

Problèmes relaxés du sac-à-dos stochastique statique

Stefanie Kosuch

Laboratoire de Recherche en Informatique (LRI)
Université Paris XI - Sud
Orsay, France
Email: Stefanie.Kosuch@lri.fr

Abdel Lisser

Laboratoire de Recherche en Informatique (LRI)
Université Paris XI - Sud
Orsay, France
Email: Abdel.Lisser@lri.fr

Abstract—Nous présentons deux variantes du problème du sac-à-dos stochastique statique. Le premier est un problème combinatoire sans contrainte connu sous le nom problème du sac-à-dos avec recours simple. La deuxième variante contient des contraintes stochastiques. Nous proposons trois méthodes différentes pour la résolution des problèmes relaxés correspondants : une pour le problème avec recours simple et deux pour le problème avec contrainte stochastique. Deux de ces méthodes demandent l'estimation du gradient d'une fonction indicatrice en espérance; nous présentons deux façons de procéder. Des résultats numériques sont données pour la résolution des problèmes relaxés et combinatoires.

I. INTRODUCTION

Le problème du sac-à-dos consiste à choisir un sous-ensemble d'objets qui maximise une fonction objectif. Plus précisément, nous supposons que chaque objet a un bénéfice ou bénéfice par unité de poids ainsi qu'un poids spécifique et notre but est de maximiser le bénéfice total en respectant une contrainte de capacité. Parmi les multiples applications possibles, on trouve des problèmes de transport, de finance ou la conception d'emploi du temps ("scheduling").

Dans le cas du problème déterministe, tous les paramètres (poids, bénéfices, capacité) sont connus. Cependant, beaucoup de problèmes réels présentent la difficulté que leurs paramètres ne sont pas déterminés a priori. Ces valeurs peuvent être modélisées par des variables aléatoires continues ou discrètes et le problème est transformé en *problème d'optimisation stochastique*. Comme le problème déterministe, le problème du sac-à-dos stochastique est au moins NP-difficile (voir [1]).

Dans cet article, nous considérons que les poids des objets suivent une loi normale dont la moyenne et l'écart-type sont connus. La capacité et le bénéfice par unité de poids des objets restent déterminés. On peut imaginer comme application une entreprise de logistique qui connaît ses capacités de transport mais qui n'a que des informations approximatives (fourni par les clients) sur la taille des objets à transporter. Une autre application serait l'ordonnancement : Il y a un ensemble de tâches possibles à traiter et chacune de ces tâches rapporte un gain spécifique. Le temps d'exécution de chaque tâche n'est pas connu à l'avance, alors que la période du fonctionnement de la machine est fixée.

Nous présentons deux variantes du problème du sac-à-dos stochastique statique. Le premier est un problème combinatoire sans contrainte connu sous le nom *problème du sac-à-*

dos avec recours simple. La deuxième variante contient des contraintes stochastiques.

Dans la section II, nous présentons les problèmes originaux, i.e. combinatoires. Cependant, dans cet article nous résolvons les relaxations de ces problèmes. Les résultats de cet article peuvent néanmoins servir à la résolution des problèmes combinatoires car les relaxations fournissent des bornes supérieures. Ces bornes peuvent par exemple être utilisées dans un algorithme du type "branch-and-bound" (voir e.g. [2] et [3]).

Dans la section III les problèmes continus sont, entre autres, résolus par des algorithmes du type gradient stochastique. Ceci nécessite le calcul (resp. l'estimation) du gradient des fonctions en espérance contenant une fonction indicatrice. Dans la section III-A nous présentons deux façons possibles de procéder (cf. [3]).

Les méthodes présentées dans cet article sont plus ou moins connues des spécialistes de l'optimisation stochastique continue. Par contre, d'après notre connaissance, c'est la première fois qu'elles sont utilisées pour résoudre un problème stochastique combinatoire, particulièrement pour le problème du sac-à-dos stochastique. Dans la section IV, nous allons présenter quelques résultats obtenus.

II. FORMULATIONS MATHÉMATIQUES

Nous considérons un problème du sac-à-dos statique sous la forme suivante : Nous avons n objets, chaque objet ayant un poids qui est inconnu au moment où la décision du choix des objets doit être prise. En conséquence, les poids sont modélisés comme variables aléatoires et nous considérons qu'ils suivent une loi normale. Plus précisément, nous associons à l'objet i la variable aléatoire distribuée normalement χ_i avec une moyenne μ_i et un écart-type σ_i . Par χ nous notons le vecteur n -dimensionnel correspondant. Le bénéfice par unité de poids $r_i > 0$ de l'objet i et la capacité c du sac-à-dos sont supposés déterministes. L'objectif est de maximiser le bénéfice total $E[\sum_{i=1}^n r_i \chi_i x_i]$ en respectant "le mieux possible" la capacité.

Nous considérons deux variantes du problème du sac-à-dos statique, la deuxième ayant deux formulations équivalentes :

- 1) **Problème du sac-à-dos avec recours simple (SRKP)**

$$\max_{x \in \{0,1\}^n} E\left[\sum_{i=1}^n r_i \chi_i x_i\right] - d \cdot E\left[[g(x, \chi) - c]^+\right] \quad (1)$$

- 2) **Problème du sac-à-dos avec contrainte (CKP)**

a) Problème du sac-à-dos avec contrainte en probabilité (CKP)

$$\max_{x \in \{0,1\}^n} \mathbb{E} \left[\sum_{i=1}^n r_i \chi_i x_i \right] \quad (2)$$

$$\text{s.c.} \quad \mathbb{P}\{g(x, \chi) \leq c\} \geq p \quad (3)$$

b) Problème du sac-à-dos avec contrainte en espérance mathématique (ECKP)

$$\max_{x \in \{0,1\}^n} \mathbb{E} \left[\sum_{i=1}^n r_i \chi_i x_i \right] \quad (4)$$

$$\text{s.c.} \quad \mathbb{E}[\mathbb{1}_{\mathbb{R}^+}(c - g(x, \chi))] \geq p \quad (5)$$

Ici $\mathbb{P}\{A\}$ est la probabilité que l'événement A se produit. $\mathbb{E}[\cdot]$ note l'espérance mathématique et $\mathbb{1}_{\mathbb{R}^+}$ la fonction indicatrice de l'intervalle réel non-négatif. En plus, $g(x, \chi) := \sum_{i=1}^n \chi_i x_i$, $[x]^+ := \max(0, x) = x \cdot \mathbb{1}_{\mathbb{R}^+}(x)$ ($x \in \mathbb{R}$), $d \in \mathbb{R}^+$ et $p \in (0.5, 1]$.

Ces deux approches se distinguent par l'interprétation de la violation de la capacité du sac-à-dos : Dans le cas du *SRKP*, pour toute unité de surcharge une pénalité d doit être payée. Le bon choix du paramètre d permet de contrôler la surcharge attendue. Par contre, le choix du paramètre p dans les contraintes équivalentes (3) et (5) du *CKP* restreint le pourcentage des cas où la capacité est excédée.

Par la suite, nous notons *vecteur de solution* tous les $x \in \mathbb{R}^n$ tel que $x = \arg \max_{x \in X_{ad}} J(x, \chi)$ où J est la fonction objectif d'un des problèmes présentés ci-dessus et $X_{ad} \subseteq \mathbb{R}^n$ est l'ensemble admissible.

Dans tout l'article, nous appelons f (resp. F) la fonction de densité (resp. de répartition) de la distribution normale avec moyenne 0 et écart-type 1.

III. MÉTHODES DE RÉOLUTION

Dans cette section nous présentons trois méthodes de résolution pour les relaxations continues des problèmes présentés, une pour chacune des trois variantes.

A. Le problème du sac-à-dos avec recours simple

Dans cette formulation du problème du sac-à-dos stochastique, la contrainte de capacité fait partie de la fonction objectif. Ceci est réalisé en introduisant une fonction de pénalité $[\cdot]^+$ ainsi qu'un facteur de pénalité $d > 0$. d peut être interprété comme une pénalité par unité de surcharge.

Il n'est pas difficile de démontrer que le problème (1) est concave. Ceci nous permet de le résoudre en utilisant un algorithme du type gradient stochastique. Un algorithme du type gradient stochastique est une combinaison de la méthode de Monte-Carlo et de la méthode du gradient bien connue en optimisation continue. Au lieu de travailler directement avec le gradient de la fonction objectif, l'espérance mathématique dans cette fonction est approximée par des tirages au sort de la variable aléatoire à chaque itération. Plus précisément, si la fonction objectif s'écrit $J(x, \chi) = \mathbb{E}[j(x, \chi)]$, nous utilisons à l'itération k le gradient $\nabla_{x,j}(x, \chi^k)$ où χ_k est obtenu par un

Algorithme du gradient stochastique

- Choisir x^0 dans $X_{ad} = [0, 1]^n$
- A l'itération $k + 1$, tirer $\chi^k = (\chi_1^k, \dots, \chi_n^k)$ suivant la loi normale de χ
- Mettre à jour x^k :

$$x^{k+1} = x^k + \epsilon^k r^k$$

où $r^k = \nabla_j(x^k, \chi^k)$ et $(\epsilon^k)_{k \in \mathbb{N}}$ est une σ -suite

- Pour tous les $i = 1, \dots, n$: Si $x_i^{k+1} > 1$ poser $x_i^{k+1} = 1$ et si $x_i^{k+1} < 0$ poser $x_i^{k+1} = 0$

Algorithme III.1: Fonction à maximiser : $\mathbb{E}[j(x, \chi)]$

tirage au sort. Comme, dans le cas des problèmes traités dans cet article, nous avons

$$\nabla_x \mathbb{E}[j(x)] = \mathbb{E}[\nabla_{x,j}(x)]$$

il existe une autre possibilité de procéder : D'abord, nous estimons le gradient $\nabla_x \mathbb{E}[j(x)]$. Ceci nous donne une fonction en espérance $\mathbb{E}[\tilde{j}(x, \chi)]$ tel que $\mathbb{E}[\tilde{j}(x, \chi)] := \mathbb{E}[\nabla_{x,j}(x, \chi)]$ pour tous les $x \in \mathbb{R}^n$, mais généralement $\tilde{j}(\cdot) \neq \nabla_x j(\cdot)$. Malgré tout, il est théoriquement possible d'utiliser \tilde{j} au lieu de $\nabla_x j$ dans l'algorithme.

Nous allons procéder d'une telle manière dans le cas de la méthode Intégration par parties. Cependant, le remplacement de $\nabla_{x,j}$ n'est généralement admissible que pour un grand nombre d'itérations et peut provoquer des fluctuations au début de l'algorithme (voir IV).

Dans le cas de *SRKP*, nous avons $j(x, \chi) = \sum_i r_i \chi_i x_i - d \cdot [g(x, \chi) - c]^+$. Il est certain que ni j , ni $\mathbb{E}[j(x)]$ ne sont différentiables. Nous présentons dans la suite deux méthodes qui servent à estimer ces gradients.

1) *Intégration par parties:* Dans sa thèse [4], Andrieu présente un théorème (Théorème 5.5) de Andrieu, Cohen et Vázquez-Abad qui nous a servi comme inspiration pour la proposition suivante. Le théorème original sera cité dans la section III-B2.

Proposition 1: Soit χ_i ($i = 1, \dots, n$) une variable aléatoire distribuée normalement et indépendamment avec une moyenne μ_i , un écart-type σ_i et une fonction de densité φ_i . Soit $\Gamma : (\mathbb{R}^+)^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ définie comme $\Gamma(x, \chi) := \mathbb{E}[[\vartheta(x, \chi)]^+]$ avec $\vartheta(x, \chi) := \sum_{i=1}^n \chi_i x_i - c$ et $c \geq 0$. En utilisant l'intégration par parties, nous obtenons pour un $k \in \{1, \dots, n\}$ arbitraire

$$\nabla_x \Gamma(x, \chi) = \mathbb{E} \left[2 \cdot [\vartheta(x, \chi)]^+ \chi \frac{c_k}{x_k} - \mathbb{1}_{\mathbb{R}^+}(\vartheta(x, \chi)) \chi - [\vartheta(x, \chi)]^+ \vartheta(x, \chi) \frac{c_k}{x_k} \nu_k \right]$$

où $c_k := \frac{(\chi_k - \mu_k)}{\sigma_k^2}$ et $\nu^k \in \mathbb{R}^n$ tel que $\nu^k_k = 1$ et $\nu^k_i = 0$ pour tous les $i \neq k$.

Démonstration: Nous avons

$$\Gamma(x, \chi) = \int_{-\infty}^{\infty} \mathbb{1}_{\mathbb{R}^+}(\vartheta(x, \chi)) \vartheta(x, \chi) \varphi(\chi) \, d\chi$$

avec $\varphi(\chi) := \prod_{i=1}^n \varphi_i(\chi_i)$.

Définissons (pour un $k \in \{1, \dots, n\}$ arbitraire)

$$\begin{aligned} u'_{\chi_k}(x, \chi) &:= \mathbb{1}_{\mathbb{R}^+}(\vartheta(x, \chi)) \vartheta'_{\chi_k}(x, \chi) & \text{et} \\ v(x, \chi) &:= \frac{\vartheta(x, \chi) \varphi(\chi)}{\vartheta'_{\chi_k}(x, \chi)} \end{aligned}$$

Il s'en suit

$$\Gamma(x, \chi) = \int_{-\infty}^{\infty} u'_{\chi_k}(x, \chi) v(x, \chi) \, d\chi$$

Soit $\mathbb{Y}_{\mathbb{R}^+}(\cdot)$ une primitive de $\mathbb{1}_{\mathbb{R}^+}(\cdot)$. Intégration par parties par rapport à χ_k donne

$$\begin{aligned} \Gamma(x, \chi) &= [u(x, \chi) v(x, \chi)]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} u(x, \chi) v'_{\chi_k}(x, \chi) \, d\chi \\ &= - \int_{-\infty}^{\infty} \mathbb{Y}_{\mathbb{R}^+}(\vartheta(x, \chi)) v'_{\chi_k}(x, \chi) \, d\chi \end{aligned}$$

En utilisant

$$\varphi'_{\chi_k}(\chi) = -\frac{(\chi_k - \mu_k)}{\sigma_k^2} \varphi(\chi) =: -c_k \varphi(\chi)$$

nous avons

$$v'_{\chi_k}(x, \chi) = \left(1 - \frac{c_k \vartheta(x, \chi)}{x_k}\right) \varphi(\chi)$$

et, par conséquent,

$$\begin{aligned} \Gamma(x, \chi) &= - \int_{-\infty}^{\infty} \mathbb{Y}_{\mathbb{R}^+}(\vartheta(x, \chi)) \left(1 - \frac{c_k \vartheta(x, \chi)}{x_k}\right) \varphi(\chi) \, d\chi \\ &= \mathbb{E}[\mathbb{Y}_{\mathbb{R}^+}(\vartheta(x, \chi)) \left(\frac{c_k \vartheta(x, \chi)}{x_k} - 1\right)] \end{aligned}$$

Nous allons maintenant calculer le gradient de Γ par rapport à x :

$$\begin{aligned} \nabla_x \Gamma(x, \chi) &= \mathbb{E} \left[\mathbb{1}_{\mathbb{R}^+}(\vartheta(x, \chi)) \chi \left(\frac{c_k \vartheta(x, \chi)}{x_k} - 1 \right) \right. \\ &\quad \left. + \mathbb{Y}_{\mathbb{R}^+}(\vartheta(x, \chi)) c_k \left(\frac{\chi}{x_k} - \frac{\vartheta(x, \chi) \nu_k}{x_k^2} \right) \right] \end{aligned}$$

Nous pouvons choisir $\mathbb{Y}_{\mathbb{R}^+}(x) = \mathbb{1}_{\mathbb{R}^+}(x) \cdot x = [x]^+$ ce qui nous mène à

$$\begin{aligned} \nabla_x \Gamma(x, \chi) &= \mathbb{E} \left[2 \cdot [\vartheta(x, \chi)]^+ \chi \frac{c_k}{x_k} - \mathbb{1}_{\mathbb{R}^+}(\vartheta(x, \chi)) \chi \right. \\ &\quad \left. - [\vartheta(x, \chi)]^+ \vartheta(x, \chi) \frac{c_k}{x_k^2} \nu_k \right] \end{aligned}$$

□

Il s'en suit pour notre fonction objectif $J(x, \chi) = \mathbb{E}[\sum_{i=1}^n r_i \chi_i x_i] - d \cdot \mathbb{E}[[g(x, \chi) - c]^+]$:

$$\begin{aligned} \nabla_x J(x, \chi) &= \mathbb{E} \left[\begin{pmatrix} r_0 x_0 \\ \vdots \\ r_n x_n \end{pmatrix} \right] - d \cdot \mathbb{E} \left[2 \cdot [\vartheta(x, \chi)]^+ \chi \frac{c_k}{x_k} \right. \\ &\quad \left. - \mathbb{1}_{\mathbb{R}^+}(\vartheta(x, \chi)) \chi - [\vartheta(x, \chi)]^+ \vartheta(x, \chi) \frac{c_k}{x_k^2} \nu_k \right] \end{aligned}$$

2) *Approximation par convolution:* L'idée de base de cette méthode est d'approximer la fonction indicatrice $\mathbb{1}_{\mathbb{R}^+}$ par son produit de convolution avec une fonction $h_t(x) := \frac{1}{t} h\left(\frac{x}{t}\right)$ qui approxime la fonction de Dirac quand le paramètre t tend vers zéro (plus de détails sur cette méthode se trouvent dans les articles [5] et [6]). Le produit de convolution de deux fonctions est défini comme :

$$(\rho * h)(x) := \int_{-\infty}^{\infty} \rho(y) h(x - y) \, dy$$

Soit h une fonction paire, continue et non-négative telle que $\int_{-\infty}^{\infty} h(x) \, dx = 1$ et qui a son maximum au point $x = 0$. Dans ce cas, la fonction suivante peut être vue comme l'approximation d'une fonction réelle et localement intégrable ρ :

$$\rho_t(x) := (\rho * h_t)(x) = \frac{1}{t} \int_{-\infty}^{\infty} \rho(y) h\left(\frac{y-x}{t}\right) \, dy$$

Quand $\rho = \mathbb{1}_{\mathbb{R}^+}$, nous avons :

$$\rho_t(x) = \frac{1}{t} \int_0^{\infty} h\left(\frac{y-x}{t}\right) \, dy$$

et, par conséquent,

$$(\rho_t)'(x) = \frac{1}{t^2} \int_0^{\infty} h'\left(\frac{y-x}{t}\right) \, dy = -\frac{1}{t} h\left(\frac{-x}{t}\right) = -\frac{1}{t} h\left(\frac{x}{t}\right)$$

Ceci nous permet d'approximer le gradient de la fonction j par

$$\begin{aligned} \nabla(j_t)_x(x, \chi) &= r \chi \\ &\quad - d \cdot \left(-\frac{1}{t} \cdot h\left(\frac{g(x, \chi)}{t}\right) \cdot \chi \cdot g(x, \chi) \right. \\ &\quad \left. + \mathbb{1}_{\mathbb{R}^+}(g(x, \chi)) \cdot \chi \right) \end{aligned}$$

Il y a plusieurs choix possibles pour la fonction h . Dans [6] les auteurs en présentent quelques unes. Pour chacune de ces fonctions, ils calculent une valeur de référence de l'erreur quadratique moyenne du gradient approximé obtenu. Ils concluent que parmi les fonctions présentées, $h := \frac{3}{4}(1-x^2)\mathbb{1}_1(x)$ (où $\mathbb{1}_1$ est la fonction indicatrice de l'intervalle $]-1, 1[$) est le

meilleur choix concernant cette valeur. Suivant ce choix, nous obtenons au final le gradient approximé de la fonction objectif suivant :

$$\nabla(j_t)_x(x, \chi) = \begin{pmatrix} r_0 x_0 \\ \vdots \\ r_n x_n \end{pmatrix} + d \cdot \left(\frac{3}{4t} \left(1 - \left(\frac{g(x, \chi)}{t} \right)^2 \right) \mathbb{1}_1 \left(\frac{g(x, \chi)}{t} \right) \cdot \chi \cdot g(x, \chi) - \mathbb{1}_{\mathbb{R}^+}(g(x, \chi)) \cdot \chi \right)$$

B. Le problème du sac-à-dos avec contrainte stochastique

Comme présenté dans la section II, nous considérons deux problèmes du sac-à-dos avec contrainte, l'un ayant une contrainte en probabilité et l'autre une probabilité en espérance mathématique. Comme

$$\mathbb{P}\{g(x, \chi) \leq c\} = \mathbb{E}\{\mathbb{1}_{\mathbb{R}^+}(c - g(x, \chi))\}$$

ces deux formulations sont équivalentes (cf. [7]).

1) Le problème du sac-à-dos avec contrainte en probabilité:

En général, la contrainte en probabilité (3) ne définit pas d'ensemble convexe ce qui rend la solution des problèmes avec contrainte en probabilité souvent difficile.

Prékopa ([7]) a démontré que l'ensemble défini par la contrainte (3) est convexe si χ suit une distribution de densité log-concave et si g est quasi-convexe. La première condition peut facilement être prouvée pour une distribution normale et comme notre fonction g est linéaire, elle est aussi convexe.

Nous résoudrons le *CCKP* continu en le reformulant comme un problème déterministe équivalent, plus précisément sous la forme d'un problème conique de second ordre, appelé ci-après *SOCP* ([8]). Généralement, un problème du type *SOCP* a la forme suivante :

$$\max_{x \in X_{ad}} v^T x \quad (6)$$

$$\text{s.c.} \quad \|Ax + b\| \leq c^T x + d \quad (7)$$

avec $A \in \mathbb{R}^n \times \mathbb{R}^n$, $x, v, b, c \in \mathbb{R}^n$ et $d \in \mathbb{R}$. Dans ce qui suit, nous appelons une contrainte du type (7) une *contrainte SOCP*.

Soit Σ la matrice de covariance du vecteur de probabilité χ . Comme nous considérons $p > 0.5$, nous obtenons l'équivalence suivante (voir e.g. [8]) :

$$\mathbb{P}\left\{\sum_i \chi_i x_i \leq c\right\} \geq p \iff$$

$$\sum_i \chi_i x_i + F^{-1}(p) \|\Sigma^{1/2} x\| \leq c$$

Remarquons que Σ est une matrice diagonale dû au fait que les poids sont distribués de façon normale. Il s'en suit que sa racine $\Sigma^{\frac{1}{2}}$ est aussi diagonale ayant les écart-types $\sigma_i > 0$ dans sa diagonale.

Le problème (2) devient

$$\begin{aligned} \max_{x \in [0, 1]^n} & \quad \mathbb{E}\left[\sum_i r_i \chi_i x_i\right] \\ \text{s.c.} & \quad \|\Sigma^{1/2} x\| \leq -\frac{1}{\delta} \sum_i \mu_i x_i + \frac{c}{\delta} \end{aligned}$$

avec $\delta := F^{-1}(p) > 0$.

La contrainte $0 \leq x_i \leq 1$ ($i = 1, \dots, n$) du problème continu correspondant peut être reformulée comme une contrainte *SOCP* :

$$0 \leq x_i \leq 1 \iff \|A_i x\| \leq x_i \wedge \|A_i x\| \leq 1$$

avec $A_i \in \mathbb{R}^{1 \times n}$, $A_i[1, k] = 0 \forall k \neq i$ et $A_i[1, i] = 1$.

Nous obtenons le problème de type *SOCP* suivant :

$$\max_{x \in \mathbb{R}^n} v^T x \quad (8)$$

$$\text{s.c.} \quad \|\Sigma^{1/2} x\| \leq -\frac{1}{\delta} \cdot \mu \cdot x + \frac{c}{\delta} \quad (9)$$

$$\|A_i x\| \leq \nu^i x \quad i = 1, \dots, n \quad (10)$$

$$\|A_i x\| \leq 1 \quad i = 1, \dots, n \quad (11)$$

avec $v := (r_1 \mu_1, \dots, r_n \mu_n)$ et $\nu^i \in \mathbb{R}^n$ tel que $\nu^i_k = 1$ si $k = i$ et $\nu^i_k = 0$ sinon.

Notre but est de résoudre ce problème avec le programme informatique libre de Boyd et al. [9]. Malheureusement, ce programme ne sait résoudre que des problèmes avec une solution strictement admissible comme la méthode utilisée est une méthode de point intérieur. Ceci n'est pas le cas pour notre problème car la contrainte (10) est toujours saturée. Pour résoudre ce problème, nous modifions légèrement la contrainte (10) : en ajoutant un paramètre $\epsilon > 0$ au côté droit de (10), $x \equiv 0$ (par exemple) devient un "point intérieur" du problème, i.e. une solution strictement admissible. Une telle modification signifie que nous permettons les composantes de x à prendre des valeurs négatives, plus précisément entre $-\epsilon/2$ et 0. Cependant, comme nous résolvons un problème continu, cette modification ne change guère la solution si seulement ϵ est choisi suffisamment petit. De plus, la solution optimale du problème modifié est supérieure où égale à la solution exacte, i.e. apporte aussi une borne supérieure pour le problème combinatoire.

2) Le problème du sac-à-dos avec contrainte en espérance mathématique:

Comme l'ensemble défini par la contrainte (5) est le même que l'ensemble défini par la contrainte en probabilité (3), il est aussi convexe. Ceci nous permet de résoudre l'*ECKP* (4) avec un algorithme du type Arrow-Hurwicz stochastique (voir Algorithme III.2). Un algorithme Arrow-Hurwicz stochastique est un algorithme du type gradient stochastique pour des problèmes contraints d'optimisation stochastique. Il utilise des multiplicateurs Lagrangien pour intégrer les contraintes. En utilisant une des deux méthodes présentées dans la section III-A, nous pouvons déterminer (resp. approximer) le gradient

Algorithme Arrow-Hurwicz stochastique

- 1) Choisir $x^0 \in X^{ad}$ et $\lambda^0 \in [0, \infty)$ ainsi que deux α -suites $(\epsilon^k)_{k \in \mathbb{N}}$ et $(\rho^k)_{k \in \mathbb{N}}$
- 2) A l'itération $k+1$, tirer χ_{k+1} suivant sa loi normale, calculer $r^k = \nabla j(x^k, \chi_{k+1})$, $\theta^k = \nabla \Theta(x^k, \chi_{k+1})$ et mettre à jour x^{k+1} et λ^{k+1} comme suite :

$$x^{k+1} = x^k + \epsilon^k (r^k + (\theta^k)^T \lambda^k)$$

$$\lambda^{k+1} = \lambda^k - \rho^k (\Theta(x^{k+1}, \chi_{k+1}) - p)$$

- 3) Pour tous les $i = 1, \dots, n$: Si $x_i^{k+1} > 1$ poser $x_i^{k+1} = 1$ et si $x_i^{k+1} < 0$ poser $x_i^{k+1} = 0$
- 4) Pour tous les $i = 1, \dots, n$: Si $\lambda_i^{k+1} < 0$ poser $\lambda_i^{k+1} = 0$

Algorithme III.2: Fonction à maximiser : $\mathbb{E}[j(x, \chi)]$; Contrainte : $\mathbb{E}[\Theta(x, \chi)] \geq p$

de la fonction de contrainte $\mathbb{E}[\mathbb{1}_{\mathbb{R}^+}(c - \sum_i \chi_i x_i)]$. Dans le cas d'Intégration par parties, nous pouvons directement appliquer la proposition de Andrieu, Cohen et Vázquez-Abad (voir [4], Théorème 5.5) :

Proposition 2: Soit Γ définie comme $\Gamma(x, \chi) := \mathbb{E}[\mathbb{1}_{\mathbb{R}^+}(\gamma(x, \chi))]$, où $\chi \in \mathbb{R}^n$ est un vecteur aléatoire normalement distribué avec densité φ et $\gamma : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ est une fonction C^1 qui est polynomiale en au moins une composante χ_k ($k \in \{1, \dots, n\}$) du vecteur aléatoire. On suppose que $\varphi(\chi) \neq 0$ pour tous les χ . Notons $\mathbb{Y}_{\mathbb{R}^+}(\cdot)$ une primitive de $\mathbb{1}_{\mathbb{R}^+}(\cdot)$. Alors, par intégration par parties, on a

$$\Gamma(x, \chi) = \mathbb{E}[\mathbb{Y}_{\mathbb{R}^+}(\gamma(x, \chi)) M_k(x, \chi)]$$

avec

$$M_k(x, \chi) = \frac{1}{\gamma'_{\chi_k}(x, \chi)} \frac{\partial \ln(\gamma'_{\chi_k}(x, \chi) / \varphi(\chi))}{\partial \chi_k}$$

On en déduit alors que

$$\nabla_x \Gamma(x, \chi) = \mathbb{E} \left[\mathbb{1}_{\mathbb{R}^+}(\gamma(x, \chi)) \nabla_x \gamma(x, \chi) M_k(x, \chi) + \mathbb{Y}_{\mathbb{R}^+}(\gamma(x, \chi)) \nabla_x M_k(x, \chi) \right]$$

Remarque 1: Dans cette version du théorème nous supposons une distribution normale et g doit être polynomiale en au moins une composante du vecteur aléatoire. En général, on peut faire des hypothèses moins fortes si seulement

$$\left[\frac{\mathbb{Y}_{\mathbb{R}^+}(g(x, \chi)) \varphi(\chi)}{\frac{\partial g}{\partial \chi_k}} \right]_{-\infty}^{\infty} = 0$$

Dans notre cas, nous obtenons

$$M_k(x, \chi) = -\frac{\varphi'_{\chi_k}(x, \chi)}{(-g'_{\chi_k}(x, \chi)) \cdot \varphi(\chi)} = -\frac{(\chi_k - \mu_k)}{\sigma_k^2} \frac{1}{x_k}$$

$$\nabla_x M_k(x, \chi) = \frac{\varphi'_{\chi_k}(x, \chi) \nabla_x (-g'_{\chi_k}(x, \chi))}{\varphi(\chi) (-g'_{\chi_k}(x, \chi))^2} = \frac{(\chi_k - \mu_k)}{\sigma_k^2} \frac{1}{x_k^2} \cdot \nu^k$$

avec $\nu^k \in \mathbb{R}^n$ défini tel que $\nu^k_k = 1$ et $\nu^k_i = 0$ si $i \neq k$. Il s'en suit

$$\nabla_x \mathbb{E}[\mathbb{1}_{\mathbb{R}^+}(c - g(x, \chi))] = \mathbb{E} \left[\mathbb{1}_{\mathbb{R}^+}(g(x, \chi)) \frac{(\chi_k - \mu_k)}{\sigma_k^2 x_k} \left(\chi + \frac{g(x, \chi)}{x_k} \cdot \nu^k \right) \right]$$

IV. RÉSULTATS NUMÉRIQUES

Cette section sera divisée en deux sous-sections : Dans la première sous-section, nous discutons les résultats des problèmes relaxés. Les résultats de la deuxième sous-section montrent le comportement d'un algorithme du type branch-and-bound qui utilise les méthodes présentées dans cet article pour estimer des bornes supérieures.

Tous les résultats présentés sont obtenus sur une même instance exemplaire de dimension $n = 15$. Cette instance a été utilisée par Cohn et Barnhart ([2]) pour leur tests numériques (dans le cas du *ECKP* nous avons choisi une probabilité de $p = 0.6$ qui est "équivalent" à leur facteur de pénalité). Dans la sous-section IV-B nous allons comparer les résultats.

A. Problèmes continus

Figure 1 montre la convergence de l'algorithme gradient stochastique quand on utilise l'Approximation par convolution (graphe noir) où Intégration par parties (graphe gris). On observe que l'algorithme est beaucoup moins robuste dans le deuxième cas, particulièrement au début des itérations (voir discussion dans la section III-A). Sur 500 itérations, l'algorithme contenant la méthode Approximation par convolution trouve une solution de l'instance exemplaire de 4676.434 (moyenne sur 1000 tests), en utilisant l'Intégration par parties la solution trouvée est de 4658.748. La différence n'est pas très grande (0.3%). Par contre, quand on utilise cet algorithme pour fournir des bornes supérieures dans un algorithme branch-and-bound, on ne peut pas garantir que la solution du problème relaxé obtenue est une borne supérieure du problème combinatoire correspondant. Par conséquent, il est possible qu'un sous-arbre qui contient une solution très bonne (voire même optimale) soit coupé (voir section IV-B). Figure 2 montre la convergence de l'algorithme Arrow-Hurwicz quand on utilise l'Intégration par parties. Comparé avec l'Approximation par convolution, la différence n'est pas aussi grande que dans le cas du *SRKP* (4696.097 contre 4693.871). Ceci peut s'expliquer par le fait que dans le cas de l'algorithme Arrow-Hurwicz il y a des fluctuations possibles dans les deux sens.

Dans le cas du *ECKP*, les meilleurs résultats sont obtenus par l'algorithme *SOCP* de Boyd et al. qui est un algorithme primal-dual. Par contre, il n'est utilisable que pour des instances jusqu'à $n = 160$ environ, comme pour des plus grandes dimensions le besoin de mémoire devient trop important (voir [3]).

B. Problèmes combinatoires

Nous avons utilisé un algorithme du type branch-and-bound comme celui utilisé dans [2] (pour plus de détails voir [3]).

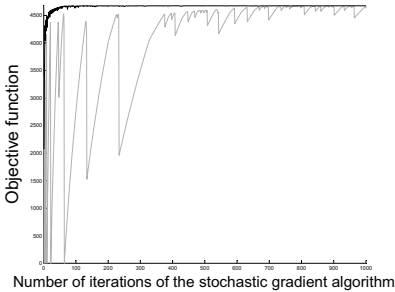


Fig. 1. Résultats de l’algorithme gradient stochastique appliqué à une instance *SRKP* : Approximation par convolution (noir) versus Intégration par parties (gris)

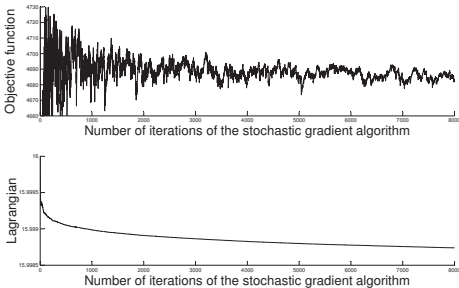


Fig. 2. Résultats de l’algorithme Arrow-Hurwicz (incl. Intégration par parties) appliqué à une instance *ECKP*

Dans le cas du *SRKP*, les deux méthodes pour calculer des bornes supérieures donnent les mêmes résultats : l’algorithme doit calculer 100 bornes supérieures avant de trouver l’optimum. Quand on utilise les bornes proposées par Cohn et Barnhart ([2]), l’algorithme calcule en tout 144 bornes. Par contre, comme le calcul d’une seule borne de Cohn et Barnhart prend beaucoup moins de temps, notre méthode n’est pas compétitive pour des petites dimensions. Cependant, à partir d’une dimension de $n = 50$, notre méthode trouve l’optimum plus vite à cause d’un nombre de bornes supérieures calculées beaucoup moins important (voir [3]).

Dans le cas du *ECKP*, l’avantage de l’utilisation d’un algorithme Arrow-Hurwicz pour calculer les bornes est le besoin de mémoire qui est moins important comparé avec l’algorithme *SOCP* de Boyd et al.. Par contre, comme les solutions des problèmes relaxés obtenues par l’algorithme Arrow-Hurwicz sont souvent un peu moins bonnes que celles

obtenues avec l’algorithme *SOCP*, il y a des cas (surtout pour des instances à partir d’une dimension de $n = 50$) où l’algorithme branch-and-bound ne trouve pas la solution optimale car un sous-arbre important a été coupé à tort.

V. CONCLUSION

Dans cet article, nous résolvons deux variantes différentes du problème du sac-à-dos stochastique relaxé. Pour ceci, trois méthodes de résolution différentes sont proposées. Comme deux de ces méthodes sont de type gradient stochastique, deux estimations du gradient d’une fonction indicatrice sont calculées.

Les problèmes combinatoires correspondants peuvent, par exemple, être résolus par une méthode de branch-and-bound qui utilise les solutions des relaxations comme bornes supérieures (cf. [3]).

Les résultats numériques ont montré que les deux algorithmes du type gradient stochastique convergent plus robustement quand on utilise des gradients obtenus par Approximation par convolution (et non par Intégration par parties).

En ce qui concerne la résolution des problèmes combinatoires, nos méthodes pour fournir des bornes supérieures dans un algorithme branch-and-bound sont le plus adaptées aux problèmes de grandes tailles : d’un côté, le calcul d’une seule borne exige beaucoup de temps mais, d’un autre côté, les bornes sont très bonnes ce qui résulte en un nombre de bornes calculées beaucoup moins important.

REFERENCES

- [1] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer-Verlag (Berlin, Heidelberg), 2004.
- [2] A. Cohn and C. Barnhart, “The stochastic knapsack problem with random weights: A heuristic approach to robust transportation planning,” in *Proceedings of the Triennial Symposium on Transportation Analysis (TRISTAN III)*, 1998.
- [3] S. Kosuch and A. Lisser, “Stochastic knapsack problems,” Université Paris XI, Laboratoire de Recherche en Informatique, Orsay, France, Tech. Rep. 1505, novembre 2008.
- [4] L. Andrieu, “Optimization sous contrainte en probabilité,” Ph.D. dissertation, Ecole Nationale des Ponts et Chaussées, 2004.
- [5] Y. M. Ermoliev, V. I. Norkin, and R. J.-B. Wets, “The minimization of semicontinuous functions: Mollifier subgradients,” *SIAM Journal on Control and Optimization*, vol. 33(1), pp. 149–167, 1995.
- [6] L. Andrieu, G. Cohen, and F. Vázquez-Abad, “Stochastic programming with probability constraints,” 2007, <http://fr.arxiv.org/abs/0708.0281> (Accessed 24 October 2008).
- [7] A. Prékopa, *Stochastic Programming*. Kluwer Academic Publishers (Dordrecht, Boston), 1995.
- [8] S. Boyd, H. Lebrét, M. S. Lobo, and L. Vandenberghe, “Applications of second-order cone programming,” *Linear Algebra and its Applications*, vol. 284, pp. 193–228, 1998.
- [9] S. Boyd, M. S. Lobo, and L. Vandenberghe, “Software for second-order cone programming,” 1995, http://www.stanford.edu/~boyd/old_software/socp/doc.pdf (Accessed 24 October 2008).