trouve son aboutissement au moment où ce niveau d'erreur est nul.

C'est ce problème qui nous intéressera dans ce chapitre avec une attention particulière pour la contrainte Ɛ = 0 qui équivaut au problème classique de la factorisation.

Problème 5 : recherche de facteurs premiers.

Nous savons depuis Euclide que tout nombre peut s'écrire comme produit de facteurs premiers. Une formulation classique du problème de factorisation consiste en la recherche de cette décomposition unique.

Nous ne traiterons pas de ce problème en particulier, notons toutefois que pour obtenir la décomposition en facteurs premiers de N il suffit de réitérer le processus de factorisation pour les nombre U et V, jusqu'à ce que les facteurs soit non factorisable i.e. premiers.

FACTORISATION PAR TRANSFERT D'INFORMATION ENTRE FACTEURS

Dans ce premier chapitre nous présentons notre algorithme de base, il est centré sur le transfert d'une quantité d'information « a » du premier (et plus petit) facteur de N vers le seconds facteur.

Pour comprendre ce processus considérons un quasi factorisation de N sous la forme $N = P * Q + E$ *on souhaite transférer une quantité d'information « a » du facteur P (on veut passer de P à P-a) vers le facteur Q. la question qui se pose est alors qu'elle est la quantité d'information b à rajouter au facteur Q pour compenser la quantité « a » supprimé de P sachant que l'on souhaite maintenir à tout moment le niveau d'erreur au minimum possible. En d'autre terme on cherche la quantité « b » qu'il faut rajouter à Q pour passer de la factorisation* $N = P * Q + E$ *à la factorisation* $N = (P - a) * (Q + b) + E'$ *avec E qui doit rester le plus faible possible. Et en particulier qu'elle quantité d'information « a » pourra être totalement compensé par un ajout d'information « b » de façon à ce que le niveau d'erreur soit nul. C'est à ces questions que nous allons répondre.*

Soit N un entier, on pose $P = \lfloor \sqrt{N} \rfloor$, $E = N - P^2$ et $R = \lfloor \sqrt{P} \rfloor$

On peut donc écrire N sous la forme $N = P * P + E$

Si N n'est pas premier, alors il existe (U, V) tel que $N = U * V$ et $U < P$ et $V > P$. dit autrement, il existe un couple (a, b) tel que $N = (P - a) * (P + b)$.

Les valeurs a et b représente alors les ajustements à apporter à P (la racine de N) pour trouver une de N. Factoriser N revient alors à trouver ce couple (a,b). Notons par ailleurs que pour factoriser N il n'est pas indispensable de trouver a et b, la seule connaissance de a, suffit pour déduire la valeur de b (si $(P - a) \mid N$ alors $b = \frac{N}{P-a} - P$).

Transfert d'information

On souhaite retrouver la factorisation de N sous la forme $N = (P-a)*(P+b)$. En parcourant les différentes valeurs de a qui sont comprise entre 1 et P. lorsque a ne correspond pas à la valeur cible la factorisation n'est pas exact, on à un niveau d'erreur E et donc une écriture de la forme $N = (P-a)*(P+b)+E \ \forall a \in [0,P]$. La factorisation est obtenue lorsque E = 0.

Nous voulons ici étudier la façon dont l'information circule entre les deux facteurs $(P-a)$, $(P+b)$ et E. En d'autre terme lorsque la valeur de a est modifié (on passe par exemple de a à a-1, comment sont ajustées les valeurs de b et le niveau d'erreur ?

Nous souhaitons donc partir du point de départ a = b =0 et donc $N = P^2 + E$ et retrouver les valeurs de a et b qui permettent d'obtenir $N = (P-a)*(P+b)$.

On en déduit l'expression de b sous la forme $b = \frac{e-aP}{P-a}$. Notons que cette valeur de b qui permet de compenser totalement la suppression de « a » du premier facteur n'est pas toujours entière. Si donc on substitue à b sa partie entière $b = \left[\frac{e-aP}{P-a}\right]$ (ou sa valeur arrondi) on introduit une erreur de factorisation E fonction de la différence entre cette valeur exact et l'arrondi : $E = (P-a)*\left(\frac{e-aP}{P-a} - \left[\frac{e-aP}{P-a}\right]\right)$ et l'erreur est nul (et donc la factorisation exacte) ssi b est entier i.e. $b = \left[\frac{e-aP}{P-a}\right]$ (Notons qu'une autre technique de factorisation pourrait se baser sur l'étude de cette seule expression afin de savoir pour qu'elle valeur de a, cette expression donne un entier i.e. pour qu'elle valeur de a $(e-ap)$ et un multiple de $(p-a)$)

On en déduit une factorisation de N comme fonction de a sous la forme

$$N = f(a) = (P-a)\left(P+\left[\frac{e-aP}{P-a}\right]\right)$$

Démonstration

$N = P^2 + E \ et \ N = (P-a)*(P+b) \rightarrow P^2 + E = P^2 - aP + bP - ab \rightarrow \ E - aP = bP - ab$

On en déduit $b = \frac{E-aP}{P-a}$ d'où la factorisation.

Nous pouvons donc déduire de cette écriture un algorithme naïf de factorisation de N.

Poser

P = Arrondi(racine(N))

E = N – P^2

Pour a = 1 à a = P faire

U := P – a

V :=Arrondi (N/U)

*E : = N – U*V*

Si E == 0 alors retourner le couple (U, V)

Sinon poser a := a+ 1 et recommencer.

Fin faire

Ou

Poser

P = Arrondi(racine(N))

E = N – P^2

Pour a = 1 à a = P faire

*b = Arrondi((E-a*P)/(P-a))*

U := P – a

V :=P+b

*E : = N – U*V*

Si E == 0 alors retourner le couple (U, V)

Sinon poser a := a+ 1 et recommencer.

Fin faire

Cette approche certes peut efficace à l'avantage de factoriser les nombres en initiant la recherche sur les grands facteurs plutôt que sur les petits. Elle est d'autant plus efficace qu'il existe un diviseur de N proche de sa racine P (ou que la quantité d'information à transférer est faible). Par ailleurs, notons que plus le diviseur recherché n'est pas contraint d'être premier, s'il existe une combinaison quelconque de facteurs premiers de N proche de P c'est cette combinaison qui sera généré par l'algorithme comme valeur de U.

FACTORISATION PAR COUT DU TRANFERT D'INFORMATION ENTRE FACTEURS

Dans la partie précédente nous avons construit la factorisation autour de la quantité d'information à transférer du premier vers le second facteur de N. ici nous ne nous intéressons plus directement à la quantité d'information à transférer d'un facteur à l'autre mais au « coup » de ce transfert. Nous verrons que ce changement réduit de façon radicale le cout de notre algorithme.

Si on considère l'écriture de N sous la forme $N = P * Q + E$ *le cout « k » du transfert de l'information « a » du premier vers le second facteur est définit par* $N = (P - a) * (Q + a + k) + E'$ *avec E' minimale. On constate donc dans cette écriture que la quantité « a » passe bien du premier vers les second facteurs et que ce transfert s'accompagne d'un ajustement d'ampleur « k » du second facteur. L'algorithme que nous allons mettre en place ici vise à trouver le cout « k » qui permet un transfert sans erreur d'une information a du premier vers le second facteur.*

Nous présentons deux approches permettant de construire cette formule de la factorisation en fonction de « k ».

Première Approche

$\forall a \in [0, P], \exists b \, ete \, tel \, que \; N = (P - a) * (P + b) + e.$ Pour a donner il suffit de prendre$alors \; b = P - \left[\frac{N}{P-a}\right] \; et \; e = N - (P - a)(P + b)$.

$N = P^2 + E \; et \; N = (P - a) * (P + b) + e = P^2 - aP + bP - ab + e \rightarrow$

➔ $E - e \coloneqq \Delta E = (b - a)P - ab$
Si on pose $k = b - a \; et \; m = a.b$ on à $\Delta E = kP - m$

Ce qui est une équation diophantienne linéaire assez simple. Une solution particulière est K = 0 et $M = -\Delta E$ la solution générale est donc de la forme $K = t \; et \; M = tP - \Delta E$, $t \in Z$

On en déduit donc que les solutions à l'équation de factorisation sont de la forme $M = kP - \Delta E$

Et $ab = kP - E \; et \; b - a = k \rightarrow b = k + a \; et \; a(k + a) = kP - \Delta E \; et$

$a^2 + ak + (kP - \Delta E) = 0$ L'ensemble des solutions à l'équation de factorisations sont donc donnée par les couples à, k avec k entier.

Toute les valeurs de k sont possibles, par contre toutes ne permettent pas d'obtenir une valeur entière pour a.

Pour la factorisation exacte on suppose E'=0 et donc $\Delta E = E$

Les solutions à notre équation de second degré sont donnée par

$\Delta = K^2 - 4(KP - E)$ la factorisation de n est possible uniquement si il existe une valeur de K tel que D est un carré parfait.

Dans ce cas a est donné par $a = \frac{-K + \sqrt{\Delta}}{2}$ et $b = a - K$.

Nous avons donc établi un lien entre la factorisation de N et la quantité k qui représente la différence d'information transféré entre les facteurs (k = b-a), nous verrons plus loin comment exploité cette relation. Pour l'instant nous allons présenter une façon plus simple de la retrouver.

Deuxième approche.

Lemme 4.1 : La factorisation de N peut donc s'écrire comme fonction de la seule variable k.

On à $N = \left(P - \frac{(-\boldsymbol{k}+[\sqrt{\Delta}])}{\mathbf{2}}\right) * \left(P + \frac{(-\boldsymbol{k}+[\sqrt{\Delta}])}{\mathbf{2}} + k\right)$ ou $\Delta = \boldsymbol{k^2 + 4(kp - E)}$

- ➔ La factorisation consiste alors à itérer sur les différents couts de factorisation pour trouver celui qui permet de transférer totalement (sans erreur) une quantité « a » du premier vers le second facteur.

Démonstration :

Soit N entier on pose $P = [\sqrt{N}]$, $E = N - P^2$ et on en déduit $N = P^2 + E$

$\forall a \in [1, P], \exists k \in IN\ tel\ que\ N = (P - a) * (P + a + k) + e. avec\ k \geq 0$

Il suffit de remarquer que dans l'expression $b = P - \left[\frac{N}{P-a}\right]$ b est forcément supérieur à a si a est supérieur à 1.

On en déduit $a^2 + ak - (kP - \Delta E) = 0$ on à une factorisation exact lorsque $e = 0 \rightarrow \Delta E = E$

Si on fixe k et qu'on résout l'équation de second degré en a, on a :

$\Delta = \boldsymbol{k^2 + 4(kp - E)}$ $\boldsymbol{et\ a} = \frac{(-\boldsymbol{k}+\sqrt{\Delta})}{\mathbf{2}}$

Notons que Δ n'est pas forcément un carré parfait, et sa racine n'est donc pas forcément entière, la valeur de a permettant d'obtenir la valeur de ΔE la plus proche possible de E (i.e. d'avoir le e minimal) est donc obtenu pour a $= \frac{(-\boldsymbol{k}+[\sqrt{\Delta}])}{\mathbf{2}}$ ou $[\sqrt{\Delta}]$ est l'arrondi de Δ.

Lemme 4.2

$N = P^2 + E$ Toutes factorisations de N peut s'écrire sous la forme $N = (P - a)Q$ avec $Q = \frac{N}{P-a}$ alors Q peut se mettre sous la forme $Q := P + a + k$ avec k réel et $k \geq -1$

Démonstration :

$N = P^2 + E$ et $N = (P - a)(P + a + k)$

$\rightarrow N = P^2 + E = (P - a)(P + a + k) = P^2 - a^2 + k(P - a)$

$\rightarrow E = -a^2 + k(P - a) \rightarrow k = \frac{a^2+E}{P-a}$

Or $a < P \rightarrow (P - a) > 0 \rightarrow signe(k) = signe(a^2 + E)$

- ➔ Si E positif alors k est positif
- ➔ Si E négatif alors k négatif ssi $a < \sqrt{-e}$ sinon k positif
- ➔ K < -1 : en effet k = -1 ssi $-(P - a) = a^2 + e \rightarrow a(a - 1) = -(P + e)$ or a(a-1) est positif (si a>1) et $-(P + e)$ est négatif car P>e et P positif.
- ➔ K ne saurait donc être inférieur à -1.

Equation de sens opposé

Plutôt que d'envisager la factorisation de N en partant de P et en recherchant le premier facteur de N qui lui est inférieur, on peut partir de P et rechercher le premier facteur qui lui est supérieur (nous verrons l'importance d cette seconde équation lorsque nous traiterons de la question du rééquilibrage des facteurs). Ce qui nous conduit à la formulation suivante du problème.

Soit N entier on pose $P = \lceil \sqrt{N} \rceil$, $E = N - P^2$ et on en déduit $N = P^2 + E$

$\forall a \in [P, N], \exists k \in IN \; tel \; que \; N = (P + a) * (P - a - k) + e. avec \; k \geq 0 \; et \; e \; minimale$

En particulier si N n'est pas premier, alors :

$\exists (a, k) \in IN \; tel \; que \; N = (P + a) * (P - a - k)$

De la même façon que précédemment, On en déduit $a^2 + ak + (kP - \Delta E) = 0$ on à une factorisation exact lorsque $e = 0 \rightarrow \Delta E = E$

Si on fixe k et qu'on résout l'équation de second degré en a, on a :

$\Delta' = \boldsymbol{k^2 - 4(kp - E)}$ $\boldsymbol{et \; a = \frac{(-k+\sqrt{\Delta})}{2}}$

Et en comparant au delta de l'équation précédente, on à $\Delta' = \Delta + 8kP$, on peut donc exploiter cette équation pour, en partant du point P, rechercher les facteurs de N à la fois parmi les valeurs inférieurs et celles supérieur à P.

Conséquences

Avant de voir comment exploiter ces résultats dans un algorithme, nous allons essayer d'en comprendre les implications.

Soit N entier, on pose $P = \lceil \sqrt{N} \rceil$ ***et*** $R = \lceil \sqrt{P} \rceil$ ***et*** $E = N - P^2$

Soit $a \in IN$ *et* $a < P$ *si* $(P - a)|$ N alors P, E, a et k ($k = \frac{N}{p-a} - P$) sont liés par l'équation suivante $a^2 + ak - (kP - E) = 0$ (1)

Nous avons exploité cette équation pour montrer que la factorisation de N pouvait s'écrire comme une fonction dont l'unique inconnu est k (et non plus a). le but de cette partie est de bien comprendre ce qu'implique réellement d'exprimer la factorisation de N en fonction de k et non plus en fonction de a.

Relation entre la quantité à transférer et le cout du transfert (i.e. entre a et k)

Commençons par nous intéresser à la relation qui existe entre a et k. si on résout l'équation (1) en a, on obtient l'expression suivante de k en fonction de a. $k = f(a) = \frac{a^2-E}{(P-a)}$ Et vu que nous nous intéressons au cout entier on posera $k = \left\lceil \frac{a^2-E}{(P-a)} \right\rceil$

$f'(a) = \frac{-3a^2+2aP-E}{(P-a)^2}$ *et* $f'(a) = 0 \rightarrow a_0 = \sqrt{P^2 - E} - P$

Plusieurs remarques peuvent être faites :

- ➔ La relation entre « a » et « k » n'est pas bijective. Le cout « k » est le même pour des plages entière de valeur de « a ». des éléments voisins ont le même cout de factorisation, mais le niveau d'erreur n'est pas le même pour tous.
- ➔ K est décroissant en a_0 puis croissant : plus il Ya d'information à transférer de petit vers le grand facteur, plus le cout du transfert est élevé.
- ➔ Si E > 0 (le niveau d'erreur initial est positif) alors k est croissant.

Soit $\lambda \in Q$ ***tel que*** $a = \lambda R$ ***(remarquons qu'on a nécessairement*** $\lambda < R$ ***car*** $a < P$ ***)***

- ***Dans ces conditions une approximation de k est donné par*** $k \approx \lambda^2$ ***.***
- ***A tout moment, le rapport entre les deux facteurs (p-a) et (p+b) est donné par :***
  Si on pose $\frac{V}{U} = \frac{P+a+k}{P-a} := 1 + \varepsilon$ ***alors*** $\varepsilon = \left(\frac{\lambda}{R}\right)^2 + 2\frac{\lambda}{R} \approx 2\frac{\lambda}{R}$

Si $N = U * V$ ***et que*** $\frac{V}{U} = 1 + \varepsilon$

- Alors la valeur de a qui permettra de factoriser N est donné par :
  $a \approx \lambda R$ ***et*** $\lambda \approx \frac{R}{2}\varepsilon \rightarrow a \approx \frac{P}{2}\varepsilon$
- La valeur de k qui permet de factoriser N est donné par :
  $k \approx \lambda^2 \approx \frac{P}{4}\varepsilon^2$

- Le rapport entre a et k est donné par :
  $\frac{k}{a} \approx \frac{\varepsilon}{2}$

➔ **La factorisation de N comme fonction de k est 1/ε fois plus rapide que sa factorisation en itérant sur les valeurs de a. plus ε sera faible plus le coup de la factorisation sera faible le coup par la méthode itérative (sur a) étant linéaire en ε et celui par k étant quadratique.**

Démonstration :

On en déduit $k \approx \frac{(\lambda R)^2 - E}{(R^2 - \lambda R)} = \frac{\lambda^2 - \left(\frac{E}{R^2}\right)}{1 - \frac{\lambda}{R}}$ $or\ E < \frac{P}{2} \rightarrow \frac{E}{R^2} \approx 0$ par ailleurs tan que λ est négligeable devant R on à $\lambda/R \approx 0$ et on en déduit $k \approx \lambda^2$ CQFD

Algorithme

Nous pouvons donc proposer un nouvel algorithme

```
P = Arrondi (Racine(N))
E = N = P*P
K = 0
Tan que  e <> 0 faire
        {
                Delta =  k^2+4(kp-E)
                Racine_delta = Arrondi (racine(Delta))
                Ajust = (-k+ Racine_delta)/2
                E = N – (P- Ajust) (P+ Ajust+k)
                K = k +1
        }
Fin faire
```

Rq : lorsque e n'est pas nul e est li niveau d'erreur minimal atteint pour tous les couple (a,b) tel que b-a = k et N = (P-a)(P+b) +e.

Analyse

L'ensemble des valeurs possibles de a est [1,P]

1. Après k itérations l'ensemble des valeurs analysé est [0, √k *R]
2. L'apport marginal de chaque itération au parcourt de [1,P] est donc dégressif.
3. L'apport marginal de la kieme itération est $\left(\sqrt{k}-\sqrt{k-1}\right)R$.
4. Dans le cadre du RSA par exemple ce n'est pas la totalité de l'ensemble [1,P] qui nous intéresse mais seulement les valeurs de a permettant d'obtenir des facteurs de rapport V/U compris entre 1 et 2 i.e. [1, λR] ou λ = …

FACTORISATION PAR REEQUILIBRAGE DES FACTEURS

Dans la partie précédente, nous avons présenté un nouvel algorithme de factorisation et nous avons démontré qu'il était d'une grande efficacité lorsque les deux facteurs étaient relativement proches. Dans cette partie nous alors donné une définition plus adapté de cette notion de proximité et analyser le cout de la factorisation par rapport à ce niveau de proximité. Puis nous allons présenter une adaptation à apporte à l'algorithme pour limiter la perte de performance lorsque le déséquilibre s'installe entre les facteurs.

NIVEAU DE DESEQUILIBRE D'UN NOMBRE

Dans cette partie nous allons discuter de la notion de niveau de déséquilibre d'un nombre.

Soit $N = U * V, V > U$, le degré d'équilibre de N peut être vu comme une mesure du degré de proximité de ses deux facteurs, de façon basique on peut mesurer ce niveau d'équilibre par le rapport ou la différence de ces fauteurs, mais nous définirons une mesure plus adapté à l'analyse de nos algorithme et montrerons comment elle est relié à ces deux mesures plus basique.

Soit $N := U * V, V > U$, on pose comme toujours $\boldsymbol{P =: \left[\sqrt{N}\right]}$ ***et*** $\boldsymbol{R := \left[\sqrt{P}\right]}$ ***nous avons montré dans le paragraphe précèdent qu'il existait un couple*** $\boldsymbol{(a,k)}$ ***tel que*** $\boldsymbol{N = (P-a)(P+a+k)}$***, on à alors*** $\boldsymbol{U := P-a}$ ***et*** $\boldsymbol{V := P+a+k.}$

On pose aussi $\boldsymbol{L = V - U}$ et $1 + \varepsilon := \frac{V}{U} \rightarrow \boldsymbol{\varepsilon := \frac{V}{U} - 1}$

On appel niveau d'équilibre de N et on note φ(N) ***la quantité définit par*** $\boldsymbol{\varphi(N) := \varphi = \left[\frac{P-U}{R}\right] = \left[\frac{a}{R}\right]}$

On dira que N est équilibrer si $\exists\, \boldsymbol{a < R\ tel\ que\ (p-a)|N}$ ***(en d'autre terme si*** $\boldsymbol{\varphi = 0}$ ***)***

La relation entre L et $\boldsymbol{\varphi}$ ***est donné par :*** $\boldsymbol{L = \varphi^2 + 2R\varphi \approx 2R\varphi}$

➔ $\boldsymbol{L \approx 2R\varphi\ et\ \varphi \approx \frac{L}{2R}}$

Le lien entre φ *et* ε *est donné par* $\varepsilon = \left(\frac{\varphi}{R}\right)^2 + 2\frac{\varphi}{R} \approx 2\frac{\varphi}{R}$

➔ $\varepsilon \approx 2\frac{\varphi}{R}$ *et* $\varphi = 2R\varepsilon$

RQ :

➔ Plus le niveau d'équilibre φ est faible plus U et V sont proche.
➔ Si N peux être factorisé de plusieurs façons comme produit de facteurs u et v alors on utilise pour calculer le niveau d'équilibre de N le facteur u le plus proche de P.

Lemme 4.1

Le cout de la factorisation par transfert d'information est égal au carré du niveau d'équilibre du nombre à factoriser.

Ce lemme a pour conséquence qu'il est quasiment inenvisageable de factoriser par transfert d'information les nombres fortement déséquilibrés.

Cet algorithme est donc d'une grande efficacité mais uniquement lorsque les facteurs sont équilibrés, pour pouvoir l'exploiter pour des propose de factorisation plus générale, nous devons être capable de rééquilibrer les facteurs, c'est l'objet du paragraphe suivant.

REEQUILIBRAGE DES FACTEURS

Nous avons vu dans les partie précédente que l'apport marginale d'une itération dans l'exploration des valeurs possible de a était d'autant plus élevé que les facteurs étaient proches (que le nombre était équilibré) par ailleurs nous avons vu que cette méthode n'était utilisable que pour des nombre peu déséquilibré. Dans cette partie nous présentons une procédure permettant de rééquilibrer un nombre et donc de rendre possible ou de faciliter sa factorisation.

Soit $N := U * V, V > U$ *et on pose* $\frac{V}{U} := 1 + \varepsilon := \lambda$

Nous savons que la quantité 1+ε peut être approximé à un degré de précision arbitraire par un nombre rationnel (consulter toutes études sur les fractions continues) i.e.

Dans certain cas on peut obtenir une expression exacte de λ comme fraction rationnelle.

$\exists(\alpha, \beta) |\ 1 + \varepsilon = \frac{\beta}{\alpha}$ *on pose* $N' = (\alpha * \beta)N = \alpha * \beta * U * V = (\beta U) * (\alpha V) := U' * V'$

Alors U' = V'

Plus généralement on peut obtenir pour un degré de précision arbitraire une fraction rationnelle donnant une approximation de λ.

$soit\ \lambda' = \frac{\beta'}{\alpha'} = (1+\varepsilon) + \acute{\varepsilon}\ \ et\ \acute{\varepsilon} \ll \varepsilon\ \ alors\ N' = (\beta U) * (\alpha V) \coloneqq U' * V'$ est un nombre plus équilibré que N en particulier le niveau de déséquilibre est passé de

$\varphi = 2R\varepsilon$ à $\varphi' = 2R\acute{\varepsilon}$ or $\acute{\varepsilon} \ll \varepsilon$ et donc $\varphi' \ll \varphi$ N' a donc un coup de factorisation très inférieur à N. plutôt que factoriser N, nous pouvons donc factoriser N' a moindre coup et retrouver les valeur U' et V' il suffit alors de les diviser par α et β pour retrouver U et V.

➔ Il nous suffit donc de trouver un λ' suffisamment proche de 1+ε pour réduire d'autant le coup de la factorisation, celui-ci devenant nul dès que l'erreur έ d'estimation de λ par λ' devient inférieur à 1/log(N,10).

La seule difficulté est que nous ne connaissons pas λ. Cependant nous en connaissons souvent un encadrement. Par exemple dans le cadre de la cryptographie RSA λ est compris entre 1 et 2.

➔ Le problème de la factorisation se résume donc à deviner la valeur de λ avec une précision suffisante.
- Vu sous cet angle il est dans une certaine mesure indépendant de la taille des nombres car seul compte leur rapport.
- Il est aussi indépendant de la distance relative des deux nombres (le coup de la factorisation est le même pour les rapports 1.1, 1.5, et 1.9 par exemple).

Comment donc deviner de façon assez précise le rapport entre les deux nombres ?

➔ En testant le « maximum » de possibilités ou du moins un certain nombre.

Aussi nous allons quadriller tout cet espace en choisissant un pas de έ. Nous allons donc considérer la série de valeur de $L = \{\lambda_i = 1 + n * \acute{\varepsilon},\ n = 0, \dots, m\}\ ou\ m = \frac{2}{\acute{\varepsilon}}\ est\ la\ taille\ du\ maillage\ et\ \acute{\varepsilon}\ son\ pas.$

Soit $(\beta_i, \alpha_i)\ tel\ que\ \lambda_i = \frac{\beta_i}{\alpha_i}$ on pose $N = \{N'_i = (\beta_i U) * (\alpha_i V)\}$ alors il existe au moins un indice i tel que le degré d'équilibre de N'i soit inférieur à $2R\acute{\varepsilon}$

Le coup de factorisation sur cette grille est donc au maximum égale à $2mR\acute{\varepsilon}$

Remarque

Notons que la construction du nombre N' par multiplication de N par un facteur λ ne conduit pas à une unique possibilité d'équilibre pour le nombre. En effet si λ n'est pas produit de deux nombres premiers se sont toutes les combinaisons de de facteur de N' qui seront envisagé pour réduire le niveau d'équilibre.

En d'autre termes si $\lambda = \lambda 1 * \lambda 2 * \ldots * \lambda n$, $N = U * V$ *et* $N' = \lambda * N$ alors N' est équilibrer si il existe une deux sous ensemble disjoint de (λi) *et* $(\lambda' i)$ de $\{\lambda 1, \lambda 2, \ldots, \lambda n\}$ tel $N' = U' * V'$ $(ou\ U' = [U * \prod(\lambda i)]\ et\ V' = [V * \prod(\lambda' i)] *])$ soit équilibré.

- ➔ Plus λ à de facteur plus il existe de combinaisons susceptible de rééquilibrer N en le multipliant par λ.
- ➔ Dans le cadre de la factorisation général (si on ne se situe pas comme dans notre étude dans un cas particulier tel celui du RSA) il est plus judicieux de choisir des réseaux dont les nœuds sont des nombres composés de plusieurs facteurs plutôt que des produits de nombres premiers.

Exemple la grille uniforme de taille 10 sur l'ensemble [1, 1.1] se présente ainsi :

Coef (1+ ε)	Num (βi)	Den (αi)	Prd (λi)
1	1	1	1
1.01	101	100	10,100
1.02	51	50	2,550
1.03	103	100	10,300
1.04	26	25	650
1.05	21	20	420
1.06	18	17	306
1.07	46	43	1,978
1.08	27	25	675
1.09	109	100	10,900
1.1	11	10	110

Algorithme

Pour chaque nœud de la grille (chaque ligne du tableau) faire

- ***Multiplier N par λ et en déduire N' = λ *N***
- ***Essayer de factoriser N' pour obtenir U' et V' (utiliser la méthode de transfert d'information ou de Fermat, sur un nombre d'itération préfixé) (notons aussi qu'a chaque itérations et pour chacun des nœud on peut rechercher à la fois les facteur inférieur et supérieur à P en utilisant D et D' = D + 8kP pour la recherche des facteurs)***
- ***Diviser U' par β pour obtenir U et V' par α pour obtenir V (ou inversement, dans tous les cas chacun de ces facteur est divisible par l'une au moins des valeurs α et β car N' = α* β*N.***
- ***Sinon essayer le nœud suivant.***

Fin faire

- ➔ Cette approche permettra de factoriser N si jamais le rapport de ses facteurs est « proche » de l'un des nœuds de notre grille.

➔ Plutôt que d'itérer sur les nœuds il peut être plus judicieux d'itérer sur le niveau k de la recherche et à chaque niveau de lancer la recherche sur tous les nœuds (ce processus est plus efficace en raison du cout marginal décroissant de l'itération d'ordre k).

NB :

Il n'est pas indispensable d'utiliser la méthode de transfert d'information entre facteur pour factoriser le nouveau nombre N', on peut utiliser n'importe qu'elle méthode qui est d'autant plus performante que le degré déséquilibre du nombre est faible. On peut même utiliser conjointement les deux méthodes par exemple lorsque N' est impair, on peut utiliser la méthode de Fermat pour factoriser N' avant d'en déduire N en éliminant les facteurs (α,β) rajoutés.

Mise en place

Préalablement à la mise en place de cet algorithme, il faut :

- Choisir les bornes de la grille : pour du RSA par exemple se sera 1 et 2
- Choisir le type de grille : nous utiliserons ici des grille uniforme, mais rien ne nous empêche de construire plus de nœuds à dans les régions de la grille ou on estime plus probable que la valeur de λ soit.
- Choisir le pas du maillage : la densité du maillage dépend de l'investissement en temps qu'on est prêt souscrire pour l'exécution de notre algorithme. Plus le maillage est dense plus la probabilité d'être proche sur un des nœuds de la valeur réelle de λ est élevé. Par contre les temps nécessaire pour factoriser N sur tous les nœuds sera plus important.
- Choisir la profondeur k de la recherche sur chacun des nœuds. Plus k sera élevé plus il sera probable de retrouver les valeurs de λ relativement éloigné des nœuds. Par contre il sera d'autant plus couteux d'exécuter l'algorithme pour l'ensemble des nœuds.

➔ Tout l'art de la factorisation par ce processus se résume dans l'art de choisir la bonne densité du maillage et la bonne profondeur de recherche, afin de minimiser le cout de factorisation.
➔ Le nombre total d'opération à réaliser sera le produit du nombre de nœuds par la profondeur de recherche à chaque nœud.
➔ Un nombre n ne sera factorisable sur une grille que si au moins un des nœuds arrive à le rééquilibrer suffisamment pour que un transfert d'information (ou méthode de Fermat) de profondeur k arrive à le factoriser (i.e. que le niveau d'équilibre de N' est inférieur à la profondeur de recherche).
➔ On peut donc calibrer la taille de la grille (n* ε) et la profondeur (k) en fonction du temps qu'on est prêt à investir pour essayer de factoriser N.

Dans la partie suivant nous allons construire un indicateur permettant de mesure la difficulté de factorisation d'un nombre par ce procédé, ce qui nous permettra de définir les caractérisés des nombres facile à factoriser et ceux des nombres difficiles à factoriser.

Complexité de la factorisation et indice K de KAMTO

L'analyse de la complexité des algorithmes de factorisation est un sujet délicat pour la plus part des algorithmes elle se base sur des séries de test et des résultats historique. Un des grands intérêts de la méthode que nous avons construit dans ce chapitre est qu'il permet d'analyser très simplement la difficulté de factoriser un nombre N donné. Dans ce paragraphe nous résumons nos résultats dans la construction d'un indice permettant de situer de façon relativement précise le cout de la factorisation d'un nombre N donné.

Dans les paragraphes précédents nous avons discuté du problème de la factorisation et envisagé différente façon de répondre à cette question.

En guise de bilant nous proposons ici la construction d'un indicateur permettant de mesurer le degré de difficulté de la factorisation d'un nombre N donné en fonction de ses facteurs.

Soit $x = u.v$, v > u nous souhaitons analyser la difficulté de retrouver u et v à partir des algorithmes présentés dans les paragraphes précédent.

On pose $\lambda = V/U$ n = log(u,10) =log(v,10) le nombre de chiffres en base 10 de u.

On appel indicateur de Kamto de x et on note K(x) = a.b si sur un réseau de pas 10^-a il existe au moins un nœud pour lequel la valeur x' de x sur ce nœud soit composé du produit de deux facteurs u' et v' vérifiant

Interprétation : le cout de factorisation d'un nombre tel que K(x) = a.b est au maximum de 1+ 10^(a+2b) opérations.

Exp : soit x = u.v et n= log(u)=log(v) si $\lambda = V/U < 10^{-n/2}$ alors K(x) = 0.0, 2 itérations sont donc suffisante pour factoriser ces nombre. Ces nombre ont un coup de factorisation similaire à celui de la méthode de Fermat.

De façon plus générale un nombre est d'indice 0.T si $\lambda <$

Les nombre dont l'indice de Kamto est de la forme 0.T sont donc ceux qui sont facilement factorisable par la méthode de Fermat par exemple, ou ceux qui peuvent être factorisé en 10^2T opération par la méthode de transfert d'information entre les

facteurs… ie les nombre ayant au moins un facteur dont la distance à la racine du nombre est inférieur à T*R ….

Les nombres d'ont l'indice de Kamto est de la forme T.0 sont ceux dont qui peuvent être rééquilibré sur un réseau de pas 10^{-T} en d'autre terme si $\lambda < 10\text{^}T$, pour ces nombres il existe un point de maillage sur le réseau de pas T permettant de factoriser en 2 itération le nombre x'

Les indice de la forme T.S sont ceux qui peuvent être quasiment rééquilibrer sur des réseau de densité T puis factoriser sur un des maillage en S opérations, ce coup e

Pour déterminer l'indice il suffit de fractionner la valeur du rapport en 3 parties

. $\lambda = 1.XXX0000000YYYYYZZZZZ$ (ou . $\lambda = 1.XMMMMMYYYYYZZZZZ$) $M = X - 1$

Les valeur ZZZ désigne tous les chiffres qui sont au-delà de n/2 leur valeur est nul dans l'indice, au-delà de cette valeur chaque chiffre Y compte pour 1 incrément dans le calcul de D (D est le nombre de Y) (on en considère maximum 10 au-delà de 10 le nombre est considéré d'indicateur très grand (il est non factorisable par l'approche) de même chaque chiffre X alimente l'incrémentation de X si au-delà du 10 x il y a encore de chiffres alors le nombre est non rééquilibrable et sont indicateur de Kamto est infini.

Ainsi tout nombre dont l'indicateur est fini est factorisable. Les limites étant fixées pour chaque système …

Par exemple si le programme dois tourner en moins d'une heure sur mon ordinateur personnel, tous les nombre dont les borne d'indice sont inférieur à 3.3 sont à bannir, car il est possible d'exécuter 10^9 opération en moins de 1 heure sur un ordi personnel.

~~EPILOGUE~~

~~*« La reconnaissance des nombres premiers et des nombres composés avec leur décomposition en facteurs premiers est connue pour être des plus importants et utiles en arithmétique. Il a tant impliqué le zèle et la sagesse des géomètres anciens comme modernes qu'il serait superflu d'en discuter plus avant… En plus, la dignité des sciences mêmes semble exiger que tous les moyens possibles soient explorés pour trouver la solution d'un problème si élégant et si célébré »*~~

~~*K.F. Gauss.*~~

~~Lorsque en juin (2011) dernier la valorisation d'un CDO ma conduit à m'intéresser à la notion de PGCD approximatif j'étais loin de me douter que ceci me conduirait jusqu'à~~

cette épineuse question de la factorisation. Et il y a 3 mois encore quand les limites de mon algorithme m'ont poussé à me pencher sur la question de la factorisation approximative je ne me doutais (heureusement) pas que j'abordais là une des questions les plus difficiles de l'arithmétique (j'aurais probablement renoncé tout de suite), d'ailleurs la question qui m'intéressait n'était pas vraiment celle de la factorisation (autrement je me serais directement penché sur les publications concernant le sujet). J'ai donc abordé non sans une certaine naïveté la question de la factorisation approximative qui m'a naturellement conduit à celle de la factorisation.

Deux mois environs après le début de mes recherches la première version de mon approche par transfert d'information était sur pieds répondant de façon assez satisfaisante à mon problème de factorisation approximative mais aussi avec une étonnante efficacité à celle de la factorisation classique. Quelque semaine après ma compréhension de l'algorithme et de ses performances était meilleure, je commençais alors à m'intéresser avec un enthousiasme débordant à ses conséquences sur les champs d'application de la factorisation, en particulier la cryptographie RSA.

Si mon algorithme ne résolvait pas de façon définitive la question de la factorisation, il montrait au moins que dans certain cas indépendamment de la taille des nombres la factorisation pouvait être relativement simple. Aussi je formulais ce qui je pensais être une recommandation à adresser aux entreprise utilisatrice du RSA quant aux précautions à prendre dans le choix de facteur composant le nombre : ceux-ci ne devait pas être trop proche l'un de l'autre faute de quoi il serait assez facile de les factoriser. Puis me lançant dans quelques lectures sur la factorisation pour m'informer de ce qui était fait d'autre dans le domaine, quelle ne fut pas ma surprise quand je découvris qu'en prenant appui sur une méthode de factorisation proposé par Fermat cette recommandation avait déjà été formulée.

Je n'avais certes pas retrouvé l'algorithme de Fermat (je fus soulagé de ne pas voir mes efforts disparaitre d'un trait) mais mon approche conduisait à des résultats similaires et à un algorithme dont les performances était comparables. Grande fut toute fois ma déception, tous mes efforts m'avaient conduit (plusieurs siècle en arrière) en un point proche du point de départs des algorithmes modernes. En effet l'idée de Fermat est à la base de presque tous les algorithmes modernes de factorisation. Je n'apportais dans le domaine certes pas d'avancé en terme de performance mais tout de même une méthodologie radicalement nouvelles aux performances prometteuses. Parvenu au point de départ de l'algorithme il ne me restait plus qu'à faire le chemin correspondant aux quatre dernières décennies de développement sur le sujet.

Après quelque jour de déceptions, je me convainquis que si j'étais parvenu en quelque semaine à un algorithme dont les performances étaient comparables à ceux des premiers algorithmes modernes, je pourrais moyennant quelques efforts le faire évoluer, et rattraper le retard qu'avait mon approche sur les modernisations de celle de Fermat. Je repris mes analyses, et au bout de quelques semaines, un après-midi dans un

~~train entre Marseille et Paris l'idée du rééquilibrage surgit. le lendemain (2 Avril) tout était clair la construction du processus de rééquilibrage qui permettait de booster mon algorithme était maitrisé et une nouvelle période de travail enthousiaste s'ouvrait.~~

~~Mes enquête ultérieure et un coup d'œil jeter aux algorithmes moderne construit sur l'idée de base de Fermat montre que eux aussi on trouver des moyens de palier aux limites de l'algorithme de base…~~

~~Mon approche est certes différente mais il y a quelque similitude avec ces algorithmes et avec leurs évolutions. Ma connaissance de ces méthodes est encore beaucoup trop limitée pour produire ici une comparaison avec ces algorithmes. Ce nouvel algorithme offre vraisemblablement de très bonnes performances (et débouche sur de nouvelle recommandation concernant le choix des clés). Je poursuivrais dans la suite l'étude des autres algorithme moderne de factorisation afin de produire un comparaison plus rigoureuse entre la méthode de transfert d'information et celle de Fermat, puis entre la méthode de rééquilibrage et le QFS, et peut-être que cette étude me permettra de trouver de nouvelle idée pour améliorer encore la nouvelle approche ici proposer, voire de combiner cette approche avec les méthodes existantes.~~

CONCLUSION

Nous utiliserons dans les chapitres suivant cette méthode de factorisation pour poursuivre notre recherche du pgcd au-delà de la racine du plus petit élément.

Chapitre 6

FACTORISATION APPROXIMATIVE ET PGCD APPROXIMATIF

La notion de factorisation approximative introduite au chapitre précédent nous à permit étant donné un entier x, de construire les couples (p,q) vérifiant x = p*q+e avec $p < \lfloor\sqrt{x}\rfloor, q > \lceil\sqrt{x}\rceil$ et minimisant le niveau d'erreur e dans l'approximation de x par le produit p*q. dans ce chapitre nous généralisons cette construction au cas d'un ensemble fini X d'entier et exploitons les résultats établis précédemment pour construire une nouvelle méthode de recherche du pgcd approximatif.

Soit $X := \{x_0, x_1, x_2 \ldots . x_n\}$ un ensemble d'entier distinct et ordonnés i.e. $x_0 < x_1 < x_2 < \cdots < x_n$.

Le pgcd approximatif P de X vérifie : $\forall xi \in X\ , \exists qi, ei\ tel\ que\ xi = P * qi + ei$. Avec la contrainte $\delta i := \frac{ei}{xi} < \delta$ pour un seuil δ fixé, ou $P < Pmin\ et\ \delta i$ minimal , Le problème de recherche du pgcd approximatif consiste alors en l'identification de P à partir des seuls xi.

Nous savons que $P \in [1, \boldsymbol{x_0} + \epsilon[$ ou $\boldsymbol{x_0}$ est le plus petit élément de l'ensemble X, (on rajoute ε car le PGCD peut être éventuellement un peu plus grand que x_0.).

Dans la première partie de cet article nous avons développé une approche permettant de retrouvé P dans le cas où celui-ci était éloigné de $\boldsymbol{r_0} = \lfloor\sqrt{\boldsymbol{x_0}}\rceil$ (voir corolaire), en effet cet algorithme est d'autant plus efficace que l'on est éloigné de r_o, ceci Vien du fait que la structure stable sur laquelle repose notre algorithme est partagé par d'autant moins d'élément que son générateur est proche de r_o. Dans ce chapitre nous allons donc compenser cette limitation en exploitant une nouvelle structure stable qui elle apparait autour de r_o, cette approche nous permettra dons de retrouver le pgcd avec une efficacité proportionnelle à sa proximité à r_o. Cette approche repose sur la notion de factorisation approximative introduite au chapitre précédent et sur laquelle nous reviendrons brièvement dans la suite.

La présentation de cette nouvelle approche se fera dans le même esprit que celui développé dans la première partie, en effet seul la structure utilisé change, l'esprit général reste le même.

FACTORISATION APPROXIMATIVE

Nous commençons par exploiter les résultats du chapitre précédent pour définir la notion de factorisation approximative sur laquelle repose notre construction du Pgcd.

Def 6.1(factorisation approximative) :
Soit x un entier on pose $r = \lfloor \sqrt{x} \rfloor$ ***et donc*** $x = r^2 + e$ ***.***
On appel factorisation approximative de x, toutes écriture de x sous la forme $x = p * q + \epsilon$ ***avec*** $p \leq r, q \geq r$ ***et*** $\epsilon < \frac{p}{2}$ ***.***
Les éléments p et q sont alors appelé facteur approximatif de x.

Lemme6.1 (factorisation approximative) :
Soit x, un entier, toutes factorisation approximative de x s'écrit de façon unique sous la forme $x = (r-a)(r+a+k) + \varepsilon$ ***pour un couple*** (a,k) ***donné***

Dem : (voir chapitre précédent : lemme 4.2)

1. $p < r \rightarrow \exists a < r$ *tel que* $p = r - a \rightarrow$ *prendre* $a := p - r$
2. $q > r \rightarrow \exists b > 0$ *tel que* $q = r + b$ par ailleurs $q = \left\lfloor \frac{x}{p} \right\rfloor = \left\lfloor \frac{x}{r-a} \right\rfloor = r + a + k$ *avec* $k = \left\lfloor \frac{a^2+e}{p-a} \right\rfloor$ et b = a+k avec k > -1.

Prendre $a := r - p$; $k := q - r - a$ *ou* $q := \left\lfloor \frac{x}{p} \right\rfloor$ *et* $\varepsilon = x - p * q$

Def 6.2(facteur approximatif de niveau k) :
Soit x un entier on pose $r = \lfloor \sqrt{x} \rfloor$ ***et P < r. on dit que P est un facteur approximatif de niveau k de x ssi on peut factoriser x sous la forme*** $= P * Q + \varepsilon = (r-a)(r+a+k) + \varepsilon$ ***avec*** $P = r - a, Q = \left\lfloor \frac{x}{P} \right\rfloor = r + a + k$ ***et*** $\varepsilon = x - P * Q$ ***. A est appelé ajustement de p et k niveau de P.***

Lemme 6.2 (calcul du niveau d'un facteur) :
Soit x un entier on pose $r := \lfloor \sqrt{x} \rfloor$ ***et*** $x = r^2 + e$ ***. Soit P< r, on pose*** $a := p - r$ ***, le niveau k de P est donné par :*** $k = \left\lfloor \frac{a^2+e}{r-a} \right\rfloor$

➔ La factorisation approximative consiste donc à séparer simplement le nombre en deux facteur qui t-a ce qu'il y'ai un résidu. L'intérêt de ce procéder est comme nous allons le voir que si on considère l'ensemble des facteurs d'une région

spécifique, nous somme capable d'identifier directement qu'elle factorisation approximative produit le plus faible niveau d'erreur.

Démonstration : voir lemme 4.1

Lemme 6.3 (point de rupture) :
Soit x un entier et k fixé l'ensemble des facteurs de niveau k de x est donné par : $\boldsymbol{P = r - a \; avec \; a \in]Amin\,, Amax\,[}$ ***et*** $\boldsymbol{Amin = \left[\frac{-(2k-1)+\sqrt{D^-}}{4}\right]}$ ***et*** $\boldsymbol{Amax = \left[\frac{-(2k+1)+\sqrt{D^+}}{4}\right]}$ ***ou*** $\boldsymbol{D^+ = (2k+1)^2 - 8\big(2e - r(2k+1)\big)}$ ***et*** $\boldsymbol{D^- = (2k-1)^2 - 8\big(2e - r(2k-1)\big)}$

La couverture d'ordre k est donc donné par : $C(k) = Amax - Amin \approx \sqrt{D^+} - \sqrt{D^-}$
….

➔ On en déduit que les facteurs de niveau k de x sont compris entre [r-a- et r+a+]

Démonstration

Soit a tel que la factorisation de x est de la forme $x = (r-a)(r+a+k) + e'$ pour k fixé. Nous recherchons le niveau d'ajustement maximal a^+ compatible avec ce niveau k.

Le niveau d'ajustement a^+ correspond au moment ou le niveau d'erreur atteint son maximum ie $e^+ = \frac{r-a}{2}$ on à alors la factorisation $x = (r-a)(r+a+k) + \frac{r-a}{2}$

On en déduit $x = r^2 + rk - a^2 - ak + \frac{r-a}{2} \rightarrow 2e = r - a - 2a^2 - 2ak$

$\rightarrow 2a^2 - a(2k+1) - r(2k+1) + 2e$

La résolution de cette équation de second degré conduit à :

$$D^+ = (2k+1)^2 - 8\big(2e - r(2k+1)\big) \; et \; on \; en \; déduit \; a^+ = \frac{\big(-(2k+1) + \sqrt{D}\big)}{4}$$

De même le niveau d'ajustement minimal a^- est obtenu lorsque le niveau d'erreur franchi la borne inférieur $e+ = -\frac{r-a}{2}$ et on a : $x = (r-a)(r+a+k) - \frac{p-a}{2}$

$\rightarrow 2a^2 - a(2k-1) - r(2k-1) + 2e$

La résolution de cette équation de second degré conduit à :

$$D^- = (2k-1)^2 - 8\big(2e - r(2k-1)\big) \; et \; on \; en \; déduit \; a^- = \frac{\big(-(2k-1) + \sqrt{D}\big)}{4}$$

➔ Notons que la différence entre les deux bornes est donné par $a^+ - a^- = \frac{(\sqrt{D+} - \sqrt{D-}-2)}{4} \approx \sqrt{D^+} - \sqrt{D^-}$

➔ On remarque aussi que $D^+ - D^- = 16r + 8k$

Maintenant que nous avons construit pour un niveau k donné, l'ensemble des ajustements et donc des facteurs de ce niveau, nous nous intéressons maintenant à celui d'entre eux qui permet d'obtenir le niveau d'erreur minimal, et définissons dans la même perspective que le diviseur idéal introduit au chapitre précédent, le facteur idéal.

Def 6.3 (facteur idéal de niveau k) :
Soit x un entier, on appel facteur idéal de niveau k de x, le facteur P de niveau k qui minimise le niveau d'erreur ε.
En d'autre terme, Pour tout a' tel que $P := r - a'$ ***et*** $x = (r - a')(r + a' + k) + \varepsilon'$ ***on a*** $|\varepsilon| < |\varepsilon'|$

➔ C'est donc le facteur de niveau k, tel que la factorisation de x par tout autre facteur de niveau k aboutit à un niveau d'erreur supérieur.

Lemme 6.4 (calcul du facteur idéal) :
Soit x un entier. On considère la factorisation initiale de x sous la forme $x = r^2 + e$ ***. Soit k donné. Le facteur idéal de niveau k de x est donné par*** $P := r - a$ ***et :*** $a = \left[\frac{(-k+\sqrt{\Delta})}{2}\right]$ ***ou*** $\Delta = k^2 + 4(kr - e)$

Démonstration : voir chapitre précédent :

Ayant définit pour un niveau k le facteur idéal ainsi que l'ensemble de facteur

Def 6.3 (sensibilité d'un idéal) :
Soit x un entier, et p := r-a le facteur idéal de niveau k de x. la quantité δ(d) qui mesure le taux d'erreur introduit pour un éloignement d par rapport à l'idéal. δ(d) est définit par :
$\delta(d) = \frac{e(r-a')}{x}$ ***ou*** $a' := r - (a + d)$ ***et*** $e(r - a') := x - (r - a') * \left[\frac{x}{r-a'}\right]$ ***e' est le niveau d'erreur associé au facteur (r-a'). avec*** $a' \in [a^-, a^+] \rightarrow d \in [a^- - a, a^+ - a]$

Evaluation de e'

$x = (r - a)(r + a + k) + e$

$et\ x = (r - a')(r + a' + k) + e' = \big(r - (a + d)\big)(r + (a + d) + k) + e'$

$x = [(r - a) - d][(r + a + k) + d] + e'$

$x = (r - a)(r + a + k) - d(r + a + k) + d(r - a) - d^2 + e'$

$x - (r - a)(r + a + k) = e' - d^2 - d(k + 2a)$

➔ $e' = e + d^2 + d(2a + k)$

De même pour $a' = a - d$ *on à*

$x = (r - a')(r + a' + k) + e' = \big(r - (a - d)\big)(r + (a - d) + k) + e'$

$x = [(r - a) + d][(r + a + k) - d] + e'$

$x = (r - a)(r + a + k) + d(r + a + k) - d(r - a) - d^2 + e'$

$x - (r - a)(r + a + k) = e' - d^2 + d(k + 2a)$

➔ $e' = e + d^2 - d(2a + k)$

On en déduit le lemme suivant :

Lemme 6.4 (calcul de la sensibilité de x au niveau k) :
Soit x un entier. P :=r-a son facteur idéal au niveau k, et e le niveau d'erreur associé à P.
La sensibilité de p à un ajustement positif d'ampleur d est donné par :

$\delta(d) -= \frac{e'}{x} = \frac{\left(e + d^2 - d(2a+k)\right)}{x} = \frac{e}{x} + \frac{\left(d^2 - d(2a+k)\right)}{x}$

$\delta(d) -= \delta(0) + \frac{d^2}{x} - d * \frac{2a+k}{x}$

De même $\delta(d) += \delta(0) + \frac{d^2}{x} + d * \frac{2a+k}{x}$

➔ *Le niveau d'erreur initial e est celui de l'idéal il est donc « faible » : e peut donc être considérer petit par rapport à d(2a+k) de même si on considère que d est petit par rapport à (2a+k) alors une bonne approximation de* $\delta(d)$ *est* $d * \left(\frac{2a+k}{x}\right)$*, cette sensibilité augmente donc avec le niveau k des facteur considéré (rappelons que a et k évolue dans le même sens : voir chapitre précédent)*

➔ *Le niveau d'erreur* $e' = e + d^2 - d(2a + k) \approx -d(2a + k)$ *ou* $e' = e + d^2 + d(2a + k) \approx d(2a + k)$ *va donc diminuer/augmenter régulièrement jusqu'à atteindre le niveau minimal p'/2 et entrainer une augmentation/diminution du niveau k qui passera à k' =k+1 ou k'=k-1*

Intéressons-nous maintenant à ce qui se passe lorsque l'ajustement devient suffisamment grand pour qu'une variation d'une unité entraine un passage du facteur au niveau suivant :

$x = (r - a)(r + a + k) + e$

et $x = (r - a')(r + a' + k') + e' = \big(r - (a + 1)\big)(r + (a + 1) + (k + 1)) + e'$

$x = [(r - a) - 1][(r + a + k) + 1 + 1] + e'$

$x = (r - a)(r + a + k) - (r + a + k) + 2(r - a) - 2 + e'$

$x - (r - a)(r + a + k) = e' - 2 + (r - 3a - k)$

➔ $\Delta e \coloneqq e' - e = (3a + k + 2) - r$

Cette variation de niveau d'erreur est minimale pour :

$\Delta e = 0 \rightarrow r - 3a - k - 2 = 0$ *or* $k = \frac{a^2 + \varepsilon}{r-a}$ *ou* $\mathcal{E}$ *est le niveau d'erreur de la factorisation initiale* $x = r\hat{}2 + \mathcal{E}$

On en déduit :

$$-a^2 - \varepsilon + (r-a)(r-3a-2) = 0 \rightarrow 2a^2 + (2-4r)a + (r^2 - 2r + \varepsilon) = 0$$

La résolution de cette équation de second degré nous donne

$a = \frac{\left[4r-2-\sqrt{(4r-2)^2-8(r^2-2r+\varepsilon)}\right]}{4}$ *et son considère son arrondi* $a := [a]$

à partir (autour) de cet instant toute variation de a va entrainer une variation de k, c'est donc le niveau maximal d'ajustement qui pourra être traiter par factorisation approximative.

Lemme 6.4 (calcul de la sensibilité de x au niveau changement de niveau) :
Soit x un entier et P := r-a un facteur de niveau k de x. si a est suffisamment grand pour que P' = r-(a+1) soit un facteur de niveau k+1 alors, la variation d'erreur du à se passage est donné par. $e' - e = (3a + k + 2) - r$

Conclusion

Nous constatons ainsi que lorsqu'un des facteurs de x est inférieur à sa racine, le niveau d'erreur ou du moins la minimisation de celui-ci ne dépend que du niveau k du facteur en question et non pas du facteur lui-même, de plus le niveau d'erreur a une croissance monotone en fonction de la distance à l'idéal. Aussi l'ensemble des facteurs de même niveau peuvent être envisagé comme un bloc unique centré sur le diviseur idéal.

FACTEUR COMMUN

Dans la première partie de cet article nous avons vu comment déterminer pour différent niveau k la factorisation approximative idéal d'un entier x. dans cette seconde partie nous verrons comment en déduire une facteur comme idéal, notre approche sera similaire à celle développé dans la première partie pour construire le diviseur commun en partant de la notion de diviseur idéal.

Les notions de relation d'ordre sont les même que ceux définit dans la première partie.

Soit $\boldsymbol{X} := \{\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2} \ldots . \boldsymbol{x_n}\}$ un ensemble d'entier distinct et ordonnés i.e. , $\boldsymbol{x_0} < \boldsymbol{x_1} < \boldsymbol{x_2} < \cdots < \boldsymbol{x_n}$.

Pour chacun de xi on considéré la factorisation initiale sous la forme $x_i = r_i^2 + e_i$ *avec* $r_i = [\sqrt{x_i}]$ *et* $e_i = x_i - r_i$

On pose $R := \{r_0, r_1, r_2 \ldots . r_n\}$ l'ensemble des racines des xi et $E := \{e_0, e_1, e_2 \ldots . e_n\}$ l'ensemble des niveaux d'erreur initiaux.

Lemme 6.1 (construction de la structure de niveau k) :
Soit $X := \{x_0, x_1, x_2 \ldots . x_n\}$ *et P un facteur de niveau k de* x_0 *ie il existe a tel que* $P = r_0 - a$ *et* $xo = P(r + a + k) + e$ *.*
On appel structure Naturelle de niveau k de X, l'ensemble $K := \{k_0, k_1, k_2 \ldots . k_n\}$ *ou ki est construit comme suit :*
Pour chaque xi

- *calculer* $a_i = r_i - P$
- *Calculer* $Q_i = \left[\frac{x_i}{P}\right]$
- *En déduire* $k_i = Q_i - (r_i + a_i)$

➔ *On peut en déduire l'ensemble des facteurs idéaux associé à la structure* $P = \{P_0, P_1, P_2 \ldots . P_n\}$ *et on pose* $P_{min} = min\{p_i \in P\}$ *et* $P_{max} = max\{p_i \in P\}$ *et ainsi* $\forall p_i \in P, P_{max} \leq p_i \leq P_{max}$

➔ *Nous pouvons aussi en déduire l'ensemble des générateurs de cette structure qui n'est rien d'autre que l'intersection de l'ensemble des générateurs de niveau ki (voir lemme 6.3) de chacun de xi. Notons que cet ensemble est non nul car il contient au moins* P_0.

Nous allons maintenant essayer de déduire de cet ensemble des facteurs idéaux le facteur commun idéal tout comme nous l'avons fait dans le cadre du diviseur idéal.

Notons premièrement que dans le cas simple ou tous les diviseurs sont identique, le facteur commun idéal est trivial et est égal à ce facteur consensuel.

Notons ensuite que dans le cas où il n'ya pas consensus il faudra forcer chacun des idéaux à un certain ajustement, la difficulté est que dans le cas présent (contrairement au cas du diviseur idéal) l'impact d'un ajustement unitaire ne produit pas une variation en terme d'erreur identique pour tous les idéaux, il y'en à qui sont plus sensible que d'autre a ces ajustement.

Def : soit $X = \{x_0, x_1, x_2 \ldots . x_n\}$ *et* $K = \{k_0, k_1, k_2 \ldots . k_n\}$ *une structure stable compatible avec X, on considéré l'ensemble* $P = \{P_0, P_1, P_2 \ldots . P_n\}$ *des facteurs idéaux de X pour la structure K. On appel* ***générateur idéal de la structure K****, l'élément P tel X' = (PQ') soit le plus proche possible de X au sens de la relation d'ordre choisi. Avec Q' donné par qi' = [xi/P]*

Construction du générateur idéal d'une structure K de X

Soit $X = \{x_0, .., x_n\}$ un ensemble d'entier, on pose $R = \{r_0, .., r_n | \ r_i = \lfloor \sqrt{x_i} \rfloor \}$

Soit $K = \{k_0, \dots, k_n\}$ une structure compatible avec X et $P = \{P_0, \dots, P_n\}$ l'ensemble des facteurs idéaux associé à la structure K (voir lemme 6.4) et $A = \{a_0, \dots, a_n \mid a_i = P_i - r_i\}$ les ajustements associé à chacun de facteurs.

On pose $P_{min} = min\{P_i \in P\}\ et\ P_{max} = max\{P_i \in P\}\ tels\ P_{min} < P_i < P_{max}\ \forall (P_i \in P)$

Def : Soit P un générateur de K on définit l'approximation X' de X par P comme la quantité $X' = P * Q'$ *avec* $Q' = \{Q'_0, \dots, Q'_n \mid Q'_i = r_i + a_i + k_i\ et\ a_i := P - r_i\}$

L'erreur d'approximation de X par X' est donné par $\delta(P) = max\left\{\frac{e_i}{x_i} \mid e_i = |x_i - x'_i| \ x_i \in X\right\}$

- ➔ Nous recherchons le générateur idéal de la structure K i.e. un facteur commun P des xi qui soit générateur de K qui minimise le niveau d'erreur δ d'approximation de X par X'.

Soit P un générateur de X, on pose $A' = \{a'_0, \dots, a'_n \mid a'_i = P - r_i\}$ l'ensemble des ajustements …

Alors

$\forall x_i \in X,\ x'_i = PQ'_i = P * (r_i + a'_i + k_i) = P * (r_i + a_i + k_i + n_i)\ ou\ n_i := a'_i - a_i$

- ➔ $x'_i = PQ'_i = (P_i + n_i) * (Q_i - n_i) = P_i * Q_i + n_i * (Q_i - P_i) - n_i^2$
- ➔ $x'_i - x_i = \varepsilon_i + n_i * (Q_i - P_i) - n_i^2$ ou ε_i est le niveau d'erreur minimal de niveau ki i.e celui de la factorisation par l'idéal : $\varepsilon_i = x_i - P_i * Q_i$
- ➔ $\delta_i^+(P) = \frac{|x'_i - x_i|}{x_i} = \left|\frac{\varepsilon_i + n_i*(Q_i - P_i) - n_i^2}{x_i}\right| = \left|\frac{\varepsilon_i}{x_i} + \frac{n_i*(Q_i - P_i)}{x_i} - \frac{n_i^2}{x_i}\right|$
- ➔ $\delta_i^+(P) = \left|\frac{\varepsilon_i}{x_i} + \frac{n_i*(2a_i - k_i)}{x_i} - \frac{n_i^2}{x_i}\right|$
- ➔

De même

$\forall x_i \in X,\ x'_i = PQ'_i = P * (r_i + a'_i + k_i) = P * (r_i + a_i + k_i + n_i)\ ou\ n_i := -a'_i + a_i$

- ➔ $x'_i = PQ'_i = (P_i - n_i) * (Q_i + n_i) = P_i * Q_i - n_i * (Q_i - P_i) - n_i^2$
- ➔ $\delta_i^-(P) = \left|\frac{\varepsilon_i}{x_i} - \frac{n_i*(2a_i - k_i)}{x_i} - \frac{n_i^2}{x_i}\right|$

Si on analyse le niveau d'erreur δ :

- Etant donné que ε est l'erreur sur l'idéal on peut supposer ε petit par rapport à $n_i * (Q_i - P_i)$ si c'est le cas, le niveau d'erreur sur l'approximation de xi dépend du rapport entre $Q_i - P_i$ et la quantité n_i. De façon générale n sera petit devant P – Q, se sera donc le rapport à x de cette distance entre les facteurs qui va porter l niveau d'erreur final. Ce niveau d'erreur va augmenter d'autant plus rapidement que cette distance sera grande.
  Par ailleurs k étant petit devant a on en déduit que c'est essentiellement la quantité 2a/x qui va définir la sensibilité au déplacement…

➔ Plus c'est deux quantité serons éloigné l'une de l'autre plus le taux d'erreur d'approximation de xi sera important.
➔ Sachant que la distance p-q augmente avec le niveau, lorsque le niveau k deviendra important c'est essentiellement cette distance qui va déterminer le niveau d'erreur.
➔ Les point pour l'lesquels l'ajustement n est négatif sont plus sensible au déplacement car le niveau d'erreur augment plus rapidement ?????

Autre formulation du niveau d'erreur

$$x_i' = PQ_i' \ \forall i \rightarrow \ \delta_i(P) := \frac{x_i - x_i'}{x_i} = 1 - \frac{PQ_i'}{x_i}$$

$$\rightarrow \ \delta_i'(P) := \frac{1-\delta_i(P)}{P} = \frac{Q_i'}{x_i} = \ \frac{Q_i}{x_i} \pm \frac{n_i}{x_i} \ \approx \frac{1}{P_i} \ \pm \frac{n_i}{x_i}$$

➔ L'impact du déplacement est d'autant plus/moins important que x l'est, il vaut donc mieux déplacer les idéaux des point de plus petite/grande valeur.

➔ *Def : On définit le taux de* ***dégradation maximale potentiel*** *du point x par :*
➔ *Soit* $x = P * Q + \ \varepsilon = (r - a)(r + a + k) + \ \varepsilon$ *la factorisation idéale de niveau k de x alors la quantité* $\varepsilon(x) = n\frac{2a}{x}$ *ou* $n = max(|P - Pmin|, |P - Pmax|)$ *ou* P_{min} *et* P_{max} *sont les bornes maximale et minimale de l'ensemble des facteurs idéaux.*
➔ *Ou plus simplement* $\varepsilon(x) = \frac{n}{x}$

Cas d'une structure à deux points :

Nous commençons par construire le facteur idéal dans le cas d'une structure à deux points, le cas plus général d'une structure à n points s'en déduit facilement en prenant comme points de référence les deux « extrémité » de l'ensemble de facteurs idéaux.

Soit $X = \{x_0 , x_1\}$, *un ensemble d'entier*, $x_i = r_i^2 + \varepsilon_i$,

soit $K = \{k_0 , k_1\}$ *une structure compatibleavec X et* $P = \{P_0 , P_1\}$ Les facteurs idéaux associés à cette structure. Avec $P_i = r_i + a_i$ on cherche le point $P = r_i + a'_i$ tel que

$\delta_0(p) = \delta_1(p)$ ou $\delta_i(p) := \frac{E_i(P)}{x_i} := \frac{(x_i - p*q_i)}{x_i} = 1 - P * \frac{q_i}{x_i}$ or $q_i = \left[\frac{x_i}{p}\right] = r_i + a'_i + k_i$ pour un certain ajustement a'.

$\delta_0(p) = \delta_1(p) \rightarrow \frac{r_0 + a'_0 + k_0}{x_0} = \frac{r_1 + a'_1 + k_1}{x_1}$

Par ailleurs on a si on suppose par exemple $P_0 < P_1$ (sinon il suffit d'inverser les rôles):

$a'_0 = a_0 + d_0$, $a'_1 = a_1 - d_1$ $et\ d_0 + d_1 = P_1 - P_0 := \Delta P \rightarrow d_1 = \Delta P - d_0$

Les quantités d_i représente le déplacement imposé à chacun des idéaux pour atteindre le consensus, qui survient lorsque les deux niveaux d'erreur sont identique (voir première partie pour les explications).

$\delta_0(p) = \delta_1(p) \rightarrow \frac{r_0 + a_0 + d_0 + k_0}{x_0} = \frac{r_1 + a_1 - \Delta P + d_0 + k_1}{x_1}$

$\rightarrow \frac{r_0 + a_0 + k_0}{x_0} + \frac{d_0}{x_0} - \frac{r_1 + a_1 - \Delta P + k_1}{x_1} - \frac{d_0}{x_1} = 0$

$\rightarrow \frac{d_0}{x_0} - \frac{d_0}{x_1} = \frac{r_1 + a_1 + k_1}{x_1} - \frac{r_0 + a_0 + k_0}{x_0} - \frac{\Delta P}{x_1} = \frac{Q_1}{x_1} - \frac{Q_0}{x_0} - \frac{\Delta P}{x_1}$

Si on pose $\boldsymbol{\Delta\delta := \frac{r_1 + a_1 + k_1}{x_1} - \frac{r_0 + a_0 + k_0}{x_0} - \frac{\Delta P}{x_1}}$

On en déduit $\boldsymbol{d_0 = \Delta\delta * \frac{x_1 * x_0}{x_1 - x_0}}$

On peut donc construire le facteur idéal P en déplaçant P_0 d'une distance égale à $[d_0]$:

$\boldsymbol{P = P0 + [d0]}$

➔ Si P n'est pas contenu dans l'ensemble de générateur de la structure alors le facteur idéal est égal à la borne des générateurs qui est la plus proche de P.

Cas d'un ensemble X quelconque

Rappelons que l'enjeu dans la recherche du facteur commun de minimiser le taux d'erreur maximale. Par ailleurs le niveau d'erreur induit en déplaçant le facteur idéal est directement proportionnel à la distance sur laquelle est déplacé cet idéal pour arriver au consensus. Aussi une première approximation consisterait en d'appliquer la démarche précédente aux deux bornes de l'ensemble des facteurs idéaux (qui sont les éléments les plus éloigné l'un de l'autre et qui de ce fait subirons le plus important déplacement de leur idéal). Mais il peut arriver que ce choix ne soit pas optimal en raison du fait que la dégradation des niveaux d'erreur par déplacement de l'idéal n'est pas identique pour l'ensemble des xi.

Nous allons donc discuter ici d'une méthode plus fine de choix des deux éléments auquel on doit appliquer la méthode précédente.

(Intuitivement on peut penser au point dont la dégradation potentielle est la plus élevé, couplé avec le point qui lui est le plus éloigné)

L'ensemble des facteurs idéaux de $X := \{x_0, x_1, x_2 \ldots . x_n\}$ *pour* $K := \{k_0, k_1, k_2 \ldots . k_n\}$ *est donné par* $P := \{p_0, p_1, p_2 \ldots . p_n\}$ *ou* $p_i = r_i - a_i$ *et* $a_i = \left[\frac{-k_i + \sqrt{{k_i}^2 - 4(k_i * r_i - e_i)}}{2} \right]$

….

Autre structure de niveau k

Une structure est dite de niveau k si son générateur est un facteur de niveau k de xo.

La structure naturelle construite précédemment n'est bien évidement pas l'unique structure de niveau k. en effet une modification du générateur peu entrainé une modification du niveau d'un des xi sans toutefois modifié le niveau de x0, les structures ainsi obtenu (par translation du générateur initiale) sont dite structure de second ordre.

Soit $K := \{k_0, k_1, k_2 \ldots . k_n\}$ *une structure de niveau* $k = ko$ *et* $P := \{P_0, P_1, P_2 \ldots . P_n\}$ *l'ensemble des facteurs idéaux qui lui sont rattachés.*

RQ1

Toutes les structures de niveau k ont en commun ko (le niveau du pivot xo est fixe, par définition du niveau de la structure) et par conséquent P0 est fait partie de l'ensemble des facteurs rattaché à toutes structure de niveau k. c'est donc un point fixe pour les structures de niveau k, nous en déduisons les conséquences pour la construction du facteur idéal de niveau k.

Lemme : Si K est la structure naturelle, alors les distance de chacun des point au facteur du pivot sont incompressible et induisent donc un taux d'erreur minimale sur les structure de ce niveau s = min{si | si s(Pi, P0)}

Point fixe

Quel que soit la structure de niveau k choisit, le facteur idéal Po de xo fera partie de l'ensemble des facteurs idéaux.

- ➔ La distance de chaque facteur à ce point fixe doit donc être tel que le niveau d'erreur implicite (si le point en question ainsi que le point fixe étaient le deux extrémité de l'ensemble des facteurs) soit inférieur au seuil toléré.
- ➔

Le générateur idéal de la structure est construit en considérant le point dont le potentiel de dégradation potentiel maximale (et le facteur idéal qui lui est le plus éloigné) et en lui appliquant le lemme …

Facteur idéal de niveau k

Lemme (construction du générateur idéal d'une structure donné)

Soit X et K

- *Construire l'ensemble P = {pi} des facteurs idéaux associé à chacun des couples (xi,ki).*
  Poser ri = R(xi). Calculer ai = f(xi, ki) en déduire Pi = ri-ai.
- *On considère l'ensemble des niveaux d'erreur ei associé à chacun de ces facteurs (ei = xi – pi*[xi/pi]).*
- *Ordonné l'ensemble des pi du plus petit au plus grand. En déduire Pi-Min et Pi-max les facteurs minimaux et maximaux de cet ensemble.*
- *Construire l'ensemble D des taux d'erreur maximaux associé à chacun des élément. Di est donné par : pour le point Pi, poser di la distance qui le sépare du point qui lui est le plus éloigné. (du moins le min entre cette distance et la distance maximale permettant de conserver le niveau k)*
- *On considère l'ensemble des pas des …*

Theoreme (construction du générateur idéal de niveau k)

P0 p1 pn

Intervalle de centre

Intervalle de gauche

Intervalle de droite

Rq

Pour un x fixé, Plus k est grand, plus De est grand. En fait l'erreur accumule un delta de second niveau qui est quasi constant…il es tres lentement perturbé par un second niveau d'erre de signe opposé qui apparait a chaque changement de k, ceux-ci étant de plus en plus fréquent, le mouvement fini par s'inverser et la seconde perturbation prend le control…

ALGORITHME DE CALCUL DU FACTEUR IDEAL DE NIVEAU K

Algo : construction du générateur ideal de niveau k
Soit X={} un ensemble d'entier distinct et ordonné x0<x1<…xn
Soit k un entier fixé.

- *Construction du générateur naturelle P d'ordre k qui n'est rien d'autre que le diviseur idéal de niveau k du pivot xo.*
- *Déduire la structure naturelle implicite de niveau k :*
    - *Construire l'ajustement de l'élément xi pour le facteur P : ai = P-ri*
    - *Construire le cofacteur qi de P pour xi : qi = [xi/P]*
    - *Déduire le niveau ki de Pi : ki = qi-(ri+ai)*
- *Déduire l'ensemble de facteur idéaux des xi pour la structure K ={ki}*
  Pi = f(xi,ki)
- *Construire la générateur idéal de la structure k.*
    - *Ordonné l'ensemble P des facteurs idéaux*

- *Pour chacun des pi, calculer le niveau d'ajustement maximale compatible : ai+ = f(xi, ki+1) et ai- = f(xi, ki-1)*
- *En déduire l'ensemble des facteurs compatibles avec la structure k : ie l'intersection de [Pi-ai- ;Pi +ai+]*
- *Considérer les deux « extrémité » de l'ensemble des facteurs ie les points les plus éloigné en prenant en compte leur erreur par unité ei.*
- *Calculer la facteur idéal de la structure P = f(P-, P+, ei)*
- *En déduire le niveau d'erreur de la structure naturelle d.*

- *Explorer les structures de second ordre :*
  - *Ordonné l'ensemble A des ajustement de la structure.*
  - *Pour chacun des points ajouter le tag ajustable ou non :*
- *Un point est non ajustable si le niveau d'erreur incompressible consécutif à son ajustement est supérieur au niveau initial d de la structure naturelle.*
- *On appel que le point Po étant un point fixe pour toutes les structures de niveau k ; chaque ajustement a provoqué un déplacement de même ampleur sur cet idéal et induit donc un niveau d'erreur… aussi tout ajustement supérieur au niveau d'ajustement provoquant une erreur supérieur à l'ajustement max, est un ajustement qui ne peut être envisagé.*
- *De mm étant donné que le point po est fixe tout ajustement provoquant un ajustement de ko est non compatible. (de même que tout ajustement supérieur…)*
- *Pour la partie A' des ajustements compatibles avec ko et d faire :*

ALGORITHME DE CALCUL DU PGCD PAR FACTORISATION APPROXIMATIVE

Dans cette partie nous présentons l'algorithme de construction du Pgcd P d'un ensemble X d'entier lorsque celui-ci est inférieur à la racine du plus petit élément de X mais pas « trop » éloigné de celui-ci.

Soit $X := \{x_0, x_1, x_2 \ldots . x_n\}$ *on suppose X ordonné.*

On pose R = {} et E = {}

On définit la fonction $fi(x, k)$ *qui pour un entier x et un niveau k donné associe le facteur idéal de x au niveau k, définit comme suit. (Calculer par le lemme 6.4)*

$P := fi(x, k) = r - a$

ou $a = \left[\frac{(-k+\sqrt{\Delta})}{2}\right]$ *ou* $\Delta = k^2 + \mathbf{4}(kr - e)$

On définit aussi la fonction inverse de fi qui à un couple x, P associe le niveau k du facteur p.

$k := fi^{-1}(x, P) = Q - (2r - P)$ *ou* $Q := \left[\frac{x}{P}\right]$

Poser x = pivot(X) := min{xi} = xo

Pour k :=0 jusqu'à k= k_{max} faire :

1. ***Poser ko :=k et Déterminer le facteur idéal P0 de niveau ko de xo (lemme 6.4):***

   $P_0 := fi(x_0, k_0)$

2. ***Construire la structure naturelle*** $\boldsymbol{K := \{k_0, k_1, k_2 \ldots . k_n\}}$ ***générée par P0***

   $k_0 = k$

   $k_i := fi^{-1}(x_i, P_0)$

3. ***Déduire l'ensemble*** $\boldsymbol{P := \{P_0, P_1, P_2 \ldots . P_n\}}$ ***des facteurs idéaux associé à K.***

   $P_i := fi(x_i, P_i)$

4. ***En déduire le générateur idéal PI de la structure K.***

   (voir algo plus loin)

5. ***Déterminer le générateur idéal des structures de niveau k.***

   (voir algo plus loin)

6. ***Mettre à jour le Pgcd Courant PC***

   Si niveau d'erreur de PI inférieur à niveau d'erreur de PC alors PC := PI ;

   Sinon conserver PC et continuer.

7. ***Tester les critères d'interruption et retourner le pgcd.***
   - Niveau de fractionnement maximale atteint k =k_{cible}
   - Ou taux d'erreur cible atteint : $\delta = \delta_{cible}$

   ➔ Retourner PC. Et interrompre la recherche.

8. ***Déterminer le niveau d'analyse suivant***

   K := k+1 et recommencer à l'étape 1

Calcul de Kmax

Comme on peut le constater l'apport marginal de chaque itération est dégressif et deviendra quasi nul au-delà d'une certaine valeur. Il s'agit ici de calculer la valeur ka au-delà de laquelle l'apport marginal de chaque itération devient trop faible..

Algorithme de construction du générateur idéal d'une structure donnée

1. Déterminer le borne de l'ensemble de facteurs idéaux
   Pmin =
   Pmax =
2. Calculer la taux d'erreur maximal de chacun des points
3. Considérer le point d'erreur max et sont opposé calculer le point optimal..

Partie 3

DIVISION EQIVALENTE ET PGCD APPROXIMATIF

DIVISION EQUIVALENTE *(recherche en cours…)*

Dans cette dernière partie nous nous intéressons au calcul du Pgcd lorsque celui-ci est très inférieur à la racine du plus petit élément. En effet dans la première partie nous avons utilisé la division approximative pour rechercher le pgcd lorsque celui-ci était très supérieur à la racine du pivot, la seconde partie en s'appuyant sur le concept de factorisation approximative à construit un procédé de recherche du pgcd autour de la racine du pivot, il nous reste donc à traiter le cas ou celui-ci est significativement inférieur à la racine du pivot.

AUTRE APPROCHE

La méthode de factorisation décrite dans la première partie de cet article permettait de rechercher les facteurs d'un nombre au voisinage de sa racine, la structure stable sur l'laquelle repose cet algorithme n'apparait qu'au voisinage de cette racine, plus loin le cout k de la factorisation évolue plus rapidement que le facteur p, ce qui rend la méthode totalement inefficace…

Dans cette seconde parie nous nous intéressons au cas ou on s'éloigne de la racine, cette partie se caractérise par le fait que p et q ne sont plus de taille relativement proche, mais q deviens un multiple de p i.e $\exists n, a \in N \ tel\ que\ q = np + a\ a < p$

Notre but ici est pour un n enter fixé d'analyser la zone de recherche des facteurs ou on peut écrire q sous la forme précédente.

Soit $x \in N : x = pq + e\ avec\ q = np + a$ on cherche les points $q' = np' + a'\ \ avec\ p' = p - b$

$x = PQ + e\ avec\ Q = nP + a \rightarrow \boldsymbol{x = nP^2 + aP + e}$

On en déduit la factorisation au point P'

$P' \coloneqq P - b \rightarrow P = P' + b \rightarrow \ \ x = n(P' + b)^2 + a(P + b) + e$

$x\ = n[P'^2 + 2bP + b^2] + aP + ab + e\ = nP'^2 + (2nb + a)P' + (nb^2 + ab + e)$

On en déduit la factorisation de x sous la forme $x = n'P'^2 + a'P' + e'$ avec :

On pose :

$\delta(b) = \left[\frac{nb^2+ab+e}{P-b}\right]\ et\ \alpha(b) = \left[\frac{2nb+a+\ \delta(b)}{P-b}\right]$

$e' = \ (nb^2 + ab + e) - \delta(b)$

$a' = \ 2nb + a + \ \delta(b) - \ \alpha(b)$

$n' = n + \alpha(b)$

P' est un facteur exact de x ssi $e' = 0\ i.e.\ \ nb^2 + ab + e\ est\ un\ multiple\ de\ P'$

$P' \mid x \leftrightarrow \exists \lambda \in N \mid nb^2 + ab + e = \lambda(P - b) \rightarrow nb^2 + (a + \lambda)b + (e - \lambda P)$

.le point b ou le niveau d'erreur est minimale est donc donné par :

$\Delta := (a + \lambda)^2 - 4n(\varepsilon - \lambda P)\ et\ b := \left[\frac{-(a+\lambda)+\sqrt{\lambda^2+\lambda(2a+4nP)+(a^2-4ne)}}{2n}\right]$

Si on itère sur les valeurs de λ on obtient les points d'erreur minimal pour q =nP.

STRUCTURE

Soit x et p tel que q = np+a

On déduit pour chacun des xi de X un niveau ni = xi/p^2

La structure N = {ni} est notre structure stable

La difficulté ici est de déterminer le niveau générateur idéal d'une telle structure t même déjà le facteur idéal pour chacun des xi au niveau ni, ceci viens du fait que si la structure est stable les quantités a et e ne le sont pas et l'évolution de a influence directement le niveau d'erreur, et même si on sait que pour n fixé, la quantité a + e est strictement croissante, il n'est pas simple d'isoler la composante purement erreur de cette somme.

Partie 3

ALGORITHME ET COMPLEXITE

Chapitre 9

ALGORITHME ET ANALYSE DE LA COMPLEXITE

Dans ce dernier chapitre nous allons nous pencher de façon plus détaillé sur l'analyse de l'algorithme de calcul des Pgcd.
Le but est donc d'utiliser les résultats des chapitres précédents pour construire un algorithme efficace de calcul.

Nous rappelons que l'ensemble des données d'entrée est un ensemble X = {x0,….}.
L'algorithme commence par une phase d'initialisation dans laquelle on procède aux opérations suivantes.

- ➔ Arrondir les données et supprimer les doublons.
- ➔ Fixer le pivot égal au plus petit élément de X. Dans notre étude nous avons choisi le plus petit élément mais notons qu'il est envisageable de choisir d'autre éléments que celui-ci, par exemple pour le calcul ….

Comme nous l'avons établi, du moment que nous connaissons l'ordre du Pgcd rechercher, nous pouvons en déduire de façon certaine le Pgcd. Il n'est donc pas étonnant que pour notre recherche de Pgcd, la boucle principal consiste en une itération sur les différents ordres, en partant de l'ordre 1, jusqu'à un maximum.

Algorithme

Soit $X = \{x_0, x_1, \dots, x_n\}$ l'ensemble des données dont on recherche le PGCD approximatif. On suppose que tous les éléments de X sont différents (sinon on supprime les doublons)

On note P_0 le pivot de X ($P_0 = Min\{x_i, x_i \in X\}$) Alors le PGCD approximatif que nous recherchons est un élément de l'ensemble S = [1, P_0].

De nombreuse optimisations peuvent être ajouté à la méthode présenté ici les détails sont disponible dans FOT2012.

1. *Ensemble des solutions et Pivot*

La fonction GetPivot(X) retourne le pivot qui sera utilisé pour la construction du PGCD, ici elle est tout simplement égale au plus petit élément de X.

Entier P0 = GetPivot(Vecteur X)

{Retourne P0 = Min(X) }

2. Construction de la structure naturelle d'ordre n

Construction du pivot d'ordre n : ***Reel $P_n := P_0/n$.***

Construction de la structure naturelle d'ordre n :

Vecteur Qn = BuildStructure(Vecteur X_input, Reel P_gen)

{Pour chacun des éléments de X calculer

Entier qi := Arrondi(xi/P_gen)

Retourne Qn := {qi} }

3. Déterminer l'ensemble des points de rupture

Construction des points de rupture à droite du pivot d'ordre n

Vecteur G+ = ComputeRigthBreakPoints(Vecteur X_input, Vecteur Q_struc)

{Pour chacun des éléments de X calculer

Réel Pi = xi/qi

*Réel bi := Pi *(1-1/(2* qi -1))*

Retourne G+ := {bi} }

Construction des points de rupture à gauche du pivot d'ordre n

Vecteur G-= ComputeLeftBreakPoints(Vecteur X_input, Vecteur Q_struc)

{Pour chacun des éléments de X calculer

Réel Pi = xi/qi

*Réel bi := Pi *(1-1/(2* qi +1))*

Retourne G- := {bi} }

Fixation des limites de l'ordre et de la structure naturelle

Reel OrderGenMin = (x_0/q_0)(1-1/(2* q_0-1))*

Reel OrderGenMax = (x_0/q_0)(1-1/(2* q_0+1))*

Reel StrucGenMin = Min{G+}

Reel StrucGenMax = Max{G-}

4. Calculer le générateur idéal de la structure

Construire l'ensemble des diviseurs idéaux de la structure

Vecteur Pn = BuildStructureIdeals(Vecteur X_input, Vecteur Q_structure)

{Pour chacun des éléments de X calculer

Réel Pi := xi/qi

Retourne Pn := {Pi} }

Calculer le générateur idéal.

Entier Gn = ComputeIdeal(*Vecteur P_ideal)*

{Reel Pmin = Min{ P_ideal}

Reel Pmax = Max{ P_ideal}

Reel Gn = (2 Pmax* Pmin)/ (Pmax+ Pmin)*

Reel AjustGn = Gn

Si (Gn > StrucGenMax) Alors AjustGn = StrucGenMax

Si (Gn < StrucGenMin) Alors AjustGn = StrucGenMin

Retourne Arrondi(AjustGn)}

Calculer le taux d'erreur induit par Gn

Réel δn = ComputeErrorRate(Vecteur X_input, Entier G_gene)

{ Pour chacun des éléments de X calculer

*, δi = 1 – (G_gene/x)*Arrondi(x/G_gene)*

Δ : = Vecteur(|δi |)

Retourne Max(Δ)}

5. Construire les structures secondaires de droite/gauche

Déterminer les générateurs des structures secondaires d'ordre n

Vecteur G = GetSndStrucGen(Vecteur G-_Breaks, Vecteur G-_Breaks)

{Pour chacun des éléments de G+ et G- faire

Si (G+(i) < OrderGenMax) Ajouter G+(i) G.

Si (G-(i) < OrderGenMin) Ajouter G-(i) G.

Retourne G }

NB : de nombreuses structures secondaires peuvent être exclu de facto de l'analyse, voir plus haut pour l'étude des méthodes de rejet.

6. *Déterminer les idéaux et niveau d'erreur des structures secondaire*

Pour chacun des éléments de G reprendre l'étape 4.

Algorithme : DETAIL CONSTRUCTION

Initialisation :

- ➔ Sélectionner et préparer les données
- ➔ Introduire un simplificateur (élément neutre)

Boucle :
Pour ordre = 1 jusqu'à ordre = maxOrdre faire

- ➔ Déterminer le générateur
- ➔ Initialiser la description des points
- ➔ Initialiser la description de l'ordre
- ➔ Calculer le Diviseur commun de l'ordre
- ➔ Mettre à jour les résultats
- ➔ Tester la condition d'arrêt
- ➔ Recommencer à l'étape 1 ou déterminer l'ordre suivant.

Préparation des donnée

- Supprimer les doublons
- Supprimer les éléments non significatifs (égal à 0 ou inférieur à un seuil donné)
- Arrondir les éléments à l'entier le plus proche et prendre leur valeur absolue.

Introduire un « fragilisateur »

Nous pouvons commencer notre algorithme par une étape préalable. Il s'agit ici de rajouter un élément qui n'aura aucun impact sur le choix du PGCD, mais permettra de réduire de façon significative la zone de recherche.

Dans le cas du PGCD classique, nous avons construit (Lemme 3.5) nous avons construit l'élément T à partir des résidus issue de la structure d'ordre 1.

Dans le cas des PGCD approximatif, …

Déterminer le générateur initial

Le générateur est donné par le rapport du pivot sur l'ordre, arrondi à l'entier le plus proche.
OrderGen = [Pivot/order]

Initialisation de la description des points

Une fois le générateur construit, nous pouvons entamer la description des points sur laquelle s'appuiera la construction du générateur idéal.

Nous regroupons ces données sur un objet **PointDescription**, identifié par la valeur xi (PointValue) du point. Ses champs sont les suivants :

- **Quotient : = [PointValue/OrderGen]**
- **Approximation := OrderGen*Quotient**
- **Error : = PointValue – Approximation**

- **NegAjust : = ((OrderGen - 2* Error)/(2* Quotient +1)) + 1**
- **PosAjust : = ((OrderGen + 2* Error)/(2* Quotient -1)) + 1**

- **Ideal := [PointValue/Quotient]**
- **MinErrorRate := (PointValue - Ideal *Quotient)/ PointValue**

- IsKo

Cout : *la construction de la description d'un point implique 15 Opération arithmétique de bases. Si donc notre ensemble X compte N entrées, l'initialisation se fera pour chaque ordre étudier en maximum 15N opération.*
Le coup est linéaire et fonction du nombre d'entrée.

Initialiser la Description de l'ordre

Pour construire le générateur idéal de l'ordre nous allons exploiter un vecteur contenant la description des points présenté précédemment. L'objet **OrderDescription** permet de regrouper l'ensemble des données utile au traitement d'un ordre.

Nous disposons ici d'un vecteur contenant l'ensemble des objets PointDescription construit précédemment.

- GenMin = OrderGen - ***Min***{ NegAjust}
- GenMin = OrderGen - ***Min***{ PosAjust}

- DivMin = ***Min***{ Ideal }

➔ DivMax = ***Max***{ Ideal }

➔ DivMed = [(2.*DivMin*DivMax)/(DivMin + DivMax)]
➔ ErrRateMax := ***Max***{**(PointValue -** DivMed ***Quotient)/ PointValue** }

***Cout** : l'initialisation de la description d'un ordre nécessite :*
5 + 3N opérations.
1 recherche double de MinMax pour fixer les MinMax des générateurs et idéaux : cette opération nécessite un parcourt des N description des points et 4 comparaison et éventuelles assignations pour chacun des tours de boucle.
Bien sur l'initialisation de chaque ordre implique au préalable la description des points présentée précédemment.
Le cout est linéaire en fonction du nombre N d'entrées.

Calculer le diviseur commun de l'ordre

Le calcul du diviseur commun de l'ordre consiste en la recherche de la structure offrant l'intervalle des diviseurs commun le plus compact.

Notons que dans le cadre du diviseur commun, il n'est pas nécessaire de procéder à cette optimisation.
Plusieurs autres caractéristiques peuvent permettre de passer outre cette étape.

➔ Ordonner le vecteur de points par ordre croissant des ajustements (Positif/négatif suivant le type de compression)
➔ Parcourir le vecteur ordonné, et calculer le diviseur idéal final :
IdealFinal := [PointValue/[PointValue/(OrderGen+Ajust)]]} Ssi le remplacement de l'idéal précèdent par ce nouvel idéal entraine un resserrement de l'intervalle des idéaux.
Sinon conserver l'idéal précédent pour cet élément et pour tous les suivant.

Critères d'interruption précoce de la construction

Rappelons que la construction de la description d'un ordre n'as d'intérêt que tan que nous pensons qu'il peut améliorer le niveau d'erreur courant. Aussi devons-nous interrompre cette construction dès que nous avons suffisamment d'éléments montrant que le générateur idéal qui en résultera ne nous permettra pas d'atteindre le niveau d'erreur cible ou d'améliorer le niveau d'erreur courant (dans le cas du δ-PGCD) dans cette partie nous allons nous pencher sur les différents tests permettant de décider de cette interruption.

- Niveau d'erreur du diviseur idéal :
  Nous avons vu (lemme) que le niveau d'erreur du générateur final était toujours supérieur au niveau d'erreur de chacun de diviseurs idéaux. Aussi si le niveau d'erreur d'un des diviseurs idéaux est supérieur au niveau d'erreur cible ou au niveau d'erreur courant, et qu'en plus aucun ajustement du quotient associé n'est

possible … alors on peut directement interrompre la construction de la description de l'ordre et passé à l'ordre suivant.

- Impossibilité d'ajustement des MinMax des diviseurs.
  Lorsque les quotients rattachés aux diviseurs minimaux et maximaux ne sont pas ré-ajustable, le générateur idéal est le générateur idéal initial, il n'est donc pas nécessaire d'exécuter l'étape précédente de recherche de l'intervalle des diviseurs le plus compact.

Mise à jour des résultats

L'algorithme consiste en une recherche du diviseur idéal produisant le niveau d'erreur minimale. Ainsi le générateur courant ne pourra remplacer le précèdent que si le niveau d'erreur résultant est inférieur au niveau courant.

Une fois les étapes précédentes terminées, nous disposons d'un nouveau couple de candidat (DivMed, ErrRateMed) pour le Pgcd, celui si sera évalué par rapport au précédent et le pgcd mis à jour.

➔ Comparer ErrRateMed avec CurrErr, CurrGcd.
➔ Si ErrRateMed < CurrErr : remplacer l'ancien couple par le nouveau.

***Cout**: cette mise à jour est effectuée au maximum une fois pour chaque Ordre et implique une comparaison et éventuellement deux assignations.*
***Le coup est donc constant pour chacun des ordres traités**.*

Décision de poursuite

Les conditions d'arrêt de l'algorithme dépendent du type de Pgcd rechercher.

- Dans le cas du PGCD classique, l'algorithme se termine lorsque ***ordre = Racine(pivot).***
- Dans le cas du PGCD d'ordre α, l'algorithme se termine lorsqu'es atteint l'ordre maximal fixé. la condition d'arrêt est : ***ordre = α***.
- Dans le cas du δ-PGCD, l'algorithme s'interrompt lorsque le taux d'erreur cible est franchi. La condition d'arrêt est **δ < δ-cible**.
- Dans le cas du PGCD d'ordre α ou de précision δ, les deux conditions précédentes (sur l'ordre et sur la précision) sont vérifiées la première qui est remplie interrompt l'algorithme.

***Cout**: cette décision qui est prise à la fin de la construction de la description de chaque ordre implique un test afin de vérifier si les conditions d'arrêt sont remplies, il s'agit soit de comparer l'ordre à un maximum soit de comparer le taux d'erreur à une cible ou les deux éventuellement. Soit maximum deux opérations pour chacun des ordres étudier.*
***Le coup est donc constant pour chacun des ordres traités**.*

Détermination de l'ordre suivant

Lorsque la décision de poursuite est prise, il faut déterminer l'ordre suivant avant de relancer la construction. Les ordres ne sont pas toujours parcourus de façon linéaire de 1 jusqu'au maximum.

La suite des ordres sur lesquels on itère dépends essentiellement du type de PGCD auquel on s'intéresse.

Nous avons vu jusqu'ici que le coup de construction du générateur idéal pour un ordre donnée était assez faible (linéaire en fonction du nombre d'entrée). Le cout final de notre algorithme dépendra donc essentiellement de notre capacité à réduire au minimum possible le nombre d'ordre qui seront étudiés afin de calculer le PGCD, et donc notre capacité à déterminer avec un coup minimal l'ordre suivant ayant un potentiel d'amélioration du niveau d'erreur courant.

PGCD classique

Ici on ne s'intéresse qu'aux ordres qui sont diviseurs du Pivot.
L'ordre suivant (NextOrder) est donc le prochain diviseur du Pivot entre l'ordre courant et la racine du pivot. Si la racine du pivot est franchi l'ordre suivant est le pivot lui-même et le PGCD vaut 1.

Un ordre est jugé utilisable si :

- Le rapport (Pivot/Ordre) est entier.

Cette approche rend le coup de l'algorithme très important dans le cadre du PGCD, nous pouvons l'améliorer en exploitant les résultats sur le fractionnement afin de déterminer le prochain ordre à envisager.

PGCD d'ordre α

L'un des grands intérêts du PGCD d'ordre α est que le cout est maitrisé, et le nombre d'itération maximal qui vaut α est connu à l'avance.

La recherche de l'ordre suivant (NextOrder) se fait entre l'ordre courant et l'ordre maximal (Alpha)

Un ordre est jugé utilisable si :

- Il garantit un taux d'erreur sur le pivot inférieur au taux d'erreur courant.

Le δ-PGCD

Contrairement au PGCD d'ordre α, le cout ici ne peut être anticipé, il est intrinsèquement lié au niveau de précision que l'on souhaite obtenir et à l'ordre permettant de corriger les données pour obtenir un PGCD d'une telle précision.

La recherche de l'ordre suivant (NextOrder) se fait entre l'ordre courant et l'ordre maximal ici égal au Pivot.

Un ordre est jugé utilisable si :

- Il garantit un taux d'erreur sur le pivot inférieur au taux d'erreur cible.

PGCD d'ordre α ou de précision δ

Le problème posé sous cette forme permet de mieux maitriser le coup lié à une recherche de δ-PGCD, en rajoutant un ordre maximal. On a alors une double condition portant à la fois sur l'ordre et sur la précision.
La recherche de l'ordre suivant (NextOrder) se fait entre l'ordre courant et l'ordre maximal (Alpha)

PARALLELISATION

De nos jours la parallélisassions es devenu un des principaux enjeux en algorithmique. Les principaux centres de calcul disposant désormais de grille de calcul importante, la capacité à utiliser de façon optimale cette ressource est un critère de plus en plus important dans le choix de l'algorithme à utiliser pour résoudre un problème donné.

L'un des grands intérêts de notre approche pour la résolution des problèmes de calcul des PGCD est que sa parallélisassions est quasi trivial. En effet nous pouvons tirer parti du fait que l'évaluation des diviseurs commun des différents ordres est quasi indépendante des autres. Aussi avons-nous la possibilité d'exécuter en parallèle ces recherches de diviseur commun.

La parallélisassions de l'algorithme se réduit donc quasiment à la transformation de sa boucle principale (sur les ordres), en une boucle distribué, partageant les calculs sur les processeurs disponible. (Avec la nouvelle norme C++, ou des outils de parallélisassions tel Intel parallèle studio, cette mise en parallèle de boucle, ne demande rien plus que l'ajout d'un mot clé).

La seule difficulté sera pour la question des PGCD approximatif, de tirer parti de la précision déjà atteinte par les éléments d'ordre supérieur, et de vérifier la condition d'arrêt.

Les différents calculs qui s'exécutent en parallèle doivent pouvoir mettre à jour une variable commune contenant le niveau de précision atteint. Cependant l'accès à cette variable doit être quelque peu réglementé. En effet étant donné que le PGCD est définit comme le générateur idéal d'ordre maximal vérifiant certaine conditions, il est important que la mise à jour de cette variable se fasse suivant les ordres i.e. l'élément

d’ordre 1 dois mettre à jour la variable avant que l’élément d’ordre 2 le fasse,…ect. Si un tel ordonnanceur est construit pour régler l’accès à cette ressource, il suffira après chaque mise à jour de vérifier le seuil attient et de le comparer à la cible. Notons que le lancement du calcul parallèle de nouveaux ordres sur les processeurs disponible se fera toujours avec le seuil courant, et que les ordres n’ayant pas réussi à améliorer le niveau d’erreur reçu en entrée ne se mettront pas dans la file d’accès au seuil car ils ne pourront pas l’améliorer.

Une seconde approche peut consister en un accès libre à la mise à jour du seuil, le premier à terminer son exécution, le met à jour, vérifie la condition d’arrêt. Si le seuil cible est atteint, le processus ne se termine pas mais tous les appels aux ordres inférieur à celui ayant permis d’atteindre le seuil cible sont stoppés. Seul se poursuivent les appels d’ordre supérieur afin de vérifier qu’il n’existe pas d’élément d’ordre supérieur permettant d’atteindre le seuil cible, auquel cas c’est plutôt ce dernier qui est le PGCD retenu.

Cette possibilité de parallélisassions simple et efficace augmente de façon très importante l’efficacité de l’algorithme.

CONCLUSION ET PISTES DE RECHERCHE

Nous avons construit dans cet article les bases de la δ-arithmétique. Dans cette généralisation de l'arithmétique classique, les propriétés ne sont plus envisagées comme des caractéristiques figées apparaissant de façon spontanée. Au contraire la δ-arithmétique s'emploie à démontrer qu'en général, elles émergent d'un processus continu qui trouve son aboutissement dans l'émergence de la propriété recherché (coïncidence de l'idéal de la δ-arithmétique, avec la propriété correspondante en arithmétique classique).

Par ailleurs la δ-arithmétique envisage toujours un élément et son voisinage comme un tout solidaire. Afin de prendre en compte cette vision intégrant les éléments dans leur voisinage, l'un de nos principaux outils a été l'identification des structures stables. Celles – ci représentent des sorte d'invariant partagé entre un élément et ses voisins et permettant donc d'analyser le voisinage comme un tout dont on peut directement extraire les informations nécessaire au calcul des quantités recherchés.

La suite de notre recherche portera sur la recherche et l'exploitation de nouvelle structure stables. Nous nous intéresserons aussi au structure quasi stable, celles-ci contrairement au première ne restent pas quasiment inchangé pour un plage de valeur considéré, mais leur évolution est suffisamment lente pour que leur valeur initiale puisse servir d'approximation de la valeur sur des plages d'une certaine taille.

Le premier succès de cette nouvelle approche a été de permettre une clarification de certain problème d'arithmétiques tel qu'ils se posent dans les environnements réel. Contrairement au monde mathématique abstrait et idéal fait dans lequel émerge les concepts fondamentaux d'arithmétique, monde fait de certitudes, d'exactitude, de précisions, de rigueur, … les environnements réel aux quels se confrontent les ingénieurs sont fait de d'incertitude, d'imprécision, et marqué par une certaine flexibilité, dont on use souvent pour simplifier ou adapter les données aux problèmes à résoudre. La δ-arithmétique se veut donc dans ce sens, une sorte d'arithmétique de l'ingénieur, s'adaptant aux problématiques quotidiennes rencontrées par ces derniers.

Cette nouvelle approche à démontré son efficacité en nous proposant de nouveaux algorithme pour la résolution de grand problèmes d'arithmétique Nos prochains travaux porteront donc sur les renforcements des bases de cette nouvelle extension de l'arithmétique. Nous continuerons d'exploité ce nouveau framework pour résoudre les problèmes d'arithmétique rencontrée au quotidien par les ingénieurs.

REFERENCE

[1] N. Howgrave-Graham. Approximate integer common divisors. In Cryptography and Lattices,pages 51–66. Lecture Notes in Computer Science 2146. Springer-Verlag, Berlin, Heidelberg, 2001.

[2] H.Cohn et N. Heninger . Approximate common divisors via lattices.

[3] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan : Fully Homomorphic Encryption over the Integers. in Advences in cryptology - ….

[7] I. Z. Emiris, A. Galligo, H. Lombardi, *Certifed approximate univariate GCDs*, J.Pure Appl. Algebra 117/118, 229-251 (1997).

18] V. Y. Pan, *Numerical computation of a polynomial GCD and extensions*, Information and Computation 167, 71-85 (2001).

[19] A. SchÄonhage, *Quasi-GCD Computations*, J. Complexity, 1, 118-137 (1985).

Annexe :