Quantitative Energy Economics

INMA2415 - Project

Université catholique de Louvain
Faculté des Sciences Appliquées
Département d'Ingénierie Mathématique

Laterre Alexandre (49-25-11-00)
Laurent Quentin (48-34-11-00)
Année Académique 2014-2015
Professeur : Anthony Papavasiliou

UCL Université catholique de Louvain

LINMA2491 - Operational Research
Homework 3

Laterre Alexandre
noma 49-25-11-00

May 17, 2015

1 Question A

Les feasible cuts n'ont pas de raisons d'être dans le problème que nous essayons de résoudre. La seule contrainte faisant intervenir la first-stage decision étant :

$$\sum_{j=1}^{M} y_{ij} \leq x_i \qquad \forall i = 1 \cdots N$$

Or, il existe toujours un y qui satisfasse cette contrainte pour autant que les valeurs x_i soient non-négatives, ce qui sera toujours le cas.

2 Question B

Passons maintenant à l'algorithme L-Shaped. Une explication générale sera d'abord donnée, pour ensuite appliquer celle-ci à notre cas. On peut écrire la formulation d'un problème linéaire two-stage comme suit :

$$\min_{x \in \mathcal{X}} \quad c^{\mathrm{T}} x + \mathfrak{Q}(x) \tag{1}$$

avec $\mathcal{X} = \{x \in \mathbb{R}^{n_1} \ : Ax = b,\ x \geq 0\}$, $\mathfrak{Q}(x) := \mathbb{E}_{\xi}\left[Q(x, \xi)\right]$ et $Q(x, \xi)$ est la valeur optimale du second-stage problem :

$$\min_{y \in \mathbb{R}^{n_2}} \quad q^{\mathrm{T}} y \text{ s.t. } Tx + Wy = h,\ y \geq 0$$

En supposant que les données contenues dans le vecteur ξ sont formées des éléments de q, h, T, W. La méthode consiste à résoudre une approximation du problème (1) à l'aide d'une représentation linéaire de $\mathfrak{Q}$. Cette approximation est appelée le *Master Program* et prend l'allure suivante :

$$\begin{aligned}
\min \quad & c^{\mathrm{T}} x + \theta && && (2)\\
& Ax = b && && (3)\\
& D_l x \geq d_l && l = 1, \ldots, r && (4)\\
& E_l\, x + \theta \geq e_l && l = 1, \ldots, s && (5)\\
& x \geq 0 && \theta \in \mathbb{R} && (6)
\end{aligned}$$

avec r, le nombre de *feasibility cuts* et s, le nombre d'*optimality cuts*. L'algorithme consiste ensuite à résoudre le master et d'envoyer la valeur x trouvée au second-stage. Deux types de contraintes peuvent ensuite être ajoutées.

feasibility cuts qui vont déterminer si $x \in \{x | \mathfrak{Q}(x) < +\infty\}$. (4)

optimality cuts qui sont des approximation locales de $\mathfrak{Q}$ sur son domaine. Sachant que $\mathfrak{Q}$ est linéaire par morceaux (en nombre fini), ces contraintes définissent de mieux en mieux $\mathfrak{Q}$. (5)

Au fur et à mesure des itérations, on va pouvoir petit à petit construire l'allure de la fonction $\mathfrak{Q}$ et de son domaine. Ce qui permettra de trouver une first-stage decision qui sera optimal au sens du problème d'origine (1). Pour plus de détails sur l'algorithme, voir chapitre 5 (B.L - Introduction to Stochastic Programming).

Application de l'algorithme L-Shaped au problème : Ressource Dispatching

Le *Master Program* et la *quality function Q* s'écrivent de la manière suivante :

$$\begin{aligned} \min_{x\geq 0} \quad & q\sum_{i=1}^{N} x_i + s \\ \text{subject to} \quad & x_i \leq c_i,\ i = 1,\dots,N \\ & E_l\, x + s \geq e_l,\ il = 1,\dots,\ s \end{aligned} \qquad \begin{aligned} \min_{y\geq 0} \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\,(t_{ij} - b) \\ \text{subject to} \quad & \sum_{i=1}^{N} y_{ij} \leq d_j^k,\ j = 1,\dots,M \quad (\mu_j^k) \\ & \sum_{j=1}^{M} y_{ij} \leq x_i,\ i = 1,\dots,N \quad (\nu_i^k) \end{aligned}$$

Le *Master Program* étant rendu très simple en raison de l'absence des *feasibility cuts*. En ce qui concerne les *optimality cuts* celles-ci prennent la forme suivante :

$$E_l = -\sum_{k=1}^{K} p_k\,\nu^k \qquad e_l = \sum_{k=1}^{K} p_k \sum_{j=1}^{M} \mu_j^k$$

E_l correspond donc à moins la moyenne des ν^k et e_l correspond à la moyenne de la somme de μ_j^k. On peut noter la contrainte de la manière suivante également :

$$\theta \geq \sum_{k=1}^{K} p_k \left[\sum_{j=1}^{M} \mu_j^k + \sum_{i=1}^{N} \nu_i^k x_i\right]$$

3 Question C

L'implémentation de la méthode est disponibles dans les fichiers ampl : *main.run* et *RessourceDispatching.mod*. Sachant que un code de base nous avait été gracieusement fourni, je ne vais pas plus détailler celui-ci sachant qu'il n'y a pas de difficultés majeures. Les résultats obtenus pour les différentes données sont disponibles sur les graphiques [1] [2] et [3]. On remarque immédiatement que le first-stage trouvé par la méthode L-Shaped est le même que celui trouvé par le programme *valid.run* et *problem.mod* qui implémente le problème à l'état brut. La méthode L-Shaped est donc beaucoup plus efficace/adaptée que la méthode du sous-gradient qui donnait une solution first-stage très moyenne. Remarquons également un petite différence pour les données *Smalldatfile.dat*. La solution L-shaped n'est pas exactement la même que celle trouvée par le programme brut. Mais aucune inquiètude, le profit optimal trouvé est le même. Le L-shaped donne juste une autre solution optimale.
Pour ce qui est du nombre d'itérations, il est assez décevant de voir que dans le cas d'un seul scénario, on ait besoin de 37 itérations. Mais de manière général, c'est déjà beaucoup mieux que la méthode du sous-gradient qui mettait un plus grand nombre d'itérations pour de moins bons résultats. La méthode L-Shaped permet par exemple de résoudre le problème pour le *Bigdatfile.dat* en 15-20 secondes.

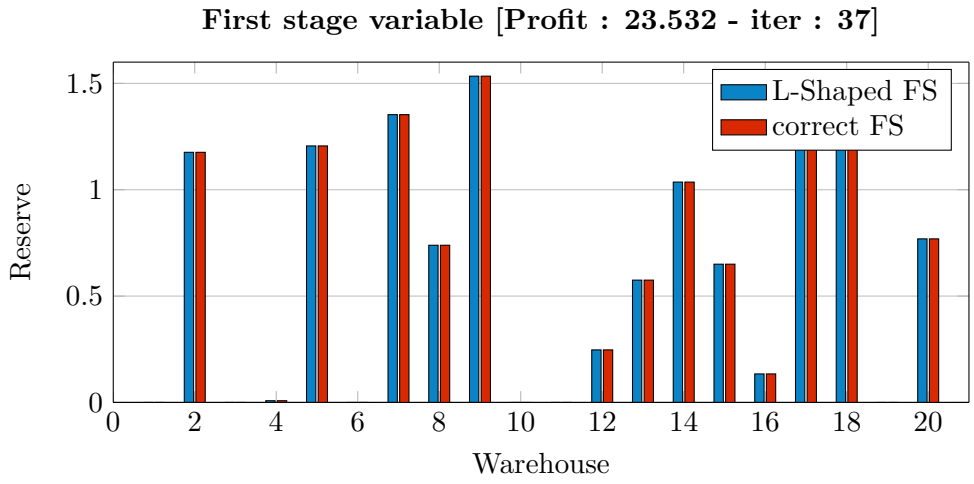

Figure 1: First stage solution - Verysmalldatfile.dat

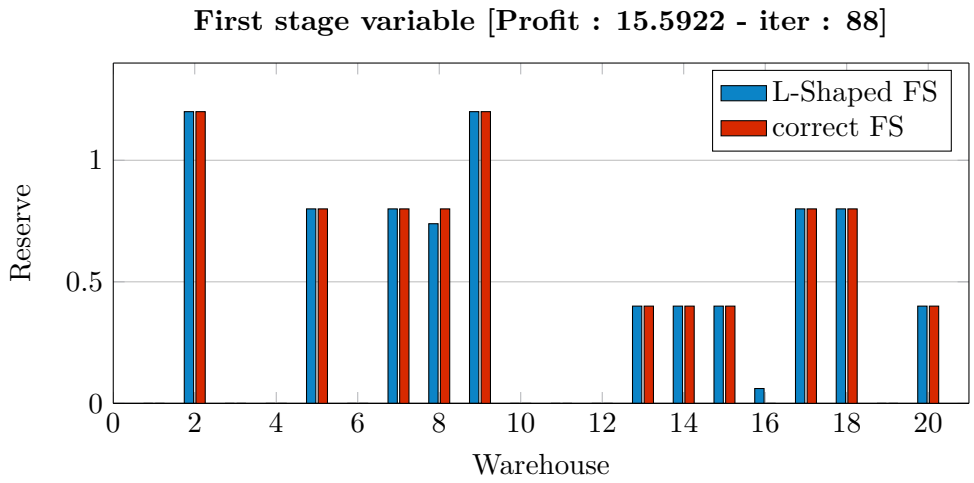

Figure 2: First stage solution - Smalldatfile.dat

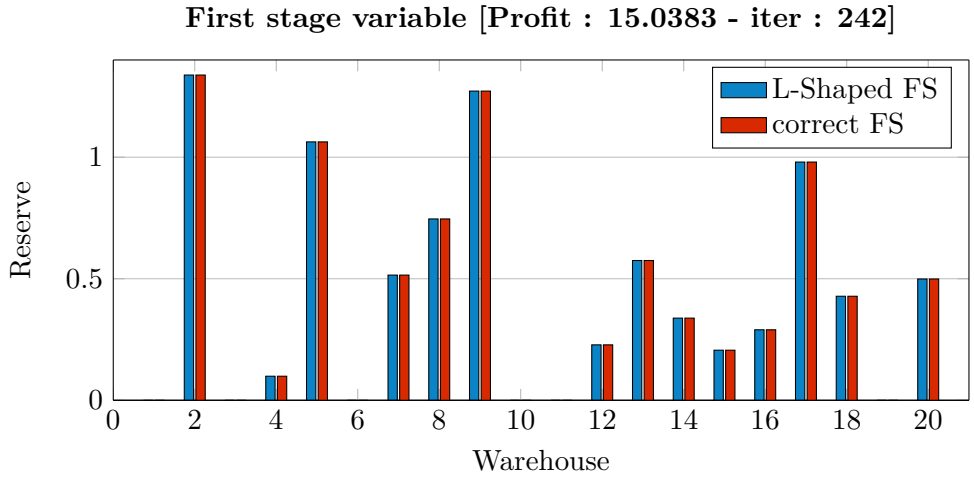

Figure 3: First stage solution - Bigdatfile.dat