Conception et analyse d'algorithmes

Robert Cori, Guillaume Hanrot, Claire Kenyon et
Jean-Marc Steyaert

Troisième partie

Applications

CHAPITRE 8

Algorithmes pour la biologie moléculaire

Les techniques de conception algorithmique et preuves de NP-complétude étudiées précédemment s'utilisent couramment dans les domaines d'application de la'lgorithmique, lesquels ne se retreignent pas à l'informatique et télécommunications. Dans ce chapitre, nous donnons quelques exemples d'applications à la biologie moléculaire.

1. Introduction

L'ADN contient les instructions permettant à la cellule de fonctionner. Il consiste en deux brins formant une hélice double.

2. Alignement de familles de séquences

Motivation. Les bases de données pour les protéines les regroupent habituellement en familles. Lorsqu'on séquence un nouveau gène, sa transcription est un mot sur l'alphabet $\{A, C, G, U\}$; on commence par le traduire en une suite d'acides aminés en utilisant une table qui à chaque groupe de trois lettres consécutives, $\{A, C, G, U\}^3$, arrocie l'un des 61 codons codant l'un des 20 amino-acides (ou l'un des 3 codons "STOP" codant la fin du gène.) Ceci définit une nouvelle protéine. Quelles sont les fonctions de cette protéine, et comment est-elle structurée en trois dimensions ? Pour répondre à cette question, la méthode usuelle consiste à consulter une base de données de protéines déjà analysées, pour y trouver une famille de protéines qui "ressemblent" à la nouvelle protéine. On postule alors que la nouvelle molécule partagera probablement la plupart des propriétés observées chez les autres membres de la famille. Pour mener à bien ce projet, une possibilité est d'avoir, pour chaque famille, une représentation consistant en un alignement multiple alignant les séquences de tous ses membres.

Définition du problème. L'alphabet Σ considéré est ici l'ensemble des acides aminés, alphabet de taille 20. Le problème se prend en entrée un entier k et une famille de k chaînes de caractères $S_1, S_2, \dots, S_k$ (les protéines de la famille), et donne en sortie un alignement des k séquences, c'est-à-dire k nouvelles chaînes de caractères $S'_1, S'_2, \dots, S'_k$, sur l'alphabet augmenté où on a ajouté le caractère "espace blanc" noté Δ, sachant que tous les S'_i doivent avoir même longueur, et que S_i se déduit de S'_i par effacement des blancs.

Il s'agit de trouver le "meilleur" alignement ; une définition populaire de cette notion est l'alignement qui minimise la somme des distances entre toutes les paires de séquences.

La distance entre deux séquences est la sommes des distances caractère par caractère :

$$\sum_{1\leq i<j\leq k}\sum_{1\leq \ell\leq |S'_i|}\delta(S'_i[\ell],S'_j[\ell]),$$

avec δ fonction de distance de $(\Sigma\cup\{\Delta\})^2$ vers les entiers positifs (distance, donc symétrique, nulle pour tout (σ,σ) et seulement pour ces paires, et satisfaisant l'inégalité triangulaire). Cette distance modélise les similarités entre acides aminés du point de vue de leurs propriétés physico-chimiques. Par exemple, les 7 acides aminés de charge globale nulle mais distribuée irrégulièrement sur l'acide aminé ont tendance à former des liens hydrogènes avec l'eau, et sont donc similaires les uns aux autres.

Alignement de deux séquences. Lorsque $k = 2$, on se trouve avec le problème d'alignement de deux séquences vu au chapitre 2. Ce problème se résoud par programmation dynamique en utilisant un tableau 2-dimensionnel, et l'algorithme a complexité $O(\ell_1\ell_2)$ pour l'alignement de deux séquences de longueurs ℓ_1 et ℓ_2 respectivement.

NP-complétude. Le problème de l'alignement de séquences multiples peut être résolu en généralisant l'algorithme d'alignement de deux séquence, avec ici un tableau k-dimensionnel ; mais la complexité de l'algorithme devient alors $O(n^k)$ pour k séquences de longueur n, ce qui n'est pas polynomial en la taille des entrées. Le théorème ci-dessous montre que la recherche d'un algorithme polynomial est vouée à l'échec si $P \neq NP$.

THÉORÈME 8.1. *Soit Σ un alphabet fixé. Etant donné un entier k, une famille $S_1,\dots,S_k$ de chaînes de caractères sur Σ, une fonction de distance δ sur $\Sigma\cup\Delta$, et un entier C, décider s'il existe un alignement de valeur au plus C est un problème NP-complet.*

Le problème est clairement dans NP.

La réduction se fait à partir du problème de couverture des arêtes d'un graphe par des sommets.

On utilise l'alphabet $\Sigma = \{a, b, 0, 1, c, d\}$ et la fonction δ ci-dessous, qui est clairement une distance.

	a	b	0	1	c	d	Δ
a	0	1	1	1	1	2	2
b	1	0	1	1	2	1	2
0	1	1	0	2	2	2	2
1	1	1	2	0	2	1	2
c	1	2	2	2	0	2	1
d	2	1	2	1	2	0	2
Δ	2	2	2	2	1	2	0

Soit G le graphe à n sommets et ℓ arêtes, et K l'entier donné en entrée du problème de couverture par sommets.

Pour chaque arête $\{v_i, v_j\}$ de G, avec $i < j$, on définit la séquence $a^{3i}ba^{3(j-i)-2}ba^{3(n-j)+3}$, de longueur $3n+3$. On ajoute de multiples copies de deux séquences particulières, C_1 copies de $t = c(001)^n 00c$ de longueur $r = 3n + 4$, et C_2 copies de $x = cd^K c$, où C_1 et C_2 sont choisis de manière appropriée. Ceci définit une famille F de séquences.

L'idée principale de la preuve qu'on a une réduction valide est de montrer que tout alignement optimal de F doit avoir une forme particulière : il doit avoir longueur r, le blanc des séquences codant les arêtes doit être ou bien tout au début ou bien tout à la fin, les séquences identiques sont alignées identiquement, l'alignement de x ne commence ni ne finit par un blanc, et les d de x sont alignés avec des 1 de t. Minimiser le coût de l'alignement revient alors à aligner les b des séquences-arêtes avec les d des x, autant que possible ; mais, pour une séquence-arête donnée, on ne peut aligner les deux b avec un d, et on ne pourra donc en aligner que 0 ou 1 ; il sera possible d'aligner un des b de toute séquence-arête avec un d si et seulement si G possède un couverture comportant K sommets.

On se borne alors à rechercher un algorithme d'approximation.

Une 2-approximation. Voici un algorithme d'approximation pour l'alignement d'une famille de k séquences, appelé l'"algorithme d'alignement étoilé au centre".

(1) Pour chaque paire i, j telle que $1 \leq i < j \leq k$, en utilisant l'algorithme d'alignement de deux séquences, calculer $D(S_i, S_j)$.

(2) Soit S^* le S_i qui minimise $\sum_{j \neq i} D(S_i, S_j)$. La chaîne de caractères S^* est appelée le centre de la famille. Quitte à renuméroter, on peut supposer que $S^* = S_1$. On pose $S'_1 \leftarrow S_1$.

(3) Pour j allant de 2 à k faire :

- Aligner S_i et S'_1 en utilisant l'algorithme d'alignement de deux séquences.
- Ceci définit S'_i et modifie S'_1 en y insérant des blancs.
- Mettre à jour $S'_2, \ldots, S'_{i-1}$ en y insérant des blancs aux mêmes endroits que dans S'_1.

La complexité en temps est facile à calculer pour k séquences de longueur n :

$$O(\frac{k(k-1)}{2} n^2 + \sum_{2 \leq j \leq k} O(jn \cdot n)) = O(k^2 n^2).$$

L'analyse de la valeur de l'alignement demande un peu plus de travail. Le résultat $S'_1, \ldots, S'_k$ de l'algorithme a valeur :

$$\sum_{1 \leq i < j \leq k} D(S'_i, S'_j) \leq \sum_{1 \leq i < j \leq k} D(S'_i, S'_1) + D(S'_1, S'_j) = (k-1) \sum_{1 \leq i \leq k} D(S'_i, S'_1) = (k-1) \sum_{1 \leq i \leq k} D(S_i, S_1).$$

La première inégalité découle de l'inégalité triangulaire, et la dernière découle du coût nul de l'alignement des blancs : $\delta(-, -) = 0$.

Soit $S^*_1, \ldots, S^*_k$ l'alignement optimal. Sa valeur satisfait :

$$\sum_{1 \leq i < j \leq k} D(S^*_i, S^*_j) \geq \sum_{1 \leq i < j \leq k} D(S_i, S_j) = \frac{1}{2} \sum_i \sum_{j \neq i} D(S_i, S_j) \geq \frac{1}{2} \sum_i \sum_j D(S_i, S_1) = \frac{k}{2} \sum_j D(S_i, S_1).$$

La dernière inégalité découle de la définition du centre S_1.

En comparant ces deux expressions, on en déduit que la valeur de l'alignement multiple obtenu par notre algorithme excède la valeur optimale d'un facteur au plus $2(k-1)/k < 2$, et on a donc bien une 2-approximation.

3. Recherche d'un motif dans un mot, et alignement local

Motivation. L'alignement global de deux protéines a un sens lorsqu'elles appartiennent à la même famille. En revanche, en général, lorsqu'on compare deux séquences anonymes, elles n'ont aucune raison de bien s'aligner, mais certaines de leurs sections internes (facteurs) peuvent se ressembler, dans des motis qui correspondent à certaines propriétés structurelles ou fonctionnelles. Il est alors important, étant donné deux séquences, de trouver les motifs qui sont similaires dans l'une et l'autre des deux séquences.

Dans le cas “pur" où il n'y a pas d'erreur, il faut repérer les motifsqui sont exactement présents dans les deux séquences. Un cas traditionnel est celui où le motif est déjà connu. La recherche d'un motif dans un texte est un grand classique de l'informatique.

Recherche de motif dans un texte. Etant donné un motif P de longueur m et un texte T de longueur n, on souhaite décider si P est un facteur de T. L'algorithme naîf pour cela a complexité $O(mn)$. L'algorithme de Knuth, Morris et Pratt rappelé ci-dessous a complexité $O(m+n)$.

Pour $i \in \{1, \dots, m\}$, soit $s(i)$ le nombre de caractères du plus grand suffixe de $P[1 \dots i]$ qui soit également un préfixe de P, sous la contrainte supplémentaire que $P[i+1]$ soit différent de $P[s(i)+1]$. Supposons cette fonction $i \mapsto s(i)$ déjà précalculée.

L'algorithme essaie d'aligner P sous $T[k+1 \dots k+m]$ sachant que les i premiers caractères de P concordent avec ceux de $T[k+1 \dots k+m]$.

Il compare $P[i+1]$ à $T[k+i+1]$.

S'il y a égalité, on s'est ramené au problème de l'alignement de P sous $T[k+1 \dots k+m]$ sachant que les $i+1$ premiers caractères concordent.

S'il n'y a pas égalité, il faut essayer un autre alignement en déplaçant P vers la droite du texte T. De combien de caractères ? Soit P' le motif décalé. Lorsqu'il y a recouvrement partiel entre P et P', le début de P' doit également être un suffixe de $P[1 \dots i]$ pour pouvoir coïncider avec le texte. En fait, il faut décaler P vers la droite d'exactement $i - s(i)$ caractères vers la droite, et on s'est ramené au problème de l'alignement de P sous $T[k+1+i-s(i) ... k+m+i-s(i)]$, sachant que les $s(i)$ premiers caractères concordent avec ceux de P.

Il est facile de voir que l'algorithme est correct, c'est-à-dire qu'il essaie tous les alignements où P a une chance de concorder avec T.

Sa complexité en temps se voit en analysant le nombre de comparaisons entre motif et texte.

Quand un caractère du texte est comparé avec un caractère du motif, et qu'il y a égalité, cette information est mémorisée et utilisée par l'algorithme, qui ne regardera plus jamais la valeur du texte à cette position. Il y a donc au plus n comparaisons couronnées de succès.

Toute comparaisons aboutissant à un échec résulte en un décalage du motif d'au moins une position vers la droite. Le motif peut être décalé au plus n fois sous le texte. Il y donc au plus n telles comparaisons.

Au total, le nombre de comparaisons est donc d'au plus $2n$.

L'algorithme utilise un précalcul de la fonction $i \mapsto s(i)$. Cette fonction dépend uniquement du motif. Elle peut se calculer en temps linéaire $O(m)$. L'algorithme a été présenté en Majeure 1.

Alignement local. Etant donné deux chaînes de caractères S_1 et S_2 de longueurs m et n respectivement, ainsi qu'un fonction f de similarité entre les caractères (plus f est grande, plus les caractères se ressemblent), il s'agit de trouver un facteur α de S_1 et un facteur β de S_2 qui ont un alignement de valeur totale maximale.

Ce problème peut être résolu en $O(mn)$ par programmation dynamique, en généralisant l'algorithme d'alignement global de S_1 et S_2 vu au chapitre 2.

4. Sous-séquences maximales

Motivation. Tout l'ADN ne contient pas d'informations sur les gènes. Une région de codage est un facteur du mot qui représente un gène. Entre deux régions de codage, se trouvent des zones intermédiaires. Typiquement, 90% du génome d'un organisme prokariote (mono-cellulaire sans noyau) est “utile" (codant des gènes). En revanche, le mécanisme de transcription des organismes eukariotes (tels que les plantes ou les animaux) utilise des introns, lesquels occupent une grande partie de l'ADN : chez l'être humain, les régions de codage de gènes occupent moins de 3% du génome.

Etant donné le génome, on souhaite identifier les facteurs qui sont probablement des régions de codage, afin d'identifier de nouveaux gènes.

Une étape préliminaire pour cela est d'identifier de longs facteurs qui ne contiennent pas de codon “STOP". Puis on regarde le mot codon par codon, en utilisant nos connaissances préalables au sujet de gènes déjà identifiés. Par exemple, si le codon qu'on vient de lire a fréquence 7% parmi les gènes déjà connus, mais seulement 4% dans l'ensemble du génome, on peut interpréter ce fait comme augmentant les chances que la région étudiée soit effectivement un gène. Ainsi, on peut supposer qu'on dispose d'une table attribuant un “score" à chaque codon ; plus le score est élevé, plus la présence du codon est significative et augmente les chances d'être effectivement en train de lire un gène.

Définition du problème. Etant donné un mot, avec pour chaque lettre un score, $X_1X_2 \dots X_n$, où X_i représente le score (positif ou négatif) de la i-ième lettre du mot, il s'agit de trouver un facteur du mot dont le score cumulé est maximum : c'est-à-dire un facteur $X_i \dots X_j$ qui maximise $\sum_{i \le k \le j} X_k$.

Algorithme. Ce problème se résoud par recherche naïve en temps $O(n^3)$, mais beaucoup plus rapidement par programmation dynamique.

Pour tout $k \in \{1, \dots, n\}$, soit a_k le meilleur score d'un facteur de $X_1 \dots X_k$, et b_k le meilleur score d'un suffixe de $X_1 \dots X_k$.

Initialement, pour $k = 0$, on a $a_k = b_k = 0$.

Les suites (a_k) et (b_k) peuvent être définies en utilisant la récurrence double ci-dessous.

$$\begin{aligned} a_k &= \max(a_{k-1}, b_{k-1} + X_k) \\ b_k &= \max(b_{k-1} + X_k, 0) \end{aligned}$$

En implémentant cet algorithme avec des tableaux pour a et pour b, il est aisé de voir que a_n se calcule en temps $O(n)$.

Pour avoir, non seulement la valeur du meilleur facteur candidat à être un gène, mais également sa position, il suffit de garder en outre la trace de la position du facteur considéré dans des variables auxiliaires :

```
a[0]<- 0 ; b[0] <-0 ; debutb <- 1 ;
pour k allant de 1 a n faire
   si b[k-1]+X[k]>a[k-1]
   alors
     a[k] <- b[k-1]+X[k];
     debuta <- debutb ;
     fina <- k
   finsi
   si b[k-1]+X[k]<0
   alors
     b[k] <- 0 ;
     debutb <- k+1
   sinon
     b[k] <- b[k-1]+X[k]
   finsi
finfaire
resultat: facteur X[debuta...fina] de valeur a[n].
```

Variantes. En fait, on souhaiterait trouver en temps linéaire, non pas simplement un seul candidat, qui serait la région qui a la plus forte probabilité de coder un gène, mais carrément un ensemble de facteurs disjoints qui soient des candidats probables.

Résoudre ce problème pour $X_1 \dots X_n$, c'est d'une part trouver un facteur $X_i \dots X_j$ de score maximum, d'autre part résoudre le problème pour $X_1 \dots X_{i-1}$ ainsi que pour $X_{j+1} \dots X_n$. Ceci définit récursivement le problème à résoudre.

Par utilisation d'une structure de données chaînée bien conçue, il est possible de résoudre ce problème également en temps linéaire.

5. Reconstruction d'arbres d'évolution

Motivation. Supposons que l'on connaisse le génome d'un ensemble d'espèces actuellement vivantes. Supposons de plus qu'on sache mesurer la similarité entre ces génomes. Les génomes d'espèces proches l'une de l'autre dans l'arbre d'évolution de la vie doivent être similaires, et on devrait donc pouvoir en déduire l'arbre d'évolution le plus probable.

Définition du problème. Etant donné n espèces, avec pour chaque paire d'espèce $\{i, j\}$, une distance $D(i, j)$ donnant le temps écoulé depuis que ces espèces ont divergé, reconstruire l'arbre d'évolution de ces espèces : les n espèces sont les n feuilles de l'arbre, chaque noeud interne a au moins deux enfants et porte une étiquette représentant le temps écoulé depuis que cette espèce a muté : pour toute paire de feuilles $\{i, j\}$, $D(i, j)$ est l'étiquette du plus proche ancêtre commun à i et à j dans l'arbre ; et sur tout chemin de la racine à une feuille, les étiquettes sont strictement décroissantes.

DÉFINITION 8.1. *Une matrice (n, n) définit une ultra-métrique si pour tout triplet $\{i, j, k\}$, le maximum de $D(i, j), D(j, k)$ et $D(k, i)$ est atteint par au moins deux de ces trois valeurs.*

THÉORÈME 8.2. *Etant donné n espèces avec leur matrice D symétrique associée, le probléme de reconstruction de l'arbre d'évolution a une solution si et seulement si D définit une ultra-métrique.*

S'il existe un arbre d'évolution, et qu'on en efface tout ce qui n'est pas sur un plus court chemin entre i et j, j et k, ou k et i, on obtient un arbre à 3 feuilles i, j, k. Si sa racine u a trois enfants, alors c'est le plus proche ancêtre commun et $D(i, j) = D(j, k) = D(k, i)$ est l'étiquette de u. Sinon, u a deux enfants ; par exemple, un de ses deux sous-arbres contient à la fois i et j, et l'autre doit alors contenir k : on a alors $D(i, k) = D(j, k)$. Ceci démontre la direction "seulement si" du théorème.

Pour la réciproque, la preuve est constructive. Soit i une espèce. Soit d le nombre de valeurs distinctes parmi $\{D(i, j), j \neq i\}$. S'il existe un arbre d'évolution associé à D, alors l'arbre doit avoir un chemin de longueur d de la racine à i, et les valeurs de $\{D(i, j\}$ partitionnent les $n - 1$ autre feuilles en parties pour chacune desquelles on pourra résoudre récursivement le problème. On combine finalement les solutions de ces problèmes en recollant ces arbres le long du chemin. Il est facile de voir que si D est une ultramétrique, alors l'arbre ainsi construit satisfera les propriétés requises.

De ce théorème, on déduit l'unicité de l'arbre d'évolution ainsi qu'un algorithme en $O(n^2)$ pour reconstruction d'arbre d'évolution.

6. Estimation de distance entre génomes

La section précédente suppose connue une mesure de similarité entre génomes calculant le temps écoulé depuis que les espèces ont divergé. Mais comment faire cette mesure ? Comment calculer la distance entre deux génomes en terme d'évolution ?

Supposons connue l'écriture du génome comme suite de gènes, ici supposée circulaire pour simplifier les détails techniques. Des espèces proches s'avèrent avoir les mêmes gènes, mais dans un ordre différent le long du génome. Lors d'une mutation, l'une des principales modifications observées consiste a prendre un morceau du génome et à le renverser : $g_1 g_2 \cdots g_i \cdots g_j \cdots g_n$ devient $g_1 \ldots g_{i-1} g_j g_{j-1} \cdots g_i g_{j+1} \cdots g_n$.

Soit $\rho(i, j) = \rho_\ell$ la probabilité d'une telle mutation lorsque la longueur $j - i + 1$ du segment renversé est égale à ℓ : puisqu'il y a n segments de longueur ℓ, on a $\sum_\ell n\rho_\ell = 1$.

Ceci définit un modèle simplifié d'évolution stochastique. Notons qu'il est réversible : la probabilité de la transition $\sigma \to \sigma'$ est égale à la probabilité de la transition $\sigma' \to \sigma$.

Si le plus proche ancêtre commun aux deux génomes σ et τ se trouve à k générations de distance, alors, par réversibilité, τ peut être engendré à partir de σ en $2k$ générations.

On est donc ramené au problème suivant : étant donné $\sigma = \sigma^{(0)}$, on suit le modèle d'évolution stochastique défini plus haut. Quelle est la valeur de $2k$ la plus probable, nombre de générations pour avoir τ ?

L'approche statistique suivante se révèle efficace. Soit $\sigma = g_1 \dots g_n$. i forme une coupe si g_i et g_{i+1} ne sont pas voisins dans τ. Soit D le nombre de i formant une coupe.

Soit D_t le nombre de coupes entre $\sigma(t)$ et $\sigma^{(0)}$, où $\sigma^{(t)}$ est le descendant de $\sigma^{(0)}$ après t mutations.

(1) On calcule $E(D_t)$ pour $t = 1, 2, 3, \dots$ en procédant par récurrence.

(2) On choisit le t tel que $|E(D_t) - D|$ soit minimum.

CHAPITRE 9

Algorithmes pour la physique statistique

Le calcul de fonctions de partition ou de valeurs critiques en physique statistique amène souvent à effectuer de longs calculs. Il est donc important d'étudier la complexité des algorithmes mis en œuvre pour ces calculs. Le modèle d'Ising est l'un des modèles de base. Le "résoudre" signifie calculer sa fonction de partition : cela n'a pu être fait explicitement de manière rigoureuse que pour des graphes très particuliers, et il reste encore à faire pour venir complètement à bout du calcul ; il en est ainsi pour les réseaux à 3 dimensions. Dans ce qui suit on va montrer que plusieurs des problèmes posés par les calculs de la physique statistique sont **NP**-complets, en particulier les calculs liés au modèle d'Ising. Une exception notable est celle du calcul de la configuration d'énergie optimale dans le problème du verre de spin lorsque le graphe est planaire. On étudiera des algorithmes d'approximation de type Monte-Carlo. Enfin on verra que si la physique a fourni des problèmes très intéressants à l'algorithmique elle lui a aussi apporté des heuristiques efficaces, en effet la méthode dite du *recuit simulé* donne dans de nombreux cas des solutions approchées à des problèmes d'optimisation combinatoire.

1. Modèle d'Ising

Ce modèle tente d'expliquer la physique des transitions de phase, il est appelé ainsi après les travaux réalisés dans les années 20 par Ernst Ising. Les transitions de phase sont des changements qualitatifs macroscopiques qui se produisent dans certains états d'un système lorsque l'on modifie légèrement les paramètres (conditions de température ou de pression par exemple), et ne sont pas toujours bien expliqués. Le modèle d'Ising décrit les interactions entre molécules dans un cristal et permet de calculer certains paramètres de changement de phase, comme la température critique par exemple. Il s'agit ici de lier ces problèmes à l'algorithmique de graphes.

On considère un graphe non orienté $G = (X, E)$ comportant n sommets et m arêtes. Dan le contexte physique, le graphe G est souvent très régulier : grille en petite dimension ou au contraire graphe aléatoire.

Par exemple dans le modèle de grille à une dimension tous les sommets du graphe sauf 2 ont deux voisins, les deux sommets restant n'en ayant qu'un seul.

FIG. 1. Un réseau à une dimension

Dans le réseau à deux dimensions le graphe considéré est une grille :

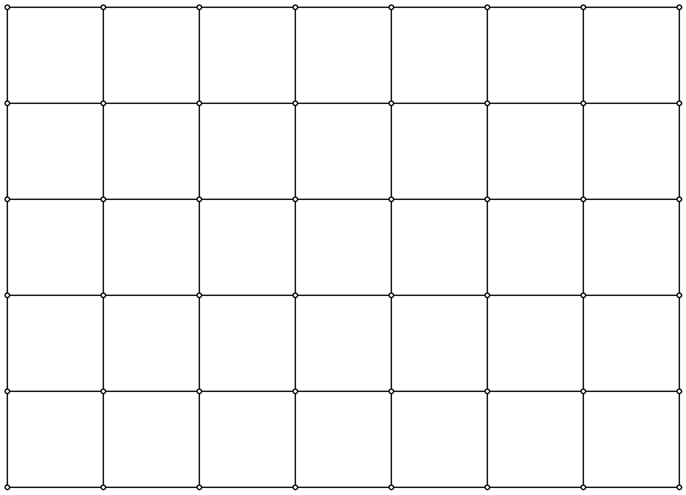

FIG. 2. Un réseau à deux dimensions

A chaque arête $e \in E$ est associée une énergie d'interaction réelle $J(e)$ qui est souvent considérée comme constante ($\forall e, \quad J(e) = J$) mais qui dans certains cas peut dépendre de e. Une *configuration* σ du graphe associe à chaque sommet un *spin*, c'est-à-dire un élément de l'ensemble $\{-1, 1\}$. On définit l'énergie de la configuration σ par

$$H(\sigma) = \sum_{e=\{x,y\} \in E} -J(e)\sigma(x)\sigma(y) - M \sum_{x \in X} \sigma(x),$$

où M est une énergie due à un champ magnétique extérieur.

Un des problèmes principaux consiste à déterminer la *fonction de partition* Z définie par :

$$Z = \sum_{\sigma} e^{-\beta H(\sigma)}$$

où la somme est prise sur les 2^n configurations possibles et où $\beta = 1/(kT)$, T étant et k la constante k. En effet, la physique statistique postule que les configurations de spins sont réparties selon la distribution de Gibbs, selon laquelle la probabilité d'une configuration σ donnée est égale à

$$P(\sigma) = \frac{e^{-\beta H(\sigma)}}{Z}.$$

Sur cette formule on constate que que lorsque la température tend vers l'infini, β tend vers 0, et la distribution devient uniforme, sans plus de corrélation entre spins voisins (état désordonné), alors que lorsque la température tend vers 0, β tend vers l'infini, les différences d'énergies sont ainsi magnifiées et la distribution se concentre sur l'état ou les états d'énergie minimale.

Fonction de partition et polynôme des coupures. Rappelons qu'une coupure de G est donnée par une partition de X en deux parties X_1 et X_2, et est constituée de toutes les arêtes qui ont une extrémité dans X_1 et l'autre dans X_2. Soit b_j le nombre de coupures à j arêtes, et $B(x)$ le polynôme

$$B(x) = \sum_{j=0}^{m} b_j x^j.$$

On remarque que chaque sous-ensemble X_1 de X donne deux coupures de même taille ainsi $B(1) = 2^{n-1}$ où n est le nombre de sommets du graphe.

THÉORÈME 9.1. *Soit G un graphe à m arêtes où l'énergie d'interaction J est constante et où le champ magnétique M est nul. Alors la fonction de partition Z satisfait :*

$$Z = \frac{2}{\theta^m} B(\theta^2),$$

où $\theta = e^{-\beta J}$ et B est le polynôme des coupures de G.

Preuve : Soit σ une configuration. Soit X_1 l'ensemble des sommets de spin 1 et X_2 l'ensemble des sommets de spin -1. Soient m_i le nombre d'arêtes qui joignent les sommets de X_i entre eux, et $c(X_1, X_2)$ le nombre d'arêtes de la coupure définie par (X_1, X_2). Le produit des spins vaut $+1$ pour les arêtes internes aux X_i et -1 pour les arêtes de la coupure. Ainsi :

$$\begin{cases} H(\sigma) & = & -J(m_1 + m_2 - c(X_1, X_2)) \\ m & = & m_1 + m_2 + c(X_1, X_2) \end{cases}$$

On obtient

$$H(\sigma) = -mJ + 2Jc(X_1, X_2)$$

et pour la fonction de partition :

$$Z = \sum_{\sigma} e^{\beta m J} e^{-2\beta c(X_1, X_2)}$$

Or chaque décomposition de X en $X_1 \cup X_2$ donne deux configurations de même énergie (en échangeant les spins $+1$ et -1), ainsi :

$$Z = 2e^{\beta m J} \sum_{j=0}^{m} b_j e^{2j\beta J}$$

et en posant $\theta = e^{-\beta J}$ on a :

$$Z = \frac{2}{\theta^m} \sum_{j=0}^{m} b_j \theta^{2j} = \frac{1}{\theta^m} B(\theta^2).$$

□

COROLLAIRE 9.1. *Le calcul de la fonction de partition est de complexité équivalente à la détermination du polynôme des coupures B du graphe G.*

Preuve : Z se calcule directement à partir de B ; réciproquement, si on connaît la valeur de Z pour m valeurs de β, une formule d'interpolation classique permet d'obtenir les coefficients de B. □

Un préliminaire au calcul de la fonction de partition est de connaître sa valeur lorsque la température tend vers 0. Elle est alors déterminée par la configuration d'énergie minimale (état de base du système), qui est aussi, à toute température, l'état le plus probable selon la distribution de Gibbs. Même ce problème très préliminaire pose déjà des questions algorithmiques intéressantes.

Prenons la situation où il n'y a pas de champ magnétique extérieur ($M = 0$) et où les interactions sont constantes.

Si $J < 0$, on est dans le cas anti-ferromagnétique (les spins de sommets voisins tendent à s'opposer l'un à l'autre), et l'état d'énergie minimale correspond à la coupure contenant le plus d'arêtes : il s'agit du problème de la coupure maximale, dont on a vu qu'il était NP-difficile.

Si $J > 0$, on est dans le cas ferro-magnétique (les spins de sommets voisins tendent à s'aligner l'un à l'autre), et (pour G graphe connexe) il y a exactement deux états d'énergie minimale, l'un où tous les spins sont égaux à $+1$, et l'autre où tous les spins sont égaux à -1. Expérimentalement, on observe effectivement qu'à haute température, la somme des spins est proche de 0, mais qu'à faible température, la somme des spins est, ou bien proche de $+n$, ou bien proche de $-n$, ces deux situations ayant égale probabilité : on parle alors de magnétisation spontanée puisqu'elle survient même en l'absence de champ magnétique extérieur.

Lorsque les interactions ne sont pas constantes, le problème est plus difficile mais peut dans certains cas de graphes être résolu de manière satisfaisante.

2. Modèle verre de spin et calcul de l'état d'énergie minimale

Dans le cas des verres de spin le cristal est un alliage contenant différents composants ainsi les différentes arêtes e du graphe ont des énergies d'interaction $J(e)$ diverses. Il s'agit de trouver la configuration pour laquelle $H(\sigma)$ est minimale.

ETAT D'ÉNERGIE MINIMALE.

Données : un graphe non-orienté $G = (X, E)$, et une force d'interaction $J(e)$ sur chaque arête $e \in E$.

Résultat : Un spin $\sigma(x) \in \{-1,+1\}$ pour chaque sommet $x \in X$, qui minimise $-\sum_{e=\{x,y\}\in E} J(e)\sigma(x)\sigma(y)$.

Par exemple si l'on considère le graphe constitué de 6 sommets et 7 arêtes formant deux carrés voisins (voir Figure), dans lequel $J(e) = 1$ sauf $J\{1,2\} = J\{5,6\} = -3$. Une solution optimale avec $H(\sigma) = -9$ est donnée par :

$$\sigma(1) = \sigma(3) = \sigma(5) = +1 \qquad \sigma(2) = \sigma(4) = \sigma(6) = -1$$

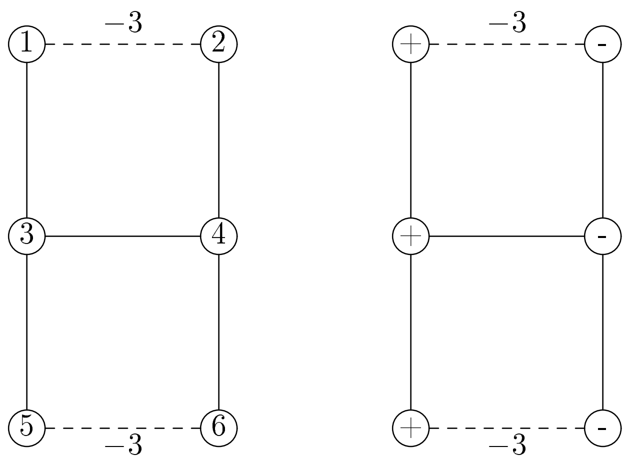

FIG. 3. Configuration d'énergie optimale

Dans le cas général le problème est NP-complet puisqu'il généralise le problème de la coupure maximale, et il est donc illusoire de chercher un algorithme qui donne rapidement une solution exacte. Cependant, dans le cas particulier où le graphe est planaire, nous allons voir qu'on a un algorithme polynomial.
Lorsqu'il existe une configuration σ_0 telle que $\sigma_0(x)\sigma_0(y)J(e)$ est positive pour toute arête e du graphe on dit que le cas est équivalent au modèle ferro-magnétique. Il est relativement facile de déceler ce type de situation. Une arête est dite négative si $J(e) < 0$.

:

PROPOSITION 9.1. *Le problème de l'état d'énergie minimale admet une solution telle que* $H(\sigma) = -\sum_{e\in E} |J(e)|$ *si et seulement si tout cycle de* G *contient un nombre pair d'arêtes négatives.*

Preuve : On construit le graphe G' obtenu en ajoutant un nouveau sommet de degré 2 sur chacune des arêtes e de G telles que $J(e) > 0$. On vérifie que (G, J) est équivalent au modèle ferromagnétique si et seulement si les sommets de G' peuvent être partagés en deux classes X_1 et X_2 telles que toute arête de G' ait une extrémité dans X_1 et une autre dans X_2. Ceci signifie que le graphe G' est ainsi 2-coloriable. Une condition nécessaire et suffisante pour qu'un graphe soit 2-coloriable est que tous ses cycles soient de longueur paire. La preuve se termine en remarquant que G' a tous ses cycles de longueur paire si et seulement si tout cycle de G contient un nombre pair d'arêtes négatives. $\square$

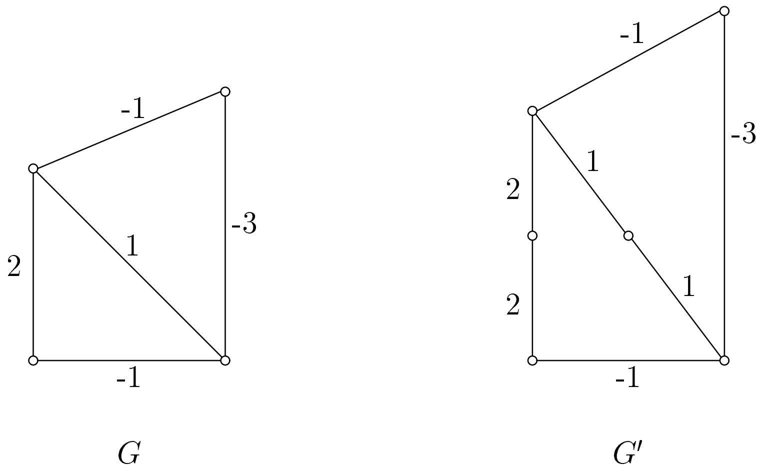

FIG. 4. Un graphe valué et le graphe à bicolorier associé

La proposition qui précède donne un critère théorique difficile à vérifier. Par contre il est assez facile de vérifier qu'un graphe est équivalent au modèle ferromagnétique en utilisant l'algorithme glouton suivant :

- On attribue la valeur $\sigma(x_0) = +1$ à l'un quelconque des sommets x_0.
- on calcule de proche en proche la valeur des $\sigma(x)$, en utilisant la règle :

  Si $e = \{x, y\}$ et $\sigma(y)$ est connu alors que $\sigma(x)$ ne l'est pas alors on choisit $\sigma(x)$ de façon à avoir :

$$\sigma(x)\sigma(y)J(e) \geq 0$$

- Une fois que l'on a déterminé les valeurs de $\sigma(x)$ pour tous les sommets du graphe on vérifie que la condition $\sigma(x)\sigma(y)J(e) \geq 0$ est satisfaite pour toutes les arêtes.

Algorithme dans le cas planaire.

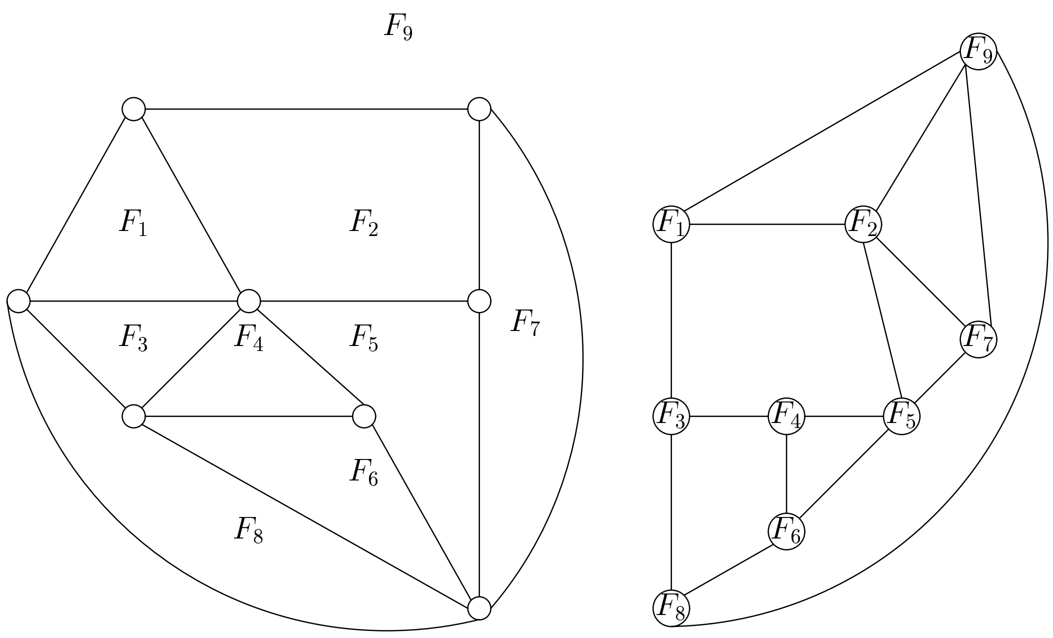

FIG. 5. Un graphe planaire et son dual

Les arêtes d'un graphe planaire connexe délimitent des domaines simplement connexes : *les faces*. Une de ces faces est non bornée on l'appelle la *face infinie*. A chacune de ces faces on associe le cycle formé des arêtes qui la séparent d'une autre face. Ces cycles jouent un rôle important, car tout cycle d'un graphe planaire peut s'obtenir comme différence symétrique d'une famille de cycles associés aux faces. Ainsi la proposition donnée plus haut se simplifie pour les graphes planaires par :

PROPOSITION 9.2. *Un graphe valué $G = (X, E, J)$ planaire est équivalent au modèle ferromagnétique si et seulement si toute face de G contient un nombre pair d'arêtes négatives.*

Une face qui contient un nombre impair d'arêtes négatives est dite *frustrée*. Un graphe associé à ces faces va permettre de calculer une configuration d'énergie minimale.

Pour toute configuration σ on dit qu'une arête $e = \{x, y\}$ est dite *contrariée* si $J(e)\sigma(x)\sigma(y) < 0$. Il est clair que l'énergie d'une configuration est déterminée par l'ensemble des arêtes contrariées. Rechercher une configuration d'énergie minimale revient à rechercher un ensemble d'arêtes contrariées dont la somme des valeurs absolues des valuations soit minimale. L'ensemble des arêtes contrariées associées à une configuration σ est noté C_σ.

PROPOSITION 9.3. *Soit $G = (X, E, J)$ un graphe valué planaire et U un sous-ensemble d'arêtes de E, U forme un ensemble d'arêtes contrariées pour une configuration σ si et seulement si :*

(1) *Toute face frustrée contient un nombre impair d'arêtes dans U.*
(2) *Toute face non frustrée contient un nombre pair d'arêtes dans U.*

Preuve : Le fait qu'un ensemble d'arêtes contrariées vérifient 1. et 2. est immédiat. Considérons la réciproque : soit J' défini par :

$J'(e) = -J(e)$ si e appartient à U, $J'(e) = J(e)$ sinon.

Alors J' défini une valuation dans laquelle aucune face n'est frustrée, il existe alors une configuration de spin σ pour laquelle aucune arête n'est contrariée pour J'. Dans cette configuration les arêtes contrariées pour J sont exactement les arêtes de U. □

On associe à un graphe planaire son graphe dual G^* construit de la façon suivante : les sommets de G^* sont les faces de G ,(noter que la face infinie fait partie des faces de G). Deux sommets sont reliés dans G^* par une arête si et seulement si ils correspondent à deux faces de G adjacentes, c'est à dire possédant une arête de G en commun sur leurs frontières.

Les arêtes du graphe dual G^* de G correspondent aux arêtes de G, les faces frustrées dans G donnent lieu à des sommets frustrés dans G^*. La proposition précédente se traduit dans G^* par.

PROPOSITION 9.4. *Un ensemble U d'arêtes forme un ensemble d'arêtes contrariées pour le graphe valué planaire $G = (X, E, J)$ si et seulement si :*

(1) *Tout sommet frustré de G^* est incident à un nombre impair d'arêtes de U.*
(2) *Tout sommet non frustré de G^* est incident à un nombre pair d'arêtes de U.*

Comme conséquence immédiate on obtient que le nombre de sommets frustrés de G^* (et donc que le nombre de faces frustrées de G) est pair. On a aussi :

PROPOSITION 9.5. *Si σ est une configuration d'énergie minimale alors l'ensemble des arêtes contrariées forme un sous graphe sans cycle de G^*.*

Preuve : S'il existe un cycle dans U on considère l'ensemble U' obtenu en supprimant dans U les arêtes du cycle, U' vérifie les conditions de la proposition car la suppression d'un cycle ne modifie pas la parité des degrés des sommets, et la configuration définie par U' est d'énergie plus faible. □

Au graphe G on associe le graphe $\hat{G}$ construit de la façon suivante :
- L'ensemble des sommets de $\hat{G}$ est constitué des sommets frustrés de G^*
- L'ensemble des arêtes de $\hat{G}$ est constitué de toutes les paires $\{x, y\}$ de sommets de $\hat{G}$.
- Chaque arête $\{x, y\}$ est valuée par la longueur du plus court chemin qui joint x à y dans G^*, où la longueur d'un chemin est la somme des valeurs absolues $|J(e_i)|$ des arêtes e_i qui en font partie.

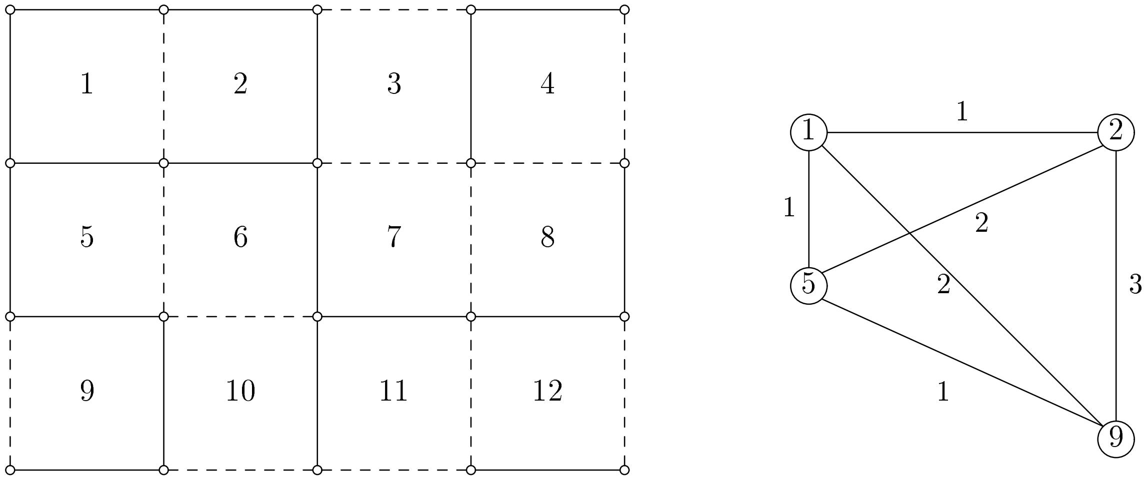

FIG. 6. Un graphe planaire valué G et le graphe $\hat{G}$

Un sous ensemble C de l'ensemble des arêtes d'un graphe est un *couplage parfait*, si deux arêtes de C n'ont pas d'extrémité commune et si tout sommet de G est extrémité d'une arête de C. La *valuation* d'un couplage parfait est la somme des valuations des arêtes.

Un couplage parfait C de valuation minimale dans $\hat{G}$ détermine un ensemble d'arêtes U_C de G^*, en effet à chaque arête de C est associé un chemin dans G^* et U_C et constitué de l'union de tous les chemins ainsi formés.

THÉORÈME 9.2. *Un couplage parfait C de valuation minimale dans $\hat{G}$ détermine un ensemble d'arêtes U_C qui forme un ensemble d'arêtes contrariées pour une configuration σ de G, cette configuration est d'énergie minimale.*

Preuve : Il est assez facile de voir que l'on peut décomposer les arêtes d'un graphe sans cycle en une famille de chemins disjoints qui relient deux à deux les sommets de degré impairs. Ces chemins déterminent donc un couplage dans le graphe des faces frustrées. Si le couplage minimal est de valuation plus faible, celui-ci détermine des chaînes reliant des sommets frustrés de G^*, il est facile de voir que ces chaînes n'ont pas d'arêtes en commun (sinon on pourrait améliorer le couplage) en choisissant l'ensemble des arêtes de ces chaînes comme arêtes contrariées on obtient une configuration d'énergie inférieure.

Conséquence Déterminer une configuration d'énergie minimale revient à déterminer un couplage minimal dans le graphe $\hat{G}$ il existe des algorithmes permettant d'effectuer ceci en temps polynomial ; ils ont été évoqués dans le chapitre 3 (Flots et Couplages).

Sur la figure ci-dessus on a représenté un graphe régulier à 20 sommets, les interactions sont telles que $J(e) = 1$ (traits normaux) ou $J(e) = -1$ (traits en pointillés).

3. Calcul de la fonction de partition dans le cas planaire

Une avancée majeure dans l'étude du modèle d'Ising, due à Kasteleyn, Fisher et Temperley dans les années 60, résoud le modèle dans le cas où le graphe est planaire, en ramenant le problème à l'évaluation d'un certain déterminant.

Une transformation locale du graphe réduit le problème à l'étude de coupages parfait sur un autre graphe planaire G', plus précisément, au calcul de la somme, sur tous les couplages parfaits de G', du poids du couplage ; le poids d'un couplage est le produit des poids de ses arêtes.

Prenons un cas particulier simple où la démonstration contient déjà des idées essentielles. Comment calculer le poids des couplages parfaits de la grille de côté n ?

La grille étant un graphe biparti, colorions ses sommets en noir et blanc, et considérons un couplage parfait comme une bijection π de l'ensemble des N sommets noirs vers l'ensemble des N sommets blancs. Le poids total des couplages parfaits est alors :

$$X = \sum_{\tau} w_{1\tau(1)} w_{2\tau(2)} \dots w_{N\tau(n)}.$$

Considérons maintenant la matrice A, matrice $N \times N$ dont l'entrée $\{i, j\}$ vaut 0 si le sommet noir i n'est pas adjacent au sommet blanc j, et a pour norme w s'ils sont reliés par une arête de poids w. On pose alors $a_{i,j} = w$ si l'arête $\{i, j\}$ de la grille est verticale, et $a_{i,j} = i$, la racine carrée de -1, si l'arête est horizontale.

La propriété essentielle de cette matrice (celle qui l'on peu généraliser lrosqu'on considère un graphe planaire biparti quelconque, au lieu de la grille), est que le produit des a_{ij} autour de n'importe quelle face est égal à -1 fois le produit des poids w_{ij} correspondants.

Par définition, le déterminant de A vaut :

$$det(A) = \sum_{\tau} \epsilon(\tau) a_{1\tau(1)} a_{2\tau(2)} \dots a_{N\tau(n)}.$$

On va montrer par récurrence que

$$\frac{\epsilon(\tau) a_{1\tau(1)} a_{2\tau(2)} \dots a_{N\tau(n)}}{w_{1\tau(1)} w_{2\tau(2)} \dots w_{N\tau(n)}}$$

est constant, indépendamment de τ, d'où on déduit que la norme du déterminant de A est la valeur X recherchée.

Considérons deux permutations quelconques τ_1 et τ_2. Chacune correspond à un ensemble d'arêtes de la grille. Leur superposition définit des cycles. Parmi ces cycles, considérons un cycle C qui soit "le plus interne" possible dans ce graphe planaire, c'est-à-dire que les sommets à l'intérieur de ce cycle ont tous la même image par τ_1 et par τ_2. Définissons la "distance" entre τ_1 et τ_2 comme la somme des surfaces des cycles les plus internes, où la surface d'un cycle est le nombre total de sommets strictement à l'intérieur du cycle.

On considère une face de la grille qui soit à l'intérieur de C et qui touche C, et laissée globalement invariante par τ_1 ou par τ_2. (Il n'est pas difficile de montrer qu'il en existe nécessairement une). San perte de généralité, supposons que ce soit τ_1. Appliquons alors à τ_1 la transposition qui prend les deux sommets noirs sur ce cycle et qui échange leurs images. On obtient τ_1', et par récurrence on peut appliquer l'hypothèse à (τ_1', τ_2) .

Il ne reste plus qu'à montrer que

$$\frac{\epsilon(\tau_1)a_{1\tau_1(1)}a_{2\tau_1(2)}\dots a_{N\tau_1(n)}}{w_{1\tau_1(1)}w_{2\tau_1(2)}\dots w_{N\tau_1(n)}} = \frac{\epsilon(\tau_1')a_{1\tau_1'(1)}a_{2\tau_1'(2)}\dots a_{N\tau_1'(n)}}{w_{1\tau_1'(1)}w_{2\tau_1'(2)}\dots w_{N\tau_1'(n)}}.$$

Ces deux expressions ne diffèrent que sur la face utilisée. Puisqu'on a appliqué une transposition, $\epsilon(\tau_1') = -\epsilon(\tau_1)$. Quand aux rapports des coefficients aux poids, autour de la face il vaut 1 pour τ_1 et $i^2 = -1$ pour τ_1', ou l'inverse (selon que c'est τ_1 ou τ_1' qui utilise les deux arêtes verticales du carré). Dans les deux cas, on a bien $(-1)\times 1 = i^2$ ou $(-1)\times i^2 = 1$, ce qui termine la démonstration.

Ainsi, grâce à un calcul de déterminant, on peut évaluer le poids total des couplages parfaits sur une grille carrée dont les arêtes ont un poids quelconque.

Cette approche se généralise à n'importe quel graphe planaire biparti : il suffit de choisir les poids des arêtes autour des faces de façon que le produit des poids autour de n'importe quelle face soit égal à ± 1, selon le signe de la permutation élémentaire associée.

Dans le cas de graphes planaires non bipartis, une généralisation plus technique résoud le problème en exprimant le poids total des couplages parfaits comme une somme de quatre déterminants.

4. Calcul de la fonction de partition et méthode de Monte Carlo

Au delà du cas planaire, les choses se compliquent, et en trois dimensions les solutions sont généralement numériques. Une approche populaire consiste à faire une marche aléatoire dans l'espace des configurations, où une étape consiste à choisir un sommet au hasard et à modifier son spin avec une certaine probabilité, choisie de façon que la distribution stationnaire de la chaîne de Markov ainsi définie soit bien la distribution de Gibbs (On parle alors d'un échantillonneur de Gibbs ou d'une dynamique de Glauber). On simule alors cette marche aléatoire, en s'arrêtant au bout d'un temps suffisamment long pour considérer que la distribution à ce stade est proche de la distribution stationnaire, et donc que la configuration obtenue à cet instant est à peu près un échantillon de la distribution de Gibbs. Cela permet alors de calculer la fonction de partition. Un problème de cette approche est son manque de rigueur : en particulier, on ne peut de manière réaliste simuler la chaîne de Markov que pendant un temps polynomial en la taille du graphe d'entrée, alors que le nombre de configurations est exponentiel. Au moment où on s'arrête, on risque d'être encore loin de la distribution stationnaire, et donc d'avoir une mauvaise image de la distribution de Gibbs. Ce problème s'attaque en définissant la distance entre deux distributions, et en montrant, pour des chaînes de Markov bien choisies, que le temps de convergence vers la distribution stationnaire est polynomial.

Les outils de conception et d'analyse rigoureuse de tels algorithmes ont été développés avec une participation active des informaticiens depuis le début des années 90, mais demandent des connaissances de probabilités qui dépassent le cadre de ce cours.

5. Recuit simulé

Pour résoudre certains problèmes d'optimisation combinatoire, on procède par améliorations successives d'une configuration. Dans le contexte considéré ici cela se traduit par partir d'une configuration quelconque et tenter de l'améliorer en modifiant l'une de ses composantes.

On obtient, en réitérant ce procédé une configuration qui est meilleure que toutes ses "voisines", on dit que l'on a atteint un *optimum local*. Rien ne dit que cet optimum local soit l'optimum général ou même qu'il en soit proche. Et c'est là l'inconvénient de cette méthode.

Pour tenter de pallier cet inconvénient, on modifie l'algorithme en faisant intervenir une part de hasard. On part d'une configuration quelconque σ, on cherche une configuration voisine σ' si celle ci est telle que $H(\sigma') < H(\sigma)$ on remplace σ par σ' et on recommence le procédé. Si $H(\sigma') > H(\sigma)$, on tire au hasard un nombre λ, compris entre des valeurs à déterminer, et on remplace σ par σ' si $H(\sigma') > H(\sigma) - \lambda$ puis on répète le procédé. Tout l'art réside dans la détermination du λ, celui-ci peut d'ailleurs varier au cours de l'exécution de l'algorithme. L'algorithme se termine quand on est resté suffisamment longtemps sur la même configuration. Cette façon de procéder s'appelle la méthode du recuit simulé, elle est utilisée dans certains cas pour résoudre de façon approchée des problèmes d'optimisation combinatoire NP complets.