Évitabilité de puissances additives en combinatoire des mots

Florian Lietard

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Évitabilité de puissances additives en combinatoire des mots

THÈSE

présentée et soutenue publiquement le 11 décembre 2020

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention Informatique)

par

Florian Liétard

Composition du jury

Président :	Sylvain Contassot-Vivier
Rapporteurs :	Michaël Rao Michel Rigo
Examinateurs :	Valérie Berthé Anne de Roton Anna Frid
Directeurs de thèse :	Damien Jamet Thomas Stoll

Institut Élie Cartan de Lorraine — UMR 7502
Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Résumé

Ce document est principalement consacré à l'étude de l'évitabilité des cubes additifs dans les points fixes de morphismes. Les problèmes d'évitabilité des puissances additives sont connus pour avoir des implications dans la théorie des semi-groupes. Depuis un article publié en 2013 par J. Cassaigne, J.D. Currie, L. Schaeffer et J.O. Shallit, nous savons qu'il est possible de construire sur l'alphabet $\{0, 1, 3, 4\}$ un mot infini qui évite les cubes additifs, i.e., un mot qui ne contient pas trois facteurs consécutifs de même taille et même somme. Nous commençons par expliquer notre démarche de recherche avec cet article comme point de départ et par discuter des similarités entre les différents morphismes permettant d'éviter les cubes additifs sur des alphabets de 4 lettres. Nous expliciterons la façon dont nous avons programmé en `C++` la recherche de morphismes créant des mots infinis sans puissances additives.

Nous étendons ensuite la preuve de l'article de 2013 à un ensemble infini de morphismes (correspondant à une classe d'équivalence). Après l'étude d'un exemple nous développons une démonstration basée sur des substitutions entre les morphismes.

Le résultat principal de ce document est qu'il est possible de donner, sur chaque alphabet de 4 lettres qui ne soit pas une transformation affine de $\{0, 1, 2, 3\}$, un morphisme explicite possédant un point fixe infini sans cubes additifs. Ce travail, effectué en collaboration avec Matthieu Rosenfeld, a donné lieu à la publication d'un article. Pour démontrer ce résultat nous utilisons un argument contenu dans l'article de 2013, des majorations effectuées après disjonction des cas et des arguments de symétries entre les alphabets.

Enfin, nous nous intéressons à l'évitabilité des cubes additifs sur $\{0, 1, 2, 3\}$ afin de tenter de répondre à une question posée en 2018 par M. Rao et M. Rosenfeld. Cet alphabet est le seul de quatre lettres pour lequel notre résultat principal n'apporte pas de réponse. Nous représentons graphiquement les mots possédant des puissances additives et nous développons deux programmes informatiques parallélisés. Le premier permet de détecter efficacement les puissances additives dans un mot très long, le second permet de créer un mot de plus de 70 millions de lettres sans cube additif sur $\{0, 1, 2, 3\}$. Nous améliorons ainsi significativement la précédente borne connue (1.4×10^5). Pour parvenir à ce résultat nous inversons périodiquement l'ordre de priorité pour le choix des lettres dans la construction du mot comme nous le suggéraient les représentations graphiques. Nos programmes utilisent des approches à la fois multi-ordinateurs et multi-threads pour gagner en efficacité.

Abstract

The present thesis is dedicated to the various aspects of the problem of avoiding additive cubes in the fixed points of morphisms. Problems concerning the avoidability of additive powers are closely related to questions in the theory of semigroups. Since the publication of the article of J. Cassaigne, J.D. Currie, L. Schaeffer and J.O. Shallit (2013) we know that it is possible to construct an infinite word over $\{0, 1, 3, 4\}$ that avoids additive cubes, i.e., a word that avoids three consecutive blocks of same size and same sum. We first explain the methods used by the authors in their article, and then use it as a starting point for our discussions, with the ultimate aim to clarify the various similarities and connections between the different morphisms that allow to avoid additive cubes on alphabets over 4 letters.

We next discuss our implementation in C++ of the investigation of theses morphisms, and then proceed to give an infinite family of morphisms (corresponding to classes of equivalence) that avoid additive cubes. After this investigation, we give a general proof scheme that is based on substitutions between morphisms.

The main result of the thesis is that for any alphabet of 4 letters, with the sole exception of the alphabet $\{0, 1, 2, 3\}$ and its affine transformations, there is an explicit morphism whose infinite fixed point avoids additive cubes over the given alphabet. This work has been carried out in collaboration with Matthieu Rosenfeld and gave rise to an article in a peer-reviewed journal. In order to show this result, we use arguments from the article of Cassaigne et al., several numerical estimates for the underlying quantities, case disjunction as well as symmetry considerations for the alphabets under consideration.

In the final part of the thesis we then study the question of avoiding additive cubes over $\{0, 1, 2, 3\}$ with the hope to answer a question posed by M. Rao and M. Rosenfeld in 2018. This alphabet is the only 4-letters alphabet where our result does not apply. We first study by graphical means the words that contain additive powers, and we discuss and implement in a second step two parallelized computer programs. The first program detects in an efficient way the occurrence of additive powers in very long words, whereas the second one allows to create long words over $\{0, 1, 2, 3\}$ without introducing any additive cube. With the help of these programs, we obtain a word of over 70 million letters that avoids additive cubes over $\{0, 1, 2, 3\}$. This largely improves on the former known bound (1.4×10^5). To obtain our result, as suggested by our graphical considerations, we periodically reverse the order of priority for the choice of the letters in the construction of our word. Our programs use multi-computers and multi-threads settings to gain considerably on efficiency.

Remerciements

Dans toute cette aventure qui a duré presque quatre ans et demi, j'ai eu le chance de toujours pouvoir compter sur un tissu social et familial d'une qualité sans égale. Parce que sans vous tous je n'aurais jamais pu vivre tout ceci, je tiens à vous remercier du fond du cœur, en espérant n'oublier personne.

Avant tout je tiens à remercier ma famille : Maman, Papa, Thibault, Bastien, Jean Pierre, Nanou, Mamie. En plus de tout ce que je vous dois pour mon parcours et mes études, vous m'avez toujours offert un endroit où me ressourcer, un chez moi à quelques centaines de kilomètres de Nancy dans lequel je pouvais venir piocher une énergie nouvelle pour repartir ensuite dans le tumulte de la ville. Je vous aime et vous me manquez.

Diane, ma chérie, je ne saurais jamais te dire à quel point tu m'as aidé durant toutes ces années, sans compter, sans jamais juger, sans faillir. Tu m'apportes au quotidien tant de choses merveilleuses, tant d'évasion et de bonheur, toi qui ne m'a jamais connu autrement qu'en doctorat, Merci d'être toujours là, j'aurais tant à dire qui ne tiendrait pas sur les 100 pages de cette thèse.

Je remercie également mes deux familles d'adoption Nancéiennes. La première, ma belle-famille, toujours présente pour un repas de détente ou pour bouger des meubles. Merci à Constance, Théo, Mylène, Sylvain, Christine et Thierry. La deuxième, la troupe des Fines Lames de Stanislas. Merci à vous pour tous ces moments de pur bonheur entre camaraderie, fêtes, spectacles et combats. Un merci particulier à Bertrand et Isabelle, à Amandine et Loïc mes partenaires de combat favoris, à Pascal pour sa bienveillance permanente et son talent, à Fred, à Léa pour nos évasions théâtre et littérature et pour tout le reste, à Noëlle qui m'a permis de souffler autant que j'en avais besoin dans la période de rédaction.

Je veux également remercier les doctorants, ceux de l'I.E.C.L et du Loria, évidemment je pense en premier à Matthieu B. et Dimitry du 513, mais aussi à Clémence et Maria pour nos discussions. Merci également à ceux qui venaient d'ailleurs. Matthieu R., merci pour cet article, c'était un plaisir d'écrire avec toi. Je pense aussi à toute l'équipe des chercheurs du B37 de Liège, merci d'avoir rendu mes conférences aussi inoubliables : Marie, Adeline, Célia, Émilie. Et surtout Manon, merci vraiment pour tout ce que tu as fait. Tu as été et tu restes un exemple à suivre pour moi.

Évidemment il y a les amis proches que je n'ai pas encore cités, ceux qui sont déjà des amis de toujours ou ceux qui vont le devenir. Un gigantesque merci, du fond du cœur, à Marion, Alyzée et Faustine, bien plus que des amies vous faîtes partie de ma famille, je vous aime toutes les trois et vous me manquez. Un merci immense à Pixy, Léa et Orlane, vous rayonnez du bonheur sur tous vos amis et j'ai de la chance de vous avoir dans mon entourage. Un énorme merci à Justine, Émeline, Raphaël et Dafné, vous m'avez tant apporté. Un merci Vendôme 30 gras pour Pauline, pour ton amitié et pour m'avoir permis de réaliser dans ton imprimerie la page de garde de ma thèse. Et merci à tous ceux et celles que je ne peux pas citer exhaustivement mais qui ont compté pour moi dans ces quatre années, vous vous reconnaîtrez, merci à vous.

Je tiens également à remercier Cécile Dartyge et Julien Cassaigne pour avoir suivi

le déroulement de ma thèse pendant quatre ans. Je remercie chaleureusement Valérie Berthé, Sylvain Contassot-Vivier, Anne de Roton et Anna Frid, c'est un privilège et un plaisir de vous avoir dans mon jury. Plus particulièrement je remercie Michel Rigo et Michaël Rao, vous avoir comme rapporteur de ma thèse est un honneur. J'adresse également un grand remerciement à la Fédération Charles Hermite et la Région Grand-Est pour le financement de ma thèse.

Enfin, un immense merci à Damien et Thomas pour m'avoir guidé dans cette aventure avec autant de bienveillance, d'humour, de rigueur scientifique et de compréhension. Vous formez un duo de directeurs exceptionnel et j'espère que de nombreux doctorants auront la chance de vous avoir pour encadrants. Je ne pourrai jamais oublier ces années de doctorat, merci à vous !

Table des matières

Liste des Figures

Liste des Tableaux

Liste des Algorithmes

Chapitre 1

Introduction et notations

Les notations et les définitions utilisées dans ce chapitre et dans la suite sont regroupées en Section 1.3. S'agissant pour la plupart des notations habituelles de la combinatoire, on pourra se référer à l'ouvrage de M. Lothaire [34]. L'organisation de ce document est expliquée en Section 1.5.

1.1 Introduction à la combinatoire des mots

La combinatoire des mots est un domaine qui a connu un nouvel essor après les travaux d'Axel Thue au début du XX^{e} siècle. Une des questions principales de ce domaine est celle de l'évitabilité de certains motifs : étant donné un motif particulier, est-il possible, sur un alphabet fini, de construire un mot infini qui évite ce motif (i.e., qui ne contient pas ce motif) ? Si la réponse est positive, on pourra alors chercher la taille du plus petit alphabet sur lequel il est possible de réaliser une telle construction et la façon optimale de la réaliser (un morphisme le plus simple possible par exemple).

On désigne par le mot *alphabet* tout au long de ce document un ensemble fini d'éléments appelés *lettres*. Cet ensemble peut être un ensemble de lettres au sens usuel du terme comme $\{a, b, c\}$ ou un sous-ensemble de $\mathbb{Z}$ comme $\{-14, 1, 2, 12\}$. On pourra également considérer des sous-ensembles finis de $\mathbb{R}$, de $\mathbb{Z}^2$ ou d'autres ensembles mais nous considérons toujours des alphabets finis. Si notre alphabet est constitué de symboles auxquelles nous décidons d'attribuer une valeur (par exemple considérer $\{b, f, t\}$ comme l'alphabet $\{2, 6, 20\}$ en fonction de la place des lettres dans l'alphabet latin) on parle de *valuation* des lettres ou de lettres *valuées*. Un *mot* est alors la donnée d'une suite finie ou infinie d'éléments de l'alphabet considéré. Un *facteur* d'un mot est un sous-mot fini constitué de lettres consécutives du mot. Par exemple, *matique* est un facteur du mot *informatique* et du mot *mathématiques*. Étant donné un alphabet fini $\mathcal{A}$, l'ensemble des mots non vides sur $\mathcal{A}$ est noté $\mathcal{A}^+$. Le mot vide est noté ε, on note $\mathcal{A}^*$ l'ensemble des mots finis sur $\mathcal{A}$.

Une question d'évitabilité très naturelle est celle des carrés : est-il possible sur un alphabet fini de construire un mot infini qui ne contienne pas deux facteurs consécutifs identiques ? En Figure 1.1 nous montrons qu'il est impossible sur un alphabet constitué

de deux symboles de construire un mot sans carré de longueur plus grande que 3. Un carré dans un mot y apparaît en rouge.

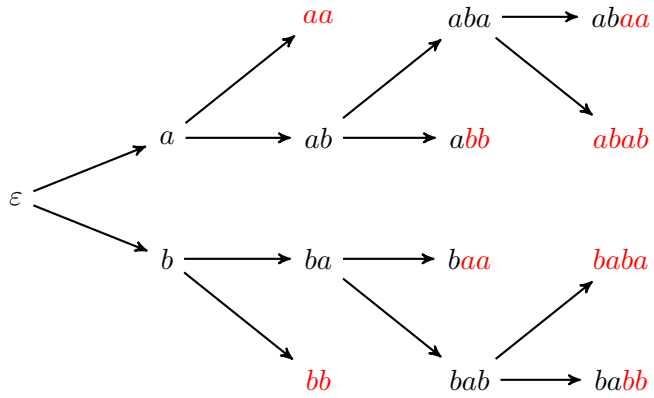


Figure 1.1 – Tous les mots de taille 4 sur $\{a, b\}$ contiennent un carré.

Montrer qu'un mot (même infini) contient un motif donné (par exemple un carré) est un problème décidable. Il suffit de parcourir le mot jusqu'à rencontrer le motif donné, ce qui se fait forcément dans un temps fini même si possiblement très long. En revanche pour montrer qu'un mot infini ne contient pas un motif il est nécessaire de trouver une méthode pour se ramener à un problème fini. Les résultats d'évitabilité que nous présentons dans l'état de l'art (voir Section 1.2) utilisent tous un morphisme pour engendrer le mot infini et les démonstrations utilisent les caractéristiques du morphisme pour démontrer que ce mot ne contient pas un certain motif.

1.2 État de l'art

1.2.1 Éviter les carrés et les cubes

Le problème de l'évitabilité des cubes est une version moins restrictive du problème des carrés : existe-t-il un mot infini sur un alphabet fini qui ne contienne pas consécutivement trois facteurs identiques ? De manière générale, on appelle *k-puissance* une suite de k facteurs identiques consécutifs. Si un mot évite les carrés, il évite nécessairement les cubes. La réciproque n'étant pas vraie, l'ensemble des mots sans carré est strictement inclut dans l'ensemble des mots sans cube. A. Thue [42, 43] montre qu'il suffit d'un alphabet de deux lettres pour construire un mot infini sans cube. Il utilise le morphisme $\varphi_1 : \{0,1\}^* \to \{0,1\}^*$ défini par

$$\varphi_1(0) = 01 \quad \text{et} \quad \varphi_1(1) = 10,$$

et montre que le mot infini

$$\mathbf{w_1} = \overrightarrow{\varphi_1^\omega}(0) = \lim_{n\to\infty} \varphi_1^n(0) = 0110100110010110100101100110100 1\cdots$$

ne contient pas de cube. Ce mot est obtenu comme limite des itérations du morphisme sur la lettre 0. Il s'agit d'un point fixe infini du morphisme φ_1. La construction par itération d'un tel mot est expliquée au début de la Section 1.4. Une définition complète de la limite et de la topologie associée se trouve dans l'ouvrage de M. Lothaire [34, Section 1.2]. Ce mot infini est appelé suite de Prouhet–Thue–Morse, de très nombreux articles de la littérature lui sont consacrés (voir[2]).

De même, A. Thue montre qu'il est possible de construire un mot sans carré sur un alphabet de 3 lettres en appliquant une transformation bien choisie au mot w_1. Comme ce mot ne contient pas de cube, il ne contient aucun facteur 111. Deux 0 dans $\mathbf{w_1}$ sont donc séparés par 0, 1 ou 2 lettres 1. En reportant la suite de ces nombres on obtient $210201210120210\cdots$ et A. Thue montre que ce mot ne contient pas de carré. Étant donné que les carrés ne sont pas évitables sur 2 lettres, il obtient ainsi le plus petit alphabet possible. Notons que les articles de A. Thue ont été initialement publié en allemand. C'est J. Berstel [9] qui en fournit une retranscription en anglais. Il donne également une large revue de la littérature sur l'étude des mots sans carré [8].

Ces résultats étant obtenus grâce à des morphismes et les preuves étant *ad hoc*, il est intéressant de se demander dans quelle mesure on peut prouver que le point fixe infini d'un morphisme (si ce dernier en possède un) évite des puissances. J. Berstel [7] montre en 1979 qu'on peut décider si le point fixe infini d'un morphisme sur 3 lettres évite ou non les carrés. Six ans plus tard, M. Leconte [32] démontre qu'il est également possible de décider si un morphisme est sans puissance. Un morphisme sans puissance est un morphisme qui transforme un mot dépourvu de k-puissance en un mot sans k-puissance.

1.2.2 Éviter les puissances abéliennes

En 1961, P. Erdős [23, 24] établit une liste de problèmes non résolus dans laquelle figure la question suivante : est-il possible, sur un alphabet fini, de construire un mot infini qui ne contienne pas de carré abélien (i.e., deux facteurs consécutifs identiques à permutation près) ? Cette question peut être étendue à celle de l'évitabilité des k-puissances abéliennes, c'est-à-dire savoir s'il est possible ou non de construire un mot infini qui ne contienne pas k facteurs consécutifs identiques à permutation les lettres près. En 1968, A.A. Evdokimov [25] est le premier à apporter une réponse positive sur un alphabet de taille 25. Deux ans plus tard, P.A.B. Pleasants [36] améliore ce résultat : il montre qu'il est possible de construire un mot infini sans carré abélien sur un alphabet de 5 lettres. La Figure 1.2 montre qu'il n'est pas possible sur un alphabet $\{a, b, c\}$ de taille 3 de construire un mot de longueur plus que 7 sans carré abélien. Le choix des lettres étant totalement symétrique, on suppose dans cette figure que le mot construit commence par un a. Pour des raisons de concision, nous faisons apparaître en rouge les mots qui ne peuvent pas être prolongés et nous ne représentons pas les mots contenant des carrés abéliens. La question est posée par T.C. Brown [13] en 1971 de savoir s'il est possible ou non de restreindre la construction d'un mot sans carré abélien à un alphabet de 4 lettres. P.A.B. Pleasants conjecture déjà que cette construction est possible grâce à des tests informatiques sur des mots de longueur 1600.

La réponse à la question de l'évitabilité des carrés abéliens sur 4 lettres est donnée

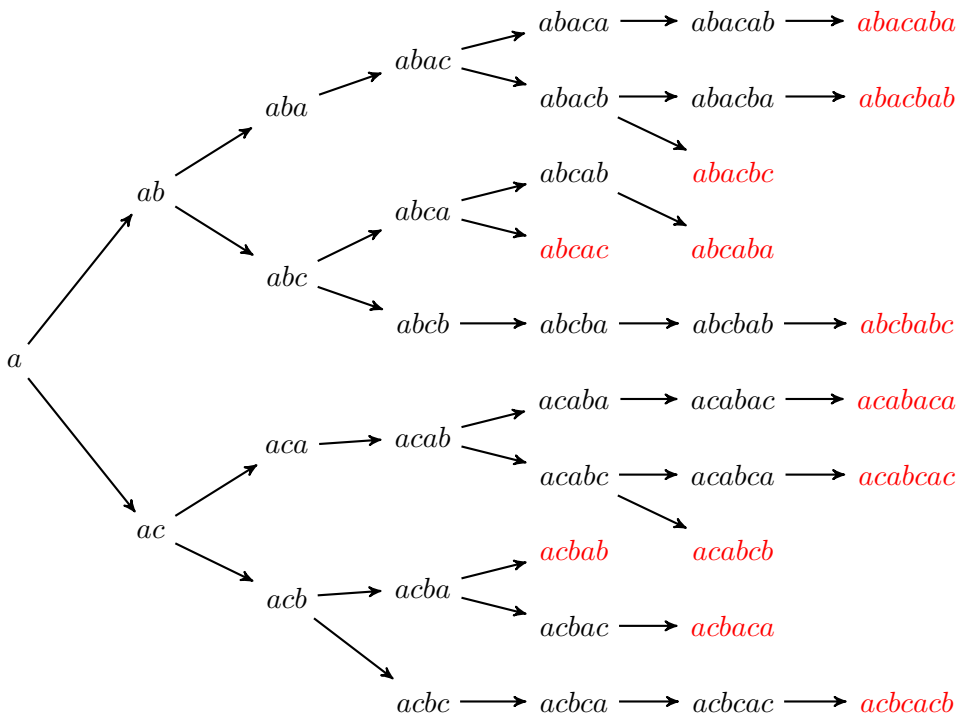


Figure 1.2 – Tous les mots commençant par la lettre a sans carré abélien sur $\{a, b, c\}$.

par V. Keränen [31] en 1992. Il utilise le morphisme, que nous notons φ_2, défini par

$$\varphi_2(a) = abcacdcbcdcadcdbdabacabadbabcbdbcbacbcdcacba-$$
$$bdabacadcbcdcacdbcbacbcdcacdcbdcdadbdcbca$$

et les images de b, c et d sont obtenus en appliquant à $\varphi_2(a)$ la permutation circulaire $(a\ b\ c\ d)$. Ce type de morphisme est appelé morphisme cyclique. Le point fixe infini de φ_2 est dépourvu de carré abélien. V. Keränen prouve que ce morphisme, de taille 85, est le plus petit morphisme cyclique permettant d'obtenir un point fixe infini évitant les carrés abéliens. Quelques années auparavant, F.M. Dekking [20] avait prouvé qu'il est possible sur un alphabet de 2 lettres de construire un mot infini qui évite les 4 puissances abéliennes grâce au point fixe infini du morphisme $\varphi_3 : \{0,1\}^* \to \{0,1\}^*$ défini par

$$\varphi_3(0) = 011 \quad ; \quad \varphi_3(1) = 0001.$$

De même il montre que le morphisme $\varphi_4 : \{0,1,2\}^* \to \{0,1,2\}^*$ défini par

$$\varphi_4(0) = 0012 \quad ; \quad \varphi_4(1) = 112 \quad ; \quad \varphi_4(2) = 022,$$

possède un point fixe infini qui évite les cubes abéliens. Ces deux résultats sont optimaux en terme de taille d'alphabets étant donné qu'il n'est pas possible d'éviter de tels motifs

sur moins de deux lettres et que les cubes abéliens ne sont pas évitables sur deux lettres (on construit l'arbre des possibilités de la même façon que précédemment et il ne se prolonge pas au delà de la taille 9).

Comme dans le cas des carrés et des cubes, les preuves sont *ad hoc* mais des travaux ultérieurs nous permettent de répondre plus simplement dans certains cas à la question de savoir si un mot évite ou non les puissances abéliennes. Par exemple Y.H. Au, A. Robertson et J.O. Shallit [4] ont montré que si dans un mot infini la différence entre les fréquences de deux lettres est bornée par une constante alors ce mot n'évite pas les carrés abéliens. Pour les mots générés par des morphismes, J.D. Currie et N. Rampersad [19] ont prouvé que, si le morphisme remplit certaines conditions, la question de savoir si son point fixe évite ou non les puissances abéliennes est décidable. Nous pouvons citer également l'article de A. Carpi [15] qui fournit une condition suffisante pour qu'un morphisme préserve l'évitement des puissances abéliennes dans les mots, et l'article de M. Rao et M. Rosenfeld [38] dans lequel ils étudient des variations des problèmes d'évitabilité des puissances abéliennes.

1.2.3 Éviter les puissances additives

En 1972, J. Justin [29, 28] introduit la notion de semi-groupe k-répétitif. Un semi-groupe est un ensemble muni d'une loi de composition interne associative.

Définition 1.2.1. *Un semi-groupe S est k-répétitif si pour tout alphabet $\mathcal{A}$, pour tout morphisme $\varphi : \mathcal{A}^+ \to S$ et pour tout mot $w \in \mathcal{A}^+$ assez long, il existe un facteur $w_1 \cdots w_k$ tel quel*

$$\varphi(w_1) = \cdots = \varphi(w_k).$$

Le vocabulaire que J. Justin emploie est celui de la combinatoire, même s'il travaille dans le contexte des semi-groupes : *"on pourra appeler X alphabet, ses éléments lettres et les éléments de XX^* mots"*. Ce sont G. Pirillo et S. Varricchio [35] qui vont introduire la notion qui lie les semi-groupes à une question d'évitabilité de motifs encore non-résolue.

Définition 1.2.2. *Un semi-groupe S est uniformément k-répétitif si pour tout alphabet $\mathcal{A}$, pour tout morphisme $\varphi : \mathcal{A}^+ \to S$ et pour tout mot $w \in \mathcal{A}^+$ assez long, il existe un facteur $w_1 \cdots w_k$ tel quel*

$$\varphi(w_1) = \cdots = \varphi(w_k),$$

et tous les facteurs $w_1, \ldots, w_k$ sont de même taille.

Ils démontrent que le fait de savoir si $\mathbb{N}^+$ est uniformément k-répétitif permettrait de savoir si tout semi-groupe uniformément k-répétitif de type fini est fini. Or F.M. Dekking [20] a prouvé qu'il existait un mot infini sur $\{0,1\}$ et un morphisme de $\{0,1\}^+$ dans $\mathbb{N}$ (la somme des lettres) qui permettent de monter que $\mathbb{N}^+$ n'est pas uniformément k-répétitif. La question devient alors : peut-on construire, sur un alphabet fini contenu dans $\mathbb{N}$, un mot infini qui évite les k-puissances additives, i.e., k facteurs consécutifs de même taille et dont la somme des lettres est la même ? La réponse est donc oui pour $k = 4$ avec l'alphabet $\{0,1\}$. En fait, cette question avait déjà été soulevée par T.C. Brown et

A.R. Freedman [14] en 1987. En étudiant les progressions arithmétiques, ils ont conjecturé qu'il n'était pas possible de construire un mot infini sans carré additif (une 2-puissance additive, respectivement un cube additif est une 3-puissance additive). Cette question a encore été posée, de façon indépendante, par L. Halbeisen et N. Hungerbühler [27]. Ces derniers montrent que, si on retire la condition sur la taille, la réponse à la question est négative : il n'est pas possible de construire sur un alphabet fini un mot infini qui n'a jamais deux blocs consécutifs de même somme.

En 2014, J. Cassaigne, J.D. Currie, L. Schaeffer et J.O. Shallit [16] démontrent que le morphisme $\varphi_0 : \{0,1,3,4\}^* \to \{0,1,3,4\}^*$ défini par

$$\varphi_0(0) = 03 \quad ; \quad \varphi_0(1) = 43 \quad ; \quad \varphi_0(3) = 1 \quad ; \quad \varphi_0(4) = 01$$

engendre un point fixe infini qui évite les cubes additifs. Il est obtenu comme la limite d'itérations du morphisme sur la lettre 0 :

$$\mathbf{w_0} = \overrightarrow{\varphi_0^\omega}(0) = \lim_{n\to\infty} \varphi_0^n(0) = 0314301103434303101101103143034343034343\cdots$$

Ils prouvent donc que $\mathbb{N}^+$ n'est pas uniformément 3-répétitif. En 2015, M. Rao [37] montre qu'il est en fait possible de construire un mot infini sans cube additif sur un alphabet de 3 lettres en appliquant une substitution de taille 25 au mot $\mathbf{w_0}$. Cependant, il est important de noter que, pour les puissances additives, les alphabets ne sont pas équivalents. Rao montre que les cubes additifs sont évitables sur $\{0,1,5\}$ et sur $\{0,1,2,4\}$, qui est, dans l'ordre lexicographique, un alphabet plus petit que $\{0,1,3,4\}$. Les techniques qu'il utilise ne permettent cependant pas de trouver un mot infini sans cube additif sur $\{0,1,2,3\}$. Il parvient à construire un mot de taille 1.4×10^5 sans cube additif sur cet alphabet et laisse ouverte la question de l'existence d'un mot infini avec les mêmes caractéristiques.

Le résultat de J. Cassaigne et al. engendre plusieurs conséquences. Tout d'abord il permet à M. Rao de diminuer la taille de l'alphabet comme nous venons de le dire. J. Justin [30] utilise également ce résultat pour en déduire de nouvelles propriétés sur les semi-groupes. En 2018, M. Rao et M. Rosenfeld [39] s'inspirent du schéma de preuve utilisé pour créer un algorithme plus général permettant, sous certaines conditions, de décider si le point fixe infini d'un morphisme évite ou non des motifs (des puissances abéliennes, additives, etc.). Pour réaliser cette généralisation ils ont recours aux matrices de Jordan et à l'utilisation des *templates*, un outil déjà utilisé par J.D. Currie, A. Aberkane et N. Rampersad [1, 18, 19]. M. Rao et M. Rosenfeld démontrent également que les carrés additifs sont évitables sur $\mathbb{Z}^2$. Ils utilisent pour cela le morphisme $\varphi_5 : (\{0,1,2\}^2)^* \to (\{0,1,2\}^2)^*$ défini par

$$\varphi_5 : \begin{cases} \begin{pmatrix}0\\0\end{pmatrix} \mapsto \begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}2\\1\end{pmatrix}\begin{pmatrix}2\\0\end{pmatrix} & \begin{pmatrix}1\\1\end{pmatrix} \mapsto \begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} \\ \begin{pmatrix}2\\1\end{pmatrix} \mapsto \begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} & \begin{pmatrix}0\\1\end{pmatrix} \mapsto \begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}2\\1\end{pmatrix} \\ \begin{pmatrix}2\\0\end{pmatrix} \mapsto \begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}2\\0\end{pmatrix} & \begin{pmatrix}1\\0\end{pmatrix} \mapsto \begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}2\\1\end{pmatrix}\begin{pmatrix}2\\0\end{pmatrix}. \end{cases}$$

En 2020, F. L. et M. Rosenfeld démontrent, en utilisant des arguments contenus dans l'article de 2014 de Cassaigne et al., qu'il est en fait possible d'éviter les cubes additifs sur tous les alphabets de 4 lettres qui ne sont pas une transformation affine de $\{0, 1, 2, 3\}$, ce dernier cas restant ouvert.

Le problème de l'évitabilité des carrés additifs est également encore ouvert à l'heure actuelle. Un article de J. Cassaigne, G. Richomme, K. Saari et L.Q. Zamboni [17] montre l'équivalence entre l'évitabilité des carrés additifs et le fait de pouvoir construire, sur un alphabet fini, un mot infini dans lequel les positions qui permettent d'éviter les carrés abéliens apparaissent avec un écart borné. Une réponse partielle à la question des carrés additifs est donnée par A.R. Freedman et T.C. Brown qui montrent que si un mot de taille au moins 50 est construit sur un alphabet $\{a, b, c, d\}$ vérifiant $a + d = c + b$, alors il contient un carré additif. Enfin, M. Rosenfeld [41, Section 3.3.2] établit que s'il existe un mot infini sur un alphabet A évitant les longs carrés abéliens, alors on peut éviter les carrés additifs sur un alphabet contenu dans $\mathbb{Z}^{|A|-1}$. Cependant, les longs carrés abéliens ne sont pas évitables sur un alphabet binaire [22] et on ne peut donc pas utiliser cet argument pour conclure.

D'autres résultats ont été obtenus sur des variations du problème des carrés additifs [21, 12]. Par exemple T. Brown [11] montre que tout mot infini possède des facteurs arbitrairement grands "proches" des carrés abéliens (i.e., deux facteurs consécutifs de même taille et dont les sommes diffèrent d'une valeur bornée par une constante ne dépendant que de l'alphabet). Les questions d'évitabilité des puissances additives amènent également H. Ardal, T. Brown, V. Jungić et J. Sahasrabudhe [3] à introduire la notion de complexité additive. Ils prouvent notamment que si un mot infini a une complexité additive bornée alors il possède des k-puissances additives pour tout entier $k > 1$. Peu après, G. Banero [5] publie un article dans lequel il étudie en détail la complexité additive dans les mots infinis et notamment les effets des morphismes sur les mots à complexité additive bornées.

La combinatoire des mots étant un domaine large il ne se limite pas à l'évitabilité des motifs, nous renvoyons le lecteur à l'article de J. Berstel et D. Perrin [10] ou à celui de M. Rigo [40] pour un aperçu.

1.3 Notations et définitions

Nous donnons dans cette partie les définitions et les notations qui nous seront utiles pour les chapitres suivants [1]. On pourra se référer à l'ouvrage de Lothaire [34] pour les notations usuelles. Étant donné un mot $w \in \mathcal{A}^*$, on note $|w|$ la taille de w, c'est-à-dire le nombre de lettres qui le composent. Pour tout $a \in \mathcal{A}$ on note $|w|_a$ le nombre d'occurrences de la lettre a dans w. Pour toute position $p \in [\![0, |w| - 1]\!]$ on désigne par $w[p]$ la p-ième lettre de w en commençant à compter à partir de 0. Par exemple si $w = python$ alors $w[0] = p$ et $w[3] = h$. Pour $q \in [\![p + 1, |w|]\!]$, on note $w[p, q)$ le facteur de w situé entre la position p incluse et la position q non incluse. Avec notre exemple, $w[1, 4) = yth$ et $w[0, 1) = w[0] = p$. Notons que pour toute position p dans w, on a $w[p, p) = \varepsilon$.

1. L'index situé à la fin de cet ouvrage permettra au lecteur de retrouver rapidement les définitions.

Définition 1.3.1. *On appelle* préfixe *d'un mot w un élément de l'ensemble*

$$P(w) = \{w[0,p) : \ p \in [\![0, |w|]\!]\}.$$

On appelle suffixe *de w un élément de l'ensemble*

$$S(w) = \{w[p,|w|) : \ p \in [\![0, |w|]\!]\}.$$

Si le préfixe (respectivement suffixe) considéré est différent de w, il est appelé préfixe propre *(respectivement* suffixe propre*).*

Introduisons maintenant les outils qui nous permettent d'utiliser des outils d'algèbre linéaire sur les mots générés par des morphismes.

Définition 1.3.2. *Soit un alphabet $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. On appelle* vecteur de Parikh *d'un mot $w \in \mathcal{A}^*$ le vecteur $(|w|_{\alpha_1}, |w|_{\alpha_2}, \ldots, |w|_{\alpha_n})^T \in \mathbb{Z}^n$. On notera*

$$\begin{aligned} \psi : \mathcal{A}^* &\longrightarrow \mathbb{Z}^n \\ w &\longmapsto (|w|_{\alpha_1}, |w|_{\alpha_2}, \ldots, |w|_{\alpha_n})^T, \end{aligned}$$

l'application qui envoie un mot sur son vecteur de Parikh.

Définition 1.3.3. *Soit φ un morphisme sur un alphabet $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$ de taille n. Sa* matrice d'incidence *est notée* $\mathtt{Mat}(\varphi)$ *et est définie par*

$$\mathtt{Mat}(\varphi) = (|\varphi(\alpha_j)|_{\alpha_i})_{i,j \in [\![1;n]\!]}.$$

Étant donné w un mot, on obtient la propriété : $\psi(\varphi(w)) = M\psi(w)$.

Il est donc possible d'étudier les itérations d'un morphisme via les puissances de sa matrice d'incidence.

Définition 1.3.4. *Soient M et N deux matrices carrées de taille n, on dit qu'elle sont des* matrices semblables *s'il existe P une matrice inversible de taille n telle que $M = P^{-1} \times N \times P$.*

Soient $\mathcal{A}_1$ et $\mathcal{A}_2$ deux alphabets finis et soient deux morphismes $\varphi_1 : \mathcal{A}_1^* \to \mathcal{A}_1^*$ et $\varphi_2 : \mathcal{A}_2^* \to \mathcal{A}_2^*$. On pose les définitions suivantes.

Définition 1.3.5. *On dit que φ_1 et φ_2 sont* similaires *si et seulement si leurs matrices d'incidence sont semblables. Autrement dit, φ_1 et φ_2 sont similaires si et seulement s'il existe une matrice inversible P telle que $P^{-1} \times \mathtt{Mat}(\varphi_1) \times P = \mathtt{Mat}(\varphi_2)$. Cette relation est une relation d'équivalence.*

Les propriétés suivantes sont des résultats généraux sur les matrices semblables.

Propriété 1.3.6. *1. Les matrices d'incidence de deux morphismes similaires admettent les mêmes valeur propre.*

2. *Si* $\varphi_1 : \mathcal{A}_1^* \to \mathcal{A}_1^*$ *et* $\varphi_2 : \mathcal{A}_2^* \to \mathcal{A}_2^*$ *sont similaires, alors* $\mathcal{A}_1$ *et* $\mathcal{A}_2$ *sont des ensembles de même cardinal.*

Définition 1.3.7. *On dit que* φ_1 *et* φ_2 *sont substitution l'un de l'autre s'il existe une substitution lettre à lettre* $s : \mathcal{A}_1 \to \mathcal{A}_2$*, telle que* $s^{-1} \circ \varphi_2 \circ s = \varphi_1$*. Cette relation est une relation d'équivalence.*

Définition 1.3.8. *On dit qu'un morphisme* φ *défini sur un alphabet* $\mathcal{A}$ *est de taille* k *si*

$$k = \max_{l \in \mathcal{A}} |\varphi(l)|.$$

On obtient de façon immédiate les propriétés suivantes.

1. Deux morphismes substitutions l'un de l'autre sont des morphismes similaires. La réciproque n'est pas vraie car la matrice de passage entre les deux morphismes n'est pas nécessairement celle d'une substitution lettre à lettre.
2. Deux morphismes substitutions l'un de l'autre sont de même taille.

Définissons maintenant plus précisément les notions de puissances, de puissances abéliennes et additives. Étant donnés un alphabet fini $\mathcal{A} \subset \mathbb{C}$ et un mot $w \in \mathcal{A}^*$, on note $\sum w$ la somme des lettres de w vues comme des nombres complexes.

Définition 1.3.9. *Soient* k *un entier strictement positif,* $\mathcal{A} \subset \mathbb{C}$ *un alphabet fini et soit* $w \in \mathcal{A}^*$*. Supposons que* w *contienne un facteur* u *constitué de* k *facteurs non vides :* $u = w_1 \cdots w_k$*. Alors* u *est*

- *une* k-puissance *si* $w_1 = \cdots = w_k$*.*
- *une* k-puissance abélienne *si* $\psi(w_1) = \cdots = \psi(w_k)$*, où* ψ *est l'application de la Définition 1.3.2.*
- *une* k-puissance additive *si* $|w_1| = \cdots = |w_k|$ *et* $\sum w_1 = \cdots = \sum w_k$*.*

Les puissances seront parfois nommées puissances pures pour éviter toute ambiguïté. Les 2*-puissances (respectivement abéliennes et additives) sont appelées des carrés (respectivement carrés abéliens et carrés additifs). De même les* 3*-puissances (respectivement abéliennes et additives) sont appelées des cubes (respectivement cubes abéliens et cubes additifs).*

Remarque 1.3.1. *Sur des alphabets de deux lettres* $\{a, b\}$*, les puissances additives coïncident exactement avec les puissances abéliennes. En effet, soient* u *et* v *deux mots non vides sur* $\{a, b\}^*$ *de même taille et même somme, alors*

$$|u|_a + |u|_b = |v|_a + |v|_b \quad et \quad a|u|_a + b|u|_b = a|v|_a + b|v|_b,$$

or $a \neq 0$ *ou* $b \neq 0$ *et on obtient*

$$|u|_a = |v|_a \quad et \quad |u|_b = |v|_b.$$

Nous définissons aussi l'équivalence abélienne et additive entre des mots.

Définition 1.3.10. *Soit $\mathcal{A} \subseteq \mathbb{C}$ un alphabet. Deux mots u et v sont* abéliennement équivalents*, noté $u \simeq_{ab} v$ si $\psi(u) = \psi(v)$,* i.e. *si u et v sont une permutation l'un de l'autre.*

Ils sont additivement équivalents*, noté $u \simeq_{ad} v$, si $|u| = |v|$ et $\sum u = \sum v$.*

Pour les puissances et les puissances abéliennes sur un alphabet de nombres complexes, le choix des lettres n'a pas d'importance, seule compte la taille de l'alphabet. Ce n'est pas le cas pour les puissances additives pour lesquelles la valeur des lettres joue un rôle majeur.

Définition 1.3.11. *On dit que deux alphabets $\mathcal{A}, \mathcal{A}' \subseteq \mathbb{C}$ de même taille sont des* alphabets équivalents *s'il existe une fonction $h : \mathcal{A} \to \mathcal{A}'$, telle que pour tous $u, v \in \mathcal{A}^*$,*

$$u \simeq_{ad} v \iff h(u) \simeq_{ad} h(v)\,.$$

Nous allons maintenant, après avoir revu rapidement la construction d'un point fixe infini, définir la notion de parents qui sera essentielle dans les chapitres suivants.

1.4 Point fixe infini et notion de parents

Soient l'alphabet $\{a, b, c, d\}$ et $\varphi : \{a, b, c, d\}^* \to \{a, b, c, d\}^*$ le morphisme défini par

$$\varphi(a) = ac \quad ; \quad \varphi(b) = dc \quad ; \quad \varphi(c) = b \quad ; \quad \varphi(d) = ab.$$

Le mot $\mathbf{w} := \lim_{n\to\infty} \varphi^n(a)$ est obtenu en itérant φ sur la lettre a, autrement dit :

$$\begin{aligned}
\varphi^1(a) &= ac\\
\varphi^2(a) &= acb\\
\varphi^3(a) &= acbdc\\
\varphi^4(a) &= acbdcabb\\
\varphi^5(a) &= acbdcabbacdcdc\\
\varphi^6(a) &= acbdcabbacdcdcacbabbabb
\end{aligned}$$

On observe que, comme l'image de a commence par un a, les itérations sont toujours des préfixes des itérations suivantes (ainsi le mot $\mathbf{w}$ obtenu est un point fixe infini de φ). Les lettres sont générées à partir de l'image des lettres précédentes par le morphisme. On représente cette dépendance dans une structure arborescente (voir Figure 1.3).

Le parcours en largeur permet d'obtenir le mot tandis que la lecture de chaque niveau permet de lire le suffixe ajouté après une itération supplémentaire du morphisme. En fait il ne s'agit pas d'un arbre (car les sommets apparaissent plusieurs fois) mais simplement d'une représentation graphique commode. Pour représenter le mot sous forme d'arbre,

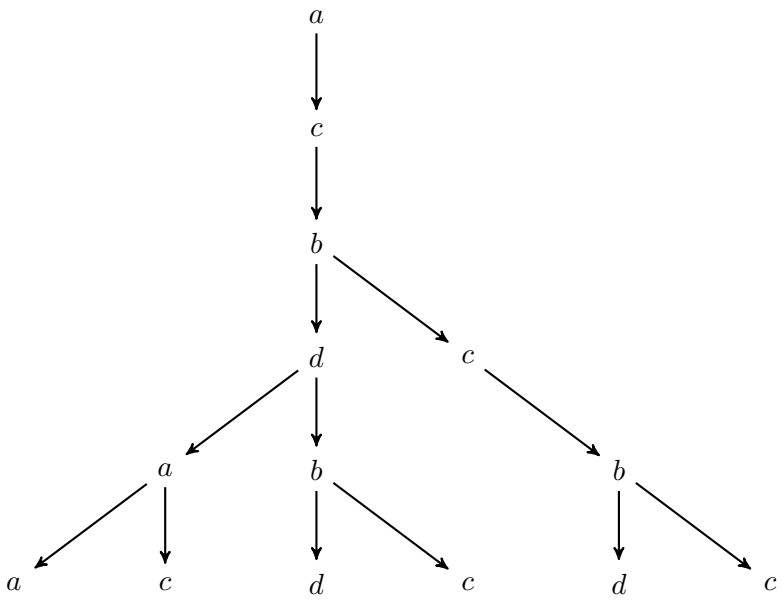


FIGURE 1.3 – 5 itérations du morphisme φ.

on construit un graphe dont les sommets sont des entiers positifs correspondant aux positions, en commençant à partir de 0, des lettres et dont les arêtes vont d'une position vers ses *enfants*. Définissons cette notion de façon intuitive. Si on construit le mot progressivement, on observe que la lettre b en position 6 a été engendrée comme image de la lettre d en position 3.

$$\underbrace{ac}_{0}\underbrace{b}_{1}\underbrace{dc}_{2}\underbrace{ab}_{3}\underbrace{b}_{4}\underbrace{ac}_{5}\underbrace{d\,c}_{6}\underbrace{d\,c}_{7}.$$
$$01\;2\;34\;56\;7\;89\;1011\;1213$$

On inscrit ainsi en rouge en dessous des positions la position parent, c'est-à-dire la position correspondant à la lettre qui a engendré celle considérée. Ainsi on dit que 5 est le parent de 8 et que 8 et 9 sont les enfants de 5. Voici maintenant la définition plus formelle des parents. Soit l'application

$$\begin{aligned}\eta : \mathbb{N} &\longrightarrow \mathbb{N}\\ p &\longmapsto |\varphi(\mathbf{w}[0,p))|.\end{aligned}$$

On obtient

$$\begin{aligned}\mathbf{w}\,[0,\eta(p))\,\mathbf{w}\,[\eta(p),\eta(p+1)) &= \mathbf{w}\,[0,\eta(p+1))\\ &= \varphi\left(\mathbf{w}[0,p+1)\right)\\ &= \varphi\left(\mathbf{w}[0,p)\right)\varphi\left(\mathbf{w}[p,p+1)\right)\\ &= \mathbf{w}\,[0,\eta(p))\,\varphi\left(\mathbf{w}[p]\right),\end{aligned}$$

donc $\mathbf{w}[\eta(p), \eta(p+1)) = \varphi(\mathbf{w}[p])$. On obtient la définition suivante.

Définition 1.4.1. *Étant donnée une position p dans $\mathbf{w}$, on appelle **parent** de p et on note $\mathtt{Par}(p)$ l'unique position t telle que $\eta(t) \leq p < \eta(t+1)$. Inversement un **enfant** d'une position p est une position q telle que $\mathtt{Par}(q) = p$.*

Position p	$\eta(p)$
0	0
1	2
2	3
3	5
4	7
5	8

Position	Parent
0	0
1	
2	1
3	2
4	
5	3
6	
7	4
8	5
9	

a

$\underline{a}c$

$a\underline{c}b$

$ac\underline{b}dc$

$acbd\underline{c}ab$

$acbdc\underline{a}bb$

$acbdca\underline{b}bac$

Figure 1.4 – Correspondance entre les positions dans $\mathbf{w}$ et leurs parents.

Ainsi 10 est un enfant de 6, 6 est un enfant de 3, le parent de 3 est 2, le parent de 2 est 1 et le parent de 1 est 0. La Figure 1.4 répertorie les liens entre les positions et leurs parents. Sur la droite, la lettre soulignée a pour image les lettres colorées en rouge. La notion de parents a été introduite en 2004 par A. Aberkane, J.D. Currie et N. Rampersad [1] en 2004. Elle se généralise aux facteurs et aux n-uplets de positions, on se reportera aux Définitions 3.1.2 et 3.1.3.

1.5 Organisation du présent document

Le Chapitre 2 présente notre démarche de recherche et établit des liens entre le morphisme φ_0 et les autres morphismes possédant un point fixe infini qui évite les cubes additifs. Nous expliquons également dans ce chapitre comment nous implémentons un programme efficace de recherche de morphismes possédant un tel point fixe.

Dans le Chapitre 3 nous montrons que le morphisme $\varphi_6 : \{0,1,2,6\}^* \to \{0,1,2,6\}^*$ défini par

$$\varphi_6(0) = 2 \quad ; \quad \varphi_6(1) = 62 \quad ; \quad \varphi_6(2) = 10 \quad ; \quad \varphi_6(6) = 60,$$

engendre le point fixe infini

$$\mathbf{w} = \overrightarrow{\varphi_6^{\omega}}(6) = 6021062260101060262262260210601010601010\cdots$$

qui évite également les cubes additifs. Nous suivons le schéma de preuve de J. Cassaigne et al. [16] pour construire un algorithme de décision valable pour ce morphisme.

Le Chapitre 4 prouve que cet algorithme est général et peut en fait être utilisé pour décider si un morphisme similaire à φ_0 et de taille 2 possède un point fixe infini sans cube additif.

Dans le Chapitre 5 nous montrons, dans un travail conjoint avec M. Rosenfeld [33], qu'il est en fait possible d'éviter les cubes additifs sur tous les alphabets de 4 lettres qui ne sont pas équivalents à $\{0, 1, 2, 3\}$. Cette preuve est constructive.

Enfin, le Chapitre 6 explore la question de l'évitabilité des cubes additifs sur l'alphabet $\{0, 1, 2, 3\}$. Nous construisons via un programme parallélisé un mot de plus de 70 millions de lettres sans cube additif sur cet alphabet, ce qui améliore la précédente taille connue qui était de 1.4×10^5 [37]. Nous expliquons les démarches qui ont engendré nos choix de programmation et l'architecture de notre algorithme.

Remarque 1.5.1. *Sauf mention du contraire, les tests informatiques sont effectués sur un ordinateur de bureau standard muni d'un processeur Intel Core i7-8550U cadencé à 1.8GHz (4.0GHz au maximum) avec 8Mo de cache et 8Go de RAM.*

Les listes du Chapitre 2 et les programmes du Chapitre 6 se trouvent à l'adresse : `https://members.loria.fr/FLietard/phdProgramsAndDocumentation/`.

Chapitre 2

De la recherche de mots évitant les cubes additifs aux morphismes similaires à φ_0

2.1 La démarche de recherche

2.1.1 Des questions issues de l'état de l'art

Reprenons l'état de l'art pour mettre en exergue les axes de recherche qui s'offraient à nous au commencement de ce travail de thèse. Le Tableau 2.1 récapitule les résultats obtenus (avec l'année de l'article correspondant) sur la taille minimale de l'alphabet nécessaire pour éviter chaque motif. Un historique plus complet peut être trouvé en Section 1.2.

	Pures	**Abéliennes**	**Additives**	
cubes	2 lettres 1906 [42] A. Thue	3 lettres 1979 [20] F. F.M. Dekking	4 lettres 2014 [16] J. Cassaigne et al.	3 lettres 2015 [37] M. Rao
carrés	3 lettres 1912 [43] A. Thue	4 lettres 1992 [31] V. Keränen	?	

Tableau 2.1 – Taille de l'alphabet nécessaire pour éviter chaque motif et année de l'article correspondant.

Pour chacun de ces résultats, les auteurs exhibent un morphisme et un point fixe infini d'un morphisme pour construire un mot infini sans puissance pure, abélienne ou additive. Le cas des carrés additifs reste ouvert depuis presque cinquante ans [14, 29, 27, 35]. Nous nous sommes intéressés en premier lieu au résultat de Julien Cassaigne, James D. Curie, Luke Schaeffer et Jeffrey O. Shallit [16] concernant l'évitabilité des cubes additifs sur un alphabet de taille 4. Le morphisme considéré est $\varphi_0 : \{0,1,3,4\}^* \to \{0,1,3,4\}^*$ défini

par

$$\varphi_0(0) = 03 \quad ; \quad \varphi_0(1) = 43 \quad ; \quad \varphi_0(3) = 1 \quad ; \quad \varphi_0(4) = 01,$$

et il constitue le point de départ de ce travail de thèse. Nous souhaitions, grâce à la compréhension de cet article, mieux appréhender les façons d'éviter les cubes additifs et dégager des idées nouvelles pour l'étude des carrés additifs. L'un de ces résultats a été atteint pendant la durée de cette thèse. Au mois de septembre 2016 nous avons donc débuté les expérimentations et la recherche de nouveaux morphismes ayant des points fixes infinis sans cube additif. Nous avons souhaité comprendre comment le morphisme φ_0, présenté ci-dessus, avait été découvert et pourquoi ce morphisme précisément avait permis de prouver l'évitabilité des cubes additifs. Le premier pas consiste à chercher les morphismes

— de la plus petite taille possible, c'est-à-dire avec des images de taille au maximum 2 puisqu'un morphisme de taille 1 ne peut pas posséder de point fixe infini,
— sur le plus petit alphabet possible, c'est-à-dire $\{0, 1, 2, 3\}$,

possédant un point fixe infini sans cube additif. Par une recherche exhaustive nous avons vérifié qu'il n'existe aucun morphisme répondant à ces critères. Nous avons effectué cette recherche en construisant l'ensemble des 20 mots d'au plus deux lettres sur l'alphabet $\{0, 1, 2, 3\}$ et en construisant les 20^4 morphismes (une image de taille au plus deux étant attribuée à chaque lettre de l'alphabet). Nous avons ensuite testé pour chaque morphisme construit s'il possédait un point fixe infini et, dans l'affirmative, si son préfixe de longueur 10000 possédait un cube additif. Ce test a révélé que si un de ces morphismes possède un point fixe infini, alors ce dernier contient un cube additif dans son préfixe de longueur 10000. Aucun de ces morphismes ne s'avère être un candidat pour fournir un point fixe infini sans cube additif. En fait 10000 est ici une borne très grossière car les préfixes de longueur 128 des points fixes infinis (s'ils existent) de ces morphismes contiennent déjà tous un cube additif. Un des morphismes qui atteint cette borne est

$$\varphi_7 : \begin{cases} 0 \mapsto 03 \\ 1 \mapsto 01 \\ 2 \mapsto 1 \\ 3 \mapsto 32 \end{cases}$$

et le point fixe infini considéré est

$$\begin{aligned} \lim_{n\to\infty} \varphi_7^n(3) = {} & 32101030103320301033232103320301033232132101033232 \\ & 10332030103323213210132101030103332321321010332321 0 \\ & 33203010332321\underbrace{32101}\underbrace{32101}\underbrace{03013}2\ldots \end{aligned}$$

dans lequel nous faisons apparaître ici la première occurrence d'un cube additif, qui se termine à la position 128.

L'étape suivante pour J. Cassaigne et al. (comme pour nous lorsque nous avons repris ces expérimentations) a été de chercher un morphisme de taille 2 toujours mais sur un alphabet contenant une lettre de plus, c'est-à-dire $\{0, 1, 2, 3, 4\}$. Ils réussirent à trouver

sur cet alphabet un morphisme candidat dont ils ont pu réduire le domaine de définition car le point fixe infini sans cube additif qu'il produisait n'utilisait que 4 lettres sur les cinq. Par morphisme candidat nous désignons un morphisme produisant un point fixe infini pour lequel nos tests informatiques ne permettent pas de détecter un cube additif dans un préfixe de plusieurs milliers de lettres. Les quatre auteurs réussirent ensuite, en utilisant des arguments de théorie des graphes et d'algèbre linéaire, à prouver que ce morphisme particulier produisait effectivement un point fixe infini sans cube additif.

Pour simplifier la rédaction, nous posons les définitions suivantes.

Définition 2.1.1. *Un morphisme φ défini sur un alphabet $\mathcal{A}$ fini* vérifie la condition Cupisca sur $\mathcal{A}$ *s'il possède un point fixe infini sans cube additif sur cet alphabet (c'est-à-dire s'il Crée Un Point fixe Infini Sans Cube Additif).*

Définition 2.1.2. *Soit un entier $N \in \mathbb{N}$, on dit qu'un morphisme $\varphi : \mathcal{A}^* \to \mathcal{A}^*$ est un* N-candidat sur $\mathcal{A}$ *s'il possède un point fixe infini dont le préfixe de longueur N est sans cube additif. On dira qu'un morphisme est* un candidat *s'il est un* 10000*-candidat.*

Le choix de 10000 comme borne est dû à nos premières expérimentations : tous les points fixes infinis dans lesquels nous avons trouvé des cubes additifs en contenaient dans leurs 2000 premiers termes. En prenant 10000 nous nous assurons une certaine "crédibilité" de nos candidats sans engendrer un temps de calcul trop long.

À la lumière de l'article [16] nous nous sommes posé plusieurs questions.

1. Peut-on trouver des morphismes structurellement différents de φ_0 vérifiant la condition Cupisca sur un alphabet fini ?
2. Peut-on trouver un unique morphisme vérifiant la condition Cupisca sur tout alphabet ou même simplement sur une infinité d'alphabets non équivalents ? La Définition 1.3.11 établit l'équivalence entre deux alphabets.
3. Peut-on trouver sur tous les alphabets d'au moins trois lettres un morphisme vérifiant la condition Cupisca ?
4. Peut-on appliquer le schéma de la preuve de l'article [16] à d'autres morphismes ?
5. Peut-on trouver des morphismes candidats sur $\{0, 1, 2, 3\}$?

Dans cet écrit, nous répondrons en partie ou de façon complète à chacune de ces questions.

2.1.2 Rechercher les morphismes candidats

Pour répondre à ces questions, nous avons commencé par rechercher des morphismes de taille 2 sur des alphabets réels de 4 lettres $\{a, b, c, d\}$ avec $a < b < c < d$. En fait on peut toujours considérer que $a = 0$ quitte à soustraire a à chaque lettre de l'alphabet, ce qui ne change pas le fait qu'un mot soit ou non sans cube additif (on peut se référer au Lemme 5.1.1 pour plus de détails). De même si nous obtenons les morphismes N-candidats sur un alphabet $\{0, b, c, d\}$, alors nous obtenons les N-candidats sur l'alphabet $\{d-d = 0, d-c, d-b, d-0\}$ et sur les alphabets $\{0, kb, kc, kd\}$ pour $k \in \mathbb{R}$ en effectuant

les substitutions lettre à lettre. Cette astuce fut la seule utilisée pour notre première recherche de morphismes 10000-candidats. Nous avons testé tous les alphabets d'entiers respectant les contraintes $0 = a < b < c < d < 26$ et trouvé 32068 morphismes 10000-candidats de taille 2. La liste complète de ces morphismes se trouve à l'adresse web donnée page 13.

En étudiant ces morphismes, nous nous sommes aperçus qu'ils étaient tous similaires à φ_0 (voir la Définition 1.3.5). Supposons qu'un morphisme vérifie la condition Cupisca sur un des alphabets d'entiers $\{a, b, c, d\}$ avec $0 = a < b < c < d < 26$. Il possède alors un point fixe infini sans cube additif, ce qui signifie que le préfixe de longueur 10000 est également dépourvu de cubes additifs. Ce morphisme est donc un candidat sur $\{a, b, c, d\}$, il appartient par conséquent à l'ensemble (exhaustif) des 32068 candidats que nous avons trouvé. Or ces 32068 candidats sont tous similaires à φ_0. Ceci implique la proposition suivante.

Proposition 2.1.3. *Soit $\{a, b, c, d\}$ un alphabet d'entiers vérifiant $0 = a < b < c < d < 26$ et soit φ un morphisme sur $\{a, b, c, d\}^*$ de taille 2 vérifiant la condition Cupisca. Alors φ est similaire à φ_0.*

Ce résultat reste pour nous relativement surprenant car nous ne parvenons pas à trouver de morphismes de taille 2 structurellement différents de φ_0 qui soit un candidat sur un de ces alphabets. Nous discuterons plus précisément dans la Partie 2.3.3 des implications de ce résultat.

Voici donc les bases sur lesquelles a commencé ce travail de thèse, une première observation de ressemblances étonnantes entre φ_0 et les morphismes qui semblaient avoir un point fixe infini évitant les cubes additifs, et une pléiade de questions. Nous allons, dans la suite de cet écrit, apporter des éléments de réponses aux questions posées. Dans la Partie 2.3.1 nous étudierons les Questions 1 et 2. Les Chapitres 3 et 4 donneront une réponse à la question 4. La Question 3 sera traitée dans la Partie 2.3.2 en utilisant les résultats du Chapitre 5 qui montrent que le morphisme φ_0 peut être transposé dans une infinité d'alphabets non équivalents et toujours vérifier la condition Cupisca. Le chapitre 6 traitera du cas particulier de l'alphabet $\{0, 1, 2, 3\}$ et de la Question 5.

2.1.3 La nécessité d'un programme parallélisé pour une recherche approfondie de morphismes

Le programme informatique que nous avons utilisé au début de ce travail de thèse était relativement simple. Nous considérions des morphismes de petite taille (deux ou trois) sur des alphabets de 4 lettres. Il nous suffisait de programmer des boucles sur toutes les images possibles pour énumérer les morphismes et tester s'il s'agissait de 10000-candidats, le tout sur un seul cœur du processeur (les différents tests ont été effectués sur des ordinateurs de bureau standards dont les caractéristiques sont données dans la Remarque 1.5.1). Lorsque nous augmentons la taille des morphismes considérés, le nombre d'images possible pour chaque lettre croît exponentiellement.

Le Tableau 2.2 récapitule, en fonction de la longueur, le nombre total de mots sans cube additif sur l'alphabet $\{0, 1, 2, 3\}$ de longueurs $1 \leq n \leq 8$. Notons que, afin de traiter

moins de cas, nous imposons la condition que les images des lettres ne contiennent pas de cube additif, sinon le point fixe infini en contiendra forcément un dès lors qu'il contient les 4 lettres de l'alphabet.

Tableau 2.2 – Nombre de mots sans cube additif sur $\{0, 1, 2, 3\}$ en fonction de leur longueur.

n	1	2	3	4	5
Nombre de mots sans cubes additifs de longueur n	4	16	60	228	864
Pourcentage par rapport aux mots de longueur n	100%	100%	94%	89%	84%
Nombre de mots sans cubes additifs de longueur au plus n	4	20	80	308	1172
Pourcentage par rapport aux mots de longueur au plus n	100%	100%	95%	91%	86%

n	6	7	8	9	10
Nombre de mots sans cubes additifs de longueur n	3152	11468	42070	150560	538214
Pourcentage par rapport aux mots de longueur n	77%	70%	64%	57%	51%
Nombre de mots sans cubes additifs de longueur au plus n	4324	15792	57862	208422	746636
Pourcentage par rapport aux mots de longueur au plus n	79%	72%	66%	60%	53%

On rappelle qu'un morphisme est de taille n s'il a au moins une de ses images de taille n. Par conséquent, il y a $20^4 - 4^4 = 159744$ morphismes de taille 2 dont les images ne contiennent pas de cubes additifs (ce qui est toujours vérifié pour des mots de deux lettres). En effet, il y a 20 choix possible dans les mots d'au plus deux lettres pour chacune des 4 images mais on doit retirer les 4^4 morphismes de taille 1. Le nombre de morphismes de taille n dont les images ne contiennent pas de cubes additifs se trouve répertorié dans le Tableau 2.3.

On observe que le nombre de morphismes avec des images sans cube additif augmente fortement, nous avons donc été contraint de changer de méthode afin de pouvoir pousser plus en avant nos recherches de morphismes candidats. Ces changements ont eu lieu principalement selon deux axes :

— l'amélioration de la technique de détection des cubes additifs
— la parallélisation des calcul effectués par le programme.

Pour gagner en efficacité sur la détection des cubes additifs nous avons procédé à quatre améliorations successives qui sont développées dans la Section 6.3. Le résultat est une fonction de test qui permet de vérifier en moins de 4 millièmes de seconde sur un ordinateur standard si un mot de longueur 10000 est bien sans cube additif. La rapidité de cette fonction est essentielle pour la recherche des morphismes 10000-candidats. Munis

Tableau 2.3 – Nombre de morphismes sur $\{0, 1, 2, 3\}$ sans cube additif dans les images des lettres, en fonction de leur longueur.

n	1	2	3	4
Nombre de morphismes de taille n	4	159744	4.08×10^7	$\simeq 9.0 \times 10^9$

n	5	6	7	8
Nombre de morphismes de taille n	$\simeq 1.9 \times 10^{12}$	$\simeq 3.5 \times 10^{14}$	$\simeq 6.2 \times 10^{16}$	$\simeq 1.1 \times 10^{19}$

n	9	10
Nombre de morphismes de taille n	$\simeq 1.9 \times 10^{21}$	$\simeq 3.1 \times 10^{23}$

de cette fonction, nous expliquons dans la Section 2.2 le principe de fonctionnement de l'algorithme qui effectue cette recherche.

2.2 Un programme multi-threads pour détecter les morphismes candidats

Nous avons souhaité utiliser la puissance de Grid5000. Créée en 2003 et développée par des chercheurs issus majoritairement de l'Inria et du Loria, Grid5000 est une infrastructure de calcul rassemblant des calculateurs parallèles répartis géographiquement sur le territoire Français et au Luxembourg. Ces calculateurs sont interconnectés par des réseaux très haut débit et forment un maillage de 828 nœuds et 12328 cœurs (chiffres de 2019) permettant de mener des expérimentations approfondies pour le calcul haute performance. Pour utiliser cette infrastructure, nous avions besoin d'un programme pouvant être déployé sur plusieurs ordinateurs et ensuite sur tous les threads de ces différentes machines. Les objectifs étaient :

- concevoir un programme qui, pour un alphabet et une taille fixés, cherche tous les morphismes candidats sur cet alphabet et de cette taille,
- faire en sorte que ce programme soit parallélisable sur un réseau de machines hétérogènes (elles ne possèdent pas nécessairement les mêmes caractéristiques en terme de processeur et nombre de cœurs),
- s'assurer que le temps de communication entre les différents éléments ne nuise pas à l'efficacité de notre programme.

Nous avons nommé ce programme *Looking4morphisms* et il fonctionne selon le principe expliqué ci-après. On pourra se référer à la Figure 2.1 qui complète ces explications. On suppose ici qu'on recherche un candidat de taille $n \in \mathbb{N}$ sur un alphabet $\{a, b, c, d\}$.

1. Un premier ordinateur est désigné comme ordinateur principal. C'est sur celui-ci que sera stocké l'ensemble du programme et exécuté le premier script.

2. On exécute sur cet ordinateur un script *launch* qui commence par lancer un programme appelé *generateAcfWords* qui écrit dans n fichiers distincts la liste des mots de longueur fixée $i \in [\![1, n]\!]$ sans cube additif (un fichier par longueur i). On note $L_1, L_2, \ldots, L_n$ ces listes.
3. Ce programme liste ensuite tous les quadruplets d'entiers appartenant à $[\![1, n]\!]^4$ et contenant au moins une fois la valeur n. On note Q la liste de ces quadruplets. Ils représentent les longueur possibles de chacune des quatre images d'un morphisme de taille n (une image pour chaque lettre de l'alphabet).
4. On lance le programme *split* qui sonde le nombre d'ordinateurs avec lesquels il pourra communiquer, notons m ce nombre. Il divise la liste L_n des mots sans cube additif de longueur n en m sous listes $L_{n,1}, L_{n,2}, \ldots, L_{n,m}$ de longueurs approximativement $\#(L_n)/m$, approximativement car cette fraction n'est pas nécessairement un entier. La division ne se fait pas en découpant L_n en m facteurs consécutifs. Nous avons estimé que si nous procédions de cette façon, les tâches ne seraient pas équitablement réparties et certaines machines risquaient de progresser beaucoup plus vite que d'autres et de perdre du temps dans l'attente des autres résultats. Nous construisons donc, pour $j \in [\![1, m]\!]$, la liste $L_{n,j}$ en prenant tous les mots dont la position dans L_n est congrue à j modulo m.
5. Une fois construites ces listes, le programme rentre dans une boucle sur les quadruplets de Q. Pour chaque quadruplet il envoie aux m ordinateurs secondaires.
   — La liste $L_{n,j}$, où j est le numéro attribué à cet ordinateur secondaire
   — Le quadruplet $(i_1, i_2, i_3, i_4) \in Q$ en cours, nous supposerons pour simplifier l'explication que $i_1 = n$ car nous savons que au moins un des éléments de ce quadruplet vaut n.
   — Les listes L_{i_2},L_{i_3} et L_{i_4}.
   — Un programme nommé *seek*.
6. Sur chaque ordinateur secondaire, de numéro j, le programme *seek* est lancé. Il entre dans une boucle sur tous les morphismes dont l'image de a est dans $L_{n,j}$, l'image de b dans L_{i_2}, l'image de c dans L_{i_3} et l'image de d dans L_{i_4}. Les tests de morphismes sont répartis entre les différents threads de l'ordinateur secondaire j. Notre programme s'adapte à l'hétérogénéité de la structure en récoltant pour chaque machine le nombre de threads qu'elle possède (noté T_j sur la Figure 2.1). Notons également que grâce à ce découpage et au vu des cardinaux des listes L_i (voir Tableau 2.2), chaque ordinateur secondaire reçoit une part de travail assez conséquente et pourra la traiter sans communication avec l'ordinateur principal jusqu'à l'obtention d'un résultat positif. Nous évitons ainsi que trop de temps soit consommé en communication entre les éléments du réseau.
7. Chaque thread reçoit un morphisme et regarde s'il possède un point fixe infini. Si la réponse est non il passe au morphisme suivant. Si la réponse est oui il construit son point fixe de longueur 10000.
8. Le thread teste si ce préfixe est bien dépourvu de cubes additifs, autrement dit si le morphisme étudié est bien un candidat. Si la réponse est non, le thread passe au

morphisme suivant. Si la réponse est oui, le morphisme est remonté à l'ordinateur secondaire.

9. Si l'ordinateur secondaire reçoit un morphisme candidat d'un de ses threads, il le remonte à l'ordinateur principal. Ce morphisme sera alors affiché à l'interface de l'utilisateur et enregistré dans un fichier sur l'ordinateur principal.
10. Lorsqu'un ordinateur secondaire a parcouru tous les morphismes issus de l'ensemble $L_{n,j} \times L_{i_2} \times L_{i_3} \times L_{i_4}$ il doit, tel que notre programme est construit actuellement, attendre que les autres ordinateurs secondaires aient terminé également afin que l'ordinateur principal lance l'étude du quadruplet de Q suivant. Ceci est un axe d'amélioration de notre programme, nous pourrons le modifier par la suite pour éliminer cet attente et que tout ordinateur libre reçoive immédiatement un nouveau cas à traiter.
11. Lorsque tous les quadruplets ont été étudiés, l'ordinateur principal contient en mémoire la liste de tous les morphismes candidats de taille n trouvés sur l'alphabet $\{a, b, c, d\}$.

En fait, ce programme est la version améliorée que nous utilisons actuellement pour la recherche des morphismes. Celui que nous avons utilisé au début de cette thèse était une version moins aboutie et qui tournait sur un seul ordinateur. La version actuelle, beaucoup plus efficace, nous permet d'effectuer des recherches de morphismes candidats sur l'alphabet $\{0, 1, 2, 3\}$, il en sera question dans la Section 6.1. Le code en `C++` du programme *Looking4Morphisms* peut être trouvé à l'adresse `https://members.loria.fr/FLietard/phdProgramsAndDocumentation/`. Il a été rédigé par Damien Jamet et F.L.

2.3 Lien entre l'évitabilité des puissances additives et les morphismes similaires à φ_0

2.3.1 Des morphismes similaires à φ_0 et à φ_0^2

Nous pouvons espérer répondre par l'affirmative à la Question 1 car une première recherche nous permet déjà de trouver des morphismes candidats. Ces derniers sont "structurellement proches" de φ_0 et la Proposition 2.1.3 rend d'autant plus intéressante la Question 4 et le fait de savoir si l'algorithme utilisé par Cassaigne et al. dans [16] peut être étendu à d'autres morphismes sur d'autres alphabets. Les Chapitres 3 et 4 apportent la preuve du théorème suivant.

Théorème 2.3.1. *Soit φ un morphisme de taille 2 similaire à φ_0, alors on peut décider algorithmiquement s'il vérifie la condition* Cupisca.

Grâce à ce théorème, nous avons démontré que les 32068 morphismes candidats de taille 2 trouvés sur les alphabets d'entiers $\{a, b, c, d\}$ avec $0 = a < b < c < d < 26$ vérifient effectivement la condition Cupisca. Le fait que d soit choisit plus petit que 26 est dû au fait que nous avons choisi les 26 lettres de l'alphabet pour notre programmation en

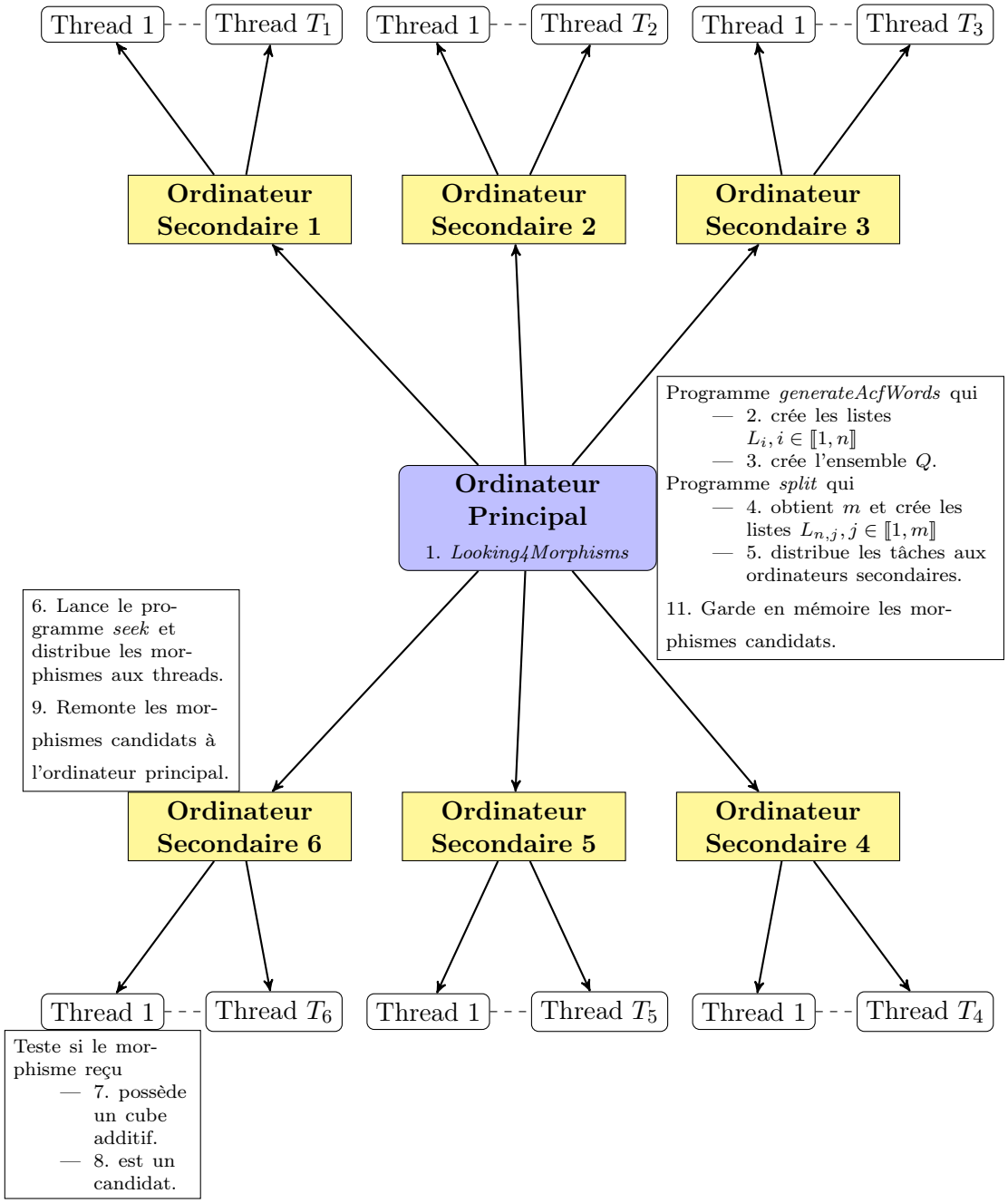


Figure 2.1 – Schéma explicatif du fonctionnement du programme *Looking4Morphisms*.

leur faisant correspondre la valeur de leur rang dans l'alphabet. Le Tableau 2.4 donne de façon exhaustive les morphismes de taille 2 vérifiant la condition Cupisca sur les alphabets $\{0, 1, 2, d\}$ pour $d \in [\![3, 9]\!]$.

Tableau 2.4 – Morphismes de taille 2 sur $\{0,1,2,d\}$, $d \in [\![3,9]\!]$, vérifiant la condition Cupisca.

$\{0,1,2,3\}$	
$\{0,1,2,4\}$	
$\{0,1,2,5\}$	$0\mapsto 52, 1\mapsto 12, 2\mapsto 0, 5\mapsto 10$ $0\mapsto 25, 1\mapsto 5, 2\mapsto 21, 5\mapsto 01$
$\{0,1,2,6\}$	$0\mapsto 62, 1\mapsto 12, 2\mapsto 0, 6\mapsto 10$ $0\mapsto 26, 1\mapsto 6, 2\mapsto 21, 6\mapsto 01$ $0\mapsto 2, 1\mapsto 62, 2\mapsto 10, 6\mapsto 60$ $0\mapsto 01, 1\mapsto 6, 2\mapsto 06, 6\mapsto 21$
$\{0,1,2,7\}$	$0\mapsto 72, 1\mapsto 12, 2\mapsto 0, 7\mapsto 10$ $0\mapsto 27, 1\mapsto 7, 2\mapsto 21, 7\mapsto 01$ $0\mapsto 01, 1\mapsto 7, 2\mapsto 07, 7\mapsto 21$
$\{0,1,2,8\}$	$0\mapsto 82, 1\mapsto 12, 2\mapsto 0, 8\mapsto 10$ $0\mapsto 28, 1\mapsto 8, 2\mapsto 21, 8\mapsto 01$ $0\mapsto 2, 1\mapsto 82, 2\mapsto 10, 8\mapsto 80$ $0\mapsto 01, 1\mapsto 8, 2\mapsto 08, 8\mapsto 21$
$\{0,1,2,9\}$	$0\mapsto 92, 1\mapsto 12, 2\mapsto 0, 9\mapsto 10$ $0\mapsto 29, 1\mapsto 9, 2\mapsto 21, 9\mapsto 01$ $0\mapsto 2, 1\mapsto 92, 2\mapsto 10, 9\mapsto 90$ $0\mapsto 01, 1\mapsto 9, 2\mapsto 09, 9\mapsto 21$

Cependant nous pouvons étendre encore le champs d'application de cet algorithme. Dans notre recherche nous n'avons trouvé aucun morphisme de taille 2 qui soit un 10000-candidat sur l'alphabet $\{0,1,2,4\}$. Nous avons en revanche trouvé des morphismes candidats de taille 3, dont celui présenté dans l'article de M. Rao et M. Rosenfeld publié en 2018 [39, Section 6.3]. Il y a sur cet alphabet 11 morphismes 10000-candidats, 5 d'entre eux sont similaires à φ_0 et les 6 autres à φ_0^2. Ils sont représentés dans le Tableau 2.5. L'algorithme généralisé que nous verrons dans le Chapitre 4 peut également être utilisé dans ces deux cas et on montre que ces 11 morphismes vérifient effectivement la condition Cupisca.

Tableau 2.5 – Morphismes de taille 3 sur $\{0,1,2,4\}$ vérifiant la condition Cupisca.

Classe de φ_0	$0\mapsto 4$ $1\mapsto 12$ $2\mapsto 0$ $4\mapsto 100$	$0\mapsto 244$ $1\mapsto 4$ $2\mapsto 21$ $4\mapsto 0$	$0\mapsto 244$ $1\mapsto 14$ $2\mapsto 1$ $4\mapsto 0$	$0\mapsto 2$ $1\mapsto 4$ $2\mapsto 21$ $4\mapsto 011$	$0\mapsto 144$ $1\mapsto 12$ $2\mapsto 4$ $4\mapsto 0$	
Classe de φ_0^2	$0\mapsto 21$ $1\mapsto 011$ $2\mapsto 214$ $4\mapsto 244$	$0\mapsto 12$ $1\mapsto 110$ $2\mapsto 412$ $4\mapsto 442$	$0\mapsto 12$ $1\mapsto 110$ $2\mapsto 412$ $4\mapsto 442$	$0\mapsto 100$ $1\mapsto 140$ $2\mapsto 14$ $4\mapsto 244$	$0\mapsto 001$ $1\mapsto 041$ $2\mapsto 41$ $4\mapsto 442$	$0\mapsto 001$ $1\mapsto 041$ $2\mapsto 41$ $4\mapsto 442$

Prenons par exemple les premiers morphismes de chacune des deux classes :

$$\varphi_8 : \begin{cases} 0 \mapsto 4 \\ 1 \mapsto 12 \\ 2 \mapsto 0 \\ 4 \mapsto 100 \end{cases} \quad \text{et} \quad \varphi_9 : \begin{cases} 0 \mapsto 21 \\ 1 \mapsto 011 \\ 2 \mapsto 214 \\ 4 \mapsto 244 \end{cases},$$

qui ont pour matrices d'incidence $\mathtt{Mat}(\varphi_8) = \begin{pmatrix} 0&0&1&2 \\ 0&1&0&1 \\ 0&1&0&0 \\ 1&0&0&0 \end{pmatrix}$ et $\mathtt{Mat}(\varphi_9) = \begin{pmatrix} 0&1&0&0 \\ 1&2&1&0 \\ 1&0&1&1 \\ 0&0&1&2 \end{pmatrix}$. On obtient les égalités suivantes :

$$\begin{pmatrix} 1&0&0&0 \\ 2&1&-1&-2 \\ -1&0&1&3 \\ -1&0&1&2 \end{pmatrix}^{-1} \times \mathtt{Mat}(\varphi_0) \times \begin{pmatrix} 1&0&0&0 \\ 2&1&-1&-2 \\ -1&0&1&3 \\ -1&0&1&2 \end{pmatrix} = \mathtt{Mat}(\varphi_8),$$

$$\begin{pmatrix} 1&0&0&0 \\ 0&1&0&-1 \\ 1&-1&1&1 \\ -1&0&0&1 \end{pmatrix}^{-1} \times \mathtt{Mat}(\varphi_0^2) \times \begin{pmatrix} 1&0&0&0 \\ 0&1&0&-1 \\ 1&-1&1&1 \\ -1&0&0&1 \end{pmatrix} = \mathtt{Mat}(\varphi_9).$$

Ces morphismes sont bien similaires respectivement à φ_0 et à φ_0^2 mais ils ne sont pas des substitutions de ces morphismes. En effet, φ_0 est de taille 2 et φ_8 est de taille 3, il ne peut exister de substitution lettre à lettre entre ces morphismes. De même, φ_0^2 est de taille 4 et φ_9 est de taille 3.

Rappelons maintenant la Question 2 posée page 17 : peut-on trouver un unique morphisme vérifiant la condition Cupisca sur tout alphabet ou même simplement sur une infinité d'alphabets non équivalents ? La réponse est oui et nous obtenons la proposition suivante.

Proposition 2.3.2. *Pour tout morphisme φ substitution de φ_0, pour tous réels a et b avec $a < b$, il existe $M_\varphi(a,b) \in \mathbb{R}$ tel que pour tout $c > M_\varphi(a,b)$ il existe $N_\varphi(a,b,c) \in \mathbb{R}$ tel que pour tout $d > N_\varphi(a,b,c)$ le morphisme φ vérifie Cupisca sur $\{a,b,c,d\}$.*

En fait nous obtenons non pas un mais 24 morphismes qui possèdent chacun une infinité de valuations sur lesquelles ils vérifient la condition Cupisca, il s'agit des 24 permutations de φ_0. La preuve de cette proposition peut être trouvée page 78. Les notions nécessaires à sa démonstration seront, pour des raisons de fluidité de lecture, introduites ultérieurement.

2.3.2 Un morphisme vérifiant la condition Cupisca et similaire à φ_0 sur chaque alphabet

Le lien entre l'évitabilité des cubes additifs sur les alphabets de taille 4 et φ_0 peut être encore consolidé par le résultat suivant, prouvé dans la Section 5.6.

Proposition 2.3.3. *Sur tout alphabet $\{a, b, c, d\}$ non-équivalent à $\{0, 1, 2, 3\}$, il existe un morphisme similaire à φ_0 vérifiant la condition Cupisca. De plus si $\{a, b, c, d\}$ non-équivalent à $\{0, 1, 2, 4\}$ alors il existe un morphisme substitution de φ_0 vérifiant la condition Cupisca sur $\{a, b, c, d\}$.*

Le but de cette partie étant de présenter les axes de recherche qui s'offrent à nous et les similitudes présentent entre les morphismes vérifiant la condition Cupisca et φ_0, nous avons préféré ne pas le densifier avec des démonstrations. De plus, ces dernières seront beaucoup plus naturelles lorsque la matière nécessaire aura été introduite et manipulée et que ces propositions se déduiront aisément des résultats que nous présenterons ultérieurement.

2.3.3 Une condition nécessaire ?

Nous observons dès à présent un lien très fort entre φ_0 et les morphismes vérifiant la condition Cupisca. Sur presque tous les alphabets on trouve des morphismes similaires à φ_0 qui vérifient la condition Cupisca et si on prend n'importe quel morphisme substitution de φ_0 alors on trouve une infinité d'alphabets non équivalents sur lesquels il vérifie la condition Cupisca. Mais ce n'est pas tout, rappelons que, sur tous les alphabets d'entiers $\{a, b, c, d\}$ avec $0 = a < b < c < d < 26$ que nous avons étudié, nous n'avons trouvé aucun autre morphisme vérifiant la condition Cupisca que ceux similaires à φ_0. Pour mener plus loin nos recherches nous avons établi la liste de tous les morphismes de taille 2 sur quatre lettres possédant un point fixe infini dont le préfixe de longueur 10000 évite les cubes abéliens. En effet, éviter les cubes abéliens est une condition nécessaire pour éviter les cubes additifs. Nous avons trouvé 288 morphismes ayant cette propriété et ils sont répertoriés dans les tableaux de la Section A.1.

Notons que si un morphisme de taille 2 vérifie la condition Cupisca il appartient forcément à cette liste (pour une certaine valuation de $\{a, b, c, d\}$). Or, cette liste ne contient que 24 morphismes similaires à φ_0. Il existe 24 permutations possibles de l'ensemble $\{a, b, c, d\}$. Comme φ_0 vérifie la condition Cupisca, son point fixe infini évite les cubes abéliens et toute permutation de ce mot évite également les cubes abéliens. Soit s une de ces 24 permutations, le morphisme $s^{-1} \circ \varphi_0 \circ s$ possède donc un point fixe infini sans cubes abéliens et appartient à la liste des 288 morphismes ci-dessus. En fait les 24 morphismes similaires à φ_0 dans cette liste sont exactement les 24 permutations possible de φ_0. On obtient la propriété suivante.

Propriété 2.3.4. *Si φ est un morphisme de taille 2 similaire à φ_0 qui vérifie la condition Cupisca, alors φ est une substitution de φ_0.*

Grâce à cette liste nous avons cherché les morphismes candidats de taille 2 sur les alphabets d'entiers $\{a, b, c, d\}$ vérifiant $0 = a < b < c < d < 100$. Nous n'avons trouvé que des candidats similaires à φ_0. Cependant, plus l'écart entre les lettres de l'alphabet augmente, plus nous sommes amenés à tester de longs préfixes pour détecter des cubes additifs dans les points fixes des morphismes pris dans la liste des 288 et non similaires

à φ_0. Par exemple, pour le point fixe infini du morphisme défini par

$$a \mapsto cd \quad ; \quad b \mapsto bd \quad ; \quad c \mapsto bc \quad ; \quad d \mapsto aa,$$

en considérant les valuations $a = 0$, $b = 18$, $c = 81$ et $d = 99$, il est nécessaire de tester un préfixe de longueur 151006 pour trouver un cube additif.

Tout ceci nous amène à poser la question suivante.

Question 2.3.1. *Un morphisme de taille* 2 *vérifiant la condition* Cupisca *est-il nécessairement similaire à φ_0 ?*

Si la réponse à cette question était oui, cela voudrait dire que pour n'importe quelle valuation des lettres des $288 - 24 = 264$ morphismes non similaires à φ_0 et dont le point fixe infini semble éviter les cubes abéliens (en tout cas il n'en contient aucun dans son préfixe de longueur 10000), le point fixe infini possède un cube additif non abélien. Cette propriété nous paraîtrait étonnante mais les recherches que nous avons menées jusqu'à présent ne nous permettent pas de répondre à la question.

Avant de clore ce chapitre explicatif, il est nécessaire de faire deux remarques. La première est qu'un aspect de la Question 4 a déjà été traitée par M. Rao et M. Rosenfeld. Grâce à des éléments introduits dans des articles de A. Abekane, J.D. Currie et N. Rampersad [1, 18, 19] ils montrent que la question de l'évitabilité des puissances additives ou abéliennes est décidable pour tout morphisme vérifiant certaines conditions [39, Théorème 1]. Notre travail diffère du leur par le fait que nous souhaitons garder un schéma de preuve le plus proche possible de celui de [16] et voir dans quelle mesure il s'applique. La deuxième remarque est que pour nos expérimentations informatiques il est plus efficace de chercher des morphismes candidats et d'étudier leurs préfixes que d'essayer d'appliquer une des preuves de [16] ou [39]. La détection de cubes additifs dans les premiers termes des points fixes infinis permet d'éliminer très rapidement la plupart des morphismes.

Chapitre 3

Un mot infini sans cube additif sur l'alphabet $\{0, 1, 2, 6\}$

Dans ce chapitre nous allons étudier le cas d'un morphisme particulier similaire à φ_0 pour comprendre le fonctionnement du schéma de preuve développé par J. Cassaigne et al. [16] et l'appliquer à un autre cas. Dans le Chapitre 4 nous verrons qu'il peut en fait être utilisé sur tout morphisme similaire à φ_0 et ainsi nous démontrerons le Théorème 2.3.1 et répondrons à la Question 4 posée page 17. Il nous a semblé, pour un souci de clarté, préférable d'étudier d'abord un exemple pour ensuite généraliser. L'exemple que nous utilisons est un morphisme substitution de φ_0. L'effet des substitutions n'est pas trivial, par exemple si nous appliquons à φ_0 la substitution qui change la lettre 3 en 2 et conserve les autres lettres on obtient le morphisme

$$\varphi(0) = 02 \quad ; \quad \varphi(1) = 42 \quad ; \quad \varphi(2) = 1 \quad ; \quad \varphi(4) = 01$$

dont le point fixe infini contient un cube additif mis en évidence ici :

$$\overrightarrow{\varphi^{\omega}}(0) = 0214 \cdot 20 \cdot 11 \cdot 02 \cdot 4242021011011021420242420242 42 \cdots$$

Nous travaillons dans ce chapitre avec l'alphabet $\Sigma = \{0, 1, 2, 6\}$ et le morphisme $\varphi_6 : \{0, 1, 2, 6\}^* \to \{0, 1, 2, 6\}^*$ déjà utilisé page 12 défini par

$$\varphi_6(0) = 2 \quad ; \quad \varphi_6(1) = 62 \quad ; \quad \varphi_6(2) = 10 \quad ; \quad \varphi_6(6) = 60$$

et qui engendre le point fixe infini

$$\mathbf{w_6} = \overrightarrow{\varphi_6^{\omega}}(6) = 6021062260101060262262260210601010601010 \cdots$$

Ce morphisme est une substitution de φ_0. Il est nécessaire, pour déterminer si $\mathbf{w_6}$ est effectivement un mot infini sans cube additif, de trouver un procédé qui permette de ramener cette question à un problème fini. C'est ce procédé que nous allons expliquer ici. La première étape consiste à passer de ce point fixe infini à un graphe permettant l'étude du problème des cubes additifs.

3.1 D'un mot infini à un graphe infini

La notion de parent dans les points fixes infinis de morphisme est expliquée en Section 1.4. Basée sur cette notion, nous donnons la définition suivante.

Définition 3.1.1. *Étant donnée une position $p \in \mathbb{N}$, sa suite ancestrale $(p_i)_{i\in\mathbb{N}}$ est la suite dont le premier terme est $p_0 = p$ et telle que p_{i+1} est la position parent de p_i.*

Cette suite est ultimement stationnaire car elle est, par définition, décroissante et à valeurs dans $\mathbb{N}$. Sa limite est toujours 0. On représente graphiquement la façon dont se construit le mot $\mathbf{w_6}$ en construisant un graphe dont les sommets sont les positions dans $\mathbb{N}$, la racine est la position 0 et où les arêtes lient les parents à leurs enfants. Ce graphe, que nous notons $\mathcal{T}$, est en fait un arbre dont la racine est 0 (si on fait exception de la boucle sur ce sommet). On représente les premiers niveaux de cet arbre en Figure 3.1.

Notons que les arêtes sont étiquetées par les préfixes propres des images (i.e., les préfixes des images, possiblement vides mais différents de l'image complète, voir Définition 1.3.1). En ordonnant les arêtes partant d'un même sommet par taille croissante des préfixes propres les étiquetant, on obtient l'ordre des enfants d'une position. Nous verrons grâce au Lemme 3.1.6 qu'il existe une bijection entre les enfants d'une position et les préfixes propres de son image. Un chemin dans le graphe entre une position et la racine correspond à la suite ancestrale de cette position.

Nous allons maintenant voir dans quelle mesure un cube additif peut être identifié dans le graphe que nous venons de créer et comment on anticipe son apparition grâce aux suites ancestrales. On rappelle que $\mathbf{w_6}[p]$ désigne la p-ième lettre du mot $\mathbf{w_6}$ en commençant à partir de 0, par exemple $\mathbf{w_6}[0] = 6$ et $\mathbf{w_6}[1] = 0$. On rappelle également que $\mathbf{w_6}[p, q) = \mathbf{w_6}[p]\mathbf{w_6}[p+1]\cdots\mathbf{w_6}[q-1]$, par exemple $\mathbf{w_6}[0, 3) = 602$. Les définitions peuvent être retrouvées page 7. Dans le début de $\mathbf{w_6}$, on met en évidence un carré additif constitué de deux facteurs de taille 7 que nous notons u_1 et u_2,

$$\mathbf{w_6} = 60210 \cdot \underbrace{6226010}_{u_1} \cdot \underbrace{1060262}_{u_2} \cdot 2622602\ldots$$

Considérons le premier facteur de ce carré. Il s'agit de $\mathbf{w_6}[5, 12)$. Nous posons les deux définitions suivantes.

Définition 3.1.2. *Étant données p et q deux positions dans $\mathbb{N}$ avec $p < q$, le facteur parent de $\mathbf{w_6}[p, q)$ est le facteur $\mathbf{w_6}[\mathtt{Par}(p), \mathtt{Par}(q))$.*

Définition 3.1.3. *Soit $n \in \mathbb{N}^*$ et soient $p_1, \ldots, p_n$ des positions dans $\mathbb{N}$. Le n-uplet parent de $(p_1, \ldots, p_n)$ est $(\mathtt{Par}(p_1), \ldots \mathtt{Par}(p_n))$.*

Le parent du couple de positions $(5, 12)$ est donc le couple constitué du parent de 5 et du parent de 12, c'est-à-dire $(3, 7)$ et $\mathbf{w_6}[3, 7)$ est le facteur parent de $\mathbf{w_6}[5, 12)$. Les couples de positions nous seront utiles pour parcourir des produits de graphes que nous définissons comme suit.

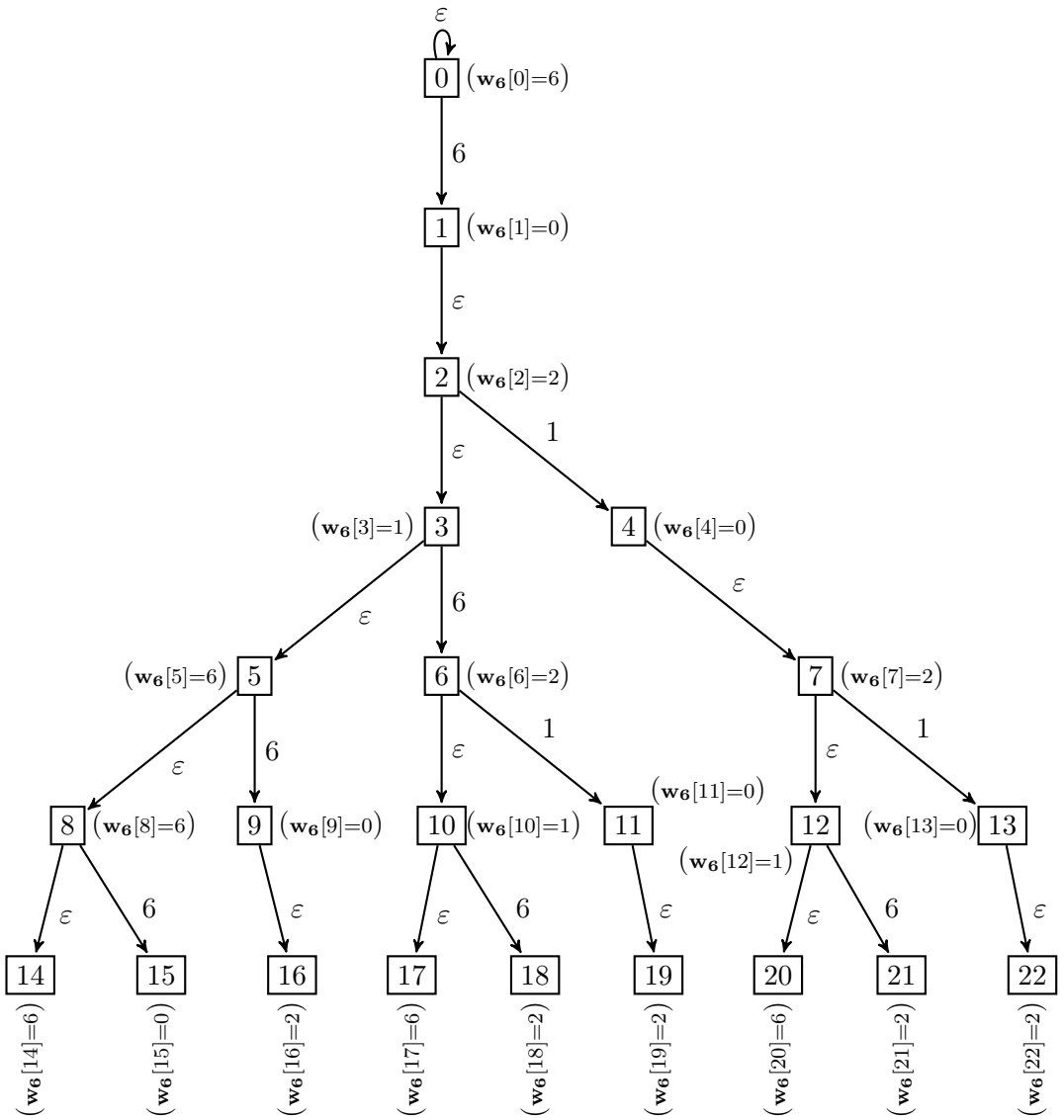


Figure 3.1 – Les 7 premiers niveaux de $\mathcal{T}$.

Définition 3.1.4. *Soient $\mathcal{G}_1$ et $\mathcal{G}_2$ deux graphes, on définit le produit tensoriel $\mathcal{G}_1 \times \mathcal{G}_2$ avec*

$$V(\mathcal{G}_1 \times \mathcal{G}_2) = V(\mathcal{G}_1) \times V(\mathcal{G}_2)$$
$$E(\mathcal{G}_1 \times \mathcal{G}_2) = E(\mathcal{G}_1) \times E(\mathcal{G}_2).$$

De plus, si e_1 et e_2 sont étiquetés respectivement par l_1 et l_2, alors l'arrête (e_1, e_2) sera étiquetée par (l_1, l_2).

Les couples de positions dans $\mathbf{w_6}$ sont donc des sommets dans le graphe $\mathcal{T}^2 = \mathcal{T} \times \mathcal{T}$. Remonter dans $\mathcal{T}^2$ un chemin entre un sommet (p, q) et la racine (avec $p < q$) revient à parcourir la suite ancestrale du facteur $\mathbf{w_6}[p, q)$. Le but est de trouver une représentation commode qui nous permette à la fois de situer les puissances additives dans $\mathbf{w_6}$ et de retrouver comment elles ont été engendrées afin de pouvoir conclure sur leur apparition ou non. Le carré u_1u_2 est constitué des deux facteurs $\mathbf{w_6}[5, 12)$ et $\mathbf{w_6}[12, 19)$, c'est-à-dire dans $\mathcal{T}^2$ des deux couples $(5, 12)$ et $(12, 19)$. Dans une puissance additive les facteurs sont toujours consécutifs. La fonction parent est croissante, donc les parents de deux facteurs consécutifs sont toujours des facteurs consécutifs dans le même ordre. Plutôt que de considérer deux couples, on considère donc le triplet de positions $(5, 12, 19)$. En remontant dans $\mathcal{T}^3$ d'un sommet vers la racine, on obtient les suites ancestrales d'un double facteur (qui peut être possiblement un carré additif) et on retrouve son origine. Pour l'exemple du triplet $(5, 12, 19)$ la suite ancestrale est constante égale à $(0, 0, 0)$ à partir du septième terme,

$$\begin{aligned}
\mathtt{Par}^6(5, 12, 19) &= \mathtt{Par}^5(3, 7, 11) \\
&= \mathtt{Par}^4(2, 4, 6) \\
&= \mathtt{Par}^3(1, 2, 3) \\
&= \mathtt{Par}^2(0, 1, 2) \\
&= \mathtt{Par}^1(0, 0, 1) \\
&= (0, 0, 0).
\end{aligned}$$

Pour détecter un cube additif, on cherche non plus dans $\mathcal{T}^3$ mais dans $\mathcal{T}^4$ si un chemin peut l'engendrer, on tient compte ainsi des contraintes liées au morphisme puisque l'arbre $\mathcal{T}$ le représente et que le morphisme ne construit pas un mot au hasard mais son point fixe qui possède une certaine structure.

Notre objectif est de construire un graphe que nous pourrons parcourir en un temps réduit par des programmes informatiques afin de déterminer si le mot contient des cubes additifs. Le problème est que pour une taille k donnée, le nombre de chemins de longueur k dans $\mathcal{T}^4$ est infini car les sommets de $\mathcal{T}$ sont en bijection avec $\mathbb{N}$. Ce nombre infini de chemins est en contradiction avec la volonté de parcourir le graphe en un temps fini. Cependant, $\mathcal{T}$ est "relativement répétitif" dans le sens où la disposition des arêtes partant d'un sommet ne peut se faire que de quatre façons différentes (une seule arête si le sommet est la position d'un 0 dans $\mathbf{w_6}$, deux arêtes sinon). Pour utiliser cette répétition, on quotiente $\mathcal{T}$ par la relation d'équivalence suivante : deux sommets s_1 et s_2 de $\mathcal{T}$ sont équivalents si et seulement si $\mathbf{w_6}[s_1] = \mathbf{w_6}[s_2]$, c'est-à-dire si les positions s_1 et s_2 correspondent à la même lettre dans $\mathbf{w_6}$. On obtient le graphe $\mathcal{Q}$, représenté en Figure 3.2, dont les sommets sont les lettres de l'alphabet $\{0, 1, 2, 6\}$ et où il existe une arête de la lettre a vers la lettre b si $b \in \varphi_6(a)$. Notons que seul l'étiquetage des sommets et des arêtes change par rapport au même graphe pour φ_0, la forme du graphe reste la même.

L'avantage de $\mathcal{Q}$ par rapport à $\mathcal{T}$ est que le nombre de chemins d'une taille fixée dans $\mathcal{Q}$ est fini, mais l'inconvénient est que nous avons perdu les informations sur les positions.

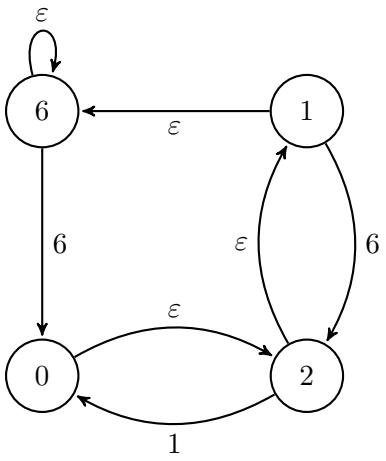


Figure 3.2 – Le graphe orienté $\mathcal{Q}$.

Étant donné que la projection de $\mathcal{T}$ dans $\mathcal{Q}$ n'est pas injective, on ne peut pas à partir d'un chemin dans $\mathcal{Q}^4$ reconstruire un chemin dans $\mathcal{T}^4$.

Définissons l'homomorphisme de graphes w qui envoie un sommet p de $\mathcal{T}$ sur $\mathbf{w_6}[p]$ dans $\mathcal{Q}$ et une arête $p \xrightarrow{\alpha} q$ de $\mathcal{T}$ sur $\mathbf{w_6}[p] \xrightarrow{\alpha} \mathbf{w_6}[q]$ dans $\mathcal{Q}$ (c'est-à-dire l'homomorphisme correspondant à la projection de $\mathcal{T}$ dans $\mathcal{Q}$). On obtient le résultat suivant.

Proposition 3.1.5. *Pour toute position p fixée, il existe une bijection, induite par w, que nous noterons $\tilde{w}_p$ entre les chemins partant de p dans $\mathcal{T}$ et ceux partant de $\mathbf{w_6}[p]$ dans $\mathcal{Q}$.*

Démonstration. Nous notons ici (et pour la suite de ce chapitre) M la matrice d'incidence de φ_6, elle est semblable à celle de φ_0 et on a

$$M = \mathrm{Mat}(\varphi_6) = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

On pose l'application $\sigma : \mathbb{N} \to \mathbb{Z}^4$ définie par

$$\sigma(p) = \psi(\mathbf{w_6}[0, p)) = \begin{pmatrix} |\mathbf{w_6}[0,p)|_0 \\ |\mathbf{w_6}[0,p)|_1 \\ |\mathbf{w_6}[0,p)|_2 \\ |\mathbf{w_6}[0,p)|_6 \end{pmatrix}.$$

Commençons par démontrer le lemme suivant.

Lemme 3.1.6. *Étant donnée une position p dans $\mathbb{N}$, il existe une bijection f entre les enfants de p et les préfixes propres de $\varphi_6(\mathbf{w_6}[p])$ définie par*

$$\begin{aligned} f : \mathit{Enf}_p &\longrightarrow \{0, 1, 2, 6\}^* \\ q &\longmapsto \mathbf{w_6}\left[\left|\varphi_6\left(\mathbf{w_6}[0, p)\right)\right|, q\right) \end{aligned}$$

avec $Enf_p = \{|\varphi_6(\mathbf{w_6}[0,p))|, \cdots, |\varphi_6(\mathbf{w_6}[0,p+1))|-1\}$. *De plus, si* q *est un enfant de* p *et* $f(q)$ *est le préfixe propre de* $\varphi_6(\mathbf{w_6}[p])$ *lié à* q *par la bijection, on a :*

$$\sigma(q) = M\sigma(p) + \psi(f(q)).$$

Démonstration. Le morphisme φ_6 envoie la lettre $\mathbf{w_6}[p]$ sur le facteur

$$\mathbf{w_6}\Big[\big|\varphi_6\left(\mathbf{w_6}[0,p)\right)\big|, \big|\varphi_6\left(\mathbf{w_6}[0,p+1)\right)\big|\Big).$$

En effet, $\varphi_6(\mathbf{w_6}[0,p+1)) = \varphi_6(\mathbf{w_6}[0,p))\varphi_6(\mathbf{w_6}[p])$ donc $\varphi_6(\mathbf{w_6}[p])$ correspond au suffixe qu'il faut ajouter à $\varphi_6(\mathbf{w_6}[0,p))$ pour obtenir $\varphi_6(\mathbf{w_6}[0,p+1))$, c'est-à-dire $\mathbf{w_6}[|\varphi_6(\mathbf{w_6}[0,p))|, |\varphi_6(\mathbf{w_6}[0,p+1))|)$. Les enfants de p sont donc les positions q telles que $q \in \mathrm{Enf}_p = \{|\varphi_6(\mathbf{w_6}[0,p))|, \cdots, |\varphi_6(\mathbf{w_6}[0,p+1))|-1\}$ et l'application

$$\begin{aligned} f : \mathrm{Enf}_p &\longrightarrow \{0,1,2,6\}^* \\ q &\longmapsto \mathbf{w_6}\Big[|\varphi_6(\mathbf{w_6}[0,p))|, q\Big) \end{aligned}$$

est une bijection entre les enfants de p et les préfixes propres de $\varphi_6(\mathbf{w_6}[p])$. Par définition

$$\begin{aligned} \sigma(q) &= \psi(\mathbf{w_6}[0,q)) \\ &= \psi\Big(\mathbf{w_6}\big[0,|\varphi_6(\mathbf{w_6}[0,p))|\big)\mathbf{w_6}\big[|\varphi_6(\mathbf{w_6}[0,p))|,q\big)\Big) \\ &= \psi\Big(\mathbf{w_6}\big[0,|\varphi_6(\mathbf{w_6}[0,p))|\big)\Big) + \psi\Big(\mathbf{w_6}\big[|\varphi_6(\mathbf{w_6}[0,p))|,q\big)\Big) \\ &= \psi\big(\varphi_6(\mathbf{w_6}[0,p))\big) + \psi\big(f(q)\big) \\ &= M\psi\big(\mathbf{w_6}[0,p)\big) + \psi\big(f(q)\big) \\ &= M\sigma(p) + \psi(a). \end{aligned}$$

□

Par définition, w préserve les étiquettes. Soit p une position fixée dans $\mathcal{T}$, démontrons que l'application $\tilde{w}_p$ qui envoie chaque arête d'un chemin partant de p sur son image par w est une bijection. Tout d'abord elle est surjective par définition car la projection de $\mathcal{T}$ dans $\mathcal{Q}$ l'est. Il suffit donc de montrer qu'elle est injective. Soit un chemin $p_0 \xrightarrow{l_1} p_1 \xrightarrow{l_2} \cdots p_n$ dans $\mathcal{T}$ et soit $w(p_0) \xrightarrow{l_1} w(p_1) \xrightarrow{l_2} \cdots w(p_n)$ son image par $\tilde{w}_p$. Supposons qu'il existe un chemin différent $p_0 \xrightarrow{l'_1} p'_1 \xrightarrow{l'_2} \cdots p'_n$ qui donne la même image. Comme $\tilde{w}_p$ préserve les étiquettes, $l'_i = l_i$ pour tout $i \in [\![1,n]\!]$. Comme les chemins sont différents, il existe $k \in [\![0,n-1]\!]$ tel que $p_k = p'_k$ mais $p_{k+1} \neq p'_{k+1}$. Cependant ces deux chemins sont étiquetés l_{k+1}. Étant donné qu'il existe un unique chemin dans $\mathcal{T}$ partant de p_k et étiqueté l_{k+1} (d'après le Lemme 3.1.6), on obtient $p_{k+1} = p'_{k+1}$, ce qui contredit l'hypothèse de départ et termine la preuve de la bijectivité de $\tilde{w}_p$. □

On définit l'application

$$\begin{aligned} W : \mathcal{T}^4 &\longrightarrow \mathcal{Q}^4 \\ (p_1,p_2,p_3,p_4) &\longmapsto (w(p_1),w(p_2),w(p_2),w(p_4)). \end{aligned}$$

De la même façon, pour un quadruplet $\mathbf{p} = (p_1,p_2,p_3,p_4)$ de positions fixé, il existe une bijection $\tilde{W}\mathbf{p} = (\tilde{w}_{p_1},\tilde{w}_{p_2},\tilde{w}_{p_3},\tilde{w}_{p_4})$ entre les chemins partant de $\mathbf{p}$ dans $\mathcal{T}^4$ et ceux partant de $W(\mathbf{p}) = (\mathbf{w_6}[p_1],\mathbf{w_6}[p_2],\mathbf{w_6}[p_3],\mathbf{w_6}[p_4])$ dans $\mathcal{Q}^4$. Nous allons utiliser cette bijection, mais un quadruplet dans $\mathcal{Q}^4$ ne nous donne des informations que sur les lettres au début et à la fin de chaque terme du triple facteur (possiblement un cube additif) que nous considérions au départ. Nous avons besoin de conserver les informations sur toutes les lettres du cube et nous utilisons pour cela le vecteur de Parikh. On pourrait travailler dans $\{0,1,2,6\}^4 \times \mathbb{Z}^4 \times \mathbb{Z}^4 \times \mathbb{Z}^4$ avec des éléments constitués d'un sommet de $\mathcal{Q}^4$ et des vecteurs de Parikh des 3 facteurs du cube. En fait il n'est pas utile de conserver les 3 vecteurs de Parikh. Notons $b_0c_0d_0$ le cube additif, il suffit de conserver $\psi(b_0)-\psi(c_0)$ et $\psi(c_0)-\psi(d_0)$. En effet, le fait que $b_0c_0d_0$ soit un cube additif est équivalent au fait que les vecteurs $\psi(b_0)-\psi(c_0)$ et $\psi(c_0)-\psi(d_0)$ soient dans l'ensemble

$$\mathfrak{L}_{\varphi_6} := \{\mathbf{v} \in \mathbb{Z}^4 : (1,1,1,1).\mathbf{v} = 0 \text{ et } (0,1,2,6).\mathbf{v} = 0\}.$$

3.2 D'un graphe infini à un graphe fini

3.2.1 Un graphe bijectivement lié à $\mathcal{T}^4$

Nous travaillons maintenant dans un graphe dont l'ensemble des sommets est $\{0,1,2,6\}^4 \times \mathbb{Z}^4 \times \mathbb{Z}^4$, ce qui est toujours infini. Notre objectif est donc de construire un graphe fini qui conserve également les informations pour nous permettre de déterminer si $\mathbf{w_6}$ possède ou non un cube additif. Notons qu'il est nécessaire de définir ce graphe fini de façon à ce qu'il existe une bijection entre les chemins partant d'un quadruplet fixé dans $\mathcal{T}^4$ et ceux partant de l'image de ce quadruplet dans ce graphe à définir. Nous allons tout d'abord expliquer comment construire un nouveau graphe, démontrer que la bijection citée précédemment existe et réduire l'étude à un sous-graphe fini.

L'utilisation de l'algèbre linéaire pour exprimer les contraintes liées au morphisme via la matrice d'incidence permet d'obtenir la proposition suivante.

Proposition 3.2.1 (Prouvée en Partie 3.3)**.** *Soient b et c deux facteurs de même taille et de même somme, soient $(b_i)_{i\in\mathbb{N}}$ et $(c_i)_{i\in\mathbb{N}}$ leurs suites ancestrales respectives. Alors, pour tout entier i on a $\|\psi(b_i)-\psi(c_i)\|_2 \leq 6.29$.*

Notons

$$\mathfrak{U} = \{\mathbf{x} \in \mathbb{Z}^4 \text{ tels que } \|\mathbf{x}\|_2 \leq 6.29\},$$

la boule dans $\mathbb{Z}^4$ de rayon 6.29. Il suffit de travailler dans un graphe dont les sommets sont dans $\{0,1,2,6\}^4 \times \mathfrak{U} \times \mathfrak{U}$. Ce graphe est fini, on peut le parcourir en un temps fini

et conclure sur la présence ou non d'un cube additif dans $\mathbf{w_6}$. On définit le graphe $\mathcal{G}$ par

$$V(\mathcal{G}) = V(\mathcal{Q}^4) \times \mathbb{Z}^4 \times \mathbb{Z}^4$$

et il existe une arête de $(\mathbf{c}', \mathbf{u}', \mathbf{v}')$ vers $(\mathbf{c}, \mathbf{u}, \mathbf{v})$ s'il existe une arête $\mathbf{c}' \to \mathbf{c}$ dans $\mathcal{Q}^4$ étiquetée par $(a_1, \dots, a_4)$, telle que les équations

$$\begin{aligned}\mathbf{u} &= M\mathbf{u}' + \psi(a_3) - 2\psi(a_2) + \psi(a_1),\\ \mathbf{v} &= M\mathbf{v}' + \psi(a_4) - 2\psi(a_3) + \psi(a_2),\end{aligned}$$

soient vérifiées.

Montrons, toujours en suivant le schéma de preuve de [16], qu'il existe bien une bijection entre les chemins partant d'un quadruplet fixé dans $\mathcal{T}^4$ et ceux partant de l'image de ce quadruplet. On sait que la suite ancestrale d'un triple facteur $b_0c_0d_0$ délimité par un vecteur (p_1, p_2, p_3, p_4) nous donne un chemin entre $(0,0,0,0)$ et (p_1, p_2, p_3, p_4) dans $\mathcal{T}^4$. On a de plus $p_1 < p_2 < p_3 < p_4$ si aucun facteur de ce triple facteur n'est vide. En utilisant l'application σ définie page 33, on obtient que les vecteurs de Parikh de chaque facteur sont donnés par

$$\begin{aligned}\psi(b_0) &= \sigma(p_2) - \sigma(p_1)\\ \psi(c_0) &= \sigma(p_3) - \sigma(p_2)\\ \psi(d_0) &= \sigma(p_4) - \sigma(p_3).\end{aligned}$$

Si p est un enfant d'une position p' de $\mathcal{T}$ et si on note $l_{(p',p)} = \mathbf{w_6}[\eta(p'), p)$ le préfixe propre de $\mathbf{w_6}[\eta(p'), \eta(p'+1)) = \varphi_6(\mathbf{w_6}[p'])$ qui étiquette l'arête entre p' et p dans $\mathcal{T}$, on a, d'après le Lemme 3.1.6 :

$$\sigma(p) = M\sigma(p') + \psi(l_{(p',p)}).$$

Notons $b_1 = \mathbf{w_6}[p_1', p_2')$ le facteur parent de $b_0 = \mathbf{w_6}[p_1, p_2)$ (ce qui signifie que $p_1' = \texttt{Par}(p_1)$ et $p_2' = \texttt{Par}(p_2)$). On obtient

$$\begin{aligned}\psi(b_0) &= \sigma(p_2) - \sigma(p_1)\\ &= M\sigma(p_2') + \psi(l_{(p_2',p_2)}) - M\sigma(p_1') - \psi(l_{(p_1',p_1)})\\ &= M\psi(b_1) + \psi(l_{(p_2',p_2)}) - \psi(l_{(p_1',p_1)}),\end{aligned}$$

et avec les mêmes notations

$$\psi(c_0) = \sigma(p_3) - \sigma(p_2) = M\psi(c_1) + \psi(l_{(p_3',p_3)}) - \psi(l_{(p_2',p_2)}),$$

et de même pour d_0. Il en découle

$$\begin{aligned}\psi(c_0) - \psi(b_0) &= M\big(\psi(c_1) - \psi(b_1)\big) + \psi(l_{(p_3',p_3)}) - 2\psi(l_{(p_2',p_2)}) + \psi(l_{(p_1',p_1)})\\ \psi(d_0) - \psi(c_0) &= M\big(\psi(d_1) - \psi(c_1)\big) + \psi(l_{(p_4',p_4)}) - 2\psi(l_{(p_3',p_3)}) + \psi(l_{(p_2',p_2)}),\end{aligned}$$

ce qui est équivalent à

$$\sigma(p_3) - 2\sigma(p_2) + \sigma(p_1) = M\big(\sigma(p'_3) - 2\sigma(p'_2) + \sigma(p'_1)\big) + \psi(l_{(p'_3,p_3)}) - 2\psi(l_{(p'_2,p_2)}) + \psi(l_{(p'_1,p_1)}) \tag{3.1}$$

$$\sigma(p_4) - 2\sigma(p_3) + \sigma(p_2) = M\big(\sigma(p'_4) - 2\sigma(p'_3) + \sigma(p'_2)\big) + \psi(l_{(p'_4,p_4)}) - 2\psi(l_{(p'_3,p_3)}) + \psi(l_{(p'_2,p_2)}), \tag{3.2}$$

Considérons maintenant l'application

$$\begin{aligned} g : \mathcal{T}^4 &\longrightarrow \mathcal{G} \\ \mathbf{p} &\longmapsto (W(\mathbf{p}), \sigma(p_3) - 2\sigma(p_2) + \sigma(p_1), \sigma(p_4) - 2\sigma(p_3) + \sigma(p_2)), \end{aligned}$$

et montrons qu'elle induit une bijection $\tilde{g}_{\mathbf{p}}$ entre les chemins partant d'une position donnée $\mathbf{p}$ dans $\mathcal{T}^4$ et ceux partant de $g(\mathbf{p})$ dans $\mathcal{G}$. Nous commençons par démontrer que

$$g(\mathbf{p}) \longrightarrow g(\mathbf{q}) \text{ arête dans } \mathcal{G} \iff \mathbf{p} \longrightarrow \mathbf{q} \text{ arête dans } \mathcal{T}^4.$$

L'implication de la droite vers la gauche découle des définitions. S'il existe une arête $\mathbf{p} \overset{(a_1,a_2,a_3,a_4)}{\longrightarrow} \mathbf{q}$ dans $\mathcal{T}^4$, elle correspond bijectivement à une arête $W(\mathbf{p}) \overset{(a_1,a_2,a_3,a_4)}{\longrightarrow} W(\mathbf{q})$ dans $\mathcal{Q}^4$ et les égalités 3.1 et 3.2 nous assurent qu'il existe une arête $g(\mathbf{p}) \longrightarrow g(\mathbf{q})$ dans $\mathcal{G}$.

Pour l'implication de la gauche vers la droite, si $(g(\mathbf{p}), g(\mathbf{q}))$ est une arête dans $\mathcal{G}$ alors il existe une arête $W(\mathbf{p}) \overset{(a_1,a_2,a_3,a_4)}{\longrightarrow} W(\mathbf{q})$ dans $\mathcal{Q}^4$ et donc une arête $\mathbf{p} \overset{(a_1,a_2,a_3,a_4)}{\longrightarrow} \mathbf{q}$ dans $\mathcal{T}^4$.

3.2.2 Situer les cubes additifs dans $\mathcal{G}$

L'application g est donc un homomorphisme de graphes entre $\mathcal{T}^4$ et $\mathcal{G}$ et la bijection $\tilde{g}_{\mathbf{p}}$ découle de la bijection $\tilde{W}_{\mathbf{p}}$. Chaque chemin dans $\mathcal{G}$ partant de $g(\mathbf{p})$ est l'image d'un chemin de $\mathbf{p}$ à un certain $\mathbf{q}$ dans $\mathcal{T}^4$ et donc se termine à $g(\mathbf{q})$. Pour un triple facteur délimité par $\mathbf{q}$, il existe un chemin entre $(0,0,0,0)$ et $\mathbf{q}$ dans $\mathcal{T}^4$ qui s'identifie à un chemin entre $g(0,0,0,0)$ et $g(\mathbf{q})$ dans $\mathcal{G}$. Malheureusement, la réciproque n'est pas vraie car étant donné un tel chemin on ne peut pas déduire que $\mathbf{q}$ délimite un triple facteur (nous n'obtenons pas nécessairement $q_1 < q_2 < q_3 < q_4$).

Rappelons que la fonction parent est croissante, étant données deux suites ancestrales $(p_i)_{i\in\mathbb{N}}$ et $(q_i)_{i\in\mathbb{N}}$, si $p_0 \leq q_0$ alors $p_i \leq q_i$ pour tout $i \in \mathbb{N}$. Par contraposée, si $p_i < q_i$ alors $p_0 < q_0$. On donne le lemme suivant :

Lemme 3.2.2 ([16]). *Étant donnée la suite ancestrale de facteurs $\{b_i\}_{i=0}^{\infty}$ pour un facteur non vide b_0, il existe k tel que b_j soit non vide pour $0 \leq j \leq k$ et vide pour $j > k$. De plus, b_k est alors la différence de deux préfixes propres d'un mot $\varphi_6(l)$ pour un certain $l \in \{0,1,2,6\}$, il y a deux possibilités :* 1 *ou* 6.

Démonstration. Supposons que $b_0 = w[p_0, q_0)$ et soient $\{p_i\}_{i=0}^{\infty}$ et $\{q_i\}_{i=0}^{\infty}$ les suites ancestrales associées respectivement à p_0 et q_0. Une suite ancestrale rencontre inévitablement

0, donc il existe $n_0 \in \mathbb{N}$ tel que pour tout $n > n_0$ on ait $p_n = q_n = 0$ et donc b_n est un facteur vide. Soit b_k le dernier facteur non vide, par définition pour tout $j > k$, b_j est vide. Mais, par définition de b_k, on a $p_k < q_k$ et donc d'après les propriétés de la fonction parent, $p_j < q_j$ pour tout $j < k$, ce qui implique que b_j sera non vide pour tout $j < k$.

Par définition des parents, $\eta(q_{k+1}) \leq q_k < \eta(q_k)$, donc $y = \mathbf{w_6}[\eta(q_{k+1}), q_k)$ est un préfixe propre de $\mathbf{w_6}[\eta(q_{k+1}), \eta(q_{k+1}+1)) = \varphi_6(\mathbf{w_6}[q_{k+1}])$. De même $x = \mathbf{w_6}[\eta(p_{k+1}), p_k)$ est un préfixe propre de $\mathbf{w_6}[\eta(p_{k+1}), \eta(p_{k+1}+1)) = \varphi_6(\mathbf{w_6}[p_{k+1}])$. Mais b_{k+1} est vide donc $p_{k+1} = q_{k+1}$, les préfixes x et y commencent donc à la même position et $xb_k = y$. Mais les préfixes propres sont dans $\{\varepsilon, 1, 6\}$ et b_k est un suffixe non vide d'un préfixe propre, d'où $b_k \in \{1, 6\}$. □

Ce lemme peut s'appliquer à un triple facteur $b_0c_0d_0$ délimité par $\mathbf{p}$ en le considérant comme un seul grand facteur délimité par p_1 et p_4. On pose

$$X := \{(p_1,p_2,p_3,p_4) \in \mathbb{N}^4 : \ p_1 \leq p_2 \leq p_3 \leq p_4,\ p_1 < p_4 \\ \text{et } \mathtt{Par}(p_1) = \mathtt{Par}(p_2) = \mathtt{Par}(p_3) = \mathtt{Par}(p_4)\}.$$

X est alors l'ensemble des triples facteurs non vides qui ont un parent vide. Considérons maintenant l'ensemble $g(X)$. Si $\mathbf{p}$ est dans X alors il délimite un facteur qui est un sous-mot de $\varphi_6(l)$, $l \in \{0,1,2,6\}$. Puisque $\varphi_6(0), \varphi_6(1), \varphi_6(2)$ et $\varphi_6(6)$ apparaissent tous dans les 7 premiers symboles de $\mathbf{w_6}$, il suffit de regarder les $g(\mathbf{p})$ pour $\mathbf{p} \in X$ tel que $p_4 < 7$. Le Lemme 3.2.2 nous permet d'affirmer que $b_0c_0d_0$ est soit 1, soit 6, donc le triple facteur est soit $\mathbf{w_6}[0,1)$, soit $\mathbf{w_6}[3,4)$, soit $\mathbf{w_6}[5,6)$. D'où :

$$g(X) = \{g(0,0,0,1), g(0,0,1,1), g(0,1,1,1), \\ g(3,3,3,4), g(3,3,4,4), g(3,4,4,4), \\ g(5,5,5,6), g(5,5,6,6), g(5,6,6,6)\}.$$

Un chemin dans $\mathcal{T}^4$ de $(0,0,0,0)$ à $\mathbf{q}$, où $\mathbf{q}$ délimite un triple facteur non vide, doit passer par X. On peut donc considérer un chemin partant d'un élément de X et non de $(0,0,0,0)$. Si notre chemin commence à $\mathbf{p} \in X$ on a $p_1 < p_4$ et donc $q_1 < q_4$, ce qui nous aide à montrer que $q_1 < q_2 < q_3 < q_4$. Démontrons le théorème suivant qui nous permet de considérer la détection des cubes additifs dans $\mathbf{w_6}$ comme un problème de parcours de graphe. On rappelle qu'on note $\Sigma = \{0,1,2,6\}$.

Théorème 3.2.3 ([16]). *Étant donné un cube additif délimité par* $\mathbf{q} \in \mathcal{T}^4$*, il y a un chemin dans* $\mathcal{G}$ *de* $g(\mathbf{p})$ *à* $g(\mathbf{q})$ *pour un certain* $\mathbf{p} \in X$ *avec* $g(\mathbf{q})$ *dans l'ensemble* $Z := \Sigma^4 \times \mathfrak{L}_{\varphi_6} \times \mathfrak{L}_{\varphi_6}$*. Réciproquement, si on a un chemin dans* $\mathcal{G}$ *démarrant d'un élément de* $g(X)$ *et se terminant à* $v \in Z$ *alors il y a un cube additif délimité par* $\mathbf{q}$ *où* $v = g(\mathbf{q})$*.*

Démonstration. Supposons que $\mathbf{q}$ délimite un cube additif, alors la différence entre deux vecteurs de Parikh des facteurs appartient à $\mathfrak{L}_{\varphi_6}$, donc $g(\mathbf{q}) \in Z$. D'après le Lemme 3.2.2 il existe un ancêtre $\mathbf{p}$ de $\mathbf{q}$ dans $\mathcal{T}^4$ tel que $\mathbf{p}$ délimite un triple facteur non vide mais

avec $\texttt{Par}(\mathbf{p})$ délimitant $\varepsilon\varepsilon\varepsilon$. Par construction, $\mathbf{p} \in X$ et le chemin dans $\mathcal{T}^4$ de $\mathbf{p}$ à $\mathbf{q}$ est envoyé sur un chemin dans $\mathcal{G}$ de $g(\mathbf{p})$ à $g(\mathbf{q})$.

Pour la réciproque, supposons qu'on ait un chemin dans $\mathcal{G}$ commençant à $g(\mathbf{p})$ et se terminant à $v \in Z$. Il correspond à un chemin dans $\mathcal{T}^4$ de $\mathbf{p}$ jusqu'à un certain $\mathbf{q}$ avec $v = g(\mathbf{q})$. Mais par définition de Z, les différences des vecteurs de Parikh des facteurs délimités par $\mathbf{q}$ sont dans $\mathfrak{L}_{\varphi_6}$, ce qui implique que tous les facteurs ont la même longueur. Or $q_1 < q_4$ car $p_1 < p_4$ et donc les facteurs ont une longueur strictement positive, d'où $q_1 < q_2 < q_3 < q_4$ et $\mathbf{q}$ délimite un cube additif. □

3.2.3 La réduction à un sous-graphe fini

Malgré l'utilité de la relation entre les cubes additifs et les chemins dans $\mathcal{G}$ il y a un bémol majeur : $\mathcal{G}$ est un graphe infini (car $\mathbb{Z}^4$ l'est) et, comme dit précédemment, il est difficile d'appliquer des algorithmes pour les graphes aux graphes infinis. La Proposition 3.2.1 montre que si b_0 et c_0 sont des facteurs de même longueur et même somme, alors leurs suites ancestrales $\{b_i\}_{i=0}^{\infty}$ et $\{c_i\}_{i=0}^{\infty}$ sont telles que $\psi(b_i) - \psi(c_i) \in \mathfrak{U}$ où $\mathfrak{U}$ est un ensemble fini qu'on peut énumérer. En fait si $\mathbf{p}$ délimite un cube additif et que $\mathbf{q}$ désigne n'importe quelle étape du chemin de $(0, 0, 0, 0)$ à $\mathbf{p}$ délimitant le facteur $d_1d_2d_3$ alors $\psi(d_i) - \psi(d_j) \in \mathfrak{U}$ pour $i, j \in \{1, 2, 3\}$. Posons $g(\mathbf{q}) = (\mathbf{c}, \mathbf{u}, \mathbf{v})$, on remarque que par définition $\mathbf{u}$, $\mathbf{v}$ et $\mathbf{u} + \mathbf{v}$ sont dans $\mathfrak{U}$. Définissons l'ensemble

$$H := \{(\mathbf{c}, \mathbf{u}, \mathbf{v}) : \mathbf{u}, \mathbf{v}, \mathbf{u} + \mathbf{v} \in \mathfrak{U} \text{ et } \mathbf{c} \in \Sigma^4\}.$$

Par définition, $g(\mathbf{q}) \in H$ et on peut donc restreindre notre recherche au sous-graphe de $\mathcal{G}$ engendré par H que nous noterons $\mathcal{G}'$. L'ensemble H est un sous-ensemble de $\Sigma^4 \times \mathfrak{U} \times \mathfrak{U}$ il contient donc au plus $4^4 \times |\mathfrak{U}| \times |\mathfrak{U}|$ éléments. $\mathcal{G}'$ est donc un graphe fini et on déduit du Théorème 3.2.3 le corollaire suivant :

Corollaire 3.2.4. *Étant donné un cube additif délimité par* $\mathbf{q}$*, il y a un chemin dans* $\mathcal{G}'$ *de* $g(\mathbf{p})$ *à* $g(\mathbf{q})$ *où* $g(\mathbf{p}) \in A := g(X) \cap H$ *et* $g(\mathbf{q}) \in B := Z \cap H$*. Réciproquement, si on a un chemin dans* $\mathcal{G}$ *commençant dans* A *et se terminant en* $\beta \in B$ *alors il y a un cube additif délimité par* $\mathbf{q}$ *tel que* $g(\mathbf{q}) = \beta$*.*

On peut énumérer les éléments de $\mathfrak{U}$ et donc a fortiori ceux de $\Sigma^4 \times \mathfrak{U} \times \mathfrak{U}$ pour ensuite les trier afin de construire H. On construit ensuite $B = Z \cap H$ en testant l'appartenance de chaque élément de H à Z. De même on vérifie facilement que $g(X) \subset H$ et donc $A = g(X) \cap H = g(X)$. Comme $\mathcal{G}'$ est de taille finie, on détermine en utilisant un ordinateur s'il existe ou non un chemin dans $\mathcal{G}'$ qui va de A à B.

Nous avons donc montré qu'il est possible de ramener l'étude des cubes additifs dans $\mathbf{w_6}$ au parcours d'un graphe fini, à condition d'être dans un cas où la Proposition 3.2.1 est valable.

3.3 Obtenir la borne nécessaire

3.3.1 Les applications propres

Nous démontrons dans ce paragraphe que des caractéristiques inhérentes à φ_6 (caractéristiques communes également à φ_0) permettent d'obtenir la Proposition 3.2.1. La matrice d'incidence de φ_6 vérifie

$$\mathrm{Mat}(\varphi_6) = P^{-1}\mathrm{Mat}(\varphi_0)P \quad \text{où} \quad P = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

En fait M est semblable à $\mathrm{Mat}(\varphi_0)$ et plus précisément φ_6 est une substitution de φ_0. Un des intérêts des matrices d'incidence est que, étant donné w un mot, on a $\psi(\varphi_6(w)) = M\psi(w)$ et on peut exprimer avec des outils de l'algèbre linéaire l'application du morphisme. Les valeurs propres de M sont, en valeurs tronquées :

$$\begin{aligned} \lambda_1 &\doteq -1,505068413621473 \\ \lambda_2 &\doteq 1,690284494616614 \\ \lambda_3 &\doteq 0,4073919595024293 + 0,4765653259296430i \\ \lambda_4 &\doteq 0,4073919595024293 - 0,4765653259296430i \end{aligned}$$

On note Q la matrice des vecteurs propres de M, normalisés de façon à ce qu'ils aient une norme (euclidienne) de 1. Si on note Λ la matrice diagonale composée des valeurs propres de M, alors $M = Q\Lambda Q^{-1}$ avec, toujours en valeurs tronquées :

$$Q^{-1} \doteq \begin{pmatrix} 0.4524 & 0.5724 & -0.6809 & -0.1806 \\ 0.3537 & 0.6568 & 0.5978 & 0.5124 \\ 0.4923 - 0.3775i & -0.6690 + 0.2094i & 0.0206 - 0.3884i & -0.1933 + 0.7925i \\ 0.4923 + 0.3775i & -0.6690 - 0.2094i & 0.0206 + 0.3884i & -0.1933 - 0.7925i \end{pmatrix}.$$

Notons que le fait de travailler avec des valeurs approchées n'est pas problématique ici. Nos opérations sont numériquement stables (des additions, des multiplications mais pas de divisions par un nombre proche de zéro) et si on utilise une précision suffisamment grande au départ on obtient une précision suffisamment grande à l'arrivée. On définit l'endomorphisme $\tau : \mathbb{C}^4 \to \mathbb{C}^4$ comme l'application correspondant à la multiplication à gauche par Q^{-1}. On définit aussi, pour $j = 1, 2, 3, 4$, les applications $\tau_j : \mathbb{C}^4 \to \mathbb{C}$ telles que, pour tout $x \in \mathbb{C}^4$ on ait $\tau(x) = (\tau_1(x), \tau_2(x), \tau_3(x), \tau_4(x))$. Nous désignons les applications τ_i par l'appellation *applications propres*.

3.3.2 Obtenir des bornes pour les applications τ_3 et τ_4

De [16] on extrait le théorème suivant.

Théorème 3.3.1 ([16]). *Si b et c sont deux facteurs, non nécessairement consécutifs, alors :*

$$|\tau_3(\psi(b) - \psi(c))| \leq C_3$$
$$|\tau_4(\psi(b) - \psi(c))| \leq C_3,$$

où $C_3 \doteq 2.1758$ est une constante.

Cette preuve correspond à celle développée dans [16]. Elle utilise la bijection f construite dans le Lemme 3.1.6 entre les chemin partant d'une position p dans $\mathcal{T}$ et ceux partant de $\mathbf{w_6}[p]$ dans $\mathcal{Q}$. Remarquons d'abord que τ_4 est simplement le conjugué complexe de τ_3 donc il suffit de construire la preuve pour τ_3.

Pour établir cette démonstration nous avons besoin du résultat suivant.

Corollaire 3.3.2. *Si $p_0p_1 \cdots p_k$ est un chemin dans $\mathcal{T}$ avec les étiquettes a_1, a_2, $\cdots$, a_k, alors :*

$$\sigma(p_k) = \sum_{i=1}^{k} M^{k-i}\psi(a_i) + M^k\sigma(p_0).$$

Si $\{p_i\}_{i=0}^{\infty}$ est la suite ancestrale d'une position p_0 et soient a_i les étiquettes pour chaque chemin de p_i à p_{i+1} pour chaque i, alors

$$\sigma(p_0) = \sum_{i=0}^{k-1} M^i\psi(a_i) + M^k\sigma(p_k)$$

et pour k assez grand $p_k = 0$, donc $\sigma(p_k) = 0$ et on obtient

$$\sigma(p_0) = \sum_{i=0}^{\infty} M^i\psi(a_i). \tag{3.3}$$

Supposons qu'on applique ce corollaire à une suite ancestrale $\{p_i\}_{i=0}^{\infty}$ et soient a_i les étiquettes pour chaque chemin de p_i à p_{i+1} pour chaque i, pour k assez grand on obtient

$$\sigma(p_0) = \sum_{i=0}^{\infty} M^i\psi(a_i). \tag{3.4}$$

Supposons que le facteur b soit délimité par p_0 et q_0,

$$\sigma(p_0) = \sum_{i=0}^{\infty} M^i\psi(a_i) \quad \text{et} \quad \sigma(q_0) = \sum_{i=0}^{\infty} M^i\psi(a'_i).$$

Ceci donne

$$\begin{aligned}\psi(b) &= \sigma(q_0) - \sigma(p_0)\\ &= \sum_{i=0}^{\infty} M^i\psi(a'_i) - \sum_{i=0}^{\infty} M^i\psi(a_i)\\ &= \sum_{i=0}^{\infty} M^i\delta_i,\end{aligned}$$

où $\delta_i = \psi(a'_i) - \psi(a_i)$. En appliquant τ_3 on obtient :

$$\tau_3(\psi(b)) = \tau_3\left(\sum_{i=0}^{\infty} M^i \delta_i\right) = \sum_{i=0}^{\infty} \lambda_3^i \tau_3(\delta_i).$$

En effet, comme $Q^{-1}M = \Lambda Q^{-1}$, on a $\tau_j(Mx) = \lambda_j \tau_j(x)$ pour tout vecteur x. En appliquant le module :

$$\begin{aligned} |\tau_3(\psi(b))| &= \left|\sum_{i=0}^{\infty} \lambda_3^i \tau_3(\delta_i)\right| \\ &\leq \left|\sum_{i=0}^{8} \lambda_3^i \tau_3(\delta_i)\right| + \left|\sum_{i=9}^{\infty} \lambda_3^i \tau_3(\delta_i)\right|. \end{aligned}$$

Trouvons maintenant une borne pour les deux parties en utilisant des techniques différentes. Pour la série infinie il suffit de borner $|\tau_3(\delta_i)|$. Puisque $\delta_i = \psi(a'_i) - \psi(a_i)$ il y a peu de valeurs possibles. En effet, a_i et a'_i sont dans l'ensemble $\{\varepsilon, 1, 6\}$, d'où :

$$|\tau_3(\delta_i)| \leq \max\Big\{|\tau_3(\psi(s) - \psi(t))| : s,t \in \{\varepsilon,1,6\}\Big\} = |\tau_3(0,0,0,1)| \doteq 0.81582, \qquad (3.5)$$

le vecteur $(0,0,0,1)$ étant obtenu pour $s = 6$ et $t = \varepsilon$. On note ce maximum $C \doteq 0.81582$. On obtient :

$$\left|\sum_{i=9}^{\infty} \lambda_3^i \tau_3(\delta_i)\right| \leq \sum_{i=9}^{\infty} |\lambda_3|^i |\tau_3(\delta_i)| \leq C \sum_{i=9}^{\infty} |\lambda_3|^i = \frac{C|\lambda_3|^9}{1-|\lambda_3|} \doteq 0.032736. \qquad (3.6)$$

En ce qui concerne la somme finie, on a :

$$\begin{aligned} \left|\sum_{i=0}^{8} \lambda_3^i \tau_3(\delta_i)\right| &= \left|\tau_3\left(\sum_{i=0}^{8} M^i \delta_i\right)\right| \\ &= \left|\tau_3\left(\sum_{i=0}^{8} M^i \psi(a'i)\right) - \tau_3\left(\sum_{i=0}^{8} M^i \psi(ai)\right)\right| \\ &= |\tau_3(\alpha') - \tau_3(\alpha)|, \end{aligned}$$

où $\alpha = \sum_{i=0}^{8} M^i \psi(a_i)$ et $\alpha' = \sum_{i=0}^{8} M^i \psi(a'_i)$. On définit l'ensemble de vecteurs $\mathfrak{D}_l(\varphi_6) \subseteq \mathbb{Z}^4$ par :

$$\mathfrak{D}_l(\varphi_6) = \left\{ \quad \sum_{i=0}^{l-1} M^i \psi(a_i) \quad : \quad a_{l-1}\cdots a_0 \text{ les arêtes d'un chemin dans } \mathcal{T} \quad \right\},$$

ce qui est équivalent à

$$\mathfrak{D}_l(\varphi_6) = \left\{ \quad \sum_{i=0}^{l-1} M^i \psi(a_i) \quad : \quad a_{l-1}\cdots a_0 \text{ les arêtes d'un chemin dans } \mathcal{Q} \quad \right\}$$

car à tout chemin de longueur l dans $\mathcal{T}$ correspond bijectivement (via $\tilde{w}$) un chemin de même longueur et avec les mêmes arêtes dans $\mathcal{Q}$. Or α et α' sont deux éléments de $\mathfrak{D}_9(\varphi_6)$, on borne donc le résultat par

$$\max\left\{|\tau_3(u)-\tau_3(v)|: \quad u,v\in\mathfrak{D}_9(\varphi_6)\right\}\doteq 1.05517 \tag{3.7}$$

obtenu pour $u=(24,12,30,24)$ et $v=(13,5,25,17)$ après une recherche sur les 301 éléments de $\mathfrak{D}_9(\varphi_6)$ via une recherche informatique (un programme en `C++` parcours le graphe $\mathcal{Q}$ et construit l'ensemble $\mathfrak{D}_9(\varphi_6)$ de façon instantanée sur un ordinateur de bureau standard). En combinant les deux bornes ainsi obtenues on trouve :

$$\begin{aligned}|\tau_3(\psi(b))| \leq& \left|\sum_{i=0}^{8}\lambda_3^i\tau_3(\delta_i)\right|+\left|\sum_{i=9}^{\infty}\lambda_3^i\tau_3(\delta_i)\right|\\ \leq& 1.05517\cdots+0.03273\cdots=1.0879\cdots=C_3/2.\end{aligned} \tag{3.8}$$

En utilisant le même raisonnement on obtient $|\tau_3(\psi(c))|\leq C_3/2$ et donc

$$|\tau_3(\psi(b)-\psi(c))|\leq|\tau_3(\psi(b))|+|\tau_3(\psi(c))|\leq C_3\doteq 2.1758. \tag{3.9}$$

Nous obtenons ici la même borne que dans l'article de J. Cassaigne et al. [16, Théorème 3.1]. Ceci est dû au fait que la substitution qui permet de passer de φ_0 à φ_6 agit non seulement sur M mais aussi sur Q^{-1} et sur les vecteurs de Parikh. On obtient $\mathfrak{D}_9(\varphi_6)=\mathfrak{D}_9(\varphi_0)$.

3.3.3 Obtenir des bornes pour les applications τ_1 et τ_2

Théorème 3.3.3. *Soient b et c deux facteurs de même taille et de même somme dans $\mathbf{w_6}$, soient $\{b_i\}_{i=0}^{\infty}$ et $\{c_i\}_{i=0}^{\infty}$ les suites ancestrales correspondantes, alors*

$$|\tau_1(\psi(b_i)-\psi(c_i))|\leq C_1, \tag{3.10}$$
$$|\tau_2(\psi(b_i)-\psi(c_i))|\leq C_2. \tag{3.11}$$

pour chaque i, avec

$$C_1=\frac{2|\tau_1(0,1,0,1)|}{|\lambda_1|-1}\doteq 2.9818$$
$$C_2=\frac{2|\tau_2(0,1,0,0)|}{|\lambda_2|-1}\doteq 1.9032$$

Cette preuve est également similaire à celle de [16], l'intérêt de la retranscrire est de montrer son adaptabilité à d'autres morphismes. On a définit le réseau euclidien

$$\mathfrak{L}_{\varphi_6}:=\{\mathbf{v}\in\mathbb{Z}^4:(1,1,1,1).\mathbf{v}=0 \text{ et } (0,1,2,6).\mathbf{v}=0\}.$$

En fait on peut écrire

$$\mathfrak{L}_{\varphi_6} = \{m(1, -2, 1, 0) + n(5, -6, 0, 1) \text{ avec } m, n \in \mathbb{Z}\} \tag{3.12}$$
$$= \{m(1, -2, 1, 0) + n(1, 2, -4, 1) \text{ avec } m, n \in \mathbb{Z}\}. \tag{3.13}$$

C'est dans cet ensemble que se situent les différences des vecteurs de Parikh de deux facteurs de même taille et même somme. L'écriture 3.12 nous permet déjà d'exprimer les vecteurs de $\mathfrak{L}_{\varphi_6}$ en fonction des entiers m et n et d'obtenir des bornes pour les applications τ_1 et τ_2 mais l'écriture 3.13 nous donne des bornes plus précises.

Remarque 3.3.1. *Ce passage utilise l'alphabet explicitement en plus d'utiliser le morphisme, il demandera un travail d'adaptation plus important au Chapire 4 lorsqu'il s'agira de montrer que ce schéma de preuve fonctionne pour de nombreux morphismes sur une infinité d'alphabets.*

Par construction $\psi(b_0) - \psi(c_0)$ appartient à $\mathfrak{L}_{\varphi_6}$ donc on peut l'écrire sous la forme $m(1, -2, 1, 0) + n(1, 2, -4, 1)$ pour $m, n \in \mathbb{Z}$ et utiliser le Théorème 3.3.1 pour borner $|\tau_3(\psi(b_0) - \psi(c_0))|$.

Lemme 3.3.4. *Soit* $\mathbf{v} = m(1, -2, 1, 0) + n(1, 2, -4, 1)$. *Alors*

$$|\tau_3(\mathbf{v})| \geq \alpha|m| \tag{3.14}$$
$$|\tau_3(\mathbf{v})| \geq \beta|n| \tag{3.15}$$

où $\alpha \doteq 1.4914$ *et* $\beta \doteq 2.1657$ *sont des constantes.*

Démonstration. Si $m = 0$ l'équation est trivialement vraie. Supposons donc que $m \neq 0$. On remarque que $\tau_3(1, 2, -4, 1) \doteq 0.72928 + 1.20278i \neq 0$. D'où

$$\begin{aligned}
|\tau_3(\mathbf{v})| &= |m\tau_3(1, -2, 1, 0) + n\tau_3(1, 2, -4, 1)| \\
&= |m||\tau_3(1, 2, -4, 1)| \left| \frac{n}{m} + \frac{\tau_3(1, -2, 1, 0)}{\tau_3(1, 2, -4, 1)} \right| \\
&\geq |m||\tau_3(1, 2, -4, 1)| \left| \operatorname{Im}\left(\frac{n}{m} + \frac{\tau_3(1, -2, 1, 0)}{\tau_3(1, 2, -4, 1)} \right) \right| \\
&\geq |m||\tau_3(1, 2, -4, 1)| \left| \operatorname{Im}\left(\frac{\tau_3(1, -2, 1, 0)}{\tau_3(1, 2, -4, 1)} \right) \right| \\
&\geq \alpha|m|
\end{aligned}$$

où

$$\alpha = |\tau_3(1, 2, -4, 1)| \left| \operatorname{Im}\left(\frac{\tau_3(1, -2, 1, 0)}{\tau_3(1, 2, -4, 1)} \right) \right| \doteq 1.17151.$$

De la même façon, $n \neq 0$ et $\tau_3(1,2,-4,1) \neq 0$. On a

$$\begin{aligned}
|\tau_3(\mathbf{v})| &= |m\tau_3(1,-2,1,0) + n\tau_3(1,2,-4,1)| \\
&= |n||\tau_3(1,-2,1,0)| \left| \frac{m}{n} + \frac{\tau_3(1,2,-4,1)}{\tau_3(1,-2,1,0)} \right| \\
&\geq |n||\tau_3(1,-2,1,0)| \left| \mathrm{Im} \left(\frac{m}{n} + \frac{\tau_3(1,2,-4,1)}{\tau_3(1,-2,1,0)} \right) \right| \\
&\geq |n||\tau_3(1,-2,1,0)| \left| \mathrm{Im} \left(\frac{\tau_3(1,2,-4,1)}{\tau_3(1,-2,1,0)} \right) \right| \\
&\geq \beta|n|
\end{aligned}$$

où

$$\beta = |\tau_3(1,-2,1,0)| \left| \mathrm{Im} \left(\frac{\tau_3(1,2,-4,1)}{\tau_3(1,-2,1,0)} \right) \right| \doteq 1.40619.$$

Nous avons montré que $|\tau_3(\psi(b_0) - \psi(c_0))| \leq 2.1758$, ce qui implique

$$1.17151|m| \leq |\tau_3(\psi(b_0) - \psi(c_0))| \leq 2.1758,$$

c'est-à-dire

$$|m| \leq \frac{2.1758}{1.17151} \doteq 1.85726.$$

Donc $m \in \{-1,0,1\}$ et de la même façon

$$|n| \leq \frac{2.1758}{1.40619} \doteq 1.54730.$$

Donc $n \in \{-1,0,1\}$, ce qui laisse seulement 9 vecteurs possibles répertoriés dans le Tableau 3.1.

Tableau 3.1 – 9 vecteurs possibles classés selon la valeur de $|\tau_3(v)|$.

v	m	n	$\lvert\tau_1(v)\rvert$	$\lvert\tau_2(v)\rvert$	$\lvert\tau_3(v)\rvert$
$(0,0,0,0)$	0	0	0	0	0
$(2,0,-3,1)$	1	1	2.76697	0.57378	1.40660
$(-2,0,3,-1)$	-1	-1	2.76697	0.57378	1.40660
$(-1,2,-1,0)$	-1	0	1.37332	0.36214	2.19770
$(1,-2,1,0)$	1	0	1.37332	0.36214	2.19770
$(-1,-2,4,-1)$	0	-1	4.14029	0.21164	2.63796
$(1,2,-4,1)$	0	1	4.14029	0.21164	2.63796
$(0,4,-5,1)$	-1	1	5.51362	0.15050	4.64747
$(0,-4,5,-1)$	1	-1	5.51362	0.15050	4.64747

Seuls 3 de ces 9 vecteurs satisfont la condition $|\tau_3(v)| \leq 2.1758$ donc $\psi(b_0) - \psi(c_0)$ doit être un de des trois vecteurs de l'ensemble suivant :

$$G := \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \begin{pmatrix} 2 \\ 0 \\ -3 \\ 1 \end{pmatrix}; \begin{pmatrix} -2 \\ 0 \\ 3 \\ -1 \end{pmatrix} \right\}.$$

Nous terminons la démonstration par récurrence sur i. Pour $i = 0$ il n'y a que trois possibilités pour $\psi(b_0) - \psi(c_0)$ qui satisfont toutes les trois les inégalités demandées dans le théorème. De plus

$$\begin{aligned}|\tau_2(\psi(b_{i+1}) - \psi(c_{i+1}))| &= \left|\tau_2\left(M^{-1}(\psi(b_i) - \psi(c_i) - \delta_i + \delta_i')\right)\right| \\ &= |\lambda_2|^{-1}\left|\tau_2(\psi(b_i) - \psi(c_i) - \delta_i + \delta_i')\right| \\ &\leq |\lambda_2|^{-1}\left(|\tau_2(\psi(b_i) - \psi(c_i)| + |\delta_i| + |\delta_i'|\right).\end{aligned}$$

En procédant comme pour l'inégalité 3.5 dans la preuve du Théorème 3.3.1 on obtenient que $|\tau_2(\delta_i)|$ et $|\tau_2(\delta_i')|$ sont bornés par $|\tau_2(0,1,0,0)| \doteq 0.65687$. D'après l'hypothèse de récurrence :

$$|\tau_2(\psi(b_i) - \psi(c_i))| \leq \frac{2|\tau_2(0,1,0,0)|}{|\lambda_2| - 1},$$

ce qui donne :

$$|\tau_2(\psi(b_{i+1}) - \psi(c_{i+1}))| \leq |\lambda_2|^{-1}\left(\frac{2|\tau_2(0,1,0,0)|}{|\lambda_2| - 1} + 2|\tau_2(0,1,0,0)|\right) = \frac{2|\tau_2(0,1,0,0)|}{|\lambda_2| - 1}.$$

Ceci termine la preuve par récurrence de l'inégalité 3.11 et par un raisonnement similaire on obtient l'inégalité 3.10.

□

3.3.4 Obtenir la borne finale et conclure

Puisque $\tau_j(\psi(b_i) - \psi(c_i))$ est borné pour $j \in \{1,2,3,4\}$, alors $\tau(\psi(b_i) - \psi(c_i))$ est borné également, et, comme τ est inversible, $\psi(b_i) - \psi(c_i)$ est borné. En effet, définissons l'ensemble

$$\mathfrak{U}_0 := \{\mathbf{u} \in \mathbb{Z}^4 : |\tau_i(\mathbf{u})| \leq C_i \text{ pour } 1 \leq i \leq 4\}.$$

Soit $\mathbf{u} = \sum_{i=1}^4 d_i e_i \in \mathfrak{U}_0$ écrit dans la base euclidienne (e_1, e_2, e_3, e_4) de $\mathbb{R}^4$. Nous notons λ_1', λ_2', λ_3' et λ_4' les valeurs propres de $\tau^*\tau$ (qui sont positives) et $(. \mid .)$ le produit scalaire usuel. On obtient

$$\begin{aligned}\|\tau\mathbf{u}\|^2 = (\mathbf{u} \mid \tau^*\tau\mathbf{u}) \quad &= \left(\sum_{i=1}^4 d_i e_i \mid \sum_{i=1}^4 d_i \tau^*\tau e_i\right) \\ &= \left(\sum_{i=1}^4 d_i e_i \mid \sum_{i=1}^4 \lambda_i' d_i e_i\right) \\ &= \sum_{i=1}^4 \lambda_i' d_i^2 \geq \min_{i\in\{1,2,3,4\}} \lambda_i' \|\mathbf{u}\|^2.\end{aligned}$$

Donc $\|\tau\mathbf{u}\|^2 \leq |\tau_1\mathbf{u}|^2 + |\tau_2\mathbf{u}|^2 + |\tau_3\mathbf{u}|^2 + |\tau_4\mathbf{u}|^2 \leq C_1^2 + C_2^2 + C_3^2 + C_4^2$, ce qui donne

Proposition 3.3.5. *Si* $\mathbf{u} \in \mathfrak{U}_0$ *alors* $\|\mathbf{u}\| \leq \left(\frac{C_1^2+C_2^2+C_3^2+C_4^2}{\min_{i\in\{1,2,3,4\}} \lambda_i'}\right)^{1/2} < \sqrt{39.455} < 6.29.$

Les Théorème 3.3.1 et Théorème 3.3.3 montrent que $\psi(b_i)-\psi(c_i) \in \mathfrak{U}_0$ et la Proposition 3.3.5 montre que $\mathfrak{U}_0 \subset \mathfrak{U}$, où $\mathfrak{U}$ est une boule de $\mathbb{Z}^4$ définie page 35. On énumère donc tous les éléments de $\mathfrak{U}$ en listant les vecteurs d'entiers u de norme euclidienne plus petite que $\sqrt{39.455}$ et tels que $|\tau_i(\mathbf{u})| \leq C_i$ pour chaque i. On trouve 497 éléments dans $\mathfrak{U}$. Pour trouver ces éléments il nous suffit de lister tous ceux de la boule dans $\mathbb{Z}^4$ de rayon $\sqrt{39.455}$ et de conserver ceux qui respectent les inégalités sur les applications $\tau_i, 1 \leq i \leq 4$.

Ainsi la Proposition 3.2.1 est démontrée, elle nous permet de prouver que les ensembles A, B et H utilisés dans le Corollaire 3.2.4 sont des ensembles finis. Via un programme informatique, on construit le graphe $\mathcal{G}'$ et on vérifie qu'il ne contient aucun chemin de A vers B. Nous discutons en conclusion du Chapitre 4 des points importants de cette programmation. L'existence d'un tel chemin étant équivalente à l'existence d'un cube additif dans $\mathbf{w_6}$, ce point fixe infini est bien dépourvu de cubes additifs. En fait on peut même remplacer $\mathfrak{U}$ par $\mathfrak{U}_0$ dans la construction des graphes au départ ce qui réduit le sous-graphe à étudier.

Le fait que le mot infini étudié soit engendré par un morphisme nous permet d'introduire le concept de parent et de remonter à l'origine des cubes additifs dans des graphes. Les caractéristiques inhérentes au morphisme étudié (ses valeurs propres, ses images...) permettent de borner le nombre de cas à étudier dans les graphes introduits. C'est ainsi que fonctionne le schéma de preuve développé en 2013 par J. Cassaigne et al. Nous avons montré ici qu'il pouvait parfaitement s'appliquer à un autre morphisme substitution de φ_0. En fait la substitution $s : \{0,1,3,4\}^* \to \{0,1,2,6\}^*$ telle que

$$s(0) = 6 \quad ; \quad s(1) = 2 \quad ; \quad s(3) = 0 \quad ; \quad s(4) = 1$$

vérifie $\varphi_0 = s^{-1} \circ \varphi_6 \circ s$. Par conséquent, $\mathbf{w_6}$ est l'image par s de $\mathbf{w_0}$. La substitution a beau se faire lettre à lettre, elle peut changer les comportements et notamment créer des nouvelles puissances additives. Nous mettons ci-dessous en évidence une puissance additive de $\mathbf{w_6}$ qui n'est pas dans $\mathbf{w_0}$:

$$\begin{aligned}
\mathbf{w_6} &= 6021062260101 \cdot \overbrace{06026}^{\Sigma=14} \cdot \overbrace{22622}^{\Sigma=14} \cdot\ 6021060101060101\cdots \\
\mathbf{w_0} &= 0314301103434 \cdot \underbrace{30310}_{\Sigma=7} \cdot \underbrace{11011}_{\Sigma=4} \cdot\ 0314303434303434\cdots
\end{aligned}$$

Il est donc faux de dire que puisqu'un morphisme est substitution de φ_0 il vérifie la condition Cupisca, mais cette question est décidable via un algorithme et c'est ce que nous montrons dans le Chapitre 4.

Chapitre 4

Revisiter l'algorithme de Cassaigne et al.

Dans ce chapitre, nous montrons que le schéma de preuve développé par J. Cassaigne et al. [16] permet de créer un algorithme qui prend en entrée un morphisme de taille 2 similaire à φ_0 et donne en sortie la réponse à la question de savoir si oui ou non ce morphisme vérifie la condition Cupisca. Nous considérons dans ce chapitre un morphisme $\varphi : \{a, b, c, d\}^* \to \{a, b, c, d\}^*$, de taille 2 et similaire à φ_0. On pose $\Sigma_\varphi = \{a, b, c, d\}$ et on supposera que $a = 0$ car on peut toujours retrancher a à toutes les lettres de l'alphabet sans changer le fait que le morphisme vérifie ou non la condition Cupisca. Une plus longue discussion sur l'équivalence entre les alphabets se trouve en Section 5.1.

4.1 Le morphisme et une première condition nécessaire

Le but de ce chapitre est de démontrer le Théorème 2.3.1 énoncé auparavant et laissé sans preuve.

Théorème 2.3.1. *Soit φ un morphisme de taille 2 similaire à φ_0, alors on peut décider algorithmiquement s'il vérifie la condition Cupisca.*

L'algorithme de décision utilise les outils développés par Cassaigne et al. dans [16], il prend en entrée le morphisme φ et décide en sortie s'il vérifie ou non la condition Cupisca. La Propriété 2.3.4 indique que si un morphisme de taille 2 similaire à φ_0 vérifie la condition Cupisca alors il est nécessairement une substitution de φ_0. Plus précisément, ce morphisme se trouve dans l'ensemble $\mathcal{E}$ des 24 morphismes répertoriés dans le Tableau 4.1 (on les obtient en appliquant à l'alphabet les 24 permutations possibles). Notons que ces morphismes produisent un point fixe infini qui évite les cubes abéliens puisqu'il s'agit de l'image de $\mathbf{w_0}$ par une substitution et que $\mathbf{w_0}$ évite les cubes additifs et donc les cubes abéliens. Ceci nous donne un premier test pour notre algorithme : si φ n'est pas dans $\mathcal{E}$ il ne vérifie pas la condition Cupisca. Nous obtenons les étapes explicitées dans l'Algorithme 1. On suppose dans cet algorithme que la fonction *FonctionTest* prend en entrée un morphisme substitution de φ_0 et décide s'il vérifie ou non la condition Cupisca.

Tableau 4.1 – Les 24 morphismes de l'ensemble $\mathcal{E}$.

$a \mapsto dc, b \mapsto c, c \mapsto ab, d \mapsto db$	$a \mapsto dc, b \mapsto bc, c \mapsto a, d \mapsto ba$
$a \mapsto db, b \mapsto ac, c \mapsto b, d \mapsto dc$	$a \mapsto db, b \mapsto a, c \mapsto cb, d \mapsto ca$
$a \mapsto d, b \mapsto cd, c \mapsto ca, d \mapsto ba$	$a \mapsto d, b \mapsto ba, c \mapsto bd, d \mapsto ca$
$a \mapsto cd, b \mapsto d, c \mapsto cb, d \mapsto ab$	$a \mapsto cd, b \mapsto bd, c \mapsto ba, d \mapsto a$
$a \mapsto cb, b \mapsto ad, c \mapsto cd, d \mapsto b$	$a \mapsto cb, b \mapsto a, c \mapsto da, d \mapsto db$
$a \mapsto c, b \mapsto dc, c \mapsto ba, d \mapsto da$	$a \mapsto c, b \mapsto ba, c \mapsto da, d \mapsto bc$
$a \mapsto bd, b \mapsto ca, c \mapsto cd, d \mapsto a$	$a \mapsto bd, b \mapsto bc, c \mapsto d, d \mapsto ac$
$a \mapsto bc, b \mapsto da, c \mapsto a, d \mapsto dc$	$a \mapsto bc, b \mapsto bd, c \mapsto ad, d \mapsto c$
$a \mapsto b, b \mapsto da, c \mapsto ca, d \mapsto cb$	$a \mapsto b, b \mapsto ca, c \mapsto db, d \mapsto da$
$a \mapsto ad, b \mapsto cd, c \mapsto ab, d \mapsto b$	$a \mapsto ad, b \mapsto ac, c \mapsto bd, d \mapsto c$
$a \mapsto ac, b \mapsto dc, c \mapsto b, d \mapsto ab$	$a \mapsto ac, b \mapsto ad, c \mapsto d, d \mapsto bc$
$a \mapsto ab, b \mapsto d, c \mapsto ad, d \mapsto cb$	$a \mapsto ab, b \mapsto c, c \mapsto db, d \mapsto ac$

Entrées : Un morphisme φ de taille 2
Résultat : Vrai si φ vérifie la condition Cupisca, Faux sinon

```
si φ similaire à φ0 alors
    si φ ∉ E alors
        retourner Faux ;
    sinon
        retourner FonctionTest(φ) ;
    fin
sinon
    retourner "Erreur : morphisme non pris en compte par l'algorithme." ;
fin
```

Algorithme 1 : Premières étapes de l'algorithme de test pour la condition Cupisca sur les morphismes de taille 2 similaires à φ_0.

Arrivé à ce stade de notre algorithme de décision, on suppose que le morphisme φ est une substitution de φ_0 et qu'il appartient donc à l'ensemble $\mathcal{E}$. Nous notons les images des lettres comme suit.

$$\begin{aligned}\varphi(a) &= w_a\\ \varphi(b) &= w_b\\ \varphi(c) &= w_c\\ \varphi(d) &= w_d.\end{aligned}$$

Avec $|w_l| \leq 2$ pour $l \in \Sigma_\varphi$ et $|w_0| + |w_a| + |w_b| + |w_c| = 7$. Ces deux conditions viennent du fait que φ est une substitution de φ_0. On sait également qu'il existe $l \in \Sigma_\varphi$ qui soit la première lettre de $\varphi(l)$ et telle que $|\varphi(l)| = 2$ afin que

$$\mathbf{w} = \overrightarrow{\varphi^\omega}(l) = \lim_{n\to\infty} \varphi^n(l)$$

soit le point fixe infini de φ qui évite les cubes abéliens. On note M_φ la matrice d'incidence du morphisme φ.

4.2 La création des graphes infinis

Nous noterons pour toute la suite s la substitution qui existe entre φ et φ_0 de telle sorte que

$$\varphi_0 = s^{-1} \circ \varphi \circ s.$$

Nous construisons, de la même façon que dans la Section 3.1, les graphes $\mathcal{T}_\varphi$ et $\mathcal{Q}_\varphi$ qui sont respectivement l'arbre représentant les liens entre les positions et leurs enfants dans $\mathbf{w}$ (représenté en Figure 4.1) et le graphe quotient de $\mathcal{T}_\varphi$ par la relation d'équivalence suivante : pour tous p et q dans $\mathbb{N}$, p est équivalent à q si et seulement si $\mathbf{w}[p] = \mathbf{w}[q]$ (représenté en Figure 4.2). On remarquera que les sommets de $\mathcal{Q}_\varphi$ et les étiquettes des deux graphes sont les images par la substitution de ceux pour les graphes de φ_0.

Il est important de noter que les graphes ont la même disposition des arêtes dans tous les cas et par conséquent, pour un n fixé, le même nombre de chemins de taille n. Le lemme suivant est toujours vrai et sa démonstration est tout à fait analogue à celle du Lemme 3.1.6.

Lemme 4.2.1. *Étant donnée une position p dans $\mathbb{N}$, il existe une bijection f entre les enfants de p et les préfixes propres de $\varphi(\mathbf{w}[p])$ définie par*

$$\begin{aligned} f : Enf_p &\longrightarrow \{a,b,c,d\}^* \\ q &\longmapsto \mathbf{w}[|\varphi(\mathbf{w}[0,p))|, q)) \end{aligned}$$

avec $Enf_p = \{|\varphi(\mathbf{w}[0,p))|, \cdots, |\varphi(\mathbf{w}[0,p+1))| - 1\}$. De plus, si q est un enfant de p et $f(q)$ est le préfixe propre de $\varphi(\mathbf{w}[p])$ lié à q par la bijection, on a :

$$\sigma(q) = M_\varphi \sigma(p) + \psi(f(q)).$$

Par conséquent l'homomorphisme de graphes w_φ qui envoie un sommet p de $\mathcal{T}_\varphi$ sur $\mathbf{w}[p]$ dans $\mathcal{Q}_\varphi$ et une arête $p \xrightarrow{\alpha} q$ de $\mathcal{T}_\varphi$ sur $\mathbf{w}[p] \xrightarrow{\alpha} \mathbf{w}[q]$ dans $\mathcal{Q}_\varphi$ induit toujours une bijection $\tilde{w}_p^\varphi$ entre les chemins partant de p dans $\mathcal{T}_\varphi$ et ceux partant de $\mathbf{w}[p]$ dans $\mathcal{Q}_\varphi$. De même, l'application

$$\begin{aligned} W_\varphi : \mathcal{T}_\varphi^4 &\longrightarrow \mathcal{Q}_\varphi^4 \\ (p_1,p_2,p_3,p_4) &\longmapsto (w_\varphi(p_1), w_\varphi(p_2), w_\varphi(p_2), w_\varphi(p_4)) \end{aligned}$$

induit également, pour un quadruplet $\mathbf{p} = (p_1,p_2,p_3,p_4)$ de positions fixé dans $\mathcal{T}_\varphi^4$, une bijection $\tilde{W}_\mathbf{p}^\varphi = (\tilde{w}_{p_1}^\varphi, \tilde{w}_{p_2}^\varphi, \tilde{w}_{p_3}^\varphi, \tilde{w}_{p_4}^\varphi)$ entre les chemins partant de $\mathbf{p}$ dans $\mathcal{T}_\varphi^4$ et ceux partant de $W_\varphi(\mathbf{p}) = (\mathbf{w}[p_1], \mathbf{w}[p_2], \mathbf{w}[p_3], \mathbf{w}[p_4])$ dans $\mathcal{Q}_\varphi^4$.

On peut continuer le processus et créer le graphe $\mathcal{G}_\varphi$ défini par

$$V(\mathcal{G}_\varphi) = V(\mathcal{Q}_\varphi^4) \times \mathbb{Z}^4 \times \mathbb{Z}^4$$

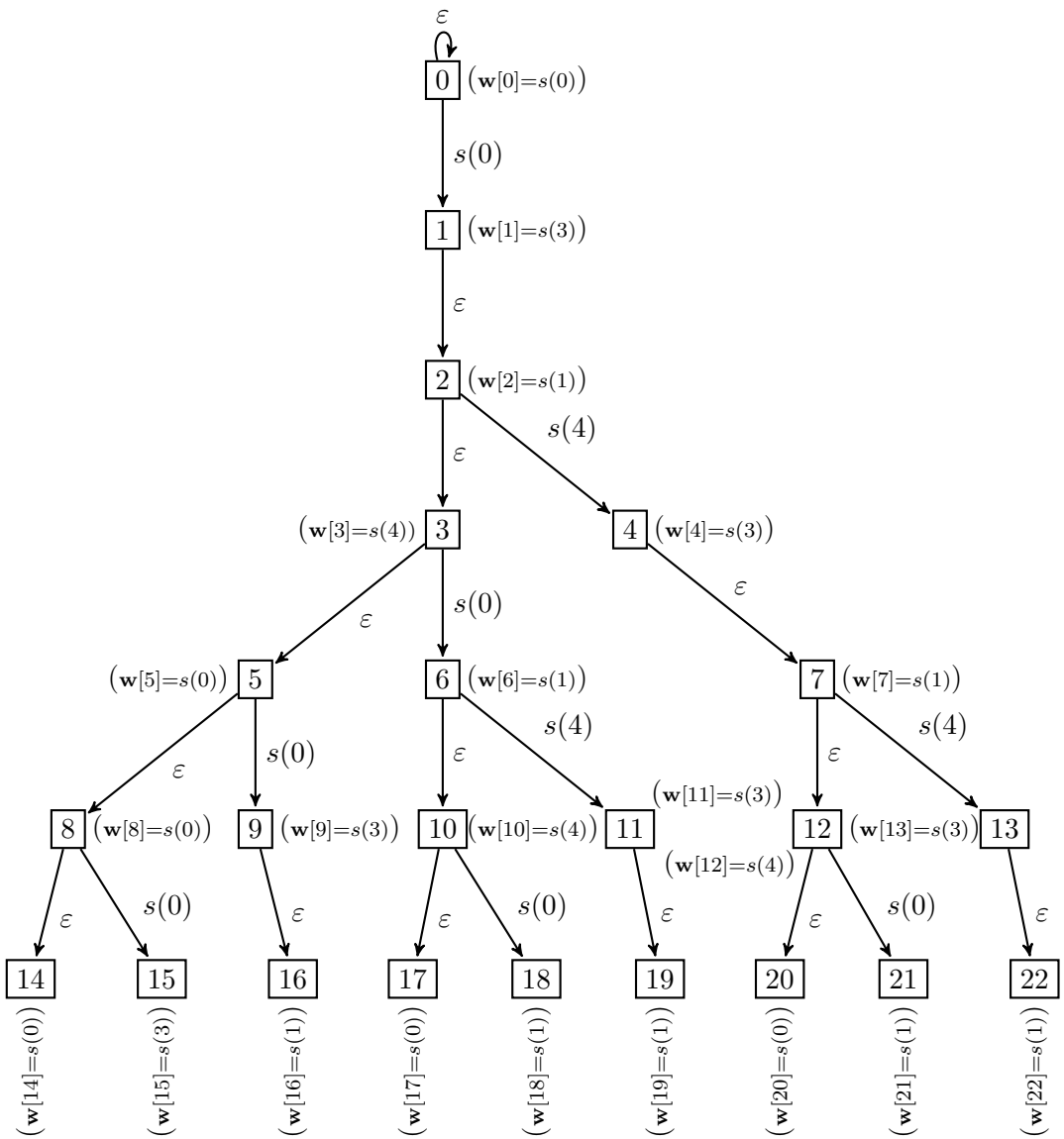


Figure 4.1 – Les 7 premiers niveaux de $\mathcal{T}_\varphi$.

et il existe une arête de $(\mathbf{c}', \mathbf{u}', \mathbf{v}')$ vers $(\mathbf{c}, \mathbf{u}, \mathbf{v})$ s'il existe une arête $\mathbf{c}' \to \mathbf{c}$ dans $\mathcal{Q}_\varphi^4$ étiquetée par $(a_1, \ldots, a_4)$, telle que les équations

$$\begin{aligned} \mathbf{u} &= M_\varphi \mathbf{u}' + \psi(a_3) - 2\psi(a_2) + \psi(a_1), \\ \mathbf{v} &= M_\varphi \mathbf{v}' + \psi(a_4) - 2\psi(a_3) + \psi(a_2), \end{aligned}$$

soient vérifiées. La bijection induite par W_φ étant définie et les graphes comportant le

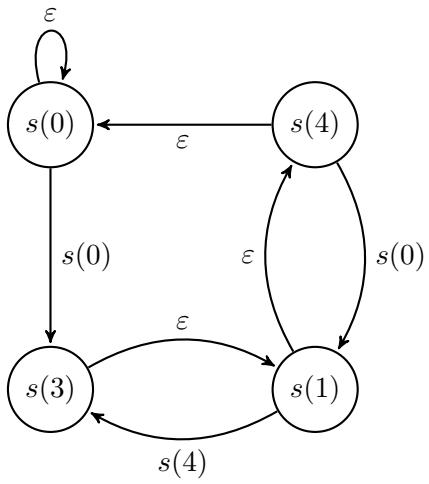


Figure 4.2 – Le graphe orienté $\mathcal{Q}_\varphi$.

même nombre de chemins pour n'importe quel choix de φ dans $\mathcal{E}$, l'application

$$\begin{aligned} g_\varphi : \mathcal{T}_\varphi^4 &\longrightarrow \mathcal{G}_\varphi \\ \mathbf{p} &\longmapsto (W_\varphi(\mathbf{p}), \sigma(p_3) - 2\sigma(p_2) + \sigma(p_1), \sigma(p_4) - 2\sigma(p_3) + \sigma(p_2)), \end{aligned}$$

induit également une bijection $\tilde{g}_{\mathbf{p}}^{\varphi}$ entre les chemins partant d'une position donnée $\mathbf{p}$ dans $\mathcal{T}_\varphi^4$ et ceux partant de $g_\varphi(\mathbf{p})$ dans $\mathcal{G}_\varphi$.

4.3 Situer les cubes additifs dans $\mathcal{G}_\varphi$

Si $b_0c_0d_0$ est un cube additif de $\mathbf{w}$, alors les vecteurs $\psi(b_0) - \psi(c_0)$ et $\psi(c_0) - \psi(d_0)$ sont dans l'ensemble

$$\mathfrak{L}_\varphi := \{\mathbf{v} \in \mathbb{Z}^4 : (1,1,1,1).\mathbf{v} = 0 \text{ et } (a,b,c,d).\mathbf{v} = 0\},$$

et réciproquement. De façon similaire au Lemme 3.2.2 nous obtenons le résultat suivant.

Lemme 4.3.1. *Étant donnée une suite ancestrale de facteurs $\{b_i\}_{i=0}^{\infty}$ pour un facteur non vide b_0, il existe k tel que b_j soit non vide pour $0 \leq j \leq k$ et vide pour $j > k$. De plus, b_k est alors la différence de deux préfixes propre d'un mot $\varphi(l)$ pour un certain $l \in \{a,b,c,d\}$, il y a deux possibilités : $s(0)$ ou $s(4)$.*

Démonstration. Nous discutons seulement dans cette preuve de la raison pour laquelle les deux possibilités sont $s(0)$ ou $s(4)$, le reste de la preuve est exactement similaire à celle du Lemme 3.2.2. Puisque $\varphi_0 = s^{-1} \circ \varphi \circ s$, les images w_a, w_b, w_c et w_d sont les images par s des quatre images de φ_0. Comme les préfixes propres non vides des images de φ_0 sont 0 et 4, les préfixes propres non vides des images de φ sont $s(0)$ et $s(4)$. □

Ce lemme peut également s'appliquer à un triple facteur $b_0c_0d_0$ délimité par $\mathbf{p} = (p_1, p_2, p_3, p_4)$ en le considérant comme un seul grand facteur délimité par p_1 et p_4. On pose

$$X_\varphi := \{(p_1,p_2,p_3,p_4) \in \mathbb{N}^4 : \ p_1 \leq p_2 \leq p_3 \leq p_4, \ p_1 < p_4 \\ \text{et } \mathtt{Par}(p_1) = \mathtt{Par}(p_) = \mathtt{Par}(p_3) = \mathtt{Par}(p_4)\}.$$

Comme $\mathbf{w}$ est l'image de $\mathbf{w_0}$ par s, les mots $\varphi(a), \varphi(b), \varphi(c)$ et $\varphi(d)$ apparaissent tous dans les 7 premiers symboles de $\mathbf{w}$. Pour donner l'ensemble $g_\varphi(X_\varphi)$ il suffit donc de regarder les $g_\varphi(\mathbf{p})$ pour $\mathbf{p} \in X_\varphi$ tel que $p_4 < 7$. Le Lemme 3.2.2 nous permet d'affirmer que $b_0c_0d_0$ est soit $s(0)$, soit $s(4)$, donc le triple facteur est soit $\mathbf{w}[0,1)$, soit $\mathbf{w}[3,4)$, soit $\mathbf{w}[5,6)$. D'où :

$$g_\varphi(X_\varphi) = \{g_\varphi(0,0,0,1), g_\varphi(0,0,1,1), g_\varphi(0,1,1,1), \\ g_\varphi(3,3,3,4), g_\varphi(3,3,4,4), g_\varphi(3,4,4,4), \\ g_\varphi(5,5,5,6), g_\varphi(5,5,6,6), g_\varphi(5,6,6,6)\}.$$

Un chemin dans $\mathcal{T}_\varphi^4$ de $(0,0,0,0)$ à $\mathbf{q}$, où $\mathbf{q}$ délimite un triple facteur non vide, doit passer par X_φ. On obtient le théorème suivant, analogue au Théorème 3.2.3. La preuve est identique.

Théorème 4.3.2. *Étant donné un cube additif délimité par* $\mathbf{q}$*, il y a un chemin dans* $\mathcal{G}_\varphi$ *de* $g_\varphi(\mathbf{p})$ *à* $g_\varphi(\mathbf{q})$ *pour un certain* $\mathbf{p} \in X_\varphi$ *avec* $g_\varphi(\mathbf{q})$ *dans l'ensemble* $Z_\varphi = \Sigma_\varphi^4 \times \mathfrak{L}_\varphi \times \mathfrak{L}_\varphi$*. Réciproquement, si on a un chemin dans* $\mathcal{G}_\varphi$ *démarrant d'un élément de* $g_\varphi(X_\varphi)$ *et se terminant à* $v \in Z_\varphi$ *alors il y a un cube additif délimité par* $\mathbf{q}$ *où* $v = g_\varphi(\mathbf{q})$*.*

4.4 Réduire à un sous-graphe fini

Comme nous avions admis au Chapitre 3 la Proposition 3.2.1, nous admettons pour l'instant le résultat suivant qui sera démontré dans la Section 4.5.

Proposition 4.4.1. *Soient* b *et* c *deux facteurs dans* $\mathbf{w}$ *de même taille et de même somme, soient* $(b_i)_{i\in\mathbb{N}}$ *et* $(c_i)_{i\in\mathbb{N}}$ *leurs suites ancestrales respectives. Alors, pour tout entier* i *on a* $\|\psi(b_i) - \psi(c_i)\|_2 \leq C_\varphi$*, où* C_φ *est une constante fixée.*

Nous notons

$$\mathfrak{U}_\varphi := \{\mathbf{x} \in \mathbb{Z}^4 \text{ tels que } \|\mathbf{x}\|_2 \leq C_\varphi\}$$

et

$$H_\varphi := \{(\mathbf{c}, \mathbf{u}, \mathbf{v}) : \mathbf{u}, \mathbf{v}, \mathbf{u}+\mathbf{v} \in \mathfrak{U}_\varphi \text{ et } \mathbf{c} \in \Sigma_\varphi^4\}.$$

Par définition, $g_\varphi(\mathbf{q}) \in H_\varphi$ et on restreint notre recherche au sous-graphe de $\mathcal{G}_\varphi$ engendré par H_φ que nous noterons $\mathcal{G}'_\varphi$. L'ensemble H_φ est un sous-ensemble de $\Sigma_\varphi^4 \times \mathfrak{U}_\varphi \times \mathfrak{U}_\varphi$ il contient donc au plus $4^4 \times |\mathfrak{U}_\varphi| \times |\mathfrak{U}_\varphi|$ éléments. $\mathcal{G}'_\varphi$ est donc un graphe fini et on déduit du Théorème 4.3.2 le corollaire suivant :

Corollaire 4.4.2. *Étant donné un cube additif dans* $\mathbf{w}$ *délimité par* $\mathbf{q}$*, il y a un chemin dans* $\mathcal{G}'_\varphi$ *de* $g_\varphi(\mathbf{p})$ *à* $g_\varphi(\mathbf{q})$ *où* $g_\varphi(\mathbf{p}) \in A_\varphi := g_\varphi(X_\varphi) \cap H_\varphi$ *et* $g_\varphi(\mathbf{q}) \in B_\varphi := Z_\varphi \cap H_\varphi$*. Réciproquement, si on a un chemin dans* $\mathcal{G}_\varphi$ *commençant dans* A_φ *et se terminant en* $\beta \in B_\varphi$ *alors il y a un cube additif délimité par* $\mathbf{q}$ *tel que* $g_\varphi(\mathbf{q}) = \beta$*.*

Comme $\mathcal{G}'_\varphi$ est de taille finie on détermine en utilisant un ordinateur s'il existe ou non un chemin dans $\mathcal{G}'_\varphi$ qui va de A_φ à B_φ et donc conclure sur la présence ou non de cubes additifs dans $\mathbf{w}$.

4.5 Généraliser le calcul des bornes

Nous avons montré que pour tout morphisme φ substitution de φ_0 on peut ramener la recherche de cubes additifs dans $\mathbf{w}$ au parcours d'un graphe fini à condition de pouvoir vérifier la Proposition 4.4.1. C'est cette proposition que nous allons maintenant démontrer. La matrice d'incidence de φ vérifie

$$\mathtt{Mat}(\varphi_0) = S^{-1} M_\varphi S,$$

où S désigne la matrice de la substitution s faisant le lien entre φ et φ_0. Puisque M_φ est semblable à $\mathtt{Mat}(\varphi_0)$, les deux matrices possèdent les mêmes valeurs propres qui sont par ordre de module décroissant :

$$\begin{aligned}
\lambda_1 &\doteq 1,690284494616614\\
\lambda_2 &\doteq -1,505068413621473\\
\lambda_3 &\doteq 0,4073919595024293 + 0,4765653259296430i\\
\lambda_4 &\doteq 0,4073919595024293 - 0,4765653259296430i.
\end{aligned}$$

Notons Λ la matrice diagonale composée des valeurs propres de M_φ, on a

$$\Lambda \doteq \begin{pmatrix} \lambda_1 & 0 & 0 & 0\\ 0 & \lambda_2 & 0 & 0\\ 0 & 0 & \lambda_3 & 0\\ 0 & 0 & 0 & \lambda_4 \end{pmatrix}.$$

Soit Q_φ la matrice des vecteurs propres de M_φ, normalisés de façon à ce qu'ils aient une norme de 1. Notons Q_0 la matrice des vecteurs propres de $\mathtt{Mat}(\varphi_0)$ normalisés de la même façon. Alors $M_\varphi = Q_\varphi \Lambda Q_\varphi^{-1}$ et $\mathtt{Mat}(\varphi_0) = Q_0 \Lambda Q_0^{-1}$. D'où

$$\begin{aligned}
\mathtt{Mat}(\varphi_0) &= S^{-1} M_\varphi S\\
&= S^{-1} Q_\varphi \Lambda Q_\varphi^{-1} S\\
&= (S^{-1} Q_\varphi) \Lambda (S^{-1} Q_\varphi)^{-1}.
\end{aligned}$$

Ceci implique que $Q_\varphi = S Q_0$. Posons $\tau_\varphi : \mathbb{C}^4 \to \mathbb{C}^4$ l'endomorphisme correspondant à la multiplication à gauche par Q_φ^{-1}. On notera encore les applications propres $\tau_{j,\varphi} : \mathbb{C}^4 \to \mathbb{C}$, pour $j = 1, 2, 3, 4$, telles que pour tout $x \in \mathbb{C}^4$ on ait $\tau_\varphi(x) = (\tau_{1,\varphi}(x), \tau_{2,\varphi}(x), \tau_{3,\varphi}(x), \tau_{4,\varphi}(x))$.

Théorème 4.5.1 ([16]). *Si b et c sont deux facteurs, non nécessairement consécutifs, alors :*

$$|\tau_{3,\varphi}(\psi(b) - \psi(c))| \leq C_3$$
$$|\tau_{4,\varphi}(\psi(b) - \psi(c))| \leq C_3,$$

où $C_3 \doteq 2.1758$ est une constante.

Démonstration. Le début de cette preuve est identique à celui de la preuve du Théorème 3.3.1, nous ne le retranscrivons pas ici. Il est nécessaire d'adapter la preuve cependant pour obtenir les équations 3.5 à 3.9.

Pour l'équation 3.5, posons, pour $\zeta \in \mathcal{E}$,

$$\Delta(\zeta) = \left\{\psi(a_i') - \psi(a_i) \text{ où } a_i \text{ et } a_i' \text{ sont les préfixes propres des images de } \zeta\right\}.$$

On obtient

$$|\tau_{3,\varphi}(\delta_i)| \leq \max_{\zeta\in\mathcal{E}} \left(\max_{\delta\in\Delta(\zeta)} |\tau_{3,\zeta}(\delta)| \right) \doteq 0.81582,$$

et on a donc encore

$$\left| \sum_{i=9}^{\infty} \lambda_3^i \tau_{3,\varphi}(\delta_i) \right| \leq \sum_{i=9}^{\infty} |\lambda_3|^i |\tau_{3,\varphi}(\delta_i)| \leq C \sum_{i=9}^{\infty} |\lambda_3|^i = \frac{C|\lambda_3|^9}{1-|\lambda_3|} \doteq 0.032736.$$

Pour l'équation 3.7, on note

$$\begin{aligned} \mathfrak{D}_l(\varphi) &= \left\{ \sum_{i=0}^{l-1} M_\varphi^i \psi(a_i) \quad : \quad a_{l-1}\cdots a_0 \text{ les arêtes d'un chemin dans } \mathcal{T}_\varphi \right\} \\ &= \left\{ \sum_{i=0}^{l-1} M_\varphi^i \psi(a_i) \quad : \quad a_{l-1}\cdots a_0 \text{ les arêtes d'un chemin dans } \mathcal{Q}_\varphi \right\}. \end{aligned}$$

On obtient ainsi

$$\max_{\zeta\in\mathcal{E}} \left\{ \max \left\{ |\tau_{3,\zeta}(u) - \tau_{3,\zeta}(v)| : \quad u, v \in \mathfrak{D}_9(\zeta) \right\} \right\} \doteq 1.05517.$$

Cette valeur est obtenue après une recherche via un programme informatique dans les éléments des ensembles $\mathfrak{D}_9(\zeta)$, qui contiennent tous 301 éléments car les graphes ne changent que par leurs étiquettes et pas par leur nombres de chemins de taille fixée. En combinant les deux bornes ainsi obtenues on trouve, comme dans l'équation 3.8 :

$$|\tau_{3,\varphi}(\psi(b))| \leq 1.05517\cdots + 0.032736\cdots = 1.0879\cdots = C_3/2$$

et $|\tau_{3,\varphi}(\psi(c))| \leq C_3/2$, donc

$$|\tau_{3,\varphi}(\psi(b) - \psi(c))| \leq |\tau_{3,\varphi}(\psi(b))| + |\tau_{3,\varphi}(\psi(c))| \leq C_3 \doteq 2.1758.$$

□

Théorème 4.5.2. *Soient b et c deux facteurs de même taille et de même somme dans $\mathbf{w}$, soient $\{b_i\}_{i=0}^{\infty}$ et $\{c_i\}_{i=0}^{\infty}$ les suites ancestrales correspondantes, alors*

$$|\tau_{1,\varphi}(\psi(b_i) - \psi(c_i))| \leq C_{1,\varphi} \tag{4.1}$$
$$|\tau_{2,\varphi}(\psi(b_i) - \psi(c_i))| \leq C_{2,\varphi} \tag{4.2}$$

pour chaque i, avec $C_{1,\varphi}$ et $C_{2,\varphi}$ deux constantes explicites.

Démonstration. On supposera dans toute la suite que $a = 0$ comme expliqué dans l'introduction de ce chapitre. On a définit le réseau euclidien

$$\begin{aligned}\mathfrak{L}_\varphi &:= \{\mathbf{v} \in \mathbb{Z}^4 : (1,1,1,1).\mathbf{v} = 0 \text{ et } (a,b,c,d).\mathbf{v} = 0\} \\ &= \left\{ m\left(\frac{c-b}{b}, \frac{-c}{b}, 1, 0\right) + n\left(\frac{d-b}{b}, \frac{-d}{b}, 0, 1\right) \text{ avec } mc + nd \in b\mathbb{Z} \right\}.\end{aligned}$$

Par construction $\psi(b_0) - \psi(c_0)$ appartient à $\mathfrak{L}_\varphi$ donc on peut l'écrire sous cette forme et utiliser le Théorème 4.5.1 pour borner $|\tau_{3,\varphi}(\psi(b_0) - \psi(c_0))|$.

Lemme 4.5.3. *Soit $\mathbf{v} \in \mathfrak{L}_\varphi$, alors*

$$|\tau_{3,\varphi}(\mathbf{v})| \geq \alpha|m| \tag{4.3}$$
$$|\tau_{3,\varphi}(\mathbf{v})| \geq \beta|n| \tag{4.4}$$

où

$$\alpha = \left|\tau_{3,\varphi}\left(\frac{d-b}{b}, \frac{-d}{b}, 0, 1\right)\right| \left|\mathrm{Im}\left(\frac{\tau_{3,\varphi}\left(\frac{c-b}{b}, \frac{-c}{b}, 1, 0\right)}{\tau_{3,\varphi}\left(\frac{d-b}{b}, \frac{-d}{b}, 0, 1\right)}\right)\right|$$

et

$$\beta = \left|\tau_{3,\varphi}\left(\frac{d-b}{b}, \frac{-d}{b}, 1, 0\right)\right| \left|\mathrm{Im}\left(\frac{\tau_{3,\varphi}\left(\frac{d-b}{b}, \frac{-d}{b}, 0, 1\right)}{\tau_{3,\varphi}\left(\frac{c-b}{b}, \frac{-c}{b}, 1, 0\right)}\right)\right|$$

sont des constantes.

Démonstration. On remarque que pour tout $\zeta \in \mathcal{E}$ et pour tout $b, c, d \in \mathbb{Z}$:

$$\tau_{3,\zeta}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right) = 0 \quad \Leftrightarrow \quad \tau_{3,\zeta}\left(c-b, -c, b, 0\right) = 0 \quad \Leftrightarrow \quad c = b = 0$$

Et de façon similaire :

$$\tau_{3,\zeta}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right) = 0 \quad \Leftrightarrow \quad b = d = 0$$

Or $0 < b < c < d$ donc on a toujours $\tau_{3,\zeta}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right) \neq 0$ et $\tau_{3,\zeta}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right) \neq 0$. Si $m = 0$ l'équation est trivialement vraie. Supposons donc que $m \neq 0$. On remarque que

$\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right) \doteq 0.72928 + 1.20278i \neq 0$. D'où

$$\begin{aligned}
|\tau_{3,\varphi}(\mathbf{v})| &= \left| m\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right) + n\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)\right| \\
&= |m| \left|\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)\right| \left|\frac{n}{m} + \frac{\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)}{\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)}\right| \\
&\geq |m| \left|\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)\right| \left|\mathrm{Im}\left(\frac{n}{m} + \frac{\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)}{\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)}\right)\right| \\
&\geq |m| \left|\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)\right| \left|\mathrm{Im}\left(\frac{\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)}{\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)}\right)\right| \\
&\geq \alpha|m|
\end{aligned}$$

où

$$\alpha = \left|\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)\right| \left|\mathrm{Im}\left(\frac{\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)}{\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)}\right)\right|.$$

De la même façon, $n \neq 0$ et $\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right) \neq 0$.

$$\begin{aligned}
|\tau_{3,\varphi}(\mathbf{v})| &= \left| m\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right) + n\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)\right| \\
&= |n| \left|\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)\right| \left|\frac{m}{n} + \frac{\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)}{\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)}\right| \\
&\geq |n| \left|\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)\right| \left|\mathrm{Im}\left(\frac{m}{n} + \frac{\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)}{\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)}\right)\right| \\
&\geq |n| \left|\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)\right| \left|\mathrm{Im}\left(\frac{\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)}{\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)}\right)\right| \\
&\geq \beta|n|
\end{aligned}$$

où

$$\beta = \left|\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)\right| \left|\mathrm{Im}\left(\frac{\tau_{3,\varphi}\left(\frac{d-b}{b}, -\frac{d}{b}, 0, 1\right)}{\tau_{3,\varphi}\left(\frac{c-b}{b}, -\frac{c}{b}, 1, 0\right)}\right)\right|.$$

□

Nous avons montré que $|\tau_{3,\varphi}(\psi(b_0) - \psi(c_0))| \leq 2.1758$, ce qui implique

$$\alpha|m| \leq |\tau_{3,\varphi}(\psi(b_0) - \psi(c_0))| \leq 2.1758,$$

c'est-à-dire

$$|m| \leq \frac{2.1758}{\alpha},$$

et de la même façon

$$|n| \leq \frac{2.1758}{\beta}.$$

Ceci laisse seulement un nombre fini de choix pour m et n et donc un ensemble fini de vecteurs possibles pour $\psi(b_0) - \psi(c_0)$ que nous notons Ω_φ. Contrairement à la démonstration du Théorème 3.3.3, nous ne pouvons donner explicitement ces vecteurs car les bornes pour choisir m et n dépendent intrinsèquement de l'alphabet Σ_φ. Posons

$$C'_{2,\varphi} = \max_{\zeta \in \mathcal{E}} \left(\max_{\delta \in \Delta(\zeta)} |\tau_{2,\zeta}(\delta)| \right) \doteq 0.65687.$$

Pour $i = 0$ nous avons montré qu'il n'y avait qu'un nombre fini de possibilités (le cardinal de Ω_φ) pour $\psi(b_0) - \psi(c_0)$ pour chaque φ. Notons

$$C_{2,\varphi} = \max \left\{ \max_{\zeta \in \mathcal{E}} \left(\max_{v \in \Omega_\zeta} (|\tau_{2,\zeta}(v)|) \right) ; \frac{2C'_{2,\varphi}}{|\lambda_2| - 1} \right\}.$$

En particulier

$$\begin{aligned} C_{2,\varphi} &\geq \frac{2C'_{2,\varphi}}{|\lambda_2| - 1} \\ &\geq \frac{1}{|\lambda_2|}(C_{2,\varphi} + 2C'_{2,\varphi}). \end{aligned}$$

Cette dernière majoration nous permet d'écrire

$$\begin{aligned} |\tau_{2,\varphi}(\psi(b_{i+1}) - \psi(c_{i+1}))| &= \left|\tau_{2,\varphi}\left(M_\varphi^{-1}(\psi(b_i) - \psi(c_i) - \delta_i + \delta'_i)\right)\right| \\ &= |\lambda_2|^{-1} \left|\tau_{2,\varphi}(\psi(b_i) - \psi(c_i) - \delta_i + \delta'_i)\right| \\ &\leq |\lambda_2|^{-1} \left(|\tau_{2,\varphi}(\psi(b_i) - \psi(c_i)| + |\tau_{2,\varphi}(\delta_i)| + |\tau_{2,\varphi}(\delta'_i)|\right). \end{aligned}$$

Par définition $|\tau_{2,\varphi}(\delta_i)| < C'_{2,\varphi}$ et d'après l'hypothèse de récurrence

$$|\tau_{2,\varphi}(\psi(b_i) - \psi(c_i))| \leq C_{2,\varphi},$$

ce qui donne

$$|\tau_{2,\varphi}(\psi(b_{i+1}) - \psi(c_{i+1}))| \leq |\lambda_2|^{-1} \left(C_{2,\varphi} + 2C'_{2,\varphi}\right) \leq C_{2,\varphi}.$$

Ceci termine la preuve par récurrence de l'inégalité 4.2 et par un raisonnement similaire on obtient l'inégalité 4.1.

□

Soient b et c deux facteurs dans $\mathbf{w}$ de même taille et de même somme, soient $(b_i)_{i\in\mathbb{N}}$ et $(c_i)_{i\in\mathbb{N}}$ leurs suites ancestrales respectives. Alors, pour tout entier i on a $|\tau_{j,\varphi}(\psi(b_i) - \psi(c_i))|_2 \leq C_{j,\varphi}$ pour tout $j \in \{1, 2, 3, 4\}$ Les outils d'algèbre linéaire cités page 46 nous permettent d'affirmer que, comme $\tau_{j,\varphi}(\psi(b_i) - \psi(c_i))$ est borné pour tout $j \in \{1, 2, 3, 4\}$, alors $\tau_\varphi(\psi(b_i) - \psi(c_i))$ est borné également et, comme τ_φ est inversible, $\psi(b_i) - \psi(c_i)$ est borné.

Proposition 4.5.4. *Étant donné l'ensemble*

$$\mathfrak{U}_{\varphi,0} = \{\mathbf{u} \in \mathbb{Z}^4 : |\tau_{i,\varphi}(\mathbf{u})| \leq C_{i,\varphi} \text{ pour } 1 \leq i \leq 4\},$$

si $\mathbf{u} \in \mathfrak{U}_{\varphi,0}$ *alors* $\|\mathbf{u}\| \leq C_\varphi := \left(\frac{C_{1,\varphi}^2+C_{2,\varphi}^2+C_3^2+C_{4,\varphi}^2}{\min_{i\in\{1,2,3,4\}} \lambda_i'}\right)^{1/2} C_\varphi$. *Autrement dit,* $\mathfrak{U}_{\varphi,0} \subset \mathfrak{U}_\varphi$.

En particulier, on en déduit la Proposition 4.4.1.

4.6 Des mots sans puissances additives sur d'autres alphabets

Nous avons limité, dans les sections précédentes, l'extension de ce schéma de preuve aux morphismes similaires à φ_0 et de taille 2 sur des alphabets de 4 lettres. Cependant, en utilisant des méthodes tout à fait similaires, on montre que le morphisme $\varphi_{10} : \{0,1,2\}^* \to \{0,1,2\}^*$ défini par

$$\varphi_{10}(0) = 22 \quad ; \quad \varphi_{10}(1) = 12 \quad ; \quad \varphi_{10}(2) = 10,$$

possède un point fixe infini

$$\mathbf{w_{10}} = \overrightarrow{\varphi_{10}^\omega}(1) = \lim_{n\to\infty} \varphi_{10}^n(1) = 12101222121010101210122212221222\cdots$$

qui évite les 4-puissances additives. Notons que F.M. Dekking [20] a déjà montré que le morphisme $\varphi_3 : \{0,1\}^* \to \{0,1\}^*$ défini par

$$\varphi_3(0) = 011 \quad ; \quad \varphi_3(1) = 0001$$

possède également un point fixe infini qui évite les 4-puissances additives. Il montre en fait qu'il évite les 4-puissances abéliennes mais sur un alphabet de deux lettres les puissances additives sont toutes des puissances abéliennes. L'alphabet est plus petit que celui de φ_{10} mais le morphisme est plus long et non uniforme. J.D. Currie et A. Aberkane [18] ont montré en 2009 qu'il existe un morphisme uniforme de taille 25 qui évite les 4-puissances additives sur $\{0,1\}$.

La matrice d'incidence M_{10} de φ_{10} est

$$M_{10} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 2 & 1 & 0 \end{pmatrix}.$$

Les valeurs propres de M_{10} sont

$$\lambda_1 = 2 \quad , \quad \lambda_2 = -\frac{1+\sqrt{5}}{2} = -1.618\cdots \quad \text{et} \quad \lambda_3 = -\frac{1-\sqrt{5}}{2} = 0.618\cdots.$$

Notons Q la matrice des vecteurs propres de M, normalisés grâce à la norme euclidienne. Soit Λ la matrice diagonale des valeurs propres de M, i.e., $\Lambda = Q^{-1}MQ$. On obtient, par approximation numérique :

$$Q_{10}^{-1} \doteq \begin{pmatrix} 0.599999982 & 0.599999982 & 0.599999982 \\ 0.800000000 & 0.247213588 & -0.647213588 \\ 0.800000000 & -0.647213588 & 0.247213588 \end{pmatrix}.$$

Notons $\tau^{10} : \mathbb{R}^3 \to \mathbb{R}^3$ la multiplication à gauche par Q_{10}^{-1}, et pour $j \in \{1,2,3\}$, posons encore $\tau_j^{10} : \mathbb{R}^3 \to \mathbb{R}$ les applications linéaires telles que $\tau^{10}(x) = (\tau_1^{10}(x), \tau_2^{10}(x), \tau_3^{10}(x))$ pour tout $x \in \mathbb{R}^3$. De la même façon que pour les morphismes de $\mathcal{E}$ on construit l'arbre $\mathcal{T}_{10}$ représentant les liens entre les positions et leurs parents et le graphe orienté $\mathcal{Q}_{10}$ obtenu par projection que nous représentons en Figure 4.3 et en Figure 4.4.

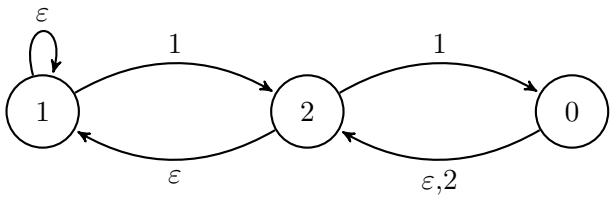


Figure 4.3 – Le graphe orienté $\mathcal{Q}_{10}$.

L'application propre τ_3^{10} est associée à la valeur propre λ_3 qui est de module strictement plus petit que 1. On peut donc utiliser les mêmes arguments que dans le Théorème 3.3.1 et on obtient le résultat suivant.

Théorème 4.6.1. *Si b et c sont deux facteurs de $\mathbf{w_{10}}$ (pas nécessairement consécutifs) alors*

$$|\tau_3^{10}(\psi(b) - \psi(c))| \leq C_3^{10} \quad \text{avec} \quad C_3^{10} < 2.67803.$$

De plus, en réalisant une démonstration par récurrence comme pour le Théorème 3.3.3 on parvient à borner également la première et la deuxième application propre.

Théorème 4.6.2. *Soient b_0 et c_0 deux facteurs de $\mathbf{w_{10}}$ de même taille et même somme, soient $\{b_i\}_{i=0}^{\infty}$ et $\{c_i\}_{i=0}^{\infty}$ les suites ancestrales qui leurs correspondent. Alors*

$$|\tau_1^{10}(\psi(b_i) - \psi(c_i))| \leq C_1^{10} \quad \text{et} \quad |\tau_2^{10}(\psi(b_i) - \psi(c_i))| \leq C_2^{10}$$

pour tout i, avec $C_1^{10} < 1.20001$ et $C_2^{10} < 2.89443$.

Les bornes ainsi obtenues nous permettent d'établir la proposition suivante en procédant comme pour la Proposition 3.3.5.

Proposition 4.6.3. *Si $\mathbf{u} \in \{\mathbf{u} \in \mathbb{Z}^3 : |\tau_i^{10}(\mathbf{u})| \leq C_i^{10}$ pour $1 \leq i \leq 3\}$ alors*

$$\|\mathbf{u}\| \leq \left(\frac{(C_1^{10})^2 + (C_2^{10})^2 + (C_3^{10})^2}{\min_{i\in\{1,2,3\}} \lambda_i'} \right)^{1/2} < \sqrt{21.238},$$

où λ_1', λ_2', et λ_3' désignent les valeurs propres de $(\tau^{10})^* \tau^{10}$.

Ensuite, on raisonne sur les chemins dans $\mathcal{T}_{10}^5 = \mathcal{T}_{10} \times \mathcal{T}_{10} \times \mathcal{T}_{10} \times \mathcal{T}_{10} \times \mathcal{T}_{10}$ pour retracer l'origine des 4-puissances additives que l'on identifie grâce à des quintuplets de positions. Là encore on construit une bijection entre les chemins partant d'un point précis dans $\mathcal{T}_{10}^5$ et ceux partant de l'image de ce point dans un sous graphe fini que l'on peut parcourir via un programme informatique.

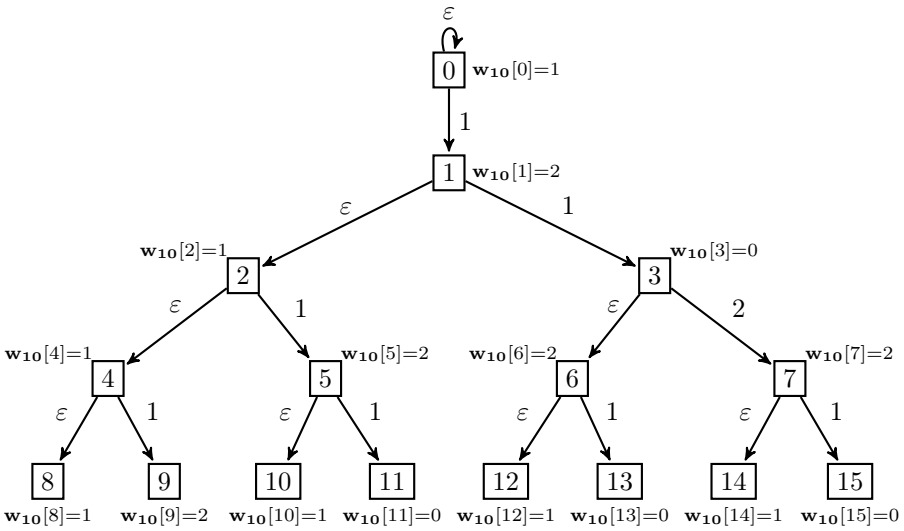


Figure 4.4 – Les 5 premiers niveaux de $\mathcal{T}_{10}$.

En conclusion, il est possible d'adapter ce schéma de preuve à bien d'autres morphismes différents de φ_0. Une généralisation encore plus importante de cet algorithme peut être trouvée dans l'article de M. Rao et M. Rosenfeld [39]. Ils démontrent qu'on peut, avec des conditions beaucoup moins restrictives que la similarité à φ_0, construire un algorithme qui décide si le point fixe infini d'un morphisme évite ou non les puissances additives ou abéliennes.

Enfin, la programmation informatique du parcours de graphe doit être effectuée avec quelques précautions. C'est ce dernier qui permet, à la fin de notre algorithme, de conclure sur la présence ou non de puissances additives dans les points fixes infinis. Dans le cas des morphismes de l'ensemble $\mathcal{E}$ le graphe fini à explorer contient plusieurs dizaines de millions de sommet. Il est donc nécessaire pour obtenir une réponse rapide et pour pouvoir tester de nombreux morphismes de définir informatiquement une structure de graphe légère avec un parcours efficace. Nous avons programmé ce parcours en `C++` grâce aux *structures* en les créant de la façon la plus adaptée possible, notamment en utilisant des pointeurs sur les sommets et en ne créant que les sommets par lesquels passe le parcours pour ne pas surcharger la mémoire. Les calculs sur les matrices ont été réalisés grâce à la librairie `Eigen`. Pour un morphisme donné similaire à φ_0 notre programme décide en $0,5$ secondes s'il vérifie ou non la condition Cupisca (sur des ordinateurs de bureau standard, voir Remarque 1.5.1).

Chapitre 5

Éviter les cubes additifs sur les alphabets d'au moins 4 lettres

Ce travail est issu d'une collaboration avec Matthieu Rosenfeld, chercheur à l'université d'Aix-Marseille. Il a donné lieu à la publication d'un article [33] suite à la conférence *Developments in Language Theory 2020*.

5.1 Préliminaires

Nous rappelons que l'équivalence entre deux alphabets est définie page 10. Montrons d'abord que pour tout alphabet dans $\mathbb{C}$, soit nous savons déjà que les cubes additifs sont évitables sur cet alphabet, soit il est équivalent à un alphabet dans $\mathbb{N}$. Donnons d'abord une condition suffisante pour que deux alphabets soient équivalents :

Lemme 5.1.1. *Soient $a \in \mathbb{C}_{\neq 0}$, $b \in \mathbb{C}$ et $f : \mathbb{C} \to \mathbb{C}$ la fonction telle que pour tout x, $f(x) = ax + b$, alors pour tous mots u et v :*

$$u \simeq_{ad} v \iff f(u) \simeq_{ad} f(v)\,.$$

Démonstration. Par définition, $|w| = |f(w)|$ pour tout mot w, donc

$$|u| = |v| \iff |f(u)| = |f(v)| \text{ pour tous } u, v \in \mathcal{A}^*.$$

De plus, si $|u| = |v|$, alors

$$\begin{aligned}
&\sum u = \sum v, \\
\iff & a\sum u + |u|b = a\sum v + |u|b && (\text{car } a \neq 0) \\
\iff & a\sum u + |f(u)|b = a\sum v + |f(v)|b && (\text{car } |f(u)| = |u| = |v| = |f(v)|) \\
\iff & \sum f(u) = \sum f(v).
\end{aligned}$$

Ceci entraîne

$$\left(|u| = |v| \text{ et } \sum u = \sum v\right) \iff \left(|f(u)| = |f(v)| \text{ et } \sum f(u) = \sum f(v)\right).$$

□

Rappelons que deux nombres complexes a et b sont rationnellement indépendants si $k_1 a + k_2 b = 0$ pour $(k_1, k_2) \in \mathbb{Z}$ implique $k_1 = k_2 = 0$.

Lemme 5.1.2. *Soit $\mathcal{A} \subseteq \mathbb{C}$ un alphabet, alors une des propositions suivante est vérifiée :*

(i) Si $|\mathcal{A}| \leq 2$, alors les cubes additifs ne sont pas évitables sur $\mathcal{A}$.

(ii) Si $|\mathcal{A}| > 2$ et il existe $a, b, c \in \mathcal{A}$, tels que $b - a$ et $c - a$ sont rationnellement indépendants, alors les cubes additifs sont évitables sur $\mathcal{A}$,

(iii) Si $|\mathcal{A}| > 2$ et pour tous $a, b, c \in \mathcal{A}$ deux à deux distincts, $b - a$ et $c - a$ sont rationnellement dépendants, alors il existe un alphabet $\mathcal{A}' = \{0, a_1, \ldots, a_{m-1}\} \in \mathbb{N}$ où les a_i sont co-premiers, tel que $\mathcal{A}$ et $\mathcal{A}'$ sont équivalents.

Démonstration. Démontrons les affirmations dans l'ordre.

(i) Cette assertion vient du fait que les cubes abéliens ne sont pas évitables sur un alphabet de taille au plus 2 (voir par exemple [20]).

(ii) Comme $b - a$ et $c - a$ sont rationnellement indépendants, pour tous $k_1, k_2, k_3 \subseteq \mathbb{Z}$, si $0k_1 + (b-a)k_2 + (c-a)k_3 = 0$ alors $k_2 = k_3 = 0$. Donc étant donnés deux mots $u, v \in \{0, b-a, c-a\}^*$, si $\sum u = \sum v$ alors u et v ont le même nombre d'occurrences de $b - a$ (resp. $c - a$). Si en outre $|u| = |v|$ alors ils ont aussi le même nombre d'occurrences de 0. Donc pour tous mots $u, v \in \{0, b-a, c-a\}^*$, si $u \simeq_{ad} v$ alors $u \simeq_{ab} v$. Grâce au Lemme 5.1.1 (en prenant $f : x \mapsto x + a$), si $u, v \in \{a, b, c\}^*$ sont tels que $u \simeq_{ad} v$ alors $u \simeq_{\text{ab}} v$. Puisque les cubes abéliens sont évitables sur 3 lettres, alors les cubes additifs sont évitables sur $\mathcal{A}$.

(iii) Soit $\mathcal{A} = \{b_1, \ldots, b_m\}$ avec $b_1 < b_2 \cdots < b_m$. Pour tout i, $b_i - b_1$ et $b_2 - b_1$ sont rationnellement dépendants, ce qui implique $\frac{b_i - b_1}{b_2 - b_1} \in \mathbb{Q}$. Il existe un entier $q \in \mathbb{Z}$ tel que pour tout i, $q\frac{b_i - b_1}{b_2 - b_1} \in \mathbb{Z}$ et $\gcd\left(q\frac{b_2 - b_1}{b_2 - b_1}, q\frac{b_3 - b_1}{b_2 - b_1}, \ldots, q\frac{b_m - b_1}{b_2 - b_1}\right) = 1$. Soit $s = \min_{1 \leq i \leq m}\left(q\frac{b_i - b_1}{b_2 - b_1}\right)$. On peut appliquer le Lemme 5.1.1 avec $f : x \mapsto q\frac{x - b_1}{b_2 - b_1} - s$ et on obtient l'alphabet $\{q\frac{b_1 - b_1}{b_2 - b_1} - s, q\frac{b_2 - b_1}{b_2 - b_1} - s, q\frac{b_3 - b_1}{b_2 - b_1} - s, \ldots, q\frac{b_m - b_1}{b_2 - b_1} - s\}$ qui vérifie toutes les conditions demandées. □

En conclusion, répondre à la question de l'évitabilité des cubes additifs sur les alphabets de la forme $\{0, a_1, \ldots, a_m\} \in \mathbb{N}$ avec les a_i co-premiers permet de répondre à la question pour tous les alphabets finis sur $\mathbb{C}$. Remarquons que dans le cas (iii), on peut ajouter la condition $a_1 < a_m - a_{m-1}$ (il suffit d'appliquer $f : x \mapsto a_m - x$ à l'alphabet si elle n'est pas déjà vérifiée). Ajoutons également que dans le cas où $|\mathcal{A}| = 2$, on peut

éviter les 4-puissances additives (avec un argument similaire à celui de (i) et le fait que les 4-puissances abéliennes sont évitables sur 2 lettres [20]).

Remarque 5.1.1. *Chaque alphabet* $\{a_0, a_1, \ldots, a_m\} \subset \mathbb{N}$ *est équivalent à l'alphabet* $\{0, 1, f(a_2), \ldots, f(a_m)\} \subset \mathbb{Q}$, *avec* $f : x \mapsto \frac{x-a_0}{a_1-a_0}$. *Donc, dans les Section 5.2 et Section 5.3, au lieu de considérer des alphabets de quatre entiers on considérera des alphabets de la forme* $\{0, 1, c, d\} \subset \mathbb{Q}$.

5.2 Le mot infini $\mathbf{W}_{a,b,c,d}$

Pour tous réels $a, b, c, d \in \mathbb{R}$, soit $\varphi_{a,b,c,d} : \{a, b, c, d\}^* \to \{a, b, c, d\}^*$ le morphisme

$$\varphi_{a,b,c,d}(a) = ac \quad ; \quad \varphi_{a,b,c,d}(b) = dc \quad ; \quad \varphi_{a,b,c,d}(c) = b \quad ; \quad \varphi_{a,b,c,d}(d) = ab.$$

Soit $\mathbf{W}_{a,b,c,d}$ son point fixe infini. Cassaigne et al. montrent dans [16] que le mot $\mathbf{W}_{0,1,3,4}$ évite les cubes additifs, ce qui signifie en particulier que $\mathbf{W}_{0,1,3,4}$ évite les cubes abéliens. Cette dernière propriété ne dépend pas de l'alphabet choisi, on en déduit donc le lemme suivant.

Lemme 5.2.1. *pour tous entiers* a, b, c, d *deux à deux distincts, le mot* $\mathbf{W}_{a,b,c,d}$ *évite les cubes abéliens.*

Soit $M_\varphi = M(\varphi_{a,b,c,d}) = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$ la matrice d'incidence de $\varphi_{a,b,c,d}$ et τ le vecteur propre de M_φ correspondant à l'approximation numérique

$$\tau \doteq \begin{pmatrix} 0.5788 - 0.5749i \\ -0.3219 + 0.2183i \\ -0.0690 + 0.6165i \\ -0.1662 - 0.6810i \end{pmatrix},$$

qui est liée à la valeur propre $0.4074 + 0.4766i$ de M_φ et dont on peut retrouver une définition précise dans la Section 2.1 de [16] (on pourra aussi se référer à la construction de τ_φ dans la Section 4.5 de ce document). Notons que travailler avec des approximations numériques n'est pas un problème. Toutes nos opérations sont numériquement stables (additions, multiplications mais pas de divisions par des nombres proches de zéro) et si on commence avec une précision suffisamment grande, on termine les calculs avec une précision suffisamment grande. De plus, il existe une extension algébrique de $\mathbb{Q}$ de degré 24 qui contient toutes les valeurs propres de la matrice (d'après Mathematica) et on pourrait donc utiliser la preuve originale de [16, Theorem 8] pour obtenir une valeur exacte de la constante C et effectuer ensuite des calculs exacts dans cette partie. Cependant, conserver des expressions impliquant des radicaux dans les calculs est beaucoup plus inefficace et plus difficile à suivre dans les explications pour le lecteur. Nous faisons appel à un autre théorème de [16].

Théorème 5.2.2 ([16, Theorem 8]). *Il existe une constante C telle que pour tous facteurs u et v de $\mathbf{W}_{a,b,c,d}$ (non nécessairement consécutifs) on ait*

$$|\tau \cdot (\Psi(u) - \Psi(v))| < C$$

avec $2.175816 < C < 2.175817$.

Résumons l'idée derrière le Théorème 5.2.2. Le comportement asymptotique des vecteurs de Parikh des facteurs est fortement lié au comportement asymptotique des itérations de la matrice M_φ (car $\Psi(\varphi(u)) = M_\varphi(\Psi(u))$). De plus, la valeur propre correspondant à ce vecteur propre est de norme plus petite que 1. Donc $\tau(\Psi(u))$ est borné pour tout facteur u. Le Théorème 5.2.2 donne une approximation de la borne dans ce cas particulier. Grâce au Lemme 5.2.1 et au Théorème 5.2.2 on déduit un troisième résultat.

Lemme 5.2.3. *Pour tous $a, b, c, d \in \mathbb{R}$, soit $M_{a,b,c,d} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ a & b & c & d \end{pmatrix}$. Supposons que $\mathbf{W}_{a,b,c,d}$ contient un cube additif, alors il existe un vecteur $x \in \ker(M_{a,b,c,d}) \cap \mathbb{Z}^4 \setminus \{0\}$ tel que $|\tau \cdot x| < C$, où C est la constante fixée dans le Théorème 5.2.2.*

Démonstration. Soit uvw un cube additif facteur de $\mathbf{W}_{a,b,c,d}$. D'après le Lemme 5.2.1, uvw ne peut pas être un cube abélien. Donc soit $\Psi(u) \neq \Psi(v)$ ou $\Psi(v) \neq \Psi(w)$. Sans perte de généralité, supposons que $\Psi(u) \neq \Psi(v)$. Dans ce cas, posons $x = \Psi(u) - \Psi(v) \neq 0$, avec $x \in \mathbb{Z}^4$. Comme uvw est un cube additif, $|u| = |v|$ et $|u|_a a + |u|_b b + |u|_c c + |u|_d d = |v|_a a + |v|_b b + |v|_c c + |v|_d d$. Ceci implique que $M_{a,b,c,d}(\Psi(u) - \Psi(v)) = 0$, ce qui peut se réécrire comme $x \in \ker(M_{a,b,c,d})$. On montre donc que $x \in \ker(M_{a,b,c,d}) \cap \mathbb{Z}^4 \setminus \{0\}$. Comme u et v sont, par hypothèse, deux facteurs de $\mathbf{W}_{a,b,c,d}$, le Théorème 5.2.2 entraîne que $|\tau \cdot x| < C$, ce qui termine la preuve. □

Ce lemme contient l'idée principale de l'article : si on veut connaître les quadruplets (a, b, c, d) tels que le mot $\mathbf{W}_{a,b,c,d}$ évite les cubes additifs, on doit seulement étudier le réseau euclidien $\ker(M_{a,b,c,d}) \cap \mathbb{Z}^4 \setminus \{0\}$.

5.3 Le cas du mot infini $\mathbf{W}_{0,1,c,d}$

Nous étudions le réseau euclidien $\ker(M_{0,1,c,d}) \cap \mathbb{Z}^4 \setminus \{0\}$ pour montrer que dans de nombreux cas les cubes additifs sont évitables sur $\{0, 1, c, d\}$.

Théorème 5.3.1. *Soient $c, d \in \mathbb{R}$. Supposons que $d > c > 1$, $c \notin \{5/4, 4/3, 3/2, 2\}$ et $d \notin \{6 - 4c, 5 - 3c, 4 - 2c, 3 - c, 2c - 3, 2c - 2, 2c - 1, 3c - 3, 2\}$. Alors $\mathbf{W}_{0,1,c,d}$ évite les cubes additifs.*

Démonstration. D'après le Lemme 5.2.3, il suffit de montrer que pour tout $x \in \ker(M_{0,1,c,d}) \cap \mathbb{Z}^4 \setminus \{0\}$, on a $|\tau \cdot x| \geq C$. Réécrivons tout d'abord cet ensemble de façon plus adéquate. On peut vérifier directement que si $\alpha := (c - 1, -c, 1, 0)$ et $\beta := (d - 1, -d, 0, 1)$ alors α, β est une base de $\ker(M_{0,1,c,d})$. Pour tous réels m et n, si $m\alpha + n\beta$ est un vecteur

d'entiers, alors $m \in \mathbb{N}$ (resp. $n \in \mathbb{N}$) car autrement la troisième (resp. quatrième) coordonnée ne serait pas un entier, de plus $mc+nd \in \mathbb{Z}$ car sinon la première et la deuxième coordonnée ne seraient pas des entiers. On en déduit que

$$\ker(M_{0,1,c,d}) \cap \mathbb{Z}^4 = \{m\alpha + n\beta | m, n \in \mathbb{Z}, mc + nd \in \mathbb{Z}\}.$$

Donc il suffit de montrer que, sous nos hypothèses, pour tous $m, n \in \mathbb{Z}$ avec $mc+nd \in \mathbb{Z}$ et $(m, n) \neq (0, 0)$, on a

$$|\tau \cdot (m\alpha + n\beta)| \geq C. \tag{5.1}$$

Montrons que (5.1) est vérifiée dans le cas $n = 0$. Dans ce cas, $m \neq 0$ et $|\tau \cdot m\alpha| = |m||\tau \cdot \alpha|$ et $mc \in \mathbb{Z}$. Une étude numérique donne

$$f_0(b) := |\tau \cdot \alpha| \doteq \sqrt{1.83908 + b(-3.05698 + 1.44043b)}.$$

Le minimum de f_0 est atteint en

$$b \doteq \frac{3.05698}{2 \times 1.44043} \doteq 1.06114.$$

Donc pour tous $x, y \in \mathbb{R}$ avec $x < y$ et $1.06114 < y$ le minimum de f_0 sur l'intervalle $[x, y]$ est donné par $f_0(\max(1.06114, x))$. Décomposons ce problème suivant la valeur de c.

- Si $c > 2.85$ un calcul direct donne $|\tau \cdot \alpha| > C$ et $|\tau \cdot m\alpha| > C$.
- Si $c \in [1.9, 2.9] \setminus \{2\}$, un calcul donne $|\tau \cdot \alpha| > \frac{C}{2}$. De plus, dans ce cas les conditions $m \in \mathbb{Z}$ et $mc \in \mathbb{Z}$ impliquent que $|m| \geq 2$ (car $c \notin \mathbb{Z}$) et $|\tau \cdot m\alpha| > C$.
- Si $c \in [1.55, 1.95]$, un calcul donne $|\tau \cdot \alpha| > \frac{C}{3}$. De plus, dans ce cas, les conditions $m \in \mathbb{Z}$ et $mc \in \mathbb{Z}$ impliquent que $|m| \geq 3$ (car $2c \notin \mathbb{Z}$) et on obtient $|\tau \cdot m\alpha| > C$.
- Si $c \in [1.3, 1.65] \setminus \{4/3, 3/2\}$, un calcul donne $|\tau \cdot \alpha| > \frac{C}{4}$. De plus, dans ce cas, les conditions $m \in \mathbb{Z}$ et $mc \in \mathbb{Z}$ impliquent que $|m| \geq 4$ (car $3c, 2c \notin \mathbb{Z}$) et on obtient $|\tau \cdot m\alpha| > C$.
- Si $c \in]1, 1.35] \setminus \{5/4, 4/3\}$, un calcul donne $|\tau \cdot \alpha| > \frac{C}{5}$. De plus, dans ce cas, les conditions $m \in \mathbb{Z}$ et $mc \in \mathbb{Z}$ impliquent que $|m| \geq 5$ (car $4c, 3c, 2c \notin \mathbb{Z}$) et on obtient $|\tau \cdot m\alpha| > C$.

Montrons maintenant que ce résultat est également vrai si $|n| \geq 4$ et $m \in \mathbb{Z}$.

$$\begin{aligned}
|m\tau \cdot \alpha + n\tau \cdot \beta| &= |n||\tau \cdot \alpha| \left| \frac{m}{n} + \frac{\tau \cdot \beta}{\tau \cdot \alpha} \right| \\
&\geq |n||\tau \cdot \alpha| \left| \mathrm{Im}\left(\frac{m}{n} + \frac{\tau \cdot \beta}{\tau \cdot \alpha} \right) \right| \\
&\geq |n||\tau \cdot \alpha| \left| \mathrm{Im}\left(\frac{\tau \cdot \beta}{\tau \cdot \alpha} \right) \right| \\
&\geq k|n|
\end{aligned} \tag{5.2}$$

avec

$$k = |\tau \cdot \alpha| \left| \mathrm{Im}\left(\frac{\tau \cdot \beta}{\tau \cdot \alpha} \right) \right|.$$

L'évaluation numérique de k donne

$$\begin{aligned} k^2 &\doteq \frac{1}{c^2 - 2.12228\,c + 1.27676}\Big(0.217137\,d^2 + 0.533079\,dc + 0.327181c^2 \\ &\qquad + 0.217127\,d - 0.911556c + 0.634921\Big), \\ k^2 - \left(\frac{C}{4}\right)^2 &\doteq \frac{1}{c^2 - 2.12228\,c + 1.27676}\Big(0.257151 + 0.0312991c^2 \\ &\qquad + c(-0.283614 + 0.533079d) + (-0.742604 + 0.217137d)d\Big). \end{aligned}$$

Le dénominateur $c^2 - 2.12228\,c + 1.27676$ est positif pour tout réel c. Donc, le signe de $k^2 - \left(\frac{C}{4}\right)^2$ est celui du numérateur. Pour un d fixé, le minimum du numérateur est atteint en $c \doteq 0.00443843 - 0.00834245d < 0$ (car $d > 1$). Le numérateur est une fonction croissante de c pour $c > 0$ et en particulier, à d fixé avec $1 \leq c < d$ le minimum est atteint en $c = 1$ et vaut $0.00483619 + (-0.209525 + 0.217137d)d$ qui est positif car $d > 1$. On conclut que $k > \frac{C}{4}$ et l'équation (5.2) implique que si $|n| \geq 4$, alors

$$|m\tau \cdot \alpha + n\tau \cdot \beta| > C.$$

Il reste à étudier les cas $|n| \in \{1, 2, 3\}$. Dans (5.1) il suffit d'étudier les cas $n \in \{1, 2, 3\}$. Nous traitons ces trois cas de façon similaire. Commençons par $n = 1$, ce qui donne en approximation numérique

$$\begin{aligned} P_{c,d,1}(m) &:= |\tau \cdot (m\alpha + \beta)|^2 - C^2 \\ &\doteq -4.16782 + 0.712407m - 1.17373cm + 1.83908m^2 - 3.05698cm^2 \\ &\quad + 1.44043c^2m^2 + (-1.17373 - 3.05698m + 2.88085cm)d + 1.44043d^2. \end{aligned}$$

$P_{c,d,1}(m)$ est un trinôme en d. Le calcul du discriminant donne $P_{c,d,1}(m) > 0$ pour tout $c \in \mathbb{R}$ si et seulement si

$$\Delta_c(d) :\doteq 25.3914 + 3.07144m - 1.25108m^2 < 0.$$

Notons que le fait que c n'apparaisse pas n'est pas une coïncidence numérique. On peut le vérifier en utilisant le fait que $P_{c,d,1}(m)$ est de la forme $(x + ym + z(d + cm))^2 + (x' + y'm + z'(d + cm))^2 - C^2$ avec , $y, z \in \mathbb{R}$. C'est une équation du second degré en m et on obtient

$$m \notin [-3.44178, 5.89681] \implies |\tau \cdot (m\alpha + \beta)| > C.$$

Il suffit de vérifier que, sous nos hypothèses, pour tout $m \in \{5, 4, 3, 2, 1, 0, -1, -2, -3\}$ tel que $mc + d \in \mathbb{Z}$, $P_{c,d,1}(m) > 0$. Détaillons les cas $m = -3$ et $m = 4$. Numériquement, on obtient

$$P_{c,d,1}(-3) \doteq 10.2467 + 12.9638c^2 + c(-23.9917 - 8.64256d) + d(7.99723 + 1.44043d),$$

qui est un polynôme du second degré en d. On en déduit

$$P_{c,d,1}(-3) > 0 \iff d \in]-\infty, 3c - 3.54573[\cup]3c - 2.00625, \infty[.$$

Comme $d \neq 3c-3$ alors soit $P_{c,d,1}(-3) > 0$ soit $-3c + d \notin \mathbb{Z}$. La condition $P_{c,d,1}(4) > 0$ est équivalente à $d \in]6.1107 - 4c, \infty[$. Comme $d > c > 1$ et $d \neq 6 - 4c$ alors soit $P_{c,d,1}(4) > 0$ soit $d + 4c \notin \mathbb{Z}$. Les autres cas sont tout à fait similaires. Nous donnons dans le Tableau 5.1, pour chaque cas, la condition sur les réels et l'hypothèse qui nous permet de conclure.

Tableau 5.1 – Théorème 5.3.1 : Étude de $P_{c,d,1}(m)$ pour $m \in \{5, 4, 3, 2, 1, 0, -1, -2, -3\}$.

(A) **: Une condition équivalente pour** d	**Une condition suffisante pour obtenir** (A)
$P_{c,d,1}(5) > 0 \Leftrightarrow d \in]6.78141 - 5c, \infty[$	$d > c > 1$
$P_{c,d,1}(4) > 0 \Leftrightarrow d \in]6.1107 - 4c, \infty[$	$d > c > 1$ et $d \neq 6 - 4c$
$P_{c,d,1}(3) > 0 \Leftrightarrow d \in]5.26804 - 3c, \infty[$	$d > c > 1$ et $d \neq 5 - 3c$
$P_{c,d,1}(2) > 0 \Leftrightarrow d \in]4.31762 - 2c, \infty[$	$d > c > 1$ et $d \neq 4 - 2c$
$P_{c,d,1}(1) > 0 \Leftrightarrow d \in]3.27931 - c, \infty[$	$d > c > 1$ et $d \neq 3 - c$
$P_{c,d,1}(0) > 0 \Leftrightarrow d \notin [-1.34171, 2.15655]$	$d > c > 1$ et $d \neq 2$
$P_{c,d,1}(-1) > 0 \Leftrightarrow d \in]c + 0.939592, \infty[$	$d > c$
$P_{c,d,1}(-2) > 0$ $\Leftrightarrow d \notin [2c - 3.02493, 2c - 0.404774]$	$d \notin \{2c-3, 2c-2, 2c-1\}$
$P_{c,d,1}(-3) > 0$ $\Leftrightarrow d \notin [3c - 3.54573, 3c - 2.00625]$	$d \neq 3c - 3$

Le cas suivant est $n = 2$ et on le traite de façon similaire. Numériquement on obtient

$$\begin{aligned} P_{c,d,2}(m) &:= |\tau \cdot (m\alpha + 2\beta)|^2 - C^2 \\ &\doteq -2.46898 + 1.42481m - 2.34745cm + 1.83908m^2 - 3.05698cm^2 \\ &\quad + 1.44043c^2m^2 + (-4.6949 - 6.11397m + 5.76171cm)d + 5.76171d^2. \end{aligned}$$

ce qui donne, en calculant le discriminant $P_{c,d,2}(m) > 0, \forall d \in \mathbb{R}$ si et seulement si

$$\Delta_c(d) :\doteq 78.9442 + (24.5715 - 5.00433m)m < 0.$$

C'est une équation du second degré en m et la résoudre permet d'obtenir

$$m \notin [-2.21427, 7.12433] \implies |\tau \cdot (m\alpha + \beta)| > C.$$

Nous devons vérifier que, sous nos hypothèses, pour tout $m \in \{7, 6, 5, 4, 3, 2, 1, 0, -1, -2\}$ tel que $mc + 2d \in \mathbb{Z}$, $P_{c,d,2}(m) > 0$. Chaque cas peut être traité de la même façon que dans le cas $n = 1$. On donne, dans le Tableau 5.2, pour chaque cas la condition sur les réels et l'hypothèse qui nous permet de conclure.

Tableau 5.2 – Théorème 5.3.1 : Étude de $P_{c,d,2}(m)$ pour $m \in \{7,6,5,4,3,2,1,0,-1,-2\}$.

(B_1) **: Une condition équivalente pour** d	**Une condition suffisante pour obtenir** (B_1)
$P_{c,d,2}(7) > 0 \Leftrightarrow d \notin]3.91363 - 3.5c, 4.32919 - 3.5c[$	$d > c > 1$
$P_{c,d,2}(6) > 0 \Leftrightarrow d \notin]3.00088 - 3c, 4.1808 - 3c[$	$d > c > 1$
$P_{c,d,2}(5) > 0 \Leftrightarrow d \notin]2.30029 - 2.5c, 3.82024 - 2.5c[$	$d > c > 1$
$P_{c,d,2}(4) > 0 \Leftrightarrow d \notin]1.67431 - 2c, 3.38509 - 2c[$	$d > c > 1$
$P_{c,d,2}(3) > 0 \Leftrightarrow d \notin]1.09888 - 1.5c, 2.89938 - 1.5c[$	$d > c > 1$
$P_{c,d,2}(2) > 0 \Leftrightarrow d \notin]0.566427 - c, 2.3707 - c[$	$d > c > 1$
$P_{c,d,2}(1) > 0 \Leftrightarrow d \notin]0.0766769 - 0.5c, 1.79931 - 0.5c[$	$d > c > 1$
$P_{c,d,2}(0) > 0 \Leftrightarrow d \notin] - 0.363621, 1.17847[$	$d > c > 1$
$P_{c,d,2}(-1) > 0 \Leftrightarrow d \notin [-0.732884 + 0.5c, 0.486592 + 0.5c]$	$d > c$
$P_{c,d,2}(-2) > 0 \Leftrightarrow d \notin [-0.925154 + c, -0.382276 + c]$	$d > c$

On procède de la même façon pour le cas $n = 3$, on obtient

$$\begin{aligned} P_{c,d,3}(m) &:= |\tau \cdot (m\alpha + 2\beta)|^2 - C^2 \\ &\doteq 0.362434 + 2.13722m - 3.52118bm + 1.83908m^2 - 3.05698cm^2 \\ &\quad + 1.44043c^2m^2 + (-10.5635 - 9.17095m + 8.64256cm)d + 12.9638d^2, \end{aligned}$$

et le calcul du discriminant donne $P_{c,d,3}(m) > 0, \forall d \in \mathbb{R}$ si et seulement si

$$\Delta_c(d) \doteq 92.7941 + 82.929m - 11.2597m^2 < 0.$$

C'est une équation du second degré en m et on obtient

$$m \notin [-0.986756, 8.35184] \implies |\tau \cdot (m\alpha + \beta)| > C.$$

On vérifie que, sous nos hypothèses, pour tout $m \in \{8,7,6,5,4,3,2,1,0\}$ tel que $mc+3d \in \mathbb{Z}$, $P_{c,d,3}(m) > 0$. Chaque cas peut être traité de la même façon que dans les cas $n = 1, 2$. On donne, dans le Tableau 5.3, pour chaque cas la condition sur les réels et l'hypothèse qui nous permet de conclure.

Cela termine la preuve du Théorème 5.3.1. □

En fait, en utilisant une symétrie de plus on peut améliorer le résultat précédent et obtenir le théorème suivant.

Théorème 5.3.2. *Pour tout* $(c,d) \in \mathbb{R}^2 \setminus \mathcal{F}$ *les cubes additifs sont évitables sur* $\{0,1,c,d\}$

Tableau 5.3 – Théorème 5.3.1 : Étude de $P_{c,d,3}(m)$ pour $m \in \{8,7,6,5,4,3,2,1,0\}$.

(B_2) **: une condition équivalente pour** d	**Une condition suffisante pour obtenir** (B_2)
$P_{c,d,3}(8) > 0 \Leftrightarrow d \notin]3.00699 - 2.66667c, 3.46726 - 2.66667c[$	$d > c > 1$
$P_{c,d,3}(7) > 0 \Leftrightarrow d \notin]2.45816 - 2.33333c, 3.30867 - 2.33333c[$	$d > c > 1$
$P_{c,d,3}(6) > 0 \Leftrightarrow d \notin]2.00508 - 2c, 3.05432 - 2c[$	$d > c > 1$
$P_{c,d,3}(5) > 0$ $\Leftrightarrow d \notin]1.59624 - 1.66667c, 2.75573 - 1.66667c[$	$d > c > 1$
$P_{c,d,3}(4) > 0$ $\Leftrightarrow d \notin]1.21937 - 1.33333c, 2.42518 - 1.33333c[$	$d > c > 1$
$P_{c,d,3}(3) > 0 \Leftrightarrow d \notin]0.870753 - c, 2.06637 - c[$	$d > c > 1$
$P_{c,d,3}(2) > 0$ $\Leftrightarrow d \notin]0.551146 - 0.666667c, 1.67855 - 0.666667c[$	$d > c > 1$
$P_{c,d,3}(1) > 0$ $\Leftrightarrow d \notin]0.266517 - 0.333333c, 1.25575 - 0.333333c[$	$d > c > 1$
$P_{c,d,3}(0) > 0 \Leftrightarrow d \notin]0.0358908, 0.778955[$	$d > c > 1$

avec

$$\mathcal{F} = \left\{ \left(\frac{10}{9}, \frac{14}{9}\right), \left(\frac{9}{8}, \frac{3}{2}\right), \left(\frac{9}{8}, \frac{13}{8}\right), \left(\frac{8}{7}, \frac{10}{7}\right), \left(\frac{8}{7}, \frac{11}{7}\right), \left(\frac{8}{7}, \frac{12}{7}\right), \left(\frac{7}{6}, \frac{11}{6}\right), \right.$$
$$\left(\frac{7}{6}, \frac{3}{2}\right), \left(\frac{7}{6}, \frac{5}{3}\right), \left(\frac{6}{5}, \frac{8}{5}\right), \left(\frac{6}{5}, \frac{9}{5}\right), \left(\frac{6}{5}, 2\right), \left(\frac{5}{4}, \frac{7}{4}\right), \left(\frac{5}{4}, 2\right), \left(\frac{5}{4}, \frac{9}{4}\right), \left(\frac{5}{4}, \frac{5}{2}\right),$$
$$\left(\frac{5}{4}, \frac{11}{4}\right), \left(\frac{5}{4}, 3\right), \left(\frac{5}{4}, \frac{13}{4}\right), \left(\frac{5}{4}, \frac{7}{2}\right), \left(\frac{4}{3}, 2\right), \left(\frac{4}{3}, \frac{7}{3}\right), \left(\frac{4}{3}, \frac{8}{3}\right), \left(\frac{4}{3}, 3\right), \left(\frac{4}{3}, \frac{10}{3}\right),$$
$$\left.\left(\frac{4}{3}, \frac{11}{3}\right), \left(\frac{4}{3}, 4\right), \left(\frac{3}{2}, \frac{5}{2}\right), \left(\frac{3}{2}, 3\right), \left(\frac{3}{2}, \frac{7}{2}\right), \left(\frac{3}{2}, 4\right), \left(\frac{3}{2}, \frac{9}{2}\right), \left(\frac{3}{2}, 5\right), (4,5) \right\}$$
$$\cup \left(\{(2,t), (t, 2t-2), (t, 2t-1), (t, 3t-3) : t \in \mathbb{R}\} \cap \{(c,d) : d > c > 1\}\right)$$

Démonstration. Soit l'ensemble d'équations paramétriques

$$\begin{aligned}\mathcal{X} = &\{(5/4, t), (4/3, t), (3/2, t), (2, t), (t, 6-4t), (t, 5-3t), (t, 4-2t),\\ &(t, 3-t), (t, 2t-3), (t, 2t-2), (t, 2t-1), (t, 3t-3), (t, 2)\}\end{aligned}$$

Pour tout couple $e = (x(t), y(t))$ d'équations paramétriques, on note $\mathcal{C}(e)$ la courbe paramétrée associée (l'ensemble de points du plan définit par $\{(x(t), y(t)) : t \in \mathbb{R}\}$). D'après le théorème précédent, pour tous $c, d \in \mathbb{R}$ avec $c > d > 1$ et $(c,d) \notin \bigcup_{e \in \mathcal{X}} \mathcal{C}(e)$ les cubes additifs sont évitables sur $\{0, 1, c, d\}$. De plus, pour tous $c, d \in \mathbb{R}$ avec $d > c > 1$, l'alphabet $\{0, 1, c, d\}$ est équivalent à l'alphabet $\{0, 1, \frac{d-1}{d-c}, \frac{d}{d-c}\}$ (par la transformation

affine $x \mapsto \frac{d-x}{d-c}$). Soit $f : \mathbb{R}^2 \to \mathbb{R}^2$, $(x, y) \mapsto \left(\frac{y-1}{y-x}, \frac{y}{y-x}\right)$. On en déduit que, pour tous $c, d \in \mathbb{R}$ avec $d > c > 1$ et $(c, d) \notin \bigcup_{e\in\mathcal{X}} \mathcal{C}(f \circ e)$ les cubes additifs sont évitables sur $\{0, 1, c, d\}$. Soit

$$\mathcal{F} = \left(\bigcup_{e\in\mathcal{X}} \mathcal{C}(f \circ e)\right) \cap \left(\bigcup_{e\in\mathcal{X}} \mathcal{C}(e)\right) \cap \{(c, d) : d > c > 1\}.$$

On obtient finalement que pour tous $c, d \in \mathbb{R}$ avec $d > c > 1$ et $(c, d) \notin \mathcal{F}$ les cubes additifs sont évitables sur $\{0, 1, c, d\}$. Calculons précisément $\mathcal{F}$. Tout d'abord

$$\mathcal{C}(\{f \circ e : e \in \mathcal{X}\}) = \mathcal{C}\left(\left\{(t, 6t-4), (t, 5t-3), (t, 4t-2), (t, 3t-1), (t, \frac{3}{2}t-1),\right.\right.$$
$$\left.\left.(t, 2(t-1)), (2, t)(t, 3(t-1)), (t, 2t), (t, 5t-4), (t, 4t-3), (t, 3t-2), (t, 2t-1)\right\}\right).$$

On obtient les ensembles donnés dans le théorème en calculant l'intersection des deux ensembles (ce qui s'obtient en résolvant 169 équations grâce à Mathematica par exemple). □

5.4 Le cas du mot infini $\mathbf{W}_{1,0,c,d}$

Nous effectuons globalement la même preuve que dans la partie précédente pour prouver le résultat suivant.

Théorème 5.4.1. *Soient $c, d \in \mathbb{R}$. Supposons qu'on ait $d > c > 1$, $d \notin \{2, c+1, c+2, 2c+2, 2c+1, 2c, 3c, 3c+1, 1+\frac{c}{2}, \frac{1}{2}+c\}$, alors $\mathbf{W}_{1,0,c,d}$ évite les cubes additifs.*

Démonstration. En suivant la preuve du Théorème 5.3.1, il suffit de prouver que, sous nos hypothèses, pour tous $m, n \in \mathbb{Z}$ avec $mc + nd \in \mathbb{Z}$, $|\tau \cdot (m\alpha + n\beta)| > C$, où $\alpha = (-c, c-1, 1, 0)$ et $\beta = (-d, d-1, 0, 1)$. Montrons d'abord que c'est le cas lorsque $n = 0$. Nous n'avons besoin que de deux sous-cas.

- Si $c > 1.71$ un calcul donne $|\tau \cdot \alpha| > C$ et $|\tau \cdot m\alpha| > C$.
- Si $c \in]1, 2[$, un calcul donne $|\tau \cdot \alpha| > \frac{C}{2}$. De plus, dans ce cas, les conditions $m \in \mathbb{Z}$ et $mc \in \mathbb{Z}$ impliquent que $|m| \geq 2$ (car $c \notin \mathbb{Z}$) et on obtient $|\tau \cdot m\alpha| \geq C$.

Montrons maintenant que le résultat est vrai pour $|n| \geq 4$ et $m \in \mathbb{Z}$. Le même calcul donne

$$|m\tau \cdot \alpha + n\tau \cdot \beta| \geq k|n| \tag{5.3}$$

avec

$$k = |\tau \cdot \alpha| \left|\mathrm{Im}\left(\frac{\tau \cdot \beta}{\tau \cdot \alpha}\right)\right|.$$

En effectuant exactement le même raisonnement que dans la preuve du Théorème 5.3.1 on vérifie que $k^2 - (\frac{C}{4})^2 > 0$ pour tous $d > c > 1$. En combinant avec l'équation (5.3) on obtient que si $|n| \geq 4$, alors

$$|m\tau \cdot \alpha + n\tau \cdot \beta| > C.$$

Il reste à étudier les cas $|n| \in \{1, 2, 3\}$, qui se ramènent aux cas $n \in \{1, 2, 3\}$. Commençons par étudier le cas $n = 1$. Encore une fois $P_{c,d,1}(m) := |\tau \cdot (m\alpha + \beta)|^2 - C^2$ est une équation du second degré en d. En calculant le discriminant on obtient $P_{c,d,1}(m) > 0, \forall c \in \mathbb{R}$ si et seulement si

$$\Delta_c(d) :\doteq 25.3914 + 3.07144m - 1.25108m^2 < 0.$$

Ceci est une équation du second degré en m, en la résolvant on obtient

$$m \notin [-3.44178, 5.89681] \implies |\tau \cdot (m\alpha + \beta)| > C.$$

Nous devons vérifier que pour tout $m \in \{5, 4, 3, 2, 1, 0, -1, -2, -3\}$ tel que $mc + d \in \mathbb{Z}$, $P_{c,d,1}(m) > 0$. Les calculs s'effectuent de la même façon que dans la preuve du Théorème 5.3.1 et on donne, dans le Tableau 5.4 les hypothèses et les conditions sur les réels qui nous permettent de conclure.

Tableau 5.4 – Théorème 5.4.1 : Étude de $P_{c,d,1}(m)$ pour $m \in \{5, 4, 3, 2, 1, 0, -1, -2, -3\}$.

(C_1) : une condition équivalente pour d	**Une condition suffisante pour obtenir (C_1)**
$P_{c,d,1}(5) > 0 \Leftrightarrow d \notin]-0.781405 - 5c, 1.35518 - 5c[$	$d > c > 1$
$P_{c,d,1}(4) > 0 \Leftrightarrow d \notin]-1.1107 - 4c, 1.80675 - 4c[$	$d > c > 1$
$P_{c,d,1}(3) > 0 \Leftrightarrow d \notin]-1.26804 - 3c, 2.08636 - 3c[$	$d > c > 1$
$P_{c,d,1}(2) > 0 \Leftrightarrow d \notin]-1.31762 - 2c, 2.25822 - 2c[$	$d > c > 1$
$P_{c,d,1}(1) > 0 \Leftrightarrow d \notin]-1.27931 - c, d > 2.34218 - c[$	$d > c > 1$
$P_{c,d,1}(0) > 0 \Leftrightarrow d \notin [-1.15655, 2.34171]$	$d > c > 1$ et $d \neq 2$
$P_{c,d,1}(-1) > 0 \Leftrightarrow d \notin]-0.939592 + c, 2.24702 + c[$	$d > c$ et $d \notin \{c + 1, c + 2\}$
$P_{c,d,1}(-2) > 0 \Leftrightarrow d \notin]-0.595226 + 2c, 2.02493 + 2c[$	$d \notin \{2c + 2, 2c + 1, 2c\}$
$P_{c,d,1}(-3) > 0 \Leftrightarrow d \notin]0.00625218 + 3c, 1.54573 + 3c)[$	$d \notin \{3c, 3c + 1\}$

Pour le cas $n = 2$, les éléments à vérifier sont les cas $m \notin [-2.21427, 7.12433]$. Nous devons vérifier que, sous nos hypothèses, pour tout $m \in \{7, 6, 5, 4, 3, 2, 1, 0, -1, -2\}$ tel que $mc + 2d \in \mathbb{Z}$, $P_{c,d,2}(m) > 0$. On donne, dans le Tableau 5.5 les conditions sur les réels et les hypothèses qui nous permettent de conclure.

Enfin, pour le cas $n = 3$ on utilise la même méthode. Le calcul du discriminant de $P_{c,d,3}(m)$, vu comme un trinôme en d, permet d'obtenir

$$m \notin [-0.986756, 8.35184] \implies |\tau \cdot (m\alpha + \beta)| > C.$$

Tableau 5.5 – Théorème 5.4.1 : Étude de $P_{c,d,2}(m)$ pour $m \in \{7,6,5,4,3,2,1,0,-1,-2\}$.

(C_2) : une condition équivalente pour d	Une condition suffisante pour obtenir (C_2)
$P_{c,d,2}(7) > 0 \Leftrightarrow d \notin]0.170814 - 3.5c, 0.586373 - 3.5c[$	$d > c > 1$
$P_{c,d,2}(6) > 0 \Leftrightarrow d \notin]-0.180798 - 3c, 0.999122 - 3c[$	$d > c > 1$
$P_{c,d,2}(5) > 0 \Leftrightarrow d \notin]-0.320242 - 2.5c, 1.19971 - 2.5c[$	$d > c > 1$
$P_{c,d,2}(4) > 0 \Leftrightarrow d \notin]-0.385091 - 2.c, 1.32569 - 2.c[$	$d > c > 1$
$P_{c,d,2}(3) > 0 \Leftrightarrow d \notin]-0.399384 - 1.5c, 1.40112 - 1.5c[$	$d > c > 1$
$P_{c,d,2}(2) > 0 \Leftrightarrow d \notin]-0.370696 - c, 1.43357 - c[$	$d > c > 1$
$P_{c,d,2}(1) > 0 \Leftrightarrow d \notin]-0.299307 - 0.5c, 1.42332 - 0.5c[$	$d > c > 1$
$P_{c,d,2}(0) > 0 \Leftrightarrow d \notin]-0.178467, 1.36362[$	$d > 1$
$P_{c,d,2}(-1) > 0 \Leftrightarrow d \notin]0.0134084 + 0.5c, 1.23288 + 0.5c[$	$d > c$ et $d \neq 1 + \frac{c}{2}$
$P_{c,d,2}(-2) > 0 \Leftrightarrow d \notin]0.382276 + c, 0.925154 + c[$	$d > c$ et $d \neq \frac{1}{2} + c$

Nous devons vérifier, sous nos hypothèses, que pour tout $m \in \{8,7,6,5,4,3,2,1,0\}$ tel que $mc + 3d \in \mathbb{Z}$, $P_{c,d,3}(m) > 0$. Les 9 équations permettent de terminer la preuve du Théorème 5.4.1 □

5.5 Les alphabets restants

Grâce aux Théorème 5.3.2 et Théorème 5.4.1 on obtient :

Théorème 5.5.1. *Soit*

$$\mathcal{F} = \left\{ \left(\frac{10}{9}, \frac{14}{9}\right), \left(\frac{9}{8}, \frac{13}{8}\right), \left(\frac{8}{7}, \frac{11}{7}\right), \left(\frac{7}{6}, \frac{5}{3}\right), \left(\frac{6}{5}, \frac{8}{5}\right), \left(\frac{6}{5}, 2\right), \left(\frac{5}{4}, \frac{7}{4}\right), \left(\frac{5}{4}, 2\right), \right.$$
$$\left(\frac{5}{4}, \frac{9}{4}\right), \left(\frac{5}{4}, \frac{5}{2}\right), \left(\frac{5}{4}, \frac{13}{4}\right), \left(\frac{5}{4}, \frac{7}{2}\right), \left(\frac{4}{3}, 2\right), \left(\frac{4}{3}, \frac{7}{3}\right), \left(\frac{4}{3}, \frac{8}{3}\right), \left(\frac{4}{3}, \frac{10}{3}\right), \left(\frac{4}{3}, \frac{11}{3}\right),$$
$$\left(\frac{3}{2}, \frac{5}{2}\right), \left(\frac{3}{2}, 3\right), \left(\frac{3}{2}, \frac{7}{2}\right), (4,5), \left(\frac{4}{3}, \frac{5}{3}\right), \left(\frac{3}{2}, 2\right), \left(\frac{8}{5}, \frac{9}{5}\right), \left(\frac{5}{3}, 2\right), \left(\frac{7}{4}, \frac{9}{4}\right),$$
$$\left. \left(2, \frac{5}{2}\right), (2,3), (2,4), (2,5), \left(\frac{5}{2}, 3\right), \left(\frac{5}{2}, \frac{9}{2}\right), (3,4), (3,5), (3,6), (4,6), (4,9) \right\}$$

et $(c,d) \in \mathbb{R}^2 \setminus \mathcal{F}$*, alors les cubes additifs sont évitables sur* $\{0,1,c,d\}$.

Démonstration. Cet ensemble est obtenu en prenant l'intersection des ensembles de paires interdites des Théorème 5.3.2 et Théorème 5.4.1. □

Nous rappelons les théorèmes suivants, issus de la littérature, afin d'étudier les cas restants (ceux de la forme $\{0,1,c,d\}$ avec $(c,d) \in \mathcal{F}$).

Théorème 5.5.2 ([37]). *Les cubes additifs sont évitables sur les alphabets suivants :*

$$\{0,1,5\},\{0,1,6\},\{0,1,7\},\{0,2,7\},\{0,3,7\},\{0,1,8\},\{0,3,8\},\{0,1,9\},\{0,2,9\},\{0,4,9\}.$$

Théorème 5.5.3 ([39, Théorème 9]). *Les cubes additifs sont évitables sur les alphabets*

$$\{0,2,3,6\},\{0,1,2,4\},\{0,2,3,5\}.$$

Presque tous les alphabets restants contiennent en fait un alphabet équivalent à un des Théorème 5.5.2 ou Théorème 5.5.3. Ceci nous permet d'obtenir notre résultat principal :

Théorème 5.5.4. *Pour tous rationnels p et q avec $p < q$ et $(p,q) \neq (2,3)$, les cubes additifs sont évitables sur $\{0,1,p,q\}$.*

Démonstration. $\left\{0,1,\frac{10}{9},\frac{14}{9}\right\}$ contient un alphabet équivalent à $\{0,1,5\}$ (on applique la transformation affine $x \mapsto 9x-9$ à l'ensemble $\left\{1,\frac{10}{9},\frac{14}{9}\right\}$). D'après le Théorème 5.5.2, il est possible de construire sur ces deux alphabets un mot infini sans cube additif. On effectue la même démarche pour chacun des alphabets restants et le Tableau 5.6 donne l'alphabet équivalent issu du Théorème 5.5.2 ou du Théorème 5.5.3. Ceci termine la

Tableau 5.6 – Tous les alphabets restants, à l'exception de $\{0,1,2,3\}$, contiennent un alphabet équivalent à un issu des Théorème 5.5.2 ou Théorème 5.5.3.

$\left(\frac{10}{9},\frac{14}{9}\right),\left(\frac{5}{4},\frac{7}{2}\right),\left(\frac{4}{3},\frac{5}{3}\right),(4,5),\left(2,\frac{5}{2}\right),(2,5),(3,5)$	$\{0,1,5\}$
$\left(\frac{6}{5},\frac{8}{5}\right),\left(\frac{6}{5},2\right),\left(\frac{5}{4},\frac{5}{2}\right),\left(\frac{5}{3},2\right),\left(\frac{5}{2},3\right),(3,6),(4,6)$	$\{0,1,6\}$
$\left(\frac{7}{6},\frac{5}{3}\right),\left(\frac{4}{3},\frac{10}{3}\right)$	$\{0,1,7\}$
$\left(\frac{3}{2},\frac{7}{2}\right)$	$\{0,2,7\}$
$\left(\frac{5}{4},\frac{7}{4}\right),\left(\frac{4}{3},\frac{7}{3}\right)$	$\{0,3,7\}$
$\left(\frac{8}{7},\frac{11}{7}\right),\left(\frac{4}{3},\frac{11}{3}\right),\left(\frac{3}{2},\frac{5}{2}\right)$	$\{0,1,8\}$
$\left(\frac{5}{4},2\right),\left(\frac{4}{3},\frac{8}{3}\right)$	$\{0,3,8\}$
$\left(\frac{9}{8},\frac{13}{8}\right),\left(\frac{5}{4},\frac{13}{4}\right),\left(\frac{8}{5},\frac{9}{5}\right)$	$\{0,1,9\}$
$\left(\frac{5}{2},\frac{9}{2}\right)$	$\{0,2,9\}$
$\left(\frac{5}{4},\frac{9}{4}\right),\left(\frac{7}{4},\frac{9}{4}\right),(4,9)$	$\{0,4,9\}$
$\left(\frac{4}{3},2\right),\left(\frac{3}{2},3\right)$	$\{0,2,3,6\}$
$\left(\frac{3}{2},2\right),(2,4)$	$\{0,1,2,4\}$
$(3,4)$	$\{0,1,3,4\}$

preuve. □

En reformulant on obtient :

Corollaire 5.5.5. *Soit $\mathcal{A} \subset \mathbb{C}$ un alphabet vérifiant $|\mathcal{A}| \geq 4$. Si $\mathcal{A}$ n'est pas équivalent à $\{0,1,2,3\}$ alors les cubes additifs sont évitables sur $\mathcal{A}$. En particulier si $|\mathcal{A}| \geq 5$ alors les cubes additifs sont évitables sur $\mathcal{A}$.*

Notons que nous avons montré que pour tous les alphabets d'entiers de taille 4 sauf un nombre fini (à une relation d'équivalence près définie dans la Partie 1.3.11) le mot $\mathbf{W}_{a,b,c,d}$ peut être utilisé pour éviter les cubes additifs. Ce n'est probablement pas le seul point fixe infini avec cette propriété. En effet, tant que la matrice d'adjacence du morphisme a au moins une valeur propre de norme strictement plus petite que 1, on peut déduire une inégalité similaire à celle du Théorème 5.2.2 (voir Proposition 7 dans [39] pour plus de détails). Si le mot évite les cubes abéliens, on peut montrer une inégalité similaire à celle du Lemme 5.2.3. La condition de ce lemme sera alors suffisamment forte pour étudier le réseau euclidien de la même manière que celle utilisée ici.

5.6 Démonstration de la Proposition 2.3.3

Rappelons la proposition énoncée en Section 2.3.2 et démontrons la maintenant que nous disposons des résultats nécessaires.

Proposition 2.3.3. *Sur tout alphabet $\{a,b,c,d\}$ non-équivalent à $\{0,1,2,3\}$, il existe un morphisme similaire à φ_0 vérifiant la condition* Cupisca*. De plus si $\{a,b,c,d\}$ non-équivalent à $\{0,1,2,4\}$ alors il existe un morphisme substitution de φ_0 vérifiant la condition* Cupisca *sur $\{a,b,c,d\}$.*

Tout d'abord, grâce aux Théorème 5.3.1 et Théorème 5.4.1, nous connaissons une infinité d'alphabets sur lesquels un morphisme substitution de φ_0 vérifie la condition Cupisca énoncée page 17. Supposons étudier un alphabet $\mathcal{A}'$ qui soit équivalent à un alphabet $\mathcal{A}$ sur lequel $\varphi_{a,b,c,d}$ vérifie la condition Cupisca. Il existe alors une substitution lettre à lettre $s : \mathcal{A} \to \mathcal{A}'$ qui permet d'établir l'équivalence entre ces deux alphabets et qui préserve les puissances additives. Le morphisme $s \circ \varphi_{a,b,c,d} \circ s^{-1} : \mathcal{A}' \to \mathcal{A}'$ vérifie donc la condition Cupisca et est une substitution de φ_0. On obtient le lemme suivant.

Lemme 5.6.1. *Si $\mathcal{A}'$ est équivalent à un alphabet $\mathcal{A}$ sur lequel un morphisme substitution de φ_0 vérifie la condition* Cupisca*, alors il existe aussi un morphisme substitution de φ_0 sur $\mathcal{A}'$ qui vérifie la condition* Cupisca*.*

On peut donc déduire du Théorème 5.5.1 que nous connaissons sur tout alphabet un morphisme substitution de φ_0, hormis pour les alphabets équivalents à ceux d'un petit ensemble, noté $\mathcal{F}$ dans le Théorème 5.5.1. Les alphabets de l'ensemble $\mathcal{F}$ sont traités grâce à la littérature dans le Théorème 5.5.4, mais nous donnons dans le Tableau 5.7 pour chacun d'eux, exceptés les alphabets $\{0,1,2,3\}$ et $\{0,1,2,4\}$, un alphabet équivalent et un morphisme substitution de φ_0 qui vérifie la condition Cupisca sur cet alphabet. Pour $\{0,1,2,4\}$ nous donnons un morphisme similaire à φ_0. Ceci termine la preuve de la Proposition 2.3.3 grâce au Lemme 5.6.1.

5.7 Démonstration de la Proposition 2.3.2

Rappelons la Proposition 2.3.2 que nous avons énoncé en Section 2.3.1. Elle permettait de consolider le lien entre les morphismes vérifiant la condition Cupisca et les morphismes

Tableau 5.7 – Chaque alphabet restant $(0,1,c,d)$, à l'exception de $\{0,1,2,3\}$, est équivalent a un alphabet (a',b',c',d') sur lequel le morphisme ϕ similaire à φ_0 vérifie la condition Cupisca.

(c,d)	Transformation affine	(a',b',c',d')	ϕ
$(\frac{10}{9},\frac{14}{9})$	$x \mapsto 9x$	$(0,9,10,14)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{5}{4},\frac{7}{2})$	$x \mapsto 4x$	$(0,4,5,14)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{4}{3},\frac{5}{3})$	$x \mapsto 3x$	$(0,3,4,5)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(4,5)$	$x \mapsto x$	$(0,1,4,5)$	$a' \mapsto a'c',\ b' \mapsto d'c',\ c' \mapsto b',\ d' \mapsto a'b'$
$(2,\frac{5}{2})$	$x \mapsto 2x$	$(0,2,4,5)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(2,5)$	$x \mapsto x$	$(0,1,2,5)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(3,5)$	$x \mapsto x$	$(0,1,3,5)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{6}{5},\frac{8}{5})$	$x \mapsto 5x$	$(0,5,6,8)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{6}{5},2)$	$x \mapsto 5x$	$(0,5,6,10)$	$a' \mapsto d'c',\ b' \mapsto c',\ c' \mapsto a'b',\ d' \mapsto d'b'$
$(\frac{5}{4},\frac{5}{2})$	$x \mapsto 4x$	$(0,4,5,10)$	$a' \mapsto a'c',\ b' \mapsto d'c',\ c' \mapsto b',\ d' \mapsto a'b'$
$(\frac{5}{3},2)$	$x \mapsto 3x$	$(0,3,5,6)$	$a' \mapsto d'c',\ b' \mapsto c',\ c' \mapsto a'b',\ d' \mapsto d'b'$
$(\frac{5}{2},3)$	$x \mapsto 2x$	$(0,2,5,6)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(3,6)$	$x \mapsto x$	$(0,1,3,6)$	$a' \mapsto a'c',\ b' \mapsto d'c',\ c' \mapsto b',\ d' \mapsto a'b'$
$(4,6)$	$x \mapsto x$	$(0,1,4,6)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{7}{6},\frac{5}{3})$	$x \mapsto 6x$	$(0,6,7,10)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{4}{3},\frac{10}{3})$	$x \mapsto 3x$	$(0,3,4,10)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{3}{2},\frac{7}{2})$	$x \mapsto 2x$	$(0,2,3,7)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{5}{4},\frac{7}{4})$	$x \mapsto 4x$	$(0,4,5,7)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{4}{3},\frac{7}{3})$	$x \mapsto 3x$	$(0,3,4,7)$	$a' \mapsto a'c',\ b' \mapsto d'c',\ c' \mapsto b',\ d' \mapsto a'b'$
$(\frac{8}{7},\frac{11}{7})$	$x \mapsto 7x$	$(0,7,8,11)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{4}{3},\frac{11}{3})$	$x \mapsto 3x$	$(0,3,4,11)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{3}{2},\frac{5}{2})$	$x \mapsto 2x$	$(0,2,3,5)$	$a' \mapsto a'c',\ b' \mapsto d'c',\ c' \mapsto b',\ d' \mapsto a'b'$
$(\frac{5}{4},2)$	$x \mapsto 4x$	$(0,4,5,8)$	$a' \mapsto d'c',\ b' \mapsto c',\ c' \mapsto a'b',\ d' \mapsto d'b'$
$(\frac{4}{3},\frac{8}{3})$	$x \mapsto 3x$	$(0,3,4,8)$	$a' \mapsto a'c',\ b' \mapsto d'c',\ c' \mapsto b',\ d' \mapsto a'b'$
$(\frac{9}{8},\frac{13}{8})$	$x \mapsto 8x$	$(0,8,9,13)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{5}{4},\frac{13}{4})$	$x \mapsto 4x$	$(0,4,5,13)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{8}{5},\frac{9}{5})$	$x \mapsto 5x$	$(0,5,8,9)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{5}{2},\frac{9}{2})$	$x \mapsto 2x$	$(0,2,5,9)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{5}{4},\frac{9}{4})$	$x \mapsto 4x$	$(0,4,5,9)$	$a' \mapsto a'c',\ b' \mapsto d'c',\ c' \mapsto b',\ d' \mapsto a'b'$
$(\frac{7}{4},\frac{9}{4})$	$x \mapsto 4x$	$(0,4,7,9)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(4,9)$	$x \mapsto x$	$(0,1,4,9)$	$a' \mapsto d'c',\ b' \mapsto b'c',\ c' \mapsto a',\ d' \mapsto b'a'$
$(\frac{4}{3},2)$	$x \mapsto 3x$	$(0,3,4,6)$	$a' \mapsto d'c',\ b' \mapsto c',\ c' \mapsto a'b',\ d' \mapsto d'b'$
$(\frac{3}{2},3)$	$x \mapsto 2x$	$(0,2,3,6)$	$a' \mapsto a'c',\ b' \mapsto d'c',\ c' \mapsto b',\ d' \mapsto a'b'$
$(\frac{3}{2},2)$	$x \mapsto 4-2x$	$(0,1,2,4)$	$a' \mapsto c'b',\ b' \mapsto a'b'b',\ c' \mapsto c'b'd',\ d' \mapsto b'd'd'$
$(2,4)$	$x \mapsto x$	$(0,1,2,4)$	$a' \mapsto c'b',\ b' \mapsto a'b'b',\ c' \mapsto c'b'd',\ d' \mapsto b'd'd'$
$(3,4)$	$x \mapsto x$	$(0,1,3,4)$	$a' \mapsto a'c',\ b' \mapsto d'c',\ c' \mapsto b',\ d' \mapsto a'b'$

substitution de φ_0. Cette proposition a également été laissée sans démonstration mais nous disposons maintenant des outils pour réaliser sa preuve.

Proposition 2.3.2. *Pour tout morphisme φ substitution de φ_0, pour tous réels a et b avec $a < b$, il existe $M_\varphi(a,b) \in \mathbb{R}$ tel que pour tout $c > M_\varphi(a,b)$ il existe $N_\varphi(a,b,c) \in \mathbb{R}$ tel que pour tout $d > N_\varphi(a,b,c)$ le morphisme φ vérifie* Cupisca *sur $\{a,b,c,d\}$.*

Démonstration. Grâce au Lemme 5.6.1 on peut supposer qu'on travaille sur l'alphabet $\{0,1,c' = \frac{c-a}{b-a}, d' = \frac{d-a}{b-a}\}$ (quitte à appliquer la transformation affine $h : x \mapsto (x-a)/(b-a)$). Il existe sur cet alphabet 24 morphismes substitutions de φ_0. Le Théorème 5.3.1 nous assure que le morphisme $\varphi_{0,1,c',d'}$ définit par

$$\varphi_{0,1,c',d'}(0) = 0c' \quad ; \quad \varphi_{0,1,c',d'}(1) = d'c' \quad ; \quad \varphi_{0,1,c',d'}(c') = 1 \quad ; \quad \varphi_{0,1,c',d'}('d) = 01,$$

vérifie la condition Cupisca dès lors que $d' > c' > 1$, $c' \notin \{5/4, 4/3, 3/2, 2\}$ et $d' \notin \{6-4c', 5-3c', 4-2c', 3-c', 2c'-3, 2c'-2, 2c'-1, 3c'-3, 2\}$. Posons

$$\begin{aligned} M'_\varphi(0,1) &= \max\{5/4, 4/3, 3/2, 2\} = 2, \\ N'_\varphi(0,1,c') &= \max\{6-4c', 5-3c', 4-2c', 3-c', 2c'-3, 2c'-2, 2c'-1, 3c'-3, 2\}. \end{aligned}$$

On sait que si $c' > M'_\varphi(0,1)$ et $d' > N'_\varphi(0,1,c')$ alors $\varphi_{0,1,c',d'}$ vérifie la condition Cupisca. Par conséquent, si $c > M_\varphi(a,b) = (b-a)M'_\varphi(0,1) + a$ et $d > N_\varphi(a,b,c) = (b-a)N'_\varphi(0,1,c') + a$ alors $\varphi_{a,b,c,d}$ vérifie la condition Cupisca.

De la même façon, d'après le Théorème 5.4.1, le morphisme $\varphi_{1,0,c',d'}$ définit sur l'alphabet $\{0,1,c',d'\}$ par

$$\varphi_{1,0,c',d'}(0) = 1c' \quad ; \quad \varphi_{1,0,c',d'}(1) = d'c' \quad ; \quad \varphi_{1,0,c',d'}(c') = 0 \quad ; \quad \varphi_{1,0,c',d'}('d) = 10,$$

vérifie la condition Cupisca dès lors que $d' > c' > 1$, $d' \notin \{2, c'+1, c'+2, 2c'+2, 2c'+1, 2c', 3c', 3c'+1, 1+\frac{c'}{2}, \frac{1}{2}+c'\}$. Posons

$$\begin{aligned} M'_\varphi(0,1) &= 1, \\ N'_\varphi(0,1,c') &= \max\{2, c'+1, c'+2, 2c'+2, 2c'+1, 2c', 3c', 3c'+1, 1+\frac{c'}{2}, \frac{1}{2}+c'\}. \end{aligned}$$

On obtient que si $c > M_\varphi(a,b) = (b-a)M'_\varphi(0,1) + a$ et $d > N_\varphi(a,b,c) = (b-a)N'_\varphi(0,1,c') + a$ alors $\varphi_{b,a,c,d}$ vérifie la condition Cupisca.

Ce procédé est également applicable aux 22 morphismes restants : on démontre des théorèmes similaires aux Théorème 5.3.1 et Théorème 5.4.1 et on obtient pour chacun les bornes $M_\varphi(a,b)$ et $N_\varphi(a,b,c)$. □

Chapitre 6

Éviter les cubes additifs sur $\{0, 1, 2, 3\}$

6.1 Les difficultés rencontrées sur cet alphabet

Dans leur article [39] publié en 2018, M. Rao et M. Rosenfeld prouvent que les cubes additifs sont évitables sur l'alphabet $\{0, 1, 2, 4\}$ en utilisant un morphisme de taille 3 et soulèvent une question.

Question 6.1.1. *Peut-on éviter les cubes additifs sur l'alphabet* $\{0, 1, 2, 3\}$ *?*

Cet alphabet est le seul restant des 3 alphabets mis en lumière dans la question que Rao [37] posait déjà en 2015. Il réussissait à l'époque à construire un mot de taille 1.4×10^5 sans cube additif sur cet alphabet. Notons que, suite au Chapitre 5 et plus particulièrement au Corollaire 5.5.5, il s'agit du seul alphabet (à équivalence près, définie en page 10) de taille supérieure ou égale à 4 sur lequel la question de l'évitabilité des cubes additifs n'est pas résolue.

Nous avons effectué de nombreux tests sur $\{0, 1, 2, 3\}$. Pour montrer l'évitabilité des cubes additifs sur les autres alphabets d'entiers, nous avons exhibé un morphisme et démontré que son point fixe infini évite les cubes additifs via des arguments combinatoires. Pour rechercher un morphisme candidat potentiel, il suffit de vérifier s'il a un point fixe infini, de générer un grand préfixe de celui-ci (10^6 lettres par exemple) et de vérifier si ce long préfixe contient ou non des cubes additifs. Si la réponse est non, on peut alors espérer avoir trouvé un mot infini sans cube additif sur l'alphabet $\{0, 1, 2, 3\}$. Nos recherches sur cet alphabet ont été vaines et nous affirmons qu'il n'existe aucun morphisme de taille plus petite que 7 qui vérifie cette propriété. Cependant les tests deviennent de plus en plus longs car le nombre de morphismes à tester croît très rapidement lorsqu'on augmente la taille des images (le Tableau 2.3 répertorie l'ordre de grandeur du nombre de morphismes en fonction de leur taille). Nous avons établit ce résultats à l'aide d'une recherche exhaustive grâce au programme *Looking4Morphisms* et à l'infrastructure de calcul Grid5000 (nous détaillons cette recherche et la façon dont nous utilisons le calcul parallèle en Section 2.2).

Il est important de noter que le changement d'alphabet, même lorsqu'il se fait par substitution lettre à lettre, est loin d'être une opération sans conséquence sur la présence de puissances additives. Par exemple les deux mots ci-dessous diffère d'une substitution lettre à lettre :

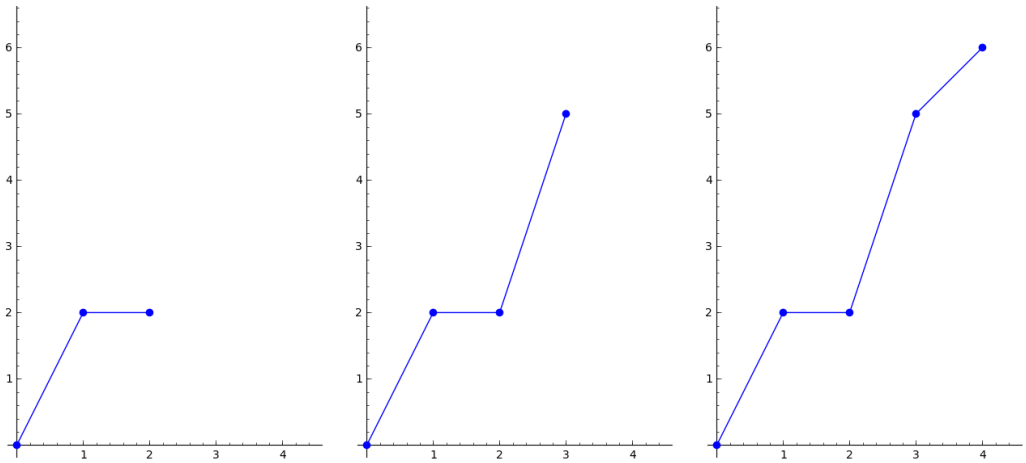
$$\begin{aligned}\mathbf{w} &= 6021062260101 \cdot \overbrace{06026}^{\Sigma=14} \cdot \overbrace{22622}^{\Sigma=14} \cdot\ 6021060101060101\cdots \\ \mathbf{w_0} &= 0314301103434 \cdot \underbrace{30310}_{\Sigma=7} \cdot \underbrace{11011}_{\Sigma=4} \cdot\ 0314303434303434\cdots\end{aligned}$$

Néanmoins, comme nous l'avions vu page 30, le premier contient un carré additif là où le deuxième n'en contient pas. Il semble donc difficile d'utiliser les résultats sur d'autres alphabets pour en tirer des conclusions sur $\{0,1,2,3\}$. M. Rao a également fourni dans [37] des critères pour que l'image par une substitution d'un mot sans puissances additives soit un mot sans puissances additives, mais nos recherches dans cette direction ont également été infructueuses pour conclure sur $\{0,1,2,3\}$.

En outre, les cubes additifs sont évitables grâce à des morphismes de taille 2 (similaires à φ_0) sur tous les alphabets exceptés $\{0,1,2,4\}$ et $\{0,1,2,3\}$. Pour $\{0,1,2,4\}$ il n'existe aucun morphisme de taille 2 permettant de construire un point fixe infini sans cube additif, mais il existe des morphismes de taille 3 avec cette propriété. Ceci constitue une première augmentation de la taille des morphismes requis et pour $\{0,1,2,3\}$, si cette taille existe, elle doit être au moins égale à 7. Enfin, on sait grâce à un argument de A. Freedman et T.C. Brown [26] que les carrés additifs ne sont pas évitables sur $\{0,1,2,3\}$, ni sur aucun alphabet de quatre lettres $\{a,b,c,d\}$ vérifiant $a+d=c+b$. Ils démontrent que tout mot de taille plus grande que 50 sur $\{0,1,2,3\}$ contient nécessairement un carré additif. En fait on peut construire 16 mots de taille 50 sans carrés additifs sur cet alphabet, l'un d'entre eux est :

$$w_{50} = 10203201323020102032013203231323020102032013230201.$$

Cependant et pour contraster avec tout ce qui vient d'être dit, nous avons réussi à construire un mot de taille 70×10^6 sur $\{0,1,2,3\}$ qui ne contient aucun cube additif. Nous utilisons pour cela une technique que nous nommons *Up and Down* dont nous allons entreprendre d'expliquer le principe et la programmation. Nous terminerons en exposant les différentes méthodes que nous avons créées pour tester si les longs mots générés ne contenaient effectivement pas de cubes additifs. Il est nécessaire de tenir compte de la cardinalité des ensembles considérés tout au long de la conception du programme pour obtenir des résultats dans un temps raisonnablement court. Pour cette raison, nous avons choisi un langage de programmation connu pour ses performances, le `C++` et nous parallélisons les calculs lorsque cela nous est possible (voir Section 6.3.3).



(a) Mot : 20. **(b)** Mot : 203. **(c)** Mot : 2031.

Figure 6.1 – Représentation du mot $w = 2031$.

6.2 Créer un mot très long : l'approche *Up and Down*

6.2.1 La représentation graphique des mots

Pour tenter de créer un mot très long sans cube additif nous avons préféré raisonner dans un premier temps graphiquement.

Définition 6.2.1. *On appelle représentation graphique d'un mot w de longueur n l'ensemble*

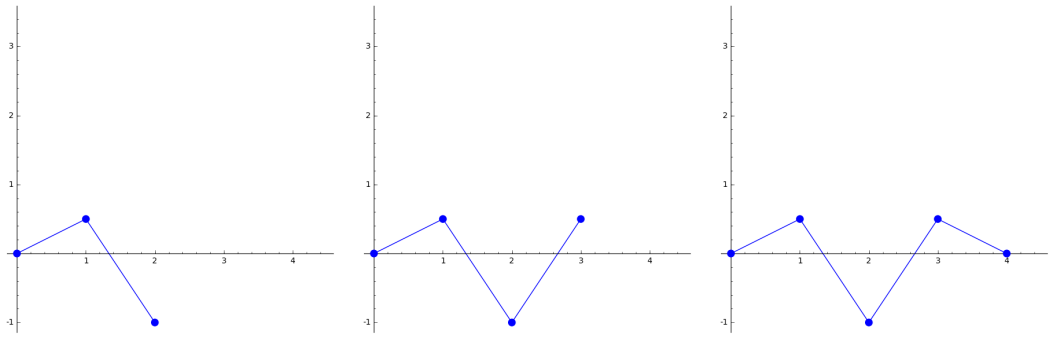
$$\bigcup_{0<p<n} \left[\left(p-1, \sum w[0,p-1) \right), \left(p, \sum w[0,p) \right) \right] \subset \mathbb{R}^2,$$

où $\sum w[0,p)$ désigne la somme des lettres (vues comme des entiers) du préfixe de longueur p de w et vaut 0 si ce préfixe est vide.

De façon plus intuitive, on définit récursivement cette représentation :
- Le mot vide ε est représenté par un point à l'origine du repère.
- Soit w un mot de longueur au moins $p+1$ et dont la représentation du préfixe de longueur p admet comme point de plus grande abscisse (x_p, y_p). Alors on obtient la représentation du préfixe de longueur $p+1$ en ajoutant à la représentation du préfixe de longueur p le segment $[(x_p, y_p), (x_p + 1, y_p + w[p+1])]$, où $w[p+1]$ est vue comme un entier.

Remarque 6.2.1. *Dans le deuxième point on obtient directement $x_p = p$ grâce à la définition.*

Par exemple en Figure 6.1 on observe la construction de la représentation graphique du mot $w = 2031$.



(a) Mot : 20 **(b)** Mot : 203 **(c)** Mot : 2031

Figure 6.2 – Représentation du mot $w = 2031$ avec un décalage de -1.5.

Cette représentation a un inconvénient cependant : si on trace un mot très long la hauteur du graphique risque de devenir très importante. Il suffit, pour palier ce problème, de décaler l'alphabet, c'est-à-dire de soustraire ou d'ajouter la même valeur à toutes les lettres. Concrètement au lieu de considérer que la lettre 3 produit un déplacement de 3 à la verticale et que la lettre 0 ne change pas l'ordonnée, on peut décaler de -1.5 et considérer alors que la lettre 3 correspond à un déplacement vertical de 1.5 et la lettre 0 à un déplacement vertical de -1.5. On applique une translation à l'alphabet et on sait, d'après le Lemme 5.1.1 que cette transformation est sans effet sur la présence ou non de puissances additives. On obtient un nouveau tracé, moins étiré verticalement, représenté en Figure 6.2. Nous appliquerons ce type de transformation pour les représentations graphiques de mots qui suivent.

Nos choix de programmation, qui seront exposés en Section 6.2.4, font que les mots que nous créons ont, relativement à leur longueur, la même proportion de 3 que de 0 et la même proportion de 1 que de 2. Pour cette raison, ce décalage uniforme de 1.5 nous permettra de conserver des représentations "proches" de l'axe des abscisses.

6.2.2 La représentation graphique des puissances additives

Les points (x, y) de la représentation graphique du mot w avec un décalage d vérifient

$$x \in \mathbb{N} \Longrightarrow y = \sum_{i=0}^{x-1} (w[i] - d),$$

où les lettres de w sont vues comme des entiers. Nous traitons ici l'exemple d'un carré additif mais cette propriété se généralise à toute puissance additive. Supposons qu'il existe $p, k \in \mathbb{N}$ avec $k > 0$ tels que w possède un carré additif de longueur $2 \times k$ en

position p, c'est-à-dire,

$$\begin{aligned}
\sum_{i=p}^{p+k-1} w[i] &= \sum_{i=p+k}^{p+2k-1} w[i] \\
\Leftrightarrow \sum_{i=p}^{p+k-1} (w[i]-d) &= \sum_{i=p+k}^{p+2k-1} (w[i]-d) \\
\Leftrightarrow \sum_{i=0}^{p+k-1} (w[i]-d) - \sum_{i=0}^{p-1} (w[i]-d) &= \sum_{i=0}^{p+2k-1} (w[i]-d) - \sum_{i=0}^{p+k-1} (w[i]-d) \\
\Leftrightarrow \frac{\sum_{i=0}^{p+k-1}(w[i]-d) - \sum_{i=0}^{p-1}(w[i]-d)}{(p+1\times k)-(p+0\times k)} &= \frac{\sum_{i=0}^{p+2k-1}(w[i]-d) - \sum_{i=0}^{p+k-1}(w[i]-d)}{(p+2\times k)-(p+1\times k)}.
\end{aligned}$$

Ceci signifie que les points $A = (p, \sum_{i=0}^{p-1}(w[i]-d))$, $B = (p+k, \sum_{i=0}^{p+k-1}(w[i]-d))$ et $C = (p+2k, \sum_{i=0}^{p+2k-1}(w[i]-d))$ appartiennent tous les trois à la représentation graphique de w, sont alignés dans cet ordre et que B est le milieu de $[AC]$. Par exemple, sur la Figure 6.3 on a représenté le mot 203102301202312 qui contient le carré additif $0230\cdot1202$ en position 4.

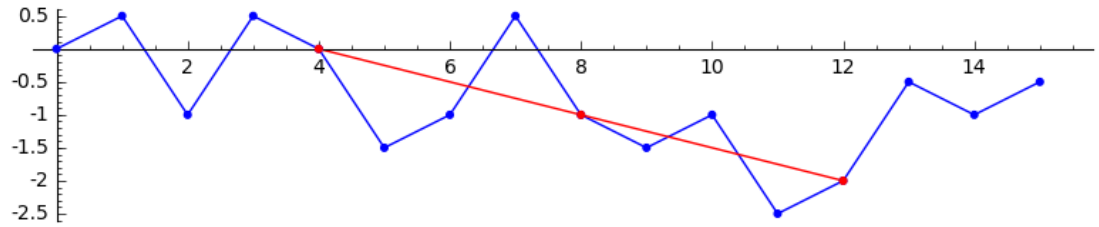


Figure 6.3 – Le mot $2031 \cdot 0230 \cdot 1202 \cdot 312$ représenté avec un carré additif.

6.2.3 L'intuition de l'approche *Up and Down*

6.2.3.1 Utiliser la représentation graphique

Nous expliquons ici la genèse de l'approche *Up and Down* grâce à l'étude des carrés additifs mais les idées restent valables sur les puissances additives en général. Nous connaissons une manière de représenter un mot et ses puissances additives. Nous nous sommes alors demandé quelle est la forme de représentation graphique dans laquelle on retrouve le moins possible de carrés additifs. On devrait dire en fait quelle est la forme de représentation graphique qui correspond à un mot dans lequel on retrouve le moins possible de carrés additifs, mais pour un souci de concision nous allons confondre dans ce paragraphe les mots et leurs représentations graphiques. Une démarche bien connue en combinatoire additive est une construction classique de Behrend [6] qui, pour éviter des progressions arithmétique à trois termes, utilise le fait qu'une droite a au maximum deux

points d'intersection avec une sphère. Une approche similaire, où les droites intersectent le moins possible le graphe de notre mot grâce à des "vagues", est donc assez naturelle.

Pour éviter les puissances additives les segments posent problème car un segment constitué de 3 points entiers sur une représentation graphique correspond à un carré additif dans le mot qu'il représente. Mais supposons qu'on arrive à approcher les segments par les représentations graphiques de mots sans carrés additifs et appelons encore ces approximations des *segments*. On peut, en recollant deux de ces *segments* faire apparaître des carrés additifs, par exemple si le premier *segment* termine par la lettre qui commence le second *segment*. Négligeons encore ces effets de bord : si un mot ne convient pas pour approcher un segment il y a, si le segment est assez long, des centaines d'autres candidats pour l'approcher qui pourront mieux convenir. Donc négligeons le problème de bord lors des recollements. Nous avons cherché à savoir quels carrés additifs peuvent apparaître entre différents *segments*. Nous avons constaté que si l'agencement est bien choisi, il y a en fait assez peu de possibilités pour faire apparaître un carré additif.

6.2.3.2 Limiter les possibilités de puissances additives

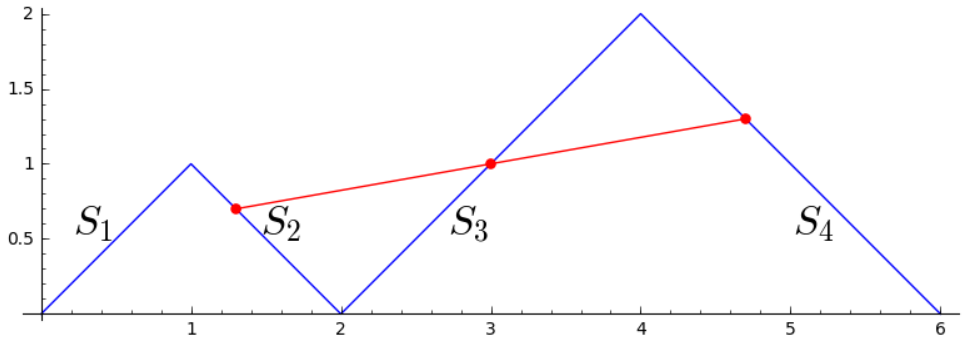


Figure 6.4 – La représentation graphique d'un cas de carré additif.

Dans le cas représenté sur la Figure 6.4, nous représentons 4 segments correspondant à la fonction f définie sur $[0, 8]$ par

$$f(x) = \begin{cases} x & \text{si} \quad 0 \leq x \leq 1, \\ -x+2 & \text{si} \quad 1 \leq x \leq 2, \\ x-2 & \text{si} \quad 2 \leq x \leq 4, \\ -x+6 & \text{si} \quad 4 \leq x \leq 6. \end{cases}$$

On obtient le résultat suivant.

Propriété 6.2.2. *S'il existe 3 points A, B et C appartenant au graphe de f et tels que :*
- *A, B et C ne sont pas tous les trois sur le même segment,*
- *$\overrightarrow{AB} = \overrightarrow{BC}$;*

alors B est le milieu de S_2 ou de S_3 ou alors C est au milieu de S_4.

Démonstration. Supposons qu'il existe trois points A, B et C vérifiant ces conditions. En particulier ils sont alignés dans cet ordre et puisqu'il ne sont pas tous les trois sur un même segment ils sont tous sur des segments différents. Il y a trois cas à considérer : $A \in S_1$ et $C \in S_3$, $A \in S_1$ et $C \in S_4$ ou $A \in S_2$ et $C \in S_4$. Notons x_A, x_B, x_C les abscisses respectives des points A, B et C et y_A, y_B, y_C leurs ordonnées. En particulier $x_B = (x_A + x_C)/2$ et $y_B = (y_A + y_C)/2$. Supposons que $A \in S_1$ et $C \in S_3$, on obtient alors que $B \in S_2$ car deux de ces trois points ne peuvent pas être sur le même segment. On a

$$y_A = x_A \quad ; \quad y_B = -x_B + 2 \quad ; \quad y_C = x_C - 2.$$

Ce qui donne :

$$\begin{aligned} & y_A + y_C = x_A + x_C - 2 \\ \Leftrightarrow \quad & 2y_B = 2x_B - 2 \\ \Leftrightarrow \quad & -2x_B + 4 = 2x_B - 2 \\ \Leftrightarrow \quad & 4x_B = 6 \\ \Leftrightarrow \quad & x_B = \frac{3}{2}. \end{aligned}$$

Le point B se trouve donc nécessairement au milieu du segment S_2. Supposons maintenant que $A \in S_1$ et $C \in S_4$,

$$y_A = x_A \quad ; \quad y_C = -x_C + 6.$$

Et les abscisses sont encadrés :

$$0 \leq x_A \leq 1 \text{ et } 4 \leq x_C \leq 6.$$

On a :

$$2 \leq x_B \leq \frac{7}{2}.$$

Ce qui veut dire que $B \in S_3$. Or

$$\begin{aligned} & y_A + y_C = x_A - x_C + 6 \\ \Leftrightarrow \quad & y_B = \frac{1}{2}(x_A - x_C) + 3 \\ \Leftrightarrow \, & x_B - 2 = \frac{1}{2}((2x_B - x_C) - x_C) + 3 \\ \Leftrightarrow \quad & x_C = 5. \end{aligned}$$

Le point C est donc au milieu de S_4 dans ce cas.

Supposons pour terminer que $A \in S_2$ et $C \in S_4$,

$$y_A = -x_A + 2 \quad ; \quad y_B = x_B - 2 \quad ; \quad y_C = -x_C + 6.$$

Par un raisonnement similaire au premier cas étudié on obtient que le point B se trouve au milieu du segment S_2. □

Prenons un mot fini et supposons que, à une homothétie près, sa représentation graphique corresponde à la Figure 6.4. S'il y a un carré additif dans ce mot qui n'est pas sur un des segments, alors soit ce carré additif se termine au milieu du segment S_4 soit sa position centrale se situe au milieu de S_2 ou au milieu de S_3. En réalisant dans la représentation graphique des montées (les segments S_1 et S_3) et des descentes (S_2 et S_4) de tailles bien choisies, on limite ainsi les possibilités de faire apparaître un carré additif.

Représentons graphiquement le mot w_{50} donné en Section 6.1, qui est un des 16 plus long mots sans carré additif sur $\{0, 1, 2, 3\}$. L'allure du graphique obtenu (en Figure 6.5) semble confirmer cette idée que, pour éviter les puissances additives, la représentation doit comporter des oscillations, c'est à dire alterner des parties globalement croissantes et des parties globalement décroissantes.

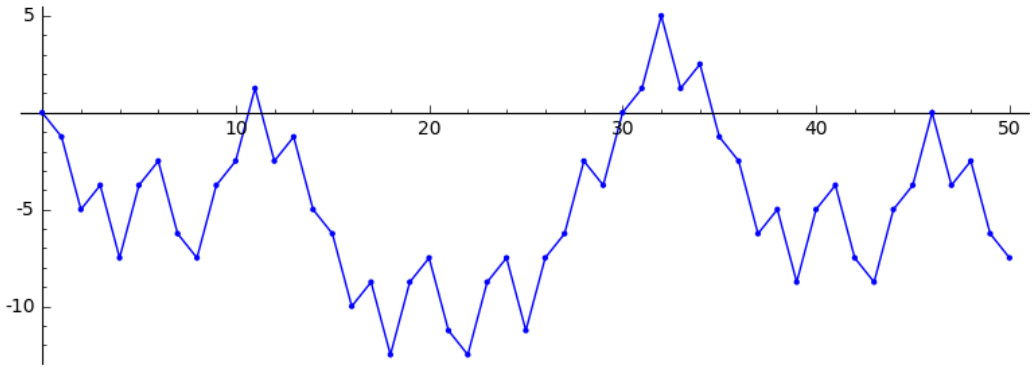


Figure 6.5 – La représentation graphique du mot w_{50}.

Ainsi nous avons eu l'idée de l'approche *Up and Down* et entrepris de transcrire dans la programmation le fait d'alterner le sens de variation au fur et à mesure de la création d'un mot long sans cube additif.

6.2.3.3 De la représentation à la programmation

Il était nécessaire, dans la création d'un programme, de trouver un moyen d'implémenter ce changement de sens. Pour donner sa direction à un segment, il faut favoriser certaines lettres. Rappelons que l'alphabet $\{0, 1, 2, 3\}$ est ici représenté avec un décalage de -1.5, ainsi si on veut créer un segment "croissant" il faudra favoriser les lettre 2 et 3 qui crée une augmentation en ordonnée respectivement de 0.5 et 1.5. À l'inverse si on veut un segment "décroissant" il faudra favoriser les lettres 0 et 1 qui créent respectivement un mouvement en ordonnée de -1.5 et -0.5. Notre but est de créer un mot long sans cube additif sans utiliser une approche morphique. On part d'un mot existant qui n'en contient pas et on essaie, petit à petit, de lui ajouter des lettres en effectuant à chaque fois un test de prolongement qui vérifie que la lettre nouvellement ajoutée n'a pas créé de cube additif. Or, nous verrons dans la Section 6.2.4 que lorsqu'on prolonge un mot existant sans créer de cube additif, on sait quelles lettres vont être testées en priorité pour effectuer ce prolongement. Il suffit alors de changer cet ordre de priorité.

Si on essaye de prolonger par les lettres 0 ou 1 en premier et que cela est possible sans créer de cubes additifs, les lettres 2 et 3 ne seront pas testées donc pas utilisées. Si le test de prolongement est ordonné pour tenter d'ajouter d'abord 0, puis 1 en cas d'échec, puis 2, puis 3, alors le mot créé petit à petit contiendra une majorité de 0 et de 1 et sa représentation graphique sera "décroissante". Si on inverse cet ordre et que le test essaye en priorité d'ajouter 3, puis 2 (en cas d'impossibilité d'ajouter 3 sans créer de cubes additifs), puis 1, puis 0, alors le mot créé petit à petit contiendra une majorité de 3 et de 2 et sa représentation graphique sera "croissante".

6.2.4 La programmation de l'approche *Up and Down*

6.2.4.1 Les conditions à respecter

Quand on est confronté à des problèmes comme la recherche d'un mot long sans cube additif sur un alphabet fixé, on peut être tenté d'utiliser l'approche informatique dite naïve. À partir d'une lettre, par exemple 0, on essaye d'en rajouter une sans créer de cube additif, puis on recommence le processus autant que possible en partant du nouveau mot nouvellement obtenu. Si rajouter une lettre n'est pas possible, alors on revient "un peu" en arrière sur ce qu'on a écrit et on essaie d'autres lettres.

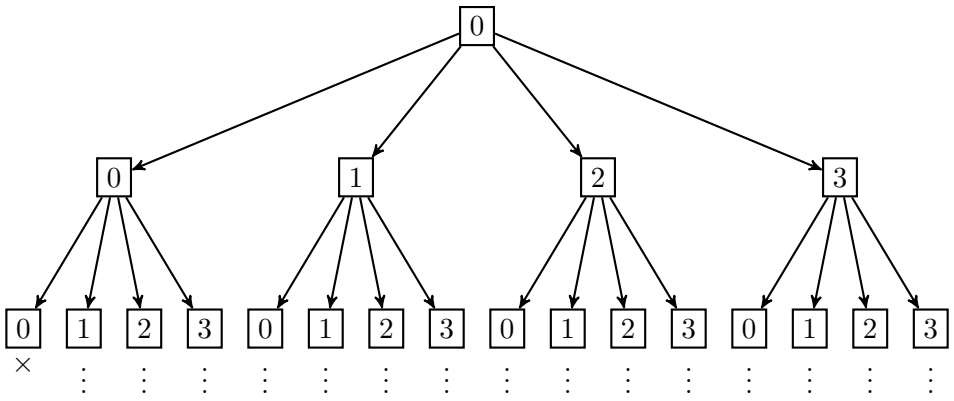


Figure 6.6 – Les trois premiers niveaux de l'arbre de construction du mot sur $\{0, 1, 2, 3\}$.

C'est un parcours d'arbre en profondeur. On conserve dans une pile les possibilités qui n'ont pas encore été explorées et on essaie de prolonger jusqu'à la taille désirée. Cependant, puisqu'on souhaite créer des mots de plusieurs millions de lettres, il faut que la programmation soit efficace en temps de calcul et raisonnable en utilisation de la mémoire. Nous devons donc respecter plusieurs conditions.

— *Condition 1* : afin de pouvoir interrompre les calculs il faut que notre programme puisse partir d'un mot existant dépourvu de cube additif pour le prolonger. Si cette condition n'est pas respectée, on devra recommencer à zéro à chaque fois.

— *Condition 2* : il est nécessaire de définir un critère pour décider quand renverser l'ordre de priorité sur l'alphabet afin d'utiliser l'approche discutée en Section 6.2.3. En pratique, on utilise un compteur du nombre d'échecs, c'est-à-dire le nombre

de tentatives infructueuses de prolonger le mot. On fixe un seuil, ici 1209 et si le compteur atteint ce seuil : on renverse l'ordre de priorité de l'alphabet, on effectue un retour en arrière de plusieurs centaines de lettres (le nombre exact a été changé plusieurs fois au cours de la construction du mot) et on reprend la construction à cet endroit. Le choix de 1209 pour le seuil sera expliqué en Section 6.2.4.2.

— *Condition 3* : l'allocation de mémoire consomme énormément de temps quand on considère des variables contenant plusieurs millions d'éléments. Il nous faut donc allouer dès le début du programme l'espace mémoire nécessaire à la création du mot pour que ce processus ne se fasse qu'une seule fois.

— *Condition 4* : toujours pour optimiser la mémoire, il est possible mais surtout nécessaire de ne pas utiliser de pile pour le parcours en profondeur. En effet, quand la taille du mot atteint plusieurs millions, il est impossible de stocker à chaque lettre ajoutée des mots de cette taille dans l'éventualité où on devrait recommencer à cette position après un retour en arrière. Mais revenons sur la façon de prolonger le mot, imaginons que celui obtenu soit $p032$, avec $p \in \{0, 1, 2, 3\}^*$ et qu'on ne puisse pas ajouter une lettre sans créer un cube additif. Le mot à tester ensuite dans l'ordre lexicographique est $p033$. Supposons que ce mot ne puisse pas non plus être prolongé, le suivant dans l'ordre lexicographique est alors $p1$ car on ne peut pas augmenter la valeur des dernières lettres en restant dans l'alphabet. C'est cet ordre que suit notre parcours de graphe et nous pouvons le suivre sans utiliser de pile : toutes les informations sont contenues dans les lettres du mot.

— *Condition 5* : il nous faut trouver une façon très efficace de tester la présence éventuelle de cubes additifs dans le mot car c'est ce test qui va être effectué des millions de fois, à chaque tentative d'ajout de lettres. Plus précisément, nos expérimentations montrent que le nombre de lettres testées pour obtenir une taille de mot donnée est proche d'une fonction linéaire de la taille du mot : on teste en moyenne 5 lettres pour pouvoir prolonger la longueur de 1 sans créer de cube additif. Pour un mot de taille un million il y aura approximativement cinq millions d'appels à la fonction qui teste la présence de cubes additifs. Nous discutons dans la Section 6.3 des améliorations successives que nous avons apportées à cette fonction.

6.2.4.2 Déterminer le seuil d'échec

Nous n'avons pas choisi au hasard un seuil de 1209 pour le nombre d'échecs. Pour déterminer la valeur du seuil et être le plus efficace possible (c'est-à-dire générer les mots de taille fixées dans le temps le plus court possible), nous avons procédé selon une démarche heuristique. Pour ce faire, nous avons lancé le même programme avec des seuils allant de 500 à 2000 et nous avons retenu celui qui obtenait un mot de longueur 10000 sans cube additif sur l'alphabet $\{0, 1, 2, 3\}$ le plus rapidement. Le programme est le suivant.

```
int main(int argc, char** argv)
{
  //initialisation du chronometre
```

```
  auto start = std::chrono::high_resolution_clock::now();
  //la fonction test cree un mot de longueur argv[1]
  //en utilisant un backtrack de 100 et un
  //seuil de 500
  test(atoi(argv[1]), 500, 100);
  auto stop = std::chrono::high_resolution_clock::now();
  //on calcule la duree d'execution
  auto duration = std::chrono::
  duration_cast<microseconds>(stop - start);
  //initialisation de seuil_min et de min avec
  //les valeurs calculees
  unsigned int seuil_min = 500;
  unsigned int min = duration.count();
  //On teste tous les seuils entre 500 et 2000
  for(int seuil = 500 ; seuil < 2000 ; ++seuil){
    start = std::chrono::high_resolution_clock::now();
    test(atoi(argv[1]), seuil, 100);
    stop = std::chrono::high_resolution_clock::now();
    duration = std::chrono::
    duration_cast<microseconds>(stop - start);
    //changement du seuil_min si on constate une amelioration
    if (duration.count() < min){
      min = duration.count();
      seuil_min = seuil;
    }
  }
  //le seuil qui a permis de conclure le plus rapidement
  // est affiche
  std::cout << "Seuil␣min␣:␣" << seuil_min << std::endl;
  return 1;
}
```

6.2.4.3 Programmation en respectant la Condition 4

Nous écrivons en langage `C++` un programme conforme à toutes ces conditions pour obtenir un mot sans cube additif sur l'alphabet $\{0,1,2,3\}$ d'une longueur largement supérieure à la précédente borne évoquée dans la Section 6.1. Commençons par expliciter ci-après les fonctions `next` et `nextR` qui prennent en entrée un mot et retournent le mot suivant dans l'ordre lexicographique naturel pour `next` et le suivant en considérant l'ordre inversé pour `nextR`. Ce sont ces fonctions qui permettent de privilégier une montée ou une descente lors de la création du mot et de respecter la Condition 4 car elles permettent de déduire à partir du mot existant le prochain à étudier sans utiliser de pile.

```
#include <iostream>
#include <iomanip>
#include <fstream>      // std::fstream
#include <sstream>      // std::stringstream
#include <chrono>
```

```
#include <smmintrin.h> // flag -msse4.1
#include <emmintrin.h> // flag -msse2

#define MIN_LETTER '0'
#define MAX_LETTER '3'
typedef unsigned long long int ull;

void next(ull* Sums, ull& LENGTH, char start, ull& failed){
  char c;
  while(1){
    for(c = start ; c <= MAX_LETTER ; ++c){
      // on essaie la lettre c:
      Sums[LENGTH+1] = Sums[LENGTH]+c;
      if (is_additive_cube_free(Sums, LENGTH+1 )){
  LENGTH++;
  return ;
      }
      else{
  ++(failed);
      }
    }
    LENGTH--; //on backtracke
    start = Sums[LENGTH+1]-Sums[LENGTH]+1;
  }
}

void nextR(ull* Sums, ull& LENGTH, char start, ull& failed){
  char c;
  while(1){
    for(c = start ; c >= MIN_LETTER ; --c){
    // on essaie la lettre c:
      Sums[LENGTH+1] = Sums[LENGTH]+c;
      if (is_additive_cube_free(Sums, 1+LENGTH)){
  LENGTH++;
  return ;
      }
      else{
  ++(failed);
      }
    }
    LENGTH--; //on backtracke
    start = Sums[LENGTH+1]-Sums[LENGTH]-1;
  }
}
```

6.3 Tester si un mot fini possède des cubes additifs

6.3.1 Un algorithme intuitif et une première amélioration

6.3.1.1 L'algorithme intuitif

Quelle que soit la façon dont nous parcourons l'arbre pour créer un mot de plus en plus long sans cube additif, il est toujours nécessaire de tester les prolongements effectués pour voir s'ils font apparaître ou non des cubes additifs. Un cube additif dans un mot peut être caractérisé par deux données : la position à laquelle il commence et la longueur d'un de ses facteurs que nous noterons respectivement p et k dans les algorithmes. l'Algorithme 2 donne la façon la plus naïve d'effectuer ce test.

Entrées : Un mot fini w
Résultat : Vrai si w est sans cube additif, Faux sinon

```
1  n ← |w| ;
2  si n < 3 alors
3  |  retourner Vrai ;
4  fin
5  s1 ← 0 ;
6  s2 ← 0 ;
7  s3 ← 0 ;
8  pour chaque 0 ≤ p ≤ n − 3 faire
9  |  pour chaque 1 ≤ k ≤ ⌊(n−p)/3⌋ faire
10 |  |  pour chaque 0 ≤ i < k faire
11 |  |  |  s1 ← s1 + w[p + i] ;
12 |  |  |  s2 ← s2 + w[p + k + i] ;
13 |  |  |  s3 ← s3 + w[p + 2k + i] ;
14 |  |  fin
15 |  |  si s1 = s2 alors
16 |  |  |  si s2 = s3 alors
17 |  |  |  |  retourner Faux ;
18 |  |  |  fin
19 |  |  fin
20 |  |  s1 ← 0 ;
21 |  |  s2 ← 0 ;
22 |  |  s3 ← 0 ;
23 |  fin
24 fin
25 retourner Vrai ;
```

Algorithme 2 : Algorithme naïf pour tester l'absence de cubes additifs.

Cet algorithme, efficace sur des mots de petites tailles et qui se programme intuitivement, possède une complexité en $O(|w|^3)$ qui le rend tout à fait inutilisable pour des

mots de plusieurs millions de lettres. Il était donc nécessaire d'améliorer ce programme de test.

6.3.1.2 Améliorer le calcul des sommes

Remarquons que les opérations dans les sommes entre les Lignes 11 et 13 changent très peu lorsque le k est incrémenté : la majorité des termes considérés dans les trois sommes seront les mêmes, seuls les bords des facteurs changent. On fixe la position à laquelle commence le triplet de facteurs que nous souhaitons étudier et on effectue une boucle sur la taille de chacun de ses facteurs (allant de 1 jusqu'au tiers de la longueur restante avant la fin du mot considéré). Supposons qu'on considère un mot w de taille 20 et qu'on étudie la présence de cubes additifs partant de la position 2. Au maximum un facteur d'un tel cube peut être de taille 6 car $2+3\times 6 \leq 20$ mais $2+3\times 7 > 20$. Imaginons que le triple facteur commençant en position 2 et de longueur 3×4 ne soit pas un cube additif, on regarde ensuite le triple facteur qui commence à la même position mais de longueur 3×5 qui a beaucoup de termes en commun avec le triple facteur précédemment étudié :

$$w = 00\overbrace{1001}^{s_1}\overbrace{0020}^{s_2}\overbrace{0100}^{s3}112001$$
$$w = 00\underbrace{10010}_{s_1}\underbrace{02001}_{s_2}\underbrace{00112}_{s3}112001$$

Sur des facteurs très longs cette partie commune est importante et il est intéressant de ne pas recalculer toute la somme mais d'effectuer seulement les changements sur les bords. Nous avons donc programmé l'Algorithme 3 (page 93) qui prend en compte cette particularité afin de réaliser moins de calculs.

Dans cet algorithme on doit gérer indépendamment les cas où le cube additif est composé de 3 facteurs de taille 1 ou de trois facteurs de taille 2 (Lignes 6 et 11). De même pour les cas où le cube additif se situe à la fin du mot (Ligne 28). Cet algorithme possède une complexité en $O(|w|^2)$ et effectue, à chaque passage dans la boucle `tant que` 43 opérations (accès mémoire, additions, tests, etc.).

Remarque 6.3.1. *l'Algorithme 3 a été ici écrit dans le sens le plus intuitif, on teste en premier les cubes pouvant apparaître au début du mot, on le parcourt dans le sens de lecture usuel. Cependant, pour le travail que nous effectuons, il est beaucoup plus intéressant de l'écrire dans l'autre sens, en considérant la position à laquelle se termine un cube additif et non pas celle à laquelle il commence. Ainsi lorsque nous ajoutons une lettre à un mot sans cube additif, si un cube additif apparaît on sait qu'il contient la dernière lettre. On peut donc réutiliser la partie du test qui permet de voir s'il existe un cube additif se terminant avec cette lettre ajoutée. Si la réponse est non alors on sait que le mot a été prolongé sans ajouter de cube additif.*

Entrées : Un mot fini w
Résultat : Vrai si w est sans cube additif, Faux sinon

```
1  n ← |w| ;
2  si n < 3 alors
3  |  retourner Vrai ;
4  fin
5  pour 0 ≤ p ≤ n − 6 faire
6  |  k ← 1;
7  |  s1 ← w[p + k − 1] ; s2 ← w[p + 2k − 1] ; s3 ← w[p + 3k − 1] ;
8  |  si s1 = s2 et s2 = s3 alors
9  |  |  retourner Faux ;
10 |  fin
11 |  k ← k + 1;
12 |  s1 ← s1 + w[p + k − 1];
13 |  s2 ← s2 − w[p + k − 1] + w[p + 2k − 2] + w[p + 2k − 1] ;
14 |  s3 ← s3 − w[p + 2k − 2] + w[p + 3k − 2] + w[p + 3k − 1] ;
15 |  si s1 = s2 et s2 = s3 alors
16 |  |  retourner Faux ;
17 |  fin
18 |  tant que k < ⌊(n−p)/3⌋ faire
19 |  |  k ← k + 1 ;
20 |  |  s1 ← s1 + w[p + k − 1] ;
21 |  |  s2 ← s2 − w[p + k − 1] + w[p + 2k − 2] + w[p + 2k − 1] ;
22 |  |  s3 ← s3−w[p+2k−2]−w[p+2k−1]+w[p+3k−3]+w[p+3k−2]+w[p+3k−1];
23 |  |  si s1 = s2 et s2 = s3 alors
24 |  |  |  retourner Faux ;
25 |  |  fin
26 |  fin
27 fin
28 pour chaque n − 5 ≤ p ≤ n − 3 faire
29 |  s1 ← w[p] ; s2 ← w[p + 1] ; s3 ← w[p + 2] ;
30 |  si s1 = s2 et s2 = s3 alors
31 |  |  retourner Faux ;
32 |  fin
33 fin
34 retourner Vrai
```

Algorithme 3 : Algorithme pour tester l'absence de cubes additifs en décalant les sommes.

6.3.2 Diminuer les calculs et optimiser la gestion de la mémoire

Pour diminuer les temps de calculs, nous avons souhaité trouver une façon plus efficace de détecter les cubes additifs. Nous avons élaboré l'Algorithme 4, également en $O(|w|^2)$, mais effectuant beaucoup moins d'opérations à chaque étape. Au lieu d'utiliser un mot nous allons utiliser le tableau des sommes cumulées[2], c'est-à-dire un tableau contenant dans sa p-ième case la somme de toutes les lettres du préfixe de longueur p. Prenons l'exemple d'un mot $w = 001001002001001120$ de longueur 18 et construisons un tableau T de 19 cases comportant chacune la somme du préfixe correspondant.

p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$w[p]$	0	0	1	0	0	1	0	0	2	0	0	1	0	0	1	1	2	0

$T =$	0	0	0	1	1	1	2	2	2	4	4	4	5	5	5	6	7	9	9

Grâce à ce tableau, la somme du facteur $w[p, q)$ correspond à $T[q] - T[p]$, en indiçant le tableau à partir de 0. Par exemple $w[2, 8) = 100100$ et sa somme vaut $T[8] - T[2] = 2 - 0 = 2$. La construction de ce tableau s'effectue en temps linéaire et nous permet de trouver la somme d'un facteur, aussi grand soit il, en seulement deux accès mémoire et une soustraction.

En outre nous utilisons dans nos fonctions un pointeur sur le tableau des sommes et pas le tableau lui-même. Le passage par référence nous permet de gagner du temps en éliminant des allocations mémoire inutiles. Nous définissons une fonction appelée `is additive cube free` qui prend donc en entrée un tableau d'entiers correspondant au tableau des sommes d'un mot et renvoie un booléen.

Remarque 6.3.2. *En pratique on tente toujours de prolonger un mot qui était au préalable sans cube additif. Si un cube additif apparaît il se termine obligatoirement à la dernière lettre du mot il n'y a donc pas besoin de faire une boucle sur la position situant ce cube additif mais seulement sur sa taille. C'est pourquoi l'Algorithme 4 travaille de la fin du mot vers le début et pas l'inverse.*

On obtient la fonction suivante qui ne détecte que les cubes additifs contenant la dernière lettre du mot.

```
typedef unsigned long long int ull;
bool is_additive_cube_free(ull* S, ull prefix_length){
//Renvoie True si le mot n'a pas de cubes additifs contenant
// sa derniere lettre, renvoie False sinon
  ull longueur = 1;
  ull a,b,c,d;

  a = prefix_length;
  b = prefix_length-longueur;
  c = b-longueur;
  d = c-longueur;
```

2. Nous remercions Sylvain Contassot-Vivier pour ses conseils sur cet algorithme.

```
  while(3*longueur <= prefix_length){
    if ((S[a]-2*S[b]+S[c] == 0) && (S[b]-2*S[c]+S[d] == 0))
      return false;
    --b;
    --(--c);
    --(--(--d));
    ++longueur;
  }
  return true;
}
```

Entrées : T : le tableau des sommes d'un mot fini w
Résultat : Vrai si w est sans cube additif, Faux sinon

$n \leftarrow |T|$;
si $n < 3$ **alors**
  retourner *Vrai* ;
fin
pour $3 \leq p \leq n$ **faire**
  $k \leftarrow 1$;
  $a \leftarrow p$;
  $b \leftarrow p - k$;
  $c \leftarrow b - k$;
  $d \leftarrow c - k$;
  tant que $3k \leq p$ **faire**
    si $T[a] - 2 \times T[b] + T[c] = 0$ *et* $T[b] - 2 \times T[c] + T[d] = 0$ **alors**
      retourner *Faux* ;
    fin
    $k \leftarrow k + 1$;
    $b \leftarrow b - 1$;
    $c \leftarrow c - 2$;
    $d \leftarrow d - 3$;
  fin
fin
retourner *Vrai* ;

Algorithme 4 : Algorithme utilisant le tableau des sommes pour tester l'absence de cubes additifs.

6.3.3 Paralléliser pour encore plus d'efficacité

Cette fonction qui teste uniquement la présence de cubes additifs se terminant à la dernière lettre du mot nous est utile pour paralléliser le programme. Nous construisons une boucle sur un entier k allant de 3 à la longueur du mot dont on veut savoir s'il contient ou non des cubes additifs. À chaque passage on considère le préfixe de longueur

k et on teste l'existence de cubes additifs se terminant à la dernière lettre de ce préfixe. S'il en existe, le programme s'arrête, le mot contient un cube additif. S'il n'en existe pas, la boucle continue et considère le préfixe de longueur $k+1$. Cette boucle peut être parallélisée grâce au modèle de programmation parallèle OpenMP (il s'agit de spécifications multi-plateformes qui permettent de développer rapidement des applications parallèles [3]). Les calculs sont alors répartis sur les threads et le temps de réponse du programme est diminué (divisé par 4 en moyenne pour une utilisation sur 4 cœurs).

6.3.4 Bilan et comparaison

En diminuant le nombre d'informations à stocker et en optimisant la gestion de la mémoire et les calculs, notamment via l'utilisation de pointeurs sur nos objets et d'un tableau contenant les sommes pré-calculées, nous sommes parvenus à diviser par un facteur proche de 2 le temps d'exécution du programme. En parallélisant les calculs nous obtenons ensuite des résultats bien meilleurs. Le Tableau 6.1 récapitule les temps d'exécution pour vérifier qu'un mot de taille n est sans cube additif. Les quatre versions du programme correspondent aux trois algorithmes expliqués précédemment et à la parallélisation du troisième algorithme. Notre programme s'arrêtant lorsqu'il détecte un cube additif, nous l'avons testé dans le pire cas possible, c'est-à-dire sur un mot qui ne contient aucun cube additif : le programme doit parcourir complètement ce mot avant de conclure. Ces tests ont été menés sur un ordinateur portable standard (voir Remarque 1.5.1).

Tableau 6.1 – Temps d'exécution en fonction de l'algorithme utilisé dans le programme vérifiant qu'un mot de taille n est sans cube additif.

n	Temps Algorithme 2	Temps Algorithme 3	Temps Algorithme 4	Temps Algorithme 4 parallélisé
10^4	$0.040s$	$0.028s$	$0.014s$	$0.004s$
10^5	$2.514s$	$2.214s$	$1.256s$	$0.289s$
10^6	$255.6s$	$221.0s$	$133.1s$	$38.0s$

6.4 Programme final et résultat

6.4.1 Le `Main`

Nous ne retranscrivons pas ici la totalité du programme, mais seulement le `Main`. La plupart des fonctions qu'il utilise ont été définies auparavant. Le lecteur souhaitant consulter le programme dans son intégralité peut le faire grâce à l'adresse web donnée page 13.

3. Plus d'informations sur ce modèle de programmation parallèle peuvent être trouvées à l'adresse `https://www.openmp.org/specifications/`

```
  void store_w(ull* Sums, ull Length){
  std::ofstream MyFile;
  std::stringstream ss;
  ss << "ACF_" << std::setw(6) << std::setfill ('0') << Length
  << "_over_" << "0123" << ".txt";
  MyFile.open(ss.str());
  ss.str("");

  for(ull i = 1; i <= Length ; ++i)
    ss << Sums[i]-Sums[i-1]-SHIFT_CHAR_TO_INT ;
  MyFile << ss.str();
  MyFile.close();
}
int main(int argc, char** argv){
  if (argc<2){
    std::cout << "./UpAndDown␣<OBJECTIVE_LENGTH>" << std::endl;
    std::cout << "./UpAndDown␣<OBJECTIVE_LENGTH>␣<FILENAME>"
    << std::endl;
    exit(EXIT_FAILURE);
  }
  ull OBJECTIVE_LENGTH;
  std::string w;
  ull* S ;
  ull longueur_en_cours;

  std::string Alphabet;

  std::chrono::time_point<std::chrono::system_clock> start, end;
  std::time_t end_time ;
  int elapsed_seconds ;

  OBJECTIVE_LENGTH = atoll(argv[1]);
  w.reserve(OBJECTIVE_LENGTH+100000);
  S = (ull*)malloc(sizeof(ull)*OBJECTIVE_LENGTH+100000);

  Alphabet = std::string("0123");
  start = std::chrono::system_clock::now();
  end = std::chrono::system_clock::now();
  end_time = std::chrono::system_clock::to_time_t(end);

  if (argc == 2){
    w = "";
    S[0] = 0;
    longueur_en_cours = 0;
  }
  else {
    std::ifstream File(argv[2]);
    std::cout << "␣filename...␣" << argv[2] << std::endl;
    std::getline(File,w);
```

```
    longueur_en_cours = w.size();
    S[0] = 0;
    File.close();
    for(ull i = 1 ; i <= w.size() ; ++i)
      S[i] = S[i-1] + w[i-1];
  }
  //  ull longueur_atteinte = w.size();
  std::cout << "Longueur␣initiale␣:␣" << longueur_en_cours
  << std::endl;
  std::cout << "Longueur␣souhaitee␣:␣" << OBJECTIVE_LENGTH
  << std::endl;

  ull failed = 0;

  std::ofstream MyFile;
  std::stringstream ss;

  ull nb = longueur_en_cours / 10000 + 1;
  ull next_step = nb*10000;
  do{
    end = std::chrono::system_clock::now();
    end_time = std::chrono::system_clock::to_time_t(end);

    while ((failed < NB_OF_FAILED) && (longueur_en_cours
    < OBJECTIVE_LENGTH) && (longueur_en_cours < next_step ))
      next(S, longueur_en_cours, MIN_LETTER, failed);

    if (longueur_en_cours == OBJECTIVE_LENGTH){
      //longueur_atteinte = longueur_en_cours;
      std::cout << "Longueur␣\t" << std::setw(8)
      << std::setfill ('␣') << longueur_en_cours
      << "␣atteinte␣:␣" << std::ctime(&end_time) ;
      store_w(S, longueur_en_cours);
      continue;
    }
    if (longueur_en_cours == next_step){
      //      longueur_atteinte = longueur_en_cours;
      std::cout << "Longueur␣\t" << std::setw(8)
      << std::setfill ('␣') << longueur_en_cours
      << "␣atteinte␣:␣" << std::ctime(&end_time) ;
      store_w(S, longueur_en_cours);
      next_step += 10000;
    }
    failed = 0;
    longueur_en_cours -= BACKTRACK_SIZE;

    while ((failed < NB_OF_FAILED)  && (longueur_en_cours
    < OBJECTIVE_LENGTH) && (longueur_en_cours < next_step ))
      nextR(S, longueur_en_cours, MAX_LETTER, failed);
```

```
    if (longueur_en_cours == OBJECTIVE_LENGTH){
      //      longueur_atteinte = longueur_en_cours;
      std::cout << "Longueur␣\t" << std::setw(8)
      << std::setfill ('␣') << longueur_en_cours
      << "␣atteinte␣:␣" << std::ctime(&end_time) ;
      store_w(S, longueur_en_cours);
      continue;
    }
    if (longueur_en_cours == next_step){
      //      longueur_atteinte = longueur_en_cours;
      std::cout << "Longueur␣\t" << std::setw(8)
      << std::setfill ('␣') << longueur_en_cours
      << "␣atteinte␣:␣" << std::ctime(&end_time) ;
      store_w(S, longueur_en_cours);
      next_step += 10000;
    }
    longueur_en_cours -= BACKTRACK_SIZE;
    failed= 0;
  }
  while (longueur_en_cours < OBJECTIVE_LENGTH);

  //On ecrit le dernier mot trouve dans le fichier associe
  // longueur_atteinte = longueur_en_cours;
  //store_w(S, longueur_atteinte);

  end = std::chrono::system_clock::now();
  end_time = std::chrono::system_clock::to_time_t(end);
  elapsed_seconds =
  std::chrono::duration_cast<std::chrono::milliseconds>
  (end-start).count();

  std::cout << "Finished␣computation␣at␣" << std::ctime(&end_time)
      << "Elapsed␣time:␣" << elapsed_seconds/1000. << "s\n";

  return 1;
}
```

Remarque 6.4.1. *Notre programme est exécuté sur un seul thread de notre processeur (hormis la fonction testant la présence de cubes additifs). C'est une première méthode qui nous semble efficace car les calculs étant un à un relativement courts, le temps de communication entre les threads compenserait le gain potentiel. En revanche ce n'est pas le cas pour le test qui permet de savoir si un mot possède ou non des cubes additifs et nous utilisons donc la programmation en parallèle dans la Partie 6.3.*

6.4.2 Résultats

Grâce aux techniques de programmation mises en place nous avons obtenu le résultat suivant.

Proposition 6.4.1. *Il existe un mot de longueur* 70880000 *sur l'alphabet* $\{0, 1, 2, 3\}$ *sans cube additif.*

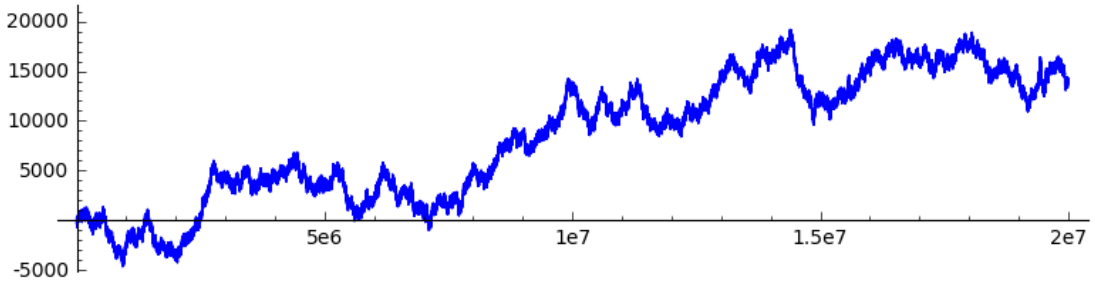


Figure 6.7 – La représentation graphique des 20×10^6 premières lettres d'un mot sans cube additif sur $\{0, 1, 2, 3\}$.

Ce mot est disponible à l'adresse web donnée page 13. La Figure 6.7 est une représentation graphique des 20×10^6 premières lettres du mot. Il possède presque autant de 0 que de 3 (approximativement 28.320% de 0 et 28.300% de 3) et presque autant de 1 que de 2 (approximativement 21.691% de 1 et 21.689% de 2). Ceci est dû au fait que notre méthode de construction donne la priorité tantôt au 0, tantôt au 3 mais jamais au 1 et au 2 en premier. Cependant, l'alternance de l'ordre de priorité de l'alphabet se fait assez fréquemment pour qu'une symétrie apparaisse.

Notons que ce mot de longueur supérieure à 70×10^6 a été construit grâce à un ordinateur de bureau standard (voir Remarque 1.5.1). Nous avons utilisé également le programme *Up and Down* pour construire des mots sans cube additif de taille 10^6 sur tous les alphabets d'entiers $\{0, a, b, c\}$ avec $0 < a < b < c < 10$. Pour chaque alphabet nous avons mesuré le temps que prend la création de ce mot et répertorié les résultats dans le Tableau 6.2.

On observe que, pour tous les alphabets, ce mot est construit en moins de 167.211 secondes, sauf :

- pour les alphabets équivalents à $\{0, 1, 2, 4\}$ ou $\{0, 1, 3, 4\}$ sur lesquels la construction met jusqu'à 171.445 secondes, soit une augmentation de moins de 2.6%,
- pour les alphabets équivalents à $\{0, 1, 2, 3\}$ sur lesquels la construction met environ 188 secondes, soit une augmentation de plus de 12%.

Il semble que la recherche d'un mot sans cube additif sur l'alphabet $\{0, 1, 2, 3\}$ soit plus difficile que sur n'importe quel autre alphabet. Même lorsqu'on considère un alphabet très "proche" comme $\{0, 19, 38, 58\}$: seule la dernière lettre est augmentée de 1 par rapport à l'alphabet $\{0, 19, 38, 57\}$ qui est équivalent à $\{0, 1, 2, 3\}$, c'est pourquoi on considère cet alphabet comme "proche" de $\{0, 1, 2, 3\}$. Malgré cela, sur $\{0, 19, 38, 58\}$, notre programme

Tableau 6.2 – Temps pour construire un mot de taille 10^6 avec le programme *Up and Down* en fonction de l'alphabet testé.

Alphabet	Temps en secondes	Alphabet	Temps en secondes	Alphabet	Temps en secondes
$\{0,1,2,3\}$	188.343	$\{0,1,2,4\}$	170.674	$\{0,1,2,5\}$	164.474
$\{0,1,2,6\}$	162.282	$\{0,1,2,7\}$	161.48	$\{0,1,2,8\}$	161.286
$\{0,1,2,9\}$	160.91	$\{0,1,3,4\}$	170.46	$\{0,1,3,5\}$	166.014
$\{0,1,3,6\}$	161.64	$\{0,1,3,7\}$	161.617	$\{0,1,3,8\}$	159.183
$\{0,1,3,9\}$	158.151	$\{0,1,4,5\}$	165.926	$\{0,1,4,6\}$	162.077
$\{0,1,4,7\}$	160.653	$\{0,1,4,8\}$	160.979	$\{0,1,4,9\}$	157.7
$\{0,1,5,6\}$	161.651	$\{0,1,5,7\}$	159.321	$\{0,1,5,8\}$	156.784
$\{0,1,5,9\}$	160.562	$\{0,1,6,7\}$	160.654	$\{0,1,6,8\}$	159.35
$\{0,1,6,9\}$	156.825	$\{0,1,7,8\}$	161.351	$\{0,1,7,9\}$	155.226
$\{0,1,8,9\}$	162.102	$\{0,2,3,4\}$	168.927	$\{0,2,3,5\}$	167.211
$\{0,2,3,6\}$	162.762	$\{0,2,3,7\}$	161.597	$\{0,2,3,8\}$	157.823
$\{0,2,3,9\}$	158.285	$\{0,2,4,5\}$	164.016	$\{0,2,4,6\}$	187.528
$\{0,2,4,7\}$	161.231	$\{0,2,4,8\}$	171.445	$\{0,2,4,9\}$	160.863
$\{0,2,5,6\}$	162.698	$\{0,2,5,7\}$	162.256	$\{0,2,5,8\}$	159.657
$\{0,2,5,9\}$	157.398	$\{0,2,6,7\}$	157.618	$\{0,2,6,8\}$	170.909
$\{0,2,6,9\}$	157.95	$\{0,2,7,8\}$	157.34	$\{0,2,7,9\}$	162.361
$\{0,2,8,9\}$	157.155	$\{0,3,4,5\}$	162.855	$\{0,3,4,6\}$	161.679
$\{0,3,4,7\}$	161.096	$\{0,3,4,8\}$	159.869	$\{0,3,4,9\}$	157.441
$\{0,3,5,6\}$	162.495	$\{0,3,5,7\}$	161.856	$\{0,3,5,8\}$	163.453
$\{0,3,5,9\}$	160.445	$\{0,3,6,7\}$	160.173	$\{0,3,6,8\}$	159.709
$\{0,3,6,9\}$	189.594	$\{0,3,7,8\}$	156.207	$\{0,3,7,9\}$	155.58
$\{0,3,8,9\}$	157.671	$\{0,4,5,6\}$	161.927	$\{0,4,5,7\}$	160.075
$\{0,4,5,8\}$	159.913	$\{0,4,5,9\}$	161.776	$\{0,4,6,7\}$	159.51
$\{0,4,6,8\}$	170.551	$\{0,4,6,9\}$	157.099	$\{0,4,7,8\}$	159.225
$\{0,4,7,9\}$	157.198	$\{0,4,8,9\}$	159.043	$\{0,5,6,7\}$	159.876
$\{0,5,6,8\}$	158.2	$\{0,5,6,9\}$	157.456	$\{0,5,7,8\}$	159.701
$\{0,5,7,9\}$	160.75	$\{0,5,8,9\}$	156.328	$\{0,6,7,8\}$	160.398
$\{0,6,7,9\}$	160.517	$\{0,6,8,9\}$	156.992	$\{0,7,8,9\}$	159.762

doit effectuer beaucoup moins de tests que sur $\{0, 1, 2, 3\}$: pour construire un mot sans cube additif de taille 10^5 il effectue 183584 essais sur $\{0, 19, 38, 58\}$ contre 275229 sur $\{0, 1, 2, 3\}$. Ceci est vraisemblablement dû au fait qu'il existe de nombreuses relations additives dans cet alphabet, $0 + 3 = 1 + 2$, $1 + 1 = 2 + 0$, $1 + 2 = 3 + 0$,...

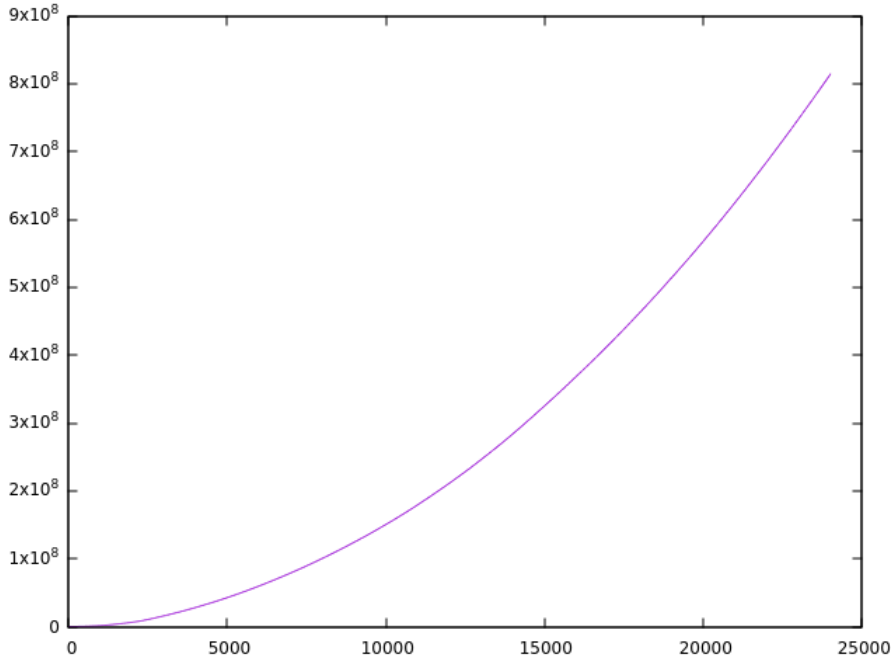
Cependant notre programme reste beaucoup plus efficace qu'une méthode classique (celle de la Section 6.2.4.1). Si on effectue la même recherche de mots sans cube additif sur $\{0, 1, 2, 3\}$ mais sans inverser périodiquement l'ordre de priorité des lettres, le programme va chercher toujours à rajouter en priorité un 0, puis un 1, puis un 2... Il construit ainsi

des suites très répétitives et lorsqu'il ne peut plus ajouter de lettre il doit parfois revenir très loin en arrière dans le mot pour trouver un préfixe prolongeable. Par exemple, cette approche directe donne un mot sans cube additif de longueur 24396 en moins d'une demi seconde avec une moyenne de 5 lettres testées pour pouvoir en ajouter une. En revanche il faut deux heures supplémentaires et une moyenne de 137188 lettres testées pour une placée pour obtenir un mot de longueur 24397. Un mot de la même taille avec notre programme *Up and Down* est obtenu en 0.12 secondes sur le même ordinateur.

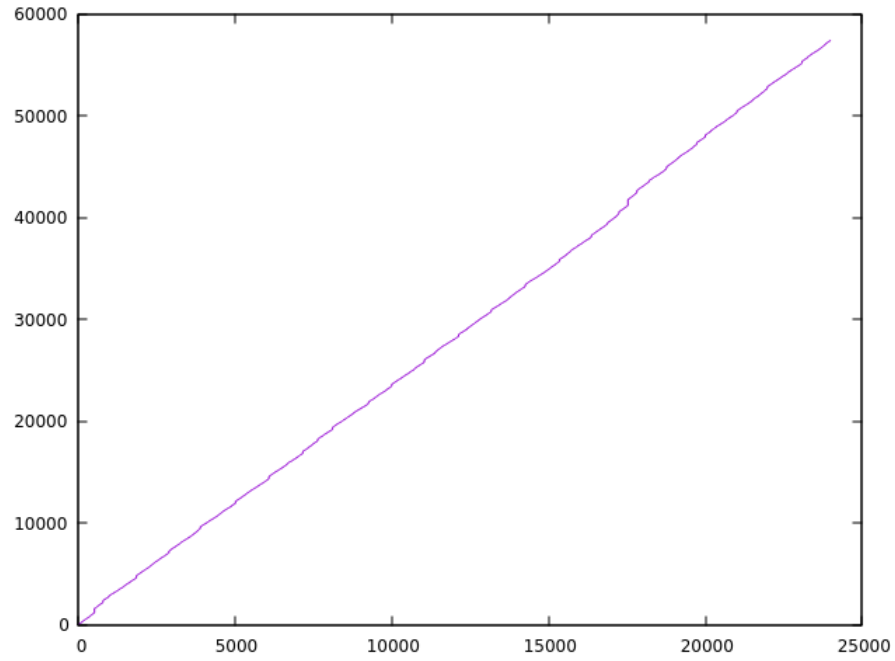
Enfin, le nombre de lettres testées pour construire un mot nous semble être un outil de mesure pertinent : il ne dépend ni de l'architecture de l'ordinateur ni des autres logiciels qui pourraient interférer et ralentir la vitesse de calcul. Les courbes de la Figure 6.8 montrent, pour chacun des deux programmes, le nombre cumulé de lettres testées pour construire un mot sur $\{0, 1, 2, 3\}$ de taille n sans cube additif, en fonction de n.

Dans le cas du programme classique, non seulement le nombre de lettres testées est beaucoup plus important en cumulé pour une même taille générée (9×10^9 pour le programme classique, 6×10^4 pour le nôtre), mais en plus ce nombre semble croître de façon au moins quadratique. Dans le programme *Up and Down*, le nombre de lettres testées reste une fonction linéaire de la taille du mot généré.

Néanmoins, nous atteignons avec le mot de 70×10^6 lettres les limites de notre programmation *Up and Down*. Obtenir ce mot nous a demandé plusieurs semaines et le temps de calcul augmente avec la longueur (car il faut tester l'absence de cubes additifs dans des mots de plus en plus longs). On pourrait améliorer ce programme en trouvant un argument permettant de tester moins de lettres ou en utilisant un schéma de construction différent d'une alternance de parties croissantes et de parties décroissantes.



(a) Programme classique.



(b) *Up and Down.*

Figure 6.8 – Comparaison en fonction des programmes du nombre cumulé de lettres testées pour construire un mot de taille 24000 sans cube additif sur $\{0, 1, 2, 3\}$.

Chapitre 7

Conclusion

L'évitabilité des puissances additives et la complexité additive sont deux sujets qui donnent lieu à une recherche active en combinatoire des mots ces dernières années. Dans le Chapitre 1 nous avons expliqué le contexte dans lequel ont émergé ces questions et présenté certaines généralisations du problème des puissances additives ainsi que des réponses partielles qui y ont été apportées. Le résultat de J. Cassaigne et al. [16] a été le point de départ de notre recherche.

Nous avons expliqué notre démarche de recherche dans le Chapitre 2 et la façon dont nous avons conçu le programme *Looking4Morphisms*. Ce programme parallélisé utilise l'infrastructure de calcul Grid5000 pour trouver des morphismes candidats, c'est à dire des morphismes possédant un point fixe infini qui ne contient pas de cubes additifs dans ses 10000 premières lettres. Nous observons que sur tous les alphabets d'entiers $\{a, b, c, d\}$ avec $0 \leq a < b < c < d < 100$, un morphisme qui vérifie la condition Cupisca est nécessairement substitution de φ_0 (la condition Cupisca est définie page 17). Ceci nous amène à poser la question suivante.

Question 1. *Un morphisme de taille 2 vérifiant la condition Cupisca est-il nécessairement substitution de φ_0 ?*

Nous rappelons que les tableaux donnés entre les pages 107 et 111 donnent les 288 morphismes de taille 2 dont les préfixes ne contiennent pas de cubes abéliens dans les 10000 premières lettres. Un morphisme de taille 2 vérifiant la condition Cupisca appartient nécessairement à cette liste mais seuls 24 d'entre eux sont substitution de φ_0. Étant donné que les morphismes de taille 3 que nous connaissons qui vérifient la condition Cupisca (voir Section 2.3.1) sont similaires à φ_0 ou à φ_0^2, nous posons une question plus générale.

Question 2. *Si φ est un morphisme vérifiant la condition Cupisca existe-t-il toujours des entiers n et k tels que φ^n soit similaire à φ_0^k ?*

Dans le Chapitre 3 et le Chapitre 4 nous développons un algorithme qui permet de prouver le Théorème 2.3.1.

Théorème 2.3.1. *Soit φ un morphisme de taille 2 similaire à φ_0, alors on peut décider algorithmiquement s'il vérifie la condition* Cupisca*.*

Cependant cet algorithme peut aussi être utilisé dans certains cas pour des morphismes qui ne sont pas de taille 2 comme nous l'avons vu en Section 4.6. M. Rao et M. Rosenfeld [39] ont fournit un algorithme plus général mais laissent ouverte la question suivante.

Question 3. *Peut-on toujours décider si un morphisme vérifie la condition* Cupisca *?*

Ils posent en fait une question plus générale ([39, Problème 2]) où ils remplacent la somme des puissances additives par une forme linéaire sur le monoïde libre engendré par l'alphabet.

Nous entreprenons dans le Chapitre 5 de caractériser les sous-ensembles de $\mathbb{N}$ sur lesquels les cubes additifs sont évitables et nous démontrons le résultat suivant.

Corollaire 5.5.5. *Soit $\mathcal{A} \subset \mathbb{C}$ un alphabet vérifiant $|\mathcal{A}| \geq 4$. Si $\mathcal{A}$ n'est pas équivalent à $\{0,1,2,3\}$ alors les cubes additifs sont évitables sur $\mathcal{A}$. En particulier si $|\mathcal{A}| \geq 5$ alors les cubes additifs sont évitables sur $\mathcal{A}$.*

Nous n'apportons pas de réponse pour les alphabets équivalents à $\{0,1,2,3\}$ et nous laissons ouverte la question de savoir si les cubes additifs sont évitables sur ces alphabets (question déjà posée en 2018 par Rao et Rosenfeld [39, Problème7]).

Néanmoins, dans le Chapitre 6 nous démontrons :

Proposition 6.4.1. *Il existe un mot de longueur* 70880000 *sur l'alphabet* $\{0,1,2,3\}$ *sans cube additif.*

Nous utilisons pour parvenir à ce résultat une programmation en C++ avec un test de la présence de cubes additifs parallélisé grâce au modèle de programmation parallèle OpenMP. Notre approche, d'abord graphique, nous permet d'améliorer nettement la précédente borne connue (1.4×10^5) et les performances des méthodes de construction directes par parcours d'arbre en profondeur (voir Section 6.4.2). Cependant, notre approche est limitée car il nous faut déjà plusieurs semaines pour obtenir le résultat de la Proposition 6.4.1.

Question 4. *Quelle approche différente de la technique* Up and Down *permettrait d'améliorer significativement la taille du plus long mot connu sans cubes additifs sur* $\{0,1,2,3\}$ *?*

Enfin, la question de l'évitabilité des carrés additifs reste ouverte.

Annexes

1 Morphismes candidats pour éviter les cubes abéliens

Les 5 tableaux suivants répertorient les 288 morphismes de taille 2 qui possèdent un point fixe infini sans cube abélien dans ses 10000 premiers termes (cités en Section 2.3.3).

Tableau A.1 – Morphismes de taille 2 sur $\{a, b, c, d\}$ possédant un point fixe infini dont le préfixe de longueur 10000 évite les cubes abéliens (tableau 1/5).

$a \mapsto dd, b \mapsto cc, c \mapsto ca, d \mapsto bd$	$a \mapsto dd, b \mapsto cc, c \mapsto ac, d \mapsto db$
$a \mapsto dd, b \mapsto cb, c \mapsto ca, d \mapsto bd$	$a \mapsto dd, b \mapsto cb, c \mapsto ca, d \mapsto ba$
$a \mapsto dd, b \mapsto cb, c \mapsto ca, d \mapsto ab$	$a \mapsto dd, b \mapsto ca, c \mapsto cb, d \mapsto ba$
$a \mapsto dd, b \mapsto bc, c \mapsto ba, d \mapsto ca$	$a \mapsto dd, b \mapsto bc, c \mapsto ac, d \mapsto db$
$a \mapsto dd, b \mapsto bc, c \mapsto ac, d \mapsto db$	$a \mapsto dd, b \mapsto bc, c \mapsto ac, d \mapsto ba$
$a \mapsto dd, b \mapsto bc, c \mapsto ac, d \mapsto ab$	$a \mapsto dd, b \mapsto ba, c \mapsto bc, d \mapsto cd$
$a \mapsto dd, b \mapsto ba, c \mapsto bc, d \mapsto ca$	$a \mapsto dd, b \mapsto ba, c \mapsto bc, d \mapsto ac$
$a \mapsto dd, b \mapsto ba, c \mapsto bb, d \mapsto cd$	$a \mapsto dd, b \mapsto ab, c \mapsto cb, d \mapsto dc$
$a \mapsto dd, b \mapsto ab, c \mapsto cb, d \mapsto dc$	$a \mapsto dd, b \mapsto ab, c \mapsto cb, d \mapsto ca$
$a \mapsto dd, b \mapsto ab, c \mapsto cb, d \mapsto ac$	$a \mapsto dd, b \mapsto ab, c \mapsto bb, d \mapsto dc$
$a \mapsto dc, b \mapsto db, c \mapsto cb, d \mapsto aa$	$a \mapsto dc, b \mapsto cb, c \mapsto aa, d \mapsto db$
$a \mapsto dc, b \mapsto c, c \mapsto ab, d \mapsto db$	$a \mapsto dc, b \mapsto bd, c \mapsto bc, d \mapsto aa$
$a \mapsto dc, b \mapsto bd, c \mapsto aa, d \mapsto bc$	$a \mapsto dc, b \mapsto bc, c \mapsto aa, d \mapsto bd$
$a \mapsto dc, b \mapsto bc, c \mapsto a, d \mapsto ba$	$a \mapsto dc, b \mapsto ac, c \mapsto bb, d \mapsto da$
$a \mapsto dc, b \mapsto a, c \mapsto b, d \mapsto db$	$a \mapsto db, b \mapsto cc, c \mapsto ab, d \mapsto da$
$a \mapsto db, b \mapsto cb, c \mapsto cd, d \mapsto aa$	$a \mapsto db, b \mapsto c, c \mapsto a, d \mapsto dc$
$a \mapsto db, b \mapsto bc, c \mapsto dc, d \mapsto aa$	$a \mapsto db, b \mapsto ac, c \mapsto b, d \mapsto dc$
$a \mapsto db, b \mapsto aa, c \mapsto cd, d \mapsto cb$	$a \mapsto db, b \mapsto aa, c \mapsto cb, d \mapsto cd$
$a \mapsto db, b \mapsto aa, c \mapsto bc, d \mapsto dc$	$a \mapsto db, b \mapsto a, c \mapsto cb, d \mapsto ca$
$a \mapsto da, b \mapsto dc, c \mapsto ca, d \mapsto bb$	$a \mapsto da, b \mapsto cd, c \mapsto ca, d \mapsto bb$
$a \mapsto da, b \mapsto cc, c \mapsto ba, d \mapsto db$	$a \mapsto da, b \mapsto cc, c \mapsto ac, d \mapsto db$
$a \mapsto da, b \mapsto cc, c \mapsto ab, d \mapsto db$	$a \mapsto da, b \mapsto ca, c \mapsto bb, d \mapsto dc$
$a \mapsto da, b \mapsto bc, c \mapsto ca, d \mapsto bb$	$a \mapsto da, b \mapsto bc, c \mapsto ca, d \mapsto bb$
$a \mapsto da, b \mapsto bc, c \mapsto aa, d \mapsto bd$	$a \mapsto da, b \mapsto bc, c \mapsto aa, d \mapsto bb$

Tableau A.2 – Morphismes de taille 2 sur $\{a, b, c, d\}$ possédant un point fixe infini dont le préfixe de longueur 10000 évite les cubes abéliens (tableau 2/5).

$a \mapsto da, b \mapsto ba, c \mapsto db, d \mapsto cc$	$a \mapsto da, b \mapsto ba, c \mapsto cb, d \mapsto cc$
$a \mapsto da, b \mapsto ba, c \mapsto cb, d \mapsto cc$	$a \mapsto da, b \mapsto ba, c \mapsto bd, d \mapsto cc$
$a \mapsto da, b \mapsto ac, c \mapsto bb, d \mapsto dc$	$a \mapsto da, b \mapsto ab, c \mapsto bb, d \mapsto dc$
$a \mapsto da, b \mapsto aa, c \mapsto cb, d \mapsto cd$	$a \mapsto da, b \mapsto aa, c \mapsto cb, d \mapsto cc$
$a \mapsto d, b \mapsto cd, c \mapsto ca, d \mapsto ba$	$a \mapsto d, b \mapsto ca, c \mapsto cd, d \mapsto b$
$a \mapsto d, b \mapsto bd, c \mapsto ba, d \mapsto c$	$a \mapsto d, b \mapsto ba, c \mapsto bd, d \mapsto ca$
$a \mapsto d, b \mapsto ba, c \mapsto a, d \mapsto bc$	$a \mapsto d, b \mapsto a, c \mapsto ca, d \mapsto cb$
$a \mapsto cd, b \mapsto db, c \mapsto cb, d \mapsto aa$	$a \mapsto cd, b \mapsto d, c \mapsto cb, d \mapsto ab$
$a \mapsto cd, b \mapsto cb, c \mapsto aa, d \mapsto db$	$a \mapsto cd, b \mapsto bd, c \mapsto bc, d \mapsto aa$
$a \mapsto cd, b \mapsto bd, c \mapsto ba, d \mapsto a$	$a \mapsto cd, b \mapsto bc, c \mapsto bd, d \mapsto aa$
$a \mapsto cd, b \mapsto bc, c \mapsto aa, d \mapsto bd$	$a \mapsto cd, b \mapsto ad, c \mapsto ca, d \mapsto bb$
$a \mapsto cd, b \mapsto a, c \mapsto cb, d \mapsto b$	$a \mapsto cc, b \mapsto dd, c \mapsto cb, d \mapsto ad$
$a \mapsto cc, b \mapsto dd, c \mapsto bc, d \mapsto da$	$a \mapsto cc, b \mapsto db, c \mapsto bc, d \mapsto da$
$a \mapsto cc, b \mapsto db, c \mapsto ba, d \mapsto da$	$a \mapsto cc, b \mapsto db, c \mapsto ab, d \mapsto da$
$a \mapsto cc, b \mapsto da, c \mapsto ba, d \mapsto db$	$a \mapsto cc, b \mapsto bd, c \mapsto da, d \mapsto ba$
$a \mapsto cc, b \mapsto bd, c \mapsto cb, d \mapsto ad$	$a \mapsto cc, b \mapsto bd, c \mapsto cb, d \mapsto ad$
$a \mapsto cc, b \mapsto bd, c \mapsto ba, d \mapsto ad$	$a \mapsto cc, b \mapsto bd, c \mapsto ab, d \mapsto ad$
$a \mapsto cc, b \mapsto ba, c \mapsto dc, d \mapsto bd$	$a \mapsto cc, b \mapsto ba, c \mapsto dc, d \mapsto bb$
$a \mapsto cc, b \mapsto ba, c \mapsto da, d \mapsto bd$	$a \mapsto cc, b \mapsto ba, c \mapsto ad, d \mapsto bd$
$a \mapsto cc, b \mapsto ab, c \mapsto da, d \mapsto db$	$a \mapsto cc, b \mapsto ab, c \mapsto cd, d \mapsto db$
$a \mapsto cc, b \mapsto ab, c \mapsto cd, d \mapsto db$	$a \mapsto cc, b \mapsto ab, c \mapsto cd, d \mapsto bb$
$a \mapsto cc, b \mapsto ab, c \mapsto ad, d \mapsto db$	$a \mapsto cb, b \mapsto dd, c \mapsto ca, d \mapsto ab$
$a \mapsto cb, b \mapsto db, c \mapsto aa, d \mapsto dc$	$a \mapsto cb, b \mapsto d, c \mapsto cd, d \mapsto a$
$a \mapsto cb, b \mapsto bd, c \mapsto aa, d \mapsto cd$	$a \mapsto cb, b \mapsto ad, c \mapsto cd, d \mapsto b$
$a \mapsto cb, b \mapsto aa, c \mapsto dc, d \mapsto db$	$a \mapsto cb, b \mapsto aa, c \mapsto db, d \mapsto dc$
$a \mapsto cb, b \mapsto aa, c \mapsto cd, d \mapsto bd$	$a \mapsto cb, b \mapsto a, c \mapsto da, d \mapsto db$
$a \mapsto ca, b \mapsto dd, c \mapsto cb, d \mapsto ba$	$a \mapsto ca, b \mapsto dd, c \mapsto cb, d \mapsto ad$
$a \mapsto ca, b \mapsto dd, c \mapsto cb, d \mapsto ab$	$a \mapsto ca, b \mapsto dc, c \mapsto bb, d \mapsto da$
$a \mapsto ca, b \mapsto da, c \mapsto cd, d \mapsto bb$	$a \mapsto ca, b \mapsto cd, c \mapsto bb, d \mapsto da$
$a \mapsto ca, b \mapsto bd, c \mapsto bc, d \mapsto aa$	$a \mapsto ca, b \mapsto bd, c \mapsto bb, d \mapsto da$
$a \mapsto ca, b \mapsto bd, c \mapsto bb, d \mapsto da$	$a \mapsto ca, b \mapsto bd, c \mapsto bb, d \mapsto aa$
$a \mapsto ca, b \mapsto ba, c \mapsto dd, d \mapsto db$	$a \mapsto ca, b \mapsto ba, c \mapsto dd, d \mapsto db$
$a \mapsto ca, b \mapsto ba, c \mapsto dd, d \mapsto cb$	$a \mapsto ca, b \mapsto ba, c \mapsto dd, d \mapsto bc$
$a \mapsto ca, b \mapsto ad, c \mapsto cd, d \mapsto bb$	$a \mapsto ca, b \mapsto ab, c \mapsto cd, d \mapsto bb$
$a \mapsto ca, b \mapsto aa, c \mapsto dd, d \mapsto db$	$a \mapsto ca, b \mapsto aa, c \mapsto dc, d \mapsto db$
$a \mapsto c, b \mapsto dc, c \mapsto ba, d \mapsto da$	$a \mapsto c, b \mapsto da, c \mapsto b, d \mapsto dc$
$a \mapsto c, b \mapsto bc, c \mapsto d, d \mapsto ba$	$a \mapsto c, b \mapsto ba, c \mapsto da, d \mapsto bc$
$a \mapsto c, b \mapsto ba, c \mapsto bd, d \mapsto a$	$a \mapsto c, b \mapsto a, c \mapsto db, d \mapsto da$

Tableau A.3 – Morphismes de taille 2 sur $\{a, b, c, d\}$ possédant un point fixe infini dont le préfixe de longueur 10000 évite les cubes abéliens (tableau 3/5).

$a \mapsto bd, b \mapsto cd, c \mapsto cb, d \mapsto aa$	$a \mapsto bd, b \mapsto cb, c \mapsto cd, d \mapsto aa$
$a \mapsto bd, b \mapsto ca, c \mapsto cd, d \mapsto a$	$a \mapsto bd, b \mapsto bc, c \mapsto dc, d \mapsto aa$
$a \mapsto bd, b \mapsto bc, c \mapsto d, d \mapsto ac$	$a \mapsto bd, b \mapsto bc, c \mapsto a, d \mapsto c$
$a \mapsto bd, b \mapsto ba, c \mapsto ad, d \mapsto cc$	$a \mapsto bd, b \mapsto aa, c \mapsto cb, d \mapsto cd$
$a \mapsto bd, b \mapsto aa, c \mapsto bc, d \mapsto dc$	$a \mapsto bc, b \mapsto dc, c \mapsto aa, d \mapsto db$
$a \mapsto bc, b \mapsto db, c \mapsto aa, d \mapsto dc$	$a \mapsto bc, b \mapsto da, c \mapsto a, d \mapsto dc$
$a \mapsto bc, b \mapsto bd, c \mapsto d, d \mapsto a$	$a \mapsto bc, b \mapsto bd, c \mapsto ad, d \mapsto c$
$a \mapsto bc, b \mapsto bd, c \mapsto aa, d \mapsto cd$	$a \mapsto bc, b \mapsto ba, c \mapsto dd, d \mapsto ac$
$a \mapsto bc, b \mapsto aa, c \mapsto dc, d \mapsto db$	$a \mapsto bc, b \mapsto aa, c \mapsto cd, d \mapsto bd$
$a \mapsto bb, b \mapsto db, c \mapsto ca, d \mapsto cd$	$a \mapsto bb, b \mapsto db, c \mapsto ca, d \mapsto cc$
$a \mapsto bb, b \mapsto da, c \mapsto cd, d \mapsto ca$	$a \mapsto bb, b \mapsto da, c \mapsto ca, d \mapsto cd$
$a \mapsto bb, b \mapsto da, c \mapsto ac, d \mapsto dc$	$a \mapsto bb, b \mapsto cb, c \mapsto dd, d \mapsto da$
$a \mapsto bb, b \mapsto cb, c \mapsto dc, d \mapsto da$	$a \mapsto bb, b \mapsto ca, c \mapsto dc, d \mapsto da$
$a \mapsto bb, b \mapsto ca, c \mapsto da, d \mapsto dc$	$a \mapsto bb, b \mapsto ca, c \mapsto cd, d \mapsto ad$
$a \mapsto bb, b \mapsto bd, c \mapsto ac, d \mapsto dc$	$a \mapsto bb, b \mapsto bd, c \mapsto ac, d \mapsto dc$
$a \mapsto bb, b \mapsto bd, c \mapsto ac, d \mapsto cc$	$a \mapsto bb, b \mapsto bc, c \mapsto dd, d \mapsto ad$
$a \mapsto bb, b \mapsto bc, c \mapsto cd, d \mapsto ad$	$a \mapsto bb, b \mapsto bc, c \mapsto cd, d \mapsto ad$
$a \mapsto bb, b \mapsto ad, c \mapsto ca, d \mapsto cd$	$a \mapsto bb, b \mapsto ad, c \mapsto ac, d \mapsto dc$
$a \mapsto bb, b \mapsto ac, c \mapsto dc, d \mapsto da$	$a \mapsto bb, b \mapsto ac, c \mapsto cd, d \mapsto ad$
$a \mapsto ba, b \mapsto dd, c \mapsto ca, d \mapsto dc$	$a \mapsto ba, b \mapsto dd, c \mapsto ca, d \mapsto dc$
$a \mapsto ba, b \mapsto dd, c \mapsto ca, d \mapsto cb$	$a \mapsto ba, b \mapsto dd, c \mapsto ca, d \mapsto bc$
$a \mapsto ba, b \mapsto dd, c \mapsto aa, d \mapsto dc$	$a \mapsto ba, b \mapsto db, c \mapsto aa, d \mapsto dc$
$a \mapsto ba, b \mapsto cc, c \mapsto db, d \mapsto da$	$a \mapsto ba, b \mapsto cc, c \mapsto cd, d \mapsto da$
$a \mapsto ba, b \mapsto cc, c \mapsto cd, d \mapsto da$	$a \mapsto ba, b \mapsto cc, c \mapsto cd, d \mapsto aa$
$a \mapsto ba, b \mapsto cc, c \mapsto bd, d \mapsto da$	$a \mapsto ba, b \mapsto cb, c \mapsto cd, d \mapsto aa$
$a \mapsto ba, b \mapsto bd, c \mapsto da, d \mapsto cc$	$a \mapsto ba, b \mapsto bd, c \mapsto ad, d \mapsto cc$
$a \mapsto ba, b \mapsto bd, c \mapsto ac, d \mapsto cc$	$a \mapsto ba, b \mapsto bc, c \mapsto dd, d \mapsto ca$
$a \mapsto ba, b \mapsto bc, c \mapsto dd, d \mapsto ad$	$a \mapsto ba, b \mapsto bc, c \mapsto dd, d \mapsto ac$
$a \mapsto b, b \mapsto dc, c \mapsto a, d \mapsto da$	$a \mapsto b, b \mapsto da, c \mapsto ca, d \mapsto cb$
$a \mapsto b, b \mapsto d, c \mapsto cb, d \mapsto ca$	$a \mapsto b, b \mapsto cd, c \mapsto ca, d \mapsto a$
$a \mapsto b, b \mapsto ca, c \mapsto db, d \mapsto da$	$a \mapsto b, b \mapsto c, c \mapsto da, d \mapsto db$
$a \mapsto ad, b \mapsto dc, c \mapsto bb, d \mapsto ac$	$a \mapsto ad, b \mapsto dc, c \mapsto ac, d \mapsto bb$
$a \mapsto ad, b \mapsto d, c \mapsto ab, d \mapsto c$	$a \mapsto ad, b \mapsto cd, c \mapsto ac, d \mapsto bb$
$a \mapsto ad, b \mapsto cd, c \mapsto ab, d \mapsto b$	$a \mapsto ad, b \mapsto cc, c \mapsto db, d \mapsto ab$
$a \mapsto ad, b \mapsto cc, c \mapsto ca, d \mapsto bd$	$a \mapsto ad, b \mapsto cc, c \mapsto ca, d \mapsto bd$
$a \mapsto ad, b \mapsto cc, c \mapsto ba, d \mapsto bd$	$a \mapsto ad, b \mapsto cc, c \mapsto ab, d \mapsto bd$
$a \mapsto ad, b \mapsto cb, c \mapsto ac, d \mapsto bb$	$a \mapsto ad, b \mapsto cb, c \mapsto aa, d \mapsto db$
$a \mapsto ad, b \mapsto cb, c \mapsto aa, d \mapsto db$	$a \mapsto ad, b \mapsto cb, c \mapsto aa, d \mapsto bb$

Tableau A.4 – Morphismes de taille 2 sur $\{a, b, c, d\}$ possédant un point fixe infini dont le préfixe de longueur 10000 évite les cubes abéliens (tableau 4/5).

$a \mapsto ad, b \mapsto ca, c \mapsto bb, d \mapsto cd$	$a \mapsto ad, b \mapsto bc, c \mapsto ca, d \mapsto db$
$a \mapsto ad, b \mapsto bc, c \mapsto ca, d \mapsto db$	$a \mapsto ad, b \mapsto bc, c \mapsto ca, d \mapsto db$
$a \mapsto ad, b \mapsto bc, c \mapsto ca, d \mapsto db$	$a \mapsto ad, b \mapsto ba, c \mapsto cb, d \mapsto dc$
$a \mapsto ad, b \mapsto ba, c \mapsto cb, d \mapsto dc$	$a \mapsto ad, b \mapsto ba, c \mapsto cb, d \mapsto dc$
$a \mapsto ad, b \mapsto ba, c \mapsto cb, d \mapsto dc$	$a \mapsto ad, b \mapsto ba, c \mapsto bb, d \mapsto cd$
$a \mapsto ad, b \mapsto ba, c \mapsto bb, d \mapsto cd$	$a \mapsto ad, b \mapsto ac, c \mapsto d, d \mapsto b$
$a \mapsto ad, b \mapsto ac, c \mapsto bd, d \mapsto c$	$a \mapsto ad, b \mapsto ac, c \mapsto bb, d \mapsto cd$
$a \mapsto ad, b \mapsto ab, c \mapsto db, d \mapsto cc$	$a \mapsto ad, b \mapsto ab, c \mapsto bd, d \mapsto cc$
$a \mapsto ad, b \mapsto ab, c \mapsto bc, d \mapsto cc$	$a \mapsto ad, b \mapsto aa, c \mapsto bc, d \mapsto dc$
$a \mapsto ad, b \mapsto aa, c \mapsto bc, d \mapsto dc$	$a \mapsto ad, b \mapsto aa, c \mapsto bc, d \mapsto cc$
$a \mapsto ac, b \mapsto dd, c \mapsto bc, d \mapsto da$	$a \mapsto ac, b \mapsto dd, c \mapsto bc, d \mapsto da$
$a \mapsto ac, b \mapsto dd, c \mapsto bc, d \mapsto ba$	$a \mapsto ac, b \mapsto dd, c \mapsto bc, d \mapsto ab$
$a \mapsto ac, b \mapsto dd, c \mapsto ab, d \mapsto cb$	$a \mapsto ac, b \mapsto dc, c \mapsto bb, d \mapsto ad$
$a \mapsto ac, b \mapsto dc, c \mapsto b, d \mapsto ab$	$a \mapsto ac, b \mapsto db, c \mapsto cb, d \mapsto aa$
$a \mapsto ac, b \mapsto db, c \mapsto cb, d \mapsto aa$	$a \mapsto ac, b \mapsto db, c \mapsto bb, d \mapsto ad$
$a \mapsto ac, b \mapsto db, c \mapsto bb, d \mapsto aa$	$a \mapsto ac, b \mapsto da, c \mapsto dc, d \mapsto bb$
$a \mapsto ac, b \mapsto cd, c \mapsto bb, d \mapsto ad$	$a \mapsto ac, b \mapsto cd, c \mapsto ad, d \mapsto bb$
$a \mapsto ac, b \mapsto c, c \mapsto d, d \mapsto ab$	$a \mapsto ac, b \mapsto bd, c \mapsto cb, d \mapsto da$
$a \mapsto ac, b \mapsto bd, c \mapsto cb, d \mapsto da$	$a \mapsto ac, b \mapsto bd, c \mapsto cb, d \mapsto da$
$a \mapsto ac, b \mapsto bd, c \mapsto cb, d \mapsto da$	$a \mapsto ac, b \mapsto ba, c \mapsto dc, d \mapsto bb$
$a \mapsto ac, b \mapsto ba, c \mapsto dc, d \mapsto bb$	$a \mapsto ac, b \mapsto ba, c \mapsto cd, d \mapsto db$
$a \mapsto ac, b \mapsto ba, c \mapsto cd, d \mapsto db$	$a \mapsto ac, b \mapsto ba, c \mapsto cd, d \mapsto db$
$a \mapsto ac, b \mapsto ba, c \mapsto cd, d \mapsto db$	$a \mapsto ac, b \mapsto ad, c \mapsto dc, d \mapsto bb$
$a \mapsto ac, b \mapsto ad, c \mapsto d, d \mapsto bc$	$a \mapsto ac, b \mapsto ad, c \mapsto b, d \mapsto c$
$a \mapsto ac, b \mapsto ab, c \mapsto dd, d \mapsto cb$	$a \mapsto ac, b \mapsto ab, c \mapsto dd, d \mapsto bd$
$a \mapsto ac, b \mapsto ab, c \mapsto dd, d \mapsto bc$	$a \mapsto ac, b \mapsto aa, c \mapsto dd, d \mapsto bd$
$a \mapsto ac, b \mapsto aa, c \mapsto cd, d \mapsto bd$	$a \mapsto ac, b \mapsto aa, c \mapsto cd, d \mapsto bd$
$a \mapsto ab, b \mapsto dd, c \mapsto ac, d \mapsto cd$	$a \mapsto ab, b \mapsto dd, c \mapsto ac, d \mapsto cb$
$a \mapsto ab, b \mapsto dd, c \mapsto ac, d \mapsto bc$	$a \mapsto ab, b \mapsto dd, c \mapsto aa, d \mapsto cd$
$a \mapsto ab, b \mapsto db, c \mapsto da, d \mapsto cc$	$a \mapsto ab, b \mapsto db, c \mapsto ca, d \mapsto cc$
$a \mapsto ab, b \mapsto db, c \mapsto ca, d \mapsto cc$	$a \mapsto ab, b \mapsto db, c \mapsto ad, d \mapsto cc$
$a \mapsto ab, b \mapsto d, c \mapsto b, d \mapsto ac$	$a \mapsto ab, b \mapsto d, c \mapsto ad, d \mapsto cb$
$a \mapsto ab, b \mapsto cc, c \mapsto dc, d \mapsto ad$	$a \mapsto ab, b \mapsto cc, c \mapsto dc, d \mapsto aa$
$a \mapsto ab, b \mapsto cc, c \mapsto db, d \mapsto ad$	$a \mapsto ab, b \mapsto cc, c \mapsto bd, d \mapsto ad$
$a \mapsto ab, b \mapsto cb, c \mapsto dd, d \mapsto da$	$a \mapsto ab, b \mapsto cb, c \mapsto dd, d \mapsto da$
$a \mapsto ab, b \mapsto cb, c \mapsto dd, d \mapsto ca$	$a \mapsto ab, b \mapsto cb, c \mapsto dd, d \mapsto ac$
$a \mapsto ab, b \mapsto c, c \mapsto db, d \mapsto ac$	$a \mapsto ab, b \mapsto c, c \mapsto ad, d \mapsto b$
$a \mapsto ab, b \mapsto bd, c \mapsto ca, d \mapsto dc$	$a \mapsto ab, b \mapsto bd, c \mapsto ca, d \mapsto dc$

Tableau A.5 – Morphismes de taille 2 sur $\{a, b, c, d\}$ possédant un point fixe infini dont le préfixe de longueur 10000 évite les cubes abéliens (tableau 5/5).

$a \mapsto ab, b \mapsto bd, c \mapsto ca, d \mapsto dc$	$a \mapsto ab, b \mapsto bd, c \mapsto ca, d \mapsto dc$
$a \mapsto ab, b \mapsto bd, c \mapsto aa, d \mapsto cd$	$a \mapsto ab, b \mapsto bd, c \mapsto aa, d \mapsto cd$
$a \mapsto ab, b \mapsto bc, c \mapsto dc, d \mapsto aa$	$a \mapsto ab, b \mapsto bc, c \mapsto dc, d \mapsto aa$
$a \mapsto ab, b \mapsto bc, c \mapsto cd, d \mapsto da$	$a \mapsto ab, b \mapsto bc, c \mapsto cd, d \mapsto da$
$a \mapsto ab, b \mapsto bc, c \mapsto cd, d \mapsto da$	$a \mapsto ab, b \mapsto bc, c \mapsto cd, d \mapsto da$
$a \mapsto ab, b \mapsto ad, c \mapsto bd, d \mapsto cc$	$a \mapsto ab, b \mapsto ac, c \mapsto dd, d \mapsto bc$

Index

Bibliographie

[1] A. Aberkane, J.D. Currie, and N. Rampersad. The number of ternary words avoiding abelian cubes grows exponentially. *J. Integer Seq*, 7(2), 2004.

[2] J.P. Allouche and J.O. Shallit. The ubiquitous Prouhet–Thue–Morse sequence. In C. Ding, T. Helleseth, and H. Niederreiter, editors, *Sequences and their Applications*, pages 1–16, London, 1999. Springer London.

[3] H. Ardal, T. Brown, V. Jungić, and J. Sahasrabudhe. On abelian and additive complexity in infinite words. *Integers*, 12(5) :795–804, 2012.

[4] Y.H. Au, A. Robertson, and J.O. Shallit. Van der Waerden's theorem and avoidability in words. *Integers*, 11(1) :61–76, 2011.

[5] G. Banero. On additive complexity of infinite words. *Journal of Integer Sequences*, 16, 2013.

[6] F.A. Behrend. On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences of the United States of America*, 32(12) :331–332, 1946.

[7] J. Berstel. Sur les mots sans carré définis par un morphisme. In Hermann A. Maurer, editor, *Automata, Languages and Programming*, pages 16–25, Berlin, Heidelberg, 1979. Springer Berlin Heidelberg.

[8] J. Berstel. Some recent results on squarefree words. In *Proceedings of the Symposium of Theoretical Aspects of Computer Science*, STACS '84, pages 14–25, Berlin, Heidelberg, 1984. Springer-Verlag.

[9] J. Berstel. *Axel Thue's papers on repetitions in words : a translation*, volume 20. Départements de mathématiques et d'informatique, Université du Québec à Montréal, 1995.

[10] J. Berstel and D. Perrin. The origins of combinatorics on words. *European Journal of Combinatorics*, 28(3) :996–1022, 2007.

[11] T. Brown. Approximations of additive squares in infinite words. *Integers : The Electronic Journal of Combinatorial Number Theory*, 5(12) :805–809, 2012.

[12] T. Brown, V. Jungić, and A. Polestra. On double 3-term arithmetic progressions. *Integers : The Electronic Journal of Combinatorial Number Theory*, #43(14), 2014.

[13] T.C. Brown. Is there a sequence on four symbols in which no two adjacent segments are permutations of one another ? *The American Mathematical Monthly*, 78(8) :886–888, 1971.

[14] T.C. Brown and A.R. Freedman. Arithmetic progressions in lacunary sets. *The Rocky Mountain Journal of Mathematics*, 17(3) :587–596, 1987.

[15] A. Carpi. On abelian power-free morphisms. *International Journal of Algebra and Computation*, 03, 11 2011.

[16] J. Cassaigne, J.D. Currie, L. Schaeffer, and J.O. Shallit. Avoiding three consecutive blocks of the same size and same sum. *Journal of the ACM*, 61(2) :1–17, 2014.

[17] J. Cassaigne, G. Richomme, K. Saari, and L.Q. Zamboni. Avoiding Abelian Powers in Binary Words with Bounded Abelian Complexity. *International Journal of Foundations of Computer Science*, 22(4) :905–920, 2011.

[18] J.D. Currie and A. Aberkane. A cyclic binary morphism avoiding abelian fourth powers. *Theoretical Computer Science*, 410(1) :44–52, 2009.

[19] J.D. Currie and N. Rampersad. Fixed points avoiding abelian k-powers. *Journal of Combinatorial Theory, Series A*, 119(5) :942–948, 2012.

[20] F.M. Dekking. Strongly non-repetitive sequences and progression-free sets. *Journal of Combinatorial Theory, Series A*, 27(2) :181–185, 1979.

[21] C.F. Du, H. Mousavi, E. Rowland, L. Schaeffer, and J.O. Shallit. Decision algorithms for fibonacci-automatic words, ii : Related sequences and avoidability. *Theoretical Computer Science*, 657 :146–162, 2017.

[22] R.C. Entringer, D.E. Jackson, and J.A. Schatz. On nonrepetitive sequences. *Journal of Combinatorial Theory. Series A*, 16(2) :159–164, 1974.

[23] P. Erdős. Some unsolved problems. *Michigan Mathematical Journal*, 4(3) :291–300, 1957.

[24] P. Erdős. Some unsolved problems. *Magyar Tud. Akad. Mat. Kutató Int. Közl*, 6(221-254) :160, 1961.

[25] A. A. Evdokimov. Strongly asymmetric sequences generated by a finite number of symbols. *Dokl. Akad. Nauk SSSR*, 179 :1268–1271, 1968.

[26] A.R. Freedman and T.C. Brown. Sequences on sets of four numbers. *Integers*, 16 :Paper No. A33, 10, 2016.

[27] L. Halbeisen and N. Hungerbühler. An application of Van der Waerden's theorem in additive number theory. *Integers : The Electronic Journal of Combinatorial Number Theory*, 0(A07), 2000.

[28] J. Justin. Characterization of the repetitive commutative semigroups. *Journal of Algebra*, 21(1) :87–90, 1972.

[29] J. Justin. Généralisation du théorème de van der Waerden sur les semi-groupes répétitifs. *Journal of Combinatorial Theory, Series A*, 12(3) :357–367, 1972.

[30] J. Justin and G. Pirillo. Repetitivity of semigroups and a result of cassaigne, currie, schaeffer and shallit. *Southeast Asian Bulletin of Mathematics*, 41(6), 2017.

[31] V. Keränen. Abelian squares are avoidable on 4 letters. In *19th International Colloquium on Automata, Languages, and Programming*, pages 41–52. Springer, 1992.

[32] M. Leconte. A characterization of power-free morphisms. *Theoretical Computer Science*, 38(1) :117–122, 1985.

[33] F. Lietard and M. Rosenfeld. Avoidability of additive cubes over alphabets of four numbers. In N. Jonoska and D. Savchuk, editors, *Developments in Language Theory, DLT 2020, Lecture Notes in Computer Science*, volume 12086, pages 192–206, Cham, 2020. Springer.

[34] M. Lothaire. *Algebraic combinatorics on words*, volume 90 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2002.

[35] G. Pirillo and S. Varricchio. On uniformly repetitive semigroups. *Semigroup Forum*, 49(1) :125–129, 1994.

[36] P.A.B. Pleasants. Non-repetitive sequences. *Mathematical Proceedings of the Cambridge Philosophical Society*, 68(2) :267–274, 1970.

[37] M. Rao. On some generalizations of abelian power avoidability. *Theoretical Computer Science*, 601 :39–46, 2015.

[38] M. Rao and M. Rosenfeld. Avoidability of long k-abelian repetitions. *Mathematics of Computation*, 85(302) :3051–3060, 2016.

[39] M. Rao and M. Rosenfeld. Avoiding two consecutive blocks of same size and same sum over $\mathbb{Z}^2$. *SIAM Journal on Discrete Mathematics*, 32(4) :2381–2397, 2018.

[40] M. Rigo. Relations on words. *Indagationes Mathematicae*, 28(1) :183–204, 2017.

[41] M. Rosenfeld. *Avoidability of Abelian Repetitions in Words*. PhD thesis, École Normale Supérieure de Lyon, 2017. 2017LYSEN033.

[42] A. Thue. Über unendliche Zeichenreihen. *Norske Vid Selsk. Skr. I Mat-Nat Kl.(Christiana)*, 7 :1–22, 1906.

[43] A. Thue. Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. *Norske Vid Selsk. Skr. I Mat-Nat Kl.(Christiana)*, 7 :1–67, 1912.