Matthieu Boyer
Planche A

Théorème de Compacité et Pavages du Plan

On considère un ensemble dénombrable de variables propositionnelles $\mathcal{X} = \{x_1, x_2, \dots\}$. Un ensemble de clauses $\mathcal{S}$ est dit finiment satisfiable si tout sous-ensemble fini de $\mathcal{S}$ est satisfiable.

1. Soit $\mathcal{S}$ finiment satisfiable et $x \in \mathcal{X}$. Montrer que soit $\mathcal{S} \cup \{x\}$ est finiment satisfiable, soit $\mathcal{S} \cup \{\neg x\}$ est finiment satisfiable.
2. On construit $(\mathcal{S}_n)_{n\in\mathbb{N}}$ par récurrence : $\mathcal{S}_0 = \mathcal{S}$, et $\mathcal{S}_{n+1} = \mathcal{S}_n \cup \{L_{n+1}\}$ où $L_{n+1} \in \{x_{n+1}, \neg x_{n+1}\}$ est choisi pour préserver la finie satisfaisabilité. Justifier que cette construction est toujours possible.
3. On définit l'interprétation $\mathcal{I}$ par : $\mathcal{I}(x_n) =$ VRAI si $x_n \in \mathcal{S}_n$, et $\mathcal{I}(x_n) =$ FAUX sinon. Montrer que $\mathcal{I}$ satisfait toutes les clauses de $\mathcal{S}$.

On dispose de tuiles de Wang (carrés colorés différemment sur chaque côté). Un pavage est valide si les couleurs des bords adjacents coïncident.

4. Soit $p_{i,j,t}$ la variable : "la case $(i, j) \in \mathbb{Z}^2$ contient la tuile $t \in \mathcal{T}$". Modéliser par des clauses :
   - Chaque case contient exactement une tuile.
   - Les contraintes de couleurs entre voisins (ex : droite de (i, j) = gauche de $(i + 1, j)$).
5. Montrer que si l'on peut paver n'importe quel carré fini $N \times N$, alors on peut paver le plan infini $\mathbb{Z}^2$.
6. Pour un carré $N \times N$ et k types de tuiles, combien y a-t-il de configurations possibles ? Pourquoi la méthode du *backtracking* (test progressif avec retour en arrière) est-elle préférable à un test exhaustif ?
7. **Cas 2-SAT :** Si chaque contrainte de voisinage ne lie que deux tuiles, on peut utiliser un graphe d'implication. Expliquez comment une contradiction de type $P \to \cdots \to \neg P$ et $\neg P \to \cdots \to P$ permet de conclure à l'impossibilité du pavage.
8. Pourquoi le théorème de compacité ne permet-il pas de créer un programme qui s'arrête toujours pour nous dire si un pavage infini existe ?

Solution : Théorème de Compacité et Pavages du Plan

1. Si les deux étaient insatisfiables, il existerait $A, B \subseteq \mathcal{S}$ finis tels que $A \cup \{x\}$ et $B \cup \{\neg x\}$ soient insatisfiables. Alors $A \cup B \subseteq \mathcal{S}$ serait insatisfiable, contradiction.
2. C'est l'application directe du résultat précédent à chaque étape n.
3. Toute clause $C \in \mathcal{S}$ est finie. Elle ne dépend que d'un nombre fini de variables $\{x_1, \dots, x_n\}$. Comme $\mathcal{S}_n$ est finiment satisfiable, l'attribution faite jusqu'à n satisfait C.
4. $\forall (i,j) : (\bigvee_{t \in \mathcal{T}} p_{i,j,t})$ et $\forall t \neq t' : (\neg p_{i,j,t} \vee \neg p_{i,j,t'})$.
Couleurs : $(\neg p_{i,j,t} \vee \neg p_{i+1,j,t'})$ si les couleurs ne correspondent pas.
5. Un sous-ensemble fini de clauses ne concerne qu'une zone finie du plan. Si tout carré fini est pavable, tout sous-ensemble fini de clauses est satisfiable. Par compacité, le plan entier l'est.
6. Il y a $k^{(N^2)}$ configurations. C'est exponentiel. Le *backtracking* évite d'explorer des milliards de solutions dès qu'un conflit est détecté localement (élagage de l'arbre de recherche).
7. Une règle "Si P est posée, alors Q est forcée" se note $P \to Q$. Si un chemin mène de P à $\neg P$, alors P est nécessairement fausse. Si le cycle existe dans les deux sens, la variable ne peut être ni vraie ni fausse : le pavage est impossible.
8. Le théorème dit que s'il n'y a pas de solution, on finira par trouver un N bloquant. Mais si le plan est pavable, on testera $N = 1, 2, 3 \dots$ à l'infini sans jamais savoir s'il existe un blocage beaucoup plus loin. On ne peut pas fixer de "borne de sécurité".

Matthieu Boyer
Planche B

CSP Orientés, Ordonnancement et Flots

On s'intéresse à un problème d'ordonnancement de n tâches. Chaque tâche i a une date de début $x_i \in \{0, 1, \dots, T\}$. Les contraintes sont orientées : elles imposent un ordre de succession ou un délai minimal entre deux tâches.

On considère des contraintes de la forme $x_j - x_i \geq d_{ij}$, signifiant que la tâche j doit commencer au moins d_{ij} unités de temps après le début de la tâche i. On représente cela par un graphe orienté de contraintes où l'arc $i \to j$ est pondéré par d_{ij}.

1. Soient trois tâches A, B, C avec les contraintes : $x_B - x_A \geq 2$, $x_C - x_B \geq 2$ et $x_A - x_C \geq -3$.
   (a) Dessinez le graphe orienté associé.
   (b) Calculez la somme des poids le long du cycle $A \to B \to C \to A$. Déduisez-en l'absence de solution.
2. Montrez que si le graphe est un DAG, il existe toujours une solution.

On tente de résoudre le problème comme un problème de satisfaisabilité.

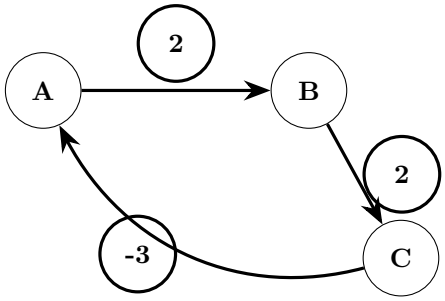
3. Étant donné un CSP, donnez une formule de la logique propositionnelle en forme normale conjonctive qui est satfiable si et seulement si le CSP l'est.
4. Quelle serait la complexité de résoudre SAT sur une telle formule en fonction de $n, t, \sum d_{i,j}$?

On complexifie le problème : chaque tâche doit être affectée à une machine. On dispose de m machines identiques. Plusieurs tâches peuvent commencer en même temps, mais le nombre total de tâches actives à un instant t ne peut dépasser m.

5. Proposez un algorithme plus efficace que réduire à SAT pour résoudre le problème.

Solution : CSP Orientés, Ordonnancement et Flots

1. **Analyse du système de contraintes :**
   (a) **Graphe orienté associé :** Les contraintes $x_j - x_i \geq d_{ij}$ sont représentées par des arcs $i \to j$ de poids d_{ij}.



   (b) La somme des poids le long du cycle $\mathcal{C} = A \to B \to C \to A$ est :

$$\sum_{(i,j)\in\mathcal{C}} d_{ij} = 2 + 2 + (-3) = 1$$

   En sommant les inégalités, on obtient : $0 \geq 1$. Cette contradiction montre que le problème est insatisfiable. Un CSP de ce type possède une solution si et seulement si son graphe de contraintes ne contient aucun cycle de poids strictement positif.

2. Si le graphe est un DAG, il ne contient aucun cycle, donc aucun cycle de poids positif. On peut construire une solution de manière gloutonne par programmation dynamique ou par un algorithme de plus long chemin :

$$x_j = \max\{0, \max_{(i,j)\in E}(x_i + d_{ij})\}$$

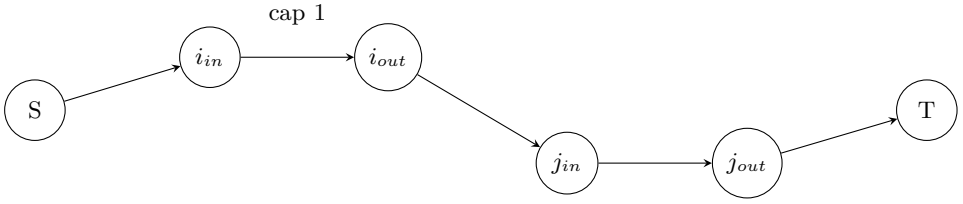
   En traitant les sommets selon un **ordre topologique**, chaque x_j est calculé à partir de valeurs déjà fixées, garantissant la satisfaction de toutes les contraintes.

3. On introduit des variables booléennes $b_{i,t}$ signifiant $x_i = t$ pour $t \in \{0, \ldots, T\}$. La formule est la conjonction des clauses suivantes :
   — $\forall i, (\bigvee_{t=0}^{T} b_{i,t}) \wedge \bigwedge_{t<t'} (\neg b_{i,t} \vee \neg b_{i,t'})$ (Chaque tâche commence à un instant unique).
   — $\forall (i,j) \in E, \forall t, t' < t + d_{ij}, (\neg b_{i,t} \vee \neg b_{j,t'})$ (Interdiction des débuts prématurés de j).

4. On utilise
   — $n(T+1)$ variables.
   — $O(m \cdot T^2)$ clauses, où m est le nombre de contraintes.

   Il y a donc un nombre gigantesque de contraintes.

5. Pour n tâches, on construit un réseau de flot biparti :
   — On scinde chaque tâche i en deux nuds i_{in} et i_{out} liés par un arc de capacité 1.
   — Un arc relie i_{out} à j_{in} si la tâche j peut succéder à i.



Au lieu de SAT, on utilise l'algorithme de Hopcroft-Karp pour calculer un couplage maximum dans le graphe biparti des compatibilités. Le nombre minimal de machines est $m = n - |Couplage\_Maximum|$. Cet algorithme polynomial ($O(m\sqrt{n})$) est optimal pour résoudre ce type de CSP structuré.

Matthieu Boyer
Planche C

Comptage de modèles

#SAT	
Entrée :	Une formule de la logique propositionnelle Φ
Sortie :	Le nombre d'interprétations la satisfaisant.

On considère un ensemble $\mathcal{X}$ de variables $\mathcal{X} = \{x_1, \dots, x_n\}$, et on note $\mathcal{B}(\mathcal{X})$ les formules de la logique propositionnelle

1. Rappelez le nombre d'interprétations de $\Phi \in \mathcal{B}(\mathcal{X})$ variables.
2. Soient Φ_1 et Φ_2 deux formules de $\mathcal{B}(\mathcal{X})$.
   (a) Sous quelles hypothèses a-t-on $\#(\Phi_1 \wedge \Phi_2) = \#\Phi_1 \times \#\Phi_2$?
   (b) Sous quelles hypothèses a-t-on $\#(\Phi_1 \vee \Phi_2) = \#\Phi_1 + \#\Phi_2$?
3. Montrez le théorème de décomposition de Shannon :

$$\#\Phi = \#(\Phi|_{x=0}) + \#(\Phi|_{x=1})$$

   Déduisez-en un algorithme pour résoudre le calcul.

On définit une forme normale décomposable et déterministe (d-DNNF) comme un graphe orienté acyclique dont les feuilles sont des littéraux ou des constantes et dont les noeuds internes sont des opérateurs logiques

— Les noeuds $\wedge$ sont décomposables : leurs fils portent sur des ensembles de variables disjoints.

— Les portes $\vee$ sont déterministes : au plus un fils peut être vrai pour une interprétation donnée.

4. Si Φ est compilée sous forme d-DNNF, donnez un algorithme qui calcule son nombre de modèles $\#\Phi$ en temps linéaire.

On généralise le problème en associant à chaque littéral l un poids $w(l) \in \mathbb{R}$. Le poids d'un modèle (interprétation satisfaisante) M est défini par le produit des poids de ses littéraux : $W(M) = \prod_{l \in M} w(l)$.

5. Comment adapter l'algorithme de parcours précédent pour calculer efficacement la somme des poids de tous les modèles, définie par $W(\Phi) = \sum_{M \models \Phi} W(M)$?
6. Soit Φ une base de connaissances encodant un domaine incertain. Expliquez comment le calcul d'une probabilité conditionnelle $P(\alpha \mid \beta) = \frac{P(\alpha \wedge \beta)}{P(\beta)}$ peut être résolu à l'aide de deux requêtes de comptage (#SAT) sur le circuit. Quelle hypothèse fait-on ici sur la distribution de probabilité des modèles ?

Solution : Comptage de modèles

1. 2^n, ce sont les fonctions $\mathcal{X} \to \{0,1\}$.
2. On a :
   (a) $\#(\Phi_1 \wedge \Phi_2) = \#\Phi_1 \times \#\Phi_2$ si les ensembles de variables apparaissant dans Φ_1 et Φ_2 sont **disjoints** ($\mathrm{Var}(\Phi_1) \cap \mathrm{Var}(\Phi_2) = \varnothing$). Dans ce cas, les choix satisfaisant Φ_1 sont indépendants de ceux satisfaisant Φ_2.
   (b) $\#(\Phi_1 \vee \Phi_2) = \#\Phi_1 + \#\Phi_2$ si les formules sont mutuellement exclusives (ou *incompatibles*), c'est-à-dire que $\Phi_1 \wedge \Phi_2 \equiv \bot$. L'ensemble des modèles de la disjonction est alors la réunion disjointe des modèles de chaque formule.
3. Soit $\mathcal{M}(\Phi)$ l'ensemble des modèles de Φ. On peut partitionner cet ensemble selon la valeur de la variable x :
   $$\mathcal{M}(\Phi) = \{M \in \mathcal{M}(\Phi) \mid M(x) = 0\} \cup \{M \in \mathcal{M}(\Phi) \mid M(x) = 1\}$$
   Par définition de la restriction, une interprétation M avec $M(x) = v$ satisfait Φ si et seulement si elle satisfait $\Phi|_{x=v}$. Comme les deux ensembles de la partition sont disjoints, le cardinal de leur union est la somme de leurs cardinaux, d'où :
   $$\#\Phi = \#(\Phi|_{x=0}) + \#(\Phi|_{x=1})$$
   Il s'agit d'un algorithme récursif (type DPLL sans arrêt précoce sur le premier modèle trouvé) :
   — Si Φ est une tautologie, retourner $2^{\text{variables restantes}}$. Si Φ est une contradiction, retourner 0.
   — Choisir une variable x, calculer récursivement le compte pour les deux branches et retourner la somme.
4. Grâce aux propriétés de décomposabilité (qui autorise le produit pour les nuds $\wedge$) et de déterminisme (qui autorise la somme pour les nuds $\vee$), on calcule $\#\Phi$ par une remontée post-ordre dans le DAG :
   — Si la feuille est un littéral l ou $\top$, sa valeur est 1. Si c'est $\bot$, sa valeur est 0.
   — Noeud $\wedge$: Retourner le produit des valeurs de ses fils.
   — Noeud $\vee$: Retourner la somme des valeurs de ses fils.

   Note : Si le circuit ne contient pas toutes les variables du domaine, il faut multiplier le résultat final par 2^{n-k} où k est le nombre de variables présentes dans le circuit.
5. L'algorithme reste structurellement identique, mais les valeurs aux feuilles et les opérations changent légèrement pour intégrer les poids $w(l)$:
   — Une feuille littérale l prend la valeur $w(l)$. Une constante $\top$ prend la valeur $w(x)+w(\neg x)$ pour chaque variable absente.
   — Noeud $\wedge$ (décomposable) : La valeur est le produit des poids de ses fils : $W(u \wedge v) = W(u) \times W(v)$.
   — Noeud $\vee$ (déterministe) : La valeur est la somme des poids de ses fils : $W(u \vee v) = W(u) + W(v)$.
6. En associant à chaque variable x un poids $w(x) = P(x)$ et $w(\neg x) = 1 - P(x)$, la somme des poids $W(\Phi)$ correspond exactement à la probabilité $P(\Phi)$. Ainsi, $P(\alpha \mid \beta) = \frac{W(\alpha \wedge \beta)}{W(\beta)}$. On résout cela en effectuant deux passes sur le circuit (ou une passe sur deux circuits différents).
   On suppose ici que les variables sont indépendantes (distribution produit). C'est cette hypothèse qui permet de justifier que le poids d'un modèle est le produit des poids de ses littéraux.