La programmation mathématique en nombres entiers

Problèmes classiques

Stanislas Francfort

Orange Labs

5 Juin 2017

Optimisation combinatoire et modélisations

- la formulation en PLNE permet donc de modéliser un grand nombre de problèmes combinatoires.
- une fois une bonne modélisation trouvée, solveur de PLNE est le seul outil nécessaire
- malheureusement, les seuls algorithmes connus pour résoudre les PLNE sont exponentiels
  - Branchement et évaluation (Branch and Bound)
- on peut l'enrichir par un algorithme de coupes.
- quand on n'a pas de formulation compacte, ou peut avoir :
  - un nombre exponentiel de variables : génération de colonnes (Branche and Price)
  - un nombre exponentiel de contraintes : génération de coupes (Branch and Cut)

Problèmes d'Optimisation Combinatoire

- Un problème d'optimisation combinatoire (OC) consiste a déterminer un plus grand (petit) élément dans un ensemble fini valué
- soit une famille $\mathcal{F}$ de sous-ensembles d'un ensemble fini $E = \{e_1, \ldots, e_n\}$
- et un système de poids $w = (w(e_l), \ldots, w(e_n))$ associé aux éléments de E
- un problème d'optimisation consiste à trouver un ensemble $F \in \mathcal{F}$ de poids $w(F) = \sum_{e \in F} w(e)$ maximum (ou minimum),

- $\mathcal{F}$ représente les solutions du problème
- elle peut être un ensemble de très grande taille (connu que par des descriptions ou des propriétés théoriques)

Problèmes d'Optimisation Combinatoire

- Une première formulation PLNE "naïve" :
- prendre une variable $0-1$ associée a chaque éléments de F
- poser le PLNE associé a ces variables
- le seul algorithme adaptée : l'énumération une a une des solutions
- Inapplicable ! !
- Mais prouve que tout problème d'OC peut être formulé comme un PLNE.

Problèmes d'Optimisation Combinatoire

- plutôt que prendre une variable $0-1$ pour chaque éléments de F :
- prendre une variable $0-1$ pour chaque élément de E
- pour définir une formulation, il faut définir des inégalités
- ces inégalités doivent impliquer que les solutions décrites par la variable doivent être dans l'ensemble $\mathcal{F}$
- il n'est pas toujours simple de déterminer une telle formulation
- plusieurs PLNE peuvent même correspondre à cette formulation

Notations de théorie des graphes

- graphes *finis* et sans *boucle* ni *arête multiple*
- graphe : $G = (V, E)$ où V les *sommets*, et E les *arêtes*
- $e \in E$ est une arête d'extrémités u et v, l'arête e peut être également notée uv.
- Soit $G = (V, E)$ un graphe. Un *sous-graphe* $H = (W, F)$ de G est un graphe tel que $W \in V$ et $F \in E$
- On dit qu'un sous-graphe $H = (W, F)$ de G *couvre* les sommets de G si $W = V$.
- On appelle *sous-graphe partiel* de G un sous-graphe $G' = (V, E')$ avec $E' \subset E$
- Si $W \subset V$, alors $E(W)$ est l'ensemble des arêtes ayant leurs deux extrémités dans W
- Le graphe $H = (W, E(W))$ est le sous-graphe de G induit par W

Notations de théorie des graphes

- Si $F \subset E$, alors $V(F)$ désigne l'ensemble des sommets de V qui sont extrémités des arêtes de F
- Si $W \subset V$, alors $\delta(W)$ est l'ensemble des arêtes avec une extrémité dans W et l'autre extrémité dans $V\ W$
- L'ensemble $\delta(W)$ est appelé une *coupe*
- On note $\delta(v)$ à la place de $\delta(v)$ pour $v \in V$
- on appelle $\delta(v)$ l'*étoile* de v. Pour
- $N(v)$ est l'ensemble des sommets *adjacents* à v
- Si $W \subset V$, on pose $N(W) = (\cup_{v \in W} N(v)) \backslash W$
- on appelle $N(W)$ l'ensemble des *voisins* de W
- Soit $F \in E$. On dénote par $G \backslash F$ le graphe obtenu a partir de G en supprimant les arêtes de F.

Notations de théorie des graphes

- Une *chaine* P dans $G = (V, E)$ est une séquence d'arêtes $e_1, e_2, \ldots, e_k$ telles que $e_1 = v_0 v_1, e_2 =' v_1 v_2, \ldots, e_k = v_{k-1} v_k$
- P est dite *chaine élémentaire* si elle passe au plus une fois par le même sommet
- Les sommets v_0 et v_k sont les *extrémités* de P
- On dit que P relie v_0 à v_k
- Le nombre k d'arêtes de P est appelé la *taille* de P
- Si $P = e_1, e_2, \ldots, e_k$ est une chaîne (resp. chaîne élémentaire) reliant v_0 à v_k et $e_{k+1} = v_0 v_k \in E$, alors la séquence $e_1, e_2, \ldots, e_k, e_{k+1}$ est appelés un *cycle* (resp. *cycle élémentaire*) de taille le $k + 1$
- Un cycle (chaîne) est dit *impair* si la taille est impaire, autrement, il est dit *pair*

Notations de théorie des graphes

- Si P est un cycle ou une chaîne et uv une arête de $E \backslash P$ avec $u, c \in V(P)$, alors uv est appelé une *corde* de P
- Un *trou* de G est un cycle sans corde
- Un graphe est dit *connexe* si, pour tout couple de sommets u et v, il existe une chaîne joignant u et v
- Un chemin (resp. cycle) élémentaire qui passe par tous les sommets d'un graphe est dit *hamiltonien*
- Un chemin (resp. cycle) élémentaire qui passe une fois par toutes les arêtes d'un graphe est dit *eulérien*
- Un graphe $G = (V, E)$ est dit *complet* si toute paire de sommets de V est jointe par une arête de E
- On note par K_q le graphe complet sur q sommets
- On appelle *clique* de G un sous-graphe complet de G
- Un stable d'un graphe est un ensemble de sommets deux a deux non adjacents.

Notations de théorie des graphes

- Un graphe est appelé *biparti* si son ensemble de sommets peut être partitionné en deux ensembles de sommets non vides V_1 et V_2 tels qu'aucun couple de sommets de V_1 (respectivement de V_2) ne soit adjacent

Theorem

un graphe est biparti si et seulement s'il ne contient pas de cycle impair

- On note un graphe *orienté* $G = (V, A)$ où A est l'ensemble des arcs
- Certaines définitions vues précédemment sur les graphes non-orientés sont alors à adapter au cas orienté
- Un *chemin* est une séquences d'arcs
- un chemin qui revient sur son sommet d'origine est un *circuit*

Notations de théorie des graphes

- On note $N^+(v)$ (resp. $N^-(v)$) l'ensemble des sommets *successeurs* (resp. *prédécesseurs*) de v
- On note $\delta^+(v)$ (resp. $\delta^-(v)$) l'ensemble des arcs *sortant* de (resp. *entrant* en) v

Affectation

- un groupe d'employés et de tâches à accomplir
- chaque tâche doit être accomplie
- chaque ouvrier ne peut pas faire plus d'une tâche
- affecter un employé à une tâche coûte un prix
- Minimiser le prix total

Affectation

- employés : n
- tâches : m
- coût d'affecter l'employé i à la tâche j : c_{ij}
- variable de décision : $x_{ij} = 1$ si l'employé i est affecté à la tâche j, 0 sinon

$$Min \sum_{ij} c_{ij} x_{ij}$$

$$\sum_{j=1}^{m} x_{ij} = 1$$

$$\sum_{i=1}^{n} x_{ij} \leq 1$$

$$x_{ij} \in \{0, 1\}$$

Localisation d'usines

- $N = \{1, \ldots; n\}$ les emplacements possibles
- $I = \{1, \ldots; m\}$ les clients
- c_j le coût de positionner l'usine à l'emplacement j
- h_{ij} le coût de satisfaire le client i depuis l'usine j
- variable de décision : $x_j = 1$ si une usine est positionnée en j, 0 sinon
- variable de décision : y_{ij} la fraction de la demande du client i satisfaite par l'usine j

Localisation d'usines

Formulation :

$$Min \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} \sum_{j=1}^{n} h_{ij} y_{ij}$$

$$\sum_{j=1}^{n} y_{ij} = 1, \ \forall i$$

$$y_{ij} \leq x_j, \ \forall j$$

$$x_j \in \{0,1\}, \ 0 \leq y_{ij} \leq 1$$

Localisation d'usines

$$Min \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} \sum_{j=1}^{n} h_{ij} y_{ij}$$

$$\sum_{j=1}^{n} y_{ij} = 1, \ \forall i$$

$$y_{ij} \leq x_j, \ \forall j$$

$$x_j \in \{0, 1\}, \ 0 \leq y_{ij} \leq 1$$

Que se passe-t-il si on remplace une série de contraintes ?
Remplacer $y_{ij} \leq x_j, \ \forall i$ par $\sum_{i=1}^{m} y_{ij} \leq m \times x_j$
Est-ce que les deux sont valides ? Laquelle est préférable ?

Localisation d'usines

- Les deux formulations donnent le même ensemble de solutions
- Mais si on relâche la contrainte $x_j \in \{0, 1\}$, alors on a deux espaces de solutions différents
- l'un est plus petit que l'autre. Lequel ?
- Que peut-on en conclure ?

Localisation d'usines

- La meilleure formulation est celle dont les solutions avec et sans la contrainte $x_j \in \{0, 1\}$ sont les plus proches
- il s'agit de la première formulation : $y_{ij} \leq x_j, \ \forall i$
- Même si elle contient plus de contraintes

Le problème du sac-à-dos

- Soit n objets, notés $i = 1, \ldots, n$, de bénéfice c_i, de poids a_i.
- On veut ranger ces objets dans un "sac"
- Poids maximum du sac : b
- Le problème de *sac-à-dos (knapsack)* :
  - choisir parmi les n objets
  - maximiser le bénéfice
  - respecter la contrainte de poids maximum
  - chaque objet $i, i \in \{1, \ldots, n\}$, doit être sélectionné au moins p_i fois
  - chaque objet $i, i \in \{1, \ldots, n\}$, doit être sélectionné au plus q_i fois.

Problème utilisé pour remplir les camions de transport, les avions ou bateaux de fret, et aussi pour gérer la mémoire d'un microprocesseur.

Le problème du sac-à-dos

Formulation du problème de sac-a-dos par un PLNE :
Pour chaque objet $i, i \in \{1, \ldots, n\}$, on crée une variable entière x_i, correspondant au nombre de fois où l'objet i est choisi.

$$\text{Max} \sum_{i=1}^{n} c_i x_i \tag{1}$$

$$\sum_{i=1}^{n} a_i x_i \leq b \tag{2}$$

$$p_i \leq x_i \leq q_i \ \forall i \in \{1, \ldots, n\} \tag{3}$$

$$x_i \in \mathbb{N} \ \forall i \in \{1, \ldots, n\} \tag{4}$$

(2) est appelée *contrainte de sac-à-dos*.
Une seule contrainte, mais problème NP-complet.

Recouvrement, pavage et partition

- Soit $E = \{1, \ldots, n\}$ un ensemble fini d'éléments
- Soit $E_1, \ldots, E_m$ des sous-ensembles de E
- A chaque ensemble E_j on associe un poids $c_j, j = 1, \ldots, m$

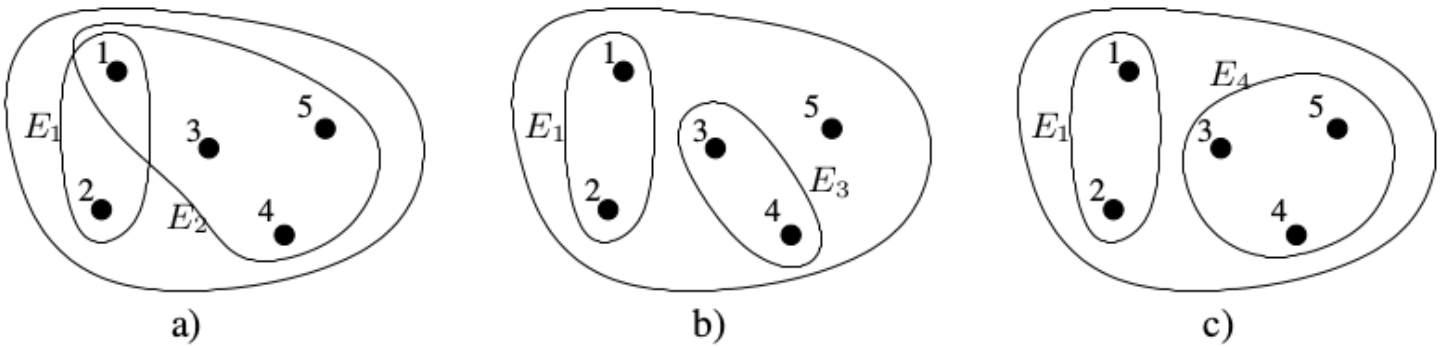
Une famille $F \subseteq \{E_1, \ldots, E_m\}$ est dite :

- un *recouvrement* de E si $\cup_{E_j \in F} E_j = E$
- un *pavage* de E si $E_j \cap E_k = \emptyset$, pour tout $j \neq k \in \{1, \ldots, m\}$
- une *partition* de E si F est a la fois un recouvrement et un pavage.

Recouvrement, pavage et partition

Exemple :

- $E = \{1, 2, 3, 4, 5\}$
- 4 sous-ensembles $E_1 = \{1, 2\}$, $E_2 = \{1, 3, 4, 5\}$, $E_3 = \{3, 4\}$ et $E_4 = \{3, 4, 5\}$.
- Alors : $\{E_1, E_2\}$ est un recouvrement, que $\{E_1, E_3\}$ est un pavage et $\{E_1, E_4\}$ est une partition.



Problème de recouvrement : déterminer un recouvrement de somme des poids des sous-ensembles est minimum (resp. maximum, minimum/ maximum).
(pavage : poids maximum, partition : poids minimum/maximum)

Recouvrement, pavage et partition

Modélisation :

- des variables binaires $x_1, \ldots, x_m$ associées aux sous-ensembles $E_1, \ldots, E_m$
- Soit A la matrice en $0-1$ dont les lignes correspondent aux éléments $1, \ldots, n$ et les colonnes aux sous-ensembles $E_1, \ldots, E_m$
- ses coefficients sont $A_{ij} = 1$ si $i \in E_j$ et 0 sinon.

Recouvrement :

$$\text{Min} \sum_{j=1}^{m} c_j x_j$$

$$\sum_{j} A_{ij} x_j \geq 1$$

$$x \in \{0,1\}^m$$

Recouvrement, pavage et partition

Pavage :

$$\text{Max} \sum_{j=1}^{m} c_j x_j$$
$$\sum_{j} A_{ij} x_j \leq 1$$
$$x \in \{0,1\}^m$$

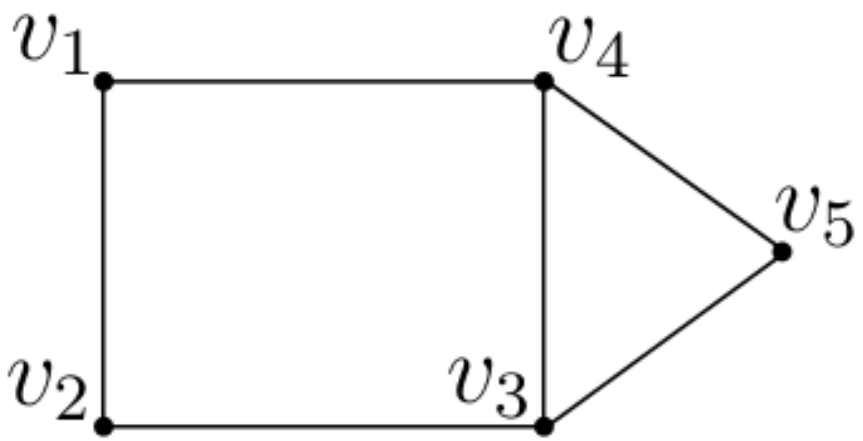
Partition :

$$\text{Max (ou Min)} \sum_{j=1}^{m} c_j x_j$$
$$\sum_{j} A_{ij} x_j = 1$$
$$x \in \{0,1\}^m$$

De nombreuses applications

Le problème du stable

- Soient $G = (V, E)$ un graphe non-orienté
- c une fonction poids pour tout sommet $v \in V$
- Un *stable* de G est un sous-ensemble $S \subseteq V$ tel qu'il n'existe aucune arête de E entre 2 sommets de S
- Le problème du stable de poids maximum consiste a déterminer un stable S de G tel que $c(S) = \sum_{c \in S} c(v)$ soit maximum.



Exemple : stable max = 2

Le problème du stable

Modélisation :

- utiliser la variable $x(u)$ valant 1 si $u \in S$, 0 sinon
- Alors l'inégalité $x(u) + x(v) \leq 1$ pour tout $uv \in E$ suffit !

$$\text{Max} \sum_{u \in V} c(u)x(u)$$
$$x(u) + x(v) \leq 1 \ \forall uv \in E$$
$$x(u) \in \{0, 1\} \ \forall u \in V$$

- une formulation est *compacte* si elle contient un nombre polynomial de variables et de contraintes
- cette formulation du stable est compacte
- En effet : une variable par sommet du graphe et une inégalité par arête

Le problème du stable

- Le problème du stable a une formulation PLNE compacte
- Le problème du stable est pourtant NP-complet et difficile à résoudre.

Autre formulation : remplacer $x(u) + x(v) \leq 1 \ \forall uv \in E$ par

$$\sum_{i \in K} x(u) \leq 1 \text{ pour toute clique } K \text{ de } G$$

- Un nombre exponentiel de contraintes (dans le pire des cas)
- la solution de la relaxation linéaire est meilleure
- Or, les algorithmes de branchement et évaluation nous disent que plus la relaxation est bonne, plus l'algorithme est rapide.

Comment choisir une bonne formulation ?
Comment décider si une contrainte est utile ou non à la relaxation ?

Le problème du voyageur de commerce

- Soit $G = (V, E)$ un graphe non-orienté
- Soit $c(e)$ un coût associé à la liaison $\forall e \in E$.
- Le problème du *voyageur de commerce* : trouver un cycle passant par chaque sommet une et une seule fois
- on appelle ce cycle *hamiltonien*
- Cas assymétrique : graphe orienté
- version pondérée : on cherche le cycle de poids minimum

Le problème du voyageur de commerce

Modélisation :

- Soit les variables $x_{ij} = 1$ si l'arc ij est dans le tour et 0 sinon.

$$
\begin{aligned}
Min \sum_{ij} c_{ij} \times x_{ij} & \\
\sum_{j \in V} x_{ij} = 1 & \qquad \forall i \in V \\
\sum_{i \in V} x_{ij} = 1 & \qquad \forall j \in V \\
0 \leq x_{ij} \leq 1 & \qquad \forall (i,j) \in V \times V \\
x_{ij} \in \mathbb{N} & \qquad \forall (i,j) \in V \times V
\end{aligned}
$$

Le problème du voyageur de commerce

Modélisation :

- Soit les variables $x_{ij} = 1$ si l'arc ij est dans le tour et 0 sinon.

$$
\begin{aligned}
Min \sum_{ij} c_{ij} \times x_{ij} & \\
\sum_{j \in V} x_{ij} = 1 & \qquad \forall i \in V \\
\sum_{i \in V} x_{ij} = 1 & \qquad \forall j \in V \\
0 \leq x_{ij} \leq 1 & \qquad \forall (i,j) \in V \times V \\
x_{ij} \in \mathbb{N} & \qquad \forall (i,j) \in V \times V
\end{aligned}
$$

La relaxation linéaire entière (car matrice totalement unimodulaire).
Mais, la solution peuvent contenir plusieurs cycles

Le problème du voyageur de commerce

Elimination des "sous-tours" par la formulation MTZ

- Pour éliminer les sous-tours, on peut utiliser la formulation de Miller-Tucker-Zemlin (MTZ).
- On ajoute de nouvelles variables réelles $u_i, i = 1..n$, pour chaque sommet
- on ajoute les contraintes :

$$u_1 = 1$$
$$2 \leq u_i \leq n, \ \forall i \neq 1$$
$$u_i - u_j + 1 \leq n(1 - x_{ij}), \ \forall i \neq 1, j \neq 1$$

- pour tout arc (i, j) où $x_{ij} = 1$, elles forcent $u_j \geq u_i + 1$,
- si une solution du PLNE précédent contient plus d'un sous-tour, alors l'un d'eux au moins ne contient pas le sommet 1 et, sur ce sous-tour, les variables u_i s'incrémentent a l'infini.

Le problème du voyageur de commerce

- La formulation MTZ a l'énorme avantage d'être compacte
- On peut donc directement utiliser une procédure de branchement et séparation (Branch& Bound)
- et donc un solveur entier
- Mais cette formulation est plus faible que la formulation par les coupes

Le problème du voyageur de commerce

Elimination des sous-tours en brisant les sous-tours
Ajouter les contraintes :

$$\sum_{i \in S, j \in S} x_{ij} \leq |S| - 1, \ \forall S \subset V, |S| \geq 2, S \neq V$$

- Elles empêchent les solutions d'avoir des sous-tours
- **Mais** en nombres exponentiel
- Utilisation des algorithmes de coupes (grâce à une étude polyédrale)
- ajouter progressivement les inégalités dans la formulation
- un algorithme de séparation, qui revient à détecter des cycles, ajoute des coupes (contraintes)

Le problème du voyageur de commerce

Elimination des sous-tours par la connexité par les coupes

- une solution du TSP est un ensemble d'arcs C qui forme un sous-graphe partiel connexe
- il y a donc au moins un arc sortant de chaque ensemble de sommets
- donc au moins une arête par coupes du graphe.

$$\sum_{e \in \delta^+(W)} x(e) \geq 1 \ \forall W \subsetneq V \text{ et } W \neq \emptyset$$

$$\sum_{e \in \delta^-(W)} x(e) \geq 1 \ \forall W \subsetneq V \text{ et } W \neq \emptyset$$

- Nombre exponentiel de contraintres de connexité.
- Utiliser un algorithme de coupes et branchement
- C'est actuellement la méthode la plus efficace

Le problème du voyageur de commerce

Cas symétrique

Modélisation :

$$Min \sum_{e \in E} c(e)x(e)$$

$$\sum_{e \in \delta(v)} x(e) = 2, \ \forall u \in V$$

$$\sum_{e \in \delta(W)} x(e) \geq 2, \ \forall W \subsetneq V \text{ et } W \neq \emptyset$$

$$x(e) \in \{0,1\}, \ \forall e \in E$$

- le nombre d'arétes incidentes à un sommet $= 2$
- l'ensemble des arêtes choisies doit être connexe
- nombre exponentiel de contraintes (difficiles à trouver)
- **Comment résoudre avec tel programme ?**
- **Comment produire les contraintes d'un tel programme**

Le problème de coloration

- Une *coloration* des sommets d'un graphe $G = (V, E)$ est une fonction $r : V \to \mathbb{N}$ telle que $r(u) \neq r(v)$ pour tout couple de sommets adjacents u, v.
- Ces entiers sont appelés les *couleurs*
- Une *k-coloration* est une coloration $r : V \to \{1, \ldots, k\}$
- Un graphe est dit *k-coloriable* s'il possède une k-coloration

- Tester si un graphe est k-coloriable est un problème NP-complet si $k \geq 3$
- mais polynomial si $k = 2$ (graphe biparti)
- Soit $G = (V, E)$ un graphe non-orienté. Le problème de coloration consiste à déterminer le plus petit k tel que G soit k-coloriable.

Le problème de coloration

Modélisation :

- $\forall u \in V$, créer un vecteur binaire $x_u = (x_u^1, \ldots, x_u^K)$
- où K est une borne supérieure sur k (au pire, $K = |V|$)
- ajouter une variable binaire w_l par couleur $l = 1, \ldots, K$ indiquant si cette couleur a été utilisée ou non

$$Min \sum_{i=1}^{K} w_l \tag{5}$$

$$\sum_{l=1}^{K} x_u^l = 1, \ \forall u \in V \tag{6}$$

$$x_u^l + x_v^l \leq w_l, \ \forall uv \in E, \ \forall 1 \leq l \leq K \tag{7}$$

$$x_u^l \in \{0, 1\}, \ \forall u \in V, \ \forall 1 \leq l \leq K \tag{8}$$

Le problème de coloration

- (6) $\Rightarrow$ chaque sommet a une et une seule couleur
- (7) $\Rightarrow$ deux sommets adjacents n'ont pas la mème couleur

- au pire, on a $n^2 + n$ variables binaires et $n \times m$ contraintes
- taille polynomiale raisonnable
- mais ce PLNE est en pratique très difficile à résoudre par des algorithmes de branchement

Le problème de coloration

- une coloration est en fait une couverture d'un graphe par un nombre minimum de stables
- La formulation suivante donne une bonne borne inférieure (et non un résultat exact)
- soit S l'ensemble des stables non vides de G.
- On associe a chaque stable $S \in \mathcal{S}$ une variable binaire t_S
- La coloration revient au PLNE (*Mehrotra et Trick 1995*).

$$Min \sum_{S \in \mathcal{S}} t_S$$

$$\sum_{s \in S\mathcal{S}} t_S = 1, \ \forall u \in V$$

$$t_S \in \{0,1\}, \ \forall S \in \mathcal{S}$$

Le problème de coloration

- nombre exponentiel de variables
- nombre très réduit de contraintes
- résolution par génération de colonnes
- génération de colonnes : consiste à générer des variables que l'on ajoute itérativement au problème
- méthode duale de la méthode des coupes

Le problème du flot max

- Soit $G = (V, A)$ un graphe orienté
- soit une capacité $c(a) \in \mathbb{N},\ \forall a \in A$
- On suppose que G contient un sommet s, appelé *source*, sans prédécesseur a partir duquel on peut atteindre tout sommet de G
- et un sommet t, appelé *puits*, qui est accessible depuis tout sommet de G.
- Un $s - t$-*flot* est un vecteur positif $x \in \mathbb{R}^m$ s'il vérifie :

$$\sum_{a \in \delta^+(u)} x(a) - \sum_{a \in \delta^-(u)} x(a) = 0,\ \forall u \in V \backslash \{s, t\}$$

- Un flot est dit *réalisable* si $x(a) \leq c(a),\ \forall a \in A$
- un flot est de capacité maximale si $\sum_{a \in \delta^+(s)} x(a)$ est maximal
- *Ford-Fulkerson* : c entier $\Rightarrow\ \exists$ un flot optimal entier
- on sait le déterminer en temps polynomial (et rapidement).

Le problème du flot max

Modélisation :

$$Max\ v$$

$$\sum_{a \in \delta^+(u)} x(a) - \sum_{a \in \delta^-(u)} x(a) = 0,\ \forall u \in V \backslash \{s, t\}$$

$$\sum_{a \in \delta^+(s)} x(a) - v = 0$$

$$\sum_{a \in \delta^-(t)} x(a) - v = 0$$

$$x(a) \leq c(a),\ \forall a \in A$$

$$v \geq 0$$

$$x(a) \geq 0,\ \forall a \in A$$

Le problème du flot max

Theorem

si $c \in \mathbb{N}$, alors le PL précédent a une solution optimale $\in \mathbb{N}$

Le problème du couplage biparti

- Soit un graphe biparti complet $G = (V_1 \cup V_2, E)$
- soit un poids $c(e)$, $\forall e \in E$
- On appelle couplage F de G un ensemble d'arêtes deux a deux non incidentes.

- Le problème du couplage biparti : trouver un couplage qui maximise la somme des poids de ses arêtes.
- Il existe plusieurs algorithmes polynomiaux efficaces pour résoudre ce problème classique
- Il a été montré qu'une affectation est entière dès que le vecteur poids c est entier

Le problème du couplage biparti

Modélisation :

$$
\begin{aligned}
Max \sum_{e \in F} & c(e)x(e) \\
\sum_{e \in \delta(u)} & x(e) \leq 1, \ \forall u \in V_1 \\
\sum_{e \in \delta(u)} & x(e) \leq 1, \ \forall u \in V_2 \\
& x(e) \geq 0, \ \forall e \in E
\end{aligned}
$$

Theorem

si $c \in \mathbb{N}$, alors ce PL a une solution optimale $\in \mathbb{N}$