

Master d'Informatique - Spécialité Androide
Module MAOA

Recherche Opérationnelle
et Optimisation Combinatoire

Partie F - Génération de colonnes
(Branch-and-Price)

Pierre Fouilhoux

Sorbonne Université

2019-2020

1. Principe de la génération de colonnes sur un exemple
2. Définitions et notations
3. Décomposition de Dantzig-Wolfe
4. Application au problème de coloration

1. Principe de la génération de colonnes sur un exemple

1.1 Le problème de multiflot-max

1.2 Génération de colonnes pour le multiflot-max fractionnaire

1.3 Le problème de multiflot de coût minimum

1.4 Initialisation d'une génération de colonnes

1.5 Résoudre le pricing

1.6 Branchement pour le multiflot entier

2. Définitions et notations

3. Décomposition de Dantzig-Wolfe

4. Application au problème de coloration

└ Principe de la génération de colonnes sur un exemple

└ Le problème de multiflot-max

1. Principe de la génération de colonnes sur un exemple

1.1 Le problème de multiflot-max

1.2 Génération de colonnes pour le multiflot-max fractionnaire

1.3 Le problème de multiflot de coût minimum

1.4 Initialisation d'une génération de colonnes

1.5 Résoudre le pricing

1.6 Branchement pour le multiflot entier

2. Définitions et notations

3. Décomposition de Dantzig-Wolfe

4. Application au problème de coloration

Un exemple : Le problème de multiflot-max

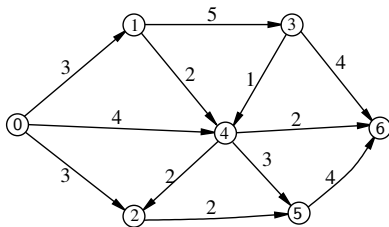
Soit $G = (V, A)$ un graphe orienté
et k **commodités**, c'est-à-dire des paires de sommets $(s_1, t_1), \dots, (s_k, t_k)$ entre
lesquelles doit circuler un flot.

Le réseau G est muni d'une capacité $b(a) \in \mathbf{N}$ associé à chaque arc $a \in A$.
Pour chaque paire $i \in \{1, \dots, k\}$, on désire trouver des (s_i, t_i) -flots entiers tels que la
somme des capacités utilisées par les flots sur un même arc respecte la capacité de cet
arc.

L'objectif du **problème de multiflot-max** est de maximiser la quantité total de flot
circulant sur le graphe.

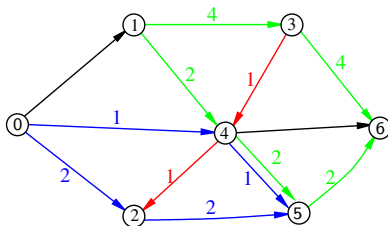
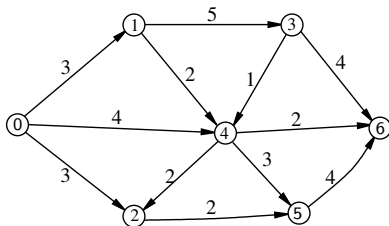
Si flots continus, le problème de **multiflot-max fractionnaire** est polynomial.
Si flots entiers, le problème de **multiflot-max entier** est NP-difficile dès que $k \geq 2$.

Un exemple : Le problème de multiflot-max



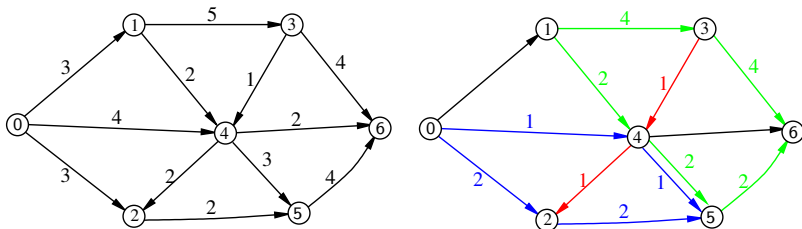
Commodités : (0,5), (1,6), (3,2)

Un exemple : Le problème de multiflot-max



Commodités : (0,5), (1,6), (3,2)

Un exemple : Le problème de multiflot-max



Commodités : (0,5), (1,6), (3,2)

On peut noter que les trois flots reliant chacune des trois commodités sont en concurrence sur le graphe.

L'arc (4, 5) porte à la fois du flot des commodités (0, 5) et (1, 6).

Notons que pour chaque nœud, il y a une conservation des flots individuelle (il n'y a pas de mélange des flots...).

Une première formulation naturelle

Considérons des variables $f^i(a)$ indiquant la quantité de flot passant par l'arc a pour desservir la commodité (s_i, t_i) .

$$\begin{aligned} \text{Max } & \sum_{i=1}^k \sum_{u \in \delta^+(s_i)} f^i(s_i u) \\ & \sum_{v \in \delta^-(u)} f^i(vu) = \sum_{v \in \delta^+(u)} f^i(uv) \quad \forall i \in \{1, \dots, k\}, \forall u \in V, \\ & \sum_{i=1}^k f^i(a) \leq b(a) \quad \forall a \in A, \\ & f^i(a) \geq 0 \quad \forall i \in \{1, \dots, k\}, \forall a \in A. \end{aligned}$$

Ce problème est donc polynomial en utilisant un PL avec km variables et $kn + m$ inégalités.

Pour un graphe de 1000 sommets, 10000 arêtes et des commodités entre toute paire de sommets du graphe, cela représente un taille de PL trop importante pour être traitée par un solveur linéaire... mais il existe une reformulation bien plus efficace !

Reformulation

Une reformulation peut être obtenue en considérant d'autres variables.

Pour chaque commodité (s_i, t_i) , $i \in \{1, \dots, k\}$, considérons $\mathcal{P}_{s_i t_i}$ l'ensemble des chemins (élémentaires) allant du sommet s_i au sommet t_i .

Posons $\mathcal{P} = \bigcup_{i=1}^k \mathcal{P}_{s_i t_i}$.

Il est bien connu qu'un (s, t) -flot est un ensemble de chemins allant de s à t et portant une part du flot allant de s à t . L'idée est de chercher, pour chaque commodité, un tel ensemble de chemins.

On associe une variable f_μ associée à chaque chemin $\mu \in \mathcal{P}$.

Reformulation

La formulation (\tilde{F}) est un programme linéaire, dite **formulation arc-chemin**, qui modélise le problème de multiflot-max fractionnaire.

$$\begin{aligned}
 (\tilde{F}) \quad & \text{Max } \sum_{\mu \in \mathcal{P}} f_{\mu} \\
 & \sum_{\substack{\mu \in \mathcal{P} \\ a \in \mu}} f_{\mu} \leq b(a) \quad \forall a \in A \\
 & f_{\mu} \geq 0 \quad \forall \mu \in \mathcal{P}.
 \end{aligned}$$

Reformulation

La formulation (\tilde{F}) est un programme linéaire, dite **formulation arc-chemin**, qui modélise le problème de multiflot-max fractionnaire.

$$\begin{aligned}
 (\tilde{F}) \quad & \text{Max} \sum_{\mu \in \mathcal{P}} f_{\mu} \\
 & \sum_{\substack{\mu \in \mathcal{P} \\ a \in \mu}} f_{\mu} \leq b(a) \quad \forall a \in A \\
 & f_{\mu} \geq 0 \quad \forall \mu \in \mathcal{P}.
 \end{aligned}$$

Si on ajoute en plus la contrainte d'intégrité, on obtient alors une formulation (F) qui est un PLNE modélisant le problème de multiflot-max entier.

$$\begin{aligned}
 (F) \quad & \text{Max} \sum_{\mu \in \mathcal{P}} f_{\mu} \\
 & \sum_{\substack{\mu \in \mathcal{P} \\ a \in \mu}} f_{\mu} \leq b(a) \quad \forall a \in A \\
 & f_{\mu} \geq 0 \quad \forall \mu \in \mathcal{P}. \\
 & f_{\mu} \in \mathbf{N} \quad \forall \mu \in \mathcal{P}.
 \end{aligned}$$

Notez donc que (\tilde{F}) est la relaxation linéaire de (F) .

└ Principe de la génération de colonnes sur un exemple

└ Génération de colonnes pour le multiflot-max fractionnaire

1. Principe de la génération de colonnes sur un exemple

1.1 Le problème de multiflot-max

1.2 Génération de colonnes pour le multiflot-max fractionnaire

1.3 Le problème de multiflot de coût minimum

1.4 Initialisation d'une génération de colonnes

1.5 Résoudre le pricing

1.6 Branchement pour le multiflot entier

2. Définitions et notations

3. Décomposition de Dantzig-Wolfe

4. Application au problème de coloration

Principe de la génération de colonnes sur un exemple

La formulation PL arc-chemin

$$\begin{aligned}
 (\tilde{F}) \quad & \text{Max} \sum_{\mu \in \mathcal{P}} f_{\mu} \\
 & \sum_{\substack{\mu \in \mathcal{P} \\ a \in \mu}} f_{\mu} \leq b(a) \quad \forall a \in A \\
 & f_{\mu} \geq 0 \quad \forall \mu \in \mathcal{P}.
 \end{aligned}$$

possède seulement m inégalités... mais un nombre exponentiel de variables !

Ce qui veut dire que même sa relaxation linéaire ne peut pas être directement obtenue par un solveur linéaire (algo du simplexe,...). On résout la relaxation linéaire d'une telle formulation par l'**algorithme de génération de colonnes** (column generation, en anglais)

Rappel : Dualité

Pour un programme linéaire (\tilde{P}) , appelé alors *primal*, le *dual* est le programme linéaire (\tilde{D}) suivant

$$(\tilde{P}) \left\{ \begin{array}{l} \text{Max} \quad z = c^T x \\ Ax \leq b \\ x \geq 0 \end{array} \right. \quad (\tilde{D}) \left\{ \begin{array}{l} \text{Min} \quad w = b^T y \\ A^T y \geq c \\ y \geq 0 \end{array} \right.$$

Sous forme "algébrique" :

$$(\tilde{P}) \left\{ \begin{array}{l} \text{Max} \quad z = \sum_{j=1}^n c_j x_j \\ \sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i = 1, \dots, m \\ x_j \geq 0 \quad \forall j = 1, \dots, n \end{array} \right. \quad (\tilde{D}) \left\{ \begin{array}{l} \text{Min} \quad w = \sum_{i=1}^m b_i y_i \\ \sum_{i=1}^m a_{ij} y_i \geq c_j \quad \forall j = 1, \dots, n \\ y_i \geq 0 \quad \forall i = 1, \dots, m \end{array} \right.$$

Les **variables duales** de (\tilde{D}) correspondent aux inégalités de (\tilde{P}) .

La matrice de (\tilde{D}) est la transposée A^T .

Les coûts de la fonction objective et les termes de droite des inégalités échangent leurs rôles.

Rappel : Dualité

- *Théorème (faible) de la dualité :*

Si (\tilde{P}) et (\tilde{D}) admettent chacun une solution \tilde{x} et \tilde{y} ,
alors $c^T \tilde{x} \leq b^T \tilde{y}$.

- *Théorème de la dualité :*

Si (\tilde{P}) admet une solution optimale (finie),
alors (\tilde{D}) aussi et de plus elles "coïncident", *i.e.*

$$(\tilde{P}) \quad \max\{c^T x \mid Ax \leq b\} = \min\{b^T y \mid y^T A = c, y \geq 0\} \quad (\tilde{D}).$$

- **Corollaire de la dualité :**

Soit x une solution de (\tilde{P}) et y une solution de (\tilde{D}) . Alors
 x et y sont solutions optimales de (\tilde{P}) et (\tilde{D}) **si et seulement si** $c^T x = b^T y$.

Principe de la génération de colonnes sur un exemple

Comme (\tilde{F}) est un programme linéaire, considérons son dual (\tilde{D}) .

On note λ_a la variable duale associée à l'inégalité associée à un arc $a \in A$ de (\tilde{F}) .

$$\begin{aligned} \text{Max} \quad & \sum_{\mu \in \mathcal{P}} f_{\mu} \\ & \sum_{\substack{\mu \in \mathcal{P} \\ a \in \mu}} f_{\mu} \leq b(a) \quad \forall a \in A \\ & f_{\mu} \geq 0 \quad \forall \mu \in \mathcal{P}. \end{aligned}$$

$$\begin{aligned} \text{Min} \quad & \sum_{a \in A} b(a) \lambda_a \\ & \sum_{a \in \mu} \lambda_a \geq 1 \quad \forall \mu \in \mathcal{P}, \\ & \lambda_a \geq 0 \quad \forall a \in A. \end{aligned}$$

En tant que transposé, le dual (\tilde{D}) possède alors un nombre exponentiel de lignes mais par contre un nombre fini de variables !

Principe de la génération de colonnes sur un exemple

Comme (\tilde{F}) est un programme linéaire, considérons son dual (\tilde{D}) .

On note λ_a la variable duale associée à l'inégalité associée à un arc $a \in A$ de (\tilde{F}) .

$$\begin{aligned} \text{Max} \quad & \sum_{\mu \in \mathcal{P}} f_{\mu} \\ & \sum_{\substack{\mu \in \mathcal{P} \\ a \in \mu}} f_{\mu} \leq b(a) \quad \forall a \in A \\ & f_{\mu} \geq 0 \quad \forall \mu \in \mathcal{P}. \end{aligned}$$

$$\begin{aligned} \text{Min} \quad & \sum_{a \in A} b(a)\lambda_a \\ & \sum_{a \in \mu} \lambda_a \geq 1 \quad \forall \mu \in \mathcal{P}, \\ & \lambda_a \geq 0 \quad \forall a \in A. \end{aligned}$$

En tant que transposé, le dual (\tilde{D}) possède alors un nombre exponentiel de lignes mais par contre un nombre fini de variables !

La condition d'optimalité donné par le corollaire de la dualité est la suivante :

Si l'on sait produire à la fois :

- une solution f^* du primal (\tilde{F})
 - et une solution λ^* du dual (\tilde{D})
 - qui ont même valeur objective,
- alors f^* est optimale pour (\tilde{F}) .

Principe de la génération de colonnes sur un exemple

Initialisation :

Dans ce cas simple de multiflot-max fractionnaire, il est facile d'avoir une solution réalisable.

En effet, pour chaque commodité (s_i, t_i) , prenons un chemin quelconque μ_i^0 reliant s_i à t_i .

Considérons alors l'ensemble \mathcal{P}_0 de ces chemins.

Et les k variables f_μ associée aux chemins de \mathcal{P}_0 .

Alors la formulation arc-chemin restreinte à ces variables admet une solution **et** cette solution est bien une solution du problème de multiflot-max fractionnaire (au pire les valeurs des flots véhiculés sont toutes nulles).

Considérons :

- le PL initial (\tilde{F}_0) qui est la restriction de (\tilde{F}) aux k variables $f_\mu, \mu \in \mathcal{P}_0$.
- son dual (\tilde{D}_0) .

Notons que (\tilde{D}_0) est aussi la restriction de (\tilde{D}) aux k inégalités associées aux chemins $\mu \in \mathcal{P}_0$.

Principe de la génération de colonnes sur un exemple

Programme linéaire initial (\tilde{F}_0) et son dual (\tilde{D}_0).

$$\begin{array}{ll}
 \text{Max} & \sum_{\mu \in \mathcal{P}_0} f_\mu \\
 & \sum_{\substack{\mu \in \mathcal{P}_0 \\ f_\mu \geq 0}} f_\mu \leq b(a) \quad \forall a \in A \\
 & \forall \mu \in \mathcal{P}_0.
 \end{array}
 \qquad
 \begin{array}{ll}
 \text{Min} & \sum_{a \in A} b(a)\lambda_a \\
 & \sum_{a \in \mu} \lambda_a \geq 1 \quad \forall \mu \in \mathcal{P}_0, \\
 & \lambda_a \geq 0 \quad \forall a \in A.
 \end{array}$$

Résoudre le PL compact (\tilde{F}_0) (et en même temps son dual (\tilde{D}_0)) est polynomial.

Notons f^0 la solution optimale de (\tilde{F}_0)
 et λ^0 la solution optimale de (\tilde{D}_0).

Remarques essentielles :

- λ^0 est un vecteur candidat à être solution de (\tilde{D}) tout entier.
 - f^0 possède seulement k composantes parmi le nombre exponentiel possible dans (\tilde{F}).
- Posons alors f^* le vecteur de (\tilde{F}) obtenu en fixant à 0 toutes les colonnes n'apparaissant pas dans \mathcal{P}_0 : **alors f^* est une solution de (\tilde{F}) !**

Principe de la génération de colonnes sur un exemple

On peut déjà tester l'optimalité de f^* :

En effet, par dualité forte, on sait que les fonctions objective de f^0 et λ^0 sont identiques pour (\tilde{F}_0) et pour (\tilde{D}_0) .

De plus, par construction (ajout des valeurs nulles), la fonction objective pour f^0 dans (\tilde{F}_0) est la même que celle de $*$ dans (\tilde{F}) .

Donc le corollaire de la dualité nous dit que :

Si λ^0 est solution de (\tilde{D}) , alors f^* est optimale pour (\tilde{F}) .

Principe de la génération de colonnes sur un exemple

On peut déjà tester l'optimalité de f^* :

En effet, par dualité forte, on sait que les fonctions objective de f^0 et λ^0 sont identiques pour (\tilde{F}_0) et pour (\tilde{D}_0) .

De plus, par construction (ajout des valeurs nulles), la fonction objective pour f^0 dans (\tilde{F}_0) est la même que celle de f^* dans (\tilde{F}) .

Donc le corollaire de la dualité nous dit que :

Si λ^0 est solution de (\tilde{D}) , alors f^* est optimale pour (\tilde{F}) .

Or tester si λ^0 est solution de (\tilde{D}) revient à tester si le vecteur λ^0 vérifie les contraintes du dual, c'est-à-dire si

$$\sum_{a \in \mu} \lambda_a \geq 1 \quad \forall \mu \in \mathcal{P}$$

Cette condition est donc vérifiée si le vecteur λ^0 vérifie ce lot d'inégalités qui est en nombre exponentiel : autant qu'il y a de chemins !

On appelle ce test : le problème de "pricing" (ou en français le problème de génération d'une colonne).

Principe de la génération de colonnes sur un exemple

Algorithme de génération de colonnes :

- Si le problème de pricing renvoie "vraie" : alors vecteur dual λ^0 vérifie les inégalités et la solution primale f^* est optimale.
- Sinon (le problème de pricing renvoie "faux") : alors il existe une inégalité violée par λ^0 : cette inégalité correspond à un chemin $\mu \in \mathcal{P}$. On ajoute alors la variable f_μ au PL maître

$$(\tilde{F}_{i+1}) \leftarrow (\tilde{F}_i) \text{ plus la variable } f_\mu$$

Et on recommence pour $(\tilde{F}_{i+1})!$

C'est le principe de génération de colonnes (ou de variables).

Principe de la génération de colonnes sur un exemple

Validité de l'algorithme :

L'algorithme, dans le pire des cas, va ajouter toutes les variables possibles : il est donc trivialement valide.

Complexité de l'algorithme :

Il s'agit de la même question que la complexité de la méthode de coupes : en effet, le problème de pricing est le problème de séparation pour le dual de la formulation !

Donc par le résultat de Groëtschel-Lovasz-Schrijver :

si le problème de pricing se résout en temps polynomial (i.e. si λ^0 vérifie les inégalités), alors la réitération du problème de pricing dans une méthode de génération de colonnes est polynomiale !

Principe de la génération de colonnes sur un exemple

Le problème de pricing de cette génération de colonnes pour le problème de multiflot est de tester les inégalités du dual pour chaque chemin de \mathcal{P} , c'est-à-dire si

$$\sum_{a \in \mu} \lambda'_a \geq 1 \quad \forall \mu \in \mathcal{P}.$$

C'est-à-dire

Tester s'il existe un chemin μ correspondant à une contrainte violée et répondre VRAI en produisant ce chemin

Et sinon répondre FAUX.

On recherche donc un algorithme capable de résoudre le problème de pricing : **un tel algorithme est appelé en anglais pricer.**

Principe de la génération de colonnes sur un exemple

Ici ce problème de pricing se ramène à déterminer, pour chaque paire de commodités (s_i, t_i) , un plus court chemin de s_i à t_i selon les poids λ^0 associés aux arcs de A :

Si le plus court de ces chemins est de valeur ≥ 1 , alors toutes les inégalités du dual sont vérifiées : le problème de pricing renvoie FAUX.

Sinon, le plus court de ces chemins viole l'inégalité et ce chemin μ^* est retourné par le problème de pricing.

Dans ce dernier cas, on crée la variable f_{μ^*} et on l'ajoute au problème maître.

Comme les coûts duaux λ^0 sont positifs, la recherche d'un plus court chemin est polynomial (par exemple avec l'algorithme de Dijkstra $O(m \log(n))$), le problème de pricing peut être résolu en $O(km \log(n))$: un tel algorithme résout bien le problème de pricing (c'est un pricer).

Comme ce pricer est polynomial, sa réitération dans une génération de colonnes est alors polynomiale !

La formulation arc-chemin pour le problème de multiflot-max fractionnaire peut donc être résolue en temps polynomial !

Principe de la génération de colonnes sur un exemple

On peut noter que cette méthode de génération de colonnes suit le principe de l'algorithme du simplexe : entrée d'une variable en base en fonction des coûts réduits.

La méthode de génération de colonnes suit ainsi les bonnes propriétés observées (et par encore tout à fait bien comprises) de l'algorithme du simplexe comme le fait de rarement nécessiter beaucoup d'itérations (et donc de variables).

Malheureusement, la méthode de génération de colonnes suit aussi les mauvaises propriétés de l'algorithme du simplexe : bien souvent il s'agit d'itérations fortement dégénérées où plusieurs bases donnent la même solution ! Et il est plus difficile que dans le cas du simplexe de faire converger la méthode : des outils d'optimisation continue sont souvent nécessaires.

Principe de la génération de colonnes sur un exemple

On peut noter que cette méthode de génération de colonnes suit le principe de l'algorithme du simplexe : entrée d'une variable en base en fonction des coûts réduits.

La méthode de génération de colonnes suit ainsi les bonnes propriétés observées (et par encore tout à fait bien comprises) de l'algorithme du simplexe comme le fait de rarement nécessiter beaucoup d'itérations (et donc de variables).

Malheureusement, la méthode de génération de colonnes suit aussi les mauvaises propriétés de l'algorithme du simplexe : bien souvent il s'agit d'itérations fortement dégénérées où plusieurs bases donnent la même solution ! Et il est plus difficile que dans le cas du simplexe de faire converger la méthode : des outils d'optimisation continue sont souvent nécessaires.

Dans le cas du problème de multiflot fractionnaire, la convergence est rapide et cette formulation est très efficace, bien plus efficace que la formulation naturelle avec un nombre polynomial mais très élevée de variables.

└ Principe de la génération de colonnes sur un exemple

└ Le problème de multiflot de coût minimum

1. Principe de la génération de colonnes sur un exemple

1.1 Le problème de multiflot-max

1.2 Génération de colonnes pour le multiflot-max fractionnaire

1.3 Le problème de multiflot de coût minimum

1.4 Initialisation d'une génération de colonnes

1.5 Résoudre le pricing

1.6 Branchement pour le multiflot entier

2. Définitions et notations

3. Décomposition de Dantzig-Wolfe

4. Application au problème de coloration

Le problème de multiflot de coût minimum

Le problème de multiflot-max considéré jusqu'ici est une version simplifiée du problème classique de multiflot.

Soit $G = (V, A)$ un graphe orienté et k **commodités**, c'est-à-dire des triplets $(s_1, t_1, d_1), \dots, (s_k, t_k, d_k)$ où (s_i, t_i) est une paire de sommets entre lesquelles doit circuler un flot de valeur au moins d_i .

Le réseau G est muni :

- d'une capacité $b(a) \in \mathbf{N}$ associé à chaque arc $a \in A$
- et d'un coût unitaire (positif) d'utilisation des arcs $c(a)$.

L'objectif du **problème de multiflot** (de coût minimum) est de déterminer k flots reliant les commodités avec un coût total minimal.

Ce problème de multiflot est un modèle assez réaliste pour le routage des données entre les paires de clients voulant communiquer dans un réseau de télécommunications.

Si flots continus, le problème de **multiflot fractionnaire** est polynomial.

Si flots entiers, le problème de **multiflot entier** est NP-difficile dès que $k \geq 2$.

Une première formulation naturelle

Comme pour le multiflot-max, on peut considérer des variables $f^i(a)$ indiquant la quantité de flot passant par l'arc a pour desservir la commodité (s_i, t_i) .

$$\begin{aligned}
 \text{Min } & \sum_{i=1}^k \sum_{a \in A} c(a) f^i(a) \\
 & \sum_{v \in \delta^-(u)} f^i(vu) = \sum_{v \in \delta^+(u)} f^i(uv) & \forall i \in \{1, \dots, k\}, \forall u \in V \setminus \{s_i, t_i\}, \\
 & \sum_{v \in \delta^-(s_i)} f^i(vs_i) + d_i = \sum_{v \in \delta^+(s_i)} f^i(s_i v) & \forall i \in \{1, \dots, k\}, \\
 & \sum_{v \in \delta^-(t_i)} f^i(vt_i) = \sum_{v \in \delta^+(t_i)} f^i(t_i v) + d_i & \forall i \in \{1, \dots, k\}, \\
 & \sum_{i=1}^k f^i(a) \leq b(a) & \forall a \in A, \\
 & f^i(a) \geq 0 & \forall i \in \{1, \dots, k\}, \forall a \in A.
 \end{aligned}$$

Ce problème est donc polynomial par ce PL mais sa taille est bien trop importante pour être traitée par un solveur entier.

Reformulation

On utilise la même reformulation (\tilde{F}) qui est un programme linéaire, dit **formulation arc-chemin**, qui modélise le problème de multiflot fractionnaire

$$\begin{aligned}
 (\tilde{F}) \quad & \text{Min } \sum_{\mu \in \mathcal{P}} c(\mu) f_{\mu} \\
 & \sum_{\mu \in \mathcal{P} \mid a \in \mu} f_{\mu} \leq b(a) \quad \forall a \in A \\
 & \sum_{\mu \in \mathcal{P}_{s_i t_i}} f_{\mu} = d_i \quad \forall i \in \{1, \dots, k\} \\
 & f_{\mu} \geq 0 \quad \forall \mu \in \mathcal{P}.
 \end{aligned}$$

où la notation $c(\mu)$ désigne le coût total des arcs du chemin μ : $c(\mu) = \sum_{a \in \mu} c(a)$.

Si on ajoute en plus la contrainte d'intégrité, on obtient alors une formulation (F) qui est un PLNE modélisant le problème de multiflot entier.

- └ Principe de la génération de colonnes sur un exemple

- └ Le problème de multiflot de coût minimum

Pricing

Comme (\tilde{F}) est un programme linéaire, considérons son dual (\tilde{D}) .

Posons λ_a la variable duale associée à l'inégalité associée à un arc $a \in A$ de (\tilde{F}) ,

et ω_i la variable duale associée à l'inégalité associée à une commodité $i \in \{1, \dots, k\}$.

$$\begin{aligned}
 (\tilde{F}) \quad & \text{Min} \sum_{\mu \in \mathcal{P}} c(\mu) f_{\mu} \\
 & \sum_{\mu \in \mathcal{P} \mid a \in \mu} f_{\mu} \leq b(a) \quad \forall a \in A \\
 & \sum_{\mu \in \mathcal{P}_{s_i; t_i}} f_{\mu} = d_i \quad \forall i \in \{1, \dots, k\} \\
 & f_{\mu} \geq 0 \quad \forall \mu \in \mathcal{P}
 \end{aligned}$$

$$\begin{aligned}
 (\tilde{D}) \quad & \text{Max (fonction objective inutile)} \\
 & \sum_{a \in \mu} \lambda_a + \omega_i \leq c(\mu) \quad \forall i \in \{1, \dots, k\}, \forall \mu \in \mathcal{P}_i \\
 & \lambda_a \leq 0 \quad \forall a \in A \\
 & \omega_i \in \mathbf{R} \quad \forall i \in \{1, \dots, k\}
 \end{aligned}$$

└ Principe de la génération de colonnes sur un exemple

└ Initialisation d'une génération de colonnes

1. Principe de la génération de colonnes sur un exemple

1.1 Le problème de multiflot-max

1.2 Génération de colonnes pour le multiflot-max fractionnaire

1.3 Le problème de multiflot de coût minimum

1.4 Initialisation d'une génération de colonnes

1.5 Résoudre le pricing

1.6 Branchement pour le multiflot entier

2. Définitions et notations

3. Décomposition de Dantzig-Wolfe

4. Application au problème de coloration

Initialisation

Pour appliquer le principe précédent, il faut obtenir un ensemble de chemins \mathcal{P}' tel que la **formulation restreinte** $\tilde{F}_{\mathcal{P}'}$ possède une solution...

C'est l'étape d'initialisation de la génération de colonnes.

Or déterminer une solution réalisable pour ce problème de multiflot n'est pas évident (bien que polynomial). Il y a plusieurs façons de procéder :

- soit déterminer heuristiquement une solution réalisable (mais cela n'est pas garanti de fonctionner)
- soit déterminer heuristiquement beaucoup de chemins (jusqu'à ce que cela soit réalisable)
- soit démarrer "à vide" c'est-à-dire partir d'une formulation avec seulement un chemin par commodité (donc pas forcément réalisable) : dans ce cas, un primal non réalisable correspond à des coûts réduits infinis... On peut par contre utiliser le pricer classique avec des coûts particuliers, dit coût de Farkas, (ils proviennent du Lemme de Farkas (voir cours sur l'approche polyédrale) : ces coûts guident vers une solution réalisable.

└ Principe de la génération de colonnes sur un exemple

└ Résoudre le pricing

1. Principe de la génération de colonnes sur un exemple

1.1 Le problème de multiflot-max

1.2 Génération de colonnes pour le multiflot-max fractionnaire

1.3 Le problème de multiflot de coût minimum

1.4 Initialisation d'une génération de colonnes

1.5 Résoudre le pricing

1.6 Branchement pour le multiflot entier

2. Définitions et notations

3. Décomposition de Dantzig-Wolfe

4. Application au problème de coloration

Pricing et Pricer

Supposons à présent que l'on a trouvé un ensemble \mathcal{P}' de chemins tels que la formulation $\tilde{F}_{\mathcal{P}'}$ ait une solution.

Le problème de pricing est de tester s'il existe un chemin $\mu \in \mathcal{P}$ tel que

$$\sum_{a \in \mu} \lambda_a + \omega_i \leq c(\mu) \quad \forall i \in \{1, \dots, k\}, \forall \mu \in \mathcal{P}_i$$

Une façon efficace de répondre à ce problème est :

- d'effectuer l'étape suivant pour chaque $i \in \{1, \dots, k\}$
- de se rappeler que $c(\mu) = \sum_{a \in \mu} c(a)$

Alors l'inégalité s'écrit

$$\sum_{a \in \mu} (c(a) - \lambda_a) \geq \omega_i$$

où ω_i est une donnée fixe pour chaque itération i .

Pricing et Pricer

Posons alors une longueur auxiliaire associé aux arcs du graphe

$$l_a = c(a) - \lambda_a \quad \forall a \in A$$

Comme $\lambda_a \leq 0$, alors cette longueur est positive !!

Le pricer revient à nouveau ici, pour chaque commodité i , à recherche un plus court chemin entre s_i et t_i selon le poids l_a .

Si cette plus courte longueur est supérieur à d_i , alors il n'existe aucune variable améliorant le problème maître dans \mathcal{P}_i . Sinon on a déterminer une variable à ajouter (on peut en ajouter plusieurs à la fois).

Ce pricer est donc polynomial et la résolution de la formulation arc-chemin pour le problème du multiflot fractionnaire est donc aussi polynomiale (et très efficace).

└ Principe de la génération de colonnes sur un exemple

└ Branchement pour le multiflot entier

1. Principe de la génération de colonnes sur un exemple

1.1 Le problème de multiflot-max

1.2 Génération de colonnes pour le multiflot-max fractionnaire

1.3 Le problème de multiflot de coût minimum

1.4 Initialisation d'une génération de colonnes

1.5 Résoudre le pricing

1.6 Branchement pour le multiflot entier

2. Définitions et notations

3. Décomposition de Dantzig-Wolfe

4. Application au problème de coloration

Problème de multiflot entier

Le problème de multiflot entier est en revanche NP-difficile.

Sa formulation arc-chemin est un PLNE :

$$\begin{aligned}
 (F) \quad & \text{Min} \sum_{\mu \in \mathcal{P}} c(\mu) f_{\mu} \\
 & \sum_{\mu \in \mathcal{P} \mid a \in \mu} f_{\mu} \leq b(a) \quad \forall a \in A \\
 & \sum_{\mu \in \mathcal{P}_{s_j t_j}} f_{\mu} = d_j \quad \forall j \in \{1, \dots, k\} \\
 & f_{\mu} \geq 0 \quad \forall \mu \in \mathcal{P} \\
 & f_{\mu} \in \mathbf{N} \quad \forall \mu \in \mathcal{P}
 \end{aligned}$$

dont la relaxation linéaire est (\tilde{F}) est celle du multiflot fractionnaire.

Donc la relaxation linéaire de ce PLNE peut être résolue en temps polynomial par un algorithme de génération de colonnes.

Il est donc possible d'utiliser cet algorithme de génération de colonnes pour résoudre chacun des nœuds d'un arbre de branchement sur les variables f_{μ} : un tel algorithme est appelé **algorithme de génération de colonnes et branchement** ou **Branch-and-Price algorithm**, en anglais.

Problème de multiflot entier

La règle classique de branchement sur les variables entières :

- sélectionner une variable fractionnaire f_μ parmi celles non-entière de la relaxation \tilde{F} .
- prendre la partie entière inférieure v de f_μ
- créer deux sous-problèmes fils, l'un où $f_\mu \leq v$ et l'autre avec $f_\mu \geq v + 1$

Cette règle n'est pas très efficace pour cette formulation (F) : en effet, il existe de nombreux chemins possibles, certains très proches les uns des autres à une arête près.

De nombreuses règles de branchement plus efficaces ont été tentées... passant souvent par réécrire tout ou partie la formulation pour avoir un branchement efficace.

Une des difficultés est que l'ajout d'inégalités dans le problème maître conduit à créer de nouvelles variables duales à prendre en compte dans le coût dual de la colonne à calculer...

Nous verrons plus loin le cas d'un branchement complet pour le problème de coloration.

1. Principe de la génération de colonnes sur un exemple
2. Définitions et notations
3. Décomposition de Dantzig-Wolfe
4. Application au problème de coloration

Notations

Considérons une formulation PLNE décrite de la façon suivante :

$$\begin{aligned}
 & \text{Min } \sum_{s \in \mathcal{S}} c(s) t_s \\
 (P) \quad & \sum_{s \in \mathcal{S}} \chi_k^s t_s \geq b_k \quad \forall k \in \mathcal{K} \\
 & t_s \in \mathbf{N} \quad \forall s \in \mathcal{S}
 \end{aligned}$$

où

- \mathcal{S} est l'ensemble des variables entières de la formulation (potentiellement en nombre exponentiel).
- $c(s)$ est le coût d'une variables.
- \mathcal{K} est un ensemble d'inégalités (a priori en nombre polynomial).
- pour une contrainte k donnée, χ_s^k est le coefficient de la varuble t_s pour cette inégalité : notez que le vecteur colonne χ_s correspond ainsi à un vecteur décrivant la variable t_s .

Algorithme de génération de colonnes

L'algorithme de génération de colonnes permet de résoudre la relaxation linéaire (\tilde{P}) de (P) peut-être décrit comme suit

- 0- Partir d'un ensemble S_0 de variables tel que la formulation PL (\tilde{P}_0) restreinte à S_0 ait une solution.

$$S^* \leftarrow S_0$$

- 1- Résoudre le PL compact (\tilde{P}^*) restreint aux variables de S^* : soit t^* la solution optimale de (\tilde{P}^*) et λ_k^* le coût dual associé à l'inégalité $k \in \mathcal{K}$
- 2- Tester s'il existe une variable t_s dans $S \setminus S^*$ telle que son **coût réduit** soit strictement négatif (c'est le problème de pricing) :

$$\bar{c}(s) = c(s) - \sum_{k \in \mathcal{K}} \chi_k^s \lambda_k^*$$

- 3- S'il n'en existe pas alors STOP : t^* est solution fractionnaire optimale pour (\tilde{P})
- 4- Sinon $S^* \leftarrow S^* \cup \{t_s\}$ et GOTO 1

Algorithme de Branch-and-Price

Pour résoudre la formulation PLNE (P) il faut utiliser un arbre de branchement où la relaxation linéaire de chacun des PL associés aux nœuds est résolu par l'algorithme de génération de colonnes : c'est **l'algorithme Branch-and-Price**.

Or une génération de colonnes converge parfois lentement.

Néanmoins les algorithmes de Branch-and-Price ont fait leurs preuves pour :

- obtenir des bornes inférieures intéressantes pour un problème (problème d'ordonnancement, de coloration,...)
- obtenir des bonnes solutions avec une garantie expérimentale par différentes méthodes d'approximation, arrondis,...
- et, pour certains problèmes, la génération de colonnes est la meilleure méthode pour obtenir des solutions entières optimales : problème de tournées de véhicules (voir exo TD), de coloration (voir fin de ce cours), problème de découpes,...

Algorithme de Branch-and-Price

Heuristique primale :

Les formulations menant à des génération de colonnes produisent en général de bonnes solutions entières :

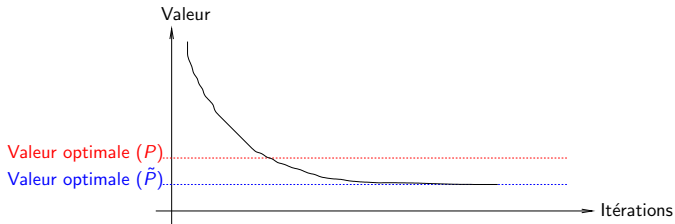
- par arrondi des valeurs fractionnaires
- par “stop-and-solve” : on peut par exemple stopper le déroulement de la génération de colonnes à une itération quelconque et résoudre par Branch-and-Bound la formulation restreinte compacte obtenues en cours de route
- par analyse “statistique” des colonnes générées les plus utilisées dans une solution optimales fractionnaires : dans l'exemple du multiflot c'est repérer les arêtes les plus utilisées par un flot pour construire de manière gloutonne une solution
- ...

Algorithme de Branch-and-Price

Une difficulté : obtenir une borne inférieure rapidement

Une difficulté qui n'apparaît pas en génération d'inégalités (Branch-and-Cut) est que tant que la convergence de la génération de colonnes n'a pas été obtenue, la méthode ne produit pas de borne inférieure :

En effet, regardons ce schéma où la courbe suit l'amélioration de la valeur de la fonction objective pendant une génération de colonnes : on voit que la courbe tend peu à peu vers la solution optimale fractionnaire de la relaxation linéaire (\tilde{P}).



Comme la valeur optimale de (P) est au-dessus de la valeur optimale de \tilde{P} , on ne peut savoir qu'on a obtenu une borne inférieure qu'à la convergence.

Algorithme de Branch-and-Price

Une difficulté : obtenir une borne inférieure rapidement

Or cette convergence est parfois très lente.

Il existe par contre une façon d'obtenir une borne inférieure pour la valeur objective de (P) .

Notons :

- z_P la valeur optimale d'une solution entière de (P)
- $z_{(P)}$ la valeur optimale de la relaxation (\tilde{P})
- z_{iter} la valeur obtenue pour une itération quelconque
- λ_{min}^* la plus petite valeur parmi les k coûts duaux λ_k^* , $k \in \mathcal{K}$
- et B une borne supérieure telle que $B \geq \sum_{s \in \mathcal{S}} t_s$

(Si $t_s \in \{0, 1\}$, la borne B est le nombre maximum de variables dans une solution).

Alors on a

$$z_{iter} + B\lambda_{min}^* \leq z_{\tilde{P}} \leq z_P$$

En effet, par le théorème faible de la dualité, la solution z_{iter} plus que B fois la valeur du plus petit coût dual (qui est négatif) est une borne inférieure de la valeur optimale.

Outils et mise en œuvre

La mise en œuvre des méthodes de Branch-and-Price est assez complexe :

- gestion des variables qui, même en nombre polynomial, deviennent trop importantes pour la résolution des relaxation (suppression en fonction d'une évaluation de leur activité)
- convergence de la génération de colonnes
- utilisation d'un ou plusieurs pricer pour ne pas systématiquement utiliser un pricer trop lent (utilisation de pricer heuristique)
- réaliser des compromis entre résoudre la relaxation linéaire et accélérer le branchement
- règle et stratégie de branchement efficaces...

Si l'on désire résoudre un PLNE avec un nombre exponentiel de variables, il existe des framework

- commerciaux : Concert technology de Cplex-IBM ou Gurobi.
- universitaires : SCIP de la ZIB, BCP de COIN-OR, Symphony de COIN-OR
- un produit universitaire BapCode de l'Université de Bordeaux tente même de mettre en œuvre automatiquement une méthode de décomposition (dite de Dantzig-Wolfe).
- on peut avoir besoin de gérer des formulations avec un nombre exponentiel de variables et de contraintes! Il faut donc un algorithme de **Branch-and-cut-and-Price (BCP)** : il n'existe à pratiquement que les outils universitaires SCIP de la ZIB et BCP de COIN-OR.

1. Principe de la génération de colonnes sur un exemple

2. Définitions et notations

3. Décomposition de Dantzig-Wolfe

3.1 Principe général

3.2 Qualité de la relaxation de Dantzig-Wolfe

4. Application au problème de coloration

1. Principe de la génération de colonnes sur un exemple

2. Définitions et notations

3. Décomposition de Dantzig-Wolfe

3.1 Principe général

3.2 Qualité de la relaxation de Dantzig-Wolfe

4. Application au problème de coloration

Principe général

Soit $c \in \mathbf{R}^n$ un vecteur-colonne, $A \in \mathbf{R}^{n \times m}$, $a \in \mathbf{R}^m$, $B \in \mathbf{R}^{n \times m'}$ et $b \in \mathbf{R}^{m'}$. On considère le PLNE (P) suivant

$$(P) \begin{cases} \text{Min } c^T x \\ Ax \geq a \\ Bx \geq b \\ x \in \mathbf{N}^n \end{cases}$$

où :

- les inégalité $Ax \geq a$ sont "couplantes", c'est-à-dire qu'elles concernent une bonne partie des variables
- les $Bx \geq b$ sont "décomposables".

On dit que l'ensemble d'inégalités $Bx \geq b$ est **décomposable** si le problème

$$(SP) \begin{cases} \text{Min } d^T x \\ Bx \geq b \\ x \in \mathbf{N}^n \end{cases}$$

(avec d ici un poids quelconque) est la juxtaposition de plusieurs problèmes sur des espaces de variables indépendantes.

Principe général

Le problème (SP) peut être reformulé différemment en utilisant une description dite "par les solutions".

On considère pour cela l'ensemble Q des solutions de (SP) : c'est-à-dire tous les points entiers du polyèdre $Bx \geq b$ (cet ensemble est en général exponentiel sur la taille du problème !)

$$Q = \{x \in \mathbf{N}^n \mid Bx \geq b\}$$

Supposons ici que Q soit fini, c'est-à-dire que $\text{conv}(Q)$ soit un polytope : on peut alors noter $Q = \{\chi^1, \dots, \chi^{|Q|}\}$ ces solutions qui sont donc des vecteurs entiers.

Remarquons qu'alors les solutions de Q sont données par

$$\sum_{q=1}^{|Q|} \chi^q t_q \quad \text{avec} \quad \sum_{q=1}^{|Q|} t_q = 1$$

où t_q sont des variables binaires.

Principe général

Déterminer une solution de (SP) revient donc à choisir un vecteur χ dans Q , en d'autre terme cela revient à réécrire (SP) de la façon suivante :

$$(SP) \left\{ \begin{array}{l} \text{Min } d^T x \\ x = \sum_{q=1}^{|Q|} \chi^q t_q \\ \sum_{q=1}^{|Q|} t_q = 1 \\ t_q \in \{0, 1\} \quad \forall q \in \{1, \dots, |Q|\}. \end{array} \right.$$

Cette écriture "naïve" revient à énumérer toutes les solutions et donner à x le vecteur la meilleure d'entre elles : on peut noter que les variables t sont ici des valeurs binaires permettant de définir quels vecteurs choisir dans Q .

Principe général

Remarquons que l'on pourrait ici prendre des variables t réelles pour définir (SP) : on appelle cela la convexification.

$$(SP) \left\{ \begin{array}{l} \text{Min } d^T x \\ x = \sum_{q=1}^{|Q|} \chi^q t_q \\ \sum_{q=1}^{|Q|} t_q = 1 \\ t_q \geq 0 \quad \forall q \in \{1, \dots, |Q|\}. \end{array} \right.$$

En effet, le polyèdre associé est naturellement entier car chacun de ses points extrêmes correspond à exactement à un élément de Q (c'est-à-dire une solution de (SP)).

Principe général

Utilisons cette reformulation de (SP) dans (P) :

$$(P) \left\{ \begin{array}{l} \text{Min } c^T x \\ Ax \geq a \\ x = \sum_{q=1}^{|Q|} \chi^q t_q \\ \sum_{q=1}^{|Q|} t_q = 1 \\ t_q \in \{0, 1\} \quad \forall q \in \{1, \dots, |Q|\}. \end{array} \right.$$

Et faisons alors disparaître x !

$$(P) \left\{ \begin{array}{l} \text{Min } \sum_{q=1}^{|Q|} (c^T \chi^q) t_q \\ \sum_{q=1}^{|Q|} (A \chi^q) t_q \geq a \\ \sum_{q=1}^{|Q|} t_q = 1 \\ t_q \in \{0, 1\} \quad \forall q \in \{1, \dots, |Q|\}. \end{array} \right.$$

Principe général

On définit la **Relaxation de Dantzig-Wolfe** comme la relaxation fractionnaire (MP) de (P) ainsi définie :

$$(MP) \left\{ \begin{array}{l} \text{Min } \sum_{q=1}^{|Q|} (c^T \chi^q) t_q \\ \sum_{q=1}^{|Q|} (A \chi^q) t_q \geq a \\ \sum_{q=1}^{|Q|} t_q = 1 \\ t_q \geq 0 \end{array} \quad \forall 1 \in \{1, \dots, |Q|\}.$$

Cette relaxation n'est pas la relaxation linéaire classique de (P).

Il est important de remarquer que (MP), appelé *problème maître*, est un PL mais où le fait que les solutions de (SP) soit entière a été en partie pris en compte.

Principe général

La formulation (MP) contient par construction un nombre exponentiel de variables autant que de solutions dans Q !

Afin de résoudre (MP), on peut alors mettre en place un algorithme de génération de colonnes.

En posant π le vecteur dual des inégalités provenant de $Ax \geq a$ et ρ la valeur duale provenant de la contrainte d'égalité, le problème de pricing revient à rechercher s'il existe une solution de $\chi^q \in Q$ telle que son coût réduit est négatif, *i.e.*,

$$(c - \pi^T A)\chi^q - \rho < 0$$

Ainsi le pricer revient à résoudre le problème (SP) avec le coût $d = (c - \pi^T A)$: si ce poids est négatif strictement, on ajoute une colonne χ^q à Q . Et on réitère cela jusqu'à ce que (SP) renvoie une plus petite solution de coût réduit positif : ce qui signifie que la solution du master restreint est une solution optimale de (MP).

Principe général

Le pricer peut donc se résoudre par PLNE en utilisant :

$$(SP) \begin{cases} \text{Min } (c - \pi^T A)^T x \\ Bx \geq b \\ x \in \mathbf{N}^n \end{cases}$$

On peut alors voir (SP) comme le sous-problème de la méthode de Dantzig-Wolfe.

Bien entendu, il est tout à fait possible de résoudre le problème de pricing par d'autres moyens algorithmiques, surtout s'ils sont plus efficaces.

1. Principe de la génération de colonnes sur un exemple

2. Définitions et notations

3. Décomposition de Dantzig-Wolfe

3.1 Principe général

3.2 Qualité de la relaxation de Dantzig-Wolfe

4. Application au problème de coloration

Relaxation de Dantzig-Wolfe

Comme l'on cherche à résoudre un PLNE (P), cette génération de colonnes ne mène qu'à une relaxation fractionnaire (MP) de ce PLNE, dite **relaxation de Dantzig-Wolfe**.

En général, cette relaxation (MP) est meilleure que la relaxation linéaire du PLNE (P)!

De nombreuses bornes ont été fournies par cette méthode pour des problèmes d'optimisation combinatoire.

De plus un algorithme de Branch-and-Price basée sur cette décomposition de Dantzig-Wolfe est souvent efficace.

Amélioration de la relaxation linéaire ?

En fait, si la relaxation (MP) de Dantzig-Wolfe est plus intéressante en général que la relaxation linéaire de (P), il existe de nombreux cas où les deux relaxations sont identiques !

C'est le cas dans ce cadre très fréquent :

Theorem (Geoffrion)

Si (SP) est entièrement décrit par $\text{conv}(x \in \mathbf{R}^+ \mid Bx \geq b)$, c'est-à-dire si cette enveloppe convexe est entière, alors les deux relaxations linéaires et de Dantzig-Wolfe sont identiques.

Ce résultat se déduit de l'écriture de (P) où l'on peut remplacer (SP) par un polyèdre dont chacun des sommets est une de ses solutions.

On peut interpréter ce résultat en disant que, dans ce cas, la décomposition de Dantzig-Wolfe sur (SP) n'apporte pas rien sur l'intégrité de (SP).

Amélioration de la relaxation linéaire ?

A partir du résultat précédent, on peut noter que, dans tous les cas où le problème de pricing d'une génération de colonnes est polynomial, alors la relaxation linéaire de la génération de colonnes est la même qu'une version non décomposée.

Dans le cas de la relaxation du multiflot entier, ce sera bien la même valeur !

Par contre, lorsque le pricing est NP-difficile, alors on constate le plus souvent une forte amélioration !

Dantzig-Wolfe versus décompo lagrangienne

On peut se demander quelle relaxation est la plus intéressante entre la relaxation de Dantzig-Wolfe et la relaxation lagrangienne.

Theorem

Les relaxations de Dantzig-Wolfe et la relaxation lagrangienne ont même valeurs.

Preuve : En effet, la relaxation lagrangienne de (P) est

$$(LRP) \left\{ \begin{array}{l} \text{Min } c^T x - \gamma^T (Ax - a) \\ x = \sum_{q=1}^{|Q|} \chi^q t_q \\ \sum_{q=1}^{|Q|} t_q = 1 \\ t_q \in \{0, 1\} \end{array} \quad \forall q \in \{1, \dots, |Q|\}$$

avec γ le vecteur colonne optimal des multiplicateurs de Lagrange.

Dantzig-Wolfe versus décompo lagrangienne

On définit le dual lagrangien de la manière suivante

$$(LDP) \left\{ \begin{array}{l} \max_{\gamma \in R_+^{|a|}} \text{Min } c^T x - \gamma^T (Ax - a) \\ x \in Q \end{array} \right.$$

On peut linéariser le min en utilisant Q l'ensemble des solutions de (SP) .

$$(LDP) \left\{ \begin{array}{l} \text{Max } \eta \\ \eta \leq c^T x^q - \gamma^T (Ax^q - a) \quad \forall q \in \{1, \dots, |Q|\} \\ \gamma \in R_+^{|a|} \\ \eta \in R \end{array} \right.$$

En notant λ_q les variables duales associées aux contraintes du dual lagrangien, on peut écrire son dual de la façon suivante (en ajoutant sans perte de généralité la normalisation des valeurs λ .)

$$(DLDP) \left\{ \begin{array}{l} \text{Min } \sum_{q=1}^{|Q|} (c^T x^q) \lambda_q \\ \sum_{q=1}^{|Q|} (Ax^q) \lambda_q \geq a \\ \sum_{q=1}^{|Q|} \lambda_q = 1 \\ \lambda_q \geq 0 \quad \forall q \in \{1, \dots, |Q|\} \end{array} \right.$$

On obtient la relaxation (MP) dans l'espace de variable correspondant à la décomposition de Dantzig-Wolfe. Donc le dual de la formulation par décomposition de Dantzig-Wolfe correspond au dual lagrangien. \square

1. Principe de la génération de colonnes sur un exemple
2. Définitions et notations
3. Décomposition de Dantzig-Wolfe
4. Application au problème de coloration

Problème de coloration

Soit $S \subseteq \mathbf{N}$. Une *coloration* des sommets d'un graphe $G = (V, E)$ est une fonction $r : V \rightarrow S$ telle que $r(u) \neq r(v)$ pour tout couple de sommets adjacents u, v . Les éléments de l'ensemble S sont appelés les *couleurs* disponibles. Une *k-coloration* est une coloration $c : V \rightarrow \{1, \dots, k\}$. 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 $k \geq 3$, et il est polynomial si $k = 2$. En effet, si $k = 2$, il suffit de tester si le graphe est biparti, c'est-à-dire s'il ne contient pas des cycle impair. Ce qui peut se faire par un simple parcours de graphe.

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*.

Première formulation

On associe à chaque sommet u de V un vecteur binaire à K dimensions $x_u = (x_u^1, \dots, x_u^K)$, où K est une borne supérieure sur la coloration de G (au maximum $K = |V|$).

Comme l'on désire minimiser le nombre de couleur, on ajoute une variable binaire w^l par couleur $l = 1, \dots, K$ indiquant si cette couleur a été utilisée ou non.

Le problème est donc équivalent au programme

$$(P_C) \left\{ \begin{array}{ll} \text{Min} & \sum_{l=1}^K w^l \\ & \sum_{l=1}^K x_u^l = 1 \quad \forall u \in V, \\ & x_u^l + x_v^l \leq w^l \quad \forall e = uv \in E \text{ et } 1 \leq l \leq K, \\ & x_u^l \in \{0, 1\} \quad \forall u \in V \text{ et } 1 \leq l \leq K, \\ & w^l \in \{0, 1\} \quad \forall 1 \leq l \leq K. \end{array} \right.$$

Les premières contraintes sont celles couplante ($Ax \leq a$) et les suivantes celles décomposables par "couleur l " ($Bx \geq b$).

Décomposition

Ainsi le sous-problème est

$$(SP_C) \left\{ \begin{array}{ll} \text{Min } \sum_{l=1}^K (d_l^0 w^l + d_l x^l) & \\ x_u^l + x_v^l \leq w^l & \forall e = uv \in E \text{ et } 1 \leq l \leq K \\ x_u^l \in \{0, 1\} & \forall u \in V \text{ et } 1 \leq l \leq K \\ w^l \in \{0, 1\} & \forall 1 \leq l \leq K \end{array} \right.$$

qui est en fait la somme des PLNE pour chaque couleur $l \in \{1, \dots, K\}$

$$(SP_C^l) \left\{ \begin{array}{ll} \text{Min } d_l^0 w^l + d_l x^l & \\ x_u^l + x_v^l \leq w^l & \forall e = uv \in E \\ x_u^l \in \{0, 1\} & \forall u \in V \\ w^l \in \{0, 1\}. & \end{array} \right.$$

Le problème est donc décomposé en K sous-problèmes (SP_C^l) : un par couleur l .

Décomposition

On peut voir que le problème (SP_C) consiste donc à déterminer K stables !

Un vecteur solution appartenant à l'ensemble Q des solutions est donc un vecteur représentant K stables et pour chaque stable l un entier w^l indiquant si ce stable l est ou non l'ensemble vide.

Posons alors \mathcal{S} l'ensemble des stables G et $\chi^{\mathcal{S}}$ l'ensemble des vecteurs d'incidence de ces stables. Alors Q peut être défini comme un vecteur composé de toutes les combinaisons de K vecteurs $\chi^S \in \chi^{\mathcal{S}}$ chapeauté chacun de 0 si $S = \emptyset$ et 1 sinon.

Suivant le principe de Dantzig-Wofle, (SP_C^l) peut aussi s'écrire

$$(SP_C^l) \left\{ \begin{array}{ll} \text{Min } d_l^0 w^l + d_l x^l & \\ x_u^l + x_v^l \leq w^l & \forall e = uv \in E \\ x^l = \sum_{s \in \mathcal{S}} \chi^s t_s^l & \\ \sum_{s \in \mathcal{S}} t_s^l = 1 & \\ t_s^l \in \{0, 1\} & \forall s \in \mathcal{S}. \end{array} \right.$$

Décomposition

On peut remarquer que si $d_l^0 > 0$ et pour une solution (entière) de (SP_C^l) , on a en fait $w^l = \sum_{s \in \mathcal{S}} t_s^l$.

En effet, en considérant uniquement des vecteurs d'incidence de stables, la contrainte sur les arêtes n'a qu'au plus un des terme de gauche non nul. De plus, dans une solution optimale, w_l sera donc 0 ou 1. En fait, w_l vaudra 0 si la couleur l n'est attribué à aucun stable et 1 sinon : d'où le resultat.

On peut alors écrire

$$(SP_C^l) \left\{ \begin{array}{l} \text{Min } d_l^0 \sum_{s \in \mathcal{S}} t_s^l + d_l x^l \\ x^l = \sum_{s \in \mathcal{S}} \chi^s t_s^l \\ \sum_{s \in \mathcal{S}} t_s^l = 1 \\ t_s^l \in \{0, 1\} \end{array} \right. \quad \forall s \in \mathcal{S}.$$

Décomposition

En utilisant cette écriture de (SP) , on peut réécrire (P_C) de la manière suivante

$$(P_C) \left\{ \begin{array}{l} \text{Min } \sum_{l=1}^K \sum_{s \in \mathcal{S}} t_s^l \\ \sum_{l=1}^K x_u^l = 1 \quad \forall u \in V, \\ x^l = \sum_{s \in \mathcal{S}} \chi^S t_s^l \\ \sum_{s \in \mathcal{S}} t_s^l = 1 \\ t_s^l \in \{0, 1\} \quad \forall s \in \mathcal{S} \text{ et } 1 \leq l \leq K \end{array} \right.$$

Décomposition

On peut noter ici que toutes les variables et vecteurs ne dépendent pas de I !!!

On peut donc réécrire de la façon suivante

$$(P_C) \left\{ \begin{array}{l} \text{Min } \sum_{s \in \mathcal{S}} t_s \\ \sum_{s \in \mathcal{S}} \chi_u^s t_s = 1 \quad \forall u \in V, \\ t_s \in \{0, 1\} \quad \forall s \in \mathcal{S} \end{array} \right.$$

Cette écriture est une formulation, bien classique et bien compréhensible pour le problème de coloration !

$$(P_C) \left\{ \begin{array}{l} \text{Min } \sum_{s \in \mathcal{S}} t_s \\ \sum_{s \in \mathcal{S} \mid u \in s} \chi_u^s t_s = 1 \quad \forall u \in V, \\ t_s \in \{0, 1\} \quad \forall s \in \mathcal{S} \end{array} \right.$$

Génération de colonnes pour la coloration

On peut alors considérer la relaxation linéaire (MP_C) de ce programme

$$(MP_C) \left\{ \begin{array}{l} \text{Min } \sum_{s \in \mathcal{S}} t_s \\ \sum_{s \in \mathcal{S}} \chi_u^s t_s = 1 \quad \forall u \in V, \\ t_s \geq 0 \quad \forall s \in \mathcal{S} \end{array} \right.$$

Considérons un sous-ensemble \mathcal{S}^* de stables tels qu'il existe une coloration avec ces stables (on peut aisément obtenir une telle initialisation).

Notons (MP_C^*) la relaxation linéaire de (P_C) restreinte à \mathcal{S}^* . Notons t^* la solution optimale t^* de ce programme restreint (MP_C^*) et π_v les coûts duaux associés à ses inégalités.

Alors t^* est la solution optimale de (MP_C) si et seulement si

$$(1 - \sum_{v \in V} \pi_v \chi_v^s) \geq 0 \quad \forall s \in \mathcal{S}$$

c'est-à-dire s'il n'existe pas de stable s tel que $\sum_{v \in \mathcal{S}} \pi_v < 1$.

Génération de colonnes pour la coloration

Le problème de pricing de génération de colonnes revient ainsi à rechercher un stable de poids maximum selon les coûts duaux λ_v , $v \in V$. Si ce stable maximal est plus grand stable est de poids < 1 alors on a trouvé un colonne à ajouter à (MP) sinon on a atteint la relaxation linéaire de (MP) .

La borne obtenue est une des meilleures bornes inférieures connues pour le problème de coloration.

De plus, en utilisant cette génération de colonnes dans un algorithme de Branch-and-Price, on a pu obtenir des solutions optimales (entières) pour de nombreuses instances difficile.

En effet, le principe de branchement pour la formulation (MP) est assez simple à mettre en œuvre : on peut utiliser la règle de Ryan-And-Foster qui propose de créer deux fils à partir d'un couple de sommet (u, v) non adjacent du graphe : un fils où u et v ont même couleur et un fils où u et v sont de couleurs différentes. L'astuce de mise en œuvre repose sur le fait que pour un fils, on identifie les deux sommets dans le graphe et pour l'autre, on ajoute une arête : ainsi à chaque profondeur de l'arbre, le graphe étudié est simplifié.