M415 : La Programmation Linéaire comme méthode d'Approximation

S. PERENNES.

DUT INFO - IUT Nice Côte d'Azur

29 avril 2022

Le Retour du Sac

Données : O ensemble d'objets de volume prix $(V_o, P_o), o \in O$, V volume du sac.

Heuristique pour le sac

- Prendre les objets cher idiot (cher mais gros)
- Prendre les objets petit idiot (petit mais sans valeur)
- **Bonne idée** : regarder le rapport valeur sur volume.

Efficacité d'un objet : $Eff_o = \frac{P_o}{V_o}$ c est le revenu par unité de volume.

Propriété

$$Opt \leq V \times max_{o \in O} Eff_o$$

Le Retour du Sac (bis)

Heuristique pour le sac

- Trier les objets par efficacité
- Parcourir les objets et les placer dans le sac si il reste de la place.

Propriété

Tant que tout objets passent le sous sac est optimal

Supposons que les objets $1, 2 \ldots, i$ passent. Ils occupent le volume $V' = \sum_{j \in [1,i]} V_i \leq V$. Le volume V_i est occupé avec la plus grande efficacité possible. L'optimal pour un sac de volume V' est donc $P' = \sum_{j \in [1,i]} P$.

Conséquence

Si les objets sont petits, i.e $V_o \leq V/k$ (pour k grand) c est presque optimal.

Remarques

Approche

- On a considéré la procédure comme continue.
- On a juste utilisé des notions de contrainte physique
- Nous avons négligé le coté discret.

Idée

Se ramener à un problème continu, considérer les objets comme fractionnables.

Une formulation mathématique

Formulation discrète

Maximiser $\sum_{o \in S} P_o$ (Prix de la solution)

Avec $S \subset O$

- Sous la contrainte $\sum_{o \in S} P_o \leq V$ (Contrainte de Volume)

Formulation avec des variables

- On pose $x_o = 1$ si $o \in S$, $x_o = 0$ si $o \notin S$.

  Maximiser $\sum_{o \in O} x_o \times P_o$ (Prix de la solution)
- Sous la contrainte $\sum_{o \in O} x_o \times P_o \leq V$ (Contrainte de Volume)

Exemple

Objets ensemble de couples (Prix, Volume)
0 : (2,3), 1 : (5,5), 2 :(8,4) 3 :(6,4) 4 :(4,3), 5 :(7 :2)
Volume de 15
Système :

$$\begin{array}{cc} \text{Maximiser} & 2x_0 + 5x_1 + 8x_2 + 6x_3 + 4x_4 + 5x_5 \\ \text{avec} & 3x_0 + 5x_1 + 4x_2 + 4x_3 + 3x_4 + 2x_5 \leq 15 \\ \text{et} & x_i \in \{0, 1\}, i \in 1, \ldots 5 \end{array}$$

Système Linéaire ! Discret

- l'objectif (le coût) est linéaire en les variables.
- Il y a une contrainte linéaire aussi, celle de volume.
- Attention les variables sont Discrètes et valent 0 ou 1.

La Relaxation Fractionnaire

Principe

On remplace la condition $x_i \in \{0,1\}$ par $x_i \in [0,1]$

0 : (2,3), 1 : (5,5), 2 :(8,4) 3 :(6,4) 4 :(4,3), 5 :(7 :2)

Volume de 15

Système :

$$\begin{array}{rc} \text{Maximiser} & 2x_0 + 5x_1 + 8x_2 + 6x_3 + 4x_4 + 5x_5 \\ \text{avec} & 3x_0 + 5x_1 + 4x_2 + 4x_3 + 3x_4 + 2x_5 \leq 15 \\ \text{et} & x_i \geq 0, i \in 1, \ldots, 5 \\ \text{et} & x_i \leq 1, i \in 1, \ldots, 5 \end{array}$$

Système (d'Inéquations) Linéaire

- l'objectif (le coût) est linéaire en les variables.
- Il y a une contrainte linéaire principale de volume.
- Pour chaque variable il y a deux contrainte ≥ 0 et ≤ 1.

La programmation linéaire

Principe

- C'est comme les systèmes d'équations classiques avec des $=$ mais on autorise $\leq, \geq, =$.
- On veut minimiser ou maximiser une expression linéaire, type $x + 2y + 5z$

Intérêt

1. Il existe des algorithme efficaces (Polynomiaux).
2. Avec un ordinateur on peut résoudre des systèmes avec des millions de variables et de contraintes.
3. Pour une expression $3x = 5y + 4z - 10w \leq 19$ on parle de *contrainte linéaire*.
4. l'expression a optimiser (min ou max) est appelée objectif.

La programmation linéaire

Utilisation

- On utilise un solveur ou une bibliotèque.
- Solveurs populaires commerciaux : (Cplex, Gurobi) (centaines de milliers d'heures de devellopement), solveurs libres (lpsolve, coin)
- API avec Java, Python, etc.
- Cplex et Gurobi sont gratuits pour les étudiants et chercheuses.

La programmation linéaire Entière

Quid du problème initial ?

- Pour notre problème de sac à dos on ne cherche pas une solution fractionnaire, car $x_o = \frac{1}{3}$ est ambigu. On ne peut pas prendre $\frac{1}{3}$ de l'objet. On peut cependant interpréter cela comme prendre o avec 1 chance sur 3.
- nous cherchons en fait une solution entière au problème de programation linéaire.
- On parle de **MIPS** (Mixed Integral Programing System, Système linéaire en variable mixtes). Mixte car certaines variables sont entières et d'autres non (fractionnaires).

Les solveurs précités resolvent aussi les MIPS. Pour ce faire ils combinent un solveur linéaire très performant + des arbres de recherche, de la *génération de contraintes* + plein d'astuces et d'heuristiques.

Formulation du Sac à Dos sous forme de MIPS

0 : (2,3), 1 : (5,5), 2 :(8,4) 3 :(6,4) 4 :(4,3), 5 :(7 :2)
Volume de 15
Système :

$$
\begin{array}{rc}
\text{Maximiser} & 2x_0 + 5x_1 + 8x_2 + 6x_3 + 4x_4 + 5x_5 \\
\text{avec} & 3x_0 + 5x_1 + 4x_2 + 4x_3 + 3x_4 + 2x_5 \leq 15 \\
\text{et} & x_i \geq 0, i \in 1, \ldots, 5 \\
\text{et} & x_i \leq 1, i \in 1, \ldots, 5 \\
\text{et} & x_i, i \in 1, \ldots, 5 \text{ est Entière}
\end{array}
$$

MIPS

On peut saisir directement ce système dans un solveur de MIPS et résoudre ainsi assez efficacement le problème du sac à dos.

Utilisation de la Prog. Linéaire Entière.

Deux démarches :

1. Trouver une *Bonne* formulation du problème.
2. Étude fine de la structure de la formulation $\rightarrow$ algorithme Ad-hoc dont on peut prouver une certaine efficacité.

1. Une *Bonne*[a]. formulation signifie peu de contraintes et de variables mais une bonne approximation du système entier *MIPS* par le système relaché (sans les contraintes d'integrité).
2. C'est ad-hoc et souvent (très) difficile

a. En pratique c'est une formulation qui fonctionne bien

Avantages

- Générique.
- Une fois la formulation trouvée : On peut utiliser les solveurs puissants et aboutis comme boites noires.

Un second exemple de formulation

Le sac à doc bi-dimensionel

- Le sac offre un volume V mais supporte un poids maximal W.
- Les objets sont des triplets $(V_i = Volume, W_i = Poids, P_i = Prix)$
- On veut maximiser le prix du sac.

Le système est quasi identique, mais il y a deux contraintes principales (volume, poids) :

$$\begin{array}{cc}
\sum_{o \in O} x_o V_o \leq V & \text{Volume ok} \\
\sum_{o \in O} x_o W_o \leq W & \text{poids ok} \\
\text{Maximiser } \sum_{o \in O} x_o P_o \leq V & (\textit{Objectif}) \\
\forall i \in O & x_i \geq 0 \\
\forall i \in O & x_i \leq 1, \\
\forall i \in O & x_i \text{ est Entière}
\end{array}$$

Formulation pour le problème de couverture

Le problème de couverture

- Entrée : Un ensemble S et une liste L de sous ensembles de S indexés par I ; donc $L = \{S_i, i \in I\}$
- Trouver le plus petit nombre d'ensembles dont l'union est S

- en notation mathématique : $Min\{|J|, J \subset I \mid \cup_{j \in J} S_j = S\}$.

Formulation

- Une variable par ensemble S_i, x_i étant associée à S_i $(i \in I)$.
- une contrainte par élément s de S imposant que s soit couvert.

  Donnée : On doit connaitre les ensembles $S_i, i \in I$
  $\forall s \in S, \forall i \in I : a(s,i) = 1$ si $s \in S_i$, 0 sinon.

Formulation pour le problème de couverture (II)

Système :

$x_i = 1$ signifie que S_i est choisi ($i \in J$), sinon $x_i = 0$ ($i \notin J$)

$$\begin{array}{ll} \forall s \in S, \sum_{i \in I} x_i a(s,i) \geq 1 & s \text{ est couvert} \\ \sum_{j \in I} x_j & \text{Minimiser } |J| \\ \forall i \in I, x_i \leq 1 & \\ \forall i \in I, x_i \geq 0 & \\ \forall i \in I & x_i \text{ est Entière} \end{array}$$

1. La première ligne dit que pour tout sommet s pour au moins un $i \in I$ un ensemble contenant s (ie $a(s,i) = 1$) est choisi ($x_i = 1$). Autrement dit $s \in \cup_{i \in J} S_i$. (contrainte de couverture).
2. La seconde ligne ne fait que compter le cardinal de J.
3. les 3 dernières contraintes imposent que $x_i \in \{0, 1\}$.

Formulation pour le problème de couverture (exemple)

Données

- $S = \{1, 2, 3, 4, 5, 6, 7\}$
- Il y a 5 sous ensembles : $S_a = \{1, 2, 3, 7\}, S_b = \{2, 3, 4\}, S_c = \{2, 4, 7\}, S_d = \{1, 5, 6\}, S_e = \{2, 4, 6\}$
- donc $I = \{a, b, c, d, e\}$.
- Il y a 7 sommets donc 7 contraintes de couverture.

sommet	equation	
1	$x_a + x_d \geq 1$	$1 \in S_a, 1 \in S_d.$
2	$x_a + x_b + x_c + x_e \geq 1$	$2 \in S_a, 2 \in S_c, 2 \in S_e.$
3	$x_a + x_b \geq 1$	...
4	$x_b + x_c + x_e$	...
5	$x_d \geq 1$	...
6	$x_d + x_e \geq 1$	...
7	$x_a + x_c \geq 1$	...

Formulation pour le problème de couverture (exemple suite)

Données

- $S = \{1, 2, 3, 4, 5, 6, 7\}$
- Il y a 5 sous ensembles : $S_a = \{1, 2, 3, 7\}, S_b = \{2, 3, 4\}, S_c = \{2, 4, 7\}, S_d = \{1, 5, 6\}, S_e = \{2, 4, 6\}$
- donc $I = \{a, b, c, d, e\}$.
- Il y a 7 sommets donc 7 contraintes de couverture.

On ajoute :

$$\textit{Minimiser} : x_a + x_b + x_c + x_d + x_e$$

et

$$\begin{array}{ll} x_i, i \in \{a, b, c, d, e\} & \geq 0 \\ x_i, i \in \{a, b, c, d, e\} & \leq 1 \\ x_i, i \in \{a, b, c, d, e\} & \text{est entier} \end{array}$$

Formulation pour le problème du rendu de monaie

Données

- 3 type de pièces : $\{1, 2, 5\}$ euros.
- 3 nombres de pièces maximum n_1, n_2, n_5.
- une somme S à réaliser.
- On veut minimiser le nombre de pièce utilisées.

Système

- 3 variables x_1, x_2, x_5, 1 vraie contrainte (la somme).
- un objectif (le nombre de pièces), des contraintes de domaine.

$x_1 + 2x_2 + 5x_5 = S$	contrainte de somme
Minimiser :	$x_1 + x_2 + x_5$
$x_i \geq 0, i \in \{1, 2, 5\}$	
$x_i \leq n_i, i \in \{1, 2, 5\}$	nombre de pièces max
$i \in \{1, 2, 5\}, x_i$	est entière

Formulation de la composition de repas (Cook problem)

Données

- Un ensemble A d'aliments.
- Un ensemble de nutriments (sucre, lipide, proteines, ...) ou qualités (poids, calories) N.
- pour chaque aliment $a \in A$ et nutriment $n \in N$ la quantité de nutriments $q(a, n)$ présente par gramme de a
- Pour chaque nutriment $n \in N$ deux bornes $min(n), max(n)$, un repas est équilibré si $\forall n \in N$ la quantité de n est dans l'intervale $[min(n), max(n)$.

Composition du repas

Le cuisinier doit déterminer la quantité x_a à utiliser pour chaque aliment $a \in A$.

Composition de repas (Cook problem, suite)

Ceci s exprime par le système suivant :

Système

$$\begin{aligned} \forall n \in N, \sum_{a \in N} x_a q(a,n) &\leq max(n) \\ \forall n \in N, \sum_{a \in N} x_a q(a,n) &\geq min(n) \\ \forall a \in A, x_a &\geq 0 \end{aligned}$$

Notons qu'en premier lieu n'y a pas d'objectif on cherche simplement un repas acceptable [1], (faisable, feasible).
Ici le système est en pratique vraiment *mixte* :

- Certaines variables $x_a, a \in A$ peuvent être fractionnaires (sucre, lait, ...)
- d'autres peuvent être entières (2 portions, 1 yahourt, ...)

On considère parfois le prix, $p(a), a \in A$ et donc l'objectif :

$$\text{Minimiser} \sum_{a \in a} x_a p(a)$$

1. repectant les contraintes diététiques

Plus court chemins (le retour)

Données : un graphe $G = (V, E)$ et une longueur positive l pour chaque arête $e \in E$.

Contexte

- Il existe un algorithme efficace spécifique (Djistra).
- La formulation simple sous forme de système linéaire.
- Formulation est importante pour résoudre les problèmes de routage (dualité, génération de chemin, voir path generation).
- elle fournit un exemple interessant en soi.

Le sommet v_0 est la *source* des chemins. $\forall u \in V$ on introduit la variable $dd(u) = d(u, v_0)$, dans la solution $dd(u)$ sera la distance $d(u) = d(u, v_0)$.

Propriétés

- $d(v_0) = 0$
- si $e = [u, v]$ connecte u et v alors $d(u) \leq d(v) + l(e)$

Plus courts chemins : système

Système

$$\begin{aligned} dd(v_0) &= 0 \\ \forall u \in V, \forall e = (u,v): \quad dd(v) &\leq dd(u) + l(e) \\ \text{Maximiser } dd(u_0) & \end{aligned}$$

Remarques

- Si on omet l'objectif de maximiser on trouve de solutions où $dd(u) < d(u)$.
- Dans le système ci dessus, on peut simplement affirmer que $dd(u) = d(u)$ pour $u = u_0$ ou le long d'un plus court chemin reliant v_0 à u_0. Ailleurs on peut avoir $dd(u) < d(u)$.
- Si on remplace l'objectif par *Minimiser* $\sum_{u \in V} dd(u)$ alors pour l optimal on aura $dd(u) = d(u)$.