

# Cours n°9: Relaxation Lagrangienne, décomposition de Dantzig-Wolfe, algorithmes de génération de colonnes et Branch&Price

Nicolas DUPIN

<https://github.com/ndupin/ORteaching>  
<http://nicolasdupin2000.wixsite.com/research>

version du 20 mars 2022

Cours distribué sous licence CC-BY-NC-SA

Issu et étendu d'enseignements donnés à l'ENSTA Paris, Polytech'  
Paris-Saclay, et à l'université Paris-Saclay

# Rappel : algorithme Branch&Bound (B&B) en PLNE

Pour un problème de minimisation, les principes de l'algorithme de Branch&Bound (ou séparation-évaluation) guidés par la relaxation PL :

- ▶ La résolution continue, résolution PL, donne des bornes inférieures.
- ▶ élagage : un noeud est élagué (inutile à explorer) si sa borne inférieure est supérieure à (au moins aussi bonne que) la meilleure solution connue.
- ▶ Branchements : restreindre les valeurs possibles d'une variable, définit un parcours arborescent (arbre binaire) ,
- ▶ Branchements et bornes : ajouter des contraintes linéaires, préserve la structure PLNE, on peut utiliser les bornes PL à chaque noeud. Cela augmente la borne inférieure.

⇒ Algorithme B&B convergeant vers une solution optimale, ou en temps arrêté fournit majoration et minoration de la valeur optimale.

# Rappels : Éléments clés pour l'efficacité pratique B&B

Avoir plus rapidement une bonne solution (primale) peut permettre de supprimer des noeuds plus tôt :

Améliorer les bornes inférieures à un noeud, peut permettre de supprimer des noeuds plus tôt :

- ▶ Le travail de modélisation PLNE influe sur la qualité des relaxations continues.
- ▶ Ajout de coupes d'intégrité : contraintes valides sur les points entiers permettant de couper des solutions fractionnaires.

Avoir plus rapidement une bonne solution (primale) peut permettre de supprimer des noeuds plus tôt :

- ▶ Heuristiques génériques pour chercher des bonnes solutions à chaque noeud.
- ▶ Démarrage à chaud : fournir initialement une bonne solution (primale) accélère la résolution PLNE. Une heuristique spécialisée en amont de la PLNE par exemple.

⇒ Compromis à trouver entre l'allongement des temps de calculs à chaque noeud, et le temps gagné à énumérer moins de noeuds.

# Motivation du cours d'aujourd'hui

- ▶ Une dernière question en suspens : comment gérer une formulation PLNE avec un nombre de variables trop grand pour être énuméré en mémoire ? Réponse : l'algorithme de génération de colonnes
- ▶ ex : Problème de coloration des sommets d'un graphe (Vertex Coloring)
- ▶ Comment améliorer les bornes duales de relaxation continue pour un PLNE ? Réponse : la relaxation Lagrangienne et la décomposition de Dantzig-Wolfe, un calcul de bornes Lagrangiennes.
- ▶ Une dernière recherche arborescente : Branch&Price.

# Plan

Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

Applications pratiques, résultats

- Cutting Stock Problem (CSP)

- Problème de coloration d'un graphe "Vertex coloring"

- Problèmes de tournées de véhicules (VRP)

- Problèmes de clustering

- Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions

# Plan

## Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

Applications pratiques, résultats

- Cutting Stock Problem (CSP)

- Problème de coloration d'un graphe "Vertex coloring"

- Problèmes de tournées de véhicules (VRP)

- Problèmes de clustering

- Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions

# Relaxation lagrangienne

On considère un PLNE sous la forme :

$$\begin{aligned} OPT = \min_{x \in \mathbb{N}^n} \quad & cx \\ & Ax \geq a \\ & Bx \geq b \end{aligned} \tag{1}$$

Où les contraintes  $Bx \geq b$  sont dites “faciles”, la résolution de problèmes sous contraintes  $Bx \geq b$  est bien plus facile que le problème complet.

Exemples :

- on a un algorithme spécifique pour résoudre  $\min_{x \in \mathbb{N}^n: Bx \geq b} c'x$
- $\min_{x \in \mathbb{N}^n: Bx \geq b} c'x$  définit des sous-problèmes indépendants

On peut alors calculer une borne inférieure  $\min_{x \in \mathbb{N}^n: Bx \geq b} c'x \leq OPT$

Dans la suite, on généralise en dualisant les contraintes difficiles  $Ax \geq a$

Les contraintes difficiles  $Ax \geq a$  sont dites couplantes ou liantes

# Dualisation lagrangienne, borne(s) lagrangienne(s)

Par dualisation Lagrangienne :

$$OPT = \min_{\substack{x \in \mathbb{N}^n \\ Bx \geq b}} \max_{\lambda \geq 0} cx + \lambda(a - Ax) \quad (2)$$

Par interversion min-max (ou dualité faible ici) :

$$OPT \geq \max_{\lambda \geq 0} \min_{\substack{x \in \mathbb{N}^n \\ Bx \geq b}} cx + \lambda(a - Ax) \quad (3)$$

Pour tout  $\lambda \geq 0$ , on a UNE borne  $I(\lambda)$  dite lagrangienne :

$$OPT \geq I(\lambda) = \min_{\substack{x \in \mathbb{N}^n \\ Bx \geq b}} cx + \lambda(a - Ax) \quad (4)$$

LA borne lagrangienne est  $\max_{\lambda \geq 0} I(\lambda)$



# Calcul de la Borne lagrangienne

La/des borne(s) Lagrangienne(s) peut/peuvent se calculer par :

- ▶ Décomposition de Dantzig Wolfe et algorithme de génération de colonnes : développé dans la suite
- ▶ Méthode des faisceaux : plans coupants type Branch&Cut
- ▶ Algorithmes de type gradient : rudimentaire
- ▶ Méthode des volumes

O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot, F. Vanderbeck, *Comparison of bundle and classical column generation* Mathematical Programming June 2008, Volume 113, Issue 2, pp 299-344

Shibasaki, R. S., Baïou, M., Barahona, F., Mahey, P., & de Souza, M. C. (2021). Lagrangian bounds for large-scale multicommodity network design : a comparison between Volume and Bundle methods. *International Transactions in Operational Research*, 28(1), 296-326.

# Idée générale commune

Pour tout  $\lambda \in \mathbb{R}_+^m$ , on calcule une borne Lagrangienne avec :

$$\begin{aligned} OPT &\geq I(\lambda) = \min_{x \in \mathbb{N}^n} cx + \lambda(a - Ax) \\ \text{s.c :} \quad & Bx \geq b \end{aligned} \tag{5}$$

On se ramène à calculer pour des valeurs différentes de  $\lambda$  des  $I(\lambda) = \min_{x \in \mathbb{N}^n: Bx \geq b} c'x \leq OPT$ , calculs plus faciles que le problème initial.

N.B : Pour  $\lambda = 0$ , on retrouve la borne initiale  $\min_{x \in \mathbb{N}^n: Bx \geq b} cx \leq OPT$

La borne Lagrangienne  $\max_{\lambda \geq 0} I(\lambda)$  peut se calculer (ou s'approcher) par un schéma numérique d'optimisation sans contrainte pourtant sur les variables  $\lambda \in \mathbb{R}_+^m$ , où on évalue des valeurs de  $I(\lambda)$  par un calcul de type  $\min_{x \in \mathbb{N}^n: Bx \geq b} c'x$

# Théorème de Geoffrion

## Théorème (Théorème de Geoffrion)

*La borne Lagrangienne est toujours meilleure que la borne de relaxation continue :*

$$\begin{array}{lll} \min_x cx & \leq \max_{\lambda \geq 0} l(\lambda) \leq & \min_x c.x \\ Ax \geq a & & Ax \geq a \\ Bx \geq b & & Bx \geq b \\ x \geq 0 & & x \in \mathbb{N}^m \times \mathbb{R}_+^p \end{array} \quad (6)$$

On va en fournir une interprétation polyédrale et une démonstration par la décomposition de Dantzig-Wolfe

Geoffrion, A.M., 1974. Lagrangean relaxation for integer programming. In Balinski, M.L. (ed.) *Approaches to Integer Programming*, Mathematical Programming Studies, Vol. 2. Springer, Berlin-Heidelberg, pp. 82-114

# Structure matricielle de décomposition

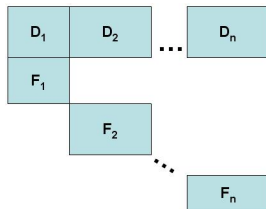
Structure matricielle usuelle :

$$x = (x_1, \dots, x_K)$$

$$cx = \sum_{k=1}^K c_k x_k$$

$$Ax \geq a : \sum_{k=1}^K D_k x_k \geq a$$

$$Bx \geq b : F_k x_k \geq b_k, \forall k \in \{1, \dots, K\}$$



Les contraintes  $Bx \geq b$  forment  $K$  ensembles de contraintes indépendants

Les calculs de bornes Lagrangiennes se ramènent à  $K$  sous-problèmes indépendants :

$$l(\lambda) = \min_{x \in \mathbb{N}^n : Bx \geq b} c'x = \sum_{k=1}^K \min_{x_k \in \mathbb{N}^{n(k)} : F_k x_k \geq b_k} c'_k x_k$$

# Structure matricielle de décomposition : exemples

- ▶ Unit Commitment/production d'électricité : les contraintes de demandes à tout instant sont des contraintes couplantes, les sous blocs indépendants définissent les contraintes techniques de fonctionnement pour chaque centrale électrique.
- ▶ Tournées de véhicules : Les contraintes liant les véhicules sont que toutes les livraisons sont partagées parmi les véhicules. Les sous blocs indépendants définissent les contraintes liées à chaque tournée de véhicule : respect des fenêtres de temps, contraintes techniques liées au véhicule.

# Multiplicité de sous problèmes

Structure matricielle usuelle :

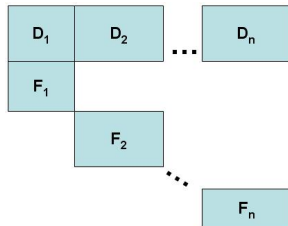
$$x = (x_1, \dots, x_K)$$

$$cX = \sum_{k=1}^K c_k x_k$$

$$Ax \geq a : \sum_{k=1}^K D_k x_k \geq a$$

$$Bx \geq b : F_k x_k \geq b_k, \forall k \in \{1, \dots, K\}$$

$$I(\lambda) = \min_{x \in \mathbb{N}^n : Bx \geq b} c'x = \sum_{k=1}^K \min_{x_k \in \mathbb{N}^{n(k)} : F_k x \geq b_k} c'_k x_k$$



Si on a des sous-problèmes multiples : ie  $c_k = c_{k'}$ ,  $D_k = D_{k'}$  et  $F_k = F_{k'}$ , on résout le même problème d'optimisation deux fois prédemment !

Multiplicité d'un sous-problème : nombre de sous problème équivalents. Si les  $K$  sous-problèmes sont partitionnés en  $K' \leq K$  sous-pb, le sous-pb  $k$  ayant une multiplicité  $m(k)$ , on n'a que  $K'$  sous-problèmes à résoudre et :

$$I(\lambda) = \min_{x \in \mathbb{N}^n : Bx \geq b} c'x = \sum_{k=1}^{K'} m(k) \times \min_{x_k \in \mathbb{N}^{n(k)} : F_k x \geq b_k} c'_k x_k$$

# Plan

Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

Applications pratiques, résultats

- Cutting Stock Problem (CSP)

- Problème de coloration d'un graphe "Vertex coloring"

- Problèmes de tournées de véhicules (VRP)

- Problèmes de clustering

- Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions

# Cadre d'application

- Cadre de Programmation linéaire : PAS DE PLNE pour la génération de colonne!!!
- Application pour un PL, où le nombre de colonnes est très grand devant le nombre de contraintes.
- On veut éviter de manipuler explicitement toutes les variables et de générer une matrice de contraintes de taille potentiellement très grande.
- $A \in \mathcal{M}^{m,n}(\mathbb{R})$ ,  $x, c \in \mathcal{M}^{n,1}(\mathbb{R})$ ,  $b \in \mathcal{M}^{m,1}(\mathbb{R})$ . Et  $m \ll n$

$$\begin{array}{ll} \min & c^T \cdot x \\ \text{s.c :} & A \cdot x \geq b \\ & x \geq 0 \end{array} \quad (7)$$



# Principes généraux

- ▶ Dans la résolution d'un tel PL, la majorité des variables est hors base et donc nulle sur les points extrêmes.
- ▶ Idée : On considère le problème restreint à un nombre de variables raisonnable, et on génère dynamiquement les variables selon le besoin.
- ▶ A chaque étape, on a une solution réalisable du problème en complétant avec des variables nulles, même coût.

# Coût réduit

- Considérant le PL suivant :

$$\begin{aligned} \min & c^T \cdot x \\ \text{s.c : } & A \cdot x \geq b \quad (\lambda) \\ & x \geq 0 \end{aligned} \tag{8}$$

- Le coût réduit associé à la variable  $x_i$  est défini par :

$$CR(x_i) = c_i - \sum_{j=1}^m \lambda_j A_{i,j}$$

# Algorithme de Génération de Colonnes (GC)

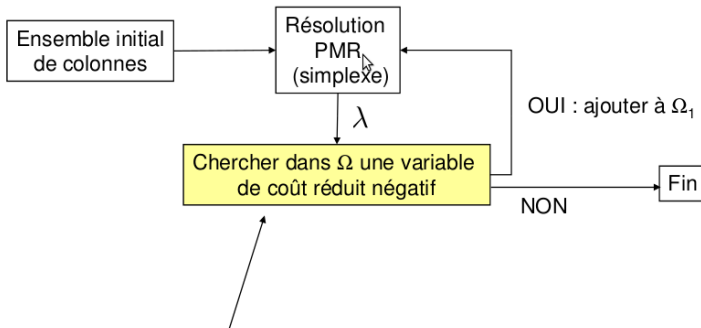
- ▶ Calcul sur problème restreint en variables. (problème maître restreint, PMR)
- ▶ Existe t'il une variable de cout réduit est strictement négatif ?  
De manière équivalente, a t'on  $\min_i CR(x_i) < 0$  ?
  - ▶ Si non, la solution restreinte est optimale sur le problème complet
  - ▶ Si oui, on rajoute la variable au PMR et on itère ...

Le problème  $\min_i CR(x_i) < 0$  est dit *problème de pricing*

N.B : on peut interpréter cela comme une généralisation de l'algorithme de simplexe, le coût réduit apparaissait dans l'algorithme de simplexe.

N.B : Pour l'implémentation, on peut utiliser un solveur PL sur le PMR, et le démarrage à chaud de simplexe primal en ajoutant une variable à la solution optimale courante du PMR.

# Algorithme de génération de colonne



**Sous-problème**

(ou problème esclave, ou oracle, ou problème de pricing)

# Pourquoi le coût réduit ?

On calcule le dual dans le problème complet :

$$\begin{aligned} \max_{\lambda \geq 0} \quad & \sum_{j=1}^m b_j \lambda_j \\ \text{s.c. : } \forall i \in \llbracket 1, n \rrbracket \quad & \sum_{j=1}^M A_{i,j} \lambda_j \leq c_i \end{aligned} \tag{9}$$

Une violation de contrainte s'écrit pour  $i \in \llbracket 1, n \rrbracket$  :

$$\sum_{j=1}^m \lambda_j A_{i,j} > c_i \iff CR(x_i) = c_i - \sum_{j=1}^m \lambda_j A_{i,j} < 0$$

On a retrouvé ici le coût réduit.

L'algorithme de génération de colonne écrit précédemment n'est ni plus ni moins qu'une génération de coupes dans le dual

Minimiser le coût réduit, c'est maximiser la violation de contrainte dans le dual.

# Théorème de Lovasz-Schrijver-Grötschel

Rappel, Théorème fondamental de l'analyse polyédrale :

## Théorème (Théorème de Lovasz-Schrijver-Grötschel)

*Pour un Programme Linéaire, si il existe un algorithme de séparation polynomial, alors le calcul du PL est aussi polynomial.*

Grötschel, M., Lovász, L., & Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2), 169-197.

On est en PL, cela s'applique au dual avec dualité forte, transformant la génération de coupes en génération de colonnes :

Sur un PL réalisable avec un nombre non énumérable de variables, la résolution du PL est polynomiale s'il existe un algorithme de pricing polynomial

# Illustration sur le problème de découpe (CSP)

- Premier cas historique, étudié par Gilmore et Gomory.
- On a des rouleaux de papier de longueur  $L$ , dont on doit découper  $M$  types de motifs.
- Un motif  $m \in \llbracket 1, M \rrbracket$  a une longueur  $l_m$  et on doit en couper  $d_m$ .
- But : déterminer la découpe, de manière à minimiser le nombre total de grand rouleaux à utiliser.

# Formulation du problème de découpe de rouleaux de papier

- Soit  $\mathcal{P}$  l'ensemble des patterns de découpes réalisables.
- Variables :  $x_p \in \mathbb{N}$  pour  $p \in \mathcal{P}$ , nombre de fois que le pattern  $p$  est choisi.
- Soit  $a_{m,p}$  le nombre de fois que le motif  $m$  est présent dans le pattern  $p$ .

$$\begin{aligned} \min_{x_p \in \mathbb{N}} \quad & \sum_{p \in \mathcal{P}} x_p \\ \text{s.c : } \forall m \in \llbracket 1, M \rrbracket \quad & \sum_{p \in \mathcal{P}} a_{m,p} x_p \geq d_m \end{aligned} \tag{10}$$

On calcule la relaxation continue par génération de colonne.



# Problème maître restreint

Soit  $\mathcal{P}^o$  un sous ensemble de  $\mathcal{P}$ . Le *problème maître restreint* est le problème où toutes les variables non comprises dans  $\mathcal{P}^o$  sont nulles.

$$\begin{aligned} \min_{x \geq 0} \quad & \sum_{p \in \mathcal{P}^o} x_p \\ \text{s.c : } \forall m \in \mathcal{M} \quad & \sum_{p \in \mathcal{P}^o} a_{m,p} x_p \geq d_m \quad (\lambda_m) \end{aligned} \tag{11}$$

Questions :

- Comment savoir si la solution optimale de ce problème restreint est la solution optimale du problème complet ?
- Si ce n'est pas le cas, comment savoir quel pattern rajouter ?

# Problème esclave, génération d'un "pattern"

- ▶ Une colonne  $p$  doit être ajoutée si  $1 - \sum_m a_{m,p} \lambda_m < 0$ .
- ▶ Mise sous forme d'un problème d'optimisation :
$$\eta^* = \min_{p \in \mathcal{P}} 1 - \sum_m a_{m,p} \lambda_m.$$
- ▶ Si  $\eta^* < 0$ , on rajoute la variable correspondant au pattern solution.
- ▶ Sinon, l'optimalité du problème maître restreint est prouvée.

# Reformulation du problème esclave

- Formulation par motifs  $x_m \in \mathbb{N}$  le nombre de fois que l'on emploie le motif  $m$ .

$$\begin{aligned} \min_{x \geq 0} \quad & 1 - \sum_{m=1}^M \lambda_m x_m \\ \text{s.c :} \quad & \sum_{m=1}^M l_m x_m \leq L \end{aligned} \tag{12}$$

- Problème de sac à dos à valeurs entières et non binaires.
- Résolution par programmation dynamique, algo pseudo-polynomial.

# Points d'implémentation

- ▶ Partir d'une solution réalisable, sinon, premier pb maître infaisable (éventuellement ajouter des pénalisations).
- ▶ Arrondis numériques : colonnes ajoutées si coût réduit  $\leq -\varepsilon$ .
- ▶ Si le pb maître devient trop gros, il peut être utile d'enlever des colonnes n'ayant pas été utilisées depuis plusieurs itérations.
- ▶ implémentation OPL avec OPL script : cutting-stock.mod, en appelant une résolution PLNE du sous-problème par le Branch&Bound de Cplex
- ▶ exemple implémentation avec OPL appelé depuis du code C++, à la manière d'OPL script fourni également dans les exemples de Cplex.

⇒ Exercice : remplacer la résolution PLNE de ss-pb par une résolution par programmation dynamique et parallélisation ?

⇒ Exercice : étendre à une résolution arborescente Branch&Price calculant de telles bornes GC à chaque noeud du parcours arborescent.

# Stabilité de la convergence

- ▶ Point crucial : stabilité de la convergence. Premières étapes, comportement très erratique des variables duales sur les premières itérations.
- ▶ Générer plusieurs colonnes de coût réduit négatif peut aider à stabiliser la convergence. Peut se faire avec résolution exacte Branch&Bound ou programmation dynamique, ou recherche locale.
- ▶ Problème plus stable avec des contraintes d'inégalité, car variables duales signées. Pour CSP, on a satisfait une demande supérieure, autorisant a priori plus de solutions, mais mêmes valeur et mêmes solutions.

# Techniques de stabilisation de génération de colonne

- ▶ Box step : R. Marsten, W. Hogan, J. Blankenship, *The Boxstep Method for Large-Scale Optimization*, Operations Research, 1975, vol 23 no 3, 389-405
- ▶ J. Valério de Carvalho, *Using Extra Dual Cuts to Accelerate Column Generation*, INFORMS Journal on Computing, 2005 vol. 17 no. 2, 175-182
- ▶ In/out separation : A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck. *In-Out Separation and Column Generation Stabilization by Dual Price Smoothing*. Submitted for publication, 2012.
- ▶ A l'aide d'heuristiques : du Merle O., Villeneuve D., Desrosiers J., and P. Hansen, *Stabilized column generation*, Discrete Mathematics, **94**,(1999), 229-237.
- ▶ W. Ben-Ameur and J. Neto : *Acceleration of cutting planes and column generation algorithms : Application to network design*. Networks, 2007, Vol 49 (1), p 3-17.
- ▶ Par calcul par point intérieur de problème maître : J. Gondzio, P. Gonzalez-Brevis, P. Munari, *New developments in the primal-dual column generation technique*, European Journal of Operational Research, 2013, Vol 224, 2013, pp 41-51.
- ▶ Pessoa, A., Sadykov, R., Uchoa, E., & Vanderbeck, F. (2018). Automation and combination of linear-programming based stabilization techniques in column generation. INFORMS Journal on Computing, 30(2), 339-360.

# Apports de (méta-)heuristiques

- ▶ Majorité du temps de calcul dans sous problèmes, entiers en général.
- ▶ Générer rapidement des colonnes de coût réduit négatif aux premières itérations suffit.
- ▶ Intérêt de résoudre le sous problème à l'optimalité : pour prouver qu'il ne sert à rien de rajouter de colonnes et l'optimalité de la solution courante.
- ▶ (Méta-)heuristiques utiles pour fournir des solutions initiales intéressantes et partir d'un meilleur signal de variables duales.
- ▶ De plus, générer plusieurs colonnes de coût réduits négatifs, et pas que la meilleure, est bénéfique à la GC pour diminuer le nombre d'itérations et participer à la stabilisation (plusieurs raisons). Les méta-heuristiques, la recherche locale, les méta-heuristiques de population sont adaptées pour cela.
- ▶ N.B : pour le simplexe, le meilleur pivot à faire entrer en base n'est pas forcément celui de meilleur cour réduit. Tout pareil pour GC!!

# Nettoyer le pool de colonnes

- ▶ A force de générer des colonnes, cela peut poser des problèmes : espace mémoire, temps de calculs de PL/PLNE de PMR ...
- ▶ On peut définir une taille maximale du pool de colonnes.
- ▶ Un critère de nettoyage : calculer le coût réduit des colonnes pour le dernier signal dual (ou la moyenne des 2/3 derniers) et éliminer les colonnes avec le plus grand coût réduit
- ▶ Les colonnes générées aux premières itérations avec un signal dual très erratique peuvent être très éloignées en coût réduit des colonnes sur le signal dual stabilisé en fin de convergence.
- ▶ Opérations de nettoyage à utiliser au minimum, il est contre-productif de l'utiliser pour gérer un petit nombre de colonnes !

Desrosiers, J. ; Lübbecke, M. A primer in column generation. In Column generation ; Springer, 2005 ; pp. 1–32.

N. Dupin, R. Parize, and E. Talbi. Matheuristics and Column Generation for a Basic Technician Routing Problem. Algorithms, 14(11) :313, 2021.



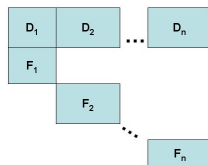
# Multiplicité de sous problèmes

Comme pour la relaxation Lagrangienne, on peut avoir des sous-pb indépendants pour minimiser le coût réduit

$$x = (x_1, \dots, x_K)$$

$$CR = \min_{k \in [1;K]} c_k x_k$$

$$Bx \geq b : F_k x_k \geq b_k, \forall k \in \{1, \dots, K\}$$



Exemple cutting stock : si on a plusieurs rouleaux de longueurs différentes, on peut distinguer els cas sur la longueur des grands rouleaux

Dans l'algorithme : générer la meilleure colonne sur le coût réduit, ie générer pour chaque ss-pb la meilleure colonne revient au même, même calculs. On a alors des calculs indépendants, qui peuvent se paralléliser à haut niveau (MPI ou OpenMP)

On n'est pas obligé de tout calculer, soit on a trouvé une colonne de coût réduit de coût négatif, on peut s'arrêter.

Avec des sous-pb hétérogènes, on peut aussi sélectionner uniquement un sous-ensemble de ss-pb, tant qu'on génère des solutions de coût réduit négatif.

Dans ce cadre, on peut utiliser de l'apprentissage par renforcement pour sélectionner les sous-pb à résoudre.

# Parallélisation et apprentissage, une vue d'ensemble

- ▶ On peut paralléliser la résolution d'un sous-problème (parallélisation bas-niveau)
- ▶ Avec des sous-pb multiples, on peut paralléliser à haut niveau (ex : MPI)
- ▶ En ayant plusieurs heuristiques de résolution de sous-pb, on peut faire une parallélisation de haut niveau pour essayer un porte-feuille d'heuristiques (cf cours MPI et méta-heuristiques)
- ▶ Machine Learning peut être utilisé pour trouver des solutions de coût réduit négatif, quand les sous pb et variables duales.
- ▶ Apprentissage par renforcement, pour choisir des sous-pb à résoudre, ou des heuristiques à utiliser en fonction de la récompense (reward) observée jusque là.

# Idées reçues fréquentes sur la génération de colonne

- ▶ ON NE RESOUT PAS DE PROBLEMES ENTIERS par génération de colonne, juste des PL. (utilise dualité forte)
- ▶ Si on prend les colonnes générées et que l'on fait un calcul MIP, s'agit t'il de l'optimum ? Réponse NON en général
- ▶ Sur CSP, on trouve souvent des solutions très proches de l'optimum.
- ▶ Résolution entière : Branch&Price, cf dernière section.
- ▶ Partir d'une solution réalisable, sinon, premier pb maître infaisable (éventuellement ajouter des pénalisations dans le PMR).

# Références

- ▶ Article fondateur : P. Gilmore, R. Gomory, *A linear programming approach to the cutting-stock problem* - Operations research, 1961.
- ▶ J.M. Valério de Carvalho, *LP models for bin packing and cutting stock problems*, European Journal of Operational Research, Vol 141, Issue 2, 2002, pp 253–273
- ▶ N. Perrot : *Integer Programming Column Generation Strategies for the Cutting Stock Problem and its Variants*, Université de Bordeaux, Thèse de Doctorat, 2005.
- ▶ M. Lübbecke, J. Desrosiers, *Selected topics in column generation* Operations Research, 2005, vol. 53 no. 6 1007-1023
- ▶ G. Desaulniers, J. Desrosiers, M. Solomon, *Column Generation*, Springer-Verlag New York Inc, 19 mai 2005

# Plan

Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

Applications pratiques, résultats

- Cutting Stock Problem (CSP)

- Problème de coloration d'un graphe "Vertex coloring"

- Problèmes de tournées de véhicules (VRP)

- Problèmes de clustering

- Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions

# Une définition

Reformulation étendue d'un MIP, obtenue par dualisation Lagrangienne, et permettant de calculer également des bornes Lagrangiennes.

Cadre entier pour simplifier la présentation. Pour le cas général, on pourra se reporter à :

F. Vanderbeck, M. Savelsbergh, *A generic view of Dantzig-Wolfe Decomposition in Mixed Integer Programming*. Operations Research Letters, Vol 34, Issue 3, May 2006, p 296–306.

# Cadre mathématique

On considère le programme linéaire en nombres entiers :

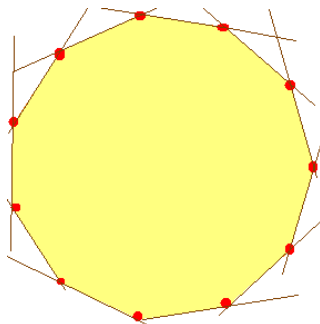
$$\begin{aligned} OPT = & \min cx \\ \text{s.c : } & Ax \geq a \\ \text{s.c : } & Bx \geq b \\ & x \in \mathbb{N}^n \end{aligned} \tag{13}$$

Où les contraintes  $Bx \geq b$  sont dites “faciles”, la résolution de problèmes sous contraintes  $Bx \geq b$  est bien plus facile que le problème complet.

On dualise les contraintes difficiles ou liantes ( $Ax \geq a$ ).

On suppose  $P = \{x \in \mathbb{N}^n, Bx \geq b\}$  est non vide et borné (cadre applicatif usuel)

## Rappel : enveloppe convexe



Enveloppe convexe d'un nombre fini de points  $a_i$  ? C'est un polyèdre le plus petit convexe contenant ces points, c'est :

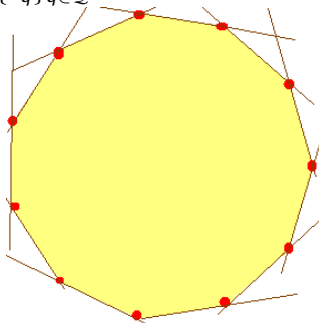
$$\text{conv}(A) = \left\{ x \in \mathbb{R}^n \mid \exists (\lambda_i) \geq 0, \sum_{i=1}^m \lambda_i = 1, x = \sum_{i=1}^m \lambda_i a_i \right\}$$

Les points  $a_i$  sont caractérisés par  $(\lambda_i) \in \mathbb{N}$



# Réciproquement

Si  $P = \{x \in \mathbb{N}^n, Bx \geq b\}$  est non vide et borné, c'est un polytope, qui s'écrit comme une enveloppe convexe d'un nombre fini de points, disons  $\{x \in \mathbb{N}^n, Bx \geq b\} = \{x_q\}_{q \in Q}$



Un point entier de  $P$  s'écrit,  $x = \sum_q \lambda_q x_q$ , avec  $\lambda_q \in \{0, 1\}^{|Q|}$  et

$$\sum_q \lambda_q = 1.$$

# Reformulation de Dantzig Wolfe

On énumère les solutions  $\{x \in \mathbb{N}^n, Bx \geq b\} = \{x_q\}_{q \in \mathcal{Q}}$

Formulation étendue, indexée sur les points entiers de  $P$  :

$$\begin{aligned} OPT = & \min_{\lambda \in \{0,1\}^{|\mathcal{Q}|}} \sum_q \lambda_q (cx_q) \\ \text{s.c : } & \sum_q \lambda_q (Ax_q) \geq a \\ & \sum_q \lambda_q = 1 \end{aligned} \tag{14}$$

Equivalent à ne considérer que les points extrêmes du polyèdre définissant l'enveloppe convexe de  $\{x \in \mathbb{N}^n, Bx \geq b\}$

$\implies$  Une meilleure borne est obtenue en prenant la borne continue de cette formulation étendue, dite reformulation de Dantzig Wolfe.

# Comparaison des bornes

On a les inclusions :

$$\{x \in \mathbb{R}^n, Bx \geq b\} \subset \left\{ x \in \mathbb{R}^n \mid \exists \lambda_q \in [0, 1]^{|Q|}, \sum_q \lambda_q = 1, x = \sum_q \lambda_q x_q \right\}$$
$$\left\{ x \in \mathbb{R}^n \mid \exists \lambda_q \in [0, 1]^{|Q|}, \sum_q \lambda_q = 1, x = \sum_q \lambda_q x_q \right\} \subset \{x \in \mathbb{R}^n, Bx \geq b\}$$

En prenant l'intersection avec  $\{x \in \mathbb{R}^n, Ax \geq a\}$ , on a l'inclusion des domaines de solutions définissant les optimisations :

$$Z_{LP}^{compact} \leq Z_{LP}^{DW} \leq OPT$$

où  $Z_{LP}^{compact}$  est la borne de relaxation continue de la formulation originelle, formulation compacte, tandis que  $Z_{LP}^{DW}$  est la borne obtenue comme relaxation continue de la reformulation de Dantzig Wolfe.

On prouvera que  $Z_{LP}^{DW}$  est une borne Lagrangienne, cela prouvera le théorème de Geoffroyon, en donnant un éclairage géométrique

# Cas d'égalité

$$Z_{LP}^{compact} \leq Z_{LP}^{DW} \leq OPT$$

où  $Z_{LP}^{compact}$  est la borne de relaxation continue de la formulation originelle, formulation compacte, tandis que  $Z_{LP}^{DW}$  est la borne obtenue comme relaxation continue de la reformulation de Dantzig Wolfe.

Dans le cas où on a l'égalité :

$$\{x \in \mathbb{R}^n, Bx \geq b\} = \left\{ x \in \mathbb{R}^n \mid \exists \lambda_q \in [0, 1]^{|Q|}, \sum_q \lambda_q = 1, x = \sum_q \lambda_q x_q \right\}$$

On a  $Z_{LP}^{compact} = Z_{LP}^{DW}$ , bornes DW et LP sont identiques.

C'est le cas où tous les points extrêmes du polytope  $\{x \in \mathbb{R}^n, Bx \geq b\}$  sont entiers. (ex : avec matrice totalement unimodulaire)

Dans ce cas, il n'y a pas d'intérêt à opérer une décomposition de DW, dont le temps de calcul des bornes est plus coûteux.

# Interprétation de la borne de Dantzig Wolfe

Calculer la borne DW, c'est "convexifier" les sous problèmes  $Bx \geq b$ .

La borne de Dantzig-Wolfe est la borne que l'on obtiendrait par relaxation continue si la description polyédrale des contraintes  $Bx \geq b$  était parfaite (avec des points extrêmes entiers).

Le gain de relaxation continue que l'on obtient provient de l'imperfection de la description polyédrale des contraintes  $Bx \geq b$

S'il est rare d'avoir des points extrêmes entiers à  $\{x \in \mathbb{R}^n, Bx \geq b\}$ , calculer en temps polynomial (ex prog. dynamique) un point extrême relatif à une direction d'optimisation est plus courant. Le théorème de Lovasz-Schrijver-Grötschel permet alors d'avoir un calcul de borne en temps polynomial, comme si on avait une description polyédrale parfaite du sous-problème.

En pratique, DW s'utilise aussi avec des algorithmes non polynomiaux mais assez efficaces en pratique sur les tailles de problèmes considérées. ex prog dynamique pseudo-polynomiale, algorithme d'étiquetage plus efficaces qu'une résolution PLNE de sous problème.

# Mise en oeuvre de la procédure de génération de colonne

Problème maître restreint à un nombre de colonnes

$$\begin{aligned} OPT = & \min_{\lambda \in \{0,1\}^{|\mathcal{Q}|}} \sum_q \lambda_q (cx_q) \\ \text{s.c : } & \sum_q \lambda_q (Ax_q) \geq a \quad (\pi) \\ & \sum_q \lambda_q = 1 \quad (\sigma) \end{aligned} \tag{15}$$

$\sigma$  variable duale non signée, contrairement à  $\pi$ .

Cout réduit d'une colonne  $q$  :  $CR_q = cx_q + \sigma - \pi Ax_q$

Le sous problème de génération de colonne est ainsi

$$\min_{x \in \mathbb{N}^n, Bx \geq b} \sigma + (c - \pi A)x$$

# Sous problème de génération de colonne

Le sous problème de génération de colonne est ainsi :

$$\begin{aligned} \min_x & \sigma + (c - \pi A)x \\ \text{s.c : } & Bx \geq b \\ & x \in \mathbb{N}^n \end{aligned} \tag{16}$$

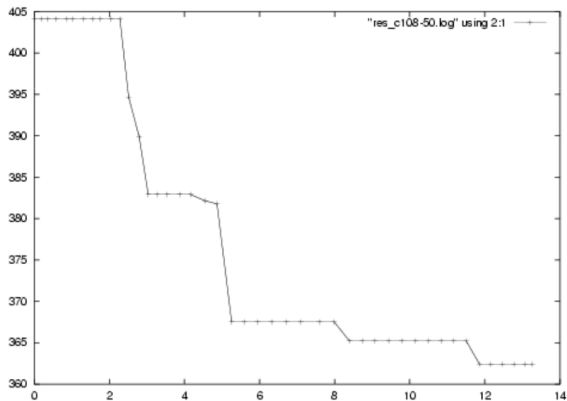
- Sous problème entier, non polynomial en général.
- En général, on cherche à avoir une résolution par un algorithme spécifique, très souvent un algorithme de programmation dynamique.

# Convergence de la borne de DW

- ▶ Valeur décroissante du PMR, mais pas forcément strictement décroissante (comme pour le simplexe, on peut avoir des dégénérescences, cas d'égalité entre deux itérations)
- ▶ A la convergence : une borne inférieure.
- ▶ Les valeurs intermédiaires du PMR ne sont pas des bornes lagrangiennes, ce sont des majorants de la borne de relaxation continue de la formulation étendue (similaire au simplexe primal). Mais ...



# Convergence de la borne de DW



# Bornes lagrangiennes à chaque itération de Dantzig Wolfe

On applique la dualisation lagrangienne à formulation étendue, de même valeur sur les points entiers :

$$\begin{aligned} l(\pi) = & \min_{\lambda \in \{0,1\}^{|\mathcal{Q}|}} \sum_q \lambda_q (cx_q) + \pi(a - \sum_q \lambda_q (Ax_q)) \\ \text{s.c : } & \sum_q \lambda_q = 1 \end{aligned} \tag{17}$$

A chaque iteration de DW, on a une borne Lagrangienne inférieure :

$$l(\pi) = \pi a + \xi(\pi) - \sigma$$

avec  $\xi(\pi) = \min_{x \in \mathbb{N}^n, Bx \geq b} \sigma + (c - \pi A)x$  valeur optimale du sous-problème.

# Bornes lagrangiennes et PMR

A chaque iteration de DW, on a une borne Lagrangienne inférieure :

$$l(\pi) = \pi a + \xi(\pi) - \sigma$$

avec  $\xi(\pi) = \min_{x \in \mathbb{N}^n, Bx \geq b} \sigma + (c - \pi A)x$ .

A la convergence de l'algorithme de génération de colonne  $\xi(\pi) = 0$ , la borne Lagrangienne est  $l(\pi) = \pi a - \sigma$

Par dualité forte, c'est la valeur optimale du PMR continu.

# Structure matricielle de décomposition

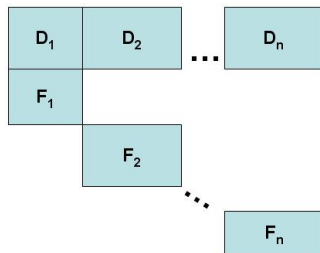
Structure matricielle usuelle :

$$x = (x_1, \dots, x_n)$$

$$Ax \geq a : \sum_{k=1}^n D_k x_k \geq a$$

$$Bx \geq b : F_k x_k \geq b_k, \forall k \in \{1, \dots, n\}$$

Les contraintes  $Bx \geq b$  forment  $n$  sous problèmes indépendants



# Reformulation de Dantzig Wolfe : cas bloc diagonal

Convexification des sous blocs  $F^k$ . Pour tout  $k$ , on énumère les solutions  $\{x \in \mathbb{N}^n, F^k x \geq b^k\} = \{x_q^k\}_{q \in Q}$

Reformulation de Dantzig Wolfe :

$$\begin{aligned} OPT = & \min \sum_k \sum_q \lambda_q^k (c^k x_q^k) \\ \text{s.c :} & \sum_k \sum_q \lambda_q^k (D^k x_q^k) \geq a \quad (\pi) \\ \forall k, & \sum_q \lambda_q^k = 1 \quad (\sigma) \\ & \lambda_q^k \in \{0, 1\} \end{aligned} \tag{18}$$

(les colonnes des différents sous problèmes indépendants se recombinent pour former une colonne du problème complet)

## Sous problèmes : cas bloc diagonal

Le sous problème de génération de colonne est alors décomposable par sous bloc :

$$\begin{aligned} SP_k \quad & \min_x \sigma^k + (c^k - \pi D^k)x \\ \text{s.c :} \quad & F^k x \geq b^k \\ & x \in \mathbb{N}^n \end{aligned} \tag{19}$$

Pour chaque sous problème de valeur strictement négative, on introduit la colonne partielle. (alors pas forcément nécessaire de calculer tous les sous-problèmes)

Le critère d'arrêt est de n'avoir aucun de ces sous-problèmes qui donne une colonne de coût réduit négatif

Pour avoir une borne Lagrangienne , il faut calculer tous les

sous-problèmes  $SP_k : I(\pi) = \pi a + \sum_{k=1}^K SP_k - \sigma$

# Multiplicité de sous problèmes

Avec des sous-problèmes multiples : ie  $c_k = c_{k'}$ ,  $D_k = D_{k'}$  et  $F_k = F_{k'}$ , on résout le même problème d'optimisation dans  $SP_k$  et  $SP_{k'}$

Si les  $K$  sous-problèmes sont partitionnés en  $K' \leq K$  sous-pb avec multiplicité, le sous-pb  $k$  ayant une multiplicité  $m(k)$ , on n'a que  $K'$  sous-problèmes à résoudre comme sous problème, et on a une borne Lagrangienne avec :

$$l(\lambda) = \min_{x \in \mathbb{N}^n: Bx \geq b} c'x = \sum_{k=1}^{K'} m(k) \times SP_k$$

# Multiplicité de sous-pb et reformulation de Dantzig-Wolfe

Avec  $K' \leq K$  sous-problèmes multiples, le sous-pb  $k$  ayant une multiplicité  $m(k)$ , la reformulation de Dantzig Wolfe est alors de manière équivalente :





$$\begin{aligned} OPT = & \min_{\lambda_q^k \in \{0,1\}} \sum_{k=1}^{K'} \sum_q \lambda_q^k (c^k x_q^k) \\ \text{s.c :} & \sum_k \sum_q \lambda_q^k (D^k x_q^k) \geq a \quad (\pi) \\ \forall k \in \llbracket 1; k' \rrbracket, & \sum_q \lambda_q^k = m(k) \quad (\sigma) \end{aligned} \quad (20)$$

Pour chaque type de sous problème  $k \in \llbracket 1; k' \rrbracket$ , on utilise  $m(k)$  colonnes pour former une solution du PMR

$\Rightarrow$  La multiplicité permet de casser des symétries, avantage crucial de la reformulation de Dantzig Wolfe par rapport à des formulations compactes



# Références

-  G. Desaulniers, G. Desrosiers, M. Solomon *Column generation*.
-  F. Vanderbeck, L.A. Wolsey, *Reformulation and Decomposition of Integer Programs*. In "50 Years of Integer Programming 1958-2008", Springer 2010.
-  M. Savelsbergh, F. Vanderbeck, *A generic view of Dantzig-Wolfe Decomposition in Mixed Integer Programming*. Operations Research Letters, Vol 34, Issue 3, May 2006, p 296–306.
-  W. Ben-Ameur, J. Neto, *A constraint generation algorithm for large scale linear programs using multiple-points separation*. Math. Program., Ser. A 107, 517–537 (2006)

# Plan

Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

**Applications pratiques, résultats**

Cutting Stock Problem (CSP)

Problème de coloration d'un graphe "Vertex coloring"

Problèmes de tournées de véhicules (VRP)

Problèmes de clustering

Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions

# Problèmes classiques pour la décomposition de Dantzig Wolfe

- ▶ Problème de découpe de papiers
- ▶ Problème de tournées de véhicules avec fenêtre de temps (VRPTW)
- ▶ Bin Packing 2D, 3D ...
- ▶ Lot sizing

# Plan

Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

**Applications pratiques, résultats**

Cutting Stock Problem (CSP)

Problème de coloration d'un graphe "Vertex coloring"

Problèmes de tournées de véhicules (VRP)

Problèmes de clustering

Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions

# Cutting Stock Problem (CSP), formulation compacte

- ▶ On a des rouleaux de papier de longueur  $L$ , dont on doit découper  $M$  types de motifs.
- ▶ Un motif  $m \in \llbracket 1, M \rrbracket$  a une longueur  $l_m$  et on doit en couper  $d_m$ .
- ▶ But : déterminer la découpe, de manière à minimiser le nombre total de grand rouleaux à utiliser.

Soit  $K$  un nombre réalisable de rouleaux. (Au pire  $K = \sum_{m=1}^M d_m$ , se calcule avec une heuristique, gloutonne ou GRASP par exemple)

- Variables  $x_{k,m} \in \mathbb{N}$  le nombre de fois que l'on découpe le motif  $m \in \llbracket 1, M \rrbracket$  dans le rouleau  $k \in \llbracket 1, K \rrbracket$ .
- Variables  $y_k \in \{0, 1\}$  si le rouleau  $k \in \llbracket 1, K \rrbracket$  est utilisé.

$$\begin{aligned} & \min_{x_{k,m} \in \mathbb{N}} \sum_{k=1}^M y_k \\ \text{s.c : } & \sum_{m=1}^M l_m x_{k,m} \leq L & \forall k \in \llbracket 1, K \rrbracket \\ & x_{k,m} \leq d_m y_k & \forall m \in \llbracket 1, M \rrbracket, \forall k \in \llbracket 1, K \rrbracket \end{aligned} \tag{21}$$

# Cutting Stock Problem (CSP), reformulation DW

- ▶ La reformulation de DW de la formulation compacte précédente donne exactement la formulation étendue et le schéma de génération de colonne présenté précédemment !!
- ▶ Pour CSP, c'est une reformulation de DW avec un unique sous-problème avec multiplicité.
- ▶ Pour CSP, la reformulation de DW et la convexification des sous-problèmes a un sens physique
- ▶ On remarque une similarité de formulation avec vertex coloring

⇒ Dans la suite, la reformulation de DW a un sens physique, on aurait pu raisonner juste avec l'écriture du dual pour écrire une génération de colonne sur une formulation étendue.

⇒ La théorie de DW permet de comprendre la relaxation Lagrangienne, et d'interpréter l'amélioration des bornes duales par une convexification des sous-problèmes.

# Plan

Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

**Applications pratiques, résultats**

Cutting Stock Problem (CSP)

Problème de coloration d'un graphe "Vertex coloring"

Problèmes de tournées de véhicules (VRP)

Problèmes de clustering

Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions

# Vertex coloring, formulation PLNE compacte

Soit  $C$  un majorant du nombre de couleurs. On utilise des variables binaires  $z_{v,c} \in \{0, 1\}$ ,  $y_c \in \{0, 1\}$  :

$$\begin{aligned} \min \quad & \sum_{c=1}^C y_c \\ \text{s.c. : } \quad & \forall v \in V, \quad \sum_{c=1}^C z_{v,c} = 1 \\ & \forall c \in \llbracket 1; C \rrbracket, \forall e = (v_1, v_2) \in E, \quad z_{v_1,c} + z_{v_2,c} \leq y_c \\ & \forall c \in \llbracket 1; C \rrbracket, \forall v \in V, \quad z_{v,c} \in \{0, 1\} \\ & \forall c \in \llbracket 1; C \rrbracket, \quad y_c \in \{0, 1\} \end{aligned} \tag{22}$$

En dualisant les contraintes  $\sum_{c=1}^C z_{v,c} = 1$ , on a  $C$  sous-problèmes identiques, pour une génération de colonnes avec un unique type de sous problème, de multiplicité  $C$ . Cela va casser les symétries



# Vertex coloring, formulation PLNE étendue

Soit  $\mathcal{S}$  l'ensemble des stables de  $G$ . Pour tout  $v \in V$ , on note  $\mathcal{S}_v$  l'ensemble des stables de  $G$  contenant  $v$ .

On utilise des variables binaires  $z_s \in \{0, 1\}$  pour  $s \in \mathcal{S}$  indiquant si le stable est choisi pour définir une couleur.

$$\begin{aligned} & \min \sum_{s \in \mathcal{S}} z_s \\ \text{s.c : } & \forall v \in V, \sum_{s \in \mathcal{S}_v} z_s = 1 \\ & \forall s \in \mathcal{S}, z_s \in \{0, 1\} \end{aligned} \tag{23}$$

Cette fois, on a une formulation avec un nombre potentiellement exponentiel de variables ! L'algorithme de génération de colonnes permettra de s'en sortir

$\Rightarrow$  C'est ici la reformulation de Dantzig-Wolfe de la première formulation compacte, permet de casser bcp de symétries.

# Vertex coloring, autre formulation PLNE étendue

Soit  $\mathcal{S}$  l'ensemble des stables maximaux de  $G$ . Pour tout  $v \in V$ , on note  $\mathcal{S}_v$  l'ensemble des stables maximaux de  $G$  contenant  $v$ .

On utilise des variables binaires  $z_s \in \{0, 1\}$  pour  $s \in \mathcal{S}$  indiquant si le stable est choisi pour définir une couleur.

$$\begin{aligned} & \min \sum_{s \in \mathcal{S}} z_s \\ \text{s.c : } & \forall v \in V, \sum_{s \in \mathcal{S}_v} z_s \geq 1 \\ & \forall s \in \mathcal{S}, z_s \in \{0, 1\} \end{aligned} \tag{24}$$

Formulation mieux adaptée à la résolution de la relaxation continue par l'algorithme de génération de colonnes

# Vertex coloring, génération de colonnes

Soit  $\mathcal{S}_0 \subset \mathcal{S}$ . On définit le problème maître restreint à  $\mathcal{S}_0$ .

$$\begin{aligned} \min \quad & \sum_{s \in \mathcal{S}_0} z_s \\ \text{s.c.} \quad & \forall v \in V, \quad \sum_{s \in \mathcal{S}_v \cap \mathcal{S}_0} z_s \geq 1 \quad (\lambda_v) \end{aligned} \quad (25)$$

La solution partielle est optimale (et donne la relaxation continue de la formulation étendue) s'il n'existe pas de stable  $s \in \mathcal{S}$  tel que  $\sum_{v \in s} \lambda_v > 1$ .

Cela revient à résoudre le problème Max Weight Stable Set Problem (MWSSP) :

$$\begin{aligned} \max \quad & \sum_{v \in V} \lambda_v x_v \\ \text{s.c.} \quad & \forall e = (v, w) \in E, \quad x_v + x_w \leq 1 \\ & \forall v \in V, \quad x_v \in \{0, 1\} \end{aligned} \quad (26)$$

Résolutions itérées de problèmes MWSSP pour converger vers une borne inférieure de vertex coloring. Si un stable tel que  $\sum_{v \in s} \lambda_v > 1$  est trouvé facilement (heuristique), pas besoin de résoudre à l'optimalité MWSSP pour itérer l'algo GC, nécessaire uniquement à la convergence.

# Plan

Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

**Applications pratiques, résultats**

Cutting Stock Problem (CSP)

Problème de coloration d'un graphe "Vertex coloring"

Problèmes de tournées de véhicules (VRP)

Problèmes de clustering

Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions

# Problèmes de tournées de véhicules

Vehicle Routing Problem (VRP) : problème de tournées de véhicules.

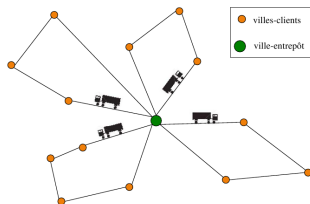
On a  $N$  livraisons indicées  $n \in \llbracket 1, N \rrbracket$ .

On a  $V$  véhicules identiques pour réaliser les livraisons, en partant et en revenant à un dépôt  $n = 0$

on note  $\mathcal{N} = \llbracket 0, N \rrbracket$  et  $\mathcal{V} = \llbracket 1, V \rrbracket$

On veut minimiser la longueur totale des trajets en utilisant plusieurs véhicules.

CVRP : avec des contraintes de capacité, il n'est pas réalisable qu'un seul véhicule desserve toutes les livraisons : chaque livraison  $n$  nécessite une capacité (par exemple volume) limitée au total à  $C$ , chaque livraison consommant une capacité  $c_n$  dans le budget  $C$ .



# CVRP, modélisation PLNE

Variables  $x_{v,n,n'} = 1$  ssi le véhicule  $v$  va au noeud  $n' \in \mathcal{N}$  immédiatement après avoir été au noeud  $n \in \mathcal{N}$ .

On peut avoir  $x_{v,0,0} = 1$  si le véhicule ne sort pas. Pour tout  $n > 0$ , on a  $x_{v,n,n} = 0$

$$\begin{aligned} \min_x \quad & \sum_{v \in \mathcal{V}} \sum_{n \in \mathcal{N}} \sum_{n' \in \mathcal{N}} d_{n,n'} x_{v,n,n'} \\ \forall n > 0, \quad & \sum_{v \in \mathcal{V}} \sum_{n' \in \mathcal{N}} x_{v,n,n'} = 1 \\ \forall v \in \mathcal{V}, \forall n > 0, \quad & \sum_{n' \in \mathcal{N}} x_{v,n,n'} = \sum_{n' \in \mathcal{N}} x_{v,n',n} \\ \forall v \in \mathcal{V}, \quad & \text{contraintes pour interdire les sous-tours du véhicule } v \\ \forall v \in \mathcal{V}, \quad & \text{autres contraintes sur la tournée du véhicule } v \\ \forall v \in \mathcal{V}, \quad & \sum_{n' \neq n} c_n x_{v,n',n} \leq C \\ \forall v \in \mathcal{V}, \forall n, n' \in \mathcal{N} \quad & x_{v,n,n'} \in \{0, 1\}, \end{aligned}$$

# Formulation étendue des CVRP

On énumère pour tout véhicule  $v$  l'ensemble des routes réalisables  $\mathcal{P}_v$ .

On suppose avoir  $K$  types de véhicules identiques, et disposer de  $m(k)$  véhicules de chaque type pour  $k \in \llbracket 1, K \rrbracket$

$z_{v,k} \in \{0, 1\}$  :  $z_{v,k} = 1$  si la route  $k \in \mathcal{P}_v$ , de coût  $c_{v,k}$  est choisie.

$$\begin{aligned} & \min_{\lambda_q^k \in \{0,1\}} \sum_{i,k} c_{v,k} z_{v,k} \\ \forall n \in \mathcal{N}, & \sum_{v,k} \mathbb{1}_{n \in k} z_{v,k} \geq 1 \quad (\pi) \\ \forall v \in \mathcal{V}, & \sum_{k \in \mathcal{P}_v} z_{v,k} = m(k) \quad (\sigma) \end{aligned} \tag{27}$$

Les sous problèmes sont souvent des ESPPRC : problèmes de plus court chemin avec contraintes de ressources, se résout assez efficacement par programmation dynamique et variantes

Feillet, D. (2010). A tutorial on column generation and branch-and-price for vehicle routing problems. 4OR, 8(4), 407-424.

# Problème de tournées de techniciens

- Problème d'affectation de tournées de techniciens devant visiter des clients dans des fenêtres de temps spécifiques.
- Soumis par France Télécom comme Challenge ROADEF.
- Multi dépôt : les techniciens partent de différents endroits
- Contraintes de compétences requises pour différentes tâches.



# Notations

- ▶  $\mathcal{I}$  : Ensemble des techniciens.
- ▶  $\mathcal{J}$  : ensemble des tâches à attribuer.
- ▶  $d_i$  : lieu de départ et de fin de journée du technicien  $i$
- ▶  $C_j^i \in \{0, 1\}$ , compétences,  $C_j^i = 1$  si le technicien  $i$  peut effectuer la tâche  $j$ , sinon  $C_j^i = 0$ .
- ▶  $D_j$  Durée de la tâche  $j$ .
- ▶  $P_j$  Coût de pénalité si la tâche  $j$  n'est pas effectuée.
- ▶  $\tilde{t}_i^{start}, \tilde{t}_i^{end}$  : Heures extrêmes de départ et d'arrivée à  $d_i$  pour le technicien  $i$
- ▶  $[\tilde{t}_j^{min}, \tilde{t}_j^{max}]$ , fenêtres de temps pour commencer la tâche  $j \in \mathcal{J}$

# Formulation compacte

Variables binaires  $u_{j,j'}^i \in \{0, 1\}$ , avec  $u_{j,j'}^i = 1$  si le technicien  $i$  effectue la tâche  $j$ , puis la tâche  $j'$ . ( $u_{j,j}^i = 0$ ).

On étend la définition avec  $u_{d_i,j}^i$  (et resp  $u_{j,d_i}^i$  respectively) indiquant la première tâche  $j$  de la journée du technicien  $i$ , et  $u_{d_i,j}^{i'} = 0$  si  $i \neq i'$ .

On introduit de plus des variables continues  $t_j$  représentant l'heure à laquelle la tâche  $j$  commence. Les bornes sur  $t_j$  sont définies avec les contraintes de fenêtres de temps :

$$\forall j \in \mathcal{J}, \tilde{t}_j^{\min} \leq t_j \leq \tilde{t}_j^{\max}$$

## Formulation compacte

$$\begin{aligned}
 & \min \sum_{j,j'} d(j,j') u_{j,j'}^i + \sum_j P_j \left( 1 - \sum_i \sum_{j'} u_{j',j}^i \right) \\
 \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, & \quad \sum_{j' \in \mathcal{J} \cup \{d_i\}} u_{j',j}^i = \sum_{j' \in \mathcal{J} \cup \{d_i\}} u_{j,j'}^i \\
 \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, & \quad \sum_{j' \in \mathcal{J} \cup \{d_i\}} u_{j',j}^i \leq C_i^j \\
 \forall i \in \mathcal{I}, & \quad \sum_{j' \in I} u_{j',d_i}^j = \sum_{j' \in I} u_{d_i,j'}^j \leq 1 \\
 (j,j') \in I, & \quad t_j + D_i + T(j,j') \leq t_{i'} + \left( 1 - \sum_i u_{j,j'}^i \right) \cdot M \\
 \forall i \in \mathcal{I}, j \in \mathcal{J}, & \quad \tilde{t}_i^{\text{start}} + T(d_i, j) \leq t_j + \left( 1 - u_{d_i,j}^i \right) \cdot M \\
 \forall i \in \mathcal{I}, j \in \mathcal{J}, & \quad t_j + D_j + T(j, d_i) \leq \tilde{t}_i^{\text{end}} + \left( 1 - u_{j,d_i}^i \right) \cdot M \\
 \forall i, j, j' & \quad u_{j,j'}^i \in \{0, 1\}, t_j \in \mathbb{R}
 \end{aligned} \tag{28}$$

# Formulation chemin du problème de tournées de techniciens

On introduit des techniciens fictifs  $j \in \mathcal{J}$  ne faisant que la tâche  $j$ , au coût de pénalisation  $P_j$ .

On énumère pour tout technicien  $i$  l'ensemble des routes réalisables  $\mathcal{P}_i$ .

$\lambda_i^k \in \{0, 1\} : \lambda_i^k = 1$  si la route  $k \in \mathcal{P}_i$ , de coût  $c_i^k$  est choisie.

$$\begin{aligned} \min \quad & \sum_{i,k} c_i^k \lambda_i^k \\ \text{s.c. : } & \forall j \in \mathcal{J}, \quad \sum_{i,k} \mathbb{1}_{j \in k} \lambda_i^k \geq 1 \quad (\pi) \\ & \forall i \in \mathcal{I}, \quad \sum_{k \in \mathcal{P}_i} \lambda_i^k = 1 \quad (\sigma) \\ & \lambda_q^k \in \{0, 1\} \end{aligned} \tag{29}$$

Formulation étendue, résolution du relâché continu par génération de colonne.

## Sous problèmes de génération de colonnes

Le sous problème de génération de colonne est alors décomposable par technicien  $i$  :

$$\begin{aligned} & \min -\sigma_i + \sum_{j,j'} (d(j,j') - \pi_j) u_{jj'} \\ \text{s.c : } & \forall j \in \mathcal{J}, \quad \sum_{j' \in \mathcal{J} \cup \{d_i\}} u_{j'j} = \sum_{j' \in \mathcal{J} \cup \{d_i\}} u_{jj'} \\ & \forall j \in \mathcal{J}, \quad \sum_{j' \in \mathcal{J} \cup \{d_i\}} u_{j'j}^i \leq C_i^j \\ & \sum_{j' \in I} u_{j' d_i}^j = \sum_{j' \in I} u_{d_i j'}^j \leq 1 \\ & \forall (j, j') \in \mathcal{J}, \quad t_j + D_j + T(j, j') \leq t_{j'} + (1 - u_{j, j'}) \cdot M \\ & j \in \mathcal{J}, \quad \tilde{t}_i^{\text{start}} + T(d_i, j) \leq t_j + (1 - u_{d_i j}) \cdot M \\ & j \in \mathcal{J}, \quad t_j + D_j + T(j, d_i) \leq \tilde{t}_i^{\text{end}} + (1 - u_{j d_i}) \cdot M \\ & \forall j, j' \quad u_{j'j} \in \{0, 1\}, \quad t_j \in \mathbb{R} \end{aligned} \tag{30}$$

Résolution MIP permet de générer plusieurs solutions de coût réduit négatif

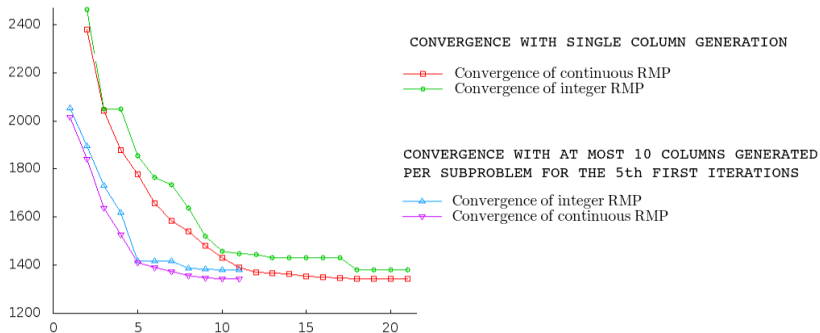
# Résultats sur les bornes duales

Bound Cuts	(1)–(8) LP No	(1)–(8) nod1 No	(1)–(8) LP (10)	(1)–(8) nod1 (10)	(12)–(19) LP No	(12)–(19) nod1 No	(12)–(19) LP (21)	(12)–(19) nod1 (21)	(27)–(29) LP No
data_3_1_10	45.7%	8.7%	11.9%	3.2%	28.1%	3.3%	8.8%	1.0%	1.8%
data_3_5_10	43.4%	0.0%	9.7%	0.0%	23.4%	0.0%	8.0%	0.0%	0.0%
data_5_1_20	44.8%	29.5%	27.9%	11.6%	35.7%	17.9%	22.0%	13.7%	1.9%
data_5_3_20	52.7%	30.3%	30.8%	17.7%	39.0%	19.9%	22.7%	15.5%	0.6%
data_5_5_20	58.1%	28.7%	27.5%	14.6%	42.1%	18.5%	25.1%	14.1%	0.3%
data_8_1_20	40.5%	20.6%	20.8%	10.6%	32.6%	10.6%	16.8%	7.8%	1.3%
data_8_3_20	54.7%	28.5%	28.6%	17.1%	35.2%	14.8%	20.0%	11.2%	1.8%
data_8_5_20	60.6%	26.7%	25.9%	11.2%	39.7%	8.1%	21.3%	1.6%	0.5%
data_10_10_40	55.9%	25.0%	25.8%	14.3%	40.8%	19.4%	20.7%	13.9%	2.0%
data_15_10_40	55.6%	18.9%	19.6%	10.7%	35.8%	11.0%	13.4%	6.3%	0.6%
data_15_20_40	53.9%	14.2%	16.1%	6.1%	32.1%	7.5%	13.1%	4.8%	0.7%

Les premières colonnes avaient déjà été montrées. A comparer avec la dernière colonne, borne LP de la formulation DW, améliore nettement les autres bornes LP, et même les bornes LP+coupes de Cplex à, la racine de l'arbre de Branch&Bound.

N. Dupin, R. Parize, and E. Talbi. Matheuristics and Column Generation for a Basic Technician Routing Problem. Algorithms, 14(11) :313, 2021.

# Convergence et stabilisation



N. Dupin, R. Parize, and E. Talbi. Matheuristics and Column Generation for a Basic Technician Routing Problem. Algorithms, 14(11) :313, 2021.

# Plan

Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

**Applications pratiques, résultats**

Cutting Stock Problem (CSP)

Problème de coloration d'un graphe "Vertex coloring"

Problèmes de tournées de véhicules (VRP)

**Problèmes de clustering**

Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions



# Rappels : Problèmes de clustering k-means/k-medoids

Soit  $E = \{x_1, \dots, x_N\}$  un ensemble de  $N$  éléments of  $\mathbb{R}^d$ . On définit  $\Pi_K(E)$ , l'ensemble des partitions de  $E$  en  $K$  sous-ensembles :

$$\Pi_K(E) = \left\{ P \subset \mathcal{P}(E) \mid \forall p, p' \in P, p \cap p' = \emptyset \text{ and } \bigcup_{p \in P} p = E \text{ and } \text{card}(P) = K \right\}$$

En définissant une fonction de coût  $f$  mesurant la dissimilarité de chaque sous ensemble de  $E$ , on définit des problèmes de clustering comme des problèmes d'optimisation indexés par  $\Pi_K(E)$  :

$$\min_{\pi \in \Pi_K(E)} \sum_{P \in \pi} f(P) \quad (31)$$

K-medoids :  $f_{\text{medoids}}(P) = \min_{y \in P} \sum_{x \in P} \|x - y\|^2$

K-means :

$$f_{\text{means}}(P) = \min_{y \in \mathbb{R}^d} \sum_{x \in P} \|x - y\|^2 = \sum_{x \in P} \left\| x - \frac{1}{\text{card}(P)} \sum_{y \in P} y \right\|^2$$

K-medoids est la version discrète de k-means : le centre d'un cluster, le médoïde, est un des points de  $E$ .

N.B : On utilise ici une distance issue d'une norme et on note  $d(x_i, x_j) = \|x_i - x_j\|$ .  
On peut considérer une distance quelconque dans la suite

# Formulation étendue générale de problèmes sum-clustering

On énumère tous les sous-ensemble de  $E$ ,  $p \in \mathcal{P} = \mathcal{P}(E)$ .  $c_p$  désigne le coût du cluster  $p$  suivant la mesure choisie.

Variables binaires  $z_p \in \{0, 1\}$  :  $z_p = 1$  ssi le sous-ensemble  $p$  est sélectionné.

$$\begin{array}{ll} \min_z & \sum_{p \in \mathcal{P}} c_p z_p \\ \text{s.c : } \forall n, & \sum_{p \in \mathcal{P}} \mathbb{1}_{n \in p} z_p = 1 \\ & \sum_{p \in \mathcal{P}} z_p = K \\ \forall p & z_p \in \{0, 1\} \end{array} \quad (32)$$

$K$  sous-ensembles sélectionnés exactement, chaque point est présent dans un cluster sélectionné (OK aussi avec inégalité, au moins un).

Si on veut avoir une taille maximale de cluster, c'est restreindre  $\mathcal{P}$

⇒ Variante de clustering partiel ?

⇒ Et pour max-clustering ?

# Formulation étendue de problèmes sum-clustering partiels

Variables binaires  $z_p \in \{0, 1\}$  :  $z_p = 1$  ssi le sous-ensemble  $p$  est sélectionné.

Variables binaires  $m_n \in \{0, 1\}$  :  $m_n = 1$  si le point  $n$  est retiré, considéré comme un outlier.

$$\begin{aligned} \min_z \quad & \sum_{p \in \mathcal{P}} c_p z_p \\ \text{s.c : } \forall n, \quad & \sum_{p \in \mathcal{P}} \mathbb{1}_{n \in p} z_p \geq 1 - m_n \\ & \sum_n m_n \leq M' \\ & \sum_{p \in \mathcal{P}} z_p \leq K \\ & z_p, m_n \in \{0, 1\} \end{aligned} \tag{33}$$

Remarque : on peut aussi avoir les inégalités dans le modèle précédent, cas  $M' = 0$  ou sans variable  $m_n$

$\Rightarrow$  Comment écrire le schéma de génération de colonnes ?

# Formulation étendue de problèmes max-clustering

Variables binaires  $z_p \in \{0, 1\}$  :  $z_p = 1$  ssi le sous-ensemble  $p$  est sélectionné.

$$\begin{aligned} & \min_{z} \max_{p \in \mathcal{P}} c_p z_p \\ \text{s.c : } & \forall n, \sum_{p \in \mathcal{P}} \mathbb{1}_{n \in p} z_p \geq 1 \\ & \sum_{p \in \mathcal{P}} z_p \leq K \\ & z_p \in \{0, 1\} \end{aligned} \tag{34}$$

Se linéarise en

$$\begin{aligned} & \min_{z, C \geq 0} C \\ \text{s.c : } & \forall n, \sum_{p \in \mathcal{P}} \mathbb{1}_{n \in p} z_p \geq 1 \\ & \forall p, c_p z_p \leq C \\ & \sum_{p \in \mathcal{P}} z_p \leq K \\ & z_p \in \{0, 1\} \end{aligned} \tag{35}$$

Remarque : on a une formulation avec un nombre non énumérable de variables ET de contraintes..

⇒ Comment écrire le schéma de génération de colonnes ?

# PMR et pricing pour problèmes sum-clustering (1)

On restreint à  $\mathcal{P}' \subset \mathcal{P}$  et on calcule le relâché continu. Deux variantes :

$$\begin{aligned} \text{PMR}(\mathcal{P}') &= \min_{z \geq 0} \sum_{p \in \mathcal{P}'} c_p z_p \\ \text{s.c. : } \forall n, \quad & \sum_{p \in \mathcal{P}'} \mathbb{1}_{n \in p} z_p \geq 1 & (\pi_n) \\ & \sum_{p \in \mathcal{P}'} z_p \leq K & (\sigma) \\ \forall p \in \mathcal{P}' & z_p \leq 1 & (\lambda_p) \end{aligned} \tag{36}$$

Ici, variables duales signées, mais des contraintes et variables duales indexées sur  $\mathcal{P}'$ . Variante, avec l'égalité et la positivité, pas besoin de contraintes de bornes supérieures pour  $z$  :

$$\begin{aligned} \text{PMR}(\mathcal{P}') &= \min_{z \geq 0} \sum_{p \in \mathcal{P}'} c_p z_p \\ \text{s.c. : } \forall n, \quad & \sum_{p \in \mathcal{P}'} \mathbb{1}_{n \in p} z_p = 1 & (\pi_n) \\ & \sum_{p \in \mathcal{P}'} z_p \leq K & (\sigma) \end{aligned} \tag{37}$$

## PMR et pricing pour problèmes sum-clustering (2)

On se ramène à une forme usuelle pour la dualité :

$$\begin{aligned} \text{PMR}(\mathcal{P}') = & \min_{z \geq 0} \sum_{p \in \mathcal{P}'} c_p z_p \\ \text{s.c. : } & \forall n, \quad \sum_{p \in \mathcal{P}'} \mathbb{1}_{n \in p} z_p \geq 1 \quad (\pi_n^+) \\ & \forall n, \quad \sum_{p \in \mathcal{P}'} -\mathbb{1}_{n \in p} z_p \geq -1 \quad (\pi_n^-) \\ & \sum_{p \in \mathcal{P}'} -z_p \geq -K \quad (\sigma) \end{aligned} \quad (38)$$

On applique la dualité forte :

$$\begin{aligned} \text{PMR}(\mathcal{P}') = & \max_{\pi_n^+, \pi_n^-, \sigma \geq 0} -K\sigma + \sum_n (\pi_n^+ - \pi_n^-) \\ \text{s.c. : } & \forall p \in \mathcal{P}', \quad -\sigma + \sum_n \mathbb{1}_{n \in p} (\pi_n^+ - \pi_n^-) \leq c_p \quad (z_p) \end{aligned} \quad (39)$$

## PMR et pricing pour problèmes sum-clustering (2')

On se ramène à une forme usuelle pour la dualité :

$$\begin{aligned} \text{PMR}(\mathcal{P}') = & \min_{z \geq 0} \sum_{p \in \mathcal{P}'} c_p z_p \\ \text{s.c. : } \forall n, & \sum_{p \in \mathcal{P}'} \mathbb{1}_{n \in p} z_p = 1 \quad (\pi_n) \\ & \sum_{p \in \mathcal{P}'} -z_p \geq -K \quad (\sigma) \end{aligned} \tag{40}$$

On applique la dualité forte :

$$\begin{aligned} \text{PMR}(\mathcal{P}') = & \max_{\pi_n \in \mathbb{R}, \sigma \geq 0} -K\sigma + \sum_n \pi_n \\ \text{s.c. : } \forall p \in \mathcal{P}', & -\sigma + \sum_n \mathbb{1}_{n \in p} \pi_n \leq c_p \quad (z_p) \end{aligned} \tag{41}$$

Forme équivalente, avec  $\pi$  non signée ici. (la dernière planche retrouve le résultat, sans connaître par coeur les autres formes de dualité, on retrouve facilement )

# PMR et pricing pour problèmes sum-clustering (3)

A partir du PMR où on a obtenu des variables duales  $\sigma \geq 0$  et  $(\pi_n)$  non signées, il faut déterminer s'il existe  $p \in \mathcal{P}$  tel que

$$-\sigma + \sum_n \mathbb{1}_{n \in p} \pi_n > c_p \quad (42)$$

Cela définit le sous problème :

$$SP = \min_{p \in \mathcal{P}} c_p - \sum_n \mathbb{1}_{n \in p} \pi_n \quad (43)$$

La génération de colonne s'arrête si  $SP \geq -\sigma$ .

Sinon, on itère en ajoutant un (ou des) ensemble(s)  $p$  tels que

$$c_p - \sum_n \mathbb{1}_{n \in p} \pi_n < -\sigma$$



# PMR et pricing pour problèmes sum-clustering partiels (1)

On restreint à  $\mathcal{P}' \subset \mathcal{P}$  et on calcule le relâché continu. On considère la variante de relaxation continue :

$$\begin{aligned} \text{PMR}'(\mathcal{P}') = \quad & \min_z \sum_{p \in \mathcal{P}} c_p z_p \\ \text{s.c : } \forall n, \quad & \sum_{p \in \mathcal{P}} \mathbb{1}_{n \in p} z_p = 1 - m_n \\ & \sum_n m_n \leq M' \\ & \sum_{p \in \mathcal{P}} z_p \leq K \\ & z_p, m_n \in \{0, 1\} \end{aligned} \tag{44}$$

On se ramène à une forme usuelle pour la dualité :

$$\begin{aligned} \text{PMR}'(\mathcal{P}') = \quad & \min_z \sum_{p \in \mathcal{P}} c_p z_p \\ \text{s.c : } \forall n, \quad & \sum_{p \in \mathcal{P}} \mathbb{1}_{n \in p} z_p = 1 - m_n & (\pi_n) \\ & \sum_n -m_n \geq -M' & (\lambda) \\ & \sum_{p \in \mathcal{P}} -z_p \geq -K & (\sigma) \\ & z_p, m_n \in \{0, 1\} \end{aligned} \tag{45}$$

# PMR et pricing pour problèmes sum-clustering partiels (2)

On applique la dualité forte :

$$\begin{aligned} \text{PMR}'(\mathcal{P}') = & \max_{\pi_n \in \mathbb{R}, \sigma, \lambda \geq 0} -K\sigma - M'\lambda + \sum_n \pi_n \\ \text{s.c : } & \forall p \in \mathcal{P}', \quad -\sigma + \sum_n \mathbb{1}_{n \in p} \pi_n \leq c_p & (z_p) \\ & \forall n, \quad \pi_n - \lambda \leq 0 & (m_n) \end{aligned} \quad (46)$$

Comme les  $N$  dernières contraintes sont toujours écrites complètement (les  $N$  variables sont toujours dans le PMR primal), cela ne change rien à précédemment, on génère des patterns  $p$  tels que  $c_p - \sum_n \mathbb{1}_{n \in p} \pi_n < -\sigma$ , mêmes problèmes de séparation/pricing !

Ici, les variables du primal  $m_n$  influent sur les valeurs numériques qui seront données aux variables duales  $\pi_n, \sigma$  pour le pb de séparation.

Expérience numérique : est ce que le clustering partiel augmente ou diminue le nombre d'itérations nécessaires à la GC pour converger ?

# Résolution de sous problème, cas général

Pour résoudre le pricing, à partir des variables duales  $\pi_n, \sigma$  du dernier PMR, il faut résoudre des sous problème, calculant un sous ensemble de  $E$ , et en comparant la valeur optimale à  $-\sigma$  :

$$SP = \min_{p \in \mathcal{P}} c_p - \sum_n \mathbb{1}_{n \in p} \pi_n \quad (47)$$

Encodage naturel :  $z_n \in \{0, 1\}$  avec  $z_n = 1$  si le point  $x_n$  est choisi dans le nouveau sous ensemble.

$$SP = \min_{p \in \mathcal{P}} c_p - \sum_n \pi_n z_n \quad (48)$$

Exprimer  $c_p$  en fonction de l'encodage  $z_n$  se fait au cas par cas selon les fonctions objectifs. On a des descriptions PNE permettant d'écrire sous-problème en un PNE que les solveurs linéaires sauront résoudre

# Résolution exacte de sous problème, cas K-means

$$SP = \min_{p \in \mathcal{P}} c_p - \sum_n \pi_n z_n \quad (49)$$

Variables binaires :  $z_n \in \{0, 1\}$  :  $z_n = 1$  ssi le point  $x_n$  est affecté au cluster.

Variables continues :

- $c \in \mathbb{R}^d$  : le centroïde du cluster.
- $d_n \geq 0$  le carré de la distance du point  $x_n$  à son centre de cluster  $c$  si  $z_n = 1$  et  $d_n = 0$  si  $z_n = 0$

$$SP_{means} = \min_{z_n, d_n, c_d} \sum_{n=1}^N d_n \quad (50)$$
$$s.c : \forall n, \quad d_n \geq \|x_n - c\|^2 - M_n(1 - z_n)$$

Non linéaire, mais Cplex et Gurobi peuvent le résoudre de manière exacte grâce à la convexité

Ici, la décomposition de Dantzig-Wolfe s'adapte à un cas non linéaire, on a décomposé selon l'axe des  $K$  clusters, les sous problèmes ont la taille  $1/K$  du problème original.

# Résolution heuristique du pb de pricing, cas K-means

Heuristique type Lloyd : itérer opérations d'ajout/retrait de points au cluster courant, et la réactualisation du centroïde, deux phases qui améliorent la fonction objectif.

Initialement, on a un sous ensemble de points, et le calcul de son centroïde  $c$ .

On améliore la solution courante en ajoutant  $x_n$  non sélectionné dans solution courante si  $\|x_n - c\|^2 < \pi_n$ , et on améliore encore en réactualisant le centroïde  
...

On peut aussi retirer un point de la solution courante si  $x_n$  si  $\|x_n - c\|^2 > \pi_n$

⇒ Heuristique strictement améliorante en 2 phases : ajout/suppression de points et réactualisation de centroïdes, s'arrête quand aucune amélioration n'est apportée

⇒ Comment l'initialiser ? A quels points appliquer cette recherche locale ?

N.B : par RL, on peut trouver plusieurs sous ensembles générant au final une colonne de coût réduit négatif, peut être intéressant pour ne pas les régénérer par la suite.

# Initialisations naïves de RL de pricing K-means

Aléatoire, type GRASP ? à moins d'avoir bcp de chances, à éviter ...

Un point  $x_n$  a d'autant plus de chances d'être sélectionné que sa valeur duale  $\pi_n$  est élevée.

Algo glouton sur les poids  $\pi_n$  ? Ne tient pas en compte des distances entre points. Idéalement, des points proches, avec des fortes valeurs  $\pi_n$ . Différente propriété des initialisations de type Forgý pour k-means.

On part d'un point  $x_n$ , avec  $\pi_n > 0$ , alors  $c = x_n$ , et on itère ... N telles initialisations (ou se restreindre aux poids  $\pi_n$  les plus forts), peut se paralléliser comme RL multi-start.

Pb : on peut rester sur des singletons, minimums locaux peu intéressants ...

⇒ Pas évident de définir des bonnes stratégies

# Calculs de plusieurs bons minimums locaux

Rappel : Générer plusieurs colonnes de coût réduit négatif a un effet bénéfique sur la stabilisation.

Comment sortir d'un bon minimum local pour trouver d'autres bons minimums locaux ?

Idées classiques de diversifications de recherche locale :

- ▶ Recherche taboue : par exemple liste taboue avec points à ne pas considérer
- ▶ Recherche locale itérée
- ▶ Adaptative Large Neighborhood Search, opérateur "destroy" pour "destroy & repair"

# Résolution exacte de sous problème, K- $\alpha$ -med (1)

$$SP = \min_{p \in \mathcal{P}} c_p - \sum_n \pi_n z_n \quad (51)$$

Qu'on choisisse l'une ou l'autre des deux formulations PLNE présentées, on arrive à des formulations similaires :

Variables binaires :

- $z_n \in \{0, 1\}$  :  $z_n = 1$  ssi le point  $x_n$  est affecté au cluster
- $y_{n,n'} \in \{0, 1\}$  :  $y_{n,n'} = 1$  ssi le point  $x_n$  est affecté au cluster et que  $x_{n'}$  en est le centre (median, medoid ou extension  $\alpha$ )

$$\begin{aligned} \min_{y,z} \quad & \sum_{n,n'} d(x_n, x_{n'})^\alpha y_{n,n'} - \sum_n \pi_n z_n \\ \text{s.c. :} \quad & \sum_n y_{n,n} = 1 \\ & \forall n, n' \quad y_{n,n'} \leq z_{n'} \\ & \forall n, n' \quad y_{n,n'} \leq z_n \end{aligned} \quad (52)$$

Ici, la décomposition de Dantzig-Wolfe pour résoudre un PLNE avec  $O(N^2)$  variables et contraintes (en utilisant la meilleure formulation), itérerait avec des problèmes de taille  $O(N^2)$  variables et contraintes ? On peut faire mieux ...



## Résolution exacte de sous problème, K- $\alpha$ -med (2)

$$SP = \min_{p \in \mathcal{P}} c_p - \sum_n \pi_n z_n \quad (53)$$

On distingue les  $N$  cas, en fixant le représentant, centre de cluster.

Soit  $n'$  le centre de cluster fixé

$$SP_{n'} = \min_z \sum_{n \neq n'} d(x_n, x_{n'})^\alpha z_n - \sum_{n \neq n'} \pi_n z_n - \pi_{n'} \quad (54)$$

On a  $SP = \min_{n'} SP_{n'}$  et de plus  $SP_{n'}$  est un problème d'optimisation linéaire sans contrainte, sa solution est triviale, on prend  $z_n = 1$  ssi  $\pi_n < d(x_n, x_{n'})^\alpha$

SP se résout alors exactement en temps  $O(N^2)$ , on est dans les hypothèses pour le théorème de Lovas-Schrijver !

En temps  $O(N^2)$ , on a même pu potentiellement générer  $N$  colonnes de coût réduit négatif, pour un impact bénéfique en stabilisation.

Si en  $O(N)$  on a trouvé des bonnes solutions améliorantes, aps besoin de tout itérer. On peut parcourir les  $x_n$  selon  $\pi_n$  décroissant pour une telle heuristique.

# De K-medoids à K-means ?

L'algorithme de pricing précédent en  $O(N^2)$  peut définir un pricing heuristique pour K-means en recalculant le centroïde et le coût du cluster du cluster définit par K-medoids.

C'est exactement ce que faisait la RL initialisée à un point  $x_{n'}$ , le critère d'ajouter  $x_n$  ssi  $\pi_n < d(x_n, x_{n'})^2$  correspond à la première itération de RL !

Différence majeure,  $O(N^2)$  pour la résolution exacte du sous-problème K-medoids, pas utilisable pour résolution exacte de K-means (à part initialisation comme démarrage à chaud de la RL)

# Plan

Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

**Applications pratiques, résultats**

Cutting Stock Problem (CSP)

Problème de coloration d'un graphe "Vertex coloring"

Problèmes de tournées de véhicules (VRP)

Problèmes de clustering

Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions

# Rappel : Unit Commitment Problem, UCP

UCP : Planifier la production d'électricité pour les différentes unités  $u \in \mathcal{U}$  de production à tout instant  $t \in \mathcal{T}$ , en satisfaisant les demandes en électricité et en minimisant les coûts de production.

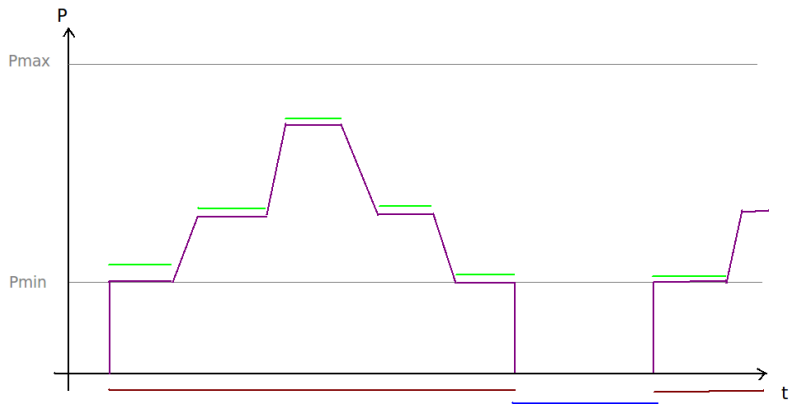
Fonction de coût : coûts de productions proportionnel à la production  $C_{prop}^u$  et coûts fixes de fonctionnement  $C_{fix}^u$  pour tout  $u \in \mathcal{U}$ .

Contrainte de demande : à tout instant  $t \in \mathcal{T}$ , la demande en puissance  $D_t$  (connue) doit être égale à la production totale à  $t$  sur toutes les unités  $u \in \mathcal{U}$ .

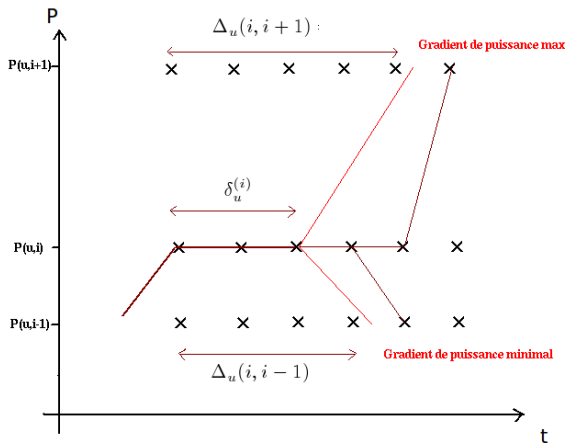
Contrainte de production : pour toute unité  $u \in \mathcal{U}$ , la production est soit nulle si l'unité est hors fonctionnement, soit entre  $Pmin_t^u$  et  $Pmax_t^u$  à l'instant  $t$ .

$\implies$  modèle simple de production d'électricité, à raffiner suivant cas d'étude

# discrete Unit Commitment Problem (UCPd)



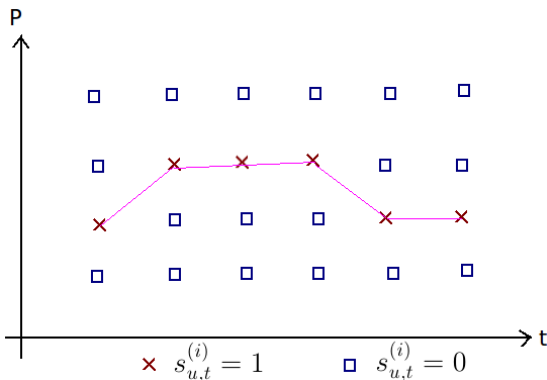
# Minimum stops constraints for UCPd



$$\Delta_u^{(i)+} = \left[ \delta_u^{(i)} + \frac{P(u, i+1) - P(u, i)}{|grad_{montee}(u)|} \right], \Delta_u^{(i)-} = \left[ \delta_u^{(i)} + \frac{P(u, i) - P(u, i-1)}{|grad_{baisse}(u)|} \right].$$

# Définition de variables sur UCP thermal discrétisé

Variables d'états  $s_{u,t}^{(i)}$  :  $s_{u,t}^{(i)} = 1$  ssi l'unité  $u$  produit à  $t$  sur le point de fonctionnement  $i$ .



variable de set up de l'UCP :  $x_{u,t} = \sum_{i \in \mathcal{I}_u} s_{u,t}^{(i)} \leq 1$ .

# Références

Travail de formulation PLNE compact :

[https://www.researchgate.net/publication/307617551\\_Tighter\\_MIP\\_formulations\\_for\\_the\\_discretised\\_unit\\_commitment\\_problem\\_with\\_min-stop\\_ramping\\_constraints](https://www.researchgate.net/publication/307617551_Tighter_MIP_formulations_for_the_discretised_unit_commitment_problem_with_min-stop_ramping_constraints)

Reformulation de Dantzig-Wolfe avec résolution polynomiale de sous-problème par programmation dynamique :

N. Dupin, Column generation for the discrete UC problem with min-stop ramping constraints. IFAC Papers Online, 52 (13), pp 529-534, 2019.  
<https://doi.org/10.1016/j.ifacol.2019.11.186> (open access)



# Bilan UCPd

- ▶ Point positif : le travail en reformulation compacte sur la modélisation PLNE des contraintes de paliers. A prouver, les points extrêmes du polyèdre sont entiers, ça étendrait des résultats sur min-up/min-down polytope . . .
- ▶ La difficulté des instances vient des contraintes de sac à dos pour les demandes.. Dans ce cas, une résolution PLNE frontale est assez efficace.
- ▶ Résultats dus à la modélisation des contraintes, donnant de bons résultats polyédraux, différent sur le vrai problème. (ici juste thermique à flamme, problème plus facile)

# Plan

Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

Applications pratiques, résultats

- Cutting Stock Problem (CSP)

- Problème de coloration d'un graphe "Vertex coloring"

- Problèmes de tournées de véhicules (VRP)

- Problèmes de clustering

- Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions

# Branch and Price

- ▶ Branch and Bound ou les bornes à chaque noeud sont les bornes calculé par reformulation de Dantzig Wolfe.
- ▶ Calculs à chaque noeud, démarrage à chaud avec colonnes déjà générées. (comme simplexe dual qui permet aussi le warmstart)
- ▶ Difficultés : apres branchement, le calcul de DW peut se retrouver infaisable, sans que le noeud soit infaisable. Pénalisations.

# Branch and Price vs Branch and Bound

- Affaire de compromis : gain borne duale/temps de calcul à chaque noeud.
- Cas où la borne lagrangienne vaut la borne LP : pas d'intérêt à utiliser le Branch and Price.

# Branchements en Branch and Price

- ▶ Brancher sur les variables de la reformulation de DW mène à des arbres de branchements non équilibrés.
- ▶ Stratégie usuelle efficace : brancher sur l'équivalent des variables de la formulation compacte.
- ▶ Pour variable compacte  $z_i = \sum_q z_{i,q} \lambda_q$ , de valeur fractionnaire calculée  $\tilde{z}_i$ .
- ▶ Branchements avec les contraintes  $\sum_q z_{i,q} \lambda_q \geq \lceil \tilde{z}_i \rceil$  et  $\sum_q z_{i,q} \lambda_q \leq \lfloor \tilde{z}_i \rfloor$

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1), 111-128.

Vanderbeck, F. (2011). Branching in branch-and-price : a generic scheme. *Mathematical Programming*, 130(2), 249-294.

# Heuristiques primales en branch and Price

- ▶ Retrouver des solutions entières avec un calcul MIP sur les colonnes générées est insuffisant en général.
- ▶ Heuristiques primales adaptées au Branch and Price : "diving heuristics", "rounding heuristics".
- ▶ C. Joncour, S. Michel, R. Sadykov, D. Sverdlov, F. Vanderbeck, *Column Generation based Primal Heuristics*, Electronic Notes in Discrete Mathematics 36 (2010) 695–702
- ▶ M. Lübbecke, C. Puchert *Primal Heuristics for Branch-and-Price Algorithms*, 2010

# Implémentation générique de reformulation DW

- ▶ En spécifiant dans un PLNE compact les contraintes à dualiser et les blocs indépendants et leur multiplicité, on peut déployer génériquement un reformulation DW et la résolution par algorithme de Branch & Price.
- ▶ Sous problèmes soit automatique par calcul MIP, soit par un solveur personnel adapté au sous problème spécifique (programmation dynamique ...).
- ▶ Génération de colonne, sa stabilisation, Heuristiques primales sont alors génériques.
- ▶ En C++, GCG (de la suite SCIP) le permet (université Aachen) : <https://gcg.or.rwth-aachen.de/>
- ▶ En C++, BaPCod, (université Bordeaux) : Sadykov, R., & Vanderbeck, F. (2021). BaPCod-a generic branch-and-price code (Doctoral dissertation, Inria Bordeaux Sud-Ouest).  
<https://hal.inria.fr/hal-03340548/document>
- ▶ Sous Julia, Coluna remplace BaPCod, (université Bordeaux) : <https://github.com/atoptima/Coluna.jl>

# Branch and Cut and Price

- Génération de colonnes et de coupes.
- Coupes d'intégrité (ex : cliques, couvertures ...) permettent d'améliorer la borne duale.
- Difficulté : les variables duales des coupes doivent être remontées dans les sous problèmes, change la nature du sous problème pour une résolution spécifique.

R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragao, M. Reis, E. Uchoa, R. Werneck, *Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem* Mathematical Programming, 2006, Vol 106, Issue 3, pp 491-511

J. Desrosiers, M. Lübbecke, *Branch-Price-and-Cut Algorithms* , 2010,

Sadykov, R. (2019). Modern branch-cut-and-price (Doctoral dissertation, Université de Bordeaux). <https://hal.inria.fr/tel-02410101/document>



# Plan

Relaxation Lagrangienne

Algorithme de Génération de Colonnes

Reformulation de Dantzig-Wolfe

Applications pratiques, résultats

- Cutting Stock Problem (CSP)

- Problème de coloration d'un graphe "Vertex coloring"

- Problèmes de tournées de véhicules (VRP)

- Problèmes de clustering

- Production d'électricité : Unit Commitment Problem (UCPd)

Résolution en nombres entiers

Conclusions