AUTOUR DE LA GÉNÉRATION ALÉATOIRE SOUS MODÈLE DE BOLTZMANN

Olivier BODINI

Mémoire d'habilitation à diriger des recherches

Spécialité Informatique

Université Pierre et Marie Curie (UPMC-Sorbonne Universités),
LIP6, équipe APR.

Habilitation soutenue le 3 décembre 2010 devant le jury constitué de :

F. Bergeron, rapporteur, Professeur, Université du Québec, Montréal, Canada.

P. Duchon, examinateur, Professeur, Université Bordeaux 1, Bordeaux.

P. Flajolet, rapporteur, DR INRIA, INRIA Rocquencourt, Le Chesnay.

P. Fraigniaud, examinateur, DR CNRS, Université Paris Diderot, Paris.

B. Gittenberger, examinateur, Institut für Diskrete Mathematik und Geometrie, Wien, Austria.

N. Schabanel, examinateur, DR CNRS, Université Paris Diderot, Paris.

G. Schaeffer, rapporteur, DR CNRS, École Polytechnique, Palaiseau.

M. Soria, examinatrice, Professeur, UPMC, Paris.

UPMC SORBONNE UNIVERSITÉS

Table des matières

Remerciements

C'est un lieu commun, tout le monde vous le dira, la recherche n'est pas un métier mais une passion. Cette passion je l'ai assouvie grâce à vous tous, chers collègues, chers thésards, chers maîtres à penser. Car la recherche n'est certes pas une passion solitaire, c'est une passion qui se partage, qui se vit à plusieurs, qui s'échange et se diffuse entre nous. Et c'est lors de toutes nos interactions, tous nos travaux en commun, tous ces moments d'apprentissages et de questionnements, que progressivement se modèle notre propre vision de la science et des objets à étudier. Cette vision reste en perpétuelle mutation et sans nul doute, vous tous enrichirez-vous encore ce jardin secret. Je vous en remercie du plus profond de mon cœur.

Mon parcours n'a pas été linéaire, il a suivi le cours sinueux d'un torrent de montagne, parfois tourmenté, parfois plus calme, je fus ballotté de l'UPMC à l'ENS-Lyon, de l'ENS-Lyon à l'université Montpellier-2, pour finalement accoster de nouveau sur les berges accueillantes de l'UPMC. Ce parcours fut pour moi un chemin initiatique qui m'a fait voyager de l'informatique aux mathématiques, de l'algèbre à l'algorithmique, de l'analyse aux structures de données. Durant celui-ci, je me suis enrichi de tant de paysages, de la théorie des graphes à la théorie de mots, de la géométrie algébrique effective à la théorie des pavages, de la théorie additive à l'analyse d'algorithmes ...

Merci infiniment à tous ceux avec qui j'ai eu la chance de publier : A. Darrasse [9, 37], P. Duchet [10], T. Fernique [11, 14, 15, 16, 12, 13], E. Fusy [17], D. Gardy [19, 18], S.K. Hwang [37], A. Jacquot [20, 21, 18], D. Jamet [22], M. Latapy [23], S. Lefranc [10, 24], J. Lumbroso [25], B. Nouvel [26], C. Pivoteau [17], Y. Ponty [27], D. Regnault [12, 13], O. Roussel [19], E. Remila [28, 14, 15, 16], E. Rivals [29], M. Soria [9, 37]. Mais aussi à tous ceux avec qui je rêve toujours d'écrire un jour un article ...

Vous tous m'avez ouvert les yeux sur l'immensité des connaissances et des savoirs et sur l'infinie profondeur de la science.

Ce métier est le plus beau du monde, grâce à vous. J'aurais trop de crainte à entamer ici une liste exhaustive de personnes à remercier, trop peur à l'idée d'oublier l'un de vous, je vous remercie donc tous anonymement mais avec le plus grand respect.

Je ne peux oublier de remercier le LIP6, l'équipe APR et en particulier le groupe Génération Aléatoire. Merci donc à Michèle, Maryse et Antoine et à tous les jeunes, Carine, Alexis, Olivier, Jérémie, Xiaomin, Alice, pour leur gentillesse et les conditions de travail idéales qu'ils ont sues créer.

Merci à tous ceux qui m'ont aidé à corriger les typos dans ce mémoire. Ce remerciement s'adresse tout spécialement à Antoine Genitrini et Olivier Roussel. Merci aussi à Jérémie Lumbroso for his perfect english.

Un remerciement particulier à P. Flajolet et à M. Soria dont les qualités humaines et scientifiques m'ont fait beaucoup progresser et sans qui cette habilitation n'aurait jamais eu lieu.

Et, je ne peux conclure sans un coupable remerciement à Sandrine et Quentin qui me supportent (dans tous les sens) et que j'aime bien plus que tout ogre n'a jamais aimé la chair des enfants.

Je tiens également à remercier les membres de mon jury, et tout particulièrement mes rapporteurs. Je suis honoré de pouvoir leur présenter mon travail.

Résumé

Ce mémoire a comme cadre général la combinatoire analytique et l'algorithmique sous-jacente. Il aborde essentiellement la question de la génération aléatoire uniforme de structures combinatoires, et plus particulièrement la génération aléatoire sous modèle de Boltzmann. Dans ce modèle, la distribution de taille des structures combinatoires suit une loi bien connue en physique statistique – décrivant l'énergie libre d'un système – et la relaxation qui consiste à ne plus imposer la taille de l'objet à générer permet de concevoir des générateurs efficaces. Ce mémoire présente la génération sous modèle de Boltzmann introduite par P. Duchon, P. Flajolet, G. Louchard et G. Schaeffer en 2004 et différentes extensions et prolongements possibles du modèle. Nous avons choisi de décrire les extensions pour les classes colorées, les classes multiparamétrées, les classes multiplicatives de Dirichlet, les classes différentielles et pour le produit d'Hadamard. La dernière partie regroupe un certain nombre d'applications de ces extensions allant de la physique statistique (génération de partitions planes), à la génération de pavages, en passant par la génération de structures ordonnées.

Abstract

This dissertation's main topic is analytic combinatorics and its underlying algorithmic. It focuses on uniformly sampling random combinatoric structures, in particular through the Boltzmann model. With this model, the size of combinatoric structures is distributed according to a law well-known in statistical physics for describing the free energy of systems, but the usual constraint, that the generated objects be of a given size, is relaxed. This allows for very efficient samplers. This dissertation presents random sampling through the Boltzmann model as introduced by P. Duchon, P. Flajolet, G. Louchard and G. Schaeffer in 2004, and the several possible extensions and additions to this model. We have chosen to mention the addition of colored classes, of multi-parameterized classes, of Dirichlet classes with a multiplicative size, of differential classes, and of construction of the Hadamard product. The last part details a number of application of these extensions in different domains spanning statistical physics (see the generation of plane partitions), random sampling of tilings, random sampling of ordered structures.

1 Préambule

Nous avons fait le choix de présenter dans ce mémoire des travaux très récents sur la génération aléatoire sous modèle de Boltzmann : deux conférences de 2010 (les générateurs de Boltzmann multiparamétrés [27], un produit d'Hadamard pour les générateurs de Boltzmann [19]), un article à paraître sur les générateurs pour les classes différentielles [30], et deux travaux en cours de finalisation (générateurs de Boltzmann pour les classes colorées [20] et générateurs pour les classes multiplicatives de Dirichlet). Ce choix a été motivé par le désir de montrer la remarquable flexibilité du modèle de Boltzmann introduit par Duchon, Flajolet, Louchard, Schaeffer [40]. Cette théorie a l'élégance de toutes les belles théories, à savoir de passer aux généralisations de manière limpide. Notre choix a aussi été motivé par l'apport fondalemental que peut révéler cette approche pour des applications concrètes. Les possibilités accrues, un réel changement d'échelle dans la taille des objets que l'on peut générer ouvre la porte à de nombreuses et cruciales applications dans le domaine de l'informatique (simulation de réseaux sociaux, test logiciel, débogages,...), comme dans celui de la physique statistique (modèle pour les quasi-cristaux, modèle de dimères sur réseau, cristal de Wulff (partition plane),...), ou dans celui de la bio-informatique (génération de structures ARN, interactions protéines-ARN,...).

2 Introduction

Dans ce mémoire, il a été privilégié la cohérence à l'exhaustivité, je ne présenterai donc pas en détail l'ensemble des travaux de recherche que j'ai menés depuis la thèse de doctorat (soutenue en novembre 1999). Ceci fut pour moi un choix difficile à réaliser. Il ne signifie nullement un renoncement ou un discrédit de mes autres travaux. Mais un désir de ne pas articuler superficiellement mes thèmes de recherche alors que leur évolution tient en grande partie à mes changements d'affectation. Je tenterai donc essentiellement ici à mettre en exergue et à synthétiser l'apport que constitue la méthode de Boltzmann dans le monde de la génération aléatoire, tout en y inscrivant mes propres contributions (applications, extensions, généralisations, etc.). J'insisterai aussi fortement sur les perspectives qui me semblent centrales pour ce domaine, et qui constituent en grande partie mon projet de recherche à long terme. Enfin, dans un chapitre très résumé, je mentionnerai succinctement mes travaux les plus significatifs autour de la théorie des pavages, la logique quantitative et l'analyse des graphes aléatoires.

Ce chapitre introductif présente les objets qui sont au fondement de la génération aléatoire automatisable, à savoir les classes combinatoires spécifiables. La génération aléatoire joue un rôle de plus en plus marqué dans l'analyse et la compréhension des structures complexes (Internet, réseaux sociaux, réseaux de téléphonie, phénomène de transition de phase en physique statistique, formation des quasi-cristaux, modélisation des ARN, etc.). La génération aléatoire est aussi la clef de voûte du test intensif et de la validation logicielle. Nous sommes fréquemment confrontés au problème de construire des générateurs aléatoires performants et effectifs dans le sens où ils doivent être facilement implémentables et être capables de produire des objets aux dimensions adéquates.

Plusieurs nouvelles familles de générateurs ont vu le jour récemment. Parmi elles, citons la méthode

récursive [44], la méthode du couplage arrière (CFTP) introduite par Propp (1996-98) [68] et les générateurs de Boltzmann introduits par Duchon, Flajolet, Louchard, Schaeffer (2003) [40].

Les générateurs de Boltzmann reposent sur le fait de générer les objets combinatoirement spécifiables avec une probabilité essentiellement proportionnelle une exponentielle de la taille de l'objet. En particulier, les objets sont tirés uniformément parmi ceux de même taille. Ces générateurs sont automatiquement constructibles depuis la spécification de la classe. Plusieurs ordres de grandeur sur la taille des objets générables ont été gagnés par rapport aux méthodes récursives de Nijenhuis et Wilf [44]. Ce gain repose essentiellement sur un relâchement autorisé sur les objets engendrés : à savoir de ne plus imposer une taille fixée mais seulement une taille approchée pour la sortie. Ce relâchement permet alors une génération (toujours uniforme parmi les objets de même taille) en temps moyen linéaire en fonction de la taille de l'objet obtenu, ce qui est bien évidemment dans un sens optimal.

2.1 Classes combinatoires

Les classes combinatoires constituent un objet d'étude classique, elles sont à la base de la méthode symbolique [43]. Nous proposons dans cette introduction un cadre général, permettant de décrire de manière unifiée une large variété de "marquages" pour les classes combinatoires (incluant les classiques classes étiquetées, non-étiquetées et pointées). Bien que seules les classes combinatoires étiquetées et non-étiquetées admettent des séries énumératives faciles à construire depuis une description symbolique des classes, il nous est apparu intéressant d'envisager de porter notre attention sur d'autres marquages qui semblaient faire sens. La section 3.1 traite de la génération aléatoire des structures colorées, tandis qu'en perspective nous discuterons de la pertinence des structures semi-étiquetées. Pour résumer, ce cadre nous sert dans ce mémoire de support pour définir la notion d'objets esquissés et colorés dont la génération aléatoire est l'objet du chapitre 3.1 et il contient le germe d'un projet de recherche axé autour de nouveaux types de marquages.

Notre approche ici peut être vue comme une introduction à une version très simplifiée de théorie des espèces [5]. Cette approche nous est néanmoins apparue comme syntaxiquement suffisante et bien adaptée à l'implémentation sur machine des générateurs aléatoires que nous désirions traiter. Pour cela, notons $\mathfrak{A}$ l'alphabet constitué des 7 symboles suivants $\Box\langle\rangle[]()$. Nous appellerons *carré* le symbole $\Box$, les autres symboles correspondent à 3 types de parenthésage (ces parenthèses permettront ultérieurement de coder la notion de séquence, de cycle et de multi-ensemble).

Nous notons $\mathcal{L}$ le langage sur $\mathfrak{A}$ (qui correspond à tous les parenthésages possibles avec 3 types de parenthèses) défini comme suit :

- $\Box, \langle\rangle, (), []$ appartiennent à $\mathcal{L}$
- $\forall k \in \mathbb{N}^*, A_1, \cdots, A_k \in \mathcal{L} \Rightarrow (A_1 A_2 \cdots A_k) \in \mathcal{L}$, les *séquences*
- $\forall k \in \mathbb{N}^*, A_1, \cdots, A_k \in \mathcal{L} \Rightarrow [A_1 A_2 \cdots A_k] \in \mathcal{L}$, les *cycles (séquences circulaires)*
- $\forall k \in \mathbb{N}^*, A_1, \cdots, A_k \in \mathcal{L} \Rightarrow \langle A_1 A_2 \cdots A_k \rangle \in \mathcal{L}$, les *multi-ensembles*

Nous appellerons les éléments de $\mathcal{L}$ des *proto-objets combinatoires*. Nous pouvons d'ores et déjà définir la *taille* d'un proto-objet A comme le nombre d'atomes contenus dans A, c'est à dire le nombre d'occurrences de la lettre $\Box$ dans A. Par exemple, $(\langle[\Box\Box](\Box\Box)\rangle\Box)$ est un proto-objet combinatoire de taille 5.

De plus, pour tout proto-objet A, on note $Atom(A)$ l'ensemble des occurrences (positions) de $\Box$ dans A. En particulier, $|Atom(A)|$ est la taille de A et chaque élément de $Atom(A)$ correspond à exactement une occurrence identifiée de $\Box$ dans A.

Pour le moment, nous disposons juste d'un support, il nous faut dorénavant exprimer les symétries des cycles et des multi-ensembles en définissant des classes d'équivalences sur les proto-objets. Il nous faut néanmoins préalablement introduire la notion naturelle de couleur. Un proto-objet *multicolore* est un couple, noté A^f, où A est un mot de $\mathcal{L}$ et f est une fonction de $Atom(A)$ dans $\mathbb{N}$ (c'est la *coloration*). Naturellement, pour a un atome dans A, on appelle $f(a)$ la *couleur* de a. Et pour simplifier les écritures, on note $\mathcal{L}^c$ l'ensemble des proto-objets multicolores. Laissons le lecteur remarquer qu'un même proto-objet a en général plusieurs colorations (fonction f) possibles.

Nous pouvons maintenant définir récursivement la notion d'équivalence $\equiv$ entre proto-objets multicolores de $\mathcal{L}^c$ comme suit :

– Deux carrés multicolores sont équivalents si et seulement s'ils ont la même couleur :

$$\forall f, f', \quad \Box^f \equiv \Box^{f'} \Leftrightarrow f(\Box) = f'(\Box)$$

– Deux séquences sont équivalentes si et seulement si leurs sous-structures sont en même nombre et équivalentes une à une :

$$\forall A_1, \cdots, A_k, B_1, \cdots, B_k \in \mathcal{L}, (A_1 A_2 \cdots A_k)^f \equiv (B_1 B_2 \cdots B_k)^{f'}$$
$$\Leftrightarrow$$
$$\forall i \in \{1, \cdots, k\}, f(Atom(A_i)) = f'(Atom(B_i)) \text{ et } A_i^{f|_{Atom(A_i)}} \equiv B_i^{f'|_{Atom(B_i)}}$$

– Deux cycles sont équivalents si et seulement si leurs sous-structures sont équivalentes à permutation circulaire près :

$$\forall A_0, \cdots, A_{k-1}, B_0, \cdots, B_{k-1} \in \mathcal{L}, [A_0 A_1 \cdots A_{k-1}]^f \equiv [B_0 \cdots B_{k-1}]^{f'}$$
$$\Leftrightarrow$$
$$\exists n \forall i, f(Atom(A_i)) = f'(Atom(B_{i+n \bmod k}))$$
$$\text{et}$$
$$A_i^{f|_{Atom(A_i)}} \equiv B_{i+n \bmod k}^{f'|_{Atom(B_{i+n \bmod k})}}$$

– Deux multi-ensembles sont équivalents si et seulement si leurs sous-structures sont équivalentes à permutation près :

$$\forall A_0, \cdots, A_{k-1}, B_0, \cdots, B_{k-1} \in \mathcal{L}, \langle A_0 \cdots A_{k-1} \rangle^f \equiv \langle B_0 \cdots B_{k-1} \rangle^{f'}$$
$$\Leftrightarrow$$
$$\exists \sigma \in S_k \text{ le groupe symétrique d'ordre } k, \ \forall i, f(Atom(A_i)) = f'(Atom(B_{\sigma(i)}))$$
$$\text{et}$$
$$A_i^{f|_{Atom(A_i)}} \equiv B_{\sigma(i)}^{f'|_{Atom(B_{\sigma(i)})}}$$

Nous avons à présent défini les symétries liées aux cycles et aux multi-ensembles. On peut alors définir l'ensemble $\mathcal{O}^c$ formé des *objets combinatoires multicolores* comme étant l'ensemble $\mathcal{L}^c/\equiv$. La notion de classes combinatoires en découle naturellement dès lors que l'on mentionne que la notion de taille passe aux classes d'équivalences : la *taille* d'un objet combinatoire multicolore est la taille d'un quelconque proto-objet qui le représente.

Définition 2.1.1 *Une classe combinatoire multicolore est un sous-ensemble avec multiplicité de $\mathcal{O}^c$ ayant un nombre fini d'objet de même taille.*

Nous optons ici pour des classes définies comme des ensembles à multiplicité (c'est à dire des sous-ensembles de $\mathcal{O}^c \times \mathbb{N}^*$). Cela permet d'éviter le traditionnel problème lié à l'union disjointe. L'*union* $A \cup B$ de deux classes est la classe contenant tous les éléments de A et B et dont la multiplicité de a dans $A \cup B$ est la somme des multiplicités de a dans A et dans B.

Classiquement, les objets de taille 0 sont appelé les *objets neutres* et le symbole $\Box$ l'atome. Par exemple, $\{()()[]\}$ est un objet neutre. Manifestement, $\mathcal{O}^c$ contient une infinité d'objets neutres. On prendra garde à éviter la confusion entre la classe combinatoire vide $\varnothing$ et une classe combinatoire réduite à un objet neutre.

Afin de bien asseoir ces définitions, regardons l'exemple suivant qui les illustre simplement :

L'enjeu de cet exposé quelque peu indigeste est de permettre la considération aisée de certaines restrictions sur f, menant à différents types de *marquage* pour les objets combinatoires. Nous retrouvons bien évidement les marquages classiques (non-étiqueté, étiqueté), mais aussi d'autres formes de marquages moins communs mais néanmoins très raisonnables. Le tableau 1 suivant liste certaines restrictions et les marquages associés :

Commentons plus précisément certaines lignes de ce tableau. Le marquage semi-étiqueté fait l'objet d'un travail de recherche joint avec M. Bodirsky. Dans ce cas, chaque atome prend une couleur (deux

$$\mathcal{C} = \left\{ \left(\boxed{1} \right), \left\langle \left[\boxed{2} \boxed{1} \right] \boxed{1} \right\rangle, \left\langle \boxed{1} \left[\boxed{1} \boxed{2} \right] \right\rangle \right\}$$

FIGURE 1: Une classe combinatoire $\{1, 2\}$-colorée contenant trois objets et dont les deux objets de taille 3 sont identiques

Fonction	Codomaine	Marquage
Bijection	$\{1, \cdots, n\}$ pour les objets de taille n	Étiqueté
Fonction constante $f(x) = 1$	$\{1\}$	Non-étiqueté
Fonction point : $\exists! y, \forall x \neq y, f(x) = 1$ et $f(y) = 0$	$\{0, 1\}$	Punaisage
Fonction sans restriction	$\{1, \cdots, k\}$ avec k fixé	k-coloré
Surjection	$\exists k \leqslant n, \{1, \cdots, k\}$	Semi-étiqueté
Fonction sans restriction	$\{1, \cdots, n\}$ pour un objet de taille n	Coloré

TABLE 1: Différents types de marquages

atomes pouvant avoir la même couleur), mais si un atome a une couleur j alors il y a au moins un atome de couleur i pour tout $i < j$. Dans un travail en cours avec Bodirsky, nous montrons qu'il est possible de mettre en relation les séries génératrices des classes semi-étiquetées et celles des séries non-étiquetées.

Le punaisage correspond au marquage d'un atome d'un objet non-étiqueté, on prendra garde à ne pas faire la confusion avec le pointage classique des objets étiquetés [43] p86. Le punaisage ainsi défini induit des séries génératrices beaucoup plus compliquées à décrire à cause des symétries.

Pour conclure cette section, donnons deux exemples basiques de ce que peuvent coder les classes colorées et les classes semi-étiquetées.

Considérons que l'on désire compter ou estimer asymptotiquement le nombre de colliers de n perles constitués à partir de n types de perles différents. La classe des colliers ainsi définis est la classe combinatoire colorée $\textsc{Cyc}(\mathcal{Z})$.

Quel est le nombre de façons de faire des groupes avec n personnes où la seule chose qui compte est la taille des différents groupes et l'ordre des groupes les uns par rapport aux autres (un groupe G_1 de 2 personnes et un groupe G_2 d'une personne ne sont pas pareil qu'un groupe G_1 d'une personne et un groupe G_2 de deux personnes). La classe des groupes ainsi formée est une classe combinatoire semi-étiquetée $\textsc{Seq}(\mathcal{Z})$. Notons que cette classe très simple peut aussi être interprétée comme la classe non-étiquetée $\textsc{Seq}(\mathcal{Z} \times \textsc{Seq}(\mathcal{Z}))$.

2.2 Classes spécifiables et séries génératrices associées

Le cadre des classes combinatoires est en général très largement surdimensionné et peu en adéquation avec une vision constructiviste imposée par le désir d'applications. Nous reprenons l'approche symbolique qui consiste à étudier les classes combinatoires construites à partir d'un jeu fini de règles de base. En calquant cette démarche, on peut très naturellement définir des opérateurs de construction ensemblistes ($+$, $\times$, SEQ, SET, CYC) [1] sur les proto-objets de sorte à introduire la notion de proto-classes combinatoires spécifiables :

- La proto-classe atomique $\{\square\}$ est une proto-classe spécifiable.
- L'opérateur $+$ correspond à l'union (au sens des ensembles avec multiplicité) de deux proto-classes spécifiables.

1. On fait le choix délibérément restrictif ici concernant le nombre de constructeurs, en particulier, on ne définira pas l'opérateur classique PSet.

- La proto-classe $\mathcal{A} \times \mathcal{B}$ définie à partir de deux proto-classes spécifiables $\mathcal{A}$ et $\mathcal{B}$ est la proto-classe : $\{(AB)|(A,B) \in \mathcal{A} \times \mathcal{B}\}$. A noter que le produit tel qu'il est défini ici n'est pas associatif. Il est donc de bon aloi de définir $\textsc{Seq}_k$ comme $\{(A_1 \cdots A_k)|A_i \in \mathcal{A}\}$.
- La proto-classe $\textsc{Seq}(\mathcal{A})$ (resp. $\textsc{Cyc}(\mathcal{A})$ et $\textsc{Set}(\mathcal{A})$) correspond à la proto-classe :

$$\{(A_1 \cdots A_k) \text{ (resp. } [A_1 \cdots A_k] \text{ et } \langle A_1 \cdots A_k \rangle)|A_i \in \mathcal{A} \text{ et } k \in \mathbb{N}\}.$$

- Construction par récursion bien fondée, c'est à dire des récurrences de la forme $\mathcal{T} = \mathcal{F}(\mathcal{T}, \mathcal{Z})$ où $\mathcal{F}$ est une fonction sur les proto-classes combinatoires telle que les objets de taille n de $\mathcal{T}$ s'expriment par $\mathcal{F}$ à partir d'objets de $\mathcal{T}$ de tailles strictement plus petites.

Il es important de remarquer que les constructeurs Seq, Set, Cyc ne s'appliquent qu'aux classes n'ayant pas d'objet neutre.

On peut maintenant donner une définition des classes constructibles (nous renvoyons le lecteur à [43] page 33 pour une définition classique).

Définition 2.2.1 *Une* classe combinatoire constructible (ou spécifiable) $\mathcal{A}(\mathcal{Z})$ *est le quotient par* $\equiv$ *d'une proto-classe constructible.*

Par exemple, $\mathcal{T} = \mathcal{Z}\mathcal{T} + \textsc{Set}(\mathcal{Z} \times \textsc{Seq}(\mathcal{Z}))$ définit bien une classe combinatoire. En effet, les objets de taille n sont définis récursivement à partir des objets de $\mathcal{T}$ de taille $n-1$.

Dans le cas classique des classes combinatoires non-étiquetées (resp. étiquetées), nous disposons d'une grammaire de réécriture permettant à partir d'une spécification d'obtenir automatiquement la série énumérative ordinaire, à savoir $C(x) = \sum_{\gamma \in \mathcal{C}} x^{|\gamma|}$ (resp. exponentielle, à savoir $\hat{C}(x) = \sum_{\gamma \in \mathcal{C}} \frac{1}{|\gamma|!} x^{|\gamma|}$) associée. Le tableau 2 ci-dessous résume cette grammaire.

Dans le cas d'autres marquages, l'obtention d'une formule pour le cycle et le multi-ensemble est en général beaucoup plus délicat à cause des symétries.

	spécification	fonction génératrice ordinaire (non étiquetée)	f. g. exponentielle (étiquetée)
$\{\varepsilon\}$, $\{\square\}$	$1 \ , \ \mathcal{Z}$	$1 \ , \ x$	$1 \ , \ x$
Union	$\mathcal{C} = \mathcal{A} + \mathcal{B}$	$C(x) = A(x) + B(x)$	$\hat{C}(x) = \hat{A}(x) + \hat{B}(x)$
Produit	$\mathcal{C} = \mathcal{A} \times \mathcal{B}$	$C(x) = A(x) \times B(x)$	$\hat{C}(x) = \hat{A}(x) \times \hat{B}(x)$
Séquence	$\mathcal{C} = \textsc{Seq}(\mathcal{A})$	$C(x) = \frac{1}{1-A(x)}$	$\hat{C}(x) = \frac{1}{1-\hat{A}(x)}$
Multi-ensemble	$\mathcal{C} = \textsc{Set}(\mathcal{A})$	$\exp\left(\sum_{k=1}^{\infty} \frac{1}{k} A\left(x^k\right)\right)$	$\hat{C}(x) = \exp(A(x))$
Cycle	$\mathcal{C} = \textsc{Cyc}(\mathcal{A})$	$\sum_{k=1}^{\infty} \frac{\varphi(k)}{k} \ln \frac{1}{1-A\left(x^k\right)}$	$\hat{C}(x) = \ln \frac{1}{1-A(x)}$

Table 2: Séries génératrices pour les opérateurs classiques

2.3 Modèle de Boltzmann

Cette section résume la présentation des générateurs de Boltzmann faite par Duchon, Flajolet, Louchard, Schaeffer dans [40]. Revenons donc au cadre classique des classes combinatoires étiquetées et non-étiquetées. Pour une classe $\mathcal{C}$ donnée, le modèle de Boltzmann de paramètre x (un réel positif) attribue aux objets de $\mathcal{C}$ une probabilité qui est proportionnelle à une exponentielle de sa taille. Plus précisément, on a dans le cas non-étiqueté (resp. étiqueté) :

$$\mathbb{P}_x(\gamma) = \frac{x^{|\gamma|}}{C(x)} \text{ avec } C(x) = \sum_{\gamma \in \mathcal{C}} x^{|\gamma|} = \sum_{n \in \mathbb{N}} c_n x^n$$

$$(\text{resp. } \hat{\mathbb{P}}_x(\gamma) = \tfrac{1}{|\gamma|!}\frac{x^{|\gamma|}}{\hat{C}(x)} \text{ avec } \hat{C}(x) = \sum_{\gamma\in\mathcal{C}} \tfrac{1}{|\gamma|!}x^{|\gamma|} = \sum_{n\in\mathbb{N}} c_n \tfrac{1}{n!}x^n)$$

Un *générateur de Boltzmann* $\Gamma\mathcal{C}_x$ pour $\mathcal{C}$ est un algorithme qui engendre des objets de $\mathcal{C}$ suivant une distribution conforme à l'un de ces modèles. Pour un paramètre donné, deux objets de même taille ont la même probabilité d'être tirés. On notera également que, contrairement à la méthode récursive, cette probabilité met en jeu la valeur de la série génératrice au point x, et non plus les coefficients de la série. Pour que les probabilités soient bien définies, la série $C(x)$ doit donc converger en x. En d'autres termes, x doit appartenir à l'intervalle $[0, \rho[$ (il est parfois possible d'étendre ce domaine à $[0, \rho]$), où ρ est la singularité dominante [2] de la série ordinaire $C(x)$ (ou de la série exponentielle $\hat{C}(x)$ dans le cas étiqueté). Dans ce modèle, il n'est à priori pas nécessaire de calculer les coefficients des séries génératrices mais seulement de calculer des valeurs de séries génératrices en un point. Cette évaluation est à priori de complexité algorithmique inférieure. Utilisant la méthode de Newton, les auteurs Pivoteau, Salvy et Soria [65] ont donné une façon efficace pour résoudre cette question.

Analysons un peu plus en détail les distributions de Boltzmann. Pour ce faire, notons N (resp. $\hat{N}$) la variable aléatoire correspondant à la taille des structures engendrées dans le cas non-étiqueté (resp. dans le cas étiqueté), alors la probabilité de tirer un objet de taille n par un générateur de paramètre x est :

$$\mathbb{P}_x(N = n) = \frac{c_n x^n}{C(x)}, \text{ dans le cas ordinaire/non étiqueté,}$$
$$(\text{resp. } \mathbb{P}_x(\hat{N} = n) = \frac{c_n x^n}{n!\hat{C}(x)}, \text{ dans le cas exponentiel/étiqueté.})$$

On en déduit les expressions suivantes pour la série de probabilité ordinaire [3] de N (resp.$\hat{N}$) :

$$\mathbb{E}_x\left[u^N\right] = \frac{C(ux)}{C(x)} \qquad (\text{resp. } \mathbb{E}_x\left[u^{\hat{N}}\right] = \frac{\hat{C}(ux)}{\hat{C}(x)}),$$

d'où l'on tire sans peine les moments factoriels $\mathbb{E}_x\left[N^{[k]}\right]$ d'ordre k ainsi que l'écart-type $\sigma(N)$:

$$\mathbb{E}_x\left[N^{[k]}\right] = \frac{x^k C^{(k)}(x)}{C(x)} \qquad (\text{resp. } \mathbb{E}_x\left[\hat{N}^{[k]}\right] = \frac{x^k \hat{C}^{(k)}(x)}{\hat{C}(x)}),$$

et $\sigma_x(N) = \sqrt{\mathbb{E}_x\left[N^{[2]}\right] - \mathbb{E}_x\left[N\right](\mathbb{E}_x[N]-1)} = \sqrt{\dfrac{\mathrm{d}^2 C(x)}{\mathrm{d}(\ln(x))^2}}$, (resp. $\sigma_x(\hat{N}) = \sqrt{\dfrac{\mathrm{d}^2 \hat{C}(x)}{\mathrm{d}(\ln(x))^2}}$.)

Rappelons aussi que l'on passe facilement des moments factoriels aux moments ordinaires $\mathbb{E}_x\left[N^k\right]$ en utilisant la relation $\mathbb{E}_x\left[N^k\right] = \sum_{i\in\{1,..,k\}} a_i \mathbb{E}_x\left[N^{[i]}\right]$ où les a_i sont les coefficients du polynôme Φ_k défini récursivement par la relation : $\Phi_1(z) = z$ et $\Phi_i(z) = z(\Phi'_{i-1}(z) + \Phi_{i-1}(z))$. En particulier, $\mathbb{E}_x\left[N^2\right] = \mathbb{E}_x\left[N^{[2]}\right] + \mathbb{E}_x[N]$ et $\mathbb{E}_x\left[N^3\right] = \mathbb{E}_x\left[N^{[3]}\right] + 3\mathbb{E}_x\left[N^{[2]}\right] + \mathbb{E}_x[N]$.

On mentionne pour conclure cette section que la variance s'exprime très simplement en fonction de la dérivée de l'espérance :

$$\mathbb{V}_x[N] = x\mathbb{E}'_x[N], \qquad (\text{resp. } \mathbb{V}_x[\hat{N}] = x\mathbb{E}'_x[\hat{N}]).$$

Ceci assure en particulier qu'étant donnée $\mathcal{C}$, une classe combinatoire, l'espérance (comme fonction de x) de la taille suivant une distribution de Boltzmann est croissante sur l'intervalle $[0, \rho[$ où ρ est la singularité dominante de la série $C(x)$. En d'autres termes, la génération d'objets de grande taille nécessite de prendre pour paramètre une valeur souvent proche de la singularité dominante ρ.

2. Par le théorème de Pringsheim, la singularité dominante est sur l'axe réel positif.
3. Rappelons que la série ordinaire de probabilité $p_x(u)$ est par définition $p_x(u) = \mathbb{E}_x\left[u^N\right] = \sum \mathbb{P}_x(N = n)u^n$.

2.4 Règles de construction

Le point crucial maintenant est la construction automatique d'un générateur de Boltzmann à partir de la spécification de la classe. Le tableau 3 résume les règles de construction du générateur dans le cas des classes étiquetées (on parlera du cas non-étiqueté ultérieurement), tandis que le tableau 4 rappelle les principales distributions utilisées pour les générateurs de Boltzmann. On vérifie alors sans peine qu'un générateur de Boltzmann (libre) engendre un objet γ en un temps linéaire par rapport à la taille de γ. Ceci nous amène naturellement à la question suivante : quid de la complexité si l'on désire un objet d'une certaine taille ?

$Classe\mathcal{A}$	Description	$A(z)$	$\Gamma\mathcal{A}(x)$
ε	Classe neutre	1	**return** ε
$\mathcal{Z}$	Classe atomique	z	**return** $\square$
$\mathcal{B}\times\mathcal{C}$	Produit	$B(z)\times C(z)$	**return** $(\Gamma\mathcal{B}(x),\Gamma\mathcal{C}(x))$
$\mathcal{B}+\mathcal{C}$	Union	$B(z)+C(z)$	**if** $Bern\left(\frac{B(x)}{B(x)+C(x)}\right)$ **then return** $\Gamma\mathcal{B}(x)$ **else return** $\Gamma\mathcal{C}(x)$
$\textsc{Seq}(\mathcal{B})$	Séquence	$\frac{1}{1-B(z)}$	$l := Geom(B(x))$ **return** $\underbrace{(\Gamma\mathcal{B}(x),\ldots,\Gamma\mathcal{B}(x))}_{l \text{ fois}}$
$\textsc{Set}(\mathcal{B})$	Multi-ensemble	$e^{B(z)}$	$l := Pois(B(x))$ **return** $\underbrace{\{\Gamma\mathcal{B}(x),\ldots,\Gamma\mathcal{B}(x)\}}_{l \text{ fois}}$
$\textsc{Cyc}(\mathcal{B})$	Cycle	$\ln\left(\frac{1}{1-B(z)}\right)$	$l := Logarithmic(B(x))$ **return** $\underbrace{\langle\Gamma\mathcal{B}(x),\ldots,\Gamma\mathcal{B}(x)\rangle}_{l \text{ fois}}$

Table 3: Règles pour construire un générateur de Boltzmann pour les classes combinatoires spécifiables étiquetées

Distribution	Notation	Definition
Bernoulli	Bern(p)	$\mathbb{P}(0)=1-p$ et $\mathbb{P}(1)=p$ (avec $0\leqslant p\leqslant 1$)
Geometric	Geom(λ)	$\mathbb{P}(k)=\lambda^k(1-\lambda)$ (avec $k\in\mathbb{N}$ et $0\leqslant\lambda<1$)
Poisson	Pois(λ)	$\mathbb{P}(k)=e^{-\lambda}\frac{\lambda^k}{k!}$ (avec $k\in\mathbb{N}$ et $\lambda\in\mathbb{R}^{+*}$)
Logarithmique	Logarithmic(λ)	$\mathbb{P}(k)=-\ln(1-\lambda)^{-1}\frac{\lambda^k}{k}$ (avec $k\in\mathbb{N}^*$ et $0\leqslant\lambda<1$)
Positive Poisson	Pois$_{\geqslant 0}(\lambda)$	$\mathbb{P}(k)=\frac{1}{e^\lambda-1}\frac{\lambda^k}{k!}$ (avec $k\in\mathbb{N}^*$ et $\lambda\in\mathbb{R}^{+*}$)

Table 4: Distributions utiles dans les générateurs de Boltzmann.

2.5 Génération en taille approchée ou exacte, complexité algorithmique

Un générateur de Boltzmann en taille approchée $\Gamma\mathcal{A}(x,\varepsilon)$ (resp. taille exacte) consiste simplement à rajouter une étape de rejets sur la taille de la sortie d'un générateur de Boltzmann libre tant que l'on n'a pas obtenu un objet dont la taille se situe dans l'intervalle $[n(1-\varepsilon),n(1+\varepsilon)]$ (resp. tant que la sortie n'est pas de taille n).

Afin d'optimiser cette procédure de rejet, il convient de bien choisir le paramètre de contrôle x. Une stratégie raisonnable (mais ce n'est pas la seule) pour régler le paramètre x consiste à choisir

la valeur de x de sorte que l'espérance de la taille de la sortie vaille n, c'est-à-dire prendre comme paramètre la solution $x_n \in [0, \rho[$ de l'équation :

$$x\frac{C'(x)}{C(x)} = n.^{4}$$

Cette solution existe et est unique si on fait l'hypothèse raisonable que l'espérance de la taille tend vers l'infini et la variance reste strictement positive. Il est bon de noter que cette valeur x_n est aussi la valeur qui maximise la probabilité de tirer un objet de taille n (s'il en existe). En effet, un calcul simple montre que la fonction $x \mapsto \frac{c_n x^n}{C(x)}$ est maximum en x_n. Pour obtenir des résultats précis sur la complexité en temps du rejet, il nous faut préalablement, en suivant la classification introduite dans [40], répartir les distributions des tailles sous modèle de Boltzmann en trois types principaux : concentré (comme pour les classes de la forme $\textsc{Set}(\mathcal{A})$), plat (comme pour les langages rationnels) ou piqué (comme pour les langages hors contextes) dont voici les définitions précises :

Définition 2.5.1 – *Soit ρ la singularité dominante de la série génératrice. La distribution de Boltzmann d'une classe combinatoire $\mathcal{C}$ est* plate *(resp.* piquée*) si et seulement si sa série génératrice $C(z)$ est analytique en 0, a un rayon de convergence $\rho > 0$ et satisfait les deux conditions suivantes (appelées conditions de Δ-singularité) :*
(i) La série $C(z)$ admet ρ comme unique singularité sur le cercle $|z| = \rho$ et est prolongeable analytiquement dans le domaine $\Delta(r, \theta) = \{z \in \mathbb{C} | z \neq \rho, |z| < r, \arg(z - \rho) \in (-\theta, \theta)\}$, pour un certain $r > \rho$ et un certain θ satisfaisant $0 < \theta < \pi/2$;
(ii) Quand z tend vers ρ dans le Δ-domaine, $C(z)$ satisfait un développement singulier du type : $C(z) \sim_{z \to \rho^-} c_0(1 - z/\rho)^{-\alpha} + o((1 - z/\rho)^{-\alpha})$ avec $\alpha \in \mathbb{R}\backslash(-\mathbb{N})$ (resp. $\alpha \in \mathbb{R}^+$).
*La quantité $-\alpha$ est appelée l'*exposant singulier *de $C(z)$.*
– *Notons $\mu_1(x) = \mathbb{E}_x(N)$, $\mu_2(x) = \mathbb{E}_x(N^2)$ et $\sigma^2(x)$ la variance. La distribution de Boltzmann d'une classe combinatoire $\mathcal{C}$ est* concentrée *si et seulement si $\sigma(x)/\mu_1(x) \to 0$ quand $x \to \rho^-$.*

Pour chacune de ces distributions, le tableau 5 suivant donne la complexité algorithmique des générateurs de Boltzmann en taille approchée (resp. exacte).

Type	En taille exacte	En taille approchée
Concentrée	$o(n^2)$ [5]	$\mathcal{O}(n)$
Plate	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
Piquée	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$ (après pointage(s) de la classe)

TABLE 5: Complexité algorithmique des générateurs de Boltzmann suivant le type de distribution du modèle de Boltzmann

Pour conclure cette introduction, mentionnons qu'il existe des optimisations comme le rejet anticipé qui consiste essentiellement à ne pas attendre la fin de la génération d'un objet, mais d'interrompre le générateur dès lors que le nombre d'atomes déjà construits dépasse la taille que l'on s'est fixée. Une autre approche consiste à ne pas fixer le paramètre de Boltzmann de sorte à avoir l'espérance de la taille désirée, mais de fixer celui-ci (ceci est possible si la série est convergence en sa singularité dominante) de sorte à avoir une espérance de taille infinie. Cette technique est généralement appelée générateurs de Boltzmann singuliers. Elle présente de multiples intérêts. En particulier, elle est souvent plus rapide, couplable avec le rejet anticipé, permettant des optimisations dans le cas des schémas sur-critiques (technique du saut de puce) ...

2.6 Conclusion et perspectives

La méthode de Boltzmann ouvre de nombreuses questions qui ne sont pour le moment que partiellement résolues.

4. Les adorateurs de la méthode du point col remarqueront immédiatement le lien entre cette formule et leur formule fétiche.

Nous nous sommes en particulier intéréssé à mieux cerner le degré de généralité que peut avoir la méthode de Boltzmann, sa qualité à passer aux extensions (multivariés, système différentiel, autres formes de marquages, nouveaux constructeurs, etc). Nous avons très largement orienté notre recherche dans cette direction. Ces questions forment la trame du coeur de ce mémoire.

3 Extension de l'expressivité des générateurs

Il existe plusieurs voies pour étendre l'expressivité des générateurs de Boltzmann. Nous allons tout au long de ce mémoire explorer chacune d'elles :

1. Construire des générateurs de Boltzmann pour d'autres types de colorations. Les deux grands types de marquage (étiqueté et non étiqueté) peuvent en effet être inscrits comme nous l'avons montré précédement dans une vision plus générale des marquages. Cette approche sera développée dans la section 3.1, où nous donnerons une façon efficace de générer des objets dans une classe colorée [20]. Ce travail a été effectué avec Jacquot et a fait l'objet d'une présentation à Gascom'08.

2. Dans de nombreuses situations, la seule information de taille est très largement lacunaire et le problème réel est de générer des objets d'une certaine taille, mais aussi ayant certaines caractéristiques additionnelles. Ceci passe par une extension naturelle du cadre symbolique à une version paramétrée où plusieurs caractéristiques sont prises simultanément en compte. Nous avons donc proposé avec Ponty dans un article accepté à Aofa'10 [27] une extension de la génération de Boltzmann au domaine multi-paramétré avec une analyse précise dans le cadre des langages rationnels et hors-contextes (voir section 3.2).

3. Une autre direction consiste à ne plus regarder les classes combinatoires classiques et leurs séries génératrices ordinaires associées, mais les classes combinatoires multiplicatives (où l'on définit la taille d'un couple comme le produit des tailles de ses éléments, et non comme la somme) avec comme séries associées des séries de Dirichlet. Ces classes multiplicatives sont introduites dans la thèse de Hwang [50]. Cette extension de la génération sous modèle de Boltzmann permet de générer efficacement des classes que l'on rencontre en théorie des nombres (factorisation de branchement, factorisation ordonnée, composition et partition d'entiers, etc) où les séries de Dirichlet sont un outil fondamental (voir section 3.3).

4. Un opérateur central dans l'espace des classes combinateurs est bien sûr l'opérateur de pointage qui permet d'atteindre des classes combinateurs très importantes liées aux structures de données. Il nous est apparu comme crucial d'avoir un générateur de Boltzmann pour l'opérateur point, plusieurs travaux de Roussel, Soria [71], Bodini, Jacquot [21] et Bodini, Roussel, Soria [30] ont abordé cette question sous des angles variés. Nous décrivons dans la section 3.4 les résultats obtenus avec Roussel et Soria dans [30], où nous appliquerons cela à la génération de structures combinatoires différentielles du première ordre (dont les exemples les plus célèbres sont les arbres croissants)

5. Il nous est aussi apparu fondamental d'étendre l'automatisation aux plus grands nombres de constructeurs combinatoires de base. Dans l'article originel, seuls les constructeurs $+$, $\times$, Seq, Set, Cyc, ont été traités. Par la suite, Flajolet, Fusy et Pivoteau [42] ont proposé des générateurs de Boltzmann pour les constructeurs Set et Cyc dans le cas non-étiqueté. Nous avons continué ce processus de complétion en ajoutant les opérateurs Cycle bicoloré avec Jacquot [21] (non développé dans ce mémoire) et produit de Hadamard avec Gardy et Roussel [19] (voir section 3.5). Parallèlement, Darrasse, Panagiotou, Roussel, Soria ont trouvé un générateur de Boltzmann pour le mélange de deux langages rationnels [36].

3.1 Générateurs de Boltzmann pour les classes colorées

Les résultats exposés dans cette section ont fait l'objet d'une présentation par A. Jacquot à Gascom'08 [20].

Les générateurs de Boltzmann ont donc été décrits de manière assez exhaustive (c'est à dire pour la plupart des constructeurs classiques : $+$, $\times$, Seq, Set, Cyc) dans le cas des classes combinatoires étiquetées et non étiquetées [40, 42]. Dans cette partie, nous nous intéressons à la génération sous modèle de Boltzmann des objets combinatoires colorés tels que nous les avons définis dans l'introduction. Rappelons brièvement qu'un objet combinatoire a ayant n atomes est *coloré* si et seulement si chacun de ses atomes peut être coloré avec une couleur prise dans $\{1, \cdots, n\}$. Notre motivation ici est de pouvoir analyser le type suivant de schéma de construction : considérons que la construction d'un objet de taille n est distribuée (partagée) à n processeurs. Chaque processeur signe l'atome qu'il a construit. Un objet coloré est alors exactement une des façons possibles de construire un tel objet. Être capable de générer aléatoirement des classes colorées permettrait de mettre en lumière et conjecturer des propriétés à propos de ces objets. Notre approche consiste à commencer par générer de manière efficace des objets combinatoires k-colorés. Dans ce cas, chaque atome peut être coloré avec une couleur prise dans $\{1, \cdots, k\}$, la valeur k ne dépend plus ici de la taille (voir fig.2). Les classes k-colorées sont classiques. Les générateurs de Boltzmann introduits par Flajolet Fusy Pivoteau [42] permettent de les générer (il suffit pour cela de substituer $k\mathcal{Z}$ à $\mathcal{Z}$ dans la spécification). Ils apparaissent dans de nombreux problèmes comme les colliers k-colorés [46], les arbres d'expression avec k types de fonctions n-aires, les arbres k-colorés planaires, les chemins de Motzkin k-coloré [73], etc.

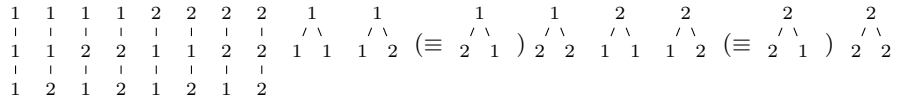


Figure 2: Les arbres généraux non-planaires 2-colorés de taille 3

La figure 2 met en exergue en quoi les symétries augmentent la difficulté pour générer uniformément les objets. Afin de générer plus efficacement [6] que dans [42] des objets k-colorés, nous introduisons une classe d'objets (les objets esquissés) intermédiaire entre les objets sans symétrie et les objets k-colorés. Un *objet esquissé* est par définition un proto-objet associé à une partition de ces atomes. Ce concept découle de la théorie de Pólya [5]. Cette structure forme un rehaussement (dans le sens où les k-colorations en sont des spécialisations) pour toutes les k-colorations. De manière informelle, les atomes appartenant à la même classe de la partition peuvent être considérés comme ayant la même couleur tandis que des atomes dans des classes disjointes peuvent être de couleur identique ou non. Nous montrons que pour tout k, un générateur "abstrait" pour les objets esquissés peut être spécialisé (par un choix judicieux des paramètres) pour devenir un générateur pour n'importe quels objets k-colorés. Dans un sens, ce modèle unifie toutes les k-colorations. En particulier, les générateurs pour les objets esquissés peuvent être adaptés de sorte à obtenir une génération efficace d'objets colorés (génération en taille approchée).

Ainsi les esquisses nous permettent de générer des objets colorés sous modèle de Boltzmann bien que la série génératrice ordinaire ne soit pas analytique en 0 (ce qui constitue une obstruction à la théorie classique de Boltzmann). Par exemple, soit $\mathcal{A}$ la classe combinatoire spécifiable non-étiquetée $\mathcal{A} = \text{Seq}(\mathcal{Z})$, il est bien connu que la série ordinaire vaut $1/(1-z)$. Dans ce cas, la série génératrice ordinaire pour la même classe mais dans le cadre des objets colorés vaut $\sum_{n\geqslant 0} n^n z^n$ qui est clairement non-analytique en 0. À première vue, il semble impossible d'obtenir un générateur de type Boltzmann pour cette classe avec ce type de marquage, puisque le paramètre x doit normalement être pris dans le disque de convergence de la série génératrice.

6. Dans un sens qui s'éclairera par la suite.

Classes combinatoires esquissées

Rappelons rapidement qu'un sous-ensemble S de $\mathcal{P}(A)$ (l'ensemble de tous les sous-ensembles de A) est une *partition* de A si et seulement si les parties de S sont disjointes, S ne contient pas $\varnothing$ et l'union des éléments de S est égale à A.

Un *objet combinatoire esquissé* a_σ est un proto-objet combinatoire a muni d'une partition σ de ses atomes $Atom(a)$. Cette partition est appelée *esquisse* et représente en un sens des symétries dans l'objet.

Remarque 3.1.1 *Dans la théorie classique de Pólya [5], on associe un automorphisme s sur $Atom(a)$, mais cette information est ici surdimensionnée par rapport à nos besoins. Nous ne désirons que marquer des atomes indistinguables, une partition est clairement suffisante à cette fin.*

Définition 3.1.2 *Une classe combinatoire esquissée est un multi-ensemble d'objets combinatoires esquissés pour lesquels il n'y a qu'un nombre fini d'objets de chaque taille.*

Maintenant, on peut définir comme pour les classes classiques des opérateurs de constructions, l'union, le produit et la réplication d'objets combinatoires esquissés. Il faut juste définir le sens que l'on donne aux esquisses :

Produit (concaténation). On définit alors l'esquisse σ_m de $m = (bc)$ comme $\sigma_b \cup \sigma_c$. En d'autres mots, $(bc)_{\sigma_b \cup \sigma_c}$.

Réplication. Le proto-objet est juste une k-répétition $\delta_k a = a \cdots a$.

L'esquisse de $\delta_k a$ est définie comme suit : soit $\{P_1, \cdots, P_m\}$ l'esquisse de a alors l'esquisse de $\delta_k a$ est $\{\bigcup_i P_1^i, \cdots, \bigcup_i P_m^i\}$ où P_1^i correspond à P_i mais sur les copies d'atomes de la i-ième occurrence de a.

Un exemple est sans doute plus parlant, soit

$$a_\sigma = (\langle[\square\square](\square\square)\rangle\square)_{\{\{1,3\},\{2,5\},\{4\}\}}$$

(le premier et le troisième $\square$ sont dans la même classe, ...), alors $\delta_2 a_\sigma$ vaut

$$(\langle[\square\square](\square\square)\rangle\square)(\langle[\square\square](\square\square)\rangle\square)_{\{\{1,3\}\cup\{6,8\},\{2,5\}\cup\{7,10\},\{4\}\cup\{9\}\}}$$

Une coloration d'un objet esquissé de taille n et d'esquisse σ est *consistante* si et seulement si sur chaque sous-ensemble de σ tous les atomes ont la même couleur. Nous dirons aussi qu'une esquisse est *c-admissible* pour une coloration c si et seulement si chaque partie de σ est monochrome.

3.1.1 Classes combinatoires esquissées spécifiables

On va bien sûr utiliser le même formalisme que pour la méthode symbolique (voir chapitre introductif). Les classes combinatoires esquissées sont exactement les classes que l'on peut construire à partir des règles suivantes [7] :

- **Classe atomique** : $\mathcal{Z} = \{\square\}$, l'esquisse est réduite à un singleton.
- **Union** : $\mathcal{A} \cup \mathcal{B}$ est le multi-ensemble formé de l'union des deux classes.
- **Produit** : $\mathcal{A} \times \mathcal{B} = \{(ab)_{\sigma_a \cup \sigma_b} | a \in \mathcal{A}, b \in \mathcal{B}\}$
- **Séquence** : $\textsc{Seq}(\mathcal{A}) = \bigcup_{i=0}^{\infty} \mathcal{A}^i$
- **Réplication** : $\Delta_k(\mathcal{A}) = \{\delta_k a | a \in \mathcal{A}\}$
- **Multi-ensemble** :
  $\textsc{Set}(\mathcal{A}) = \{\{B\} | B$ une concaténation des réplications des éléments dans $\mathcal{A}\}$
- **Cycle** :
  $\textsc{Cyc}(\mathcal{A}) = \{[B] | B$ une réplication d'éléments concaténés de $\mathcal{A}\}$

7. Nous faisons ici un abus important qu'il est bon de garder en mémoire : les classes combinatoires esquissées sont des proto-classes, leurs éléments sont des proto-objets.

Afin de pouvoir coder l'esquisse dans une série génératrice, nous allons devoir utiliser une infinité de variables aléatoires $s_1, \cdots, s_n, \cdots$; la variable s_i représentant un ensemble de i atomes appartenant à une même classe. Prenons l'exemple d'un objet composé de deux atomes, cet objet a potentiellement deux esquisses possibles $a_1 = (\square\square)_{\{\{1,2\}\}}$ et $a_2 = (\square\square)_{\{\{1\},\{2\}\}}$, le monôme associé à la première esquisse sera s_2 tandis que le monôme associé à la deuxième esquisse sera s_1^2.

Nous pouvons automatiquement construire les séries génératrices multivariées associées aux classes combinatoires esquissées grâce aux règles suivantes :

- **Classe neutre** : $F_{\mathcal{E}} = 1$
- **Classe atomique** : $F_{\mathcal{Z}} = s_1$
- **Union** : $F_{\mathcal{A}+\mathcal{B}} = F_{\mathcal{A}} + F_{\mathcal{B}}$
- **Produit** : $F_{\mathcal{A}\times\mathcal{B}} = F_{\mathcal{A}} \times F_{\mathcal{B}}$
- **Séquence** : $F_{\textsc{Seq}(\mathcal{A})} = \dfrac{1}{1 - F_{\mathcal{A}}}$
- **Réplication** : $F_{\Delta_k(\mathcal{A})}(s_1, s_2, \cdots) = F_{\mathcal{A}}(s_k, s_{2k}, \cdots)$
- **Multi-ensemble** : $F_{\textsc{Set}(\mathcal{A})} = \exp\left(\sum_{k\geqslant 1} \frac{1}{k} F_{\Delta_k(\mathcal{A})}\right)$
- **Cycle** : $F_{\textsc{Cyc}(\mathcal{A})} = \sum_{k\geqslant 1} -\frac{\varphi(k)}{k} \ln(1 - F_{\Delta_k(\mathcal{A})})$

Il est important de noter que $F_{\mathcal{A}}$ n'est pas une fonction génératrice énumérative pour les objets esquissés dans $\mathcal{A}$. Cette construction peut aussi faire penser à une approche multivariée telle que décrite dans [43] p165-166. Mais le concept est assez différent, cette fonction est construite de sorte à avoir les propriétés suivantes. En notant pour $t \in \mathbb{N}$, $f_{\mathcal{A}}(x,t) = F_{\mathcal{A}}(tx, tx^2, \cdots, tx^k, \cdots)$, un résultat classique de la théorie de Pólya nous assure que $[x^n] f_{\mathcal{A}}(x,t) = |\mathcal{A}_n(\mathcal{Z}_1 + \cdots + \mathcal{Z}_t)|$. La substitution de s_i par tx^i exprime donc que les paquets de i atomes (formant une même classe) représentés par s_i peuvent être colorés d'exactement t différentes façons et ils doivent être considérés globalement comme un objet de taille i (d'où le x^i). Il en découle que la fonction $f_{\mathcal{A}}(x,t)$ est la fonction génératrice (énumérative) ordinaire pour la classe combinatoire t-colorée $\mathcal{A}^{[t]}$. Dans un sens, la série génératrice $F_{\mathcal{A}}$ contient les informations de toutes les séries génératrices pour toutes les classes combinatoires t-colorées. En fait, ces séries génératrices multivariées ne sont autres que les sommes d'indices de cycle, bien connues en théorie de Pólya. Elles apparaissent aussi de manière fondamentale pour introduire la notion de point cyclique [31].

Donnons un exemple de ce que permet cette notion. Considérons $\mathcal{S} = \textsc{Set}(\mathcal{Z})$, on peut calculer par itération les premiers termes de la série génératrice des objets esquissés :

$$F_{\textsc{Set}(\mathcal{Z})} = 1 + s_1 + s_1^2/2 + s_2/2 + s_1^3/6 + s_1 s_2/2 + s_3/3 + \cdots$$

Maintenant pour obtenir la série génératrice énumérative pour les objets 3-colorés $\textsc{Set}(\mathcal{Z})$, il suffit de remplacer s_i par $3x^i$. On en déduit :

$$F_{\textsc{Set}(\mathcal{Z})} = 1 + 3x + 6x^2 + 10x^3 + \cdots$$

Si, l'on désire maintenant énumérer les objets 2-colorés, il suffit de remplacer s_i par $2x^i$, ce qui nous permet d'obtenir :

$$F_{\textsc{Set}(\mathcal{Z})} = 1 + 2x + 3x^2 + 4x^3 + \cdots$$

Remarque fondamentale : Dans les classes esquissées spécifiables, toutes les esquisses sont formées à partir des opérateurs de réplication. Afin que, lors de l'implémentation, il ne soit pas nécessaire de stocker la partition, nous allons simplement garder les réplications sous forme non-expansées. Ce choix induit un certain gain en mémoire, d'autant meilleur que les objets ont beaucoup de symétries.

Modèle de Boltzmann pour les classes esquissées

L'idée ici est d'avoir un relèvement des générateurs de Boltzmann des classes k-colorées au niveau abstrait des classes esquissées. À cette fin, nous définissons la distribution sous modèle de Boltzmann

pour la classe esquissée $\mathcal{A}$ comme suit : la probabilité d'obtenir un objet a_σ avec σ une partition ayant n_i classes de cardinal i vaut $\dfrac{[s_1^{n_1}s_2^{n_2}\cdots]\,F_{\mathcal{A}}\left(s_{(n)\in\mathbb{N}^*}\right)s_1^{n_1}s_2^{n_2}\cdots}{F_{\mathcal{A}}\left(s_{(n)\in\mathbb{N}^*}\right)}$.

Dès lors, si l'on désire un générateur de Boltzmann pour les classes t-colorées $\mathcal{A}(\mathcal{Z})$, il ne restera plus qu'à spécialiser notre générateur en prenant $s^i = tx^i$ et pour chaque classe p de σ, de tirer uniformément une couleur c_p dans $\{1,\cdots,t\}$ afin d'assigner aux atomes de p la couleur c_p.

Remarque 3.1.3 *En prenant $t = 1$, nous obtenons exactement les générateurs pour les classes combinatoires spécifiables non-étiquetées décrites dans [42] dont ils sont clairement inspirés.*

Classe $\mathcal{A}$	Description	$F_{\mathcal{A}}(s_1, s_2, \cdots)$	Générateur $\Gamma\mathcal{A}(s_1,\cdots)$
ε	Classe neutre	1	**return** ε
$\mathcal{Z}$	Classe atomique	s_1	**return** $\square$
$\mathcal{B}\times\mathcal{C}$	Produit	$F_{\mathcal{B}}(s_1,\cdots)\times F_{\mathcal{C}}(s_1,\cdots)$	**return** $(\Gamma\mathcal{B}(s_1,\cdots),\Gamma\mathcal{C}(s_1,\cdots))$
$\mathcal{B}+\mathcal{C}$	Union	$F_{\mathcal{B}}(s_1,\cdots)+F_{\mathcal{C}}(s_1,\cdots)$	**if** $Bern\left(\dfrac{F_{\mathcal{A}}(s_1,\cdots)}{F_{\mathcal{A}+\mathcal{B}}(s_1,\cdots)}\right)$ **then return** $\Gamma\mathcal{B}(s_1,\cdots)$ **else return** $\Gamma\mathcal{C}(s_1,\cdots)$
$\textsc{Seq}(\mathcal{A})$	Séquence	$\frac{1}{1-F_{\mathcal{A}}(s_1,\cdots)}$	$l := Geom(A(s_1,\cdots))$ **return** $\underbrace{(\Gamma\mathcal{A}(s_1,\cdots),\ldots,\Gamma\mathcal{A}(s_1,\cdots))}_{l \text{ fois}}$
$\Delta_k(\mathcal{A})$	Réplication	$F_{\mathcal{A}}(s_k,,s_{2k},\cdots)$	$A := \Gamma\mathcal{A}_{(s_k,s_{2k},\cdots)}$ **return** $\delta_k A$
$\textsc{Set}(\mathcal{A})$	Multi-ensemble	$F_{\textsc{Set}(\mathcal{A})} = \exp\sum_{k\geqslant 1}\frac{1}{k}F_{\Delta_k(\mathcal{A})}$	voir algorithme 1
$\textsc{Cyc}(\mathcal{A})$	Cycle	$F_{\textsc{Cyc}(\mathcal{A})} = \sum_{k\geqslant 1}-\frac{\varphi(k)}{k}\ln(1-F_{\Delta_k(\mathcal{A})})$	voir algorithme 2

TABLE 6: Règles pour construire un générateur de Boltzmann pour les classes combinatoires spécifiables esquissées

Algorithme 1 : $\Gamma\,\textsc{Set}(\mathcal{A})_{(s_1,s_2,\cdots)}$

Input : les paramètres $s_1, s_2, \cdots$
Output : un multi-ensemble
Soit K la variable aléatoire à valeurs dans $\mathbb{N}$ vérifiant $\mathbb{P}(K \leqslant k) = \prod_{j>k}\exp(-\frac{1}{j}F_{\mathcal{A}}(s_j, s_{2j}, \cdots))$

Tirer k selon la loi de K
$S \leftarrow \epsilon$
if $k > 0$ **then**
  for j **from** 1 **to** $k-1$ **do**
    Tirer q suivant une loi de Poisson de paramètre $\frac{1}{j}F_{\mathcal{A}}(s_j, s_{2j}, \cdots)$
    for i **from** 1 **to** q **do**
      $A_i \leftarrow \Gamma\Delta_j(\mathcal{A})_{(s_1,s_2,\cdots)}$
      $S \leftarrow Concat(S, A_i)$
  Tirer q suivant une loi de Poisson$_{\geqslant 1}$ de paramètre $\frac{1}{k}F_{\mathcal{A}}(s_k, s_{2k}, \cdots)$
  for i **from** 1 **to** q **do**
    $A_i \leftarrow \Gamma\Delta_k(\mathcal{A})_{(s_1,s_2,\cdots)}$
    $S \leftarrow Concat(S, A_i)$
  return $\langle S\rangle$

Algorithme 2 : $\Gamma\,\textsc{Cyc}(\mathcal{A})_{(s_1,s_2,\cdots)}$

Input : les paramètres $s_1, s_2, \cdots$
Output : un cycle

Soit K la variable aléatoire à valeurs dans $\mathbb{N}^*$ vérifiant $\mathbb{P}(K=k) = -\frac{1}{F_{Cyc(\mathcal{A})}}\frac{\varphi(k)}{k}\ \ln(1-F_{\Delta_k(\mathcal{A})})$
Tirer k selon la loi de K
Soit L une variable aléatoire à valeurs dans $\mathbb{N}^*$ vérifiant $\mathbb{P}(L=l) = -\frac{(F_{\Delta_k(\mathcal{A})})^l}{l}\frac{1}{\ln(1-F_{\Delta_k(\mathcal{A})})}$
Tirer l suivant la loi de L
$M \leftarrow \epsilon$
for i **from** 1 **to** l **do**
    $A_i \leftarrow \Gamma\mathcal{A}_{(s_k,s_{2k},\cdots)}$
    $M \leftarrow Concat(M, A_i)$
return $[\delta_k M]$

3.1.2 Générateur de Boltzmann en taille approchée pour les classes colorées

Revenons au problème de la génération des objets colorés, nous allons utiliser les générateurs d'objets esquissés pour résoudre cette question. Nous procédons de la manière suivante : considérons que l'on désire générer un objet coloré de taille n dans la classe combinatoire $\mathcal{A}$. En premier lieu, il faut résoudre l'équation suivante afin de calibrer le paramètre x_n qui régit la taille de la sortie :

$$x \cdot \frac{\frac{\partial}{\partial x} f_{\mathcal{A}}(x,n)}{f_{\mathcal{A}}(x,n)} = n.$$

Puis, nous pouvons obtenir un objet esquissé en utilisant le générateur de Boltzmann avec les paramètres $s_i = n \cdot x_n^i$. À ce stade, si nous colorions les objets avec n couleurs, le générateur ne serait uniforme que pour la classe $\mathcal{A}_n^{[n]}$, c'est à dire uniquement quand l'on a tiré un objet esquissé de taille n. On doit donc "débiaiser" le générateur quand la taille de l'objet tiré n'est pas exacte. L'idée pour rendre uniforme le générateur consiste à faire un rejet approprié.

Nous ne décrirons pas le détail de cette phase dans ce mémoire. Nous renvoyons le lecteur à l'article initial [20]. Nous obtenons finalement l'algorithme 3 dont la complexité est décrite dans le théorème ci-dessous :

Théorème 3.1.4 *L'algorithme 3 est un générateur de Boltzmann en taille approchée pour les objets colorés de la classe combinatoire $\mathcal{A}$. Son coût est en*

$$\mathcal{O}\left(n\left((1-\epsilon)\cdot\frac{f_{\mathcal{A}}(x_n,n)}{f_{\mathcal{A}}(x_n,(1-\epsilon)n)} + (1+\epsilon)^{(1+\epsilon)n}\cdot\frac{f_{\mathcal{A}}(x_n,n)}{f_{\mathcal{A}}(x_n,(1+\epsilon)n)}\right)\right)$$

Algorithme 3 : $\Gamma_{colored}$

Input : les paramètres n, x_n
Output : un objet coloré
1 Tirer un objet esquissé a avec les paramètres $s_i = n \cdot x_n^i$
2 Soit $\tilde{n} \leftarrow$ la taille de a
3 Soit $\tilde{n}_i$ le nombre de parties dans l'esquisse qui sont de cardinal i
4 **if** $\tilde{n} \notin [(1-\epsilon)n, (1+\epsilon)n]$ **then**
5 recommencer $\Gamma_{colored}$
6 **else**
7 **switch** $\tilde{n}$ **do**
8 **case** $\tilde{n} = n$
9 **return** une n-coloration de a
10 **case** $\tilde{n} < n$
11 Soit α le minimum dans $\{\sum n_i | \sum i n_i = \tilde{n}$ et $[s_1^{n_1} s_2^{n_2} \cdots] F_{\mathcal{A}}(s_1, s_2, \cdots) \neq 0\}$
12 Tirer X suivant une loi de Bernoulli de paramètre $\left(\frac{\tilde{n}}{n}\right)^{-\alpha+\sum \tilde{n}_i}$
13 **if** $X = 1$ **then**
14 **return** une $\tilde{n}$-coloration de a
15 **else**
16 recommencer $\Gamma_{colored}$
17 **case** $\tilde{n} > n$
18 Tirer X suivant une loi de Bernoulli de paramètre $\left(\frac{\tilde{n}}{n}\right)^{-\tilde{n}+\sum \tilde{n}_i}$
19 **if** $X = 1$ **then**
20 **return** une $\tilde{n}$-coloration de a
21 **else**
22 recommencer $\Gamma_{colored}$

Remarque 3.1.5 *Malheureusement, un calcul explicite du rapport* $\dfrac{f_{\mathcal{A}}(x_n, n)}{f_{\mathcal{A}}(x_n, (1+\epsilon)n)}$ *est actuellement hors de notre portée.*

Expérimentalement, il semble qu'il soit souvent plus judicieux de choisir $\tilde{n} > n$*. Dans ce cas, il paraît plus approprié de modifier la ligne 4 dans l'algorithme par*
(4') if $\tilde{n} \notin [n, (1+\epsilon)n]$ then recommencer $\Gamma_{colored}$ *ce qui permet d'éviter le cas ligne 10.*

3.1.3 Conclusion et perspectives

Nous avons dans cette section essayé de mettre en lumière que les objets esquissés sont un outil puissant et performant pour générer sous modèle de Boltzmann des objets k-colorés. C'est aussi une vision combinatoire de la notion de coloration qui nous a permis de construire des générateurs en taille approchée pour les objets colorés par l'ajout d'une phase de rejet. Nous avons été surpris que seule la connaissance du minimum dans $\{\sum n_i | \sum i n_i = \tilde{n}$ et $[s_1^{n_1} s_2^{n_2} \cdots] F_{\mathcal{A}}(s_1, s_2, \cdots) \neq 0\}$ soit nécessaire pour débiaiser la génération d'objets colorés.

Cette étude de la génération des objets colorés nous entraîne naturellement vers la question suivante : comment générer efficacement des objets colorés avec des contraintes sur les couleurs ? Par exemple, comment générer des mots sur l'alphabet $\{a, b\}$ ayant le même nombre de a et de b. Ce problème est le corps de la section suivante, consacrée à l'extension des générateurs de Boltzmann aux classes combinatoires multi-paramétrées. Dans la même veine, l'article [21] décrit la construction d'un générateur de Boltzmann pour le cycle équicoloré. Mais nous ne développerons pas ce résultat dans ce mémoire.

3.2 Générateurs de Boltzmann multi-paramétrés

Cette section repose sur les résultats publiés avec Y. Ponty dans l'article [27]. Ils sont le prolongement d'idées issues du groupe de A. Denise (LRI) qui visent à faire de la génération multiparamétrée par la méthode récursive. Comme nous l'avons précédemment mentionné, la génération aléatoire est au cœur de la simulation des jeux de données complexes. Elle apparaît significativement dans des domaines applicatifs comme les réseaux complexes (en biologie, Internet ou les réseaux sociaux), ou pour le test logiciel (validation, performance). Elle permet aussi de prédire le comportement d'algorithmes (complexités et cohérence statistique des résultats), de visualiser des propriétés limites (telles que les transitions de phases en physique statistique), de simuler des modèles réels (graphes aléatoires pour le web).

Dans l'ensemble de ces exemples complexes, une description monovariée est très largement insuffisante. Il est par exemple fondamental pour la modélisation du web de pouvoir modifier des paramètres comme le degré moyen du graphe. Pour le test logiciel, il est plus important d'explorer toutes les composantes fortement connexes du graphe de dépendance, plutôt que de chercher l'uniformité des données. Il est donc très naturel et utile d'étendre les générateurs de Boltzmann aux classes combinatoires multi-paramétrées. C'est à dire à des classes combinatoires où l'on distingue un certain nombre d'atomes. L'exemple le plus simple que l'on puisse proposer est la classe combinatoire des mots sur deux lettres, dont la spécification est $\textsc{Seq}(2\mathcal{Z})$. Si l'on veut distinguer les deux lettres, on peut décrire la classe par $\textsc{Seq}(\mathcal{Z}_1 + \mathcal{Z}_2)$ où $\mathcal{Z}_1$ et $\mathcal{Z}_2$ représentent respectivement les deux lettres. C'est autour de cette idée (décrite dans le petit livre violet [43]) que l'on va développer la notion de générateur de Boltzmann multi-paramétré.

Dans cette partie, nous noterons, s'il n'y pas de confusion, Σ l'ensemble des atomes distingués, $\mathcal{C}$ la classe combinatoire multi-paramétrée sur Σ, et n la taille des objets engendrés.

Classes combinatoires multi-paramétrées, fonctions génératrices multivariées, distributions pondérées

Nous allons rappeler ici sous une forme constructive équivalente la notion de *paramètre hérité* décrite dans [43] à la définition d'une classe combinatoire k-paramétrée spécifiable. Une *classe combinatoire k-paramétrée spécifiable* est construite à partir des règles suivantes :

- Soit c'est la classe vide ou l'une des k classes atomiques distinguées.
- Soit elle est inductivement construite à partir des constructeurs classiques : $+$, $\times$, Seq, Set, Cyc.
- Soit elle est construite à partir d'elle-même par une spécification récursive bien fondée (comme par exemple, $\mathcal{T} = \mathcal{Z}_1 + \mathcal{Z}_2 \times \mathcal{T}$).

Fonctions génératrices multivariées Nous privilégions ici une version multivariée, plutôt qu'une version pondérée (équivalente) introduite dans Denise, Roques, Termier, [38].

La fonction génératrice multivariée (dans les variables $z, \pi_1, \cdots, \pi_k$) ordinaire associée à une classe combinatoire k-paramétrée $\mathcal{C}$ est naturellement définie comme :

$$C(z, \pi_1, \cdots, \pi_k) = \sum_{n \geqslant 0} c_{n,p_1,\cdots,p_k} z^n \pi_1^{p_1} \cdots \pi_k^{p_k} = \sum_{n \geqslant 0} c_{n,\boldsymbol{p}} z^n \boldsymbol{\pi}^{\boldsymbol{p}}$$

où $c_{n,\boldsymbol{p}}$ est le nombre d'objets de $\mathcal{C}$ qui sont de taille n et qui ont p_i atomes de type i. Le vecteur $(p_1, \cdots, p_k)$ est la *composition* de l'objet.

Avec cette notion, nous pouvons sans peine obtenir des règles de construction pour les séries génératrices multivariées à partir de la spécification multi-paramétrée. Ces règles sont décrites dans le tableau 7.

La notion de fonction génératrice pondérée découle naturellement de cette définition par affectation des variables $\pi_1, \cdots, \pi_k$. Précisément, une pondération est un vecteur $\boldsymbol{\pi}$ qui permet d'assigner un poids positif $\pi_i \in \mathbb{R}^+$ à chaque type d'atomes $t_i \in \Sigma$ de sorte que la série pondérée (monovariée) est juste l'évaluation de la série multivariée en $\boldsymbol{\pi}$.

Finalement, ceci nous amène à introduire la notion de modèle de Boltzmann multi-paramétré (qui généralise le modèle de Boltzmann classique) par la distribution des objets de taille n et de composition $\boldsymbol{p}$ suivante :

$$\mathbb{P}_{x,\boldsymbol{\pi}}(w) = \frac{\boldsymbol{\pi}^{\boldsymbol{p}} x^n}{C(\boldsymbol{\pi}, x)}. \tag{1}$$

Proposition 3.2.1 *Soit N (resp. N_i) la variable aléatoire associée à la taille (resp. le nombre d'occurrences de l'atome t_i) d'un objet suivant une $(x, \boldsymbol{\pi})-$distribution de Boltzmann sur la classe $\mathcal{C}$. Alors les espérances de N et N_i sont liées aux dérivées partielles de la fonction génératrice multivariée $C(\boldsymbol{\pi}, z)$ par les formules suivantes :*

$$\mathbb{E}_{x,\boldsymbol{\pi}}(N) = \frac{x\frac{\partial C(\boldsymbol{\pi},x)}{\partial x}}{C(\boldsymbol{\pi}, x)} \qquad et \qquad \mathbb{E}_{x,\boldsymbol{\pi}}(N_i) = \frac{\pi_i \frac{\partial}{\partial \pi_i} C(\boldsymbol{\pi}, x)}{C(\boldsymbol{\pi}, x)} \tag{2}$$

Nous avons dorénavant tous les éléments pour étendre les générateurs de Boltzmann au cadre des classes combinatoires multi-paramétrées. Les règles de construction sont dévoilées dans le tableau 7.

Classes	Description	$C(\boldsymbol{\pi}, z)$	$\Gamma\mathcal{C}(\boldsymbol{\pi}, x)$
Epsilon	$\mathcal{C} = \{\varepsilon\}$	$C(\boldsymbol{\pi}, z) = 1$	ε
Atomes	$\mathcal{C} = \{t_i\}$	$C(\boldsymbol{\pi}, z) = \pi_{t_i} z$	t_i
Union	$\mathcal{C} = \mathcal{A} + \mathcal{B}$	$A(\boldsymbol{\pi}, z) + B(\boldsymbol{\pi}, z)$	$\text{Bern}\left(\frac{A(\boldsymbol{\pi},x)}{C(\boldsymbol{\pi},x)}, \frac{B(\boldsymbol{\pi},x)}{C(\boldsymbol{\pi},x)}\right) \longrightarrow \Gamma\mathcal{A}(\boldsymbol{\pi}, x) \mid \Gamma\mathcal{B}(\boldsymbol{\pi}, x)$
Produit	$\mathcal{C} = \mathcal{A} \times \mathcal{B}$	$A(\boldsymbol{\pi}, z) \times B(\boldsymbol{\pi}, z)$	$(\Gamma\mathcal{A}(\boldsymbol{\pi}, x), \Gamma\mathcal{B}(\boldsymbol{\pi}, x))$
Séquence	$\mathcal{C} = \textsc{Seq}(\mathcal{A})$	$(1 - A(\boldsymbol{\pi}, z))^{-1}$	$l := Geom(B(\boldsymbol{\pi}, x));$ $\underbrace{(\Gamma\mathcal{B}(\boldsymbol{\pi}, x), \cdots, \Gamma\mathcal{B}(\boldsymbol{\pi}, x))}_{l \text{ fois}}$
Multi-ensemble	$\textsc{Set}(\mathcal{A})$	$\exp\left(\sum_{m=1}^{\infty} \frac{1}{m} A(\boldsymbol{\pi}^m, z^m)\right)$	voir algorithme 4
Cycle	$\textsc{Cyc}(\mathcal{A})$	$\sum_{m=1}^{\infty} \frac{\varphi(m)}{m} \ln \frac{1}{1 - A(\boldsymbol{\pi}^m, z^m)}$	voir algorithme 5

TABLE 7: Séries génératrices multivariées et leurs générateurs de Boltzmann $\Gamma C(\boldsymbol{\pi}, x)$. (Pour les opérateurs de Pólya, $\boldsymbol{\pi}^m$ est une notation usuelle pour $\pi_1^m, \cdots, \pi_k^m$).

Algorithme 4 : $\Gamma\,\textsc{Cyc}(\mathcal{A})_{(x,\boldsymbol{\pi})}$

Input : les paramètres x, π
Output : un cycle [a]

Soit K la variable aléatoire à valeurs dans $\mathbb{N}^*$ vérifiant $\mathbb{P}(K = k) = -\frac{1}{F_{Cyc(\mathcal{A})}} \frac{\varphi(k)}{k} \ln\left(1 - F_{\Delta_k(\mathcal{A})}\right)$
Tirer k selon la loi de K
Soit L la variable aléatoire à valeurs dans $\mathbb{N}^*$ vérifiant $\mathbb{P}(L = l) = -\frac{(F_{\Delta_k(\mathcal{A})})^l}{l} \frac{1}{\ln(1 - F_{\Delta_k(\mathcal{A})})}$
Tirer l selon la loi de L
$M \leftarrow \epsilon$
for i **from** 1 **to** l **do**
    $A_i \leftarrow \mathcal{A}(\boldsymbol{\pi}^k, x^k)$
    $M \leftarrow Concat(M, A_i)$
return $[\delta_k M]$

a. Dans cet algorithme, $F_{\mathcal{X}}$ désigne la série génératrice de $\mathcal{X}$.

Algorithme 5 : $\Gamma\,\textsc{Set}(\mathcal{A})_{(x,\boldsymbol{\pi})}$

Input : les paramètres x, π
Output : un multi-ensemble
Soit K la variable aléatoire à valeurs dans $\mathbb{N}$ vérifiant $\mathbb{P}(K \leqslant k) = \prod_{j>k} \exp\left(-\frac{1}{j} A(\boldsymbol{\pi}^j, x^j)\right)$
Tirer k selon la loi de K
$S \leftarrow \epsilon$
if $k > 0$ **then**
  for j **from** 1 **to** $k-1$ **do**
    Tirer q suivant une loi de Poisson de paramètre $\frac{1}{j} A(\boldsymbol{\pi}^j, x^j)$.
    for i **from** 1 **to** q **do**
      $A_i \leftarrow \Gamma\Delta_j(\mathcal{A})_{(x,\boldsymbol{\pi})}$
      $S \leftarrow Concat(S, A_i)$
  Tirer q suivant une loi de Poisson$_{\geqslant 1}$ de paramètre $\frac{1}{k} A(\boldsymbol{\pi}^k, x^k)$
  for i **from** 1 **to** q **do**
    $A_i \leftarrow \Gamma\Delta_k(\mathcal{A})_{(x,\boldsymbol{\pi})}$
    $S \leftarrow Concat(S, A_i)$
  return $\langle S \rangle$

Composition, fréquence et tolérance

A partir de la notion de composition précédemment introduite, nous définissons naturellement la notion de fréquence des occurrences de chaque type d'atomes t_i dans l'objet $w \in \mathcal{C}$, par :

$$\boldsymbol{p}(w) := (|w|_{t_i}/n)_{i\in[1,k]}\,.$$

Notre principal objectif est de générer uniformément des objets $w \in \mathcal{C}$ ayant une fréquence *proche* d'une fréquence cible $\boldsymbol{f} \in [0,1]^k$.

Il nous faut en premier lieu définir plus explicitement la notion de proximité convenable dans ce contexte. Soit $\boldsymbol{\epsilon}$ un k-uplet de réels positifs et $\alpha \in \mathbb{R}^+$ un réel positif, un objet $w \in \mathcal{C}$ sera qualifié de $(\boldsymbol{\epsilon}, \alpha)$-acceptable si et seulement si sa fréquence appartient à l'intervalle de *tolérance* :

$$\forall i \in [1,k], \boldsymbol{p}(w)_i \in I(f_i, \epsilon_i, \alpha)$$

où $I(f, e, a) := [f - f^a n^{a-1} e, f + f^a n^{a-1} e]$. Cette définition capture le cas où l'on veut une composition exacte (fixée) en prenant $\alpha = 1$ et $\epsilon_i = 1/n$, pour tout $i \in [1,k]$.

La génération aléatoire multi-paramétrée a déjà fait l'objet de travaux dans le cadre de la méthode récursive sur la classe des langages rationnels et hors contexte sur k lettres [38]. Leur approche permet de générer des objets de taille n en $\Theta(n^k)$ (resp. $\Theta(n^{2k})$) opérations arithmétiques, pour les langages rationnels (resp. hors contexte).
Plus spécifiquement encore, utilisant certaines propriétés des séries holonomes, Bertoni, Massazza, Radicioni [6] proposèrent une optimisation en $\Theta(n)$ dans le cadre des langages rationnels sur deux lettres. Malheureusement, une extension directe de leur technique ne fournit qu'un algorithme en $\Theta(n^{k-1})$ quand on passe à des langages rationnels sur k lettres [69].

Notre programme suit la philosophie générale de la génération sous modèle de Boltzmann que l'on résume en les trois phases suivantes :

Phase I. Déterminer une pondération c'est-à-dire une affectation pour le vecteur $\boldsymbol{\pi}$ afin que le vecteur espérance pour la composition coïncide avec la composition désirée.

Phase II. Tirer un objet par la méthode de Boltzmann en rejetant les objets de taille inacceptable (voir section 3.2.2).

Phase III. Rejeter les objets de compositions non-désirées, jusqu'à l'obtention d'un objet de taille et composition acceptable.

Tolérance		Composition	
		Sans	$\Omega(\sqrt{n})$
Taille	Sans	$\mathcal{O}\left(n^{2+k/2}\right)$	$\mathcal{O}\left(n^2\right)$
	$\Theta(n)$	$\mathcal{O}\left(n^{1+k/2}\right)$	$\mathcal{O}\left(n\right)$

TABLE 8: Complexités en moyenne suivant les tolérances en taille de composition du générateur de Boltzmann k-paramétré dans le cas des langages rationnels et hors contexte de grammaire fortement connexe

Bien que les phases II et III soient distinguées dans ce schéma afin de simplifier l'analyse de complexité, nous les fusionnerons en une même phase de rejet dans une implémentation réelle de l'algorithme. La partie algorithmique de la génération aléatoire multivariée sous modèle de Boltzmann diffère très peu des travaux déjà effectués dans le cas monovarié, mais une analyse générale de la complexité algorithmique est dans le cas multivarié un véritable défi à relever. En effet, la complexité de la phase de rejet Phase III est profondément liée à une analyse complète des distributions limites des paramètres multivariés. En particulier, cette étude passe par une description fine des phénomènes de transition de phase en analyse asymptotique. Autant dire que dans cette section nous nous restreignons à l'étude de quelques situations particulières. Nous analyserons donc les cas suivants : le cas des grammaires rationnelles, le cas des grammaires hors contexte et quelques exemples où le théorème des quasi-puissances de Hwang s'applique. Nous obtenons dans le cas des grammaires, pour chaque combinaison de tolérances sur la taille et la composition, les complexités résumées dans la table 8.

3.2.1 Calibrage des poids (Phase I)

Commençons par se poser la question de la détermination du vecteur $\boldsymbol{\pi}$ tel que le rejet en taille et composition soit le plus efficace possible (Phase III). Nous proposons et explorons deux voies, les deux visent à obtenir une pondération pour laquelle l'espérance de la composition soit proche de la composition ciblée. La première approche utilise une itération de Newton, la seconde utilise une approximation asymptotique pour la valeur de z qui simplifie beaucoup la relation poids/fréquences.

Dans la suite, nous noterons par $\boldsymbol{\mu}(x, \boldsymbol{\pi})$ le vecteur espérance $(\mathbb{E}_{x,\boldsymbol{\pi}}(N_1), \cdots, \mathbb{E}_{x,\boldsymbol{\pi}}(N_k))$.

Calibrage par valeurs espérées. La méthode de Newton repose sur l'idée de faire des approximations linéaires itérées de sorte à estimer de plus en plus finement la valeur d'une racine d'un système d'équations. Cette approche est généralement très efficace dès lors que la valeur initiale n'est pas trop éloignée de la racine à évaluer. Ici, nous nous intéressons à la détermination de la racine unique $(z_0, \boldsymbol{\pi_f})$ du système $\boldsymbol{\mu}(z_0, \boldsymbol{\pi}) = n\boldsymbol{f}$. L'algorithme 6 est une version très légèrement modifiée de la méthode de Newton qui teste à chaque étape si l'approximation linéaire faite améliore ou pas l'estimation de la racine. Ce test échoue uniquement quand l'estimation courante est trop éloignée de la solution. Si tel est le cas, par dichotomie, une nouvelle cible est introduite entre la position courante et la cible originelle comme étape préliminaire. Atteindre cette cible nous rapproche donc de la solution et permet à terme d'appliquer la méthode de Newton classique.

Rappelons ici brièvement, la complexité de la méthode de Newton classique, Soit $\boldsymbol{f}$ et n la composition cible et la taille. Supposons que la matrice jacobienne $J(\mathbb{E}_{z_0}(\boldsymbol{\pi_f}))$ ne soit pas singulière [8], alors l'algorithme 6 renvoie $(z_0, \boldsymbol{\pi}_1)$ tel que l'espérance de la composition $\boldsymbol{\mu}(z_0, \boldsymbol{\pi}_1)$ satisfait :

$$||\boldsymbol{\mu}(z_0, \boldsymbol{\pi}_1) - n\boldsymbol{f}|| < \epsilon.$$

De plus, il existe un voisinage B de $(z_0, \boldsymbol{\pi_f})$ tel que, pour tout $\boldsymbol{\pi}_0 \in B$, l'algorithme 6 avec comme condition initiale $\boldsymbol{\pi}_0$ converge quadratiquement vers $\boldsymbol{\pi_f}$ (à savoir $\exists C > 1, \forall k \geqslant 0, ||\boldsymbol{\pi}_k - \boldsymbol{\pi_f}|| \leqslant C^{-2k}$ où $\boldsymbol{\pi}_{k+1} := J(\mathbb{E}_{z_0})^{-1}(\boldsymbol{\pi}_k) \cdot (n\boldsymbol{f} - \mathbb{E}_{z_0}(\boldsymbol{\pi}_k)) + \boldsymbol{\pi}_k$).

8. À savoir qu'il n'y a pas de dépendance entre les espérances du nombre de chaque atome.

Algorithme 6 : Traquer le poids

Input : les paramètres initiaux z_0 et $\boldsymbol{\pi}_0$, une composition $\boldsymbol{f}$, une taille n et une précision numérique ϵ

Output : les poids

Soit $\mathbb{E}_{z_0}$ la fonction de l'espace des poids dans $\mathbb{R}_+^k$ tel que $\mathbb{E}_{z_0}(\boldsymbol{\pi}) = \boldsymbol{\mu}(z_0, \boldsymbol{\pi})$
Soit $J(\mathbb{E}_{z_0}(\boldsymbol{\pi}))$ la matrice jacobienne $\mathbb{E}_{z_0}$ évaluée en $\boldsymbol{\pi}$
$\boldsymbol{\pi} := \boldsymbol{\pi}_0$
repeat
    end :=true ; $\boldsymbol{c} := n\boldsymbol{f}$; $N := ||\boldsymbol{c} - \mathbb{E}_{z_0}(\boldsymbol{\pi})||$
    while $N > \epsilon$ **do**
        $\boldsymbol{\pi}_{aux} := \boldsymbol{\pi}$
        $\boldsymbol{\pi} := J(\mathbb{E}_{z_0})^{-1}(\boldsymbol{\pi}) \cdot (n\boldsymbol{f} - \mathbb{E}_{z_0}(\boldsymbol{\pi})) + \boldsymbol{\pi}$
        if $N < ||\boldsymbol{c} - \mathbb{E}_{z_0}(\boldsymbol{\pi})||$ **then**
            $\boldsymbol{\pi} := \boldsymbol{\pi}_{aux}$; $\boldsymbol{c} := (\boldsymbol{c} + \mathbb{E}_{z_0}(\boldsymbol{\pi}))/2$; end :=false
until *end=true*;
return $\boldsymbol{\pi}$

Calibrage asymptotique. Étant donné que l'on vise généralement la génération d'objets de grande taille, une option naturelle consiste à résoudre le système asymptotique, c'est-à-dire le système des développement asymptotiques des équations. Dans le cas particulier des langages rationnels et hors contexte dont la grammaire est irréductible et apériodique et dont la série génératrice $C(\boldsymbol{\pi}, z)$ a $\rho(\boldsymbol{\pi})$ comme singularité dominante, nous obtenons le système suivant. Pour tout atome distingué (lettre) t de fréquence désirée f_t et quand z tend vers $\rho(\boldsymbol{\pi})$, nous avons :

$$\mathbb{E}_{z,\boldsymbol{\pi}}(N_t) \sim \tfrac{1}{2}\pi_t n \frac{\frac{\partial}{\partial \pi_t}\rho(\boldsymbol{\pi})}{\rho} = nf_t \quad \text{si } \rho(\boldsymbol{\pi}) \text{ est une singularité rationnelle,}$$
$$\mathbb{E}_{z,\boldsymbol{\pi}}(N_t) \sim -\pi_t n \frac{\frac{\partial}{\partial \pi_t}\rho(\boldsymbol{\pi})}{\rho} = nf_t \quad \text{si } \rho(\boldsymbol{\pi}) \text{ est une singularité algébrique.}$$

Remarque 3.2.2 *Considérons maintenant l'espérance $\mathbb{E}_n(N_t)$ du nombre d'atomes (lettres) t dans un objet de taille fixée n. Alors, on trouve dans [38] une estimation asymptotique identique pour $\mathbb{E}_n(N_t)$. Par conséquent, les poids calculés par notre méthode peuvent tout aussi bien être utilisés dans le cadre de la méthode récursive multivariée introduite dans [38].*

3.2.2 Efficacité du rejet sur la taille (Phase II)

À ce stade, nous supposons que le vecteur des poids $\boldsymbol{\pi}$ a été déterminé de sorte que la composition moyenne soit égale à la composition désirée. Nous devons maintenant étudier la génération de m objets. Dans le cadre des langages hors contexte sous la distribution pondérée $\boldsymbol{\pi}$, le problème a été précédemment étudié par Denise, Roques, Termier, [38] en ce qui concerne la méthode récursive, et un algorithme en $\mathcal{O}(m \cdot n)$ opérations arithmétiques a été proposé. Bien que d'apparente faible complexité, la croissance exponentielle des entrées numériques utilisées par cet algorithme accroît la complexité empirique à $\Theta(m \cdot n^2)$ en temps et $\Theta(n^2)$ en mémoire. Ceci est une motivation importante à l'analyse de la génération sous modèle de Boltzmann multi-paramétré.

Génération de Boltzmann pondérée

Rappelons que la génération de Boltzmann trouve sa puissance dans la capacité à relâcher les contraintes de taille et repose essentiellement sur le calcul de l'oracle (c'est à dire l'évaluation de la série génératrice en une valeur donnée). À une modification mineure, le calcul de l'oracle dans le cas multivarié est sensiblement identique à celui du cas monovarié, et les techniques classiques (itératives, méthodes de Newton, etc.) peuvent être adaptées sans peine. De même, les règles de construction des générateurs pour les classes multi-paramétrées sont aux ajustements près, identiques à celles des

générateurs monovariés (voir le tableau 7). Le principe de rejet sur la taille, lui aussi est similaire (voir l'algorithme 7).

Algorithme 7 : Algorithme de rejet $\Gamma_2\mathcal{A}(x, \boldsymbol{\pi}; n, \varepsilon)$

Input : les paramètres $x, \boldsymbol{\pi}$
Output : un objet de $\mathcal{A}$ de taille dans $I(n, \varepsilon) := [n(1-\varepsilon), n(1+\varepsilon)]$
repeat
  $\gamma := \Gamma A(\boldsymbol{\pi}, x)$
until $|\gamma| \in I(n, \varepsilon)$;
return (γ)

Analyses de complexité

Il nous faut comme dans le cadre classique, définir des types de distribution de Boltzmann assez généraux pour pouvoir appliquer des théorèmes. L'aspect multi-paramétré nous impose des définitions un peu plus fines que dans le cas monovarié. Les types définis sont néanmoins les extensions naturelles des types introduits dans l'article initial (concentré, plat et piqué).

Les distributions $\Pi-$concentrées. Ce sont là les distributions de Boltzmann les plus avantageuses. En effet, ce type correspond au cas où la distribution de Boltzmann se concentre autour de la valeur moyenne. Plus précisément, la distribution est dite $\Pi-$*concentrée* si la série génératrice multivariée vérifie les trois conditions suivantes :

- Π est un sous-ensemble compact de l'espace des paramètres pour lequel la série $C(\boldsymbol{\pi}, z)$ est analytique en $z = 0$ pour tout $\boldsymbol{\pi} \in \Pi$.
- Pour tout $\boldsymbol{\pi} \in \Pi$, l'espérance $\mathbb{E}_{x,\boldsymbol{\pi}}(N)$ tend vers l'infini quand x tend vers la singularité dominante (par valeurs inférieures).
- Soit $\sigma_{x,\boldsymbol{\pi}}(N)$ l'écart type, la fonction $\eta(x) := \sup_{\pi\in\Pi} \left\{ \frac{\sigma_{x,\pi}(N)}{\mathbb{E}_{x,\pi}(N)} \right\}$ tend vers 0 quand x tend vers la singularité (ceci est équivalent par compacité de Π à $\forall \pi \in \Pi$, $\frac{\sigma_{x,\pi}(N)}{\mathbb{E}_{x,\pi}(N)}$ tend vers 0).

Théorème 3.2.3 *Soit $\boldsymbol{\pi} \in \Pi$ et x_n la racine dans $(0, \rho_\pi)$ de $\mathbb{E}_{x,\boldsymbol{\pi}}(N) = n$. Sous les hypothèses de $\Pi-$concentration, le générateur en taille approchée $\mu\mathcal{C}(x_n, \pi; n, \varepsilon)$ aboutit, en moyenne, après $\frac{1}{1-\left(\frac{\eta(x_n)}{\varepsilon}\right)^2}$ essais (de plus cette valeur moyenne tend vers 1 quand $n \to \infty$). En particulier, si $\mathcal{C}$ est spécifiable, alors le coût total d'un générateur en taille approchée est en $\mathcal{O}(n)$ en moyenne.*

Il faut maintenant analyser la situation dans le cas d'autres types de distributions. Nous allons qualifier une classe combinatoire de bien-conditionnée si et seulement si l'exposant singulier $\alpha_{\boldsymbol{\pi}}$ de sa singularité dominante est négatif. Suivant les travaux de Duchon, Flajolet, Louchard, Schaeffer [40], on observe que toute grammaire peut être pointée suffisamment de fois jusqu'à ce que l'exposant singulier de la série génératrice devienne négatif. Comme l'opérateur point laisse inchangée la distribution pondérée sur les sous-classes des objets de taille donnée, il s'ensuit que l'on peut restreindre notre analyse aux seules classes pondérées dont la distribution de Boltzmann est plate.

Théorème 3.2.4 (Essentiellement prouvé dans [40]) *Soit $\mathcal{C}_{\boldsymbol{\pi}}$ une classe combinatoire bien conditionnée et x_n la racine dans $[0, \rho_{\boldsymbol{\pi}})$ de $\mathbb{E}_{x,\boldsymbol{\pi}}(N) = n$. Alors la complexité $X_\varepsilon[n]$ du générateur $\Gamma_2\mathcal{C}(x_n, \boldsymbol{\pi}; n, \varepsilon)$ décrit par l'algorithme 7 est telle que :*

a) Pour un tirage en taille exacte ($\varepsilon = 0$) : $X_\varepsilon[n] \in \mathcal{O}\left(\frac{\kappa\Gamma(\alpha_{\boldsymbol{\pi}})n^2}{\alpha_{\boldsymbol{\pi}}^{\alpha_{\boldsymbol{\pi}}}} + c(\boldsymbol{\pi})n\right)$

b) Pour un tirage en taille approchée ($\varepsilon > 0$) : $X_\varepsilon[n] \in \mathcal{O}\left(\frac{\kappa n}{\zeta_{\alpha_{\boldsymbol{\pi}}}(\varepsilon)} + c(\boldsymbol{\pi})\right)$

où κ est le coût intrinsèque du générateur, $\alpha_{\boldsymbol{\pi}}$ est l'exposant singulier de $C(\boldsymbol{\pi}, z)$,

$$\zeta_{\alpha_{\boldsymbol{\pi}}}(\varepsilon) := \frac{\alpha_{\boldsymbol{\pi}}^{\alpha_{\boldsymbol{\pi}}}}{\Gamma(\alpha_{\boldsymbol{\pi}})} \int_{-\varepsilon}^{\varepsilon} (1+s)^{\alpha_{\boldsymbol{\pi}}-1} e^{-\alpha_{\boldsymbol{\pi}}(1+s)} \mathrm{d}s,$$

$\Gamma(x)$ est la fonction gamma, et $c(\boldsymbol{\pi})$ est conjecturé être en $\mathcal{O}(n)$.

Il faut noter que ce théorème est un théorème local, en ce sens que l'on a une complexité pour chaque pondération $\boldsymbol{\pi}$. Ce théorème dit donc que pour toute pondération donnée on a une complexité en $\mathcal{O}(n)$ (resp. $\mathcal{O}(n^2)$) pour une génération en taille approchée (resp. en taille exacte). Maintenant, un esprit malin pourrait se plaire à faire dépendre $\boldsymbol{\pi}$ de n. Bien que l'on ne soit pas capable de caractériser précisément l'impact de $c(\boldsymbol{\pi})$ dans les deux complexités, nous sommes convaincus que le biais reste limité et conjecturons donc que les complexités trouvées restent valides pour toute composition ciblée *qui a du sens*. Par exemple, si la pondération $\boldsymbol{\pi}$ assure dans la composition moyenne l'existence d'au moins un atome de chaque type.

Dans le cas particulier des classes de langages rationnels, le théorème suivant procure une évaluation explicite pour l'efficacité de la génération en taille approchée. Il repose sur la décomposition en éléments simples de la série génératrice rationnelle associée au langage. Celle-ci peut être obtenue pour n'importe quelle série génératrice pondérée $C(\boldsymbol{\pi}, z)$, et a la forme suivante

$$C(\boldsymbol{\pi}, z) = \sum_{i=1}^{r} \sum_{k=1}^{m_i} (1 - z/\rho_i)^{-\alpha_{i,k}} h_{i,k} + P(z) \tag{3}$$

où $P(z)$ est un polynôme de degré borné, r le nombre de singularités distinctes et m_i la multiplicité de ρ_i triés par module croissant. Dans les séries génératrices pondérées les valeurs de ρ_i, $P(z)$, $h_{i,k}$, k et r dépendent de $\boldsymbol{\pi}$.

Théorème 3.2.5 *Soit $\mathcal{C}_{\boldsymbol{\pi}}$ un langage rationnel, x_n la racine dans $[0, \rho_{\boldsymbol{\pi}})$ de $\mathbb{E}_{x,\boldsymbol{\pi}}(N) = n$ et $\varepsilon > 0$ la tolérance, alors le générateur en taille approchée $\Gamma_2\mathcal{C}(x_n, \boldsymbol{\pi}; n, \varepsilon)$ s'achève après le nombre de rejets de $\Gamma C_{\boldsymbol{\pi}}(x, b)$ suivant :*

$$\frac{C(\boldsymbol{\pi}, x_n)}{\left(\sum_{i=1}^{r} \sum_{k=1}^{m_i} \binom{n+k-1}{k-1} (\rho_i)^{-n} h_{i,k} + [z^n]P(z)\right)(x_n)^n}.$$

3.2.3 Complexité du rejet sur les compositions (Phase III)

Principe général

Là encore, le schéma consiste à procéder par rejet en écartant les objets dont la composition n'est pas acceptable.

Cela donne le générateur avec rejet $\Gamma_3\mathcal{A}(x, \boldsymbol{\pi}; n, \boldsymbol{m}, \varepsilon, \sigma)$ pour une classe $\mathcal{A}$ où x est un réel, $\boldsymbol{\pi}$ un k-vecteur, $\boldsymbol{m}$ une fonction de $\mathbb{N}$ dans $\mathbb{R}^k$, et ε une tolérance :

Algorithme 8 : $\Gamma_3\mathcal{A}(x, \boldsymbol{\pi}; n, \boldsymbol{m}, \varepsilon, \sigma)$

Input : les paramètres $x, \boldsymbol{\pi}, n, \boldsymbol{m}, \varepsilon, \sigma$
Output : un objet de $\mathcal{A}$ de taille s dans $I(n, \varepsilon)$
et pour tous les atomes π_i, le nombre d'occurrences de Z_i est dans $I(m_i(s), \varepsilon, \sigma) := [m_i(s) - m_i(s)^\sigma \varepsilon, m_i(s) + m_i(s)^\sigma \varepsilon]$
repeat
  $\gamma := \Gamma_2\mathcal{A}(x, \boldsymbol{\pi}; n, \varepsilon)$
until $\forall i, |\gamma|_i \in I(m_i(s), \varepsilon, \sigma)$;
return (γ)

Dans de nombreux cas importants de classes combinatoires, la distribution de la composition d'un objet est concentrée autour de sa moyenne. Il s'ensuit qu'un générateur fondé sur un principe de rejet a des chances d'aboutir après assez peu d'essais, dès lors que la composition désirée coïncide avec

la composition moyenne. Nous dégageons dans ce qui suit deux familles importantes pour lesquelles nous pouvons assurer de manière précise l'efficacité de notre générateur. Ces familles sont l'ensemble des langages rationnels de grammaire fortement connexe et la famille des langages hors contexte irréductibles et simples. Nous montrons que l'on peut engendrer un mot de composition exacte au bout de $\mathcal{O}\left(n^{k/2}\right)$ tentatives en moyenne. De plus, en acceptant un intervalle de tolérance pour la composition de n^β $(\beta > 1/2)$ sur le nombre d'occurrences de chaque lettre, notre générateur renvoie un objet acceptable après un nombre constant de tentatives.

La suite est dédiée au développement rigoureux de ces idées. Notons $U_n(\boldsymbol{\pi}_0)$ la variable aléatoire k-multivariée qui suit la loi $\mathbb{P}(U_n(\boldsymbol{\pi}_0) = \boldsymbol{a}) = \frac{\boldsymbol{\pi}_0^{\boldsymbol{a}} \cdot [z^n \boldsymbol{\pi}^{\boldsymbol{a}}] C_{\boldsymbol{\pi}}(z)}{[z^n] C_{\boldsymbol{\pi}_0}(z)}$, à savoir la distribution des paramètres pour les objets de taille n. De plus, nous désignons par $\boldsymbol{\mu}(n, \boldsymbol{\pi}_0)$ le vecteur moyen de $U_n(\boldsymbol{\pi}_0)$ et par $\boldsymbol{V}(n, \boldsymbol{\pi}_0)$ sa matrice de variance-covariance. S'il n'existe pas de corrélation stricte entre les paramètres, la matrice $\boldsymbol{V}(n, \boldsymbol{\pi}_0)$ est définie positive (et donc inversible). On peut alors définir une norme sur les vecteurs composition par $\boldsymbol{u}$ comme $||\boldsymbol{u}||_{\boldsymbol{V}^{-1}} = \sqrt{\boldsymbol{u}^T \boldsymbol{V}(n, \boldsymbol{\pi}_0)^{-1} \boldsymbol{u}}$. Maintenant, soit $\boldsymbol{V}$ une matrice définie positive, nous désignons par $\kappa(\boldsymbol{V}) := \inf_{||\boldsymbol{u}||_\infty = 1} \{||\boldsymbol{u}||_{\boldsymbol{V}}\}$, l'infinum des distances [9] entre la sphère unité et le centre de l'espace de Banach.

Définition 3.2.6 *La* condition de σ-concentration *est définie par :*

$$\limsup_{n \to \infty} \left(||\boldsymbol{\mu}(n, \boldsymbol{\pi})||_\infty\right)^\sigma \cdot \kappa\left(\boldsymbol{V}(n, \boldsymbol{\pi})^{-1}\right) = c > \frac{\sqrt{k}}{\varepsilon}.$$

Les deux théorèmes suivants montrent que le nombre de rejet moyen peut être exprimé en fonction de la croissante de l'"écart-type". En particulier, en cas de bonne concentration, le théorème 3.2.7 montre que le nombre de rejets moyen est constant. Le théorème 3.2.8 pour sa part, donc une estimation du nombre de rejets pour un tirage en taille exacte dans le cas d'une distribution limite gaussienne.

Théorème 3.2.7 (Composition approchée) *Soit x_n et $\boldsymbol{\pi}_{\boldsymbol{a}}$ la solution de*

$$\mathbb{E}_{x, \boldsymbol{\pi}}(N) = n \text{ et } \mathbb{E}_{x, \boldsymbol{\pi}}(N_i) = a_i.$$

La fonction $\boldsymbol{m}$ est définie par $\boldsymbol{m} : s \mapsto \mathbb{E}_{s, \boldsymbol{\pi}_{\boldsymbol{a}}}(N_i)$. De plus, faisons l'hypothèse que la condition de σ-concentration est vérifiée pour un $\sigma \leqslant 1$.
Alors le nombre moyen d'essais (en $\Gamma_2\mathcal{A}(x_n, \boldsymbol{\pi}; n, \varepsilon)$) du générateur à composition approchée

$$\Gamma_3\mathcal{C}(x_n, \boldsymbol{\pi}_{\boldsymbol{a}}; n, \boldsymbol{m}, \varepsilon, \sigma)$$

est borné supérieurement par

$$\sup_{s \in I(n, \varepsilon)} \frac{\left(\varepsilon \cdot \kappa\left(\boldsymbol{V}(s, \boldsymbol{\pi}_{\boldsymbol{a}})^{-1}\right) \cdot ||\boldsymbol{\mu}(s, \boldsymbol{\pi}_{\boldsymbol{a}})||_\infty^\sigma\right)^2}{\left(\varepsilon \cdot \kappa\left(\boldsymbol{V}(s, \boldsymbol{\pi}_{\boldsymbol{a}})^{-1}\right) \cdot ||\boldsymbol{\mu}(s, \boldsymbol{\pi}_{\boldsymbol{a}})||_\infty^\sigma\right)^2 - k}$$

qui tend vers une constante quand $n \to \infty$.

Théorème 3.2.8 (Composition exacte) *Considérons que $(U_n(\boldsymbol{\pi}_{\boldsymbol{a}}))$ converge en loi vers une distribution gaussienne multidimensionnelle de moyenne $\boldsymbol{\mu}$ et de matrice de variance-covariance $\boldsymbol{V}$ proportionnelle à $f(n)$. Alors le générateur en composition exacte $\Gamma_3\mathcal{C}(x_n, \boldsymbol{\pi}_{\boldsymbol{a}}; n, \boldsymbol{m}, 0, 1)$ s'achève après un nombre moyen de tentatives en*

$$\mathcal{O}\left((2\pi)^{k/2} (\det(\boldsymbol{V}))^{1/2}\right) = \mathcal{O}\left(f(n)^{k/2}\right).$$

9. Rappelons que la norme infinie est définie par $||\boldsymbol{u}||_\infty = \max(|u_1|, \cdots, |u_k|)$.

Langages rationnels : le théorème de Bender-Richmond-Williamson [2]

Le théorème de Bender-Richmond-Williamson [2, Théorème 1] donne des conditions suffisantes pour que la distribution limite des paramètres d'un langage rationnel $\mathcal{R}$ suive une distribution gaussienne multidimensionnelle. Rappelons qu'un langage rationnel est *irréductible* si et seulement si son automate minimal $\mathcal{A}$ est fortement connexe, et apériodique (à savoir si les longueurs des cycles dans l'automate $\mathcal{A}$ ont un pgcd égal à 1). Il faut de plus que le réseau de périodicité des paramètres Λ, défini dans [2] (Définition 2) ait la dimension de l'espace des paramètres (nous dirons par la suite qu'il est *plein*) pour éviter des corrélations triviales entre les occurrences des lettres.

Théorème 3.2.9 *Soit $\mathcal{R}_{\boldsymbol{\pi}}$ un langage rationnel pondéré dont la grammaire est minimale irréductible et apériodique, et x_n la racine dans $[0, \rho_{\boldsymbol{\pi}})$ de $\mathbb{E}_{x,\boldsymbol{\pi}}(N) = n$. Considérons que le réseau de périodicité des paramètres Λ soit plein. Alors :*
a) $\forall \sigma > 1/2$, $\forall \varepsilon > 0$, le générateur en taille approchée $\Gamma_3\mathcal{R}(x_n, \boldsymbol{\pi}; n, \varepsilon, \sigma)$ s'achève après $\mathcal{O}(1)$ essais en moyenne.
b) Pour $\sigma = 1/2$, $\exists \varepsilon_0 > 0$, $\forall \varepsilon > \varepsilon_0$ le générateur en composition approchée $\Gamma_3\mathcal{R}(x_n, \boldsymbol{\pi}; n, \varepsilon, \sigma)$ achève après $\mathcal{O}(1)$ tentatives en moyenne
c) Le générateur en composition exacte $\Gamma_3\mathcal{R}(x_n, \boldsymbol{\pi}; n, 0, 1)$ s'achève après $\mathcal{O}\left(n^{k/2}\right)$ essais en moyenne.

Arrêtons-nous un instant pour discuter des pré-requis du théorème 3.2.9. Si la matrice $\boldsymbol{M}$ n'est pas apériodique, alors il existe une puissance d tel que $\boldsymbol{M}^d$ est apériodique. Donc, on peut toujours ramener le problème à une liste de d problèmes où les classes sont apériodiques, et le théorème 3.2.9 s'applique sous les mêmes hypothèses. L'*irréductibilité* n'est pas nécessaire dès lors qu'une composante fortement connexe domine asymptotiquement, à savoir quand le schéma associé ne fait intervenir que des compositions sous-critiques et super-critiques (ici le sens de composition est celui de [43, Théorème IX.2]). Cependant le cas des compétitions entre différentes composantes fortement connexes dans un langage non-irréductible tient vraiment du challenge et requiert un sérieux développement qui est en cours d'étude. Finalement nous désirons mentionner que moyennant de mineures modifications des résultats similaires peuvent être étendus aux cadres plus généraux des matrices de transfert.

Langages hors contextes : le théorème de Drmota [39]

Un théorème de Drmota [39] donne des conditions suffisantes très similaires pour la distribution limite du vecteur composition dans le cas des langages hors contexte ce qui nous permet aussi d'appliquer le théorème 3.2.8. Plus précisément, nous retrouvons la condition d'irréductibilité (à savoir le graphe de dépendance de la grammaire doit être fortement connexe). Les hypothèses sur le réseau des paramètres et l'apériodicité sont remplacées par un concept très similaire, la grammaire est dite de *type simple* qui de manière grossière repose sur l'existence d'un cône de dimension $k+1$ centré sur l'origine $\mathbf{0}$ de l'espace des coefficients.

Théorème 3.2.10 *Soit $\mathcal{C}_{\boldsymbol{\pi}}$ un langage hors contexte pondéré obtenu à partir d'une grammaire de type simple $\mathcal{G}$ ([39, Theorem 1]) et dont le graphe de dépendance est fortement connexe. Alors les complexités décrites dans le théorème 3.2.9 sont valides pour $\mathcal{C}_{\boldsymbol{\pi}}$.*

Similairement au cas rationnel, la condition de forte connexité peut être relâchée pour les grammaires dont le comportement est dominé par une unique composante fortement connexe. Une caractérisation formelle de ces grammaires repose sur la notion de schéma (sous/super)-critique (voir [43, théorème IX.2]).

3.2.4 Conclusion et perspectives

Dans cette section, nous avons donc adapté et généralisé la méthode de Boltzmann au cadre des classes paramétrées. Sous des conditions explicitées et naturelles sur les langages rationnels et hors contexte, nous obtenons des générateurs ayant une complexité en $\mathcal{O}(n^{2+k/2})$ pour une génération en taille et composition exacte, ce qui surpasse les meilleurs algorithmes connus jusqu'à présent (en $\mathcal{O}(n^k)$ et $\mathcal{O}(n^{2k})$ respectivement pour les langages rationnels et hors contexte). Mais bien au delà,

nous montrons qu'un léger relâchement des contraintes sur la taille et la composition (relâchement en $\Omega(\sqrt{n})$) nous permet d'attendre un générateur de complexité moyenne linéaire, ce qui rend accessibles un grand nombre de problèmes de génération inenvisageables jusqu'à présent.

Cette section ouvre un vaste domaine de recherche autour de l'analyse des générateurs de Boltzmann multi-paramétrés. Elle n'est qu'une petite brique de base exhibant quelques classes pour lesquelles une analyse est réalisable. Néanmoins, un grand nombre de questions restent ouvertes, par exemple sur les liens entre des pondération et des fréquences "raisonnables" qui nous permettraient d'achever dans un cadre général l'analyse de la phase de rejet sur la taille.

La philosophie intrinsèque de cette section est que la génération sous modèle de Boltzmann multi-paramétré est efficace dès lors que la distribution des paramètres est bien concentrée.

Ce raffinement par l'ajout de paramètres nous amène naturellement à nous poser la question de la nature de ces paramètres et en particulier, de la manière dont il convient de leur associer une taille. Prenons l'exemple très simple de la factorisation des entiers. On peut par factorisation associer de manière univoque à un entier strictement positif sa factorisation ; en d'autres termes $\mathbb{N}^* = \textsc{Set}(\mathcal{P})$ où $\mathcal{P}$ désigne l'ensemble des nombres premiers. Néanmoins, il semble ici plus pertinent de définir la taille d'un ensemble de nombres non pas comme la somme des tailles telles que nous l'avons définie précédemment, mais comme le produit des tailles des composantes. Cette approche est le sujet d'étude de la section suivante.

3.3 Générateurs de Dirichlet, version multiplicative des générateurs de Boltzmann

La méthode symbolique classique traite des classes combinatoires construites à partir d'opérateurs simples ($+, \times, \textsc{Seq}, \textsc{Set}, \textsc{Cyc}$). Une caractéristique fondamentale est que la taille d'un n-uplet (ou d'un multi-ensemble de n éléments, ou d'un cycle) est défini comme la somme des tailles des éléments qui le constituent. Très naturellement, il apparaît que définir la taille d'un n-uplet comme le produit des éléments qui le constituent a aussi du sens. En particulier, cette variante se prête bien aux problèmes liés à la théorie des nombres où un nombre peut être interprété comme un multi-ensemble de nombres premiers (et dans ce cas, il faut définir la taille de manière multiplicative). Des structures très intéressantes, comme les factorisations à branchement sont exprimables grâce à ces classes combinatoires multiplicatives.

Dans cette section, nous présentons comment construire automatiquement (à partir de la spécification) des générateurs aléatoires pour des structures combinatoires spécifiables dont la taille est comptée de façon multiplicative (par la suite, nous désignerons ces classes comme les *classes combinatoires multiplicatives*). Cette variante a été pour la première fois introduite par HK Hwang dans sa thèse [50]. Notre objectif consiste à adapter la méthode de Boltzmann au cadre des classes combinatoires multiplicatives de sorte à générer des objets combinatoires de façon efficace en respectant une équiprobabilité de tirage pour des éléments de même taille.
A l'instar du modèle de Boltzmann, les *générateurs aléatoires de Dirichlet* reposent fondamentalement sur des séries génératrices. Néanmoins, ici, les séries génératrices seront des séries de Dirichlet. Nous verrons en effet que les séries de Dirichlet ont la propriété importante du *morphisme de produit* : le produit de deux séries de Dirichlet associées à deux classes combinatoires multiplicatives $\mathcal{A}$ et $\mathcal{B}$ est la série de la classe produit $\mathcal{A} \times \mathcal{B}$. De manière similaire au modèle de Boltzmann, les générateurs de Dirichlet dépendent aussi d'un paramètre de contrôle s réel : la distribution de la taille des objets générés à partir d'une classe combinatoire multiplicative $\mathcal{A}$ suivra un modèle de Dirichlet si la probabilité de tirer un objet γ de taille $|\gamma|$ est proportionnelle à $|\gamma|^{-s}$ (dans le modèle de Boltzmann, cette probabilité s'exprimait proportionnellement à $x^{|\gamma|}$). La taille de sortie du générateur n'est donc pas fixée, mais les objets de même taille sont tous générés avec la même probabilité.

Comme nous l'avons mentionné, l'énumération des classes combinatoires multiplicatives est très naturelle en utilisant les séries de Dirichlet : $\sum_{n\geqslant 1} a_n n^{-s}$. Nous verrons que l'étude de la complexité des générateurs de Dirichlet nécessite de manipuler assez finement ces séries.
Dans le premier paragraphe, nous allons présenter la notion de classes combinatoires multiplicatives spécifiables ainsi que les séries génératrices de Dirichlet qui leur sont associées. Nous présenterons les algorithmes de génération aléatoire. Le dernier paragraphe sera consacré à une étude de la complexité des générateurs de Dirichlet.

3.3.1 Classes combinatoires multiplicatives spécifiables et séries de Dirichlet

Les classes combinatoires multiplicatives spécifiables

Les classes combinatoires multiplicatives ne différent que d'un point des classes combinatoires classiques. Soit α et β deux objets combinatoires de tailles $|\alpha|$ et $|\beta|$, pour les classes combinatoires multiplicatives, on définit $|(\alpha, \beta)| = |\alpha| \cdot |\beta|$. En d'autres termes, seule la définition de taille des objets de la classe produit est transformée. Cette modification se répercute bien évidement sur les notions de Seq, Set et Cyc.
Cette très légère modification va néanmoins profondément modifier la combinatoire des classes, c'est à dire la manière dont s'agencent et se construisent les objets. En particulier, les objets de poids 1 qui jouaient un rôle si particulier dans les classes combinatoires classiques, vont prendre le rôle d'objets neutres. De cela découle la nécessité de définir de nouveaux "atomes", ces nouveaux atomes vont être des objets de taille p où p est un nombre premier. Nous sommes donc amenés à définir les *classes premières* $\mathcal{Z}_p$ comme les classes constituées d'un unique élément de taille p (premier).

Définition 3.3.1 *Une classe combinatoire dont la taille est définie multiplicativement et pouvant*

Description	Classe combinatoire	Série de Dirichlet
Classe première	$\mathcal{A} = \mathcal{Z}_p$	$A(s) = p^{-s}$
Union disjointe	$\mathcal{C} = \mathcal{A} + \mathcal{B}$	$C(s) = A(s) + B(s)$
Nombres premiers	$\mathcal{P} = \sum_{p\in\mathbb{P}} \mathcal{Z}_p$	$P(s) = \sum_{p\in\mathbb{P}} p^{-s}$
Entiers naturels	$\mathcal{I} = \sum_{n\in\mathbb{N}^*} \mathcal{Z}_n$	$\zeta(s) = \sum_{n\in\mathbb{N}^*} n^{-s}$
Produit cartesien	$\mathcal{C} = \mathcal{A} \times \mathcal{B}$	$C(s) = A(s) \cdot B(s)$
Séquence	$\mathcal{B} = \textsc{Seq}(\mathcal{A})$	$B(s) = \frac{1}{1-A(s)}$
Multi-ensemble	$\mathcal{B} = \textsc{Set}(\mathcal{A})$	$B(s) = \exp\left(\sum_{k\geqslant 1} \frac{1}{k} A(k\cdot s))\right)$
Cycle	$\mathcal{B} = \textsc{Cyc}(\mathcal{A})$	$B(s) = \sum_{k\geqslant 1} \frac{\varphi(k)}{k} \ln\left(\frac{1}{1-A(k\cdot s)}\right)$

TABLE 9: Règles de constructions pour les séries de Dirichlet

être définie[10] *par des constructeurs classiques* $(+, \times, \textsc{Seq}, \textsc{Set}, \textsc{Cyc})$ *moyennant la modification de la définition de la taille à partir de* classes premières $\mathcal{Z}_p$ *est appelée* classe combinatoire multiplicative spécifiable. *Par la suite, on les désignera par le sigle CCMS.*

Les séries génératrices associées aux CCMS

Par analogie avec les classes combinatoires ordinaires, on associe aux classes combinatoires multiplicatives des séries génératrices énumératives. Ce sont les séries de Dirichlet qui s'adaptent le mieux à la modification de la définition de la taille. Soit $\mathcal{A}$ une classe combinatoire multiplicative, sa série génératrice associée est :

$$A(s) = \sum_{\gamma\in\mathcal{A}} |\gamma|^{-s} = \sum_{n\geqslant 1} a_n n^{-s}$$

où a_n est le nombre d'objets de taille n.
De même que pour les séries génératrices ordinaires, on a une transposition des règles d'assemblage combinatoire en termes de séries génératrices. Ceci nous donne un dictionnaire qui permet d'avoir directement les séries génératrices, à partir des expressions : Il se pose évidement à cet instant la question de l'extraction des coefficients. Tout comme dans le cas des séries génératrices ordinaires, nous disposons de trois procédés d'extraction. Le premier est l'analogue de la dérivation pour les séries entières : $a_1 = \lim_{s\to\infty} A(s)$, puis si l'on connait $a_1, \cdots, a_k$, alors $a_{k+1} = \lim_{s\to\infty} \frac{A(s) - \sum_{1\leqslant i\leqslant k} a_i i^{-s}}{(k+1)^{-s}}$.
Le deuxième (formule découverte par Perron) est l'analogue de la formule de Cauchy :

$$\sum_{1\leqslant k\leqslant n} [k^{-s}]A(s) = \frac{1}{2i\pi} \int_{c-i\infty}^{c+i\infty} n^s \frac{A(s)}{s} \mathrm{d}s.$$

Il faut se rendre à l'évidence, l'extraction des coefficients dans les séries de Dirichlet est moins aisée que dans les séries entières ordinaires. Le troisième nécessite d'avoir une spécification pour la CCMS, dans ce cas, il est généralement possible d'en déduire une récurrence sur les coefficients de la série de Dirichlet.

3.3.2 Générateur de Dirichlet pour CCMS

Un *générateur de Dirichlet (libre)* pour une CCMS $\mathcal{A}$ est un algorithme permettant de tirer aléatoirement un objet de $\mathcal{A}$ dont la distribution suit la loi suivante :

$$\mathbb{P}_s(\alpha) = \frac{|\alpha|^{-s}}{A(s)}$$

10. Cette définition peut être récursive.

On note $\Gamma(\mathcal{A})_s$ le générateur de Dirichlet associé à la CCMS $\mathcal{A}$.

Tout comme dans le cas des générateurs de Boltzmann, il est possible de considérer deux niveaux de rejet à partir des générateurs de Dirichlet libres : un rejet pour obtenir un générateur en taille approchée et un rejet permettant d'obtenir un générateur en taille exacte. Néanmoins, pour certaines valeurs de taille, le générateur de Dirichlet peut se révéler inefficace (du fait de la forte fluctuation des coefficients).

Nous allons décrire les différents algorithmes correspondants aux constructeurs des CCMS. Ces algorithmes sont très largement inspirés des algorithmes pour les générateurs de Boltzmann classiques. La preuve de leur validité peut aisément être calquée à partir de celle des générateurs de Boltzmann.

Atome : $\Gamma(\mathcal{Z}_p)_s \to \square_p$

Somme : Soit $\mathcal{A}, \mathcal{B}$ et $\mathcal{C}$ trois CCMS telles que $\mathcal{C} = \mathcal{A} + \mathcal{B}$.

```
Γ(C)_s → if Bern(A(s)/C(s)) then
    return Γ(A)_s
else
    return Γ(B)_s
```

Nombres premiers : Cet algorithme permet de tirer un élément de la classe $\mathcal{P}$ définie dans le tableau 3.3.1.

```
Γ(P)_s →
r := random(0..P(s))
k := 0
p := 0
M := P(s)
while M > r do
    k := k + 1
    p := p_k (où p_k désigne le k-ième nombre premier)
    M := M − p_k^{−s}
return □_p
```

Entiers naturels : Cet algorithme permet de tirer un élément de la classe $\mathcal{I}$ définie dans la section 3.

```
Γ(I)_s →
r := random(0..ζ(s))
k := 0
M := ζ(s)
while M > r do
    k := k + 1
    M := M − k^{−s}
return □_k
```

Produit : Soit $\mathcal{A}, \mathcal{B}$ et $\mathcal{C}$ trois CCMS telles que $\mathcal{C} = \mathcal{A} \times \mathcal{B}$.

```
Γ(C)_s →return (Γ(A)_s Γ(B)_s)
```

Séquence : Soit $\mathcal{A}$ et $\mathcal{B}$ deux CCMS telles que $\mathcal{B} = \textsc{Seq}(\mathcal{A})$

```
Γ(B)_s →
k := Geom(A(s))
return (Γ(A)_s ··· Γ(A)_s) (k fois)
```

Multi-ensemble : Soit $\mathcal{A}$ et $\mathcal{B}$ deux CCMS telles que $\mathcal{B} = \textsc{Set}(\mathcal{A})$

```
1  Γ(B)_s →
2  M := ∅; k := Indice_Max(A(s), s)
3  for i = 1 to k − 1 do
4  |  p_i := Pois((1/i) A(i.s))
5  |  for j = 1 to p_i do
6  |  |  α := Γ(A)_is
7  |  |  Concaténer i fois α à M
8  if k ≠ 0 then
9  |  p_k := Pois_⩾1((1/k)(A(k.s)))
10 |  for j = 1 to p_k do
11 |  |  α := Γ(A)_ks
12 |  Concaténer k fois α à M
13 return ⟨M⟩
```

Avec $Indice\_Max$ définit par :

```
1  Indice_Max(A, s) →
2  U := random(); V := ln(1/U)
3  p := log(B(s)); k := 0
4  while U < S do
5  |  k := k + 1
6  |  p := p − A(ks)/k
7  return k
```

L'idée de ce générateur est de commencer par tirer le nombre maximum d'objets différents qui vont apparaître dans le Set et ensuite de remplir l'ensemble généré avec des réplications de ces objets [42].

Espérance et variance du générateur

Suivant le principe de Boltzmann, il nous faut calibrer le paramètre s afin de viser une taille d'objet donnée. Un calcul rapide montre que la variable aléatoire N de la taille des objets suivant une distribution de Dirichlet vérifie :

$$\begin{aligned}\mathbb{E}_s(N) &= \frac{\sum_{\gamma\in\mathcal{A}} |\gamma|^{-s}\cdot|\gamma|}{A(s)} \\ &= \frac{A(s-1)}{A(s)}\end{aligned}$$

$$\begin{aligned}\mathbb{V}_s(N) &= \frac{1}{A(s)}\cdot\sum_{\gamma\in\mathcal{A}}(|\gamma|-\mathbb{E}_s(X))^2\cdot|\gamma|^{-s} \\ &= \frac{A(s)\cdot A(s-2)-A(s-1)^2}{A(s)^2}\end{aligned}$$

Remarque 3.3.2 *À ce niveau, il nous faut mettre en avant une différence importante avec le modèle de Boltzmann classique : lorsque le paramètre de contrôle s est compris entre ρ et $\rho+1$, bien que la série $A(s)$ soit bien définie, elle a une espérance infinie. Ceci découle de la décroissance lente de la distribution. Cela préfigure l'importance des générateurs singuliers, à savoir qu'il va être préférable de ne pas calibrer le générateur par espérance, mais tirer les objets avec un générateur dont l'espérance de la taille de la sortie est infinie, quitte à faire du rejet anticipé pour éviter la formation des objets de trop grande taille.*

3.3.3 Étude de complexité

Dans cette partie, nous étudions la complexité du générateur, et plus particulièrement le nombre moyen de rejets nécessaire pour obtenir un objet dans la fenêtre $[(1-\varepsilon)n.(1+\varepsilon)n]$. Mais commençons par une banalité :

Génération libre Il n'y a, concernant le générateur libre, aucune difficulté à adapter la preuve de complexité des prégénérateurs de Boltzmann libres. Nous avons donc le résultat suivant :

Théorème 3.3.3 *La génération d'un objet par le générateur aléatoire a une complexité linéaire en la taille de sortie.*

Remarquons néanmoins que les générateurs de Dirichlet retournant souvent des nombres, il est dans certains cas envisageable d'avoir une génération libre en $\mathcal{O}(\ln(n))$.

Génération avec rejet La question réside essentiellement dans l'analyse de la phase de rejet. La situation des générateurs de Dirichlet est très singulière par rapport à celle des générateurs de Boltzmann. Tout d'abord, les distributions de Dirichlet sont en général extrêmement inhomogènes (forte fluctuation suivant la taille). En particulier, une analyse d'un générateur en taille exacte parait trop fluctuante pour être pertinente. Nous nous attacherons donc ici à une étude du générateur en taille approchée.

Supposons que la série de Dirichlet admette un développement asymptotique de la forme :

$$f(s) \sim_{s\to s_0} \kappa(s-s_0)^{-\alpha}\left(\ln\frac{1}{s-s_0}\right)^{\beta}.$$

Alors on peut "avec les mains" considérer que les coefficients a_n se comportent asymptotiquement comme n^{s_0-1}. Ceci est bien évidement complètement faux car les a_n ont un comportement beaucoup plus complexe à décrire que cela. Mais nous pouvons néanmoins extrapoler que puisque les séries $\sum_n a_n n^{-s}$ et $\sum_n n^{-s+s_0-1}$ ont le même domaine de convergence alors elles doivent partager la propriété suivante : $\sum_{k=1}^{n} a_k k^{-s} \approx \sum_{k=1}^{n} k^{-s+s_0-1}$. Il nous est maintenant possible de pressentir la probabilité $\mathbb{P}((1-\varepsilon)n < N < (1+\varepsilon)n)$. En effet, ceci n'est rien d'autre que $\sum_{k=1}^{(1+\varepsilon)n} \frac{a_k k^{-s}}{f(s)} - \sum_{k=1}^{(1+\varepsilon)n} \frac{a_k k^{-s}}{f(s)}$. Donc asymptotiquement, nous avons que $\mathbb{P}((1-\varepsilon)n < N < (1+\varepsilon)n) \approx \sum_{k=(1-\varepsilon)n}^{(1+\varepsilon)n} k^{-s+s_0-1}$. Un rapide calcul montre que cette probabilité est maximale quand $s = s_0 + \frac{1}{\ln(n)}$. Nous obtenons alors que $\mathbb{P}((1-\varepsilon)n < N < (1+\varepsilon)n) = \mathcal{O}(\ln(n))$. Il s'ensuit que le nombre moyen d'essais que fait le générateur de Dirichlet pour engendrer un objet de taille approchée (dans l'intervalle $[(1-\varepsilon)n, (1+\varepsilon)n]$) si l'on prend comme paramètre $s = s_0 + \frac{1}{\ln(n)}$ est en $\mathcal{O}(\ln(n))$.

Une démonstration rigoureuse demande néanmoins plus de travail. Il nous faut pour cela caractériser asymptotiquement la probabilité $\mathbb{P}((1-\varepsilon)n < N < (1+\varepsilon)n)$ de tirer un objet dans la plage escomptée en utilisant comme paramètre du générateur $s = s_0 + \frac{1}{\ln(n)}$.

Pour cette estimation, il faut utiliser le théorème de transfert suivant (qui est l'analogue pour les séries de Dirichlet du théorème de transfert asymptotique pour les séries entières) : Si $g(s) \sim (s-1)^{-\alpha}\left(\ln\frac{1}{s-1}\right)^{\beta}$ au voisinage de sa singularité dominante alors sous réserve de conditions qui sont les équivalents des conditions de Δ-analyticités dans le cas des séries entières, nous avons pour $\alpha > 0$,

$$\sum_{1\le k\le N}{}^{*}[k^{-s}]g(s) \sim N\frac{(\ln N)^{\alpha-1}}{\Gamma(\alpha)}(\ln\ln N)^{\beta}\left(1+\sum_{k\ge 1}\frac{c'_k}{(\ln\ln N)^k}\right)$$

Où $\sum_{1\le k\le N}{}^{*} a_k$ indique que le dernier terme de la somme est $a_N/2$ et non a_N. L'analyse est en cours de finalisation. Néanmoins, l'intuition décrite au paragraphe précédent et les simulations que nous avons effectuées concordent toutes vers le fait que le nombre moyen de rejets est en $\mathcal{O}(\ln(n))$.

3.3.4 Conclusion et perspectives

Il est bon de mentionner une différence notable entre les générateurs de Dirichlet et les générateurs de Boltzmann : le choix du paramètre ne doit pas être fait de sorte à avoir une espérance de la taille de la sortie égale à la taille désirée. Afin d'optimiser le nombre de rejets, il est donc nécessaire de prendre un paramètre qui impose une taille moyenne infinie. Notre étude nous a permis de conjecturer le nombre de rejets moyen nécessaire pour effectuer un tirage en taille approchée. Le résultat conjecturé (nombre moyen d'essais logarithmique) est d'un point de vue applicatif tout à fait raisonnable. Ce travail doit se prolonger afin de donner une évaluation rigoureuse du coût d'un générateur libre et proposer des applications variées (cryptographie, factorisation,...). Nous renvoyons le lecteur au chapitre Applications pour un exemple lié à la théorie des pavages.

3.4 Équation différentielle du premier ordre pour les classes combinatoires

Les résultats de cette section ont fait l'objet d'une publication avec O. Roussel et M. Soria [30]. Les deux dernières sections 3.4 et 3.5 abordent la question de l'expressivité intrinsèque des générateurs de Boltzmann. Notre objectif, ici, est donc d'accroître le nombre d'opérateurs supportés par la méthode de Boltzmann, et non plus de modifier le générateur de sorte à atteindre d'autres champs. Dans un premier temps, dans cette section, nous nous focalisons sur l'opérateur point. C'est un opérateur central pour la spécification des structures de données. En particulier, les classes constituées d'objets "à histoire mémorisée" (c'est à dire des objets où la manière dont ils ont été construits compte) sont des classes dont la description nécessite l'opérateur point. À titre d'exemple, les arbres croissants sont des objets à histoire mémorisée. En effet, ce sont des arbres dont les nœuds sont numérotés par leur rang dans l'ordre de construction (en particulier, les ancêtres ont toujours un numéro plus petit que celui de leur descendants). L'opérateur point joue aussi un rôle majeur dans la génération par méthode récursive, où une forme "normale" est obtenue grâce aux règles $(\textsc{Cyc}\,\mathcal{A})' = \textsc{Seq}\,\mathcal{A}$ et $(\textsc{Set}\,\mathcal{A})' = \mathcal{A}' \times \textsc{Set}\,\mathcal{A}$.

Avant de continuer, rappelons succintement les notions de pointage et même, plutôt dans ce mémoire, de dérivation (les deux notions sont quasi-identiques). En effet, il est possible de définir en combinatoire une notion naturelle qui correspond, au niveau des séries génératrices, à effectuer une dérivation. Cette notion va nous permettre de définir des classes combinatoires par des équations différentielles structurelles.

Tout objet étiqueté α de taille n possède une dérivée α' obtenue à partir de α en substituant l'atome de plus grande étiquette dans α par un *trou* (c'est à dire un emplacement dont la taille est nulle). La dérivée de α est donc un objet de taille $n-1$ avec un *trou*.

La *dérivée* d'une classe combinatoire est la classe combinatoire obtenue en dérivant tous les objets de la classe : c'est à dire $\mathcal{A}' = \{\alpha' | \alpha \in \mathcal{A}\}$.

Nous verrons ci-dessous en quoi cette notion est utile pour définir des classes combinatoires représentant des structures ordonnées, en particulier des structures de données. Pour une approche plus détaillée, nous renvoyons le lecteur à la thèse de Greene, où à une introduction à la théorie des espèces [48, 5, 4].

Définition 3.4.1 *Une classe combinatoire $\mathcal{T}$ est dite* différentielle de premier ordre *si elle satisfait la relation suivante : $\mathcal{T}' = \mathcal{F}(\mathcal{Z}, \mathcal{T})$ où $\mathcal{F}(\cdot,\cdot)$ est une fonction bivariée quelconque construite à partir des constructeurs classiques ($\times, \textsc{Seq}, \textsc{Set}, \textsc{Cyc}$), satisfaisant $\mathcal{F}(\mathcal{Z}, \varnothing) \neq \varnothing$* [11].

En notant f la fonction bivariée sur les fonctions génératrices associée à la fonction $\mathcal{F}$, la fonction génératrice exponentielle associée à $\mathcal{T}' = \mathcal{F}(\mathcal{Z}, \mathcal{T})$ satisfait l'équation $\frac{\mathrm{d}}{\mathrm{d}z}T(z) = f(z, T(z))$. On obtient donc de manière équivalente $T(z) = \int_0^z f(u, T(u))\mathrm{d}u + a$ où a est donné comme condition initiale exprimant le nombre d'objets nuls dans la classe.

Par exemple, intéressons-nous un instant à la classe des arbres croissants binaires. Il est bien connu que cette classe est en bijection avec les *permutations alternantes*. Elle admet pour spécification l'équation différentielle structurelle suivante :

$$\begin{aligned} \mathcal{T}' &= \varepsilon + \mathcal{T} \times \mathcal{T} \\ \mathcal{T}_0 &= \varnothing \end{aligned}$$

De sorte que $\mathcal{F}(\mathcal{Z}, \mathcal{T}) = \varepsilon + \mathcal{T} \times \mathcal{T}$ satisfait l'hypothèse $\mathcal{F}(\mathcal{Z}, \varnothing) = \varepsilon \neq \varnothing$. En termes de fonctions génératrices nous obtenons donc $\frac{\mathrm{d}}{\mathrm{d}z}T(z) = 1 + T^2(z)$ et $T(0) = 0$, qui donnant lieu à une équation à variables séparées s'intègre aisément en $T(z) = \tan(z)$.

3.4.1 Générateurs de Boltzmann

L'idée fondamentale proposée par Flajolet et Soria est de modifier *probabilistiquement* la valeur du paramètre x au cours des appels récursifs du générateur. Intrinsèquement, cette idée sert à obtenir un

11. Cette condition assure que l'itération de Picard converge vers une série entière. En fait, à chaque itération, on gagne au moins un terme correct de plus.

générateur de Boltzmann pour la classe $\mathcal{T}$ à partir d'un générateur de Boltzmann pour sa classe dérivée $\mathcal{T}'$. Un calcul aisé montre que la densité de probabilité suivante vérifie les propriétés escomptées :

$$\delta_x^T(u) = \frac{xT'(ux)}{T(x) - T(0)} = \frac{xf(ux, T(ux))}{T(x) - T(0)}$$

De plus, cette densité a la propriété importante d'être croissante (comme fonction de u) sur $[0, 1]$ quel que soit $x \in [0, \rho_T]$ [12].

Nous sommes maintenant en mesure de présenter un algorithme pour générer un objet de $\mathcal{T}$ à partir d'un générateur de $\mathcal{T}'$.

Algorithme 9 : Générateur de Boltzmann $\Gamma\mathcal{T}$ à partir d'un générateur $\Gamma\mathcal{T}'$

Input : le paramètre de Boltzmann x $(0 < x < \rho_T)$
Output : un objet de $\mathcal{T}$
1 **if** Bern $\left(\frac{T(0)}{T(x)}\right)$ **then**
2 | **return**
3 **else**
4 | un objet tiré uniformément dans $\mathcal{T}_0$
5 Tirer U dans $[0, 1]$ suivant la densité δ_x^T
6 Tirer aléatoirement un objet f à partir de la classe $\mathcal{F}(\mathcal{Z}, \mathcal{T})$ utilisant $\Gamma\mathcal{T}'(Ux)$
7 **return** L'objet (étiqueté) $(\mathcal{Z}, f)$, où l'atome $\mathcal{Z}$ a la plus grand étiquette

Il est alors immédiat de noter que de cet algorithme découle un générateur de Boltzmann pour $\mathcal{T}' = \mathcal{F}(\mathcal{Z}, \mathcal{T})$. Il suffit en effet de faire des appels récursifs à $\Gamma\mathcal{T}'$ à la ligne 5 lors de l'instruction $\Gamma\mathcal{T}'(Ux)$ qui est équivalente à $\Gamma\mathcal{F}(\mathcal{Z}, \mathcal{T})(Ux)$.

3.4.2 Complexité algorithmique

La question principale est maintenant d'évaluer, tout comme dans le cas des générateurs de Boltzmann précédemment exposés, la complexité en temps de ce générateur. Comme précédemment donc, nous allons évaluer la complexité de l'algorithme 9 en fonction de la taille de l'objet produit. Il nous faut au préalable faire deux hypothèses dont nous vérifierons la validité par la suite. L'une concerne l'évaluation de la série $T(x)$, c'est à dire en utilisant notre terminologie le calcul de l'oracle. Nous supposons donc ici que l'évaluation numérique de $T(x)$ pour n'importe quel x dans le disque de convergence peut être fait en $\mathcal{O}(n)$. L'autre hypothèse concerne le tirage d'une variable aléatoire réelle suivant une densité donnée δ_x. Nous supposons ici que cette opération peut être effectuée en temps constant $\mathcal{O}(1)$.

Sous les hypothèses décrites ci-dessus, l'algorithme 9 est un générateur libre de Boltzmann dont la complexité est quasi-linéaire [13] en la taille de la sortie.

Remarque 3.4.2 *Dans la grande majorité des cas rencontrés quand on fait de la génération aléatoire, l'information sur le* profil *est fondamentale par rapport à celle de l'étiquetage. Ceci constitue la raison pour laquelle il nous est apparu raisonnable de négliger le ré-étiquetage des objets dans le processus de génération. À noter qu'il en est de même dans le cas des générations d'objets étiquetés dans les sections précédentes : l'étiquetage est laissé à la charge de l'utilisateur en fin de processus.*

Nous sommes bien évidement confrontés maintenant à la question de la complexité des générateurs en taille exacte et en taille approchée. Il est bien connu (voir [43]) que les classes appartenant à la famille des arbres croissants ont toutes des séries génératrices qui satisfont un développement singulier du type $T(x) \sim C(1-x)^{-\alpha} \ln^{\beta}\left(\frac{1}{1-x}\right)$. En utilisant les résultats précédemment décrits [40], il en

12. On vérifie sans peine que $\delta_0^T(u) = 1$ et $\delta_{\rho_T}^T(u) = \text{Dirac}(u - \rho_T)$ quel que soit $\mathcal{T}$.
13. En moyenne, il faut $\mathcal{O}(\ln n)$ évaluations, chacune d'elles pouvant être effectuée en temps linéaire.

découle donc que nous sommes toujours soit dans le cas de distributions piquées, soit dans celui de distributions plates. Dans tous les cas, moyennant pointage si nécessaire, on obtient donc une complexité linéaire pour la génération en taille approchée et quadratique en taille exacte.

3.4.3 Calcul de l'oracle : évaluations de $T(x)$

On a besoin, durant la génération, d'évaluer la série génératrice $T(x)$ en plusieurs valeurs (imprédictibles car aléatoires) de x. En général, T est uniquement décrit de manière implicite à partir de l'équation différentielle fonctionnelle. Trois chemins s'offrent à nous pour le calcul de l'oracle :

- utiliser une simple méthode itérative,
- utiliser une approche selon la méthode de Newton,
- utiliser une approche suivant la méthode de Runge-Kutta.

Nous montrons qu'en général la plus simple des méthodes est en un sens suffisante afin d'assurer un calcul rapide. Plus précisément, nous montrons qu'après n itérations on a calculé "presque correctement" un nombre exponentiel de termes dans le développement limité de T.

L'algorithme procède de la manière suivante :

Algorithme 10 : Approximation de la série $T(x)$

Input : $x \in \mathbb{R}$ avec $0 \leqslant x < \rho_T$, $n \in \mathbb{N}$
Output : un polynôme approximant T par valeur inférieure

```
T_0(z) := T(0)
for i from 1 to n do
    Calculer f̃(z) = f(z, T_{i-1}(z))
    T_i(z) = T(0) + ∫ f̃(z) à l'ordre 2^i
return T_n(x), l'évaluation du polynôme T_n en x
```

Remarque 3.4.3 *Notons que bien évidement, il ne faut calculer qu'une fois le polynôme approché T_n. Après cela, il n'est plus nécessaire que d'évaluer celui-là en différents points. Les spécialistes de la méthode récursive pourront objecter que cette approche consiste à pré-calculer les coefficients de la série génératrice et en ce sens diffère peu de la méthode récursive.*

L'idée directrice est la suivante, en appliquant n itérations, on obtient certes uniquement les coefficients corrects de T jusqu'au rang n, mais aussi de manière presque correcte ses coefficients jusqu'à un ordre exponentiel en n. En effet, les classes combinatoires décrites par des équations différentielles du premier ordre partagent toutes la propriété fondamentale suivante : pour une taille donnée n, la majorité des objets de taille n ont une hauteur [14] en $\ln n$ (concentration gaussienne). En d'autres termes, l'espérance de la hauteur d'un objet de taille n est de l'ordre de $\ln n$. Ceci implique la proposition importante suivante.

Le générateur de Boltzmann de $\mathcal{T}$ utilise en moyenne seulement $\mathcal{O}(\ln n)$ appels récursifs quand il est calibré pour générer des objets de taille n.

L'évaluation en x du polynôme $T_n(x)$ donne une bonne approximation de l'évaluation de $T(x)$ dans le sens suivant :

$$\forall z > x, |T(x) - T_n(x)| \leqslant \text{cst} \cdot \left(\frac{z}{\rho}\right)^{n+1}$$

où ρ est le rayon de convergence de la série $T(z)$.

14. La hauteur d'un objet étant la hauteur de son arbre syntaxique.

3.4.4 Tirage suivant une densité de probabilité donnée

Le deuxième point à expliciter est le tirage d'un réel suivant une distribution δ_x sur $[0,1]$.

Pour tirer une variable aléatoire X suivant la densité δ_x, nous allons appliquer une méthode d'inversion. Premièrement, on remarque que la distribution cumulée Δ_x de δ_x admet une forme très simple :

$$\begin{aligned}\Delta_x^T(t) &= \int_0^t \delta_x^T(u)\mathrm{d}u \\ &= \int_0^t \frac{xT'(ux)}{T(x)-T(0)}\mathrm{d}u \\ &= \frac{T(tx)-T(0)}{T(x)-T(0)}\end{aligned}$$

Donc, par la méthode d'inversion, il suffit donc de tirer un U uniformément dans $[0,1]$ et de résoudre en t l'équation

$$\begin{aligned}\Delta_x^T(t) &= U \\ \frac{T(tx)-T(0)}{T(x)-T(0)} &= U \\ T(tx) &= (1-U)T(0)+UT(x)\end{aligned}$$

Ce qui se fait aisément car T est croissante (il suffit par exemple de procéder par dichotomie ou par une itération de Newton).

Accordons-nous un instant pour une petite réflexion. La méthode d'inversion permettant le tirage de t donne intrinsèquement l'évaluation $T(tx)$. En particulier, si l'on considère le sous-problème où $\mathcal{T}' = \mathcal{F}(\mathcal{T})$, dans cette situation, le générateur de Boltzmann pour $\mathcal{F}(\mathcal{T})$ n'utilise que des évaluations de fonctions de la valeur $T(tx)$, il n'est donc pas utile de calculer t et toutes les évaluations nécessaires pour le générateur de Boltzmann peuvent être obtenues à partir de la relation

$$T(tx) = (1-U)T(0)+UT(x).$$

Enfin, dans ce cas de figure, l'équation est à variables séparées, ce qui permet d'obtenir une forme intégrale simple pour $T(z)$. Reprenons pour étayer notre discours le problème de la génération des arbres croissants binaires. Nous donnons, ci dessous un générateur qui prend en paramètre non plus x mais $T(x)$, ainsi $\Gamma'(\mathcal{T})(T(x))$ est équivalent en termes de ce qu'il produit à $\Gamma(\mathcal{T})(x)$:

Algorithme 11 : Générateur de Boltzmann optimisé $\Gamma'(\mathcal{T})(\tau)$ pour les arbres croissants binaires.

Input : τ
Output : Un objet de $\mathcal{T}$, c'est à dire un arbre binaire croissant

```
Tirer U uniformément dans [0,1] ;
τ_new := Uτ;
if Bern(1/(1+τ_new²)) then
    f = Feuille
else
    f = Noeud(ΓT(τ_new), ΓT(τ_new))
return L'objet étiqueté f
```

3.4.5 Conclusion et perspectives

La méthode proposée ici permet de générer efficacement (en $\mathcal{O}(n\ln(n))$) des classes combinatoires obtenues à partir d'une équation différentielle du premier ordre. Si de plus l'équation différentielle est

de la forme $\mathcal{T}' = \mathcal{F}(\mathcal{T})$ alors on peut construire des générateurs de Boltzmann de complexité linéaire. Une extension naturelle consiste à regarder les systèmes différentiels de premier ordre, à savoir

$$\begin{cases} \mathcal{T}_1' = \mathcal{F}_1(\mathcal{Z}, \mathcal{T}_1, \cdots, \mathcal{T}_k) \\ \mathcal{T}_2' = \mathcal{F}_2(\mathcal{Z}, \mathcal{T}_1, \cdots, \mathcal{T}_k) \\ \qquad \cdots \\ \mathcal{T}_k' = \mathcal{F}_k(\mathcal{Z}, \mathcal{T}_1, \cdots, \mathcal{T}_k) \end{cases}$$

Ces systèmes sont fondamentaux pour l'étude des modèles d'urnes, ils jouent aussi un rôle important car ils permettent de "simuler" toutes les équations différentielles linéaires de tout ordre. Bien évidemment, le générateur de Boltzmann associé à ce système peut être construit sans peine en "vectorialisant" l'algorithme 9. Néanmoins, l'analyse en complexité du générateur de Boltzmann en taille approchée cache certaines difficultés que nous n'avons pas encore fini d'étudier.

Enfin, un dernier mot pour dire que l'algorithme 9 peut être adapté de sorte à retrouver un générateur de Boltzmann pour l'opérateur boîte [72]. En effet, rappelons que $\mathcal{C} = \mathcal{A}^{\square} \star \mathcal{B}$ se dérive en $\mathcal{C}' = \mathcal{A}' \times \mathcal{B}$ dont la génération peut être faite en dérivant la grammaire de $\mathcal{A}$. Ceci nous amène naturellement à la question de mettre au point des générateurs de Boltzmann pour l'ensemble des constructeurs classiques en méthode symbolique. Le chapitre suivant traite le problème du générateur de Boltzmann pour le produit de Hadamard.

3.5 Générateur de Boltzmann pour le produit de Hadamard

Cette section reprend les travaux publiés avec D. Gardy et R. Roussel [19]. Nous proposons dans cette section un générateur de Boltzmann en taille approchée pour le produit de Hadamard. Avant d'entrer dans le vif du sujet, rappelons le célèbre paradoxe qui sera la clé de voute de notre approche. Considérons le problème suivant : des invités arrivent un à un à une réception ; quel est le temps moyen (le nombre d'invités à être rentrés) que l'on doit attendre de sorte que au moins deux invités aient la même date d'anniversaire ? La réponse est 25 ; elle est étonnamment basse, et est intimement liée au célèbre problème des anniversaires. Maintenant considérons la variante où on désire de plus que l'anniversaire en commun ait lieu obligatoirement entre une femme et un homme. La réponse reste très basse, il faut attendre en moyenne que rentrent 34 personnes. Le paradigme des anniversaires apparaît de manière variée en analyse d'algorithme et en cryptologie [77]. Dans cette section notre algorithme pour générer les produits de Hadamard, repose lui aussi sur le boys-and-girls birthday paradoxe. Il faut bien entendre que chercher à tirer en taille approchée un couple d'objets de même taille tient beaucoup de la gageure. L'idée directrice est de tirer des objets en taille approchée et d'attendre la première collision de taille. C'est en cela que va nous servir le paradoxe des anniversaires.

Cette recherche de générateur de Boltzmann pour le produit de Hadamard s'inscrit dans un effort joint afin de proposer le plus de briques de base possible pour les générateurs de Boltzmann. Mentionnons dans la même veine, le mélange (shuffle) [36] ou l'opérateur boîte. Le produit de Hadamard consiste à apparier deux objets de même taille, ce produit se révèle très utile dans l'étude de la marche de l'ivrogne, ou dans le problème de partition en cycle de graphes. De manière générale, il est souvent utile de disposer de plusieurs objets de même taille afin de faire des comparaisons autour de propriétés fortement liées à la taille.

Rappelons ici le concept classique de produit de Hadamard :

Définition 3.5.1 *Le* produit de Hadamard *de deux classes $\mathcal{B}$ et $\mathcal{C}$, que nous noterons $\mathcal{B} \odot \mathcal{C}$, est le sous-ensemble de $\mathcal{B} \times \mathcal{C}$ formé des couples de deux éléments de même taille. De plus, soit $\alpha = (\beta, \gamma) \in \mathcal{A}$, on définit la taille de α comme $|\alpha| = |\beta| = |\gamma|$.*

3.5.1 Générateur de taille approchée pour le produit de Hadamard

Comme nous l'avons vu précédemment, des générateurs de Boltzmann ont été construits pour une large variété d'opérateurs combinatoires ($+$, $\times$, Seq, Set, etc.) ; notre objectif dans cette section est de présenter et d'analyser un générateur en taille approchée pour le produit de Hadamard $\mathcal{A} = \mathcal{B} \odot \mathcal{C}$.

Une approche naïve pourrait être la suivante : on tire un objet en taille approchée dans l'une des deux classes, disons $\mathcal{B}$, puis on tire un objet dans la classe $\mathcal{C}$ en taille exacte de sorte qu'il ait la taille de celui que l'on vient de tirer dans $\mathcal{B}$. Cette approche donne un algorithme dont la complexité est soit $\mathcal{O}(n^2)$ quand les deux classes suivent une distribution plate, soit $\mathcal{O}(n\sigma)$ quand l'une des distributions est plate et l'autre est concentrée avec un écart-type σ ou quand les deux distributions sont concentrées (dans ce cas $\sigma = \min(\sigma_B, \sigma_C)$).

Cet algorithme est bien trop naïf pour ne pas être perfectible, quitte à faire usage d'un peu de mémoire. Reprenons maintenant l'idée des paradoxes des anniversaires, qui mène au générateur avec rejet suivant : nous tirons des objets dans $\mathcal{B}$ et $\mathcal{C}$ en taille approchée, jusqu'à ce que l'on obtienne un objet de $\mathcal{B}$ et un objet de $\mathcal{C}$ (généralement pas tirés au même moment) de même taille. Explicitement :

Algorithme 12 : Générateur en taille approchée $\Gamma\mathcal{A}$ pour $\mathcal{A} = \mathcal{B} \odot \mathcal{C}$

Input : l'intervalle de confiance désiré $R = [(1-\varepsilon)n, (1+\varepsilon)n]$ pour la taille de la sortie ; les probabilités q_1 et $q_2 = 1 - q_1$
Output : un objet $\mathcal{A}$ dont la taille est dans R

```
B ← ∅ ; C ← ∅
Choisir x_1 tel que l'espérance de la taille de la sortie de ΓB(x_1) soit n
Choisir x_2 tel que l'espérance de la taille de la sortie de ΓC(x_2) soit n
repeat
    Décider de tirer dans B avec probabilité q_1, ou dans C avec probabilité q_2
    if B est choisi then
        Tirer un objet b dans B en utilisant ΓB(x_1) de taille dans R
        if la taille de l'objet est non répertoriée then B ← B ∪ {b}
    else
        Tirer un objet c dans C en utilisant ΓC(x_2) de taille dans R
        if la taille de l'objet n'a pas encore été répertoriée then C ← C ∪ {c}
until ∃(b, c) ∈ B × C, |b| = |c|;
return a = (b, c)
```

Cet algorithme engendre *des ensembles* d'objets à partir des deux générateurs $\Gamma\mathcal{A}(x)$ et $\Gamma\mathcal{B}(x)$. Son temps d'exécution sera bien évidement fortement relié au cardinal de ces deux ensembles, à savoir au nombre d'éléments tirés. Comme on construit des ensembles à partir de $\mathcal{A}$ et de $\mathcal{B}$, la complexité en espace devra aussi être prise en compte. L'analyse de ces complexités repose fondamentalement sur une interprétation en termes de paradoxe des anniversaires généralisé. Il en découle le résultat suivant :

Théorème 3.5.2 *Le nombre moyen d'appels aux générateurs des deux classes $\mathcal{A}$ et $\mathcal{B}$ nécessaire pour engendrer en taille approchée un produit de Hadamard vaut asymptotiquement :*
- $\mathcal{O}(\sqrt{n})$ *si les deux classes $\mathcal{B}$ et $\mathcal{C}$ ont des distributions de Boltzmann plates,*
- $\mathcal{O}(\sqrt{\sigma_B})$ *si la classe $\mathcal{B}$ a une distribution concentrée et $\mathcal{C}$ un distribution plate,*
- $\mathcal{O}(\sqrt{\sigma})$ *avec $\sigma = \min(\sigma_B, \sigma_C)$ quand les deux classes $\mathcal{B}$ et $\mathcal{C}$ ont des distributions concentrées.*

Comme les objets de $\mathcal{B}$ et $\mathcal{C}$ peuvent être générés en taille approchée en temps linéaire, le coût moyen de cet algorithme est $\mathcal{O}(n\sqrt{\sigma})$ (avec $\sigma = n$ pour les distributions plates) versus $\mathcal{O}(n\sigma)$ pour la version naïve, et l'espace mémoire moyen utilisé est du même ordre que le temps moyen. Dans toutes les situations *le temps d'exécution de cet algorithme est meilleur que celui de l'algorithme naïf.*

La section suivante donne les idées directrices permettant d'obtenir ce résultat.

3.5.2 Problème d'anniversaires

Dans cette section, nous présentons donc une manière de modéliser notre algorithme en termes de problèmes des anniversaires. Notre analyse reposera ensuite centralement sur un élégant résultat de Selivanov [74].

Comme on garde les ensembles de tous les objets de $\mathcal{B}$ et $\mathcal{C}$ que l'on a précédemment tirés, le temps (et l'espace) d'exécution de notre algorithme est entièrement déterminé par le temps d'attente de la première collision, à savoir le premier instant où l'on possède *un objet de $\mathcal{B}$ et un objet de $\mathcal{C}$ de même taille.* Ceci peut donc être interprété comme une variante du problème classique des anniversaires. Soit n la taille moyenne d'un objet et ϵ le paramètre de tolérance, étant donné que l'on ne garde que les objets de $\mathcal{B}$ et $\mathcal{C}$ dont la taille est dans l'intervalle $[n(1-\epsilon), n(1+\epsilon)]$, on peut considérer que l'on dispose de $N = 2n\epsilon - 1$ urnes (correspondant aux différentes tailles possibles). Dans ce modèle, chaque boule pourra alors prendre deux couleurs (une symbolisant les éléments de $\mathcal{B}$ et l'autre ceux de $\mathcal{C}$). Suivant ce point de vue, notre générateur prend la forme d'un problème d'allocation aléatoire :
- On choisit une couleur $k \in \{1, 2\}$ pour une boule avec une probabilité donnée q_k : $q_1 + q_2 = 1$.
- Sachant que la boule que l'on a tirée a la couleur k, nous la déposons dans une urne i avec probabilité $p_{k,i}$, $p_{k,i}$ étant la probabilité conditionnelle que l'urne i soit tirée, sachant que la boule a la couleur k, et $\sum_{i=1}^{N} p_{k,i} = 1$.

– En conséquence la probabilité qu'une boule aille dans l'urne i est donc $\pi_i = p_{1,i}q_1 + p_{2,i}q_2$.

On appelle *collision,* ou *anniversaire,* le moment où *deux boules de différentes couleurs apparaissent dans une même urne.* Naturellement, intéressons-nous à τ_N la variable aléatoire décrivant le nombre de boulesqu'il est nécessaires de tirer avant la première collision : τ_N donne exactement le nombre d'essais nécessaires avant d'obtenir notre produit de Hadamard.

Le paradoxe des anniversaires est un célèbre problème en probabilité discrète (voir Feller [41] pour une approche standard, et [45] pour une reformulation dans le cadre de la méthode symbolique). La variante à boules colorées n'est pas une variation très classique ; à notre connaissance elle apparaît pour la première fois dans Popova [66] pour deux couleurs et des urnes non-uniformes, puis dans un article de Nishimura et Subaya [59], où le cas des urnes uniformes est abordé. Le résultat le plus significatif, de notre point de vue, vient de Selivanov [74], qui considère le cas général de k couleurs, et donne un théorème limite pour la distribution du temps d'attente de la première collision. La version ci-dessous est un reconditionnement de ce théorème à notre problématique :

Théorème 3.5.3 (Selivanov) *Considérons que $N \to +\infty$ et que q_1 et q_2 sont constants. Supposons de plus que $v_2 := \sum_i \pi_i^2$ est $o(1)$ quand $N \to +\infty$. Soit α le facteur de normalisation $\alpha = 2\sqrt{v_2 q_1 q_2}$. Alors la variable normalisée $\alpha\tau_N$ est asymptotiquement distribuée suivant une distribution de Rayleigh (de densité $te^{-t^2/2}\mathbf{1}_{t\geqslant 0}$), de plus,*

$$\mathbb{E}[\tau_N] = \sqrt{\frac{\pi}{4v_2 q_1 q_2}}\,(1 + o(1)).$$

3.5.3 Distribution limite pour τ_N

La question qu'il reste à traiter est de vérifier que les hypothèses de Selivanov sont vérifiées par les distributions de Boltzmann plate et concentrée. Un simple calcul montre que c'est le cas si les classes $\mathcal{B}$ et $\mathcal{C}$ ont une distribution de Boltzmann plate ou concentrée, avec x_n et y_n les paramètres permettant de centrer la taille de sortie des générateurs en n. On en déduit alors que si $\mathcal{B}$ et $\mathcal{C}$ sont toutes deux plates, $v_2 = \Theta(1/m)$; de même si $\mathcal{B}$ est concentrée et $\mathcal{C}$ plate, $v_2 = \Theta(1/\sigma_b)$, et si $\mathcal{B}$ et $\mathcal{C}$ sont toutes deux concentrées, $v_2 = \Theta(1/\sigma)$ où $\sigma = \min(\sigma_b, \sigma_c)$.

De ceci découle directement le nombre de tirages moyen, ainsi que la complexité en temps et en espace du générateur en taille approchée pour le produit de Hadamard. Il suffit simplement de garder en mémoire que $\mathcal{B}$ et $\mathcal{C}$ peuvent être générés en taille approchée en temps linéaire. Ceci peut être résumé ainsi :

Soit $\mathcal{B}$ et $\mathcal{C}$ deux classes combinatoires, et posons τ_n le temps d'attente pour tirer un objet dans $\mathcal{B} \odot \mathcal{C}$.

– Si $\mathcal{B}$ et $\mathcal{C}$ ont des distributions de Boltzmann plates, $\mathbb{E}[\tau_n]$ est de l'ordre de $\mathcal{O}(\sqrt{n})$.

– Si la distribution de Boltzmann de $\mathcal{B}$ est concentrée et celle de $\mathcal{C}$ plate, $\mathbb{E}[\tau_n] = \mathcal{O}\left(\sqrt{\sigma_b}\right)$.

– Si les deux classes $\mathcal{B}$ et $\mathcal{C}$ ont une distribution concentrée, $\mathbb{E}[\tau_n] = \mathcal{O}(\sqrt{\min(\sigma_b, \sigma_c)})$.

3.5.4 Conclusion et perspectives

Nous avons présenté dans cette section un générateur de Boltzmann en taille approchée pour le produit de Hadamard. Notre générateur opère en temps moyen en $O(n\sqrt{\sigma})$, où σ est le plus petit écart-type pour les distributions de Boltzmann des deux classes combinatoires constituant le produit. Nous devons mentionner qu'il existe parfois une alternative plus efficace pour obtenir un générateur pour le produit de Hadamard des cas *particuliers.* Par exemple, un résultat classique de Borel stipule que le produit de Hadamard de deux langages rationnels est lui-même un langage rationnels. Si l'on désire construire un générateur de Boltzmann dans ce cas de figure, plutôt que d'utiliser notre générateur, il est préférable de construire le langage rationnel produit (de Hadamard) à partir des automates déterministes associés aux deux langages de départ, et d'utiliser cette spécification produit pour construire le générateur de Boltzmann. Cette approche assure que le générateur en taille approchée est linéaire.

Notre démarche peut être étendue dans plusieurs directions. Une première extension serait de considérer le cas du k-produit de Hadamard, c'est à dire un k-uplet d'objets tous de même taille. L'analyse de l'algorithme requiert une version étendue du théorème de Selinanov où l'on attend que

k couleurs apparaissent dans une même urne. Une étude préliminaire semble indiquer que, bien que supérieur, le temps d'attente moyen (nombre de tirages) reste de l'ordre de $o(n)$.

Une autre extension serait d'avoir un vrai générateur de Boltzmann libre ce qui nous permettrait de l'utiliser même quand le produit de Hadamard se trouve être un opérateur intermédiaire et non comme un opérateur terminal. Par exemple, notre générateur ne peut pas actuellement générer des objets dont la spécification est $\textsc{Seq}(\mathcal{A} \odot \mathcal{B})$. Pour arriver à cela, il faudrait que la distribution de la localisation de la première collision suive une distribution de Boltzmann de la classe produit de Hadamard, ce qui n'est actuellement pas le cas. L'analyse du processus de rejets nécessaire à corriger la distribution est un travail en cours.

4 Applications

Ce chapitre est dédié à la mise en application des techniques que l'on vient de développer. Nous présentons des applications dans des champs variés : pavages, partitions planes, arbres croissants, marches de l'ivrogne, etc. On propose plusieurs figures montrant des objets de taille de l'ordre de plusieurs milliers d'atomes, sur lesquels des propriétés limites peuvent être observées (forme limite pour les partitions planes, profondeur typique pour les arbres croissants, surface visitée durant la marche de l'ivrogne, etc.). Ceci témoigne du grand intérêt que peut présenter une génération aléatoire efficace.

4.1 Exemple pour le générateur du produit de Hadamard : analyse comparative entre la marche de l'ivrogne classique et une marche de l'ivrogne contrainte

Nous donnons dans cette section un exemple où notre générateur de Boltzmann en taille approchée est utile pour construire un objet aléatoire de grande taille. Le problème que l'on propose de comparer à la marche de l'ivrogne classique est le suivant.

Le chemin de l'ivrogne commence à l'origine, fait un chemin aléatoire en "zig-zag" sans mémoire et revient finalement à l'origine. Plus précisément, considérons ici que le chemin reste dans le premier quadrant $\mathbb{N}^2$ de $\mathbb{Z}^2$ et que chaque mouvement élémentaire (pas) correspond à une translation de vecteur $NE = (1,1)$, $SE = (1,-1)$, $NW = (-1,1)$ ou $SW = (-1,-1)$. La projection sur l'axe (Ox) (resp. (Oy)) d'un chemin de n pas donne un chemin de Dyck de longueur n constitué de pas élémentaires droite/gauche (resp. haut/bas). Réciproquement, étant donnés deux chemins de Dyck C_x et C_y de même longueur respectivement sur $\{x, \bar{x}\}$ et $\{y, \bar{y}\}$, une marche de l'ivrogne peut être construite en choisissant, au i-ème pas :

- NE si le couple constitué respectivement des i-èmes entrées de C_x et C_y est le couple (x, y),
- SE si le couple est $(x, \bar{y})$,
- NW si le couple est $(\bar{x}, y)$ ou
- SW si le couple est $(\bar{x}, \bar{y})$.

Il est bien connu [1, 70] qu'il est possible de tirer un chemin de taille exacte n en temps linéaire : il s'ensuit que le temps pour tirer une marche de l'ivrogne de taille n est linéaire.

Maintenant, supposons que l'ivrogne est tellement ivre qu'il n'est pas capable de faire trois pas consécutifs dans une même direction (Nord, Sud, Est ou Ouest). En d'autres mots, la projection de ce chemin sur l'axe (Ox) (resp. l'axe (Oy)) est un chemin de Dyck sans trois pas consécutifs identiques. Or, les chemins de Dyck sans trois pas consécutifs identiques admettent une spécification hors contexte assez simple : $D = 1 + x\bar{x}D + xx\bar{x}\bar{x}D + xx\bar{x}Dx\bar{x}\bar{x}D$ pour laquelle, d'après ce que l'on sait, il n'existe aucun algorithme linéaire pour faire une génération aléatoire en taille exacte. La série génératrice pour une telle spécification est $d(z) = (1 - z - z^2 - \sqrt{1 - 2z - z^2 - 2z^3 + z^4})/2z^3$ dont la singularité dominante est $\rho = (3 - \sqrt{5})/2 \simeq 0,381966012$. Par conséquent, les chemins de Dyck ainsi contraints ont une distribution de Boltzmann piquée [40], et quitte à pointer une fois la grammaire, nous disposons d'un générateur de Boltzmann en taille approchée en temps moyen linéaire.

Nous désirons comparer cette marche restreinte avec la marche de l'ivrogne classique. Grâce à notre générateur nous sommes en mesure de générer de très grandes marches aléatoires. Les figures suivantes montrent deux marches aléatoires. La marche aléatoire contrainte a été générée en premier, avec comme paramètre $x = 1 - \rho \simeq 0,61803398$, qui assure une taille moyenne de $n = 14376$. Le générateur étant en taille approchée, nous obtenons finalement une marche aléatoire contrainte de taille $n = 14561$. Dès lors nous générons une marche de l'ivrogne aléatoire classique (en générant deux chemins de Dyck de taille 14561).

Nous pouvons observer que l'ivrogne classique explore une zone plus ample du quart de plan. On peut de même émettre à partir de ces observations des conjectures sur les positions extrémales ou bien encore sur la taille du plus petit rectangle contenant la marche.

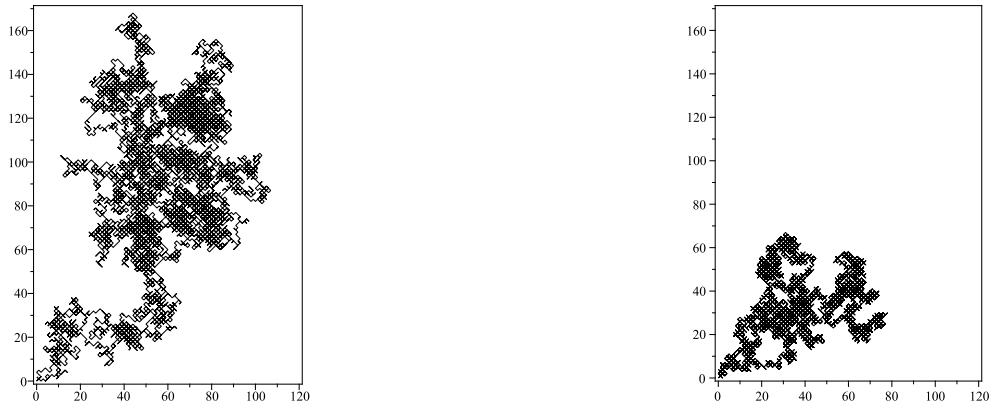


FIGURE 3: à gauche une marche de l'ivrogne classique ; à droite une marche aléatoire contrainte de même taille (graphiques réalisés grâce à une implémentation de O. Roussel)

4.2 Exemple de générateur de Boltzmann multi-paramétré

Nous présentons dans cette section plusieurs applications des générateurs multi-paramétrés. La première application rentre dans le cadre des structures concentrées. La seconde montre un générateur de Boltzmann pour une classe où le paramètre n'est pas un paramètre hérité (ceci dépasse donc un peu le cadre de la présentation que l'on a faite et qui se limitait aux paramètres hérités).

4.2.1 Partitions d'entier ayant un nombre donné de parties

La famille des structures dont la distribution de Boltzmann est concentrée contient, informellement, les classes dont le "principal" constructeur est de type SET ; par exemple les partitions d'entier, les partitions planes, les partitions d'ensemble, les modèles de gaz idéaux d-dimensionnels. Pour illustrer notre propos, nous présentons un générateur aléatoire pour la classe des partitions d'entier où le nombre de parties est fixé. La spécification peut être décrite par $\textsc{Set}(\mathcal{U}\,\textsc{Seq}_{\geqslant 1}(\mathcal{Z}))$. Nous pouvons noter que si l'on fixe à k le nombre de parties dans une partition d'entiers, il est possible d'utiliser la spécification $\textsc{Set}_k(\textsc{Seq}_{\geqslant 1}(\mathcal{Z}))$ et d'appliquer le générateur de $\textsc{Set}_k$ décrit dans [42]. Mais ce générateur est dans un sens à mi-chemin entre un générateur de Boltzmann et un générateur récursif. En effet, il repose sur l'extraction des coefficients de u^k dans la série génératrice bivariée associée à $\textsc{Set}(\mathcal{U} \times \mathcal{A})$ où $\mathcal{A}$ est une classe combinatoire. Nous proposons ici, de sorte à éviter ce précalcul coûteux, d'utiliser pleinement notre générateur de Boltzmann multi-paramétré. A cette fin, nous n'avons qu'à engendrer un objet de $\textsc{Set}(\mathcal{U}\,\textsc{Seq}_{\geqslant 1}(\mathcal{Z}))$ sous le modèle de Boltzmann bivarié pour obtenir en temps moyen linéaire l'exemple suivant (voir fig. 4.2.1). (Noter qu'une partition aléatoire ayant un nombre de parties bornés peut aussi être rapidement générée via la spécification $\prod_{1\leqslant i\leqslant k}(\textsc{Seq}(\mathcal{Z}^i))$. En effet, les partitions utilisant des entiers bornés sont en bijection avec les partitions ayant un nombre borné de parties). Notre exemple "confirme" expérimentalement la courbe limite $y(x)$ implicitement définie par $\frac{1-e^{-c\theta}}{1-e^{-c(\theta+\alpha)}}e^{-cy} + \frac{1-e^{-c\alpha}}{1-e^{-c(\theta+\alpha)}}e^{-cx} = 1$ avec c donné par la condition $\int_{x=0}^{\infty} y\,\mathrm{d}x = 1$. Cette forme limite a été prouvée dans [76] pour les partitions d'entier de taille n ayant $\alpha\sqrt{n}$ parties et des sommants bornés par $\theta\sqrt{n}$.

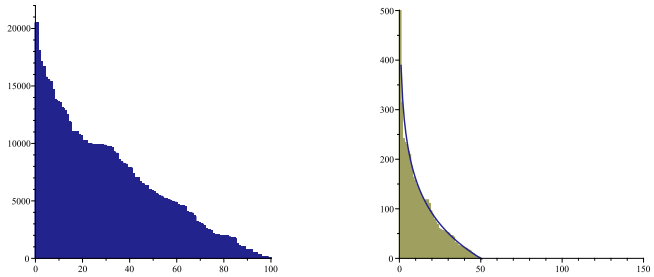


Figure 4: Gauche, une partition d'entiers de taille 667727 avec 100 parties générée avec les paramètres $x = 0,999, u = 0,57$. Droite, une partition d'entiers de taille 4970 avec 50 parties et la courbe limite théorique.

4.2.2 Arbres binaires planaires ayant une longueur de cheminement fixée

Un important exemple ayant une distribution du paramètre concentrée est la classe des arbres binaires planaires ayant une profondeur de cheminement courte. Cette notion apparaît naturellement dans l'analyse de structures de données quand l'on veut étudier le temps moyen pour accéder à une clef dans un arbre de recherche. Cet exemple illustre une situation où le paramètre n'est pas hérité, néanmoins la génération reste très efficace quand la longueur de cheminement est courte (nous ne disposons pas encore de preuve). Cette classe a donc la série génératrice suivante : $T(z,u) = z + zT(zu,u)^2$ où $u \leqslant 1$. Dans la figure (voir figure 4.2.2), nous présentons la distribution de taille de la sortie pour notre générateur de Boltzmann bivarié avec pour paramètre $u = 0,8$ et trois valeurs pour x, respectivement $x = 1, x = 1,5$ et $x = 2$. La concentration autour de la valeur moyenne est expérimentalement claire mais l'analyse théorique nécessite plus d'attention et nous n'avons pour le moment pas réussi à la conclure. Notons aussi que la concentration se dilue quand u tend vers 1, ceci peut être intuité en se rappelant que $u = 1$ correspond à des arbres binaires planaires classiques, pour lesquels la longueur de cheminement n'est pas bien concentrée (loi d'Airy).

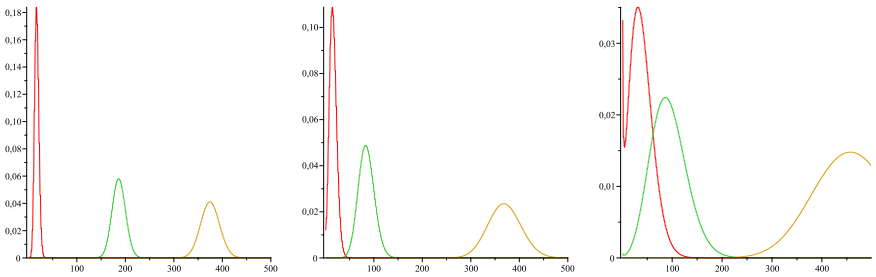


Figure 5: De gauche à droite : la distribution de la taille de la sortie, pour $u = 0,5$ et $x = 10$ (rouge), $x = 100$ (vert), $x = 200$ (jaune), pour $u = 0,8$, et $x = 1,5$ (rouge), $x = 2,5$ (vert), $x = 4$ (jaune), pour $u = 0,95$ et $x = 0,75$ (rouge), $x = 0,8$ (vert), $x = 0,9$ (jaune)

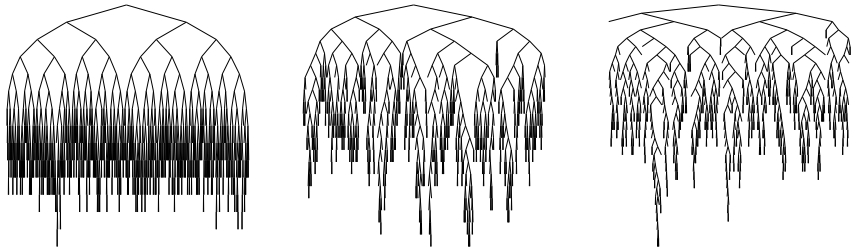


FIGURE 6: De gauche à droite : un arbre de taille 1333 et de longueur de cheminement 11834 généré avec les paramètres $x = 6, u = 0,8$; un arbre de taille 1075 et de longueur de cheminement 11606 généré avec les paramètres $x = 0,67, u = 0,98$; un arbre de taille 1333 et de longueur de cheminement 11834 généré avec les paramètres $x = 0,6035, u = 0,99$

4.2.3 Pavage équilibré avec des tuiles de TETRIS

Nous mettons ici un extrait de l'article [27]. Cela constitue un exemple de génération efficace de langage rationnel pondéré qui permet de tirer uniformément des pavages ayant le même nombre de tuiles de chaque sorte.

Un grand coup de chapeau à Y. Ponty, dont les nombreux talents, et en particulier celui de codeur efficace, nous ont permis d'avoir ce sympathique exemple de pavage aléatoire pondéré.

In this short illustration, we address the generation of Tetris tessellations, i.e. tessellations using tetraminoes of a board having prescribed width w. The Tetris game consists in placing falling tetraminoes (or *p*ieces) $\mathcal{P}$ in a $w \times h$ board. The goal of the player is to create hole-free horizontal lines which are then eliminated, and the game goes on until some piece stacks past the board ceiling. Most implementations of Tetris use the so-called *bag strategy*, which consists in giving the player sequences of permutations of the 7 types of tetraminoes, therefore inducing a uniform composition in each tetramino type. A rational specification (Built by Algorithm 13) exists for Tetris tessellations of any fixed width, but the additional constraint on composition provably throws the associated language out of the context-free class. Therefore, we choose to model the generation of uniformly distributed Tetris tessellations as a multivariate generation within a rational language. Such tessellations could in turn be used as a basic construct to build hard instances for the offline version of the algorithmic Tetris problems [33, 49].

Building the automaton of Tetris tesselations

First let us find an unambiguous decomposition of Tetris tessellations. The idea is to focus on the state of the upper band of the tessellation of height 4, or boundary of a partial tessellation. In particular for (complete) Tetris tessellations the upper band is completely filled and the associated boundary is flat. One can investigate the different ways to get to a given boundary $\mathcal{B}$ by simulating the removal from $\mathcal{B}$ of a piece p , completing the boundary after each removal so that the highest non-empty position stays on the top row. Without further restriction on the position of removal, such a decomposition would be *ambiguous* and give rise to an infinite number of different boundaries. Consequently, we enforce a canonical order on the removal of pieces by restricting it to a set of (possibly rotated) pieces $\mathcal{P}_{\mathcal{B}}$ positioned such that : a) the upper-rightmost position of the piece matches that of the boundary and b) the piece is entirely contained in the boundary. We refer to the induced decomposition as the disassembly decomposition.

Proposition 4.2.1 *The disassembly decomposition generates sequences of removals from and to flat boundaries that are in bijection with Tetris tesselations.*

The finiteness of the state space suggests Algorithm 13 that builds the automaton $\mathcal{A}_w$, generating tessellations of width w. Notice that the resulting automaton in not necessarily co-accessible, since the removal of some piece can create boundaries that cannot be completed into a flat one through

Algorithme 13 : Constructing the automaton $\mathcal{A}_w$ for tessellations of width w. Right : Growth of the number of states for increasing values of w.

Input : The board width w and the flat boundary $\mathcal{B}_w$
Output : Q the states set and σ the transition function of $\mathcal{A}_w = (\mathcal{P}, Q, \mathcal{B}_w, \{\mathcal{B}_w\}, \sigma)$

```
2  begin
3      (Q, σ) ← (B_w, ∅)
       S ← {B_w}
       while S ≠ ∅ do
4          S ⇒_pop B;
5          for p ∈ P_B do
6              B' ← B − p;
7              if B' ∉ Q then
8                  Q ← Q ∪ {B'};
9                  S ⇐_push B';
10             σ ← σ ∪ {(B, p, B')};
11     return (Q, σ)
12 end
```

Width w	#States in $\mathcal{A}_w$	#States minimal
2	4	4
3	55	55
4	80	78
5	1686	1646
6	4247	4130
7	41389	40099
8	49206	47564
9	919832	–

any sequence of removal. Consequently, we added in our implementation a test of connectedness that discards any boundary having a (dis)connected component involving a number of blocs that is not a multiple of 4, as such boundaries clearly cannot reach a flat state again. Running a minimization algorithm of the resulting automata confirms the expected explosion in the number of states (See Algorithm 13) required for increasing values of w.

Random generation

First we point out that the automaton has matching initial and final states, so the strong connectedness is obviously ensured and our theorems regarding the complexity of our generators apply. One can then translate the automaton transitions into a system of functional equations involving the (rational) generating functions associated with each states. Solving the system gives the generating functions, from which one can extract many informations.

For instance, fixing the width $w = 6$ and a number $n = 105$ of pieces, one obtains a number $h_{6,105} = 3.10^{71}$ of potential tessellations, and extracting coefficients of suitable derivatives yields :

Piece							
Frequency (%)	7.90	10.55	20.42	20.42	17.00	7.90	15.81

Consequently, the average composition of a Tetris tessellation is incompatible with the *bag strategy*, which induces uniformly distributed pieces. One can then use the results of Section 3.2.1 to compute a set of weights that ensures 1/7-th proportions in each type of pieces.

Piece							
Weight	0.93	0.84	0.38	0.38	0.46	0.93	0.42
Frequency (%)	14.3	14.1	14.2	14.2	14.2	14.3	14.5

A weight random generation for the $w = 6$ and $n = 105$, coupled with a rejection that allows the numbers of any piece to be equal to 15 ± 1, gives the instances drawn in Figure 8.

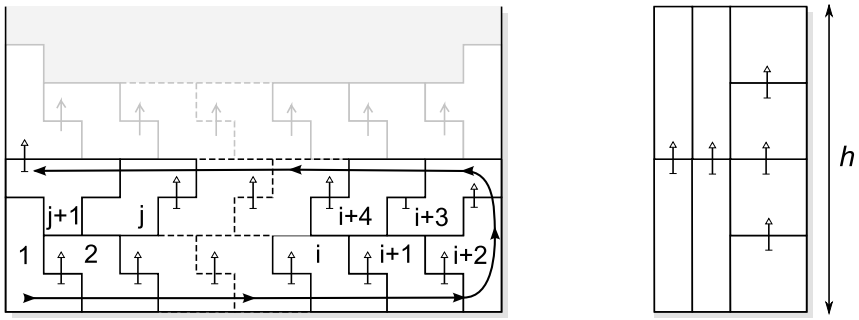


FIGURE 7: Left : Tetris tessellations associated with a unique instance. Only the most relevant dependency points are displayed here (arrows) and pieces are labelled with their rank in the only compatible instance. Duplicating the gadget preserves the uniqueness of the associated instance while allowing for the generation of tessellations of arbitrarily large dimensions. Right : Tesselation realized by $\binom{h}{h/2} \in \Theta(2^n/\sqrt{n})$ different instances.

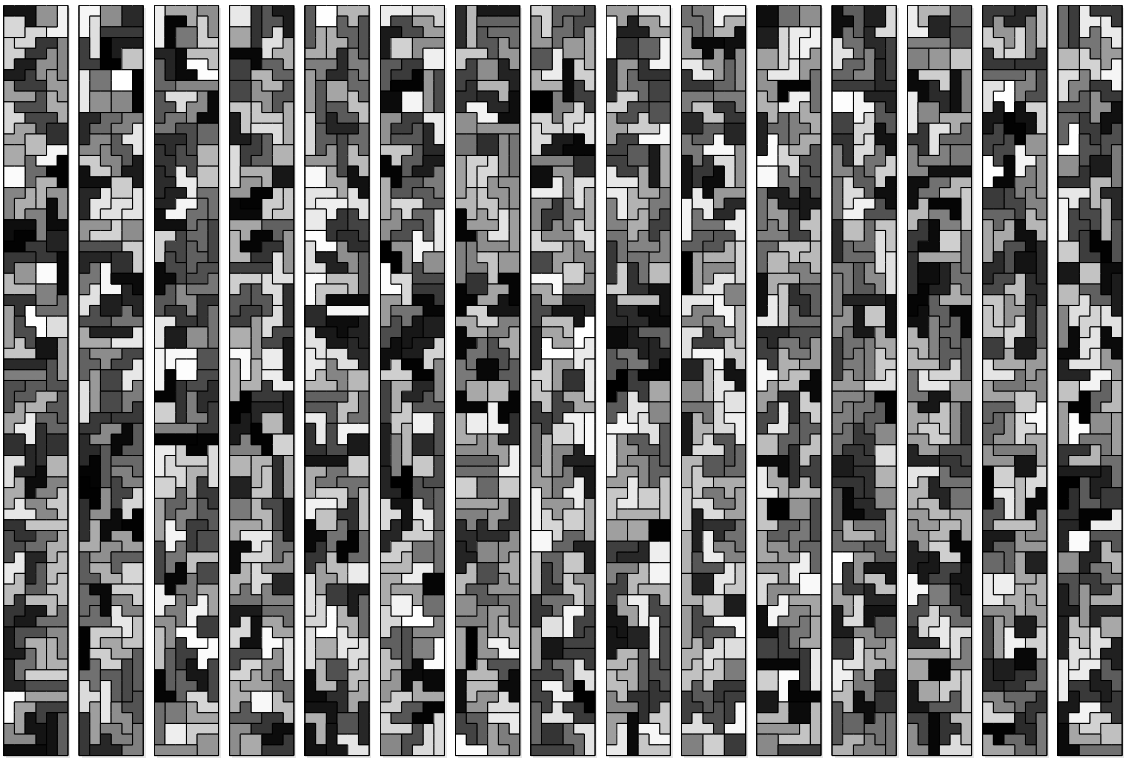


FIGURE 8: Fifteen Tetris tesselations of width 6 having uniform composition (+/- 1) in the different pieces.

From random Tetris tessellations to Tetris instances

Proposition 4.2.2 *For any Tetris tesselation $\mathcal{T}$, there exists an instance (sequence of pieces) such that $\mathcal{T}$ can be obtained.*

Let us discuss the limitations induced by Tetris tesselation as a model for Tetris instances. First it can be remarked that Tetris tesselations do not capture every possible Tetris game ending with an empty board, as one may temporarily leave *holes* which amount to disconnecting pieces in the tesselation representation. Secondly there generally exists different free pieces to choose from while

rebuilding a tesselation, and therefore different instances can lead to a given tesselation. Furthermore the number of instances highly depends on the actual tesselation (from one to an exponential in n, as illustrated in Figure 7), Consequently, using the DAGs associated with Tetris histories to draw instances for the offline version of Tetris algorithmic problems [33] would favor exponentially certain instances over others, and the uniform random generation of instances ensuring feasibility of a perfect Tetris game remains a challenging problem.

4.3 Exemple de génération sous modèle de Dirichlet : les pavages d'un rectangle discret

Le problème est le suivant : Comment énumérer et générer aléatoirement une tuile discrète non nécessairement connexe qui pave par translation un rectangle discret $[1, ..., n] \times [1, ..., m]$. Ce problème est une extension de l'article [29] écrit avec E. Rivals. L'exemple ci-dessous montre une tuile qui pave par translation le rectangle $[1, ..., 4] \times [1, ..., 6]$.

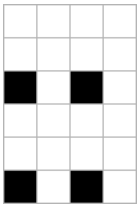


FIGURE 9: Une tuile pavante de $[1, ..., 4] \times [1, ..., 6]$.

La première remarque à faire pour simplifier le problème est qu'une tuile pavant le rectangle $[1, ..., n] \times [1, ..., m]$ est le produit cartésien d'une tuile (unidimensionnelle) qui pave le segment $[1, ..., n]$ avec une tuile qui pave le segment $[1, ..., m]$ et réciproquement.

Nous sommes donc ramenés au problème unidimensionnel. Ce problème est lié aux factorisations de Krasner [52]. Nous disposons du lemme suivant que l'on peut considérer comme un sous-produit des factorisations de Krasner, mais qui se démontre aussi par des arguments simples purement combinatoires :

Lemme 4.3.1 *On note $[b_1, t_1, b_2, t_2, ..., b_k, t_k]$ la tuile (unidimensionnelle) constituée d'un bloc de b_1 cases suivi un trou de t_1 cases, puis un bloc de b_2 cases et ainsi de suite. Si une tuile pave un segment alors tous les b_i sont égaux, et tous les t_i divisent b_1.*

Il nous faut aussi remarquer la dualité suivante : Si une tuile T pave un segment, alors en marquant les positions de la première case de tous les translatés de T qui permettent de paver le segment, on forme une tuile qui pave elle aussi le segment. Cette tuile est appelée le *dual* de T.

En combinant le lemme avec cette remarque, il est possible d'obtenir une caractérisation des tuiles pavantes que l'on va décrire de manière algorithmique.

Définition 4.3.2 (décomposition primale-duale) *On appelle* décomposition primale-duale *d'une tuile T le résultat renvoyé par l'algorithme 14.*

Algorithme 14 : Décomposition primale-duale

Input : une tuile $T = [b_1, t_1, b_1, t_2, ..., b_1, t_k]$ qui pave un segment S
Output : une suite d'entiers $D = (a_1, ..., a_k)$ qui caractérise la tuile T comme paveur de S

```
D = ()
repeat
    D = concat(b_1, D) où b_1 est la taille du premier bloc de T
    Soit T_r la tuile T_r := [1, t_1/b_1, 1, t_2/b_1, ..., 1, t_k/b_k] (qui pave le segment [1, ..., n/b_1].)
    Soit T' le dual de T_r pour le segment [1, ..., n/b_1].
    T = T'
until T ≠ [1];
return D
```

Cette décomposition permet de donner une spécification multiplicative pour les tuiles pavant un segment.

Théorème 4.3.3 *La classe combinatoire multiplicative $\mathcal{T}$ des tuiles pavant un segment est donnée par la relation : $\mathcal{T} = \mathcal{I} \times \textsc{Seq}(\mathcal{I} \backslash \mathcal{Z}_1)$. La fonction génératrice de Dirichlet associée est $T(s) = \dfrac{\zeta(s)}{2 - \zeta(s)}$.*

Exemple 4.3.4 *Décrivons l'algorithme primal-dual sur une instance. Soit T la tuile $[2, 4, 2, 4]$ qui pave le segment $[1, 12]$. $D = (2)$, $T_r = [1, 2, 1, 2]$ (qui pave $[1, 6]$) et dont le dual est $[3, 3]$. Cela donne $D = (2, 3)$ avec un nouveau $T_r = [1, 1]$ qui pave $[1, 2]$ et dont le dual est $[2]$. Finalement, la suite primale-duale associée à $[2, 4, 2, 4]$ est $[2, 3, 2]$.*

A noter que la classe combinatoire multiplicative $\textsc{Seq}(\mathcal{I} \backslash \mathcal{Z}_1)$ est connue comme la classe combinatoire multiplicative des factorisations ordonnées qui est une classe standard en théorie des nombres.

Maintenant, $T(s)$ admet un pôle d'ordre 1 en $\zeta^{-1}(2)$, il nous est possible d'utiliser notre générateur en taille approchée afin de tirer des tuiles 1D pavant un segment de grande taille, tout comme des tuiles 2D pavant un rectangle de grande surface.

Une rapide implémentation en Maple permet de générer de très grandes tuiles. Le graphique ci-dessous représente une tuile qui pave un rectangle $[1..200] \times [1..224]$. Il est possible de générer des tuiles pavant des rectangles d'aire plusieurs millions en quelques secondes.

Figure 10: Une tuile 2D pavant $[1, ..., 200] \times [1, ..., 224]$.

4.4 Exemple de générateur de Boltzmann pour les spécifications différentielles : les arbres unaire-binaires croissants

Rappelons qu'un arbre unaire-binaire croissant est une structure de données très classique dont la spécification différentielle est décrite par la récurrence $\mathcal{T}' = \varepsilon + \mathcal{T} + \mathcal{T} \times \mathcal{T}$. Un arbre unaire-binaire croissant de taille n peut être considéré comme un arbre unaire-binaire de taille n dont les noeuds sont étiquetés de manière univoque de 1 à n et tel que tout ancêtre a une étiquette plus petite que ses descendants.

Nous avons mené cette expérimentation sur un ordinateur portable cadencé à 2Ghz, disposant de 2Go de RAM. Le code est implémenté sans aucune optimisation matérielle ou logicielle en Maple. On peut atteindre alors des arbres de l'ordre de 10^7 noeuds en un temps moyen d'à peu près 10 secondes pour une génération en taille approchée (voir figure 11).

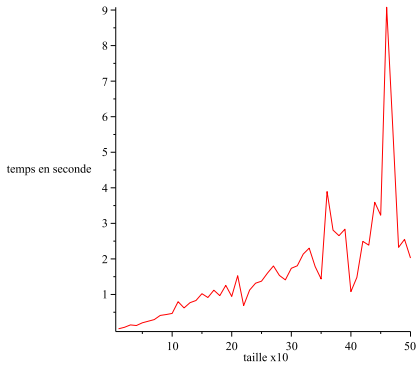


FIGURE 11: Temps moyen empirique (sur 50 essais) pour générer en taille 10%-approchée un arbre binaire croissant de taille visée n

En utilisant notre approche, nous tirons donc profit de la possibilité par la méthode de Boltzmann de tirer en taille approchée, et nous parvenons à générer des arbres de très grande taille. L'exemple illustré figure 12 a été généré en moyenne en une milliseconde sur une machine grand public.

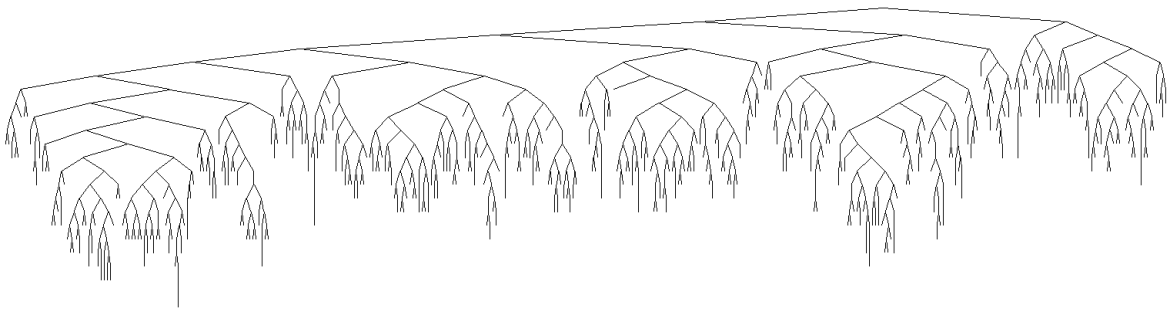


FIGURE 12: Un exemple d'arbre unaire-binaire croissant de taille 969

4.5 Générateur de Boltzmann classique pour la physique statistique : un générateur efficace pour les partitions planes

Cette section reprend directement la présentation faite dans l'article [17] écrit avec E. Fusy et C. Pivoteau, les graphiques ont été réalisés par C. Pivoteau. Nous présentons donc ici une application de la méthode de Boltzmann classique à des objets dont la description combinatoire n'est pas immédiate. Rappelons en effet que la génération sous modèle de Boltzmann repose sur la spécification combinatoire. Dans l'exemple des partitions planes, il n'existe pas de description combinatoire directe, mais un procédé explicite qui met en bijection (découverte par Pak) les partitions planes avec la classe combinatoire non-étiquetée $\textsc{Set}\left(\mathcal{Z}\,\textsc{Seq}(\mathcal{Z})^2\right)$.

Revenons un peu dans le temps, les partitions planes ont été introduites par Young [80] ; elles constituent une naturelle généralisation des partitions d'entier dans le plan, en effet elle consiste en une matrice d'entiers dont les lignes et les colonnes sont respectivement décroissantes de gauche à droite et de bas vers le haut. (alors qu'une partition d'entiers est un tableau à une dimension contenant une suite décroissante d'entiers). En complément, les partitions planes disposent d'une élégante interprétation en 3-dimension comme un empilement de cubes. Les partitions planes font l'objet d'une colossale littérature essaiminée dans les domaines les plus variés des mathématiques [3, 47, 51, 58, 79] et de la physique statistique [56, 75], et sont à la source d'avancées cruciales pour la résolution de certaines célèbres conjectures en combinatoire [81], voir [32] pour des détails historiques sur la conjecture des matrices alternantes.

Le problème de l'énumération des partitions planes a été résolu au début du siècle par MacMahon [55], qui mit en évidence la remarquable formule suivante :

$$P(x) = \prod_{r\geqslant 1} \frac{1}{(1-x^r)^r} \tag{4}$$

pour la série génératrice des partitions planes. La simplicité de cette formule pose la question naturelle de l'interprétation combinatoire que l'on pourrait lui donner. Or cette question se révèle d'un autre ordre de difficulté. Il faut attendre les années 90, pour qu'enfin une première preuve bijective directe soit donnée par Krattenthaler [53]. Le principe est inspiré d'une bijection de Novelli-Pak-Stoyanovskii [60] donnant une interprétation combinatoire de la fameuse formule des longueurs d'équerre (hook-length formula). Dans [53], Krattenthaler propose aussi comme application de sa bijection un algorithme en temps polynomial pour la génération aléatoire de partitions planes dont la représentation en 3D est inscrite dans une boîte $a \times b \times c$ donnée. En regardant par le dessus et suivant la direction $(1, 1, 1)$, un empilement de cubes peut aussi être interprété comme le pavage par des losanges d'un hexagone dont les longueurs des cotés sont (a, b, c, a, b, c). Là encore, il existe des générateurs aléatoires pour ces pavages, qui reposent soit sur la méthode de couplage arrière (coupling from the past principles) [67] ou sur une approche par déterminant (determinant algorithms) [78]. À l'instar de ces travaux, nous nous sommes intéressés ici à la génération aléatoire uniforme à taille donnée (à savoir la somme des entrées de la matrice) de partitions planes. À cette fin, nous allons utiliser une interprétation bijective de la formule de MacMahon récemment proposée par Pak [64]. Mais, avant cela, faisons un bref point sur les motivations concernant la génération aléatoire de partitions planes à taille donnée. La taille est un paramètre naturel, il correspond au volume de la partition plane (nombre de ses cubes) dans sa représentation tridimensionnelle. Récemment, plusieurs auteurs ont étudié des propriétés statistiques des partitions planes de taille donnée. En particulier, sous une distribution à taille fixée, Mutafchiev [57] a montré une loi limite pour l'entrée maximale de la matrice, tandis que Cerf et Kenyon [34] ont déterminé la forme asymptotique (parallèlement, la forme asymptotique dans le cadre des partitions planes dans une boîte —pavages d'un hexagone par des losanges— a été prouvée par Cohn, Larsen, and Propp [35]). Encore plus récemment, Okounkov et Reshetikhin, utilisant une méthode fondée sur les processus de Schur, ont proposé une nouvelle approche pour déterminer la forme limite découverte par Cerf et Kenyon [62]. Ils ont prolongé leurs travaux dans l'article [63], où ils prouvent une forme limite pour les partitions planes sous un modèle mixte entre la version non bornée et celle inscrite dans une boîte : la partition plane, ici, est contrainte dans deux de ses coordonnées à être dans un rectangle $a \times b$. (Nous allons aussi décrire des générateurs aléatoires pour cet exemple intermédiaire.)

Finalement, les physiciens ont développé de nouveaux modèles reposant sur les partitions planes, permettant d'aboutir à une version simplifiée des réseaux vésiculaires en 3-dimension [54]. Les partitions planes sont aussi reliées aux modèles d'Ising en 3-dimensions dans le réseau cubique [34].

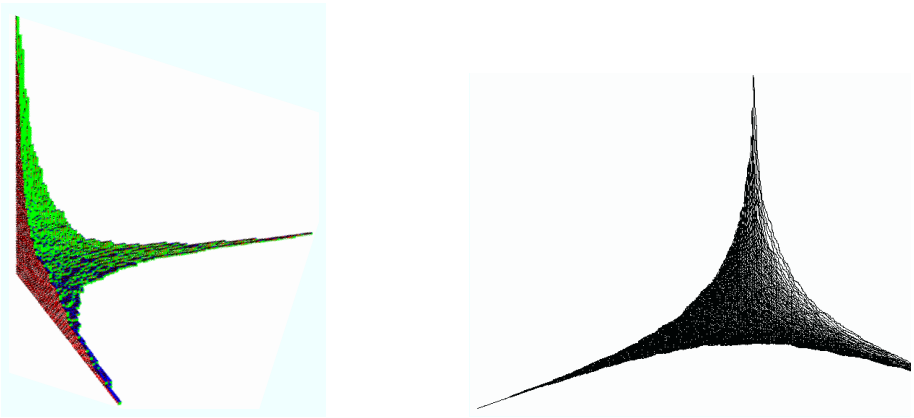
En général, les physiciens sont intéressés par des tests expérimentaux de propriétés leur permettant de conjecturer des propriétés limites sur ces modèles, en particulier des phénomèmes de transition de phases. Ceci passe par la capacité à générer aléatoirement des objets de très grande taille.

Notre approche combine donc des méthodes issues de la combinatoire bijective (bijection de Pak) avec la méthode symbolique sur laquelle repose la méthode de Boltzamnn. Comme la distribution de Boltzmann des tailles des partitions planes se révèle être concentrée, il s'avère possible de calibrer efficacement le paramètre de contrôle x de sorte à tirer efficacement des objets de taille proche et exacte par rapport à une valeur ciblée n. Plus précisément, nous obtenons un générateur en taille approchée *quasi-linéaire* en temps pour les partitions planes : pour n'importe quelle tolérance ε dans l'intervalle $]0, 1[$, notre générateur tire en effet une partition plane de taille dans $[n(1-\varepsilon), n(1+\varepsilon)]$ en temps d'exécution en $O(n(\ln n)^3)$. Les mêmes principes donnent un générateur en taille exacte en temps moyen $O(n^{4/3})$. À notre connaissance, notre algorithme est le premier générateur aléaoire en taille exacte pour les partitions planes dont le temps moyen d'exécution est polynomial.

Ceci nous a permis de générer des objets de taille supérieure à 10^7 en seulement quelques minutes sur un PC (voir 4.5). Le même principe (à savoir bijection de Pak + générateur de Boltzmann) produit aussi un générateur efficace pour les partitions planes étudiées par Okounkov et Reshetikhin qui sont contraintes à avoir deux dimensions incluses dans un rectangle $(a \times b)$. Nous avons obtenu pour les partitions planes en boîte (voir figure15(a)) un générateur en taille approchée de complexité en temps moyen en $O_{a,b,\varepsilon}(1)$ et en taille exacte en $O_{a,b}(n)$, où ε est la tolérance sur la taille (pour la génération en taille approchée) et où n est la taille ciblée.

Finalement, nos générateurs s'appliquent aussi aux partitions planes tordues introduites par Okounkov (voir figure15(b)).

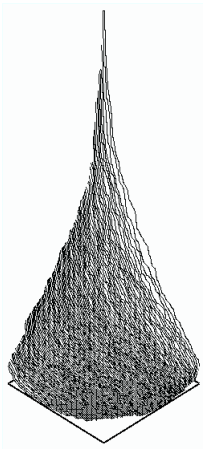
FIGURE 13:



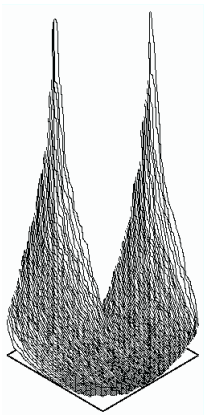
(a) Une partition plane de taille 15256, générée par $\Gamma P(0.947)$.

(b) Une partition plane de taille 1005749, générée par $\Gamma P(0.9866)$, vu dans la direction $(1, 1, 1)$.

Figure 14:



(a) Une partition plane (100×100)-emboitée de taille 999,400.



(b) A $[0..99] \times [0..99] \setminus [0..49] \times [0..49]$ Une partition plane tordue de taille $1,005,532$.

~

5 Conclusion et projets de recherche

5.1 Tout ce que l'on aurait pu et aimé développer dans ce mémoire ...

Nous proposons dans cette section une petite sélection de travaux que nous avons effectués et qui nous semblent les plus pertinents scientifiquement.

Nous avons travaillé sur la détermination de la taille minimale que peut avoir un arbre qui contiendrait comme mineurs tous les arbres de taille n, un tel arbre est dit n-contraction-universel [7]. Nous avons prouvé par une construction récursive assez sophistiquée utilisant des 2-valuations que pour tout n il existe un arbre n-contraction-universel ayant moins de $n^{1,984}$ sommets. Le fait d'avoir une construction de taille sub-quadratique est tout à fait remarquable et laisse envisager que l'on pourrait utiliser de tels arbres comme structures de données efficaces pour stocker des structures arborescentes. Néanmoins, nous avons plus généralement concentré notre attention sur des problèmes de pavages autant d'un point de vue algorithmique que structurels.

Nous avons en particulier étudié un relâchement naturel de la notion de pavage : les $\mathbb{Z}$-pavages (parfois aussi appelés pavages signés). Une région admet un $\mathbb{Z}$-pavage par un jeu de tuiles T si et seulement si en tout point de la forme la somme algébrique des tuiles (signées positivement ou négativement) qui recouvrent ce point donne 1 et que cette somme donne 0 ailleurs. Certaines conditions avaient été trouvées par Conway et Lagarias pour assurer qu'une région admet ou non un $\mathbb{Z}$-pavage. Nous avons avec B. Nouvel [26] montré que le problème pouvait être résolu de manière très générale en utilisant une version des bases de Gröbner sur les anneaux de polynômes à coefficients dans $\mathbb{Z}$. Cette même approche par la géométrie algébrique nous a permis de montrer un théorème très général sur la "nœtherialité" des jeux de tuiles [8]. Plus explicitement, pour tout jeu J de tuiles donné, il existe une famille de rectangles qui pavent exactement les mêmes rectangles que J. Les problèmes de pavages étant en dimension 2 généralement intractables (NP-complets), nous nous sommes intéressés un moment aux pavages de segments de la droite discrète. La question étant de compter, caractériser et reconnaître algorithmiquement quels sont les motifs qui pavent l'intervalle $\{1, \cdots, n\}$. Nous avons avec E. Rivals [29] trouvé des relations profondes sur la structure des motifs qui pavent un intervalle, qui nous ont par exemple permis de montrer que deux séquences de l'Encyclopedia of Integer Sequences de Sloane étaient en fait identiques. Cette étude nous a aussi permis de donner un algorithme de reconnaissance des motifs pavant un segment, en temps linéaire par rapport à la taille du motif. Une autre manière d'attaquer le problème des pavages est de se limiter à des jeux de tuiles très simples comme les dominos par exemple. Une abstraction naturelle des dominos est la notion de dimères (deux cellules côte à côte). Nous nous sommes donc interessés aux pavages d'une polycellule par des dimères. Ces résultats sont l'aboutissement de mon travail de généralisation des articles de Thurston et Conway, Lagarias. Nous avons prouvé avec T. Fernique [11], en utilisant les avancées faites sur les pavages, qu'il existe un algorithme en temps $\mathcal{O}\left(n \ln(n)^3\right)$ permettant de déterminer si un graphe biparti planaire admet un couplage parfait (ce qui est équivalent au problème de pavage par dimère d'une polycellule). Un résultat de cette nature avait été envisagé par L. Lovasz sans avoir pu être jusqu'à présent vérifié. L'étude des pavages nous a aussi amené naturellement à regarder la formation des quasi-cristaux. Avec T. Fernique et D. Regnault, nous avons proposé et analysé un modèle de formation des quasi-cristaux [16, 13] et avec T. Fernique et E. Rémila, l'accessibilité par flip des pavages du plan [14, 15, 16].

Nous avons aussi travaillé sur des problèmes d'analyse structurelle asymptotique de graphes aléatoires et de systèmes logiques. Avec A. Darrasse, M. Soria, puis H.S. Hwang nous étudions la

structure des réseaux appoloniens [9], puis des k-arbres croissants aléatoires [37], et plus particulièrement le profil de leur connectivité. Ce modèle est fondamental pour l'étude des réseaux sociaux. Avec D. Gardy et A. Jacquot, nous avons fait une analyse quantitative des logiques BCI et BCK [18]. Finalement, nous avons entamé un projet de recherche ambitieux pour résoudre le difficile problème de l'énumération asymptotique des lambda-termes. Un premier résultat a été obtenu avec D. Gardy et B. Gittenberger [61].

5.2 Projets de recherche

Comme nous avons essayé de le montrer, la génération aléatoire est un domaine actuellement très actif. Les techniques récentes ouvrent la possibilité par leur performance accrue à de vraies applications dans de multiples domaines allant des aspects les plus théoriques comme la physique statistique ou la théorie des groupes, à des aspects appliqués comme le test logiciel ou la simulation de réseaux sociaux. Notre approche a consisté à étudier et cerner le champ applicatif de la génération sous modèle de Boltzmann. Notre étude nous a permis de prouver la grande flexibilité de ces nouveaux concepts. En particulier, le passage aux classes multivariées, aux opérateurs différentiels et aux classes combinatoires multiplicatives s'est révélé être très naturel. Néanmoins, l'analyse en complexité de ces extensions est très largement plus sophistiquée. Nous avons donc ouvert plusieurs brèches qui constituent autant de perspectives de recherche à plus ou moins long terme. Nous détaillons dans la section suivante un certain nombre de ces perspectives qui constituent notre projet de recherche à venir.

5.2.1 Accroître l'expressivité intrinsèque des générateurs de Boltzmann

Vers les applications réelles ...

En complétant les travaux sur les générateurs de Boltzmann multiparamétré Ce point est à lui seul un programme de recherche ambitieux nécessitant un développement sur plusieurs années. La question sous-jacente est le développement de techniques d'analyse asymptotique dans le cadre non plus de l'analyse complexe univariée, mais multi-variée. C'est un très vaste champ d'étude faisant apparaître de multiple difficultés : topologie plus complexe induisant des phénomènes de changement de phase, une variété de type de distribution beaucoup plus riche que la simple distinction de trois types de distribution (piquée, plate et concentrée). Les enjeux des générateurs de Boltzmann multivariés sont primordiaux pour décrire les objets complexes issus des applications (graphe petit monde, structures secondaires en bio-informatique, strutures de données arborescentes,...).

En prolongeant les travaux sur les spécifications différentielles Un autre but est d'étendre nos travaux sur les équations différentielles de premier ordre au cadre des **systèmes de spécifications différentielles**. C'est à dire des spécifications faisant intervenir les opérateurs combinatoires différentiels "boîte" et "point". Cette extension permettra d'atteindre des classes d'objets fondamentaux tels que les marches aléatoires restreintes, les quad-tree, les arbres digitaux, etc.

En poursuivant le développement d'applications L'idée de pouvoir visualiser des instances "génériques" de grande taille est une source d'information importante pour orienter une étude en procurant de bonnes hypothèses à formuler. Les tas de sable, les tresses, les permutations à motif exclus, les cartes, les graphes sont autant d'objets combinatoires familiers pour lesquels avoir des générateurs efficaces serait un atout majeur, même si les générateurs de Boltzmann sont construit automatiquement à partir de spécifications, dans les exemples cités ci-dessous (tout comme pour les partitions planes) la complexité descriptive des objets nécessite un travail important au niveau des spécifications (bijection explicite, constructeur particulier, etc.).

Vers encore plus de diversité ...

En abordant la génération des classes à marquage généralisé Le champ des classes atteignables a été étendu en prenant en compte plus de types de marquage tels que les k**-colorations** ou

les classes colorées. Nous avons introduit la classe des objets **semi-étiquetées** (étant donné un objet a de taille n, a est semi-étiqueté s'il existe un $k \leqslant n$ tel que les atomes de a utilisent avec répétitions possibles toutes les couleurs dans $\{1, \cdots, k\}$). C'est un type de marquage qui nous semble avoir du sens. Il permet en particulier de décrire des objets importants tels que les chemins eulériens dans un graphe. L'extension des générateurs de Boltzmann aux classes semi-étiquetées est un de nos objectifs à moyen terme.

En développant les générateurs de Boltzmann supra-singuliers Commençons par rappeler une différence fondamentale entre la génération sous modèle de Boltzmann et la génération par méthode récursive. Dans la méthode récursive, il faut calculer les coefficients de la série génératrice, tandis que dans la méthode de Boltzmann, il faut évaluer cette série en un point. Ces deux calculs jouent le même rôle chacun dans leur cadre respectif, celui de permettre de faire des choix conditionnels durant la génération (comme s'arrêter ou continuer le processus par exemple). Or dans la méthode récursive, il n'y aucune limitation sur la manière dont croissent les coefficients, alors que parallèlement, dans la méthode de Boltzmann, il est impossible de prendre une valeur hors du domaine de convergence de la série et donc en particulier il faut que cette série soit analytique en 0 (ce qui impose une restriction sur la croissance des coefficients). Il nous faut donc plus en avant se questionner sur les raisons de cette limitation. Elles sont en fait assez simple, la définition de la distribution de Boltzmann induit que la probabilité que le processus donne un objet de taille infinie doit être nulle. Or dans certaines classes, il n'est pas possible de concevoir la génération d'objet de taille finie sans celle d'objet infinis. Afin d'éclairer un peu l'idée, regardons ensemble l'exemple suivant. Considérons les arbres binaires planaires non-étiquetés dont la spécification est $\mathcal{T} = \mathcal{Z} + \mathcal{Z} \times \mathcal{T}^2$, ces arbres ont une série génératrice "classique" $\frac{1-\sqrt{1-4\,z^2}}{2z}$. Mais il existe aussi une autre fonction qui vérifie l'équation fonctionnelle : $\tilde{C}(x) := \frac{1+\sqrt{1-4\,z^2}}{2z}$. Il est alors possible de définir une autre distribution de Boltzmann sur $\mathbb{N} \cup \infty$ par $\mathbb{P}_x(N = n) = \frac{c_n x^n}{C(x)}$ et $\mathbb{P}_x(N = \infty) = 1 - \frac{C(x)}{\tilde{C}(x)}$. En d'autres termes, il n'existe pas en général une seule distribution de Boltzmann mais plusieurs. Pour ce faire, il suffit qu'il existe plusieurs fonctions vérifiant l'équation liée à la spécification et telles que $\mathbb{P}_x(N = n) = \frac{c_n x^n}{C(x)}$ et $\mathbb{P}_x(N = \infty) = 1 - \frac{C(x)}{\tilde{C}(x)}$ définissent bien une probabilité (c'est-à-dire à valeurs dans $[0, 1]$ pour tout n). Cette vision permet d'étendre la génération de Boltzmann au même champ que la méthode récursive. Une analyse détaillée de la complexité du rejet anticipé est en projet ainsi qu'une complète description combinatoire sous-jacente.

En explicitant des règles de génération pour d'autres constructeurs En effet, les méthodes de construction de générateurs de Boltzmann ont été détaillées dans le cadre de la génération dans des classes d'objets combinatoires non étiquetées et étiquetées et ceci pour les constructeurs de base ($+$, $\times$, Seq, MSet, Cyc, etc.). Nous avons nous-même proposé des règles pour le produit de Hadamard et le cycle équicoloré. Néanmoins, des **constructeurs importants n'ont pas encore de traduction en termes de constructeur de générateurs**. La question du "**shuffle**" qui consiste à entremêler deux séquences et dont les applications sont variées en théorie des langages n'a été résolue que dans le cas des langages rationnels [36], le **cycle miroir** que l'on retrouve en théorie des graphes, la séquence sans répétition sont autant d'opérateurs dont il faudrait donner les règles pour la génération aléatoire sous modèle de Boltzmann. La question de générateurs efficaces pour les spécifications de deuxième ordre (comme par exemple $\textsc{Seq}(\mathcal{Z}\,\textsc{Seq}(\mathcal{A}))$,...) paraît aussi être une piste intéressante à explorer.

5.2.2 Optimiser et analyser les générateurs de Boltzmann

Le calcul de l'oracle La méthode Boltzmann troque le calcul des coefficients des séries génératrices contre l'évaluation de celles-ci en un point. Il est fondamental de trouver des moyens efficaces pour effectuer cette évaluation. Cette question a été étudiée par Pivoteau, Salvy et Soria [65], qui ont une approche utilisant la méthode de Newton. Néanmoins, toutes les méthodes jusqu'à présent connues se détériorent quand le point d'évaluation s'approche de la singularité dominante. Une étude détaillée pourrait permettre d'évaluer cette perte. Ceci est d'autant plus dommageable que c'est justement

proche de la singularité que sont les valeurs qui permettent de générer de gros objets. Nous envisageons dans le cas particulier des spécifications différentielles d'ordre 1, une approche par la méthode de Runge-Kutta qui semble constituer une piste prometteuse. Une autre approche envisageable dans le cas des langages rationnels et hors contexte, irréductibles et apériodiques, est d'utiliser le fait que le régime asymptotique apparaît très tôt et que l'on connait le type de la singularité dominante (un pôle simple dans le cas rationnel et une singularité d'ordre 1/2 dans le cas des langages hors contexte). Une évaluation en plusieurs points bien choisis (à distance convenable de la singularité dominante) devrait donc permettre d'approximer efficacement les valeurs a, c, ρ dans l'équivalent asymptotique $a + c(1 - z/\rho)^{-\alpha}$.

Le calibrage des paramètres La question du calibrage du paramètre de Boltzmann est aussi un point très important à discuter. C'est un prolongement naturel que Pivoteau, Salvy, Soria vont sans doute aborder. La question connexe qui consiste à estimer la singularité dominante liée à une série génératrice est aussi une question centrale qui reste globalement ouverte. Une voie à explorer serait de permettre aux générateurs de Boltzmann d' "auto-calibrer" leurs paramètres en procédant par analyse statistique de leurs propres productions.

L'estimation des erreurs d'arrondi Plusieurs personnes (O. Bodini, C. Nicaud, M. Soria,...) ont évoqué l'idée naturelle d'étudier le biais sur la distribution de Boltzmann occasionné par certaines approximations numériques durant le déroulement de l'algorithme. Cette étude a été effectuée pour la méthode récursive et devrait sans doute pourvoir se calquer pour les générateurs de Boltzmann.

Voilà ce qui constitue un premier champ de prospection très vaste, qui a pour objectif d'étendre et d'optimiser les méthodes de génération modernes. Certaines de ces approches font d'ores et déjà l'objet d'investigations, d'autres ne sont encore qu'à l'état embryonnaire, mais toutes font partie d'un mouvement cohérent dont la motivation intrinsèque est l'assise d'un nouveau modèle performant pour la génération aléatoire automatisable.

5.2.3 Application de l'analyse combinatoire et de la génération aléatoire à la physique statistique

Je ne peux conclure ce mémoire sans un mot sur mes autres centres d'intérêt et en particulier sur les pavages. Partons de l'exemple des pavages par losanges d'un grand hexagone. L'ensemble de tous ces pavages (que les physiciens appellent espace de configuration) est un avatar du modèle de dimère sur réseau. La figure ci-dessous montre un tirage aléatoire d'un pavage (figure 15).

On observe aisément sur cette figure qu'il existe une région circulaire inscrite dans l'hexagone dans laquelle les losanges semblent mis de manière non ordonnée alors que dans les zones périphériques les losanges semblent placés de manière ordonnée. Ce modèle fait en effet partie des rares exemples de modèle de la mécanique statistique où l'on peut effectivement prouver la présence d'une transition de phase (présence dans un pavage aléatoire de zone "gelée" et de zone "brassée"). Néanmoins, la prédiction de la présence d'une transition de phase est particulièrement difficile. La manière la plus efficace pour se convaincre de la présence d'un tel phénomène est de faire un tirage aléatoire uniforme sur l'espace des pavages et de "voir" s'il y a des zones d'ordre et de chaos. La preuve mathématique du phénomène sur le pavage par des losanges d'un hexagone a en effet été obtenue bien après la mise en évidence expérimentale par génération aléatoire du phénomène. La méthode CFTP a été ici prépondérante, mais elle présente néanmoins de multiples limitations. Le temps dans lequel le tirage est effectué est assez mauvais, ce qui n'autorise pas la génération d'objet très gros. Or, pour mettre en évidence des comportements limites, nous avons clairement besoin de générer des objets "limites". De plus, la méthode CFTP ne fonctionne que dans le cas particulier des modèles disposant intrinsèquement d'un treillis distributif des configurations. Or, par exemple, pour les pavages par losanges de n-gones avec $n > 6$, cette propriété n'est plus vérifiée. Dans un de ses articles, Propp indique cette restriction dans le problème ouvert suivant : de quelle manière peut-on généraliser la méthode CFTP de telle sorte qu'elle puisse s'étendre à des ensembles partiellement ordonnés plus généraux que les treillis distributifs ? Un premier objectif est donc de trouver un **générateur pour les**

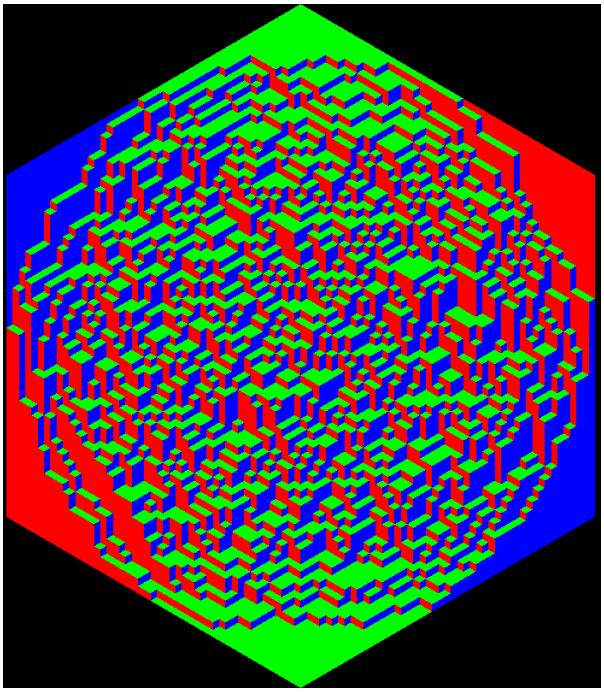


FIGURE 15: Pavage aléatoire d'un hexagone par des losanges (source web)

pavages des $(n \to d)$-zonotopes (qui sont la généralisation naturelle des pavages des n-gones) avec comme idée sous-jacente la recherche importante pour la physique statistique de modèles à transition de phase. Récemment, E. Fusy, C. Pivoteau et moi-même avons obtenu un générateur de Boltzmann efficace pour la génération de partitions planes, cette avancée est un premier pas vers la génération de partitions généralisées qui sont intimement liées au problème de pavages par losanges. Pour s'en convaincre grâce à la figure, il suffit de ne plus regarder celle-ci comme un objet en 2 dimensions (pavages) mais en 3 dimensions (empilement de cubes), ce que l'on voit alors est une partition plane dans une boîte cubique.

Un autre champ relatif à la théorie des pavages et à la génération aléatoire est la formation des quasi-cristaux. Une approche novatrice en deux phases a été proposée en 2008 par T. Fernique. Une première phase d'auto-assemblage permet de former des quasi-cristaux avec des erreurs. Une seconde phase corrige progressivement les erreurs en effectuant stochastiquement des transformations locales (flips). **L'analyse de ces processus (temps de convergence suivant les conditions initiales) apporterait une compréhension théorique sur la croissance et la formation des quasi-cristaux** qui reste actuellement un défi à relever. Ce thème est l'objet du PEPS Stochaflip. Idéalement, à terme le quasi-cristal satisfait toutes les règles de couplage et donc consiste en une structure quasi-cristalline parfaite. Cette approche est compatible avec les expérimentations, où des quasi-cristaux sont obtenus à partir d'une soupe chaude par un lent refroidissement durant lequel des flips se produisent réellement. Il reste pourtant encore non prouvé si la notion de refroidissement

peut se réduire aux seuls processus de flips ou si d'autres mécanismes doivent être pris en compte. Dans un travail réalisé, nous avons obtenu plusieurs résultats sur le temps de convergence dans un modèle unidimensional de quasi-cristal. Mais, une telle simplification ne permet pas de capturer la réelle complexité de la croissance dans le cas tridimensionnel. Notre projet de recherche consiste à étendre nos techniques de sorte à atteindre un modèle plus réaliste.

5.3 Le petit mot de la fin

Comme toutes les bonnes théories, le modèle de Boltzmann est d'une grande flexibilité, il passe sans difficulté aux multivariés, aux spécifications différentielles, à différents types de marquage, alors merci à Boltzmann et ces joyeux disciples (Duchon, Flajolet, Louchard, Schaeffer et les autres ...).

6 Liste des publications et communications

6.1 Conférences internationales avec comité de lecture et publications d'actes

1. O. Bodini, D. Gardy and B. Gittenberger (2011), "Lambda-terms of Bounded Unary Height", in ANALCO'11, San Franscico, USA, Proceedings of the Eighth Workshop on Analytic Algorithmics and Combinatorics, pp12. (to appear in SIAM Proceedings)
2. O. Bodini, Y. Ponty (2010) "Multi-dimensional Boltzmann Sampling of Languages", AOFA'10, Wien, Autriche, p49–64. DMTCS Proceedings, North America, sep. 2010. Available at : http ://www.dmtcs.org/dmtcs-ojs/index.php/proceedings/article/view/dmAM0104/3264.
3. A. Darrasse, S.K. Hwang, O. Bodini, M. Soria (2010) "The connectivity-profile of random increasing k-trees", in ANALCO'10, Austin, USA, Proceedings of the Seventh Workshop on Analytic Algorithmics and Combinatorics, p99-106, SIAM, ISBN : 978-0-898719-33-8
4. O. Bodini, T. Fernique, D. Regnault (2010) "Stochastic flips on two letters words", in ANALCO'10, Austin, USA, Proceedings of the Seventh Workshop on Analytic Algorithmics and Combinatorics, p48-55, SIAM, ISBN : 978-0-898719-33-8
5. O. Bodini, T. Fernique, E. Remila (2009) "Distance on lozenge tillings", in Discrete Geometry for Computer Imagery, DGCI'09, Montreal, Canada, LNCS Proceedings, isbn : 3-642-04396-8, p79-91, volume 5810, Springer Berlin.
6. O. Bodini, J. Lumbroso (2009) "Optimal partial tiling on Manhattan polyominoes", in Discrete Geometry for Computer Imagery, DGCI'09, Montreal, Canada. LNCS Proceedings, isbn : 3-642-04396-8, p240-251, volume 5810, Springer Berlin.
7. O. Bodini, T. Fernique, D. Regnault (2009) "Cristallization by stochastic flips", in Proccedings of Aperiodic'09, Owford, Angleterre. Journal of Physics : Conference Series, volume 226, numero 1, p1–6, url=http ://stacks.iop.org/1742-6596/226/i=1/a=012022, 2010.
8. O. Bodini, A. Darrasse and M. Soria (2008) "Distances in random Apollonian network structures", in International Conference on Formal Power Series and Algebraic Combinatorics, FPSAC'08, Santiago, Chili.
   Available at : http ://arxiv.org/PS_cache/arxiv/pdf/0712/0712.2129v1.pdf
9. O. Bodini, A. Darrasse and M. Soria (2008) "Distances in random Apollonian network structures", in International Conference on Formal Power Series and Algebraic Combinatorics, FPSAC'08, Santiago, Chili. DMTCS proceedings, p308–327.
10. O. Bodini and S. Lefranc (2006) "How to Tile by Dominoes the Boundary of a Polycube". In Discrete Geometry for Computer Imagery, DGCI'06, Sgedge, Hongrie. Lecture Notes in Computer Science, 2006, Volume 4245/2006, p630–638, DOI : 10.1007/11907350-53
11. O. Bodini and E. Rivals (2006) "Tiling an Interval of the Discrete Line.", In Annual Symposium on Combinatorial Pattern Matching, CPM'06, Barcelone, Espagne. Lecture Notes in Computer Science, 2006, Volume 4009/2006, p117–128, DOI : 10.1007/11780441-12
12. O. Bodini and T. Fernique (2006) "Planar Dimer Tilings", In International Computer Science Symposium in Russia, CSR'06, Moscou, Russie. Lecture Notes in Computer Science, 2006, Volume 3967/2006, p104–113, DOI : 10.1007/11753728-13
13. O. Bodini and B. Nouvel (2004) "$\mathbb{Z}$-Tiling and Grobner Basis", 10 pages, In International Workshop on Combinatorial Image Analysis, IWCIA'04, Auckland, Nouvelle zélande. Combinatorial

Image Analysis, Lecture Notes in Computer Science, 2005, Volume 3322/2005, p137–150, DOI : 10.1007/978-3-540-30503-3-11

14. O. Bodini (2003) "Tiling a rectangle with polyominoes", In Discrete Models for Complex Systems, DMCS'03, Lyon, France. Discrete Mathematics and Theoretical Computer Science AB(DMCS), 2003, p81–88
15. O. Bodini (2002) "On the minimum size of a contraction universal tree", WG'02, Cesky Krumlov, République Tchèque, in Graph-Theoretic Concepts in Computer Science Lecture Notes in Computer Science, 2002, Volume 2573/2002, p25–34, DOI : 10.1007/3-540-36379-3-3
16. O. Bodini (2003) "Tilings on the butterfly lattice", EuroComb'03, Pragues, République Tchèque. European Journal of Combinatorics Volume 27, Issue 7, October 2006, p1082–1087 Eurocomb'03 - Graphs and Combinatorial Structures
17. O. Bodini (2001) "Tiling a Manhattan polyomino with bars", EuroComb'01, Barcelone, Espagne. Electronic Notes in Discrete Mathematics Volume 10, November 2001, p27–29 Comb01, Euroconference on Combinatorics, Graph Theory and Applications

6.2 Journaux internationaux

1. O. Bodini, O. Roussel, M.Soria (2010) "Boltzmann samplers for first order combinatorial differential equations", Discrete Applied Mathematics, Special issue : LAGOS'09 (à paraître).
2. O. Bodini, E. Fusy, C. Pivoteau (2009) "Random Sampling of Plane Partitions" (version longue), 25 pages. In Combinatorics, Probability and Computing, Published Online by Cambridge University Press 02 Nov 2009, [17]
3. O. Bodini, T. Fernique and E. Rémila (2007) "A Characterization of Flip-accessibility for Rhombus Tilings of the Whole Plane" 8 pages. In Inf. Comput, [15]
4. O. Bodini and D. Jamet (2003) "Tiling a pyramidal polycube with "dominoes"", 16 pages, In Discrete Mathematics & Theoretical Computer Science, [22]
5. O. Bodini and M. Latapy (2003) "Generalized tilings with height functions", 19 pages. In Morfismos (7) juin 2003, [23]
6. O. Bodini and E. Remila (2003) "On edge tricolorations of triangulations of simply connected surfaces", 10 pages, numéro spécial pavage 2003 de TCS,[28]

6.3 Conférences internationales avec comité de lecture sans publication d'actes

1. O. Bodini, D. Gardy, A. Jacquot (2010) "Asymptotics and random sampling for formulae in intuitionist logical systems", Gascom'10, Montreal, Canada.
2. O. Bodini, A. Jacquot (2010) "Boltzmann samplers for v-balanced colored cycles", Gascom'10, Montreal, Canada.
3. O. Bodini, D. Gardy, O. Roussel (2010) "Boys-and-girls birthdays and Hadamard products", Proceedings of Lattice Paths and applications'10, Siena, Italie, 5p. Paper version submitted to Special Issue of the Journal of Statistical Planning and Inference. 17pp.
4. O. Bodini, A. Jacquot (2008) "Boltzmann Samplers for Colored Combinatorial Objects", Gascom'08, Bibiena, Italie.
5. O. Bodini, T. Fernique and E. Rémila (2007) "A Characterization of Flip-accessibility for Rhombus Tilings of the Whole Plane". Special Issue : 1st International Conference on Language and Automata Theory and Applications (LATA 2007), Taragone, Espagne.
6. O. Bodini, T. Fernique and E. Rémila (2007) "Characterizations of Flip-Accessibility for Domino-Tilings of the Whole Plane". In International Conference on Formal Power Series and Algebraic Combinatorics, FPSAC'07, Tianjin, Chine. 12pp.
Available at : http ://hal.archives-ouvertes.fr/lirmm-00149373/.

7. O. Bodini, C. Pivoteau and E. Fusy (2006) "Random Sampling of Plane Partitions" Gascom'06, Nancy, France.
8. O. Bodini and E. Remila (2003) "On edge tricolorations of triangulations of simply connected surfaces", in International Conference on Formal Power Series and Algebraic Combinatorics, FPSAC'03, Linkeping, Suède.

6.4 Revues nationales, thèse

1. O. Bodini, P. Duchet and S. Lefranc (2001) "Autour d'un théorème d'Erdös sur les combinaisons à coefficients + ou -1 des premiers carrés", Revue de l'enseignement supérieur (sept 2001) p. 3-8. [10]
2. Thèse en mathématiques de l'université Paris 6 intitulée "Approches combinatoire et algébrique de problèmes de pavage", 173 pages, soutenue le 17 novembre 1999.

Bibliographie

[1] D.B. Arnold and M.R. Sleep. Random generation of balanced parenthesis strings. *ACM TOPLAS*, 2(2) :122–128, january 1980.

[2] E. Bender, L. B. Richmond, and S.G. Williamson. Central and local limit theorems applied to asymptotic enumeration III : Matrix recursions. *JCT serie A*, 35 :263–278, 1983.

[3] Edward A. Bender and Donald E. Knuth. Enumeration of plane partitions. *J. Comb. Theory, Ser. A*, 13(1) :40–54, 1972.

[4] F. Bergeron, P. Flajolet, and B. Salvy. Varieties of increasing trees. *Springer*, pages 24–48, 1992.

[5] F. Bergeron, G. Labelle, and P. Leroux. *Combinatorial species and tree-like structures*. Cambridge University Press, 1998.

[6] A. Bertoni, P. Massazza, and R. Radicioni. Random generations of words in regular languages with fixed occurrences of symbols. In *Proceedings of Words'03*, volume 27, pages 332–343. TUCS Gen. Publ., 2003.

[7] O. Bodini. On the minimum size of a contraction universal tree. In *28th Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG'02)*, volume 2573 of *Lecture Notes in Computer Science*, pages 25–34. Springer, 2002.

[8] O. Bodini. Tiling a rectangle with polyominoes. In Michel Morvan and Eric Rémila, editors, *Discrete Models for Complex Systems, DMCS'03, Lyon, France, June 16-19, 2003*, volume AB of *Discrete Mathematics and Theoretical Computer Science Proceedings*, pages 81–88. DMTCS, 2003.

[9] O. Bodini, A. Darrasse, and M. Soria. Distances in random Apollonian network structures. In *DMTCS Proceedings 20th Annual international Conference on Formal Power Series and Combinatorics (FPSAC 2008)*, pages 307–318, Valparaiso-Viña del Mar Chili, 2008. 12 pages.

[10] O. Bodini, P. Duchet, and S. Lefranc. Autour d'un théorème d'erdos sur les combinaisons à coefficients + ou -1 des premiers carrés. *Revue de l'enseignement supérieur*, pages 3–8, 2001.

[11] O. Bodini and T. Fernique. Planar dimer tilings. In Dima Grigoriev, John Harrison, and Edward A. Hirsch, editors, *Computer Science - Theory and Applications, First International Computer Science Symposium in Russia, CSR 2006, St. Petersburg, Russia, June 8-12, 2006, Proceedings*, volume 3967 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 2006.

[12] O. Bodini, T. Fernique, and D. Regnault. Crystallization by stochastic flips. *Journal of Physics : Conference Series*, 226(1) :1–6, 2010.

[13] O. Bodini, T. Fernique, and D. Regnault. Stochastic flips on two-letter words. In *2010 Proceedings of the Sixth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 48–56, Austin États-Unis d'Amérique, 2010.

[14] O. Bodini, T. Fernique, and E. Rémila. A characterization of flip-accessibility for domino tilings of the whole plane. In *FPSAC'07*, page 12, 2007.

[15] O. Bodini, T. Fernique, and E. Rémila. A characterization of flip-accessibility for rhombus tilings of the whole plane. *Inf. Comput*, 206(9-10) :1065–1073, 2008.

[16] O. Bodini, T. Fernique, and E. Remila. Distances on lozenge tilings. In Srecko Brlek, Christophe Reutenauer, and Xavier Provencal, editors, *Proceedings of the 15th IAPR international conference on Discrete geometry for computer imagery*, Lecture Notes in Computer Science, pages 240–251. Springer Berlin / Heidelberg, 2009.

[17] O. Bodini, E. Fusy, and C. Pivoteau. Random sampling of plane partitions. *Comb. Probab. Comput.*, 19(2) :201–226, 2010.

[18] O. Bodini, D. Gardy, and A. Jacquot. Asymptotics and random sampling for formulae in intuitionist logical systems. In *Proceedings of Gascom'10*, 2010.

[19] O. Bodini, D. Gardy, and O. Roussel. Boys-and-girls birthdays and hadamard products. In *Proceedings of Lattice Paths and Applications'10*, page 5, 2010.

[20] O. Bodini and A. Jacquot. Boltzmann Samplers For Colored Combinatorial Objects. In *Proceedings of Gascom'08*, 2008.

[21] O. Bodini and A. Jacquot. Boltzmann samplers for v-balanced colored cycles. In *Proceedings of Gascom'10*, 2010.

[22] O. Bodini and D. Jamet. Tiling a pyramidal polycube with dominoes. *Discrete Mathematics & Theoretical Computer Science*, 9(2) :241–254, 2007.

[23] O. Bodini and M. Latapy. Generalized tilings with height functions. *Morfimos*, 7 :10, 2003.

[24] O. Bodini and S. Lefranc. How to tile by dominoes the boundary of a polycube. In Attila Kuba, László G. Nyúl, and Kálmán Palágyi, editors, *Discrete Geometry for Computer Imagery, 13th International Conference, DGCI 2006, Szeged, Hungary, October 25-27, 2006, Proceedings*, volume 4245 of *Lecture Notes in Computer Science*, pages 630–638. Springer, 2006.

[25] O. Bodini and J. Lumbroso. Optimal partial tiling of manhattan polyominoes. In Srecko Brlek, Christophe Reutenauer, and Xavier Provencal, editors, *Proceedings of the 15th IAPR international conference on Discrete geometry for computer imagery*, volume 5810 of *Lecture Notes in Computer Science*, pages 79–91. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-04397-0-8.

[26] O. Bodini and B. Nouvel. Z-tilings of polyominoes and standard basis. In *Combinatorial Image Analysis*, volume 3322/2005, pages 137–150. Lecture Notes in Computer Science, 2005.

[27] O. Bodini and Y. Ponty. Multi-dimensional boltzmann sampling of languages. *DMTCS Proceedings*, 0(01) :49–64, 2010.

[28] O. Bodini and E. Rémila. Tilings with trichromatic colored-edges triangles. *Theoretical Computer Science*, 319(1-3) :59 – 70, 2004. Combinatorics of the Discrete Plane and Tilings.

[29] O. Bodini and E. Rivals. Tiling an interval of the discrete line. In *Combinatorial Pattern Matching*, volume 4009 of *Lecture Notes in Computer Science*, pages 117–128. Springer, 2006.

[30] O. Bodini, O. Roussel, and M. Soria. Boltzmann samplers for first order combinatorial differential equations. In *to appear in Discrete Applied Mathematics.*

[31] M. Bodirsky, E. Fusy, M. Kang, and S. Vigerske. An unbiased pointing operator for unlabeled structures, with applications to counting and sampling. In *SODA '07 : Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 356–365, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[32] D.M. Bressoud. *Proofs and confirmations : the story of the alternating sign matrix conjecture.* Cambridge University Press, New York, NY, USA, 1999.

[33] R. Breukelaar, E.D. Demaine, S. Hohenberger, H.J. Hoogeboom, W.A. Kosters, and D. Liben-Nowell. Tetris is hard, even to approximate. *International Journal of Computational Geometry and Applications*, 14(1–2) :41–68, 2004.

[34] R. Cerf and R. Kenyon. The low-temperature expansion of the wulff crystal in the 3d ising model. *Communications in Mathematical Physics*, 222 :147–179, 2001. 10.1007/s002200100505.

[35] H. Cohn, M. Larsen, and J. Propp. The shape of a typical boxed plane partition. *New York J. Math*, 4 :137–166, 1998.

[36] A. Darasse, K. Panagiotou, O. Roussel, and M. Soria. Boltzmann sampler for the shuffle product. In *Proceedings of Gascom'10*, 2010.

[37] A. Darrasse, H.K. Hwang, O. Bodini, and M. Soria. The connectivity-profile of random increasing k-trees. In *2010 Proceedings of the Sixth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 99–106, Austin États-Unis d'Amérique, 2010.

[38] A. Denise, O. Roques, and M. Termier. Random generation of words of context-free language saccording to the frequencies of letters. In D. Gardy and A. Mokkadem, editors, *Mathematics and Computer Science : Algorithms,Trees, Combinatorics and probabilities*, Trends in Mathematics, pages 113–125. Birkhaüser, 2000.

[39] M. Drmota. Systems of functional equations. *Random Structures and Algorithms*, 10(1-2) :103–124, 1997.

[40] P. Duchon, P. Flajolet, G. Louchard, and G. Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probablity, and Computing*, 13(4–5) :577–625, 2004. Special issue on Analysis of Algorithms.

[41] W. Feller. *An introduction to probability theory and its applications*, volume 1. Wiley & Sons, New York, 1968.

[42] P. Flajolet, E. Fusy, and C. Pivoteau. Boltzmann sampling of unlabelled structures. In SIAM Press, editor, *Proceedings of ANALCO'07 (Analytic Combinatorics and Algorithms) Conference*, volume 126, pages 201–211, 2007.

[43] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[44] P. Flajolet, P. Zimmerman, and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science*, 132(1-2) :1–35, 1994.

[45] Ph. Flajolet, D. Gardy, and L. Thimonier. Birthday Paradox, Coupon Collectors, Caching Algorithms and Self-Organizing Search. *Discrete Applied Mathematics*, 39 :207–229, 1992.

[46] H. Fredricksen and J. Maiorana. Necklaces of beads in k colors and k-ary de bruijn sequences. *Discrete Mathematics*, 23(3) :207 – 210, 1978.

[47] E.R. Gansner. Matrix correspondences of plane partitions. *Pacific J. Math.*, 92(2) :295–315, 1981.

[48] D.H. Greene. *Labelled formal languages and their uses*. PhD thesis, Stanford University Stanford, CA, USA, 1983.

[49] H.J. Hoogeboom and W.A. Kosters. Tetris and decidability. *Inf. Process. Lett.*, 89(6) :267–272, 2004.

[50] H.-K. Hwang. Théorèmes limites pour les structures combinatoires et les fonctions arithmétiques. *Thèse, Ecole polytechnique*, 1994.

[51] D. E. Knuth. Permutations, matrices, and generalized Young tableaux. *Pacific J. Math.*, 34 :709–727, 1970.

[52] M. Krasner and B. Ranulak. Sur une propriété des polynômes de la division du cercle. *Comptes rendus de l'Académie des Sciences Paris*, 240 :397–399, 1937.

[53] C. Krattenthaler. Another involution principle-free bijective proof of stanley's hook-content formula. *Journal of Combinatorial Theory, Series A*, 88(1) :66 – 92, 1999.

[54] J. Ma and E. J. Janse van Rensburg. Rectangular vesicles in three dimensions. *J. Phys. A*, 38(19) :4115–4147, 2005.

[55] P. A. MacMahon. Memoir on the theory of the partitions of numbers. vi : Partitions in two-dimensional space, to which is added an adumbration of the theory of partitions in three-dimensional space. *Phil. Trans. Roy. Soc. London Ser. A*, 211 :345–373, 1912.

[56] T. Maeda and T. Nakatsu. Amoebas and Instantons. *International Journal of Modern Physics A*, 22 :937–983, 2007.

[57] L. Mutafchiev. The size of the largest part of random plane partitions of large integers. *Integers*, 6A13, 2006.

[58] L. Mutafchiev and E. Kamenov. On the Asymptotic Formula for the Number of Plane Partitions of Positive Integers. *ArXiv Mathematics e-prints*, January 2006.

[59] K. Nishimura and M. Sibuya. Occupancy with two types of balls. *Annals of the Institute for Statistical Mathematics*, 40(1) :77–91, 1988.

[60] J-C. Novelli, I. Pak, and A.V. Stoyanovskii. A direct bijective proof of the hook-length formula. *Discrete Math. Theor. Comput. Sci.*, 1(1) :53–67, 1997.

[61] B. Gittenberger O. Bodini, D. Gardy. Lambda-terms of bounded unary height. In *ANALCO'11, USA, San Franscico*, January 2011.

[62] A. Okounkov and N. Reshetikhin. Correlation function of Schur process with application to local geometry of a random 3-dimensional Young diagram. *ArXiv Mathematics e-prints*, July 2001.

[63] A. Okounkov and N. Reshetikhin. Random skew plane partitions and the pearcey process. *Communications in Mathematical Physics*, 269 :571–609, 2007. 10.1007/s00220-006-0128-8.

[64] I. Pak. Hook length formula and geometric combinatorics. *Lothar. Comb*, 46 :4–6, 2000.

[65] C. Pivoteau, B. Salvy, and M. Soria. Boltzmann oracle for combinatorial systems. In *Algorithms, Trees, Combinatorics and Probabilities*, pages 475–488. Discrete Mathematics and Theoretical Computer Science, 2008. Proceedings of the Fifth Colloquium on Mathematics and Computer Science. Blaubeuren, Germany. September 22-26, 2008.

[66] T. Yu. Popova. Limit theorems in a model of distribution of particles of two types. *SIAM Journal on Theory of Probability and its Applications*, 6(13) :511–516, 1968.

[67] J. Propp. Generating random elements of finite distributive lattices. *Electronic Journal of Combinatorics*, 4 :pp., 1997.

[68] J. Propp and D.B. Wilson. How to get a perfectly random sample from a generic markov chain and generate a random spanning tree of a directed graph. *J. Algorithms*, 27(2) :170–217, 1998.

[69] R. Radicioni. *Holonomic power series and their applications to languages.* PhD thesis, Facoltà di scienze matematiche, fisiche et naturali, Università degli studi di Milano, 2006.

[70] J.L. Rémy. Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire. *RAIRO Theoretical Informatics and Applications*, 19(2) :179–195, 1985.

[71] O. Roussel and M. Soria. Boltzmann sampling of ordered structures. *Electronic Notes in Discrete Mathematics*, 35 :305–310, 2009.

[72] O. Roussel and M. Soria. Boltzmann sampling of ordered structures. *Electronic Notes in Discrete Mathematics*, 35 :305–310, 2009.

[73] A. Sapounakis and P. Tsikouras. On k-colored motzkin words. *Journal of Integer Sequences*, 7(04.2.5), 2004.

[74] B.I. Selivanov. On the waiting time in a scheme for the random allocation of colored particles. *Diskretnaya Matematika*, 7(1) :134–144, 1995.

[75] A. Vershik. Statistical mechanics of combinatorial partitions, and their limit shapes. *Functional Analysis and Its Applications*, 30 :90–105, 1996. 10.1007/BF02509449.

[76] A. M. Vershik. Statistical mechanics of combinatorial partitions, and their limit shapes. *Functional Analysis and Its Applications*, 30(2) :90–105, 1996.

[77] M.C. Wendl. Collision probability between sets of random variables. *Statistics and Probability Letters*, 64(3) :249–254, 2003.

[78] D. B. Wilson. Determinant algorithms for random planar structures. In *In Proc. of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 258–267, 1997.

[79] E. M. Wright. Asymptotic partition formulae, i : Plane partitions. *Quart. J. Math. Oxford*, 49 :145–164, 1999.

[80] A. Young. On quantitative substitutional analysis. *Proc. Lond. Math. Soc.*, 33 :97–146, 1901.

[81] D. Zeilberger. Proof of the alternating sign matrix conjecture. *Electronic Journal of Combinatorics*, 3(2) R13 :1–84, 1996.