Orsay
N° d'ordre : 9992

THÈSE

présentée pour obtenir le grade de
Docteur en Sciences de l'Université Paris XI, Orsay
Spécialité : Informatique

—

Stochastic Optimization Problems with Knapsack Constraint

Stefanie KOSUCH

—

Soutenue le 21 octobre 2010 devant le jury composé de :

Directeur de thèse :	Abdel LISSER	Professeur
Rapporteurs :	Jean-Baptiste HIRIART-URRUTY	Professeur
	Maarten H. VAN DER VLERK	Professeur
Examinateurs :	Pierre CARPENTIER	Professeur
	Brigitte ROZOY	Professeur
Invitée au jury:	Laetitia ANDRIEU	Ingénieur Chercheur EDF R&D (Clamart)

Meinen Eltern.

Acknowledgements - Remerciements - Danksagungen

I want to thank Professor Abdel Lisser for the opportunity to write this Ph.D. thesis under his supervision, for the motivation he gave me and for all the many things I learned as well as experiences I gained during the last 3 years.

Special thanks to Professor Hiriart-Urruty and Professor van der Vlerk for having accepted to review my thesis and for attending my Ph.D. defense via video-conference. It was a pleasure to meet you virtually! Thanks also to Ms. Andrieu, Professor Carpentier and Professor Rozoy for being part of my defense jury. It was a pleasure to meet you in person!

Un grand merci aux membres du LRI en général et à l'équipe GraphComb en particulier pour votre accueil chaleureux, votre serviabilité, les pauses conviviales et la bonne humeur qui règne au bâtiment 490. Vous allez me manquer !

Merci Vincent, pour ton support, ton sens de l'écoute, ta patience, ton aide avec la "vraie" informatique et les corrections de mes textes en français. Bref: Merci pour être là !

Ein ganz besonderer Dank gilt meinen Eltern und meiner Familie, für die entgegengebrachte Liebe, die Freiheit die sie mir immer ließen und die vielseitige Unterstützung bei all meinen bisherigen Entscheidung. Ohne meine Freude an Mathematik und Naturwissenschaften (danke Papa!) sowie daran, anderen etwas beibringen zu dürfen (danke Mama!) wäre ich mit Sicherheit nicht dort, wo ich heute bin. Und das wäre doch schade, oder?

Résumé en Français

Introduction

Le problème de sac-à-dos consiste à choisir parmi un ensemble d'objets un sous-ensemble dont le poids total respecte une contrainte (la capacité pondérale du sac-à-dos) et telle que la valeur totale des objets choisis est maximisée.
Dans le cas du problème de sac-à-dos déterministe, tous les paramètres (poids, valeurs, capacité) sont connus. Dans cette thèse, nous étudions des variantes stochastiques du problème. Plus précisément, nous étudions quatre problèmes d'optimisation stochastiques ayant en commun une contrainte de sac-à-dos dans laquelle les poids sont supposés être aléatoires. Cet-à-dire, quand la (pré)décision de quels objets choisir doit être prise, les poids exacts de ces objets sont encore inconnus. Il s'en suit, en général, qu'on ne peut pas être sûr si les objets choisis vont respecter la capacité du sac-à-dos.
Le premier problème traité est un problème avec *recours simple*. Ceci veut dire qu'on suppose qu'en cas de violation de la contrainte de sac-à-dos (i.e. de capacité) une pénalité par unité de surcharge doit être payée. Nous supposons en plus que les valeurs des objets dépendent linéairement des poids, et sont donc aléatoires eux-mêmes. Le but est de maximiser la différence entre l'espérance de la valeur totale des objets choisis et l'espérance de la pénalité à payer. Ce problème a été étudié pour la première fois par Cohn et Barnhart dans [CB98]. Comme Cohn et Barnhart, nous faisons l'hypothèse que les poids aléatoires suivent une distribution normale.
Le deuxième problème étudié est un problème avec *contrainte probabiliste*. Dans ce modèle on suppose qu'une violation de la contrainte de capacité est admissible et aucune pénalité ne doit être payée en cas de surcharge. Par contre, la probabilité d'une violation est restreinte en introduisant une contrainte probabiliste. Comme pour le problème avec recours simple nous supposons que les poids sont distribués normalement.
Dans le cas du problème de sac-à-dos *avec recours*, qui est le troisième problème étudié, une correction de la décision prise peut être effectuée une fois que les poids exacts sont connus. Cette action de recours entraîne naturellement des coûts. Plus précisément, si nous retirons des objets, une pénalité doit être payée. Et si on en ajoute, la valeur d'un objet ajouté est strictement plus petite que si on l'avait ajouté au moment où les poids étaient encore aléatoires.
Nous étudions dans cette thèse deux cas différents du problème de sac-à-dos avec recours. D'un côté nous supposons que les poids sont normalement distribués. Ceci entraîne le problème qu'il n'est pas évident comment évaluer la fonction objectif. D'un autre côte nous étudions le cas de poids distribués discrètement. Dans ce cas, le problème a une reformulation équivalente déterministe qui, en général, est de très grande taille et difficile

à résoudre exactement. Nous étudions donc l'approximabilité du problème. Le cas de poids distribués discrètement a été étudié auparavant dans la thèse de Lopez ([Lop10], voir aussi [GLLH10]) où une relaxation semi-définite est proposée.
Le dernier problème étudié est un problème *biniveau avec contrainte de sac-à-dos probabiliste* au premier niveau. Nous faisons encore une fois l'hypothèse de poids distribués discrètement, ce qui permet une reformulation du problème initial en un problème biniveau déterministe. De tels problèmes ont été étudiés auparavant et il est bien connu comment les reformuler en problèmes équivalents avec un seul niveau (voir e.g. [AHJS97]).

Problème de sac-à-dos avec recours simple

Nous considérons un problème de sac-à-dos stochastique sous la forme suivante : Etant donnés n objets dont les poids sont inconnus 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. La valeur par unité de poids $r_i > 0$ de l'objet i et la capacité c du sac-à-dos sont supposées être déterministes. En cas de surcharge, une pénalité par unité de surcharge d doit être payée. L'objectif est de maximiser l'espérance de la valeur totale des objets choisis moins l'espérance de la pénalité à payer. Le problème peut être formulé mathématiquement comme suit :

Problème de sac-à-dos avec recours simple ($SRKP$)

$$\max_{x\in\{0,1\}^n} \mathbb{E}[\sum_{i=1}^{n} r_i\chi_i x_i] - d\cdot\mathbb{E}[[\sum_{i=1}^{n}\chi_i x_i - c]^+]$$

Ici $\mathbb{E}[\cdot]$ note l'espérance mathématique et $[x]^+ := \max(0,x) = x\cdot\mathbb{1}_{\mathbb{R}^+}(x)$ $(x\in\mathbb{R})$ avec $\mathbb{1}_{\mathbb{R}^+}$ étant la fonction indicatrice de l'intervalle réel non-négatif.

Méthode de résolution

Comme framework nous proposons d'utiliser un algorithme branch-and-bound (B&B) comme celui utilisé dans [CB98]. Pour obtenir des bornes supérieures nous résolvons des relaxations continues.
Il n'est pas difficile de démontrer que la relaxation continue du $SRKP$ est concave. Ceci nous permet de le résoudre en utilisant un algorithme gradient stochastique. Un algorithme 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 tirage au sort.

Dans le cas du $SRKP$, nous avons $j(x,\chi) = \sum_i r_i\chi_i x_i - d\cdot[\sum_{i=1}^n \chi_i x_i - c]^+$. j n'étant pas différentiable en tout point, nous approximons son gradient à l'aide d'une approximation par convolution. L'idée de base de cette méthode est de remplacer 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 [ENW95] et [ACVA07]). Le produit de convolution de deux fonctions est défini par :

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

Soit h une fonction paire, continue et non-négative telle que $\int_{-\infty}^{\infty} h(x)\,\mathrm{d}x = 1$ 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 :

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

Quand $\rho = \mathbb{1}_{\mathbb{R}^+}$, le gradient de ρ_t se laisse calculer de la manière suivante (exemple monodimensionel) :

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

Résultats Numériques

Nous comparons notre approche avec la méthode de résolution de Cohn et Barnhart ([CB98]). Pendant que l'algorithme B&B qu'on utilise est principalement celui de Cohn et Barnhart, la nouveauté dans notre approche est de résoudre des relaxations pour obtenir des bornes supérieures et couper des sous-arbres. Les bornes de Cohn et Barnhart, au contraire, sont des bornes qui se laissent calculer très facilement et dont le calcul prend beaucoup moins de temps que résoudre une relaxation. Cependant, nos tests numériques ont montrés que nos bornes sont beaucoup plus serrées. Il en résulte que l'algorithme B&B prend moins de temps sur des instances de grande taille car un plus grand nombre de sous-arbres peut être rejeté.

Problème de sac-à-dos avec contrainte probabiliste

En utilisant les définitions données pour le problème avec recours simple, le problème avec contrainte probabiliste peut être formulé comme suite :

Problème de sac-à-dos avec contrainte probabiliste ($CCKP$)

$$\max_{x\in\{0,1\}^n} \mathbb{E}\left[\sum_{i=1}^{n} r_i\chi_i x_i\right]$$
$$\text{s.t. } \mathbb{P}\{\sum_{i=1}^{n} \chi_i x_i \leq c)\} \geq p.$$

où $\mathbb{P}\{A\}$ est la probabilité d'un événement A et $p \in (0.5, 1]$. Le problème peut être reformulé en un problème avec contrainte en espérance ($ECKP$) dû à l'égalité suivante :

$$\mathbb{P}\{\sum_{i=1}^{n} \chi_i x_i \leq c\} = \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}[c - \sum_{i=1}^{n} \chi_i x_i]\right]$$

Méthode de résolution

Comme dans le cas avec recours simple nous proposons de résoudre le $CCKP$ à l'aide d'un algorithme B&B . Des bornes supérieures sont encore une fois obtenues en appliquant un algorithme de type gradient stochastique pour la résolution des relaxations continues. Plus précisément, nous appliquons un algorithme Arrow-Hurwicz stochastique. L'algorithme introduit un multiplicateur de Lagrange et résout le problème dual lagrangien de l'$ECKP$ avec un algorithme sous-gradient stochastique.
Le problème qui se pose concerne la fonction indicatrice (à l'intérieur de l'espérance de la contrainte) dont l'unique sous-gradient est zéro presque partout. Nous résolvons ce problème de deux manières différentes :
La première méthode consiste à approximer le gradient de la fonction indicatrice par des différences finies ce qui nous donne comme $h^{\text{ième}}$ composante du gradient approché :

$$\frac{\mathbb{1}_{\mathbb{R}^+}(c - g(x + \delta\nu^h, \chi)) - \mathbb{1}_{\mathbb{R}^+}(c - g(x - \delta\nu^h, \chi))}{2\delta}$$

où $g(x, \chi) := \sum_{i=1}^{n} \chi_i x_i$, $\delta > 0$ et $\nu^h \in \{0, 1\}^n$ tel que $\nu_h^h = 1$ et $\nu_i^h = 0$ quand $i \neq h$.
L'idée de la deuxième méthode est de remplacer la fonction $\theta(x, \chi) = \mathbb{1}_{\mathbb{R}^+}[c - \sum_{i=1}^{n} \chi_i x_i]$ dans l'espérance de la contrainte par une fonction $\widetilde{\theta}$ tel que $\mathbb{E}[\widetilde{\theta}(x, \chi)] = \mathbb{E}[\theta(x, \chi)]$. $\widetilde{\theta}$ est sous-différentiable en tout point et l'idée est d'utiliser un sous-gradient de $\widetilde{\theta}$ dans l'algorithme sous-gradient stochastique au lieu de celui de θ. La reformulation de $\mathbb{E}[\theta(x, \chi)]$ est obtenue à l'aide de l'intégration par parties.

Etude de la convergence de l'algorithme Arrow-Hurwicz stochastique

Culioli et Cohen ont démontré un théorème qui propose un ensemble de conditions garantissant la faible convergence de l'algorithme Arrow-Hurwicz stochastique. Dans notre cas,

certaines de ces conditions ne sont pas satisfaites, notamment la stricte concavité de la fonction objectif et la concavité de la fonction de contrainte. Alors que nous pensons que la première hypothèse peut être négligée dans le cas d'une fonction de contrainte strictement concave, la deuxième hypothèse est clairement nécessaire pour assurer la convergence de l'algorithme. Cependant, avec un bon choix de la grandeur des pas, une convergence peut être obtenue pratiquement, surtout pour des grandes valeurs de p.
Toutefois, quand on utilisait la méthode basée sur l'intégration par parties pour obtenir un sous-gradient, l'algorithme ne convergeait pas sur les premières instances testées. Après une étude plus approfondie du problème et du fonctionnement de l'algorithme, nous avons réussi à faire converger l'algorithme : d'un côte avec une simple reformulation équivalente de la contrainte probabiliste, et de l'autre côté avec un choix plus prudent de la variable avec laquelle l'intégration par parties est exécutée.

Problème de sac-à-dos avec recours

Dans le cas du sac-à-dos avec recours on considère que, pendant que les poids sont encore inconnus, quelques objets peuvent être choisis à une valeur (par unité de poids) de r_i pour objet i ($i = 1, \ldots, n$) (première étape). Une fois les poids connus, la décision peut être corrigée (deuxième étape) : Des objets supplémentaires peuvent être ajoutés à une valeur (par unité de poids) de $\widetilde{r}_i < r_i$ et/ou on peut retirer des objets si la capacité n'est pas respectée, ce qui entraîne une pénalité (par unité de poids) de $d_i > r_i$. La correction doit être effectuée de façon que, après la deuxième étape, le poids total des objets (restants) dans le sac-à-dos respecte la capacité.
Dans cette thèse nous étudions deux variantes du problème de sac-à-dos avec recours : dans la première, les poids sont supposés être distribués normalement, et dans la deuxième nous faisons l'hypothèse qu'il n'existe qu'un nombre fini de scénarios pour la réalisation des poids.

Problème de sac-à-dos avec recours et poids normalement distribué

Dans le premier modèle avec recours étudié nous supposons que les poids sont normalement distribués, que les valeurs et pénalités dépendent linéairement des poids, qu'il faut respecter une contrainte probabiliste à la première étape et qu'un échange d'objets n'est pas permis à la deuxième étape (i.e. on peut retirer des objets que dans le cas d'une surcharge et ajouter des objets que si la différence entre le poids total des objets choisis à la première étape et la capacité le permet). Les vecteurs de décisions correspondants de la deuxième étape sont dénommés y^+ (pour les objets ajoutés) et y^- (pour les objets

retirés). Le problème étudié peut être décrit comme suit :

$$
\begin{aligned}
(TSKP) \quad \max_{x\in\{0,1\}^n} \quad & \mathbb{E}\left[\sum_{i=1}^{n} r_i\chi_i x_i\right] + \mathbb{E}\left[\mathcal{Q}(x,\chi)\right] \\
\text{s.t.} \quad & \mathbb{P}\{\sum_{i=1}^{n}\chi_i x_i \le c\} \ge p. \\
& \mathcal{Q}(x,\chi) = \max_{y^+,y^-\in\{0,1\}^n} \sum_{i=1}^{n} \overline{r}_i\chi_i y_i^+ - d\sum_{i=1}^{n}\chi_i y_i^- \\
& \text{s.t. } y_k^+ \le 1 - x_k \quad \forall\, k = 1,\dots,n, \\
& \quad y_k^- \le x_k \quad \forall\, k = 1,\dots,n, \\
& \quad y_k^+ \le \mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n}\chi_i x_i)\cdot y_k^+ \quad \forall\, k = 1,\dots,n, \\
& \quad y_k^- \le \mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n}\chi_i x_i - c)\cdot y_k^- \quad \forall\, k = 1,\dots,n, \\
& \quad \sum_{i=1}^{n}(x_i + y_i^+ - y_i^-)\chi_i \le c.
\end{aligned}
$$

Bornes supérieures

Nous proposons de calculer des bornes supérieures sur la valeur optimale du $TSKP$ en résolvant une relaxation continue du problème : D'abord nous unifions les valeurs par unité de poids de la deuxième étape en supposant que pour tous les objets nous recevons une récompense de $\overline{r}_{max} = \max\{\overline{r}_i | i \in \{1,\dots,n\}\}$ par unité de poids. Il s'en suit pour le cas continu qu'à la deuxième étape nous remplissons le sac-à-dos jusqu'à sa capacité dans le cas d'une sous-charge avec une valeur par unité de poids de $\overline{r}_{max}$, et nous retirons exactement la surcharge dans le cas contraire, en payant une pénalité par surcharge de d. Une borne supérieure peut donc être obtenue en résolvant le problème avec recours simple et contrainte probabiliste suivant en appliquant une combinaison des méthodes proposées auparavant :

$$
\begin{aligned}
\max_{x\in\{0,1\}^n\setminus\{\mathrm{O}_n\}} \quad & \mathbb{E}\left[\sum_{i=1}^{n} r_i x_i\chi_i\right] + \overline{r}_{max}\cdot\mathbb{E}\left[[c - \sum_{i=1}^{n} x_i\chi_i]^+\right] - d\cdot\mathbb{E}\left[[\sum_{i=1}^{n} x_i\chi_i - c]^+\right] \\
\text{s.t.} \quad & \mathbb{P}\{\sum_{i=1}^{n} x_i\chi_i \le c\} \ge p.
\end{aligned}
$$

Bornes inférieures

Une des plus grandes difficultés concernant le $TSKP$ est d'évaluer la fonction objectif étant donné une décision de la première étape. Pour remplacer cette évaluation exacte

nous proposons des bornes inférieures sur la valeur de la fonction objectif à un point x donné. Ces bornes sont basées sur une estimation de la valeur des objets ajouté à la deuxième étape, indépendante des vecteurs de décision y^+ et y^-. Plus précisément, nous estimons la capacité qui sera (encore) non-utilisée après que la décision de la deuxième étape a été prise. Nous nous intéressons en particulier au cas d'objets similaires, i.e. au cas où la probabilité pour qu'un objet ait le double du poids d'un deuxième objet est (quasi) nulle. La meilleure borne parmi ces bornes inférieures est ensuite déterminée avec un algorithme B&B qui parcourt l'ensemble des solutions admissibles à la première étape.

Résultats numériques

Nos tests numériques ont montré que, si on ne permet que l'ajout d'objets à la deuxième étape, les bornes inférieures proposées particulièrement pour le cas de poids similaires sont de la même qualité que les bornes inférieures proposées pour le cas général (quand appliquées à une instance d'objets similaires). Par contre, dès qu'on admet le rejet d'objets, les bornes particulaires se montrent beaucoup plus serrées que les bornes générales. Pour des instances de grande taille (500-2000 objets) les bornes proposées pour le cas d'objets similaires sont même presque optimales et peuvent, dans un algorithme B&B , être déterminées en moins de 30 minutes.

Problème de sac-à-dos avec recours et poids normalement distribué

Dans la deuxième variante du problème de sac-à-dos avec recours, nous supposons qu'il n'y a qu'un nombre fini de réalisations du vecteur du poids $\chi^1, \ldots, \chi^K$ avec probabilités non-nulles correspondantes $p^1, \ldots, p^K$. Dans un premier temps, nous étudions le cas de valeurs $r_i, \widetilde{r}_i$ et de pénalités d_i déterministe. Un échange d'objets à la deuxième étape est possible. En introduisant K copies des vecteurs de décision de la deuxième étape et en traitant les contraintes de la deuxième étape pour chacun des K scénarios séparément, on obtient la reformulation déterministe suivante :

$$
\begin{aligned}
(TSK^D) \quad \max \quad & \sum_{i=1}^{n} r_i x_i + \sum_{k=1}^{K} p^k \left(\sum_{i=1}^{n} \overline{r}_i (y^+)_i^k - \sum_{i=1}^{n} d_i (y^-)_i^k \right) \\
\text{s.t.} \quad & (y^+)_i^k \leq 1 - x_i, \quad \forall\, i = 1, \ldots, n, \forall\, k = 1, \ldots, K, \\
& (y^-)_i^k \leq x_i, \quad \forall\, i = 1, \ldots, n, \forall\, k = 1, \ldots, K, \\
& \sum_{i=1}^{n} (x_i + (y^+)_i^k - (y^-)_i^k) \chi_i^k \leq c, \quad \forall\, k = 1, \ldots, K, \\
& x \in \{0,1\}^n, \\
& (y^+)^k, (y^-)^k \in \{0,1\}^n \quad \forall\, k = 1, \ldots, K.
\end{aligned}
$$

Résultats de non-approximabilité

Nous montrons dans cette thèse que la variante du TSK^D où des objets peuvent seulement être ajoutés à la deuxième étape ($AddTSK^D$) est équivalent au problème de sac-à-dos avec plusieurs contraintes $MCKP$ (aussi appelé problème de sac-à-dos multidimensionnel). Plus particulièrement, chaque instance de l'$AddTSK^D$ peut être vue comme une instance du $MCKP$ et une instance du $MCKP$ peut être résolue à l'intermédiaire d'une instance de l'$AddTSK^D$. Comme ces réductions sont polynomiales, tous les résultats d'approximation et de non-approximation du $MCKP$ peuvent être appliqués au $AddTSK^D$. Le $MCKP$ étant connu de ne pas admettre d'algorithme d'approximation avec une garantie de performance égale à une valeur constante (si $\mathcal{P} \neq \mathcal{NP}$) (voir [LY99]), le $AddTSK^D$ n'admet pas de tel algorithme non plus. Comme une instance de l'$AddTSK^D$ peut être reformulée de façon équivalente en un TSK^D, ce dernier ne peut probablement pas être approximé avec un ratio constant.
Nous étendons ce résultat à trois autres cas particuliers : le cas où des objets peuvent seulement être retirés à la deuxième étape, le cas où les valeurs et pénalités dépendent linéairement des poids et le cas où les valeurs et pénalités de la deuxième étape sont une fraction resp. un multiple fixe de la valeur correspondante de la première étape.

Problème biniveau stochastique avec contrainte de sac-à-dos

Les modèles de biniveau servent à modéliser des problèmes d'optimisation où deux parties doivent prendre une décision dont l'une dépend de l'autre. Plus précisément, la deuxième partie fait sa décision suivant la décision de la première partie qui, elle, doit assurer que le problème de la deuxième partie est réalisable. Le problème de la deuxième partie (problème du niveau bas) fait donc partie des contraintes du problème de la première partie (problème du niveau haut). Nous étudions le cas où la première partie est en plus soumise à une contrainte de sac-à-dos probabiliste. Poids et capacité sont supposés être dépendants d'un vecteur aléatoire distribué discrètement χ. La formulation mathématique du problème traité est la suivante :

$$
\begin{aligned}
(SLBP) \quad \max_{x} \quad & c_1^t x + d_1^t y \\
\text{s.t. } & A^1 x + B^1 y \leq b^1, \\
& \mathbb{P}\{w^t(\chi) x \leq s(\chi)\} \geq (1-\alpha), \\
& \mathbb{O}_{n_x} \leq x \leq \mathbb{1}_{n_x}, \\
& y \in \arg\max_{y} c_2^t x + d_2^t y, \\
& \text{s.t. } A^2 x + B^2 y \leq b^2, \\
& \qquad y \geq 0.
\end{aligned}
$$

où $c_1, c_2, \in \mathbb{R}^{n_x}$, $d_1, d_2 \in \mathbb{R}^{n_y}$, $A^1 \in \mathbb{R}^{m_1 \times n_x}$, $B^1 \in \mathbb{R}^{m_1 \times n_y}$, $b^1 \in \mathbb{R}^{m_1}$, $A^2 \in \mathbb{R}^{m_2 \times n_x}$, $B^2 \in \mathbb{R}^{m_2 \times n_y}$, $b^2 \in \mathbb{R}^{m_2}$ et $0 < \alpha \leq 1$.

Reformulation équivalente en problème bilinéaire déterministe

Due à l'hypothèse d'une distribution discrète, la contrainte probabiliste peut être remplacée par un ensemble de contraintes linéaires déterministes. Ceci demande, par contre, l'introduction de K variables binaires au niveau haut. Dans [AHJS97] les auteurs montrent comment reformuler le problème biniveau obtenu en un problème bilinéaire avec variables continues. L'idée de base est d'ajouter les contraintes primales et duales ainsi que les contraintes de complémentarité du problème du niveau bas aux contraintes du niveau haut. Cependant, les contraintes de complémentarité rendent le problème obtenu difficile à résoudre exactement.

Relaxation et algorithme itératif

Au lieu de résoudre le problème bilinéaire obtenu exactement, nous proposons de relaxer les contraintes de complémentarité dans la fonction objectif. La relaxation obtenue est la suivante :

$$
\begin{aligned}
\text{(LGN)} \quad \min_{\lambda,\mu_1,\mu_2} \max_{x,y,z} \quad & \mathcal{L}(x, y, z, \lambda, \mu_1, \mu_2) \\
\text{s.t.} \quad & A^1 x + B^1 y \leq b^1, \\
& w_k^t x \leq s_k + M_k z_k \quad \forall\, k = 1, \ldots, K, \\
& p^t z \leq \alpha, \\
& A^2 x + B^2 y \leq b^2, \\
& (B^2)^t \lambda \geq d_2, \\
& \mathbb{I}_K \mu_1 + \mathbb{I}_K \mu_2 \geq \mathbb{1}_K, \\
& \mathbb{O}_{n_x} \leq x \leq \mathbb{1}_{n_x}, \mathbb{O}_K \leq z \leq \mathbb{1}_K, \\
& y, \lambda, \mu_1, \mu_2 \geq 0.
\end{aligned}
$$

où $\mathcal{L}(x, y, z, \lambda, \mu_1, \mu_2) = c_1^t x + d_1^t y + \lambda^t (b^2 - A^2 x - B^2 y) + \mu_1^t z + \mu_2^t (\mathbb{1}_K - z) + y^t ((B^2)^t \lambda - d_2)$. Quand on fixe les variables primales ou les variables duales on obtient un problème linéaire. En plus, les contraintes sont séparables. Nous proposons donc un algorithme qui résout itérativement le sous-problème primal et le sous-problème dual. En ajoutant une coupe à chaque sous-problème dans chaque itération, la convergence vers l'optimum de la relaxation LGN est assurée.

Résultats numériques et bornes améliorées

Les résultats numériques ont confirmés la bonne convergence de notre algorithme itératif, i.e. après quelques itérations la solution du sous-problème primal et la solution du sous-

problème dual se rapprochent à une petite différence (relative) près. En plus, la relaxation a pu être résolue pour $K = 100$ et $n_x = n_y = 1000$ en 20 minutes en moyenne.
Par contre, nous avons remarqué sur quelques instances dont nous connaissions la valeur optimale que cette dernière était plutôt loin de la solution de la relaxation. Nous proposons donc une deuxième relaxation et démontrons que celle-ci donne des meilleures bornes. Une sélection d'algorithmes de la littérature pour résoudre cette deuxième relaxation est présentée.

Perspectives

Pour tous les problèmes étudiés nous avons supposé que les poids suivent une distribution particulière, soit normale, soit discrète. Il serait donc intéressant et important de savoir, si nos méthodes proposées restent applicables si on suppose une autre distribution où si on ne fixe pas de distribution du tout. Par exemple, l'algorithme gradient stochastique utilisé pour résoudre les relaxations des problèmes de sac-à-dos avec recours simple et avec contrainte probabiliste ne demande théoriquement pas d'informations sur la distribution des variables aléatoires.
En ce qui concerne le problème de sac-à-dos avec recours et poids normalement distribués, les bornes inférieures proposées pour le cas général sont encore plutôt mauvaises. Une amélioration de ces bornes sera donc envisageable.
Nous avons montré que le problème de sac-à-dos avec recours et poids discrètement distribués n'admet très probablement pas d'algorithme d'approximation avec une garantie de performance constante. Cependant, il pourrait exister un algorithme d'approximation dont la garantie de performance augmente avec le nombre d'objets ou le nombre de scénarios, puis il reste à étudier si le problème admet un $PTAS$ ("Polynomial Time Approximation Scheme") si le nombre de scénarios est fixé.
Pour le problème stochastique biniveau avec contrainte probabiliste nous n'avons, dans cette thèse, proposé qu'un algorithme pour résoudre une relaxation. Il reste évidemment à étudier comment résoudre le problème initial exactement de manière efficace.
Un type de modèle qui n'a pas été étudié dans cette thèse est le modèle avec plusieurs étapes. Une extension des bornes proposées pour le cas avec deux étapes pourrait être possible.

Contents

1 Introduction

Stochastic optimization problems belong to the more wider class of Optimization Problems under Uncertainty. Optimization under Uncertainty arises wherever a decision has to be made that is optimal with respect to some objective, while some of the involved parameters are not exactly known or the decision depends on some future event.
Although it often seems that Optimization under Uncertainty is a much younger field of research than mathematical programming in general, it has been nearly equally long studied. The paper that is generally accepted to be the first reference on Optimization under Uncertainty was that of Dantzig from 1955 on "Linear programming under uncertainty" ([Dan55]). However, Optimization under Uncertainty has been much less intensively studied than its deterministic equivalent. There are several possible reasons.
First of all it is natural that one first tries to understand and solve the easier, deterministic variant of a problem before studying its stochastic extensions. Moreover, many problems that contain uncertain parameters can be equivalently reformulated as a deterministic problem (at least approximately). As a consequence, a great percentage of problem solving methods for optimization problems that involve uncertainties are based on algorithms from deterministic optimization.
Apart from this logical reasoning, the difficulties that are caused by the introduction of randomness are clearly another important factor why so much less research has been conducted on Optimization Problems under Uncertainty. These difficulties concern on the one hand the problem itself that is often much harder to solve than its deterministic counterpart. On the other hand, one has to judge which or how much information on the random parameters can be assumed to be known to the decision maker.

Regarding the pursued objective, Optimization Problems under Uncertainty can be divided in three classes: Robust Optimization problems, Online Optimization problems and Stochastic Optimization problems. However, this classification is not distinct as there naturally exist problems that lie in more than one class.

As the class name indicates, the objective of **Robust Optimization problems** is to find solutions that behave robustly in all possible scenarios (i.e. for all possible outcomes of the random parameters). In most cases, this means that one searches for a solution that is not too bad even in the worst case. More precisely, if we are in the case of a minimization problem, the objective is to minimize the maximum objective function value over all possible scenarios. Another common approach is to minimize the maximum regret. This means that one aims to minimize the maximum difference, over all scenarios, between optimal solution value and the solution value provided by the robust solution.

A drawback of robust approaches might be that one generally has to assume the random parameters to be bounded. This is naturally the case if the random variables only admit a finite number of outcomes. More commonly, it is assumed that the random variables are uniformly distributed over a certain interval.
We refer those readers that are interested in an introduction to robust optimization and robust measures to the very well written "comprehensive survey" by Beyer and Sendhoff [BS07] that has been published very recently. The authors also provide an overview over solution and approximation techniques for robust optimization problems that have been previously proposed in the literature, without going into details.

Online Optimization problems arise when information about the random parameters arrives over time and dynamic decisions can or have to be taken based on these informations. In general, the decision maker only makes a "partial decision" based on the information he received. Moreover, once a decision has been made, it cannot be corrected later. The solution of an online optimization problem is therefore a policy that, depending on the last information revealed, provides the optimal decision to be taken with respect to an overall objective. As an example, consider a packing problem where the dimensions of the items to pack are random, but come to be known over time. Whenever the actual dimension of an item is revealed, the decision maker has to decide instantly if he accepts the item or not. Another example arises in machine scheduling where the exact length of a task is not known before the task is completed. At this moment, the decision maker has to decide which task to schedule next. A possible solution to this online problem is the policy to always execute the task with the longest expected completion time.
More information about online algorithms and online optimization can be found in the book by Fiat and Woeginger [Fe98] and the more recent survey by Albers [Alb03]. Online algorithms for discrete structures are the subject of a short paper by Winter and Zimmermann [WZ98] and Combinatorial Online problems have been addressed in [ARGK98] (and subsequent papers). Those who are interested in the more recent field of studies of Online Stochastic Combinatorial Optimization are referred to the book by Van Hentenryck and Bent [HB06].

In **Stochastic Optimization Models** the objective is mostly to minimize or maximize an expected value or a probability.
Contrary to robust models, the worst case (or the maximum regret) generally does not have any importance for the decision maker. However, the solution should either be acceptable on average and/or the probability that bad cases arise should be small enough. The main difference with online approaches is that, even if we might allow the decision progress to take place in several stages, the solution of a stochastic optimization problem is not a policy but the exact decision for each of these stages.
Stochastic Optimization problems where random parameters appear in one or several constraints can be best categorized based on the manner that these constraints are handled: Either it is acceptable that the chosen solution might be infeasible for one or more of the possible outcomes of the random variables, or it is not.

In the first case one generally speaks of the *risk* that one of the constraints might not be satisfied. This risk is then either bounded using a *chance-constraint* (see chapter 3 for more information on chance-constraints and chance-constrained problems), or the aim is to minimize the risk with respect to a certain threshold that has been fixed for the gain (*risk aversion*, *minimum-risk problem*; see e.g. [Dra72]). Another approach is to introduce "soft" constraints that can be adapted accordingly if they are not satisfied by the chosen solution. This adaptation comes naturally with a cost. *Simple recourse problems* are examples for this problem type (a survey on Simple recourse problems can be found in the introduction of chapter 2).
In the second case, there are two possibilities: Either we request that the solution satisfies the constraints for all outcomes of the random parameters. This approach generally entails very conservative solutions as we might force the solution to even satisfy the constraints in very improbable cases. The alternative is to allow later corrections of the solution. For example, we might assume that a corrective decision can be made once all the random parameters have come to be known (*two-stage problem with full recourse*) (for more information on two-stage problems with full recourse see the introduction of chapter 4). Alternatively, decisions might take place in several stages based on partial information (*multi-stage problems*). Note that, contrary to online models, multi-stage models require in general the knowledge of which parameter comes to be known at what moment. For an introduction to the structure and theory of multi-stage problems we refer the interested reader to the book on Stochastic Programming by Kall and Mayer [KM05] and the recently published lectures by Shapiro et al. [SDR09].

In this thesis, we will study three types of stochastic models: the simple recourse model, chance-constrained models as well as the two-stage model. All studied problems have in common that they involve a so called *knapsack constraint*. A knapsack constraint is a one-dimensional constraint that describes the relation between a limited resource and a set of demands. The expression "knapsack constraint" has its origin in the example of a traveler whose knapsack has a limited volume or a restricted weight capacity. The traveler thus has to choose a maximal useful set of items that respects the knapsack capacity. In this work we study the case where the item weights (or the demands) are assumed to be random, i.e. unknown at the moment the decision of which items to pack (or which demands to fulfill) has to be made. This entails the problem that in some cases one cannot be sure if the chosen items will, in the end, respect the knapsack capacity (or resource restriction). As aforementioned, the decision of how to handle this inconvenience dictates the model that has to be chosen to mathematically represent and finally solve the problem.

When modeling a problem with uncertain parameters mathematically, a second question that has to be answered is *"How much do we (assume to) know about the random parameters?"*. The answer to this question depends, on the one hand, naturally on the respective problem. On the other hand, one also has to take into account the difficulty to solve the resulting problem. As often in mathematical modeling, the aim is thus to find a good tradeoff between accuracy and tractability of the model.

When making a so called *black-box assumption*, one presumes that it is possible to draw an arbitrary large sample of the random parameters that can be used for (resp. during) the solution process. Such a black-box (or simply an available set of observations of the random parameters) might also be used to statistically determine characteristics of the underlying distribution like for example mean and variance. In some few cases the given data might even be sufficient to allow for a total description of the distribution. It is however clear that any such approach only provides us with an approximation of the underlying distribution. Further approximating the distribution for example by assuming a limited number of discrete scenarios might thus in many cases not be much more restrictive.
In this thesis, we mainly study two cases: On the one hand, we make for some of the studied models the assumption that the underlying distribution is Gaussian and that we have statistical information about mean and standard deviation for each random parameter. On the other hand, we study for two problems the case where the underlying distribution can be represented by a finite number of realizations or samples.

In the next two sections of this introduction we will give a description of the structure of the thesis as well as short summaries of the main chapters. The introduction closes with a table of used notations and other preliminary definitions.

1.1 Remark on the structure and contents of this thesis

In the following we will present the work that we have conducted during the last 3 years. Four different types of stochastic optimization problems have been studied, namely *Simple Recourse*, *Chance-Constrained*, *Two-Stage* and *Stochastic Bilevel* problems. The structure of this thesis is based on these four problem types, i.e. we dedicate one chapter to each of the mentioned approaches.
The following four chapters all start with an introduction where we give further details and references concerning the stochastic optimization problem treated in the chapter and/or its deterministic counterpart. First, we present a general formulation of the underlying problem type and discuss some structural properties. Then we introduce the particular problem studied.
The second part of the introduction concerns on the one hand former solution methods for the respective general (stochastic or deterministic) optimization problem as well as the special problem treated (if applicable). On the other hand, we outline the solution approach that we propose for the particular problem studied.
The introduction is followed by a section where we present the treated problem and state its mathematical formulation. Properties of the problem are discussed and analyzed.
In the remaining subsections the proposed solution approach is explained in detail, followed by numerical test results and their analysis.
Each of the four following chapters is closed by a conclusion which summarizes the obtained results, discusses possible extensions and gives a prospect of future work.
Remark that chapter 4 has a slightly different structure as two variants of the Two-Stage Knapsack problem are treated. While the first part of the chapter has the abovementioned structure, the second part is organized differently as we do not propose a solution approach, but prove some non-approximability results.

In the whole thesis we mainly restrict our study to linear stochastic optimization problems. More precisely, we only consider problems where the "core functions" are linear. By core functions we mean the functions inside of possible expectations, probability measures or indicator functions. This implies however in only few cases the (piecewise) linearity of the problem itself. The introductions and literature reviews therefore mainly focus on the linear case. Moreover, to simplify notations we will not always precise that the mentioned problem is linear or that we concentrate our study or analysis on the linear variant of the problem.
Similarly, we assume for all parameters that depend on a random variable that this dependency is linear. More precisely, for a random parameter $a(\chi)$ that depends on the n-dimensional random vector $\chi = (\chi_1, \ldots, \chi_n)$ we assume that there exist deterministic coefficients $a_0, \ldots, a_n$ such that

$$a(\chi) = a_0 + \sum_{i=1}^{n} a_i \chi_i$$

Concerning the literature reviews, we restrict ourselves to theoretical and algorithmic results. Applications only play a marginal role in this thesis, mainly to illustrate some of the chosen models.

1.2 Summaries of the main chapters

Chapter 2: The Simple Recourse Knapsack Problem

In chapter 2 the Simple Recourse Knapsack problem with random weights is studied and solved. The decision of which items to put in the knapsack has to be made while the exact weights of the items are still unknown. In case the chosen items lead to an overload, a penalty per overweight unit has to be paid. The objective is to maximize the expected total benefit, i.e. the expected reward minus the expected penalty cost. In this chapter the item weights are assumed to be independently normally distributed.
Special interest is given to the corresponding continuous problem. We propose to solve this relaxation using a stochastic gradient algorithm. An Approximation by Convolution method is applied to approximate the needed gradient of a nondifferentiable function. A branch-and-bound framework is chosen in order to solve the initial combinatorial problem. Numerical results on a dataset from the literature as well as a set of randomly generated instances are presented and analyzed.

The methods proposed and results obtained in this chapter have been published in *Annals of Operations Research* ([KL10c]). They have also been presented at the international conference on *Modelling, Computation and Optimization in Information Systems and Management Sciences* in Metz (France) in September 2008.

Chapter 3: The Chance-Constrained Knapsack Problem

In chapter 3 we treat the Stochastic Knapsack problem with Chance-Constraint. Once more the weights of the items are assumed to be independently normally distributed. The introduced chance-constraint restricts the percentage of cases where the chosen items lead to an overload.
As in the previous chapter, the relaxed version of the problem is solved using a stochastic gradient algorithm in order to provide upper bounds in a branch-and-bound framework. To solve the relaxation, the chance-constraint is equivalently reformulated as an expectation constraint and included in the objective by introducing a Lagrange multiplier. Two approaches to estimate the needed subgradient of the Lagrangian dual function are applied, one based on integration by parts and one using finite differences. A theoretical analysis of the convergence properties of the applied stochastic gradient algorithm is given. As we encountered severe convergence issues during the first numerical tests, we investigated in a closer study of the problem and the used approach. It is shown in this chapter, that a better convergence of the algorithm can be obtained by a simple reformulation of the problem as well as a more careful implementation of the integration by parts method.

Part of the work presented in this chapter has been conducted in cooperation with Marc Letournel. A publication with authors S. K., Marc Letournel and Abdel Lisser is close to submission. The work has been presented at the *9th Cologne-Twente Workshop on Graphs and Combinatorial Optimization* in Cologne (Germany) in May 2010 and an extended abstract can be found in the proceedings of the workshop ([KLL10]).

Chapter 4: The Two-Stage Knapsack Problem with full recourse
In the first part of chapter 4 another particular version of the stochastic knapsack problem with normally distributed weights is studied: the Two-Stage Knapsack problem. Contrary to the stochastic single-stage knapsack problems presented in chapters 2 and 3, a corrective decision can be made at the moment the actual item weights come to be known. More precisely, items can be removed from or added to the knapsack in case the capacity constraint is not satisfied or in order to increase the total gain. A chance-constraint is introduced in the first stage in order to restrict the percentage of cases where the chosen items lead to an overload in the second stage.
Instead of solving the problem exactly, we propose methods to calculate upper and lower bounds on the optimal solution value. The upper bounds are obtained by solving a continuous relaxation of the problem. It is shown, that this relaxation is in fact a Simple-Recourse Knapsack problem with chance-constraint and methods from the previous two chapters can be applied to solve the problem. The easily computable lower bounds are designed to replace the evaluation of the objective function in the chosen branch-and-bound framework as, to the best of our knowledge, there is no practical method known to exactly compute the objective function value for a given first-stage solution. Special interest is given to the case of similar items. Numerical tests have shown that in this particular case our lower bounds are close to the exact solution value. As a consequence, the initial problem can be solved to near optimality for up to 2000 items in about 30 minutes.

The work presented in the first part of chapter 4 has been partially presented at the *8th Cologne-Twente Workshop on Graphs and Combinatorial Optimization* in Paris (France) in June 2009. An extended abstract is published in the workshop proceedings ([KL09]). The full article has been accepted for publication in the Special Issue of *Discrete Applied Mathematics* dedicated to the workshop ([KL10a]). The article is in press but can already be accessed on-line.

In the second part of chapter 4 we study the Two-Stage Knapsack problem with discretely distributed item weights. We prove that the problem does not admit a constant-factor approximation algorithm, unless $\mathcal{P} = \mathcal{NP}$. This is done by a reduction from the multiply-constrained knapsack problem. Furthermore some special cases of the Two-Stage Knapsack problem are studied that seem to be less difficult than the general case. We however show that the optimal solution values of these problems cannot be approximated to a constant factor in polynomial time, either. To the best of our knowledge, this is the first study of Two-Stage Knapsack problems under the aspect of approximability.

Chapter 5: The Stochastic Bilevel Problem with Knapsack Chance-Constraint
In chapter 5 a mixed integer bilevel problem having a knapsack chance-constraint in the upper level is proposed. The problem formulation is mainly motivated by practical pricing and service provision problems as it can be interpreted as a model for the inter-

action between a service provider and clients. The probability space is assumed to be finite which allows to transform the problem into a deterministic equivalent mixed integer bilevel problem. Via a reformulation as a linear bilevel problem, a quadratic optimization problem, the so called Global Linear Complementarity problem, is obtained. Based on this quadratic (or, more precisely, bilinear) problem, we propose a procedure to compute upper bounds for the initial problem by solving a Lagrangian relaxation using an iterative linear minmax scheme. We prove our scheme to converge to an optimal solution of the relaxation, which implies the convergence to an ϵ-optimal solution in finitely many iterations. The convergence has been confirmed by a range of numerical tests. We however remarked that on some instances the obtained bounds are quiet far from the optimum. We thus propose a second relaxation that is shown to provide better bounds than the previous one. As the iterative scheme used to solve the initial relaxation is not suitable to solve this second relaxation, an overview of methods from the literature that could instead be applied is given.

Most of the work presented in this chapter has been conducted in cooperation with Pierre Le Bodic and Janny Leung. It has been presented at the *International Network Optimization Conference* in Pisa (Italy) in April 2009. An extended abstract can be found in the proceedings ([KBLL09]). The corresponding full article with authors S.K., Pierre Le Bodic, Janny Leung and Abdel Lisser has been accepted for publication in *Networks* ([KBLL10]).

1.3 Preliminaries

List of frequently used notations:

$\mathbb{E}[X]$	expectation of a random variable X
$\mathbb{P}\{A\}$	probability that event A occurs
$[x]^+$	if $x \in \mathbb{R}$: $[x]^+ := \max(0, x)$ if $x \in \mathbb{R}^n$ $(n \geq 2)$: $[x]^+ := (\max(0, x_1), \ldots, \max(0, x_n))^T$
J	objective function
j	in case J is a function in expectation, j denotes the function inside the expectation, i.e. $J(x) = \mathbb{E}[j(x, \chi)]$
I_n	identity matrix of dimension $n \times n$
$\mathbb{1}_n$	vector of ones of dimension n
$\mathbb{0}_n$	vector of zeros of dimension n
$\mathbb{1}_{\mathbb{R}^n}$	Indicator function of the positive real interval (Heaviside function)
$\mathbb{1}_{\mathcal{I}}$	Indicator function of the interval or set $\mathcal{I}$, i.e. $[\mathbb{1}_{\mathcal{I}}(x) = 1 \Leftrightarrow x \in \mathcal{I}]$
x	denotes in general the first-stage (or upper level) decision vector
y	denotes in general the second-stage (or lower level) decision vector
χ	vector of random variables
μ	vector of means of the components of χ
σ	vector of standard deviations of the components of χ
f	density function of the standard normal distribution
F	cumulative distribution function of the standard normal distribution

Used abbreviations:

B&B	branch-and-bound
CDF	cumulative distribution function
PTAS	Polynomial Time Approximation Scheme
FPTAS	Fully Polynomial Time Approximation Scheme

Throughout this thesis, when mentioning a *discrete probability distribution* or a *discretely distributed random variable*, we always refer to a random variable that can only take a finite number of values with nonzero probability.

2 The Simple Recourse Knapsack Problem

2.1 Introduction

2.1.1 Simple Recourse Problems

Simple recourse problems are special cases of the more general two-stage problems (see section 4 for more information on two-stage models). This means that decisions are made in two steps: the first decision while the random variables are still unknown, the second, corrective decision after the outcomes of the random variables have come to be known.
While simple recourse problems already appear in one of the first papers on stochastic two-stage programming by Dantzig [Dan55], their detailed study only began with a publication of Wets [Wet66].
Simple recourse problems can be stated in general as follows:

$$
\begin{aligned}
\min \quad & c^T x + \mathbb{E}\left[\mathcal{Q}(x,\chi)\right] \\
\text{s.t.} \quad & Ax = b, && \text{(2.1a)}\\
& x \in X. && \text{(2.1b)}\\
& \mathcal{Q}(x,\chi) = \min q^+ y^+ + q^- y^- && \text{(2.1c)}\\
& \qquad \text{s.t. } T(\chi)x + y^+ \geq h(\chi), && \text{(2.1d)}\\
& \qquad\quad T(\chi)x - y^- \leq h(\chi), && \text{(2.1e)}\\
& \qquad\quad y^+ \in Y^+,\ y^- \in Y^-. && \text{(2.1f)}
\end{aligned}
$$

where $c \in R^{n_1}$, $A \in \mathbb{R}^{m_1 \times n_1}$, $b \in \mathbb{R}^{m_1}$, $q^+, q^- \in \mathbb{R}^{n_2}_+$ and $q^+ + q^- > 0$. $x \in X \subseteq \mathbb{R}^{n_1}_+$ is the *first-stage decision vector* of dimension n_1 and $y^+ \in Y^+ \subseteq \mathbb{R}^{n_2}_+$ and $y^- \in Y^- \subseteq \mathbb{R}^{n_2}_+$ are the two *second-stage decision vectors* of dimension n_2.
The *technology matrix* $T(\chi) \in \mathbb{R}^{m_2 \times n_2}$ of a two-stage or simple recourse problem is the second-stage constraint matrix that multiplies the first-stage decision vector. In most references on simple recourse problems this matrix is assumed to be fix, i.e. independent of the random vector. In some publications (especially the early ones such as [Wet83]) the definition of a simple recourse problem even contains this assumption as a requirement.
In the case of *continuous simple recourse problems* we have $X = \mathbb{R}^{n_1}_+$ and $Y^+ = Y^- = \mathbb{R}^{n_2}_+$. *Simple integer recourse problems* are defined as problems of type (2.1) where the second-stage decision variables are required to be integer, i.e. $Y^+ = Y^- = \mathbb{Z}^{n_2}_+$ (see e.g. [LvdV93] or [vdV95]).

Continuous Simple Recourse Problems

When studying the second-stage problem of (2.1), one remarks that the second-stage variable y_i^+ is bounded from below by $\max(0, (h(\chi) - T(\chi)x)_i)$, while the second-stage variable y_i^- is bounded from below by $\max(0, (T(\chi)x - h(\chi))_i)$. In case of continuous second-stage decision variables and as $q^+, q^- \geq 0$, problem (2.1) can thus be reformulated as a single-stage problem in the following way:

$$\begin{aligned} \min \quad & c^T x + \mathbb{E}\left[q^+[h(\chi) - T(\chi)x]^+\right] + \mathbb{E}\left[q^-[T(\chi)x - h(\chi)]^+\right] \\ \text{s.t.} \quad & Ax = b, && \text{(2.2a)} \\ & x \in X. && \text{(2.2b)} \end{aligned}$$

For all $i = 1, \ldots, n_2$, let $T_i(\chi)$ denote the i-th row of $T(\chi)$ and $h_i(\chi)$ the i-th component of $h(\chi)$. Then define the new random variable $\xi_i(x) := h_i(\chi) - T_i(\chi)x$. Using this definition, problem (2.2) can be reformulated as follows:

$$\begin{aligned} \min \quad & c^T x + \sum_{i=1}^{n_2} \left(q_i^+ \mathbb{E}\left[[\xi_i(x)]^+\right] + q_i^- \mathbb{E}\left[[-\xi_i(x)]^+\right]\right) \\ \text{s.t.} \quad & Ax = b, && \text{(2.3a)} \\ & x \in X. && \text{(2.3b)} \end{aligned}$$

Note that this formulation only contains expectations of one-dimensional random variables. In particular we have (see e.g. [vdV95])

$$\mathbb{E}\left[[\xi_i(x)]^+\right] = \int_0^\infty \xi_i(x)\phi_x^i(\xi_i(x))d\xi_i(x) = \int_0^\infty (1 - \Phi_x^i(t))dt$$

and

$$\mathbb{E}\left[[-\xi_i(x)]^+\right] = \int_{-\infty}^0 \xi_i(x)\phi_x^i(\xi_i(x))d\xi_i(x) = \int_{-\infty}^0 \Phi_x^i(t)dt$$

where ϕ_x^i is the density and Φ_x^i the cumulative distribution function of $\xi_i(x)$.
We define

$$g^i(x) := q_i^+ \mathbb{E}\left[[\xi_i(x)]^+\right] + q_i^- \mathbb{E}\left[[-\xi_i(x)]^+\right].$$

The following results are well known and basis of most solution approaches for simple recourse problems (see for example [NW86] or [vdV95]):

Theorem 2.1. *Let $\mathbb{E}[\xi_i(x)]$ be finite. Then the following holds:*

1. *For $q_i^+ = q_i^- = 1$ it is $g^i(x) = \mathbb{E}[\xi_i(x)]$.*
2. *$g^i(x) = q_i^- \mathbb{E}[\xi_i(x)] + (q_i^+ - q_i^-)\mathbb{E}[[\xi_i(x)]^+]$*
3. *$g^i : \mathbb{R}^{n_1} \to \mathbb{R}$ is a convex function.*
4. *For all $x \in \mathbb{R}^{n_1}$ $g^i(x) < \infty$.*

5. *g^i is Lipschitz continuous.*

6. *g^i is differentiable wherever Φ_x^i is continuous.*

7. *If $\xi_i(x)$ is discretely distributed, $g^i$is piecewise linear.*

It follows that continuous simple recourse problems have a Lipschitz continuous and convex objective function that is differentiable almost everywhere. They are thus in fact "simple" to solve compared to general two-stage problems (see chapter 4).

Simple Integer Recourse

In case of integer second-stage decision variables (without upper bound) the objective function of (2.2) can be replaced by

$$c^T x + \mathbb{E}\left[q^+ \lceil h(\chi) - T(\chi)x \rceil^+\right] + \mathbb{E}\left[q^- \lceil T(\chi)x - h(\chi) \rceil^+\right]$$

Solving simple integer recourse problems is generally harder in the sense that one might loose both the convexity as well as continuity properties. Concerning the latter, the objective function of a simple integer recourse problems can be shown to be lower semi-continuous. Moreover, it is continuous if and only if the random variables χ_i are continuously distributed ([vdV95]). This implies that in case of discrete random variables we also loose the convexity property. However, even in the case of continuously distributed random variables convexity cannot be guarantied (see [KHSvdV06] for details). Therefore a lot of work has been dedicated to the study of simple integer recourse problems ([LvdV93],[KHvdV94],[vdV95]), with special interest given to the question of convexity and convexification ([KHSvdV95], [KHSvdV96],[KHSvdV06]). Most of these publications, however, concern the case where the technology matrix is fixed.

2.1.2 The Simple Recourse Knapsack Problem ($SRKP$)

In the main part of this chapter we will study the Simple Recourse Knapsack problem with random weights. This problem is a simple recourse problem with binary first- and continuous second-stage variables and a random technology vector. It can be described in two equivalent ways:
When interpreted as a recourse model, the problem can be stated as follows: In the first stage, we can choose among a set of items a subset to be added to the knapsack. This decision is made without the exact knowledge of the item weights. In the second stage, we are able to buy additional capacity in order to assure that the items chosen in the first stage respect the capacity constraint. The overall objective is to maximize the expected total gain, i.e. the benefit from the chosen items minus the expected cost of the additionally purchased capacity.
The problem can be seen as a single-stage decision problem as well. This means, that in case that the chosen items do not respect the capacity, we simply have to pay a penalty per overweight unit, i.e. no effective "decision" has to be made once the item weights

have been revealed.
Notice that in both interpretations a correction of the first-stage decision is not possible, i.e. the items chosen in the first stage cannot be removed (or additional items added) in the second stage (see chapter 4).
In this chapter, we only work with the single-stage variant of the problem. Its mathematical formulation is given in subsection 2.2. Some properties of the Simple Recourse Knapsack problem are discussed in section 2.2.1.

2.1.3 Solving Simple Recourse Problems with continuous recourse

For most stochastic optimization problems one of the main difficulties consists in the evaluation of the objective function, as this generally incorporates the computational expensive estimation of a multi-dimensional integral. However, we have seen in section 2.1.1 that in case of a continuous simple recourse problem this multi-dimensional integral is separable in the sense that it can be replaced by a sum of 1-dimensional integrals of form similar to

$$\int_0^\infty \xi_i(x)\phi_i(\xi_i(x))d\xi_i(x) = \int_0^\infty (1-\Phi_x^i(t))dt$$

where $\xi_i(x) = h_i(\chi) - T_i(\chi)x$ is a one dimensional random variable with density and cumulative distribution function ϕ_x^i and Φ_x^i, respectively. Moreover, continuous simple recourse problems have some nice properties like convexity and (piecewise) differentiability (see Theorem 2.1). They can therefore often be solved efficiently with standard algorithms for convex optimization.

Discretely distributed random variables

As with most stochastic optimization problems, a simple recourse problem can be equivalently reformulated as a deterministic problem in case of discretely distributed random variables (with known distribution). Let $\chi^1, \dots, \chi^K$ denote the possible outcomes of the random vector χ and $p^1, \dots, p^K$ the corresponding probabilities. Then the simple recourse problem (2.1) is equivalent to the following programming problem with linear objective and constraints:

$$\begin{aligned}
\min \quad & c^T x + q^+ \sum_{k=1}^K p^k y_+^k + q^- \sum_{k=1}^K p^k y_-^k \\
\text{s.t.} \quad & Ax = b, && \text{(2.4a)} \\
& y_+^k \geq h(\chi^k) - T(\chi^k)x \quad \forall\, k = 1, \dots, K, && \text{(2.4b)} \\
& y_-^k \geq T(\chi^k)x - h(\chi^k) \quad \forall\, k = 1, \dots, K, && \text{(2.4c)} \\
& y_+^k \geq 0,\ y_-^k \geq 0 \quad \forall\, k = 1, \dots, K, && \text{(2.4d)} \\
& x \in X,\ y^+ \in Y^+,\ y^- \in Y^-. && \text{(2.4e)}
\end{aligned}$$

The number of variables as well as the number of constraints, however, grows with the number of considered scenarios. Moreover, if χ is an n-dimensional random vector with

independently distributed components and if we assume for its i^{th} component K_i possible outcomes, we obtain a total number of $K = \prod_{i=1}^{n} K_i \geq (\min_{i\in\{1,\dots,n\}} K_i)^n$ scenarios, i.e. the number of scenarios is exponential in the number of components of χ. So solving (2.4) using common methods for linear programming might already be computationally cumbersome for relatively small $K_i \geq 2$.

The case of a fixed technology matrix: The first efficient algorithm specifically created for simple recourse problems has been presented by Wets in 1983 [Wet83]. The author assumes a fixed technology matrix and a discretely distributed random right hand side vector. The basic idea is to exploit the convexity and piecewise linearity of the problem in order to obtain an equivalent reformulation that significantly reduces the number of constraints. More precisely, one obtains a linear programming problem with same number of constraints as the problem obtained from (2.1) by replacing the random variables by their mean. To solve the reformulation the author of [Wet83] applies an extension of the revised simplex method. Although the number of variables in the reformulated problem is of course much higher than in the original problem, numerical tests have shown that the computing time needed to solve the simple recourse problem is comparable to that needed to solve its deterministic counterpart.

The case of a random technology matrix: As already mentioned, most methods that have been proposed for the continuous simple recourse problem assume the technology matrix to be fixed. To the best of our knowledge, the only algorithm that has been proposed specifically for simple recourse problems with random technology matrix is that of Klein Haneveld and van der Vlerk proposed in [HvdV06]. Their algorithm is a so called L-shaped method commonly used to solve stochastic problems with recourse (see the introduction of chapter 4). However, contrary to the general L-shaped method the algorithm presented in [HvdV06] makes use of the separability of the objective function. As a consequence, less subproblems have to be solved at each iteration and the use of the closed form of the recourse function leads to a decrease of the computational effort.

Continuously distributed random variables

For the case of continuously distributed random variables it has already been assumed relatively early that the simplest approach is, in general, to approximate the continuous distribution by a discrete one: On the one hand, the papers by Kall ([Kal74]) and by Olsen ([Ols76]) show that by applying this idea one can approximate the optimal solution (value) of a simple recourse problem with continuously distributed random variables to any desired precision. More important, it is possible to calculate error bounds and iteratively improve upon obtained approximate solutions without resolving the problem entirely at each iteration (see last section of [Wet83]).
Nowadays, a common approach to solve stochastic optimization problems with continuously distributed random variables, with an intractable large number of scenarios or under the assumption of a black-box model is the Sample Average Approximation (SAA)

method (e.g. [PFaS96],[Sha96],[SH98],[KSH02],[SS04]). The SAA method is especially interesting if the objective function cannot be easily evaluated. It can, in its simplest form, be applied to problems of the following type:

$$\min_{x \in X} \quad J(x) = \mathbb{E}\left[j(x, \chi)\right] \tag{2.5a}$$

where X is a deterministic set and j is an easily computable function for given x and χ. Most results from the literature concerning SAA approaches can therefore directly be applied to the single-stage representation of the simple recourse problem (2.2).
Solving problem (2.5) by an SAA method means solving the following problem:

$$\min_{x \in X} \quad \hat{J}_N(x) = \frac{1}{N} \sum_{k=1}^{N} j(x, \chi^k) \tag{2.6a}$$

where $\{\chi^1, \ldots, \chi^K\}$ is an independently and identically distributed sample of the random vector χ. Note that the SAA method is not a solution algorithm but serves to approximate the given problem by another one.
Two particular continuous distributions should be mentioned here: In [Bea61] the author shows that in case of uniformly distributed right hand sides the continuous simple recourse problem with fixed technology matrix can be equivalently reformulated as a deterministic convex quadratic optimization problem. In case of independently normally distributed components of the technology matrix, problem (2.2) is equivalent to a deterministic convex optimization problem with easily computable objective function and gradient (see section 2.2.1 for an example of such reformulation). This is mainly based on the fact that the linear combination of independently normally distributed random variables is normally distributed, as well.

Integer first-stage variables

The problem treated in this chapter is neither a continuous simple recourse problem, nor a simple recourse problem with integer recourse: While the first-stage variables are required to be integer (more precisely, binary), the second-stage variables are continuous. To the best of our knowledge, no outstanding special purpose algorithm for this class of problems has been presented in the literature.
It is clear that, based on the work by Wets [Wet83], the question of how to solve such a problem is mainly the same as for solving general linear programs with integer variables. Therefore, methods such as branch-and-bound, branch-and-cut or other integer programming frameworks that involve solving linear subproblems are also efficient for this kind of problems. Furthermore, some algorithms have been proposed for general two-stage problems with integer (or binary) first-stage solution, starting with the work of Wollmer [Wol80].
Some few papers can be found that solve particular variants of simple recourse problems

with integer first- and continuous second-stage variables, in general related to a practical problem such as capacity expansion ([Lag98]), vehicle routing [LLM92], shortest path ([VAK+03]) or the knapsack problem (see subsection 2.1.4).

2.1.4 Solving Simple Recourse Knapsack Problems

The Simple Recourse Knapsack problem has been solved in several publications:
The first to mention this variant of the stochastic knapsack problem were Cohn and Barnhart in [CB98]. Their study is motivated by a transportation problem. They propose three different kinds of upper bounds and use these in a specially created branch-and-bound (B&B) framework. The latter is based on a dominance relationship and a ranking of the items based on the mean and standard deviation of their weights (see sections 2.3.2 and 2.4 for more details). The authors assume independently normally distributed weights in order to obtain a closed form for the objective function.
In [KSH02] the authors solve the Simple Recourse Knapsack problem by the use of an *SAA* Approach. The proposed algorithm does not require any information about the underlying probability distribution. However, the authors assume the weights to be independently normally distributed in order to be able to evaluate the quality of the obtained solutions. The *SAA* reformulation is solved using a B&B algorithm. Upper bounds are given by the solution of the relaxed linear problem.
Recently, Fortz et al. ([FLLP08]) presented computational results for the case of normally distributed weights: They use the fact that the obtained Simple Recourse Knapsack problem is a convex optimization problem with continuously differentiable objective function that admits a representation in closed form. They use a nonlinear solver from the literature to solve the problem.
For the case of Poisson distributed weights the authors of [AG09] propose to solve the deterministic equivalent formulation of the Simple Recourse Knapsack problem using a B&B framework. Upper bounds are obtained by solving continuous relaxations based on two observations: First, there always exists an optimal solution whose components are all integer, except for at most one. Second, there exists an easily computable order of the components of the solution vector such that all components that are lower in this order than the fractional component have value zero, and all components that are higher have value one. Moreover, the range of components that might be fractional is restricted. It thus suffices to solve simple one dimensional problems for each of these (potentially fractional) components. Verification of the Karush-Kuhn-Tucker conditions allows for the immediate identification of an optimal solution.
We propose to solve the Simple Recourse Knapsack problem using a slightly improved version of the B&B algorithm by Cohn and Barnhart ([CB98]). The B&B algorithm is one of the most common ways to solve deterministic knapsack problems. One of the first papers in which the author solves the knapsack problem using a B&B algorithm is [Kol67]. In [MT77] a method to calculate upper bounds for the $0-1$ knapsack problem is presented and used within a B&B algorithm. Recent work has been published in [SSL07] where the authors propose a B&B algorithm for the more general polynomial knapsack

problem. An example for a publication that uses the B&B algorithm to solve a stochastic version of the knapsack problem is [CSW93].
Contrary to the work of Cohn and Barnhart, we propose to provide upper bounds for the B&B algorithm by solving continuous relaxations of the Simple Recourse Knapsack problem using a *stochastic gradient algorithm*. Stochastic gradient algorithms are designed to solve (unconstrained) problems with the objective to optimize the expectation of a certain value. This type of algorithms can be seen as a combination of Monte-Carlo approximation and gradient method. More precisely, instead of using the gradient of the objective function, the algorithm uses the gradient of the function inside the expectation. By computing this gradient for a (different) sample of the random variables in each iteration the expectation of the objective is approximated.
First papers on such iterative stochastic approximation methods where released in the middle of the last century ([RM51], [KW52],[EN67]). Since then, an extensive amount of theoretical results on the convergence of the stochastic gradient algorithm and its variants has been published ([Pol90], [LY98]). The method has found many applications, particularly in machine learning and control theory. For a survey, see the books by [NH76] and by [KY03].
Clearly, solving the continuous Simple Recourse Knapsack problem with a stochastic gradient algorithm seems problematic as the function inside the expectation of the objective function is not differentiable at all points (see the mathematical formulation of the $SRKP$ (2.7)). We therefore approximate the needed gradient using approximation by convolution (three more methods are presented in chapters 3 and 4). In [ACVA07] this approach has been proposed for chance-constrained optimization problems (see also chapter 3). The underlying *mollifier technique*, however, has already been used beforehand, for example to minimize semi-continuous functions (see [ENW95]).
In this chapter we assume the item weights to be independently normally distributed. The main impact of this assumption on our solution technique is that we have a closed deterministic equivalent reformulation of the objective function which allows for its simple evaluation. Moreover, we can directly compare our algorithm with the work of Cohn and Barnhart [CB98]. We, however, think that our method (contrary to previously proposed methods) can be easily extended to other continuous distributions or even to the case of a black-box assumption (see also the conclusion 2.5 and Future Work section 6.1).

2.2 Mathematical formulation

In the remainder of this chapter, we consider a stochastic knapsack problem of the following form: Given a knapsack of capacity c and a set of n items. Each item has a weight that is not known in advance, i.e. the decision of which items to put in the knapsack has to be made without the exact knowledge of their weights. Therefore, we handle the weights as random variables and assume that weight χ_i of item i is independently normally distributed with mean $\mu_i > 0$ and standard deviation σ_i. Furthermore, each item has a predetermined reward per weight unit $r_i > 0$. The choice of a reward per weight

unit can be justified by the fact that the value of an item often depends on its weight which we do not know in advance. However, all the methods applied and studies made in this chapter are still valid if the item rewards were independent of the item weights, either deterministic or random with known mean. In case the chosen items do not respect the knapsack capacity, a penalty $d > 0$ per overweight unit has to be paid. A possible mathematical formulation of this problem is the following:

The Simple Recourse Knapsack problem ($SRKP$)

$$\max_{x\in\{0,1\}^n} \mathbb{E}[\sum_{i=1}^{n} r_i\chi_i x_i] - d \cdot \mathbb{E}[[\sum_{i=1}^{n} \chi_i x_i - c]^+] \tag{2.7}$$

2.2.1 Properties of the $SRKP$

As we are in the case of a single recourse problem with binary first-stage but continuous second-stage decision variables, the objective function can be shown to be concave and Lipschitz continuous for any probability distribution of the random vector χ (see Theorem 2.1).
In the case of a discretely distributed random vector χ with known distribution, the $SRKP$ can be reformulated as a deterministic linear combinatorial optimization problem: For $i \in \{1,\dots,n\}$, let $\chi^1,\dots,\chi^K$ be the possible outcomes of χ and $p^1,\dots,p^K$ the corresponding probabilities. Then problem (2.7) can be stated as follows:

The Simple Recourse Knapsack problem with discretely distributed weights

$$\max_{x,z} \quad \sum_{i=1}^{n} r_i \left(\sum_{k=1}^{K} p^k \chi_i^k\right) x_i - d \cdot \sum_{k=1}^{K} p^k z^k \tag{2.8}$$

$$\text{s.t.} \quad z_k \geq \sum_{i=1}^{n} \chi_i^k x_i - c \quad \forall\, k = 1,\dots,K, \tag{2.9}$$

$$z_k \geq 0 \quad \forall\, k = 1,\dots,K, \tag{2.10}$$

$$x \in \{0,1\}^n. \tag{2.11}$$

$$\tag{2.12}$$

For the case of continuously distributed weights, let us define the random variable $\xi_x := \sum_{i=1}^{n} \chi_i x_i$. Then problem (2.7) can be stated as follows:

The Simple Recourse Knapsack problem with continuously distributed weights

$$\max_{x\in\{0,1\}^n} \sum_{i=1}^{n} r_i\mu_i x_i - d \cdot \int_c^{\infty} (\xi_x - c)\varphi_x(\xi_x)\, \mathrm{d}\xi_x \tag{2.13}$$

where φ_x is the density function of ξ_x. In the particular case of independently normally distributed random variables χ_i, ξ_x is normally distributed with mean $\hat{\mu}_x := \sum_{i=1}^{n} \mu_i x_i$,

standard deviation $\hat{\sigma}_x := \sqrt{\sum_{i=1}^{n} \sigma_i^2 x_i^2}$, density function $\varphi_x(\xi) = \frac{1}{\hat{\sigma}_x} f(\frac{\xi_x - \hat{\mu}_x}{\hat{\sigma}_x})$ and CDF $\Phi_x(y) = F(\frac{y-\hat{\mu}_x}{\hat{\sigma}_x})$. Based on these definitions, we can rewrite the objective function J of the $SRKP$ in a deterministic way using the following:

$$\begin{aligned}
\mathbb{E}[[\sum_{i=1}^{n} \chi_i x_i - c]^+] &= \int_c^\infty (\xi_x - c) \cdot \varphi_x(\xi_x) \, \mathrm{d}\xi_x \\
&= \int_c^\infty \xi_x \cdot \varphi_x(\xi_x) \, \mathrm{d}\xi_x - c \int_c^\infty \varphi_x(\xi_x) \, \mathrm{d}\xi_x \\
&= \hat{\mu} \int_c^\infty \varphi_x(\xi_x) \, \mathrm{d}\xi_x + \hat{\sigma}_x^2 \int_c^\infty \varphi_x'(\xi_x) \, \mathrm{d}\xi_x - c \int_c^\infty \varphi_x(\xi_x) \, \mathrm{d}\xi_x \\
&= \hat{\sigma}_x^2 \varphi_x(c) + (\hat{\mu}_x - c) \left[1 - \Phi_x(c)\right] \\
&= \hat{\sigma}_x \cdot f(\frac{c - \hat{\mu}_x}{\hat{\sigma}_x}) + (\hat{\mu}_x - c) \cdot \left[1 - F(\frac{c - \hat{\mu}_x}{\hat{\sigma}_x})\right]
\end{aligned}$$

This leads to the following deterministic equivalent objective function in case of independently normally distributed weights:

$$J_{det}(x) = \sum_i r_i \mu_i x_i - d \cdot \left[\hat{\sigma}_x \cdot f\left(\frac{c - \hat{\mu}_x}{\hat{\sigma}_x}\right) - (c - \hat{\mu}_x) \cdot \left[1 - F\left(\frac{c - \hat{\mu}_x}{\hat{\sigma}_x}\right)\right]\right] \tag{2.14}$$

Clearly, normally distributed random variables can have negative realizations. Assuming normally distributed weights might thus seem contradictory to the fact that item weights are always strictly positive. However, in most real life applications the ratio variance/mean of the unknown parameters is rather small. In this case, the probability of negative weights becomes negligible.

2.3 Problem Solving Method

Our basic idea to solve the $SRKP$ is to use a B&B framework as presented in [CB98] and to solve relaxations of the $SRKP$ in order to provide upper bounds. As the objective function of the $SRKP$ is concave, the continuous relaxations are concave optimization problems and we propose to apply a stochastic gradient algorithm to solve them (see Algorithm 2.3.1). While such an algorithm could be used even if the probability distribution is unknown and only a black-box is available, we assume in this chapter that the random variables are independently normally distributed, for two reasons:

1. In case of independently normally distributed random variables the objective function can be easily computed and the quality of the found solutions can thus be evaluated easily (see section 2.2.1).

2. In case of independently normally distributed random variables we can apply the B&B algorithm presented by Cohn and Barnhart in [CB98] and compare our results with the results obtained when using their approach.

This subsection is subdivided into two parts: In the first one we present the stochastic gradient algorithm. As this algorithm requires the computation of the gradient of the non-differentiable function $[x]^+$, we propose a method using approximation by convolution (hereafter called *AbC* method) to obtain an estimate of this gradient. In the second subsection, we present the applied B&B algorithm.

2.3.1 The Stochastic Gradient Algorithm and Approximation by Convolution Method

A stochastic gradient algorithm is an algorithm that combines both Monte-Carlo techniques and the gradient method often used in convex optimization. Here, the former is used to approximate the gradient of the objective function that is a function in expectation. More precisely, if the objective function is $J(x,\chi) = \mathbb{E}[j(x,\chi)]$, we use at iteration $k+1$ the gradient $\nabla_x j(x^k,\chi^k)$ (where χ^k is a random sample of χ) instead of $\nabla_x J(x^k,\chi)$. In the case of the $SRKP$, we have $j(x,\chi) = \sum_i r_i\chi_i x_i - d\cdot[\sum_{i=1}^n \chi_i x_i - c]^+$. As j is not

Stochastic Gradient Algorithm

- Choose x^0 in $X_{ad} = [0,1]^n$
- At iteration $k+1$, draw a sample $\chi^k = (\chi_1^k, ..., \chi_n^k)$ of χ according to its normal distribution
- Update x^k as follows:

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

  where $r^k = \nabla_x j(x^k,\chi^k)$ and $(\epsilon^k)_{k\in\mathbb{N}}$ is a σ-sequence
- For all $i = 1, ..., n$: If $x_i^{k+1} > 1$ set $x_i^{k+1} = 1$ and if $x_i^{k+1} < 0$ set $x_i^{k+1} = 0$

Algorithm 2.3.1

differentiable at all points, we approximate its gradient using approximation by convolution (for further details on this method see [ENW95] or [ACVA07]). More precisely we approximate the indicator function $\mathbb{1}_{\mathbb{R}^+}$: It is replaced by the convolution of $\mathbb{1}_{\mathbb{R}^+}$ and a function $h_t(x) := \frac{1}{t}h\left(\frac{x}{t}\right)$ that approaches the Dirac function when the parameter t goes

to zero. The convolution of two functions is defined as follows:

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

Using a pair, continuous and nonnegative function h with $\int_{-\infty}^{\infty} h(x)\,\mathrm{d}x = 1$ having its maximum in 0, we get the following approximation of a locally integrable real valued function ρ:

$$\rho_t(x) := (\rho * h_t)(x) = \frac{1}{t}\int_{-\infty}^{\infty} \rho(y)h\Big(\frac{y-x}{t}\Big)\,\mathrm{d}y$$

For $\rho = \mathbb{1}_{\mathbb{R}^+}$, we have:

$$\rho_t(x) = \frac{1}{t}\int_{0}^{\infty} h\Big(\frac{y-x}{t}\Big)\,\mathrm{d}y = \frac{1}{t}\int_{0}^{\infty} h\Big(\frac{x-y}{t}\Big)\,\mathrm{d}y$$

and so

$$(\rho_t)'(x) = \frac{1}{t^2}\int_{0}^{\infty} h'\Big(\frac{x-y}{t}\Big)\,\mathrm{d}y = -\frac{1}{t}h\Big(\frac{x}{t}\Big)$$

Based on this, we obtain the following approximation $\nabla(j_t)_x$ of the gradient of the function j:

$$\begin{aligned}\nabla(j_t)_x(x,\chi) &= (r_1\chi_1,...,r_n\chi_n)^T \\ &\quad - d\cdot\left(-\frac{1}{t}\cdot h\left(\frac{\sum_i \chi_i x_i - c}{t}\right)\cdot\chi\cdot(\sum_{i=1}^{n}\chi_i x_i - c) + \mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n}\chi_i x_i - c)\cdot\chi\right)\end{aligned}$$

Various functions may be chosen for h. In [ACVA07] the authors study different such choices. For each one of them, they compute a reference value for the mean square error of the obtained approximated gradient. It turns out that, among the presented functions, $h := \frac{3}{4}(1-x^2)\mathbb{1}_{]-1,1[}(x)$ is the best choice. This leads us to the following estimation of the gradient of j:

$$\begin{aligned}\nabla(j_t)_x(x,\chi) &= (r_1\chi_1,...,r_n\chi_n)^T \\ &\quad + d\cdot\left(\frac{3}{4t}\left(1-\left(\frac{\sum_i \chi_i x_i - c}{t}\right)^2\right)\mathbb{1}_1\left(\frac{\sum_i \chi_i x_i - c}{t}\right)\cdot\chi\cdot(\sum_{i=1}^{n}\chi_i x_i - c)\right. \\ &\qquad\qquad \left. - \mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n}\chi_i x_i - c)\cdot\chi\right)\end{aligned}$$

2.3.2 The Branch-and-Bound Algorithm

To calculate lower bounds on the optimal solution value, we use a B&B algorithm based on an algorithm presented in [CB98]. In the following subsection we first explain and justify the ranking of the items using dominance relationships. Then, we present the B&B algorithm (see algorithm 2.3.2).

Ranking the items

In order to define the binary tree used in the B&B algorithm, we rank our items: First, we introduce dominance relationships. The items are then ranked according to the number of items they dominate. If several items dominate the same number of items, they are ranked by their value of $\frac{r_i^2}{\sigma_i}$.
The dominance relationships are also used to prune subtrees during the B&B algorithm in order to decrease the number of considered nodes and evaluated branches: Whenever an item is rejected, we also reject all those items that are dominated by the rejected one.
To introduce dominance relationships for the $SRKP$ with normally distributed weights, we consider the variations of the (deterministic equivalent) objective function J_{det} (2.14). Clearly, the increase of one of the rewards per weight unit r_j increases the objective function if and only if $x_j > 0$.
To study the variations when changing the value of $\hat{\sigma}_x$, we calculate the derivative of J_{det} with respect to $\hat{\sigma}_x$:

$$\frac{\partial J_{det}}{\partial \hat{\sigma}_x}(x) = -d \cdot f\left(\frac{c - \hat{\mu}_x}{\hat{\sigma}_x}\right)$$

As f is strictly positive, this shows that whenever an item is replaced by another one having the same mean and reward per weight unit but smaller variance, the value of the objective function increases. Based on this study, [CB98] introduced two types of dominance relationships: We say that item i dominates item k if one of the following holds:

1. $\mu_i = \mu_k$, $r_i \geq r_k$, $\sigma_i \leq \sigma_k$
2. $\mu_i \leq \mu_k$, $\sigma_i \leq \sigma_k$, $r_i \cdot \mu_i \geq r_k \cdot \mu_k$

The Branch-and-Bound Algorithm

B&B algorithm 2.3.2 is based on the B&B algorithm by [CB98]. We just added step 4.
In step 5 the calculation of upper bounds for subtrees is realized by fixing the value of items that are higher in the tree at 1 or 0 and solving the continuous problem having the x_i of the remaining items as decision variables. In the case of the $SRKP$ and the applied stochastic gradient algorithm this is easily done: At each iteration, we just leave out the recalculation of the fixed x_i.

Branch-and-Bound Algorithm

1. Rank the items as described in section 2.3.2. This ranking defines the binary tree with the highest ranked item at the root.
2. Plunge the tree as follows: Beginning at the root of the tree, add the current item if and only if the objective function increases. Assign the maximum value of the objective function found to the variable INF. This variable stores the current lower bound of the objective function. Add the found branch to the list of branches. Set the associated upper bound SUP to infinity.
3. **If** there is no branch left on our list of branches, go to step 7.
   Else take the branch of our list of branches having the maximum objective function value. Go to step 4.
4. **If** the associated upper bound SUP is greater than the current lower bound INF, go to step 5.
   Else delete the branch from the list. Go to step 3.
5. **If** there is no accepted item left in the selected branch that does not already have a plunged or rejected subtree, delete the branch from the list. Go back to step 3.
   Else, following our ranking, choose the first accepted item that does not already have a plunged or rejected subtree. Calculate an upper bound SUP for the subtree defined by rejecting this item. Go to step 6.
6. **If** SUP $\leq$ INF, reject this subtree, go to step 5.
   Else plunge the subtree as described in 2 and add the found branch together with the value SUP to the list of branches. If the value of the objective function of this branch is greater than INF, update INF.
   Go to step 3.
7. The current value INF is the optimal solution value of problem (2.7).

Algorithm 2.3.2

2.4 Numerical results

The first part of this section contains the results of the algorithm for solving the continuous $SRKP$, namely the stochastic gradient method. In the second part of the section, the results for the B&B algorithm are presented. Both algorithms have been implemented in C-language. All tests were carried out on an Intel PC with $1GB$ RAM. We tested our methods on the same dataset as in [CB98] as well as a sample of randomly created instances for each of the chosen dimensions. The *Cohn-instance* is presented in Table 2.1. The last column states the value of the ratio r_i^2/σ_i needed for the ranking of the items. The penalty factor used is 5. For the random datasets, the weight means were generated from a normal distribution with mean 225 and standard deviation 25, the variances from a uniform distribution on the interval $[5, 50]$ and the rewards per weight unit were chosen to have equal probability to be 1, 2 or 3. As in the Cohn-instance, the penalty factor is 5. For each dimension we created 50 instances. Tables 2.2 and 2.3 show the average values

over these 50 instances.
We compared our algorithm with the method of Cohn and Barnhart presented in [CB98]. In their paper, they propose three different upper bounds to use within their B&B algorithm. However, they do not give any details of which upper bound to use at which moment. In order to compare our approach with that of Cohn and Barnhart, we therefore computed their upper bounds one after another in step 5 of our B&B algorithm. As soon as an obtained bound is sufficiently tight to prune the currently evaluated subtree, we leave out the computation of the remaining bounds. This procedure assures that the number of considered nodes is at most as large as when using their exact policy.

Object	**reward per weight unit** r_i	**mean of the weight** μ_i	**variance of the weight** σ_i^2	$\frac{r_i^2}{\sigma_i}$
1	2	212	47	0.583
2	2	203	21	0.873
3	3	246	42	1.389
4	2	223	21	0.873
5	2	230	15	1.033
6	1	233	10	0.316
7	2	235	11	1.206
8	2	222	33	0.696
9	1	210	36	0.167
10	2	299	42	0.617
11	2	256	25	0.800
12	3	250	19	2.065
13	1	194	24	0.204
14	3	207	22	1.919
15	1	182	14	0.267

Table 2.1: Values of the Cohn-instance

2.4.1 The continuous $SRKP$

An example for the convergence of the stochastic gradient method involving the *AbC* method is presented in Figure 2.4.1. As shown in the figure and confirmed by numerical tests, the best result found does not change very much (less than 1%) after iteration 500. Based on this observation, we used in the following a stopping criterion for the stochastic gradient algorithm of 500 iterations. In Table 2.2 we compare on the one hand the found optima of the continuous problems with the lowest upper bound proposed in [CB98]. On the other hand, we compare the CPU-time (in milliseconds) of the stochastic gradient algorithm with the time needed to compute all three upper bounds of Cohn and Barnhart.

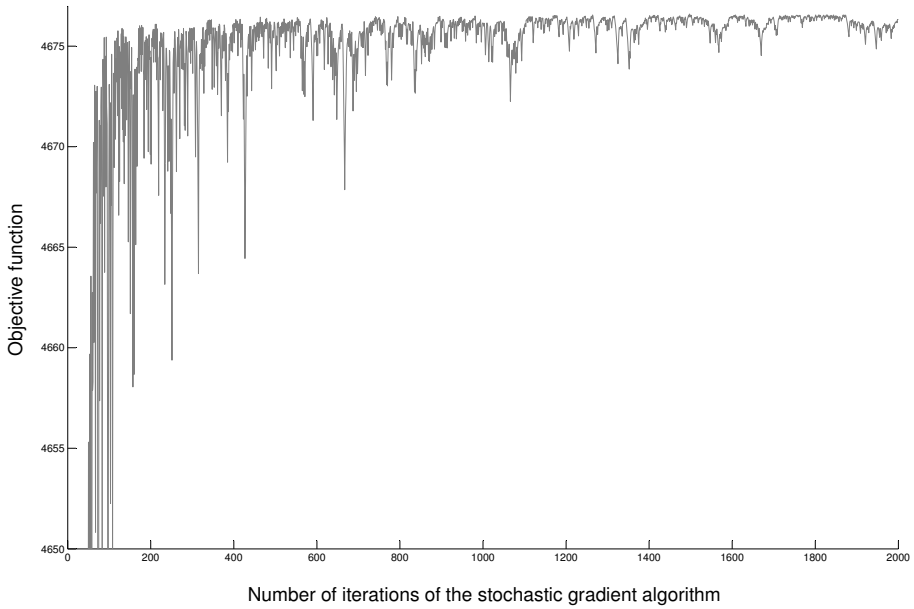

Figure 2.4.1: Convergence of the stochastic gradient algorithm solving the continuous $SRKP$

C./B. stands for Cohn/Barnhart, i.e. for the (unique) Cohn-instance of dimension 15.

We observe that especially for small dimensions it takes much less time to compute all three upper bounds proposed by Cohn and Barnhart than to solve the continuous relaxation by a stochastic gradient algorithm. But, while the CPU-time of the stochastic gradient algorithm increases proportional to the dimension, this is not the case for the upper bounds proposed by Cohn and Barnhart.

2.4.2 The combinatorial $SRKP$

The numerical results for the combinatorial problem are shown in Table 2.3. Notice that the CPU-time needed by the B&B algorithm (columns 6 and 11) is given in seconds. Columns 5 and 10 contain the number of considered nodes, i.e. the number of times an upper bound is calculated during the B&B algorithm. We allowed an average CPU-time of $1h$. * indicates that an average computing time of more than $1h$ would have been needed to solve all instances of this dimension.

We observe that when using the upper bounds of Cohn and Barnhart during the B&B algorithm much more nodes have to be considered. This can be explained by the less tighter upper bounds and, consequently, a smaller number of rejected subtrees. For small dimensions ($n = 15, 20, 30$) this is counterbalanced by the small CPU-times needed to calculate one upper bound. In the case of higher dimensional problems, the B&B algorithm involving a stochastic gradient algorithm becomes more competitive due to the tighter upper bounds and the resulting smaller number of considered nodes.

	Stochastic gradient & Approx. by convolution		Cohn/Barnhart	
n	**Optimum relaxation**	**CPU-time (msec)**	**Lowest upper bound**	**CPU-time (msec)**
C./B.	4676.208	4	4759.000	< 1
15	4934.583	4	5146.927	< 1
20	6690.744	6	6936.017	< 1
30	10279.908	9	10529.541	< 1
50	16954.343	12	17224.803	< 1
75	25519.688	16	25811.775	< 1
100	33846.095	22	34131.754	< 1
150	50607.008	31	50932.104	< 1
250	85098.136	52	85459.649	1
500	170110.459	104	170503.708	3
1000	340922.966	240	340822.740	5
5000	1703811.095	1110	1704935.949	107
20000	6813327.586	4940	6815663.089	1759

Table 2.2: Numerical results for the continuous $SRKP$

As mentioned, Table 2.3 only shows the results for the combinatorial problem in the case where the average CPU-time over all 50 instances is at most $1h$. In case of the B&B algorithm involving the stochastic gradient algorithm, this limit is respected when $n = 75$ but exceeded when $n = 100$. For $n = 100$, the CPU-time is smaller or equal to $2h$ in about 78% of the cases and only 6% of the instances needed more than $24h$ to terminate. For $n = 150$, 44% of the tests finished in at most $2h$ and 56% of the instances needed not more than $24h$.

2.5 Concluding remarks and future work

In this chapter we studied and solved a Simple Recourse Knapsack problem with random weights. We applied a B&B algorithm and solved continuous subproblems in order to provide upper bounds. The latter was done using a stochastic gradient method. Approximated gradients were computed by applying an Approximation by Convolution mollifier technique.
We want to remark at this point that the function j inside the expectation of the objective function of the Simple Recourse Knapsack problem studied is not differentiable everywhere, but subdifferentiable. Instead of a stochastic gradient algorithm we could thus have chosen a stochastic subgradient method. This will be done in the next section

	Stochastic gradient & Approximation by convolution				
n	**Upper Bound**	**CPU-time (msec) continuous**	**Optimum**	**considered nodes**	**CPU-time (sec) B-and-B**
C./B.	4676.208	4	4618	100	0.342
15	4934.583	4	4890	41	0.139
20	6690.744	6	6651	80	0.348
30	10279.908	9	10265	455	2.808
50	16954.343	12	16951	13173	131.171
75	25519.688	16	25514	63972	934.550
100	33846.095	22	*	*	*
	Cohn/Barnhart				
n	**Upper Bound**	**CPU-time (msec) continuous**	**Optimum**	**considered nodes**	**CPU-time (sec) B-and-B**
C./B.	4759.000	< 1	4618	144	0.000
15	5146.927	< 1	4890	65	0.002
20	6936.017	< 1	6651	280	0.003
30	10529.541	< 1	10265	2525	0.037
50	17224.803	< 1	16951	364960	779.325
75	25811.775	< 1	*	*	*
100	34131.754	< 1	*	*	*

* average CPU-time exceeds 1h

Table 2.3: Numerical results for the (combinatorial) $SRKP$

to solve the Chance-Constrained Knapsack problem. Moreover, as j is only nondifferentiable on a null set, this set of points could have been "neglected" as we are, in the end, interested in the expectation of the gradient. In other words, the probability to draw a sample of the random parameters for a given decision such that j is nondifferentiable at the obtained point is zero, anyway.
We compared the B&B algorithm involving the stochastic gradient method numerically with a method proposed by Cohn and Barnhard in ([CB98]). Our tests have shown that the upper bounds obtained by solving continuous relaxations are tighter than those proposed in [CB98]. Consequently, less nodes had to be considered during the B&B algorithm. This resulted for large size problems in smaller CPU-times. With a maximum average computing time of $1h$ we were thus able to solve problems of up to 75 items, while the method of Cohn and Barnhart only solved instances of up to 50 items.
In this chapter we assumed the item weights to be independently normally distributed.

For this special case Fortz et al. recently presented excellent numerical results by reformulating the problem deterministically and using a commercial nonlinear solver to solve the obtained problem ([FLLP08]). Contrary to their approach, our method seems, however, to be easily adaptable to other distributions. In fact, the assumption of a normal distribution is only once (directly) used when we establish the dominance relationships and rank the items. Other dominance relationships might be established for different distributions or one might use a standard B&B algorithm that makes use of the solutions of the continuous subproblems to decide on which item to branch. Indirectly, we use the property that in case of independently normally distributed weights the objective function of the Simple Recourse Knapsack problem can be easily computed which is a great advantage for both B&B and stochastic gradient algorithm. Studying the adaptation of our method to other distributions (or the black-box assumption) and different stochastic combinatorial problems will be an important step to show the applicability of our method on a larger scale.
In this chapter we studied an unconstrained simple recourse problem. However, one could imagine extensions of the model to other problems that require additional (deterministic) constraints. To solve these kinds of problems our method can be adapted by replacing the stochastic gradient method by a projected stochastic gradient method. Nonnegativity constraints can be handled by an active set strategy. In [KL10b] we chose such an approach to solve a Simple Recourse Shortest Path problem. Due to an open question concerning the (deterministic) active set method numerical results have not yet been achieved.
In the next section, we will extend the work presented in this section: Instead of a Simple Recourse Knapsack problem, we will solve a Chance-Constrained Knapsack problem. Although the basic idea of the solution method is the same, the case of a chance-constraint turns out to be somewhat harder. In addition to numerical tests we therefore give a theoretical analysis of the convergence of the stochastic gradient algorithm when applied to this kind of problems. As already mentioned, two more alternative approaches to the problem of nondifferentiability of the objective function (resp. constraint) will be presented. These could be applied to the Simple Recourse Knapsack problem, as well.

3 The Chance-Constrained Knapsack Problem

3.1 Introduction

3.1.1 Chance-Constrained Problems

Chance-constrained stochastic optimization problems are part of the class of *static stochastic problems*. This means, that one has to make its decision once for all and independently of the outcomes of the random parameters. In contrary, the two-stage models with full recourse that we will discuss in the next chapter are part of the *dynamic stochastic problem* type class, where at least one (partial) decision can be made after (part of) the random parameters have come to be known. The simple recourse problems studied in the previous chapter are at the boundary of both problem classes: Some authors see them as static stochastic problems as the second-stage penalty is not interpreted as an actual decision (see [Pré95]). Other authors emphasize the two-stage structure of simple recourse problems with a second-stage decision that consists in adapting the right hand side of the constraints optimally in order to make the first-stage decision feasible (see for example [KW94]).
A chance-constrained model can be considered wherever one's decision might lead to a violation of one or more of the constraints once the random parameters are known. If we are willing to take this risk in a small percentage of cases, we can replace the corresponding deterministic constraint $Ax \leq b$ by the following chance-constraint:

$$\mathbb{P}\{A(\chi)x \leq b(\chi)\} \geq p \tag{3.1}$$

where χ is once more a random vector and p a prescribed probability threshold. Similar to the case of simple recourse and two-stage problems, the matrix $A(\chi)$ is often called *technology matrix*.
The interpretation of constraint (3.1) is that we require our decision x to satisfy the constraint $A(\chi)x \leq b(\chi)$ for at least a fraction p of the possible outcomes of the random vector χ. If the matrix A contains more than one row, the constraint $A(\chi)x \leq b(\chi)$ consists of a set of linear constraints $a_j(\chi)x \leq b_j(\chi)$ and constraint (3.1) should be interpreted as the *joint chance-constraint*

$$\mathbb{P}\{a_1(\chi)x \leq b_1(\chi), \ldots, a_m(\chi)x \leq b_m(\chi)\} \geq p \tag{3.2}$$

where m is the number of rows of A. This constraint requires the m constraints $a_1(\chi)x \leq b_1(\chi), \ldots, a_m(\chi)x \leq b_m(\chi)$ to be "jointly" satisfied with at least probability p. Equivalently, (3.2) can be stated as follows:

$$\mathbb{P}\{\exists i \in \{1, \ldots, m\} : a_i(\chi)x > b_i(\chi)\} \leq 1 - p \tag{3.3}$$

i.e. we restrict the probability that one or more of the constraints $a_i(\chi)x \leq b_i(\chi)$ might be violated by our decision to $1 - p$ (see chapter 5 where this formulation is chosen with a corresponding taken risk denoted α).
In contrary, the *separate* or *individual chance-constraints*

$$\mathbb{P}\{a_j(\chi)x \leq b_j(\chi)\} \geq p_j \quad \forall\, j = 1, \ldots, m \tag{3.4}$$

restrict the probability of a violation for each constraint separately. Separate chance-constraints are clearly less restrictive, as we could theoretically obtain an optimal solution that, for any outcome of the random parameters, violates at least one constraint. More precisely, if $p_j \leq p$ for all $j = 1, \ldots, m$, we have that any solution x that satisfies constraint (3.2) also satisfies the set of constraints (3.4), while the reverse is only assured to be true if $\sum_{j=1}^{m}(1 - p_j) \leq 1 - p$.
Another difference between the joint chance-constraint (3.2) and the separate chance-constraints (3.4) concerns the question of how to evaluate the left hand side(s) of the constraint(s). In the case of separate chance-constraints this evaluation is in general easier: Let us define the random variables $X_j = a_j(\chi)x - b_j(\chi) = \sum_{i=1}^{n} a_j^i(\chi)x_i - b_j(\chi)$ $(j = 1, \ldots, m)$. Then, the right hand side $\mathbb{P}\{a_j(\chi)x \leq b_j(\chi)\}$ is equal to the function value of the CDF of X_j at the point 0.
In case of the joint chance-constraint (3.2) it is clear that the main problem lies in the possible (or even probable) dependencies of the random variables X_j. Only in case of independence between any two variables X_j and $X_{j'}$, i.e. if (a_j^T, b_j) and $(a_{j'}^T, b_{j'})$ do not depend on the same random vector χ, the left hand side of the joint chance-constraint might be evaluated with the help of the following equality:

$$\mathbb{P}\{a_1(\chi_1)x \leq b_1(\chi_1), \ldots, a_m(\chi_m)x \leq b_m(\chi_m)\} = \prod_{j_1}^{m} P\{a_j(\chi_j)x \leq b_j(\chi_j)\} \tag{3.5}$$

In the rest of the cases, one needs in general the joint probability distribution of the random vector $(X_1, \ldots, X_m)$ to evaluate the left hand side of the chance-constraint.
To the best of our knowledge, chance-constraints have first been mentioned in a paper by Charnes, Cooper and Symonds [CCS58]. The authors propose a model to optimize the production of heating oil. The demand of heating oil is clearly weather depending and thus highly random. The authors introduce two chance-constraints in their model, one concerning the sales demands that have to be met and one concerning storage limitations. In a subsequent paper [CC59], Charnes and Cooper first mention the expression *chance-constrained programming*. Both papers only treat the case of separate chance-constraints. Joint chance-constraints were first studied by Miller and Wagner in [MW65]. The authors

study the case of two jointly constrained linear constraints under the assumption that any random parameter of the first constraint is independent from any random parameter of the second one.
Main contributions to the understanding of chance-constrained optimization problems, especially concerning the case of joint constraints with dependent parameters, have been achieved by Prékopa in his papers [Pré70] and [Pré72]. In his works, Prékopa uses the expression *probability-constraint* instead of *chance-constraint*. One of the main results of the paper from 1972 is the following:

Theorem 3.1 (Prékopa [Pré72]). *Let $g_i(x, y)$ $(i = 1, \ldots, m)$ be concave functions on $\mathbb{R}^{n+K}$ (where x is an n-dimensional and y a K-dimensional vector). Let further χ be a K-dimensional random vector with logarithmic concave probability distribution. Then, the left hand side x-function of the joint chance-constraint*

$$\mathbb{P}\{g_i(x, \chi) \geq 0, i = 1, \ldots, r\} \geq p \tag{3.6}$$

is logarithmic concave in the entire space $\mathbb{R}^n$.

The following corollary is immediate, as by applying the negative logarithm to both sides of a logarithmic concave chance-constraint of type (3.1), we obtain a constraint of the form $G(x) \leq C$ with convex left hand side function G:

Corollary 3.2 (Prékopa [Pré72]). *Let $g_i(x, y)$ $(i = 1, \ldots, m)$ be concave functions on $\mathbb{R}^{n+K}$ (where x is an n-dimensional and y a K-dimensional vector). Let further χ be a K-dimensional random vector with logarithmic concave probability distribution. Then the joint chance-constraint*

$$\mathbb{P}\{g_i(x, \chi) \geq 0, i = 1, \ldots, r\} \geq p \tag{3.7}$$

defines a convex set.

The conditions for these two results have later been weakened by Tamm ([Tam76],[Tam77]) who showed that the requirement of concavity for the functions g_i can be replaced by quasi-concavity. This result is surely the most cited and most used result in chance-constrained programming as it defines a class of convex chance-constrained stochastic optimization problems. It can be generalized to the case of r-concave distributions implying r-concavity of the left hand side function of the chance-constraint (see e.g. [SDR09]) and thus r-convexity of the corresponding feasible set.
Concerning the initially studied chance-constraint (3.1), the above results directly apply to the following two special variants, provided that χ has a logarithmically (resp. r-) concave probability distribution:

$$\mathbb{P}\{Ax \leq \chi\} \geq p \quad \text{and} \quad \mathbb{P}\{\chi^T x \leq c\} \geq p \tag{3.8}$$

where A is a deterministic matrix and c a constant. More results on chance-constraints can be found in the book by Prékopa [Pré95] and the very recently published lectures by Shapiro et al. [SDR09].

Discrete probability distribution

In the case of only finitely many outcomes $\chi_1, \ldots, \chi_K$ of the random vector χ with corresponding strictly positive probabilities $p^1, \ldots, p^K$, it is often possible to replace the chance-constraint (3.1) by a set of linear constraints. To do this, we however require the components of the left hand side vector $A(\chi)x$ to be bounded from above for all feasible decisions x. This is naturally the case if the components of x are bounded, for example in case of a binary decision vector (see e.g. [Lop10]), or, more general, if x lies in a compact set (see [Rus02]). More precisely, if there exists an M such that for any feasible decision x and any outcome $\hat{\chi}$ of χ we have

$$A(\hat{\chi})x - b(\hat{\chi}) \leq M\mathbb{1}_n$$

constraint (3.1) can be replaced by the following set of deterministic constraints:

$$A(\chi^k)x \leq b(\chi^k) + M(1 - z^k)\mathbb{1}_m \quad \forall\, k = 1, \ldots, K \tag{3.9}$$

$$\sum_{k=1}^{K} p^k z^k \geq p \tag{3.10}$$

$$z_k \in \{0, 1\} \quad \forall\, k = 1, \ldots, K \tag{3.11}$$

Here, for any scenario k, we introduced an additional binary decision variable z_k. In case that scenario k is chosen to be neglected, z_k is set to 0. This "disables" the constraint $A(\chi)x \leq b(\chi)$ for the outcome χ^k of the random vector as in this scenario the constraint is satisfied for any feasible decision x (see constraints (3.9)). In case we require the constraint $A(\chi)x \leq b(\chi)$ to be active in scenario k, we set $z_k = 1$. Clearly, the probability that one of these "active" scenarios occurs must be at least p (see constraint (3.10)).
One could as well choose the reformulation in a way that the binary variable z_k is set to one if the corresponding scenario is rejected (see chapter 5).
In case of a discretely distributed random vector χ and a bounded left hand side of the linear constraint, the problem of how to evaluate the chance-constraint is solved. However, for K scenarios the chance-constraint is replaced by $K + 1$ deterministic constraints and, more important, we have to introduce K (additional) binary variables. Especially if the random variables are independent, K can be very large and the deterministic equivalent problem becomes intractable.
In the case of an unbounded left hand side or integer random variables, the case is somewhat more complicated. Most work in this direction concerns chance-constraints where only the right hand side $b(\chi) = \chi$ is random, while the technology matrix $A(\chi) = A$ is deterministic. In this case, one can define for $p \in (0, 1)$ the set of points $\mathcal{Z}_p \subset \mathbb{R}^m$ such that for all $z \in Z_p$ it is $\mathbb{P}\{z \leq \chi\} \geq p$ and there is no $y \geq z$ such that $\mathbb{P}\{y \leq \chi\} \geq p$ [1].
It is clear that these points are a subset of all the possible outcomes of χ. In case of a

[1] Note that due to our formulation of the chance-constraint and following the common definition of p-efficient points, the points in $\mathcal{Z}_p$ are not the p-efficient points of the distribution of χ, but the negatives of the p-efficient points of the distribution of $-\chi$.

finite number of scenarios, $\mathcal{Z}_p$ is thus finite. It is also nonempty, as there exists always a vector z such that $\mathbb{P}\{\chi \geq z\} \geq p$ (for $z = \chi_{min} := (\min_k \chi_1^k, \ldots, \min_k \chi_m^k)$ we even have $\mathbb{P}\{\chi \geq z\} = 1$). For the case of integer random variables, Dentcheva et al. showed the following:

Theorem 3.3 (Theorem 1 in [DPR00]). *Let $p \in (0,1)$. Then, in case of an integer random vector, the set $\mathcal{Z}_p$ is nonempty and finite.*

Based on these observations and using an enumeration of the finitely many points in $\mathcal{Z}_p$, the chance-constraint (3.1) with deterministic technology matrix and discretely distributed or integer right hand side vector can be replaced by

$$Ax \leq z \quad \text{for at least one vector} \quad z \in \mathcal{Z}_p \tag{3.12}$$

(see for example [Pré90]).
In some special cases, constraint (3.12) can be further replaced by a set of linear constraints and the introduction of an m-dimensional integer decision vector (see [DPR00]).

Normal probability distribution

Another particular and often made assumption concerning the random variables in a chance-constraint is to assume their probability distribution to be normal. First of all, the normal distribution is logarithmic concave, i.e. Theorem 3.1 and Corollary 3.2 can be applied in case the quasi-concavity of the inner functions g_i is assured. Moreover, in the case of one single chance-constraint $P\{a(\chi)x \leq \beta(\chi)\} \geq p$, we obtain the following equivalence in case that the components of the random vector $(a(\chi)^T, -\beta(\chi))$ are jointly normally distributed with corresponding $(n+1)$-dimensional vector of means μ and covariance matrix $C \in \mathbb{R}^{(n+1)\times(n+1)}$:

$$\mathbb{P}\{a(\chi)x - \beta(\chi) \leq 0\} \geq p \tag{3.13}$$

$$\Leftrightarrow \left(\mathbb{P}\{a(\chi)x - \beta(\chi)x_{n+1} \leq 0\} \geq p \quad \wedge \quad x_{n+1} = 1 \right) \tag{3.14}$$

$$\Leftrightarrow \left(F\left(\frac{-\sum_{i=1}^{n+1} x_i \mu_i}{(z^T C\, z)^{\frac{1}{2}}} \right) \geq p \quad \wedge \quad x_{n+1} = 1 \right) \tag{3.15}$$

$$\Leftrightarrow \left(F^{-1}(p)(z^T C\, z)^{\frac{1}{2}} + \sum_{i=1}^{n+1} x_i \mu_i \leq 0 \quad \wedge \quad x_{n+1} = 1 \right) \tag{3.16}$$

where z is defined such that $z^T = (x^T, x_{n+1})$.
If the means and the covariance matrix are known, the constraint function can thus be evaluated by using the common value tables for the standard normal distribution.
In the special case of independently normally distributed parameters, the above reformulation is clearly valid with diagonal covariance matrix C having the variances of the random

parameters on its diagonal. Under the assumption that $p \in [0.5, 1]$ (i.e. $F^{-1}(p) \geq 0$), chance-constraint (3.1) can thus be further rewritten as

$$\left(||(x^T, x_{n+1})C^{\frac{1}{2}}|| \leq -\frac{1}{F^{-1}(p)} \sum_{i=1}^{n+1} x_i \mu^i \quad \wedge \quad x_{n+1} = 1 \right)$$

The form of this constraint can be generalized as $\|Ax + b\| \leq c^T x + d$. Constraints of this type are known as Second Order Cone constraint due to the form of the described feasible region. It follows directly, that in case of independently normally distributed components of the vector $(a(\chi)^T, -b(\chi))$ and for $p \geq \frac{1}{2}$ constraint (3.13) defines a convex set. This result has recently been extended to the more general class of problems where the random vector $(a(\chi)^T, -b(\chi))$ has a logarithmic concave and symmetric distribution (see [LLS05]). Note that a vector of independent, logarithmic concave random variables has a logarithmic concave distribution itself (see for example [SDR09]).

3.1.2 Chance-Constrained Knapsack Problems ($CCKP$)

In this chapter we study a stochastic knapsack problem where, as in the previous chapter, the item weights are assumed to be unknown at the moment the decision of which items to put in the knapsack is made. Instead of assuming that a penalty has to be paid in case of an overweight, we require that the probability that the capacity constraint is respected equals at least a prescribed probability $p \geq \frac{1}{2}$. This probability is generally near to 1 and we assume that in the rest of the cases no penalty occurs.
In this chapter we once more make the assumption that the item weights follow independent normal distributions. Consequently, the knapsack chance-constraint

$$\mathbb{P}\{\sum_{i=1}^{n} \chi_i x_i \leq c)\} \geq p \tag{3.17}$$

defines a convex set (see section 3.1.1). Moreover, the random variable that represents the total weight of the chosen items is normally distributed, as well.
Chance-Constrained Knapsack problems have been studied before:
In [KRT97] the authors first consider a problem where the items are so called "on-off-items" with deterministic weights and rewards, i.e. they can be modeled as independent Bernoulli trials. The obtained results are then extended to the case where the items are always "on" but the weights are random and take either a value hi with a given probability, and a value $lo < hi$ otherwise.
In [GI99] the authors consider moreover the cases where the item weights are all Poisson or all exponentially distributed. As in the case of normally distributed random variables, the Poisson distribution has the advantage that a sum of Poisson random variables is a Poisson random variable as well. The left hand side of the knapsack chance-constrained can thus be easily evaluated. Note that the exponential distribution is another example for a logarithmic concave probability distribution.

In [KN08] the authors study the case where the items are assumed to be uniformly distributed over a certain interval. This is a common assumption in robust optimization and the authors propose to solve the obtained Chance-Constrained Knapsack problem via a sequence of robust knapsack problems. Their work has been extended in [Klo09] where the case of a joint chance-constraint containing a set of general linear constraints is treated with a similar approach.
In [GLLH10] the authors consider stochastic quadratic knapsack problems with chance-constraint(s). They first study the cases of a (single level) chance-constrained quadratic knapsack problem and show how, under the assumption of a finite probability distribution, the problem can be equivalently reformulated as a quadratic programming problem. Due to the $\mathcal{NP}$-hardness of the obtained problem, the authors propose linear as well as semidefinite relaxations. In the second part of the article, a two-stage extension is studied where the chance-constraint appears in the second-stage. The authors remark, that their model can be easily extended to the case where an additional chance-constraint appears in the first stage (see also [LLH09] and chapter 4).
In a very recent and to the best of our knowledge still unpublished paper ([GR09]) the Chance-Constrained Knapsack problem with independent normal distributions is studied under the aspect of approximability.
A stochastic knapsack problem related to the Chance-Constrained Knapsack problem has been studied in [MW97]: The authors assume the rewards to be random, while the item weights and the capacity are deterministic. The objective is to maximize the probability that the total gain exceeds a given threshold. The authors assume independently normally distributed rewards. Due to the strict monotonicity and continuity of the CDF of normal distributed variables, they obtain a deterministic equivalent problem with rational objective function.
More general combinatorial chance-constrained problems are considered in [Klo10]. Instead of one single knapsack chance-constraint, the author considers its problem to contain a set of separate chance-constraints. Moreover, the linear constraints are not explicitly required to be knapsack constraints, i.e. both the linear coefficients of the left hand side as well as the right hand side might take negative values. The author shows how to obtain a linear description of the feasible set in case of normally distributed coefficients as well as without any assumptions on the underlying probability distributions. However, the number of linear constraints needed to replace the chance-constraint is in general exponential.

3.1.3 Solving Chance-Constrained Problems

Most of the solution procedures proposed for problems involving a chance-constraint of the form (3.1) make use of the log- or quasi-concavity of the left hand side constraint function. They can thus not be used in general. Moreover, most algorithmic studies of chance-constrained problems assume that the technology matrix is deterministic.
One of the first such methods was proposed by Prékopa in [Pré70]. The author assumes that both the objective as well as the left hand side function of the chance-constraint

are quasi-concave having a continuous gradient. Keep in mind that the left hand side of the constraint is the CDF value of the random variable $A(\chi)x - b(\chi)$ at the point 0. It is thus continuously differentiable with respect to x whenever the underlying probability distributions are continuous. The author of [Pré70] allows the left hand side function of the constraint inside the probability measure to be more general than linear. However, he restricts its study to the case where only the right hand side is random and follows a continuous joint distribution. The author proposes a *method of feasible direction*, that iteratively solves linear programs using the gradients of objective and constraint function in order to determine a feasible corrective direction. Under the mentioned assumptions and an additional assumption linked with Slater's condition for convex programming, the method is shown to converge.
Another common method to solve chance-constrained programming problems with logarithmic concave left hand side function is the so called SUMT (Sequential Unconstrained Minimization Techniques) method (see [Pré72]). This method has been previously proposed by Fiacco and McCormick to solve more general nonlinear programming problems. The idea is to use a penalty function (in the case of a logarithmic concave constraint function this is the logarithm) in order to include the chance-constraint in the objective. The obtained unconstrained problem has the property that its objective function goes to infinity when the left hand side of constraint (3.1) approaches p from above (in case of a minimization problem). An optimal solution of the obtained unconstrained problem is thus always feasible for the initial chance-constrained problem. One then approaches the optimal solution of the initial problem by letting a factor multiplying the penalty function go to zero. Although there exist several functions that could serve as penalty function, the choice of the logarithmic function for logarithmic concave constraint functions is evident, as the obtained problem is convex (in case of a minimization) if the initial objective function was convex as well.
In [May79] Mayer proposes to use a reduced gradient method in order to solve chance-constrained problems. The idea of a reduced gradient method is to partition the decision variables in non-basic and basic variables. While the corrective direction computed for the non-basic variables aims to decrease the objective (in case of a minimization problem), the modification of the basic variables is made with respect to the feasibility of the obtained solution. To determine these directions a linear problem has to be solved at each iteration. In [May88], Mayer implemented his algorithm under the assumption of a joint normal probability distribution of the right hand side vector and a deterministic technology matrix. The computation of the left hand side constraint function and its gradient is done using a Monte-Carlo sampling method.
In case of discretely distributed random variables, the feasible region defined by chance-constraint (3.1) is clearly not convex. We have seen in subsection 3.1.1 that at least in case of a discretely distributed or integer valued random right hand side vector $b(\chi) = \chi$ and a fixed technology matrix $A(\chi) = A$ the set of feasible solutions is finite. First approaches to solve these particular problem cases included the idea of enumerating all these finitely many points (see [Pré90]). An algorithmic enumeration procedure is given in [PVB98]. In [DPR00] the authors show how to reformulate a chance-constrained pro-

gramming problem with random, integer valued right hand side and r-concave CDF as an integer programming problem. They propose an iterative cone generation method that solves a linear relaxation at each step. Lagrangian dual bounds serve to generate new feasible solutions. The idea to use a B&B algorithm to explore the set of feasible points of a chance-constrained problem with discretely distributed right hand side vector has been implemented in [BR02]. Once more linear relaxations are solved to provide upper bounds.
An algorithm for the case of a random right hand side and deterministic technology matrix that does not make use of any concavity or continuity assumptions on the underlying distribution was proposed in [CAAK06]. The only assumptions made are the nonemptiness and boundedness of the feasible region. The authors propose a branch-reduce-cut algorithm that exploits the monotonicity properties of the chance-constraint (3.1). Feasibility and optimality cuts based on duality properties reinforce the basic B&B algorithm. The authors show that in case of discretely distributed random parameters their algorithm stops after finitely many iterations with either an optimal solution or by concluding that the problem is infeasible. In case of continuous distributions, the algorithm may take infinitely many steps to converge. In this case it however generates a sequence that converges towards an optimal solution. The authors show that their approach is much more efficient than for example reformulating the problem as an integer programming problem and solving the reformulation using standard solvers.
Recently, a lot of work has been done concerning the approximation of chance-constrained problems in order to overcome the two main difficulties that are present: The difficulty of computing the left hand side of the chance-constraint and the nonconvexity of the feasible region for more general distributions. The aim of these approximations is to replace the initial problem by a simpler problem (in terms of computation) whose optimal solution almost surely respects the chance-constraint. The most straightforward such approach is the so called *scenario approximation* method that has been used in [CC04] and [NS04] to solve chance-constrained problems approximately. The idea is to replace the chance-constraint by a set of deterministic constraints that are nothing else than the inner constraint (in the linear case $A(\chi)x \leq b(\chi)$) evaluated at a set of random samples. The authors of [CC04] show that one needs a random sample that increases with $(1-p)^{-1}$ and $ln\frac{1}{\beta}$ to obtain a problem whose optimal solution satisfies the chance-constraint with at least probability β. In [NS04], it is proven that in the case of normal or uniform distributions the dependence on $(1-p)^{-1}$ is only logarithmic.
Apart from the scenario approximation approach, there have also been studies on analytic approximations of chance-constrained problems. The main advantage of these approximations is that their quality generally does not depend on the risk level $1-p$. It is however clear that they mostly require precise informations about the underlying probability distributions. Work in this direction has been published in [NS06] (for random variables with computable moment-generating function) and [Nem03] (for normally distributed or centered and bounded random variables).

3.1.4 Solving Chance-Constrained Knapsack Problems

Concerning the solution of Chance-Constrained Knapsack problems, some procedures have been proposed before. These are however mostly restricted to special probability distributions:
In [KN08] the authors assume the item weights to be uniformly distributed over a given interval. In order to obtain feasible solutions for the initial problem they propose to solve a set of robust optimization problems whose cardinality is smaller or equal to the number of items. Their algorithm runs in pseudo-polynomial time. For two special cases they even prove that the algorithm finds the optimum of the initial Chance-Constrained Knapsack problem. In both cases some similarity between the random variables is assumed.
For the case of normally (but not necessarily independently) distributed item weights, Klopfenstein proposes in a very recent publication ([Klo10]) a deterministic linear combinatorial problem that is equivalent to the Chance-Constrained Knapsack problem. Contrary to the case of more general distributions studied in this same paper, only a polynomial number of linear constraints is needed to describe the feasible set. The approach of Klopfenstein is surely the currently most promising method to solve Chance-Constrained Knapsack problems with normally distributed weights.
Recently, Goyal and Ravi presented in [GR09] a $PTAS$ for the case of independently normally distributed items. They propose a parametric linear programming reformulation that is equivalent to a two-dimensional knapsack problem.
Approximation algorithms for Bernoulli, Poisson and exponential distributions have been given in [KRT97] and [GI99]. For the last two distributions, the authors of [GI99] even present a $PTAS$.
In this chapter we once more assume the item weights to be normally distributed. This is mainly due to the possibility to compare our method with previously proposed methods. In the conclusion we will, among others, discuss the extension of our approach to more general distributions.
We propose to solve the Chance-Constrained Knapsack problem with an approach similar to that used in the previous chapter to solve the Simple Recourse Knapsack problem. More precisely, we choose a B&B framework and solve continuous relaxations using a stochastic subgradient algorithm in order to provide upper bounds.
The applied stochastic subgradient algorithm is a so called stochastic Arrow-Hurwicz algorithm that solves the Lagrangian dual problem. More precisely, we reformulate the chance-constraint as an equivalent expectation-constraint, so that the Lagrangian dual problem becomes an unconstrained (apart from the $0-1$ bounds) stochastic optimization problem with expected value objective function. When applying a stochastic subgradient algorithm to this problem, we encounter the difficulty that theoretically a subgradient of the indicator function over the real interval is used to approximate the gradient of the constraint function. However, the indicator function is differentiable at nearly all points with a gradient whose components are all zero. Two solutions to this problem are presented: On the one hand, we propose to approximate the gradient using finite differences. On the other hand, we show how the left hand side function of the expectation-constraint can

be reformulated as an expected value function whose inner function is subdifferentiable. This reformulation is obtained via integration by parts.
The applied B&B algorithm is mainly the same as in the previous chapter. However, the plunging step and the ranking of the items have to be adapted to the Chance-Constrained Knapsack problem.
Continuous Chance-Constrained Knapsack problems have been previously solved using an $SOCP$ approach. At the end of the chapter, we will thus, on the one hand, directly compare the stochastic subgradient algorithm with the $SOCP$ method (when applied to solve the continuous Chance-Constrained Knapsack problem). On the other hand, we present the performances of both algorithms when used as a subroutine of the B&B framework.

3.2 Mathematical formulation

Like in the previous chapter, we are given a knapsack with deterministic weight capacity c and a set of n items whose weights are unknown at the moment the choice of which items to put in the knapsack has to be made. Once again we make the assumption that the item weights are representable by independently, normally distributed random variables χ_i with mean μ_i and standard deviation σ_i (for $i = 1, \ldots, n$). As for the Simple Recourse Knapsack problem studied in chapter 2.7, the rewards are assumed to depend linearly on the item weights, i.e. we only have knowledge of the reward per weight unit r_i for each item $i \in \{1, \ldots, n\}$. Once more all the methods applied and studies made in this chapter are still valid if the item rewards were independent of the item weights, either deterministic or random with known mean. Contrary to the previous chapter we restrict the percentage of cases where the chosen items lead to an overload by introducing a chance-constraint. In other words, it is assumed that in a small percentage of cases an overload is acceptable and does not lead to any penalty. The resulting Chance-Constrained Knapsack problem can be stated as follows:

Chance-Constrained Knapsack problem ($CCKP$)

$$\max_{x \in \{0,1\}^n} \mathbb{E}\left[\sum_{i=1}^{n} r_i \chi_i x_i\right] \tag{3.18}$$

$$\text{s.t. } \mathbb{P}\{\sum_{i=1}^{n} \chi_i x_i \leq c)\} \geq p. \tag{3.19}$$

where $p \in [0.5, 1]$ is the prescribed probability. As in chapter 2.7, we will denote the objective function of problem (3.18) by J, while the function inside the expectation of J is named j. To simplify formulations, we will in the following sometimes use the definition of the function $g(x, \chi) := \sum_{i=1}^{n} \chi_i x_i$.

3.2.1 Properties of the $CCKP$

Property 1: Note that the choice of p (resp. $1-p$) restricts the percentage of cases where the capacity constraint is violated by the chosen items, and this independently of the amount of overweight. This is the main difference between the Chance-Constrained and the Simple Recourse Knapsack problem, as in the case of the latter the choice of the penalty d has an influence on the expected overweight. More precisely, the larger d is chosen, the smaller the expected overweight will be. However, the probability of an overweight is not restricted.

Property 2: The chance-constraint (3.19) can be equivalently reformulated as the following expectation-constraint:

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}[c - \sum_{i=1}^{n} \chi_i x_i]\right] \geq p \tag{3.20}$$

In the following we will mainly work with this equivalent formulation of the chance-constraint and call the obtained stochastic optimization problem Expectation-Constrained Knapsack problem ($ECKP$). In the same manner as we defined the functions J and j of the objective of the $CCKP$ (resp. $ECKP$), we define the constraint functions $\Theta(x) := \mathbb{E}\,[\mathbb{1}_{\mathbb{R}^+}[c - \sum_{i=1}^{n} \chi_i x_i]]$ and $\theta(x, \chi) := \mathbb{1}_{\mathbb{R}^+}[c - \sum_{i=1}^{n} \chi_i x_i]$. Note that, for any $x \in \mathbb{R}^n$, $\Theta(x)$ equals the function value of the CDF of the normally distributed variable $X := \sum_{i=1}^{n} \chi_i x_i$ when evaluated at the point c.

Property 3: Without loss of generality we can assume that $\mathbb{P}\{\chi_i \leq c\} \geq p$ for all $i \in \{1, \ldots, n\}$: any item that does not satisfy this constraint could be excluded from the beginning. It follows, that any optimal solution x^* of the $CCKP$ (3.18) has at least one nonzero component. Moreover, we have

$$\begin{aligned}
\mathbb{P}\{\max_{i\in\{1\ldots,n\}} \chi_i \leq c\} \geq p &\Rightarrow \mathbb{P}\{n \cdot \frac{\max_{i\in\{1\ldots,n\}} \chi_i}{n} \leq c\} \geq p \\
&\Rightarrow \mathbb{P}\{\sum_{j=1}^{n} \frac{\max_{i\in\{1\ldots,n\}} \chi_i}{n} \leq c\} \geq p \\
&\Rightarrow \mathbb{P}\{\frac{1}{n}\sum_{j=1}^{n} \chi_j \leq c\} \geq p
\end{aligned}$$

i.e. any optimal solution of the continuous relaxation of problem (3.18) contains at least one component x_κ with $x_\kappa \geq 1/n$. Instead of replacing $\{0,1\}^n$ by $[0,1]^n$ when relaxing the $CCKP$, we are thus allowed to replace it by $\{x \in [0,1]^n \,|\, ||x||_\infty \geq 1/n\} =: X_{cont}$. Accordingly, we define the following feasible set for the relaxed $CCKP$:

$$X_{cont}^{feas} := \{x \in X_{cont} : \Theta(x) \geq p\}$$

Property 4: Remark on the choice of normally distributed weights: The first reason why we assume the weights to be independently normally distributed in this chapter is similar to one of the reasons given in the last chapter, namely that under this assumption the left hand side of the chance-constraint is easily computable (see Property 2). This is once more used during the solution procedure, this time in order to identify feasible solutions. In addition, we use some of the properties of the normal distribution, for example that the linear combination of independently normally distributed items is still normally distributed. The probably most important property that we gain by assuming the item weights to be independently normally distributed is that constraint (3.19) defines a convex set (see the introduction). Last but not least, the relaxed Chance-Constrained Knapsack problem with independently normally distributed weights has been solved before using an $SOCP$ method (see section 3.4.2). We make advantage of this to compare our solution approach with the previously proposed one.

3.3 Problem Solving Method

The basic idea to solve the $CCKP$ is identical to what we proposed in chapter 2 to solve the $SRKP$: Due to its combinatorial nature, we solve the $CCKP$ once more using a B&B framework. To provide upper bounds, we solve continuous relaxations of the $ECKP$ using a stochastic subgradient algorithm. More precisely, we apply a so called Stochastic Arrow-Hurwicz algorithm (hereafter called SAH algorithm) to solve the Lagrangian dual of the $ECKP$. Consequently, we encounter the problem that a subgradient of the indicator function $\mathbb{1}_{\mathbb{R}^+}(\cdot)$ should theoretically be used in order to stochastically approximate the gradient of the constraint function Θ. This problem could once more be solved by applying the Approximation by Convolution method presented in chapter 2. Instead, we present two different approaches: The first one is a nonbiased estimator of the subgradient based on integration by parts (hereafter called *IP-method*). Like the Approximation by Convolution method, it has been proposed in the thesis of Andrieu [And04] to solve continuous stochastic optimization problems. The second approach is a Finite Differences estimator (*FD-method*) presented in [ACVA07]. Like the Approximation by Convolution method, the FD-method provides a biased estimation of the gradient.

3.3.1 The Stochastic Arrow-Hurwicz (SAH) Algorithm

To solve the relaxed version of the $CCKP$ we use a stochastic subgradient algorithm. To apply a subgradient method is promising as the objective function is concave and, in addition, constraint (3.19) defines a convex feasible set due to the assumption that the weights are independently normally distributed. For more information on the theory of

subdifferentials and subgradient algorithms we refer the reader to the books by Hiriart-Urruty and Lemaréchal [HUL93a] and [HUL93b].
We propose to solve the $CCKP$ with a Stochastic Arrow-Hurwicz algorithm (see Algorithm 3.3.1). More precisely, we apply this algorithm to solve the equivalent $ECKP$. The algorithm introduces a Lagrange multiplier λ to handle the expectation-constraint (see also [CC95]), i.e. the idea is to solve the following Lagrangian dual of the $ECKP$:

Lagrangian dual of the $ECKP$

$$\min_{\lambda \geq 0} \max_{x \in X_{cont}} \mathbb{E}[\sum_{i=1}^{n} r_i \chi_i x_i] - \lambda \cdot \left(p - \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}[c - \sum_{i=1}^{n} \chi_i x_i] \right] \right) \tag{3.21}$$

The SAH algorithm is a stochastic subgradient algorithm, i.e. it uses subgradients

Stochastic Arrow-Hurwicz Algorithm

1. Choose $x^0 \in X_{cont}^{feas}$ and $\lambda^0 \in [0, \infty)$ as well as two σ-sequences $(\epsilon^k)_{k \in \mathbb{N}^*}$ and $(\rho^k)_{k \in \mathbb{N}^*}$. Set $k = 1$.
2. Given x^{k-1} and λ^{k-1}, draw χ_k following its normal distribution, compute $r^k \in \partial_x j(x^{k-1}, \chi^k)$, $\tau^k = \nabla_x \theta(x^{k-1}, \chi^k)$ and update x and λ as follows:

$$x^k = x^{k-1} + \epsilon^k (r^k + \lambda^{k-1}(\tau^k))$$
$$\lambda^k = \lambda^{k-1} + \rho^k (p - \theta(x^k, \chi^k))$$

3. For all $h = 1, \ldots, n$: If $x_h^k > 1$ set $x_h^k = 1$ and if $x_h^k < 0$ set $x_h^k = 0$.
4. If $x_h^k < 1/n$ for all $h = 1, \ldots, n$, set $x_h^k = 1/n$ for one $h \in \{1, \ldots, n\}$.
5. If $\lambda^k < 0$ set $\lambda^k = 0$.
6. If $k = k_{max}$: STOP. Else: Set $k = k + 1$. Go to step 2.

Algorithm 3.3.1

of the functions inside the expectations, instead of subgradients of the expected value functions. In fact, if we denote the Lagrangian dual function of the $ECKP$ by $\mathcal{L}(x, \lambda) = \mathbb{E}[l(x, \lambda, \chi)]$, it follows that $(r^k + \lambda^{k-1}(\tau^k)) \in \partial_x l(x^{k-1}, \lambda^{k-1}, \chi^k)$ (where $\partial_x l(x, \chi)$ denotes the subdifferential of l with respect to x at the point (x, χ)) while $(p - \theta(x^k, \chi^k)) = \frac{\partial}{\partial \lambda} l(x^k, \lambda^{k-1}, \chi^k)$ (cf. Algorithm 3.3.1). Note again that in the deterministic form of the Arrow-Hurwicz algorithm we would have to use the subgradients of the Lagrangian itself, i.e. the subgradient of an expected value function. By drawing independent samples of the random variables at each iteration of the algorithm, the expectations of the subgradients are approximated.
At each iteration of algorithm 3.3.1 a subgradient of θ with respect to x at the point (x^{k-1}, χ^k) is used to approximate the gradient of Θ. However, in the case of the $ECKP$ as stated in section 3.2, θ is mainly the indicator function over the real interval. This entails two problems: First of all θ has points of discontinuity. The second, more important

problem is that at the remaining points, where θ is differentiable, all components of its gradient are zero. It is thus not clear how to approximate the gradient of Θ by sampling when using the function θ (see also [ACVA07]). In the following two subsections, we will present two ways to handle this inconvenience: either by approximation using finite differences or by reformulation of the constraint function Θ using integration by parts.
A convergence analysis of the *SAH* algorithm from both a theoretical and a practical point of view is given in section 3.4.

The FD-Method

In this approach, the h^{th} component of the gradient of θ is approximated by the corresponding difference quotient

$$\frac{\theta(x+\delta\nu^h,\chi)-\theta(x-\delta\nu^h,\chi)}{2\delta}$$

where $\delta > 0$ and $\nu^h \in \{0,1\}^n$ such that $\nu^h_h = 1$ and $\nu^h_i = 0$ for $i \neq h$. This leads to the following approximation of the h^{th} component of the gradient of θ:

$$(\nabla_x(\theta_\delta)(x,\chi))_h = \frac{\mathbb{1}_{\mathbb{R}^+}(c-g(x+\delta\nu^h,\chi))-\mathbb{1}_{\mathbb{R}^+}(c-g(x-\delta\nu^h,\chi))}{2\delta}$$

The IP-Method

The idea of the IP-method is the following: We use integration by parts to reformulate $\mathbb{E}[\theta(x,\chi)]$ which gives us an expected value function $\mathbb{E}[\widetilde{\theta}(x,\chi)]$ s.t. $\mathbb{E}[\widetilde{\theta}(x,\chi)] = \mathbb{E}[\theta(x,\chi)] = \Theta(x)$ for all $x \in X^{feas}_{cont}$. $\widetilde{\theta}$ is subdifferentiable and the idea is to use a subgradient of $\widetilde{\theta}$ in the *SAH* algorithm. This idea has been presented in the thesis of Andrieu in order to solve expectation-constrained optimization problems (see Theorem 5.5 in [And04]). We state and prove their theorem for the case of the *ECKP* with normally distributed weights. The variables and functions used in this proposition are defined as in section 3.2:

Proposition 3.4. *Let $\mathbb{Y}_{\mathbb{R}^+}(\cdot)$ be a primitive of $\mathbb{1}_{\mathbb{R}^+}(\cdot)$ and let χ be a random vector whose components χ_i are independently normally distributed with mean μ_i and standard deviation σ_i. Let $x \in X_{cont}$ and let $\kappa = \kappa(x) \in \{1,\dots,n\}$ be defined such that $x_\kappa = ||x||_\infty \geq 1/n$. Then, using integration by parts, we get*

$$\Theta(x) = \mathbb{E}\left[-\mathbb{Y}_{\mathbb{R}^+}(c-g(x,\chi))\frac{(\chi_\kappa-\mu_\kappa)}{x_\kappa\sigma^2_\kappa}\right]$$

Proof. Let φ denote the density function of the random vector $\chi = (\chi_1,\dots,\chi_n)$ and define

$$u'_{\chi_\kappa}(x,\chi) := -\mathbb{1}_{\mathbb{R}^+}(c-g(x,\chi))x_\kappa \qquad \text{and}$$
$$v(x,\chi) := -\frac{\varphi(\chi)}{x_\kappa}$$

It follows

$$\Theta(x) = \int_{-\infty}^{\infty} \mathbb{1}_{\mathbb{R}^+}(c - g(x,\chi))\varphi(\chi)\,\mathrm{d}\chi = \int_{-\infty}^{\infty} u'_{\chi_\kappa}(x,\chi)v(x,\chi)\,\mathrm{d}\chi$$

Integration by parts over χ_κ leads to

$$\begin{aligned}\Theta(x) &= \big[u(x,\chi)v(x,\chi)\big]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} u(x,\chi)v'_{\chi_\kappa}(x,\chi)\,\mathrm{d}\chi \\ &= -\int_{-\infty}^{\infty} u(x,\chi)v'_{\chi_\kappa}(x,\chi)\,\mathrm{d}\chi = -\int_{-\infty}^{\infty} \mathbb{Y}_{\mathbb{R}^+}(c - g(x,\chi))v'_{\chi_\kappa}(x,\chi)\,\mathrm{d}\chi\end{aligned}$$

In our case the random variables are independently, normally distributed. With φ^i being the density function of χ_i we get

$$\begin{aligned}\varphi'_{\chi_\kappa}(\chi) &= \prod_{i\neq\kappa}\varphi^i(\chi_i)\cdot(\varphi^\kappa)'_{\chi_\kappa}(\chi_\kappa) = \prod_{i\neq\kappa}\varphi^i(\chi_i)\cdot\left(-\frac{(\chi_\kappa-\mu_\kappa)}{\sigma_\kappa^2}\varphi^\kappa(\chi_\kappa)\right) \\ &= -\frac{(\chi_\kappa-\mu_\kappa)}{\sigma_\kappa^2}\varphi(\chi)\end{aligned}$$

It follows

$$v'_{\chi_\kappa}(x,\chi) = \frac{\partial}{\partial\chi_\kappa}\left(-\frac{\varphi(\chi)}{x_\kappa}\right) = \frac{(\chi_\kappa-\mu_\kappa)}{x_\kappa\sigma_\kappa^2}\varphi(\chi)$$

and therefore

$$\begin{aligned}\Theta(x) &= -\int_{-\infty}^{\infty} \mathbb{Y}_{\mathbb{R}^+}(c - g(x,\chi))\frac{(\chi_\kappa-\mu_\kappa)}{x_\kappa\sigma_\kappa^2}\varphi(\chi)\,\mathrm{d}\chi \\ &= \mathbb{E}[-\mathbb{Y}_{\mathbb{R}^+}(c - g(x,\chi))\frac{(\chi_\kappa-\mu_\kappa)}{x_\kappa\sigma_\kappa^2}]\end{aligned}$$

□

Based on this proposition we can replace the left hand side function Θ of the expectation-constraint (3.20) by the function

$$\mathbb{E}\left[-\mathbb{Y}_{\mathbb{R}^+}(c - g(x,\chi))\frac{(\chi_\kappa-\mu_\kappa)}{x_\kappa\sigma_\kappa^2}\right] =: \mathbb{E}[\widetilde{\theta}(x,\chi)] \tag{3.22}$$

$\widetilde{\theta}$ is subdifferentiable with respect to x: First of all note that we can choose $\mathbb{Y}_{\mathbb{R}^+}(x) = \mathbb{1}_{\mathbb{R}^+}(x) \cdot x = [x]^+$. For a given realization $\hat{\chi}$ of χ $\widetilde{\theta}$ is thus differentiable at all points x with $c \neq g(x, \hat{\chi})$ with gradient

$$\nabla_x \widetilde{\theta}(x, \chi) = \mathbb{1}_{\mathbb{R}^+}(c - g(x, \chi)) \frac{(\chi_\kappa - \mu_\kappa)}{x_\kappa \sigma_\kappa^2} \chi + [c - g(x, \chi)]^+ \frac{(\chi_\kappa - \mu_\kappa)}{x_\kappa^2 \sigma_\kappa^2} \nu^\kappa$$

where again $\nu^\kappa \in \{0, 1\}^n$ is defined such that $\nu_\kappa^\kappa = 1$ and $\nu_i^\kappa = 0$ for $i \neq \kappa$. For $c = g(x, \hat{\chi})$ a possible choice for the subgradient in the SAH is $\mathbb{O}_n$.

3.3.2 The Branch-and-Bound Framework

The B&B algorithm chosen to solve the (combinatorial) $CCKP$ is basically the same as Algorithm 2.3.2 presented in chapter 2. In the case of the $CCKP$, it is however more complicated to introduce useful dominance relationships than in the case of the $SRKP$. This is due to the fact that modifying $\hat{\sigma} := \sqrt{\sum_{i=1}^n \sigma_i^2 x_i^2}$ cannot be interpreted as easily. The only very special case where one can say that item i dominates item k is the following:

1. $\mu_i = \mu_k$, $\sigma_i = \sigma_k$, $r_i \geq r_k$

Most of the time, the items are thus simply ranked by their value of r_i. This ranking gave good results in the numerical tests (compared e.g. with the ranking used for the $SRKP$ or a randomized ranking) but can surely be improved.
Concerning the B&B algorithm itself, we have to modify step 2 of Algorithm 2.3.2 in order to respect the chance-constraint: Instead of testing if the next item increases the objective function (which is the case for each item at every time), we check whether the chance-constraint is still satisfied after adding this item. More precisely, we calculate $F(\frac{c - \sum_{i=1}^n \mu_i x_i}{\sqrt{\sum_{i=1}^n \sigma_i^2 x_i^2}})$, i.e. the CDF of the random variable $X = \sum_{i=1}^n \chi_i x_i$ at the point c. Then, depending on whether the obtained value is greater or smaller than the prescribed probability p, we accept or reject the item.

3.4 Convergence of the SAH Algorithm

3.4.1 Theoretical versus practical convergence

Culioli and Cohen proved in [CC95] a theorem that guarantees the weak convergence of the SAH algorithm towards a saddle point (x^*, λ^*) of the associated Lagrangian:

Theorem 3.5 (Culioli, Cohen (1995))**.** *Suppose the following assumptions to be satisfied:*

H1 *$\theta(\cdot, \chi)$ is subdifferentiable with subgradient uniformly bounded with respect to χ.*

H2 *The associated Lagrangian admits a saddle point (x^*, λ^*).*

H3 *J is strictly concave.*

H4 $\forall \chi \in \mathbb{R}^n$, $\theta(\cdot, \chi)$ *is locally Lipschitz continuous.*

H5 *There exist* $c_1, c_2 > 0$ *such that*
$\forall\, \chi \in \mathbb{R}^n$, $\forall\, x, y \in X_{cont}$ $||\theta(x,\chi) - \theta(y,\chi)|| \leq c_1||x-y|| + c_2$.

H6 $\Theta(x)$ *is Lipschitz continuous and concave.*

H7 *There exist* $\alpha, \beta > 0$ *such that*
$\forall\, \chi \in \mathbb{R}^n$, $\forall\, x \in X_{cont}$ $||\nabla_x j(x,\chi)|| \leq \alpha||x - x^*|| + \beta$.

H8 ϵ^k/ρ^k *is monotone in the large sens.*

H9 *There exist* $\gamma, \delta > 0$ *such that*
$\forall\, x \in X_{cont}$ $\mathbb{E}[\theta(x,\chi) - \Theta(x)]^2 < \gamma||x - x^*||^2 + \delta$.

Then, the sequence (x^k, λ^k) *is bounded and* x^k *converges weakly towards an optimal solution of the* $CCKP$ (3.18).

Unfortunately, in our case not all hypotheses are satisfied. We will however see in the next section that, after solving some technical problems, the SAH algorithm converges on all tested instances towards a point $\widetilde{x} \in X_{cont}$ such that $J(\widetilde{x}) \approx J(x^*)$. We thus think it worth discussing the above hypotheses one by one in the case of the $CCKP$ (resp. $ECKP$), with an emphasis on the question of the practical solution of the problem. A concluding remark at the end of this subsection summarizes the discussion.

H1: In case of the $ECKP$, $\theta(x,\chi) = \mathbb{1}_{\mathbb{R}^+}(c - g(x,\chi))$ is not subdifferentiable in the common sense (θ is neither convex nor continuous). But, as presented in subsection 3.3.1, it is possible to either approximate the gradient (FD-method) or to replace $\theta(x,\chi)$ by a subdifferentiable function $\widetilde{\theta}(x,\chi)$ such that $\mathbb{E}[\widetilde{\theta}(x,\chi)] = \mathbb{E}[\theta(x,\chi)]$ (IP-method).
The FD-method gives us an approximated gradient that is either $1/(2 \cdot \delta)$ (where $\delta > 0$ is a constant) or 0. It is thus trivially bounded uniformly with respect to χ.
Using the IP-method a subgradient of the obtained function $\tilde{\theta}$ at point (x,χ) is given by the expression:

$$\mathbb{1}_{\mathbb{R}^+}(c - g(x,\chi))\frac{(\chi_\kappa - \mu_\kappa)}{x_\kappa \sigma_\kappa^2}\chi + [c - g(x,\chi)]^+\frac{(\chi_\kappa - \mu_\kappa)}{x_\kappa^2 \sigma_\kappa^2}\nu^\kappa$$

Whenever $\mathbb{1}_{\mathbb{R}^+}(c - g(x,\chi)) = 1$, one observes that this subgradient behaves as a quadratic expression of χ. It is therefore not possible to bound it uniformly with respect to χ.
However, in most real life applications, the ratio variance/mean of the unknown parameters is typically small. It follows, that the corresponding density functions tend to zero very fast which makes the occurrence of exceptionally small or high valued realizations improbable. Although in a theoretical analysis these cases cannot be neglected and therefore no uniform upper or lower bound stated, they do not play any role from a numerical point of view.

H2: We conjecture that hypotheses H2 is valid. Nonetheless, we remark that H2 is connected to H3, i.e. the question of stability of the Lagrangian, which, in our case, cannot be guaranteed. Yet keep in mind that we are not looking for an optimal solution vector x, but for the optimal value of the objective function.

H3: Unfortunately, in our case J is not strictly concave, as

$$J(x) = \mathbb{E}\left[\sum_{i=1}^{n} r_i \chi_i x_i\right] = \sum_{i=1}^{n} r_i \mu_i x_i = (r_1\mu_1, \ldots, r_n\mu_n)x$$

i.e. J is a linear map. An idea to overcome this problem is to use an Augmented Lagrangian. The problem is, however, that the use of a stochastic subgradient algorithm is in general not adapted to directly solve the Augmented Lagrangian problem (see e.g. [Car10b] and the next subsection on the convexification of the constraint function). Even if we cannot for instance prove it, we however think that the strict concavity of the objective function J is insignificant as long as the constraint function and thus the Lagrangian dual are strictly concave.

H4: Clearly, for a fix $\chi \in \mathbb{R}^n$, $\theta(\cdot, \chi)$ is not locally Lipschitz continuous as it is not even continuous at all those points x with $c - g(x, \chi) = 0$. However, in the case of the IP-method, θ is replaced by a function $\widetilde{\theta}$ which, in turn, is locally Lipschitz continuous at all points $x \in X_{cont}^{feas}$.

H5: Choose $c_2 = 1$. It follows $\forall\, \chi \in \mathbb{R}^n$, $\forall\, x, y \in X_{cont}$ $||\theta(x, \chi) - \theta(y, \chi)|| \leq c_2$.

H6: As we assume the item weights to be normally distributed, $\Theta(x) = P\{g(x, \chi) \leq c\}$ is Lipschitz continuous, as

$$\Theta(x) = F\left(\frac{c - \sum_{i=1}^{n} \mu_i x_i}{\sqrt{\sum_{i=1}^{n} \sigma_i^2 x_i^2}}\right)$$

It is easy to see that Θ is continuous on X_{cont} . In addition, we have $0 \leq \Theta(x) \leq 1$ for all $x \in \mathbb{R}^n$ and as X_{cont} is a compact set, it follows that Θ is Lipschitz continuous on X_{cont}. The question of concavity of Θ has already been discussed in the introduction. In fact, Θ is not concave but logarithmic concave. In figure 3.4.1 we show the plot of the constraint function for a "realistic" two-dimensional example. Here "realistic" is used in the sens that the variances are relatively small compared to the mean and that the capacity is large enough to allow for at least one item to be packed. As parameters we have chosen the weight means 212 and 242 and variances 47 and 20, respectively, and a knapsack capacity of 270.
The figure seems to indicate that there exists a closed subset of the unit square such that Θ is concave on this subset. To verify this first impression, we compute the eigenvalues of the Hessian matrix $H(\widetilde{x})$ of the function Θ at a point $\tilde{x}$ infinitesimal close to the bottom of the set X_{cont} (in the sense where $x_i = 0$ for all $i = 1, \ldots, n$) and we find them all nonpositive.

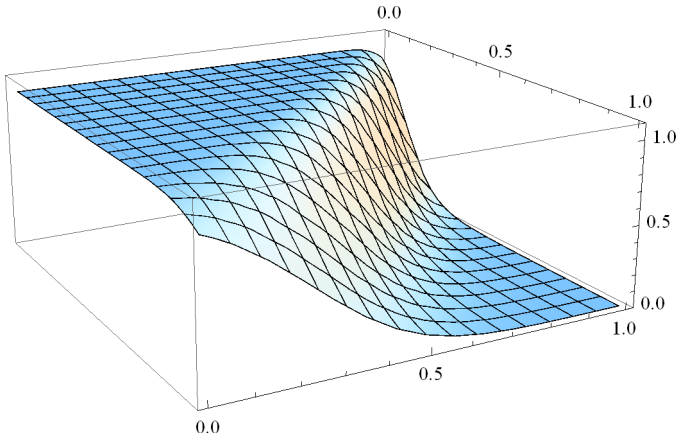

Figure 3.4.1: Constraint function Θ of the $CCKP$ in 2-dimensional case with realistic parameters

Moreover, when computing the components of the Hessian matrix in the limit $(0, 0)$ one obtains, as expected, zero. According to the fact that the determinant of the Hessian matrix $\det H(x)$ is a continuous function of x, it follows that there exists a topologically connected and closed component $C \subseteq X_{cont}$ with $\tilde{x} \in C$ such that $\det H(x) \leq 0$ for all $x \in C$. For $p_C := \min\{\Theta(x) | x \in C\}$, it thus follows that Θ is concave on the feasible set of the corresponding relaxed $CCKP$ with prescribed probability p_C.
To conclude on the above observations, we formulate the following conjecture. To prove it, it is of course sufficient to show that the components of the Hessian matrix tend to zero (or any other nonpositive value) at the limit point $\mathbb{0}_n$ with nonnegative values close to the limit point.

Conjecture 3.6. *Let $\chi_1, ..., \chi_n$ be a given set of independently normally distributed random variables with strictly positive means and let $c > 0$ be a given deterministic capacity. Then there exists $p = p(\chi_1, ..., \chi_n, c)$ with $0.5 \leq p < 1$ such that $\Theta(x)$ is concave on the set $\{x \in X_{cont} | \Theta(x) \geq p\}$.*

In other words, we conjecture that for a given (realistic) instance of the $CCKP$ there exists a p such that the corresponding Lagrangian dual function is concave over the feasible region X_{cont}^{feas} of the relaxed $CCKP$. Note, however, that the feasible region of the inner maximization of the Lagrangian dual problem is $X_{cont} \supset X_{cont}^{feas}$. Nevertheless practical convergence of the algorithm can be assured: On the one hand, one can ensure that the infeasible solutions produced by the SAH algorithm are never very far from the feasible region by choosing the step size for x accordingly. On the other hand, practical convergence can be obtained by choosing the step size for λ such that shortly after producing an infeasible solution one re-obtains a feasible one.

H7: j is linear in each component of x. H7 is thus satisfied.

H8: Condition H8 is satisfied for all sequences of type $const/k$ with $const > 0$.

H9: We have

$$\begin{aligned}&\theta(x,\chi) \leq 1 \quad \text{and} \quad \Theta(x) = \mathbb{P}\{g(x,\chi) \leq c\} \geq 0\\ &\Rightarrow \theta(x,\chi) - \Theta(x) \leq 1 \Rightarrow \mathbb{E}[\theta(x,\chi) - \Theta(x)]^2 \leq 1\end{aligned}$$

It follows that condition **H9** is satisfied for all $\delta > 1$.

Convexification of the constraint function

As the left hand side function of the knapsack chance-constraint (3.19) is logarithmic concave (see Theorem 3.1), applying a logarithm makes it a concave, and even strictly concave, function. Therefore, an idea to overcome the non-concavity of Θ is to solve the following problem instead of problem (3.18):

$CCKP$ with concave constraint function

$$\max_{x\in\{0,1\}^n} \mathbb{E}\left[\sum_{i=1}^{n} r_i\chi_i x_i\right] \tag{3.23}$$

$$\text{s.t. } \log\left(\mathbb{P}\{\sum_{i=1}^{n} \chi_i x_i \leq c)\}\right) \geq \log(p). \tag{3.24}$$

After reformulation of the probability as an expectation and introducing a Lagrange multiplier λ, we obtain the following Lagrangian dual problem:

Lagrangian dual of the $CCKP$ with concave constraint function

$$\min_{\lambda\geq 0}\max_{x\in X_{cont}} \mathbb{E}[\sum_{i=1}^{n} r_i\chi_i x_i] - \lambda\cdot\left(\log p - \log\left(\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}[c - \sum_{i=1}^{n}\chi_i x_i]\right]\right)\right) \tag{3.25}$$

The problem is, that the obtained Lagrangian dual function is not of the form $\mathcal{L}(x,\lambda) = \mathbb{E}[l(x,\lambda,\chi)]$ anymore, as clearly

$$\log\left(\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}[c - \sum_{i=1}^{n}\chi_i x_i]\right]\right) \neq \mathbb{E}\left[\log\left(\mathbb{1}_{\mathbb{R}^+}[c - \sum_{i=1}^{n}\chi_i x_i]\right)\right]$$

due to the simple fact that $\log\left(\mathbb{1}_{\mathbb{R}^+}[c - \sum_{i=1}^{n}\chi_i x_i]\right)$ is not defined for $\mathbb{1}_{\mathbb{R}^+}[c - \sum_{i=1}^{n}\chi_i x_i] = 0$, which is the case whenever $c \leq \sum_{i=1}^{n}\chi_i x_i$. It is thus not evident how to solve problem (3.25) using a stochastic subgradient algorithm (see [Car10b] where a similar problem is discussed for the Augmented Lagrangian).

Conclusion on the theoretical analysis of the convergence of the SAH Algorithm

Although some of the conditions of Theorem 3.5 by Culioli and Cohen are not satisfied in the case of the $CCKP$ with independently normally distributed weights, the above discussion leads us to believe that these conditions are not mandatory in practice or can at least be bypassed by an adapted application of the SAH algorithm. More precisely, we think that some conditions might be weakened in the above theorem such as the strict concavity of the objective function. Other conditions might be adapted to the case of the $CCKP$ (or as well weakened) such as the requirement of subdifferentiability and local Lipschitz continuity of the function θ.
The concavity issue of the Lagrangian (resp. the constraint function) clearly cannot be bypassed if one wants to guarantee the convergence of the algorithm. We conjecture, however, that at least for large values of p the Lagrangian is concave over the feasible region of the $CCKP$. By choosing the right step size, it might be possible to keep the algorithm in this region and thus to achieve convergence. This is what has been confirmed by the numerical tests presented in the next section. Although we have remarked that the right choice of the step size significantly affects the convergence of the algorithm, it is nevertheless possible to choose a σ-sequence adapted for a whole set of instances.
Beneath the issue of concavity, the question of the existence of a saddle point of the Lagrangian is still open. While the former condition is clearly not satisfied on the whole feasible region of the Lagrangian dual problem, we conjecture that the second condition is fulfilled.

3.4.2 Numerical convergence tests

All numerical tests have been carried out on an Intel PC with $2GB$ RAM. The SAH algorithm as well as the B&B framework have been implemented in C language. The random numbers needed for the runs of the SAH algorithm where generated in advance using the gsl C-library and stored in a batch.

Convergence of the SAH Algorithm involving the FD-Method

Let us first study the numerical convergence of the SAH algorithm when approximating the gradient of θ using the FD-method (see Figure 3.4.2 (bold curve)). The algorithm converges fast and only slight variances can be seen after around 300 iterations. We therefore fix the maximum number of iterations for the SAH algorithm at 500 (for numerical results see subsection 3.4.2 and Table 3.1).

Need of problem reformulation when using the SAH Algorithm involving IP-Method

During our first numerical tests we remarked that the SAH algorithm involving the IP-method diverges once the produced solution x^k is infeasible, i.e. once $c - g(x^k, \chi^k) < 0$. The reason is quite obvious:

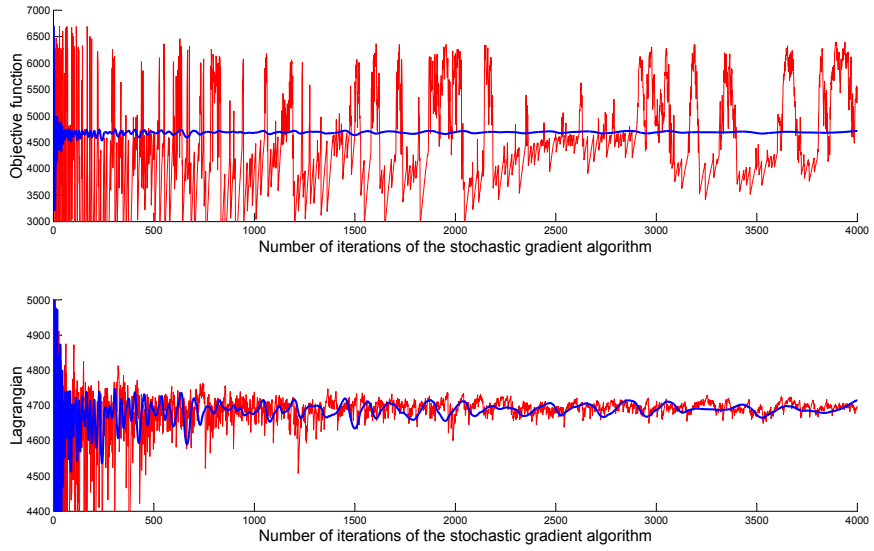

Figure 3.4.2: Convergence of the *SAH* algorithm solving the continuous *ECKP*: FD-method (bold curve) versus initial IP-method (upper figure) and IP-method with modified choice of x_{kappa} (lower figure)

The expectation-constraint states

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - g(x,\chi))\right] \geq p \Leftrightarrow p - \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - g(x,\chi))\right] \leq 0$$

We thus get the Lagrangian

$$\mathcal{L}(x,\lambda) = \mathbb{E}\left[\sum_{i=1}^{n} r_i \chi_i x_i\right] - \lambda\left(p - \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - g(x,\chi))\right]\right)$$

Using the IP-method, we rewrite this Lagrangian as follows:

$$\widetilde{\mathcal{L}}(x,\lambda) = \mathbb{E}[\sum_{i=1}^{n} r_i \chi_i x_i] - \lambda\left(p + \mathbb{E}[\mathbb{Y}_{\mathbb{R}^+}(c - g(x,\chi))\frac{(\chi_\kappa - \mu_\kappa)}{x_\kappa \sigma_\kappa^2}]\right) \tag{3.26}$$

Let us denote $\widetilde{l}$ the function inside the expectation of the Lagrangian (3.26), i.e.

$$\widetilde{l}(x,\lambda,\chi) = \sum_{i=1}^{n} r_i \chi_i x_i - \lambda\left(p + \mathbb{Y}_{\mathbb{R}^+}(c - g(x,\chi))\frac{(\chi_\kappa - \mu_\kappa)}{x_\kappa \sigma_\kappa^2}\right)$$

It follows (for a point (x,χ) with $c \neq g(x,\chi)$)

$$\left(\nabla_x \widetilde{l}(x,\lambda,\chi)\right)_h = r_h \chi_h + \lambda\left(\mathbb{1}_{\mathbb{R}^+}(c - g(x,\chi))\frac{(\chi_\kappa - \mu_\kappa)}{x_\kappa \sigma_\kappa^2}\left(\chi_h + \frac{(c - g(x,\chi))}{x_\kappa}\nu_h^\kappa\right)\right)$$

Remark that the term that multiplies λ is zero whenever the capacity constraint *is not* satisfied (i.e. when $c < g(x,\chi)$). In these cases *all* of the components of the current x^k are incremented (as all the components of $(r_1\chi_1,\ldots,r_n\chi_n)^T$ are positive) although at least one component should be decreased in order to better fit the capacity.
Of course, the expectation-constraint can be equivalently reformulated as

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(g(x,\chi)-c)\right] \leq 1-p \Leftrightarrow \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(g(x,\chi)-c)\right]-(1-p) \leq 0$$

In this case the h-th component of the gradient of $\widetilde{l}$ for a point (x,χ) with $g(x,\chi) \neq c$ is given by

$$\left(\nabla_x \widetilde{l}(x,\lambda,\chi)\right)_h = r_h\chi_h - \lambda\left(\mathbb{1}_{\mathbb{R}^+}(g(x,\chi)-c)\frac{(\chi_\kappa-\mu_\kappa)}{x_\kappa\sigma_\kappa^2}\left(\chi_h - \frac{(g(x,\chi)-c)}{x_\kappa}\nu_h^\kappa\right)\right)$$

Here the term that multiplies λ is zero whenever the capacity constraint *is* satisfied. In this case the components of x^k are incremented by the components of the positive vector $(r_1\chi_1^k,\ldots,r_n\chi_n^k)^T$ (multiplied by the corresponding factor σ^k). When the capacity constraint is not satisfied the term with coefficient λ is subtracted from this vector in order to correct x^k. For $h \neq \kappa$, this term is positive whenever $(\chi_\kappa^k - \mu_\kappa) > 0$ (see also subsection 3.4.2) and the Lagrange multiplier λ is now playing its assigned role of a penalty factor.

Convergence of the *SAH* Algorithm involving the IP-Method

When studying the behavior of the *SAH* algorithm involving the IP-method, we remark that even after the reformulation described in subsection 3.4.2 the IP-method is much less robust than the FD-method (see upper graphic of Figure 3.4.2). In this case, "robust" means that when using the FD-method the *SAH* algorithm reacts much less on the variations of the drawn samples (see also graphic 3.4.3 where for a fix decision vector subgradients are computed for 500 different samples of the random variables). This higher variance causes a slower convergence of the average IP-subgradient towards the expected subgradient (Figure 3.4.4) (both Figure 3.4.3 and Figure 3.4.4 show *one* component of the multidimensional subgradient). This might be one explanation for the higher fluctuations in the objective function values produced by the *SAH* algorithm involving the IP-method (upper graphic of Figure 3.4.2). Unfortunately, these high fluctuations provoke a slower convergence towards the optimum compared with the FD-method. As our stopping criterion is a fixed maximum number of iterations, it naturally follows that the produced solutions are, in general, less good than in the case of the FD-method.

Modified choice of x_κ

Let us once more study the h-th component of the gradient of the Lagrangian function at a point (x,χ) with $g(x,\chi) \neq c$ obtained by the IP-method after reformulation:

$$\left(\nabla_x \widetilde{l}(x,\lambda,\chi)\right)_h = r_h\chi_h - \lambda\left(\mathbb{1}_{\mathbb{R}^+}(g(x,\chi)-c)\frac{(\chi_\kappa-\mu_\kappa)}{x_\kappa\sigma_\kappa^2}\left(\chi_h - \frac{(g(x,\chi)-c)}{x_\kappa}\nu_h^\kappa\right)\right)$$

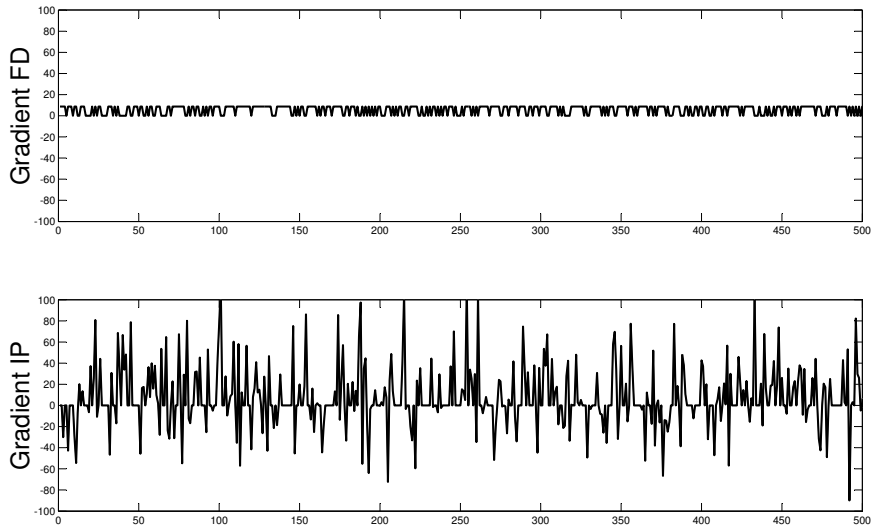

Figure 3.4.3: Calculating a subgradient for 500 samples of χ: FD-method (upper figure) versus initial IP-method (lower figure)

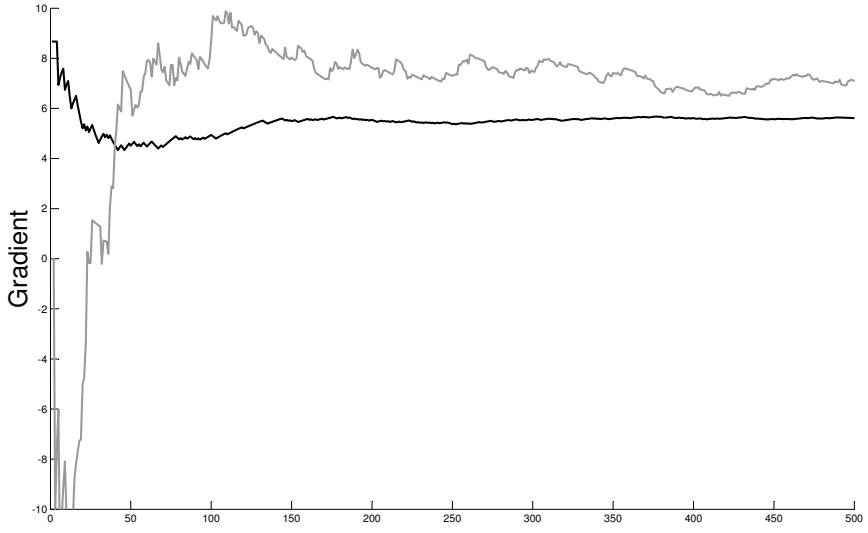

Figure 3.4.4: Calculating an expected subgradient by sampling: FD-method (black) versus initial IP-method (gray)

In case of an overload, we expect this gradient to be negative for some indexes h in order to decrease the total weight of the knapsack. However, for $h \neq \kappa$ and $\mathbb{1}_{\mathbb{R}^+}(g(x,\chi)-c) = 1$, the term that multiplies λ is positive at iteration k if and only if $(\chi_\kappa^k - \mu_\kappa) > 0$ which is the case with a probability of 50%. In the other 50% of cases, the gradient is strictly positive and *all* components of x with index different from κ are incremented despite the overload. Due to this observation we propose the following improved choice of the index κ: Instead of just choosing κ in the k^{th} iteration such that $x_\kappa^k = ||x^k||_\infty$ (see Proposition 3.4), we choose κ as follows:

$$\kappa = \arg\max_{i=1,\dots,n} \{x_i^k | x_i^k \geq 1/n \wedge (\chi_i^k - \mu_i) > 0\}$$

If $\{x_i^k | x_i^k \geq 1/n \wedge (\chi_i^k - \mu_i) > 0\} = \emptyset$, we choose κ as before, i.e. such that $x_\kappa^k = ||x^k||_\infty \geq 1/n$.
With this modification we were able to significantly improve the convergence of the *SAH* algorithm involving the IP-method (see lower graphic of Figure 3.4.2) and could fix the maximum number of iterations to 1000.

Optimum, CPU-time and gap - Comparison with an *SOCP* approach

	Arrow-Hurwicz & FD-method		**Arrow-Hurwicz & IP-method**		**MOSEK**
n	**CPU (msec)**	**Gap**	**CPU (msec)**	**Gap**	**CPU (msec)**
C./B.	1	0.03%	2	0.03%	25
15	1	0.02%	2	0.02%	22
20	1	0.02%	3	0.02%	22
30	1	0.02%	4	0.02%	22
50	3	0.01%	6	0.02%	24
75	4	0.02%	9	0.02%	26
100	5	<0.01%	12	0.01%	28
150	8	<0.01%	17	0.01%	31
250	13	0.01%	28	0.01%	37
500	25	0.01%	56	0.01%	52
1000	51	<0.01%	111	0.01%	89

Table 3.1: Numerical results for the continuous *ECKP*

We tested the *SAH* algorithm on the instance used in [CB98] (called C./B.) as well as on 50 randomly created instances for each dimension. The data given in the following tables are average values over these instances. As remarked at the end of section 3.2, the choice of normally distributed weights implies that theoretically weights can take negative

values. The test instances were therefore generated in a way that with high probability the variance/mean ratio is below 1/4 (see [CB98] or chapter 2 for details concerning the instance generation). This implies a very low probability for the realization of negative weights which has been confirmed during our numerical tests: Although a high number of scenarios had to be generated (either 500 or 1000 for each run of the SAH algorithm), we encountered no negative weight realizations.
In Table 3.1, the numerical results of the SAH algorithm involving the FD- or IP-method are compared with those using an $SOCP$ solver: The chance-constraint of the $CCKP$ with independently, normally distributed weights can be equivalently reformulated as a deterministic equivalent $SOCP$ constraint ($SOCP$=*Second Order Cone Programming*) as shown in subsection 3.1.1 (see for example [LVBL98]). Moreover, the bounding constraints $0 \leq x_i \leq 1$ $(i = 1, ..., n)$ can be replaced by two $SOCP$ constraints:

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

where $A_i \in \mathbb{R}^{1\times n}$, $A_i[1,k] = 0\ \forall k \neq i$ and $A_i[1,i] = 1$. To solve the obtained $SOCP$ problem we used the $SOCP$ solver by MOSEK in C-programming language that applies an interior point method ([MOS10]). The gaps given in the following tables are the relative gaps between the solution produced by the MOSEK software and the approximate solution obtained when using the stochastic subgradient algorithm.
First of all, we remark that the best solution values given by the SAH algorithm involving the FD-method are comparable to those produced by the IP-method. Small variances in favor of one algorithm or the other can be explained by the difficulty presented by the choice of the two σ-sequences for the SAH algorithm. Choosing the right parametrization for these sequences has an important influence on the convergence of the algorithm and the best found solution (see subsection 3.4.1).
In terms of running time, both methods outperform the $SOCP$ algorithm for small and medium size instances. For higher dimensional problems, this is still true for the FD-method. However, the SAH algorithm involving the IP-method needs approximately twice the time than when using the FD-method. This is of course due to the total number of iterations which we were able to fix at 500 when using the FD-method, while we need 1000 iterations with the IP-method in order to obtain equally good solutions. For higher dimensional instances the MOSEK interior point method needs therefore less CPU-time than the SAH algorithm involving the IP-method.

3.5 Solving the (combinatorial) $CCKP$ - Numerical Results

The combinatorial problem has been solved using the B&B algorithm described in section 3.3.2. The algorithm has been tested on the instances previously used for the tests of the SAH algorithm.

We once more stored the random numbers needed for the runs of the SAH in a batch. As the total number of runs during the B&B algorithm is unknown and the number of random numbers needed for all those runs is generally very high, we only stored random numbers for a limited number of runs. Before starting a run of the SAH we then chose randomly one of the instances of random numbers. Remark that, as the runs of the SAH are independent, one stored instance of random numbers would theoretically be sufficient.
When studying Table 3.2 one might first of all remark that the B&B algorithm that uses the $SOCP$ program to obtain upper bounds considers more nodes than when using an SAH algorithm. This is not due to a better choice of the upper bounds in the latter method as in both algorithms the upper bounds are supposed to be the same (i.e. the optima of the corresponding relaxed problems). However, as the best solution found by the approximate SAH algorithm might be slightly smaller than the optimal solution value of the relaxed problem, more branches are pruned than with the primal-dual $SOCP$ algorithm. Of course, this could theoretically also cause the pruning of a subtree that contains the optimal solution.
Similarly, the SAH algorithm involving the IP-method considers on average more nodes than when using the FD-method. This implies that the solutions values of the relaxed subproblems produced by the FD-method are less good than those obtained when using the IP-method. As both methods perform equally on the relaxed overall problems (see subsection 3.4.2), we can thus draw the conclusion that the FD-method is less robust in the following sense: Instead of choosing particular σ-sequences for each instance or even each subproblem that has to be solved during the B&B algorithm, we fixed one parametrization for each dimension. However, the subproblems solved during the B&B algorithm are mostly lower dimensional problems and the SAH algorithm using the fixed σ-sequences might therefore perform less good on these subproblems. This seems to be especially the case when using the FD-method.
Anyway, if we only allow an average computing time of up to $1h$, the $SOCP$ algorithm can only be used up to a dimension of 50. In contrary, when using the FD-method and allowing 500 iterations in the SAH algorithm, we are able to solve problems up to a dimension of 250 within an average CPU-time of $1h$.

3.6 Concluding remarks and future work

In this chapter we proposed a method to solve the Chance-Constrained Knapsack problem with independently, normally distributed weights. As framework we once more chose a B&B algorithm and to obtain upper bounds we proposed to solve the corresponding continuous relaxations of the problem. This, in turn, was done by applying a stochastic subgradient algorithm. As, contrary to the first chapter, the treated problem contains a constraint other than the $0-1$ bounds, we used a particular stochastic subgradient algorithm known under the name of Stochastic Arrow-Hurwicz algorithm. The idea of this approach is to introduce a Lagrange multiplier in order to handle the chance-constraint. The so obtained unconstrained Lagrangian dual problem with expected value objective

	Arrow-Hurwicz & FD-method			
n	**Optimum**	**considered nodes**	**CPU-time (sec) B-and-B**	**Gap**
C./B.	4595	122	0.09	0 %
15	4840	31	0.02	0 %
20	6634	69	0.07	0 %
30	10272	271	0.35	0 %
50	16975	3341	6.43	0 %
75	25548	6187	18.43	0 %
100	33895	12093	45.87	-
150	50672	37076	244.11	-
250	85189	65890	623.53	-
500	*	*	*	*

	Arrow-Hurwicz & IP-method			
n	**Optimum**	**considered nodes**	**CPU-time (sec) B-and-B**	**Gap**
C./B.	4595	122	0.21	0 %
15	4840	34	0.06	0 %
20	6634	63	0.13	0 %
30	10272	436	1.22	0 %
50	16975	7051	31.34	0 %
75	25548	23911	161.47	0 %
100	33894	98479	1049.18	-
150	*	*	*	*

	MOSEK		
n	**Optimum**	**considered nodes**	**CPU-time (sec) B-and-B**
C./B.	4595	122	0.406
15	4840	34	0.082
20	6634	66	0.236
30	10272	350	1.801
50	16975	7406	70.914
75	25548	62175	1535.520
100	*	*	*

* CPU-time exceeds 1h

Table 3.2: Numerical results for the (combinatorial) $ECKP$

function can then be solved by a stochastic subgradient method. The function inside this expectation is, however, not subdifferentiable (it is not even convex) as it is the sum of a linear and an indicator function. While the Approximation by Convolution method presented in the previous chapter would have been a possible choice to overcome this problem, we proposed two additional methods, one based on finite differences, one on integration by parts.
Due to the fact that the constraint function is only logarithmic concave (and not concave) on the feasible region of the Lagrangian problem, the convergence of the algorithm to the saddle point of the Lagrangian dual function cannot be guaranteed. Numerical results however showed that a careful choice of the initial solution and adapted step sizes lead nevertheless to a convergence of the algorithm on the tested instances. Additional convergence issues of the stochastic subgradient algorithm involving the IP-method were eliminated by choosing an adapted formulation of the chance-constraint and making a more careful choice of the decision variable used for the IP-method.
We compared our method with the alternative of solving the relaxed problem using an interior point method. It turned out, that our algorithm can solve problems with up to 250 items in around 10 minutes. When using the interior point method, solving the instances of 100 items already requires an average computing time of more than $1h$.
In case of normally distributed items the problem of nonconcavity of the Lagrangian dual could be solved by reformulating the constraint as a Second Order Cone constraint as shown in the introduction. The obtained constraint has a strictly concave left hand side function, which makes the Lagrangian dual strictly concave, as well. This approach will be used in the following chapter to solve the relaxation of the Two-Stage Knapsack problem with chance-constraint. In case of the Chance-Constrained Knapsack problem this idea would lead directly to a deterministic equivalent Lagrangian dual that can be solved with a deterministic (instead of stochastic) gradient algorithm or other methods for convex optimization (e.g. interior point methods).
Solving the Chance-Constrained Knapsack problem with concave constraint function introduced at the end of subsection 3.4.1 might be another solution to the nonconcavity problem: The idea is to apply a logarithm to both sides of the constraint. The use of a stochastic gradient or subgradient algorithm is, however, generally not adapted to the form of the corresponding Lagrangian dual function. In case of a normal distribution (and maybe some other continuous, logarithmic concave distributions), a deterministic equivalent formulation of the Lagrangian function could be obtained and the relaxed problem solved using a deterministic gradient algorithm. For distributions whose CDF is less easily computable, the question of how to solve the obtained Lagrangian dual problem is less obvious and remains to study.
A direct extension of the problems treated in chapters 2 and 3 is the Simple Recourse Knapsack problem with chance-constraint. Problems that combine simple recourse and chance-constraint have been studied previously as of course it might in some cases be interesting to control both the expected overweight and the probability that the chance-constraint is violated. As it turns out, the Two-Stage Knapsack problem with chance-constraint studied in the following section can be relaxed to a continuous Simple Recourse

Knapsack problem with chance-constraint. We will solve the latter by combining the methods and results of chapters 2 and 3.
Like in the case of the method that we proposed in the previous chapter, future work on the Chance-Constrained Knapsack problem will first of all consist in extending the proposed solution method to more general distributions. Once more, such an extension might naturally entail additional problems such as the question of the evaluation of the constraint function. As explained in the conclusion of the previous chapter, these problems seem, nevertheless, reasonably manageable. It is, however, probable that the nonconcavity of the (simple) Lagrangian and the convergence of the Stochastic Arrow-Hurwicz algorithm would still be an issue.

4 The Two-Stage Knapsack Problem with Full Recourse

4.1 Introduction

4.1.1 Two-Stage Problems

Two-stage optimization problems are probably the "oldest" ([Dan55]), and certainly the most studied stochastic optimization problems. As the name indicates, the decision process consists of two stages: In the first stage the random parameters are still unknown, whereas the second-stage decision is made after the revealing of their actual values. The second-stage decision is therefore mainly a corrective decision that aims to increase the final gain (or decrease the final cost), to make violated constraints be respected, or both. Most work on two-stage problems that can be found in the literature is devoted to the two-stage problem with linear objective functions and constraints (TSP). This particular variant is generally stated as follows:

$$
\begin{aligned}
(TSP) \quad \min \quad & c^T x + \mathbb{E}\left[\mathcal{Q}(x,\chi)\right] & \\
\text{s.t.} \quad & Ax = b, & (4.1a)\\
& x \in X \subseteq \mathbb{R}^{n_1}_+. & (4.1b)\\
& \mathcal{Q}(x,\chi) = \min \; d(\chi)^T y & (4.1c)\\
& \qquad \text{s.t.} \; W(\chi)y \geq h(\chi) + T(\chi)x, & (4.1d)\\
& \qquad\quad y \in Y \subseteq \mathbb{R}^{n_2}_+. & (4.1e)
\end{aligned}
$$

where $c \in \mathbb{R}^{n_1}$, $A \in R^{m_1 \times n_1}$ and $b \in \mathbb{R}^{m_1}$ are deterministic matrices, χ is a random vector and $d(\chi) \in \mathbb{R}^{n_2}$ is the (possibly random) second-stage reward or cost vector. $W(\chi) \in \mathbb{R}^{m_2 \times n_2}$ is called the *recourse matrix*, while $T(\chi) \in \mathbb{R}^{m_2 \times n_1}$ is the *technology matrix*. Note that the overall objective function of the TSP (4.1) is clearly not linear. In some special cases it can however be shown to be piecewise linear (see Theorem 4.1).
The TSP (4.1) is said to have *fixed recourse* if the recourse matrix is deterministic (or fixed), i.e. $W(\chi) = W$. A special case of the TSP with fixed recourse is the *simple recourse problem*. One can show that in this case the components of the second-stage decision vector depend directly on the sign of $h(\chi) + T(\chi)x$. It follows that the problem can in most cases be reformulated as a single-stage decision problem where the second-stage decision has been simplified to the act of paying a penalty in case one of the constraints is not satisfied (see chapter 2 for details). On the contrary, *full recourse problems* involve

second-stage decisions that can be seen as actual corrections of the decision made in the first stage. In this chapter we will focus on the last variant.
A different manner to classify TSPs is by the feasibility of the second-stage problem. Problems with *relatively complete recourse* are problems whose second-stage problem always has a feasible solution, no matter what feasible decision has been made in the first stage. If $Y = \mathbb{R}^{n_2}_+$, this can be formalized as follows: Let $\operatorname{pos} W := \{t | \exists y \geq 0 : Wy \geq t\}$, i.e. $\operatorname{pos} W$ is the *positive hull* of W. Then, relatively complete recourse means that one has $h(\hat{\chi}) + T(\hat{\chi})x \in \operatorname{pos} W(\hat{\chi})$ for any feasible solution x of the first-stage problem and for any realization $\hat{\chi}$ of the random vector. A special case of relative complete recourse problems are problems with *complete recourse* where $\operatorname{pos} W(\hat{\chi}) = \mathbb{R}^{m_2}$ for any realization $\hat{\chi}$ of χ. Note that continuous simple recourse problems always have complete recourse (see chapter 4).
If the second-stage variables are required to be integer or binary, the above definitions apply with the *integer hull* $\operatorname{int} W := \{t | \exists y \in \mathbb{Z}^{n_2}_+ : Wy \geq t\}$ or *binary hull* $\operatorname{bin} W := \{t | \exists y \in \{0,1\}^{n_2} : Wy \geq t\}$, respectively. Similar to the continuous case, simple integer recourse problems always have complete recourse. In contrary, TSPs with binary (or bounded integer) second-stage variables never have complete recourse as for all realization $\hat{\chi}$ of the random vector $(W(\hat{\chi})y)_i$ is bounded.

Two-Stage Problems with continuous recourse

Relatively few research has been published on the properties of the general linear two-stage problem (4.1) (see [WW67], [Kal76] or [KM05]). Most papers, surveys and books on two-stage programming restrict their study to two-stage models with fixed complete recourse. One reason is the question of how to handle the second-stage feasibility of a two-stage model in the case of random (or non-complete) recourse. Assuming complete recourse assures that for any (feasible or infeasible) first-stage decision vector and any scenario there exists a feasible second-stage solution. In particular, one does not have to worry about the case where the second-stage problem takes value $+\infty$ (since infeasible) with positive probability, which would result in an overall solution value of $+\infty$.
By duality, the definition of complete recourse is equivalent to saying that problem (4.1) has complete recourse if $W(\hat{\chi})^T\lambda \geq 0$ implies $\lambda = 0$ for all realizations $\hat{\chi}$ of χ (see for example [vdV95]). Consequently, one has a rather simple certificate for complete recourse in the case of TSPs with fixed recourse.
The following theorem is a well known result for TSPs with fixed complete recourse:

Theorem 4.1. *[see e.g. [vdV95],[KM05]] Given a two-stage problem of type* (4.1) *with fixed complete recourse and dual feasible second-stage problem (for any realization of the random parameters). Then, for any realization $\hat{\chi}$ of χ, the recourse function $\mathcal{Q}(x, \hat{\chi})$ is*

- *real-valued,*
- *piecewise linear and convex in x,*
- *Lipschitz continuous in x,*

- *subdifferentiable in x.*

Moreover, if χ follows a discrete distribution, the expected value function $\mathbb{E}[\mathcal{Q}(x,\chi)]$ is piecewise linear and convex, as well. In the case of continuously distributed random variables, $\mathbb{E}[\mathcal{Q}(x,\chi)]$ is continuously differentiable.

Similar results can be obtained for general fixed recourse problems (see [vdV95]).
Only few references are available that explicitly study the case of TSPs with random recourse matrix. For continuous second-stage decision variables the recourse function $\mathcal{Q}(\cdot,\chi) : \mathbb{R}^{n_2} \to \mathbb{R}$ (and thus the overall objective function) can, however, be shown to be convex even when the recourse matrix depends on the random vector χ.

Two-Stage Problems with integer recourse

Like in the special case of simple recourse problems (see chapter 2), general TSPs become much harder to solve when (some of) the second-stage decision variables are required to be integer. First of all, the second-stage problem of integer recourse problems is generally $\mathcal{NP}$-hard. In addition, one looses in general some nice properties of TSPs with continuous recourse variables such as continuity and convexity.
The study of structural properties of TSPs with integer recourse is, to the best of our knowledge, for instance restricted to the fixed recourse case. Most papers assume in addition complete recourse, a fixed technology matrix and/or a fix recourse cost vector $d(\chi) = d$.
One of the most general structural results for TSPs with integer recourse is due to Schultz:

Theorem 4.2 (see e.g. [Sch93],[Sch95]). *Given a two-stage problem of type* (4.1) *having*

- *fixed and complete integer recourse,*
- *fixed recourse costs,*
- *a dual feasible second-stage problem.*

Moreover, let the expectation of the random vector χ be finite. Then, the expected value function $\mathbb{E}[\mathcal{Q}(x,\chi)]$ is lower semi-continuous in x. If, in addition, χ has an absolutely continuous density, $\mathbb{E}[\mathcal{Q}(x,\chi)]$ is continuous in every point x.

4.1.2 The Two-Stage Knapsack Problems with full recourse

In this chapter we study a Two-Stage Knapsack problem with full recourse and random item weights: In the first-stage items are placed in the knapsack without knowledge of their actual weight. In the second stage, the weights of all items have been revealed. The second-stage decision consists in deciding which items to additionally put in the knapsack

and which to remove in order to increase the total gain and/or to make the capacity constraint be satisfied. The general problem can be stated as follows:

$$
\begin{aligned}
(GTSKP) \quad \min \quad & \mathbb{E}\left[r^T(\chi)x\right] + \mathbb{E}\left[\mathcal{Q}(x,\chi)\right] \\
\text{s.t.} \quad & x \in \{0,1\}^n. && \text{(4.2a)} \\
& \mathcal{Q}(x,\chi) = \max \ \overline{r}(\chi)^T y^+ - d(\chi)^T y^- && \text{(4.2b)} \\
& \qquad \text{s.t.} \ (x + y^+ - y^-)^T \chi \leq c, && \text{(4.2c)} \\
& \qquad\quad y_i^+ \leq 1 - x_i \quad \forall\, i = 1,\ldots,n, && \text{(4.2d)} \\
& \qquad\quad y_i^- \leq x_i \quad \forall\, i = 1,\ldots,n, && \text{(4.2e)} \\
& \qquad\quad y^+, y^- \in \{0,1\}^n. && \text{(4.2f)}
\end{aligned}
$$

Here χ is an n-dimensional random vector that represents the random weights and c is the capacity of the knapsack.
The second-stage decision vectors y^+ and y^- have the same dimension as the first-stage decision vector x. y^+ models which items are added in the second stage. Of course, an item can only be added if it has not already been placed in the knapsack in the first stage (constraint (4.2d)) and for all realizations $\hat{\chi}$ of χ the second-stage reward $\overline{r}(\hat{\chi})_i$ of item i should be strictly smaller that its first-stage reward $r(\hat{\chi})_i$. When we want to remove the item with index i in the second stage, we set y_i^- to 1 (which is only possible if the item has been added in the first stage, see constraint (4.2e)). In this case, a penalty $d(\chi)_i$ occurs that is strictly greater than the corresponding first-stage reward.
The $GTSKP$ is a two-stage problem with integer recourse. It has relatively complete recourse, as for any feasible first-stage decision, removing all items is always a feasible solution for the second stage. Due to the binary recourse requirement, the problem does, however, not have complete recourse. In addition, the $GTSKP$ has random recourse and technology matrices (the random weight vector) and the rewards and costs might depend on the random weights χ_i. The existing results concerning structural properties of general TSPs can therefore not directly be applied to the $GTSKP$.
In this chapter, we study two variants of this problem. Their formulations are given in sections 4.2.1 and 4.3.1. While in the first variant we assume independently normally distributed item weights, the second study concerns the case of a finite number of scenarios.
The Two-Stage Knapsack problem with full recourse has been studied before by Lisser et al. ([LLH09]). Their study concerns the more general, quadratic version of the problem where an additional reward (and cost) is introduced for each pair of items. The authors assume that some of the item weights are already known in the first stage. They therefore introduce an additional knapsack constraint in the first stage. More precisely, if in the first stage some of the items with deterministic weight are chosen, these items have to already respect a certain knapsack capacity. In the general model (4.2) we allow that the items with deterministic weight (i.e. with zero variance) chosen in the first stage do not respect the knapsack capacity. If the penalties in the second stage are small enough, it might even be optimal to choose all items in the first stage.

4.1.3 Solving Two-Stage Problems

TSPs with finite number of scenarios and continuous decision variables

Like in the special case of simple recourse problems, general TSPs can be reformulated as deterministic equivalent linear programming problems if the random vector χ has only a finite number of possible outcomes $\chi^1, \ldots, \chi^K$ with corresponding probabilities $p^1, \ldots, p^K$:

$$
\begin{aligned}
(TSP^D) \quad \min \quad & c^T x + \sum_{k=1}^{K} p^k \left(d(\chi^k)^T y^k \right) \\
\text{s.t.} \quad & Ax = b, && && \text{(4.3a)} \\
& T(\chi^k)x + W(\chi^k)y^k \leq h(\chi^k) && \forall\, k = 1, \ldots, K, && \text{(4.3b)} \\
& x \in X \subseteq \mathbb{R}^{n_1}_+, && && \text{(4.3c)} \\
& y^k \in Y \subseteq \mathbb{R}^{n_2}_+ && \forall\, k = 1, \ldots, K. && \text{(4.3d)}
\end{aligned}
$$

Parameters and variables are defined as in the general model of the TSP (4.1).
The first one to mention two-stage programs (with discretely distributed random variables) was George Dantzig in his article from 1955 ([Dan55]). Concerning the resolution of such problems the author, however, refers the reader to general linear programming techniques that do not make advantage of the special structure of TSPs.
In 1960, Dantzig published together with P. Wolfe an article on a "Decomposition principle for linear programs" ([DW60]). This so called *Dantzig-Wolfe Decomposition* can be applied to linear programs with a very particular structure. In [DM61] Dantzig and Madansky show that in case of continuous first- and second-stage decision variables, the dual of problem (4.3) has exactly this structure. This allows the authors to apply the Dantzig-Wolfe Decomposition to solve TSPs with random right hand side vector h.
In 1969, Van Slyke and Wets (see [SW69]) first proposed the so called *L-shaped method* to solve the continuous TSP^D. Until today it is one of the most famous methods to solve continuous TSPs. Contrary to the method by Dantzig and Madansky, the authors directly solve the primal problem (4.3). The notation "L-shaped" is due to the shape of the constraint matrix of the linear programming formulation (4.3). The idea of the algorithm is based on Benders Decomposition technique: The algorithm starts with solving the first-stage problem and then iteratively adds so called *feasibility* and *optimality cuts*. While feasibility cuts ensure that the final first-stage solution produced by the process is second-stage feasible, optimality cuts continuously add new lower bounds on the second-stage solution value. In [SW69] the authors restrict their study to the case where only h is random. Their method can, however, be easily extended to the case of fixed recourse problems (see for example the overview on Optimization under Uncertainty in [RR02]).
The L-shaped method has been advanced in [BL88]: Instead of adding one single cut at each iteration, the authors propose to add several cuts at once by linearizing $\mathcal{Q}(x, \chi^k)$ (where k is one of the finitely many scenarios). The advantage is that a smaller number of iterations is needed. However, there are also examples where the fact that the subproblem

contains more constraints (and therefore takes more time to be solved) considerably slows down the algorithm.
Although the L-shaped method by Van Slyke and Wets (and its later improvements) present an important and universal tool in continuous two-stage programming, its application to large scale problems is restricted. The reason is the continuously increasing number of cuts and size of the master problem. The Regularized Decomposition method by Ruszczyński ([Rus86]) was created to overcome these difficulty. The method can be applied to a more general class of problems whose objectives consist of a sum of convex piecewise-linear functions. The method is based on the same decomposition principles as the L-shaped method and can thus also be seen as an advanced version of this basic algorithm. However, a regularizing term is added to the objective function. This term is basically the squared euclidean distance between current solution vector and a so called trial point. The regularizing term serves to keep the step size (i.e. the difference between the last calculated and new solution vector) small. It thus has a stabilizing effect that results in a smaller number of needed iterations. Moreover, at most $n_1 + 2K$ cuts need to be stored at the same time due to the use of only linearly independent, active cuts. Computational tests have shown that the Regularized Decomposition method outperforms the basic L-shaped method as well as the multi-cut version of [BL88] on large scale problems (see [Rus93]). Independent tests have been conducted by Kall and Mayer that as well demonstrate the efficiency of the approach compared to other LP-solvers (see [KM93] and [Kal94]).

Integer TSPs with finite number of scenarios

In the case of the Integer TSP some of the first- and/or second-stage variables are assumed to be integer or binary.
In case where only the first-stage variables are required to be integer, the problem inherits most of the properties of the corresponding continuous TSP.
Two-stage problems with integer recourse, however, are generally much more difficult to solve than their continuous counterpart. Consequently, most algorithms that have been proposed for integer recourse problems are either restricted to special cases (e.g. fixed recourse) or have some serious drawbacks, contrary to the continuous case where the L-shaped method is mostly accepted to be an effective algorithm to exactly solve all types of two-stage problems. Moreover, while the L-shaped method by Van Slyke and Wets already dates back to 1969, the study of integer TSPs is rather young and most of the work on how to solve integer TSPs has been published in the last 17 years.
The cut generation of the L-shaped method relies heavily on the fact that in the case of continuous second-stage decision variables the recourse function $\mathcal{Q}$ is convex. In the case of integer second-stage variables this is generally not true. Laporte and Louveaux ([LL93]) solved this problem by proposing optimality cuts that are valid for arbitrary second-stage and binary first-stage variables. Their idea is based on the existence of a lower bound on the second-stage solution value due to the binary requirement in the first stage. The integrality of the first-stage variables is handled by a branch-and-cut procedure.

In [CT98] Carøe and Tind propose a method for problems with continuous first and integer second-stage variables (that however can be extended to integer first-stage variables) (see also [Car10a]). The method is based on general Benders decomposition and makes use of the theory of generalized duality in the case of (mixed) integer variables. The integrality of the second-stage decision variables is handled by either cutting plane or B&B techniques. However, the authors admit that the nonconvex integer problem that has to be solved at each iteration is an important drawback of their method. An extended version of their algorithm has been proposed in [NT08]. Among others, this extension is one of the rare exceptions of solution algorithms for integer TSPs that allows for the solution of problems with random recourse matrix.
In [SSvdV98] Schultz et al. studied the case where only the right hand side of the second-stage constraint is random. Their algorithm is mainly composed of two steps: First, one computes a so called *Gröbner basis* for the second stage problem based on the integrality of the second-stage variables. This basis allows for a rather cheap evaluation of the second-stage value function. The second step consists of computing a finite set of so called candidate solutions for the first-stage that is known to contain an optimal solution. This optimal solution can then be detected by evaluating the second-stage value function at each point of this set. The authors acknowledge that, on the one hand, the computation of the Gröbner basis is generally of "exponential complexity" and, on the other hand, the finite set of first-stage variables can be "enormous". Although they show that their algorithm is able to solve problems that are intractable when solved with common mixed-integer solvers, their method is (for instance) clearly restricted to the resolution of problems of moderate size.
The main idea of the method by Carøe and Schultz presented in [CS99] is to introduce one first-stage solution vector for each of the second-stage scenarios and to add a so called *non-anticipativity* constraint that assures that all these first-stage vectors are equal. The advantage of this approach is that the corresponding Lagrangian dual (where the non-anticipativity constraint has been relaxed in the objective function) is separable into smaller problems that can be independently solved. However, these subproblems contain first- and second-stage variables, contrary to the Benders decomposition based approaches. An important advantage compared with the approach by Carøe and Tind is the convexity (resp. concavity) of these subproblems. However, due to integrality conditions of the first-stage variables, an optimal solution of the Lagrangian Dual does not necessarily satisfy the non-anticipativity constraint. Therefore, the authors additionally apply a B&B algorithm to find a feasible solution. Like the method by Carøe and Tind this approach relies thus on the integrality of the first-stage decision variables.
Most of these early algorithmic proposition lack serious computational studies, probably due to the evident drawbacks present in each of these pioneer algorithms.
A computationally promising approach to solve integer two-stage problems with discrete distributions has been proposed by Ahmed et al. in [ATS03]: The authors present a finite branch-and-bound algorithm that does not require neither the first-stage variables to be discrete, nor the recourse matrix to be deterministic. Their idea is based on the observation that by making a change of variables one can achieve that the discontinu-

ities of the second-stage value function lie orthogonal to the variable axes. This allows for the partition of the feasible set in a finite number of rectangular regions such that the second-stage value function is constant over each of these regions. The idea relies, however, on the fact that the second-stage variables are purely integer. Moreover, the authors assume the technology matrix to be deterministic but remark that an extension to a random technology matrix with finite number of outcomes can be easily achieved. Computational comparison with previously proposed methods ([CT98],[SSvdV98]) show the efficiency of this new approach.
Another, very recent algorithmic idea that has been shown to be computational promising is the so called D^2 method by Sen and Higle ([SH05]). The method follows the principle of decomposition and convexification. The basic idea is to iteratively convexify the coupled master and subproblems (Disjunctive Decomposition). While the initial algorithm proposed in [SH05] should be seen as a general basis for more advanced and computationally efficient algorithms, several papers have been published since presenting first computational studies ([NT08],[NS08]) and extensions ([SS06a],[NS07],[SZH09]). Non of these publications, however, indicates if the underlying idea might in fact be a breakthrough in the study of integer TSPs (see [NS08] where an advanced version of the algorithm by Carøe and Tind ([CT98]) is shown to be computational comparable to the D^2 method). A universally efficient algorithm to solve integer TSPs is thus still to be found. It is, however, clear that for a large number of discrete scenarios the exact solution of any kind of TSP becomes intractable. In this case, an alternative is the approximation of the probability space by a (small) sample of scenarios. This approach is the most common approach in case of continuously distributed random variables and will be presented in the next section.

TSPs with continuously distributed random variables

Solving two-stage optimization problems with full recourse and continuously distributed random variables to optimality is, in general, intractable. The main difficulty consists naturally in the evaluation of the expectation of the recourse function $\mathcal{Q}$. This problem is most commonly avoided by approximating the probability distribution by a (relatively small) finite sample of scenarios. There are two common approaches.
In the first approach, the sampling is done before the actual solution process. In this case, the initial problem is approximated by another problem with only finitely many scenarios. The probably most well-known representative of this class of solution approaches is the Sample Average Approximation (SAA) method. The idea of this method is to independently draw N samples of the random parameters. In the approximated problem each of this scenario has probability $\frac{1}{N}$. Depending on the initial problem, the obtained SAA could be solved by one of the methods for TSPs with discretely distributed random variables discussed in the two previous subsections. In addition to the obvious advantages of SAA approaches like easier computation of the objective function or (sub)gradients by approximation, it can be shown that for an increasing sample size and under some mild conditions the solutions of the SAA problem converge (with probability one) to an opti-

mal solution of the initial problem. However, it is in general difficult to a priori determine a sample size sufficient for a desired approximation precision (for a recent review of results concerning SAA approaches see [Sha07]).
One of the first SAA algorithms for TSPs has been proposed in [SH98] to solve fixed recourse problems with independent random right hand side vector and technology matrix. The authors show how to obtain a subgradient of the approximated expectation of the second-stage function. They argue that due to the differentiability of the expected value function $\mathbb{E}[\mathcal{Q}(\cdot, \chi)]$ this subgradient is a "consistent estimator" of the gradient of $\mathbb{E}[\mathcal{Q}(\cdot, \chi)]$. Based on these observations the authors develop a trust region method to solve the SAA approximation. Depending on the quality of the obtained solutions new samples can be generated and the sample size be adapted during the algorithm.
For the case of a finite (but probably large) feasible set for the first-stage decision variables, Kleywegt et al. [KSH02] study the SAA approach under the aspect of convergence. They prove that, under some mild assumptions (that are always satisfied if the random variables are discretely distributed), the solutions of the SAA converge at exponential rate towards an optimal solution of the initial problem. Moreover, the authors show that the sample size needed to obtain a certain precision only grows logarithmically in the size of the first-stage problem, while most other SAA approaches need a sample size linear in the first-stage problem size (see for example [Sha07]).
Ahmed and Shapiro ([AS02]) extended the SAA approach to solve TSPs with integer recourse.
In the second approach to approximate the solution of two-stage optimization problems with continuously distributed random variables, the sampling is done during the solution process. A well-known representative of such algorithms is the stochastic (sub)gradient algorithm used to solve (sub)differentiable, continuous stochastic optimization problems (see chapters 2 and 3). The most famous of such approaches for two-stage optimization is the "Stochastic Decomposition method" by Higle and Sen ([HS91],[HS96]). This algorithm is based on the L-shaped method by Van Slyke and Wets. The optimality cuts are computed based on a continuously growing sample from the underlying probability distribution: At each iteration a new sample is added and the new as well as already existing optimality cuts are updated accordingly. In their work, the authors, however, assume complete recourse. Therefore, no feasibility cuts are needed. To the best of our knowledge, no comparable algorithm has been proposed for the integer recourse case.

4.1.4 Approximation Algorithms for Two-Stage Problems

In the second part of this chapter we present non-approximability results for some Two-Stage Knapsack problems with finitely many second-stage scenarios.
Approximation algorithms for two-stage problems can be divided in two classes: First, there are those approaches that approximate the given problem by another, slightly "simpler" problem that can be solved with common solution methods. Examples are convex approximations of the objective function (see [vdV04]) or SAA methods (see [KSH02]

and subsection 4.1.3). However, these approximations generally do not affect the $\mathcal{NP}$-hardness of the initial problem. In the second class one can find those algorithms that aim to solve $\mathcal{NP}$-hard two-stage problems approximately in polynomial time. These algorithms are generally created for one particular two-stage problem. In the following, we want to give examples of problems that admit algorithms of the second class and outline the idea of the corresponding algorithm:

1) Stochastic Single Resource Service Provision problem ($SSRP$) ([DST03], [SvdV03]):
The problem consists in deciding which services to install (first-stage decision) in order to be able to maximize, in the second stage, the profit obtained by meeting (part of) the random demands. A single resource capacity constraint restricts the resources available to install services (first-stage) and to accept demands (second-stage).
In [DST03] and [SvdV03] a finite number K of demand scenarios is assumed. The authors prove that the obtained problem is strongly $\mathcal{NP}$-hard, i.e. that no $FPTAS$ exists, unless $\mathcal{P} = \mathcal{NP}$. Furthermore, the authors show that any basic solution of the continuous relaxation has at most K interior first- and second-stage decision variables (where interior means that theses variables are neither equal to their lower, nor to their upper bound). They develop an LP round-down heuristic with approximation ratio of $\max\{\frac{1}{K+1}, \frac{1}{n}\}$ where n is the number of services available. Moreover, they show that this bound is tight. For the case where installing all services in the first stage is a feasible solution, they even proof a constant ratio.

2) Stochastic Scheduling ([SS07]): In the first-stage a provider has to decide which subset out of a given set of clients to accept to serve and which clients to defer to a different provider. In the second stage it comes to be known which clients actually make their request. The provider receives a reward for each client that he can, in the end, serve. This reward is naturally higher than the reward he obtains when deferring a client to the competition.
The authors of [SS07] assume a finite number of second-stage scenarios that is polynomial in the input size. They show that, by extending a $\frac{1}{2}$-approximation algorithm for the corresponding deterministic problem, the stochastic variant can be approximated with the same approximation ratio. The idea is to apply the approximation algorithm to each of the second-stage scenarios. This gives a dual feasible solution that is used to construct the first-stage decision. Then the second-stage solutions are computed using an adapted version of the approximation algorithm for the deterministic case.

3) Stochastic Maximum-Weight Matching ([KS06]): In this model every edge of a given graph has a deterministic first-stage and a scenario dependent second-stage weight. Edges chosen in the first stage cannot be rejected in the second stage, but additional edges can be chosen. The objective is to maximize the expected weight of the second-stage matching.
The authors of [KS06] assume a finite number of scenarios for the second-stage weight vector. They show that the obtained problem in $\mathcal{NP}$-complete and propose a very

simple approximation algorithm with performance ratio $\frac{1}{2}$: The idea is to independently compute optimal matchings for the first-stage weights as well as for each of the possible second-stage weight vectors. Then one compares the objective function value of the first-stage solution with the expectation over the obtained second-stage solutions. Their algorithm chooses therefore either the solution where the whole matching is decided in the first stage, or the solution where no edge at all is chosen in the first stage. As the deterministic weighted matching problem is polynomial time solvable, their algorithm is polynomial in the input size as well as the number of scenarios.

4) Stochastic Minimum-Weight Bipartite Matching I ([KKMU07]): Given a bipartite graph such that every edge has a deterministic first stage and a random/scenario dependent second-stage weight. In the first stage edges can be chosen. In the second stage, the edge set has to be completed in case it does not already *contain* a *perfect* matching of the bipartite graph. Contrary to the work in [KS06] presented in 3) the final set of chosen edges thus does not have to be a matching, but has to contain a maximum cardinality matching. Naturally, purchasing edges in the first stage is less expensive than in the second stage. The aim is to minimize the expected total cost.
The authors of [KKMU07] present an approximation algorithm with approximation ratio in $\mathcal{O}(n^2)$. The algorithm first solves the continuous relaxation of the problem and then decides, based on the components of the solution vector, which edges to buy in the first, and which in the second stage.

5) Stochastic Minimum-Weight Bipartite Matching II ([KKMU07]): Contrary to the previous problem 4) the second-stage edge costs are assumed to be a multiple of the first-stage edge cost (with fix coefficient) and thus known. However, one assumes the set of vertices to match to be random.
In [KKMU07] the authors show that in case of finitely many second-stage scenarios this problem is a special case of problem 4). For the general case they show that there exists an approximation ratio such that approximating the solution within this ratio is $\mathcal{NP}$-hard.

6) Stochastic Set Cover ([SS06c],[RS06],[SS06b]): The problem is the following: Given a set of n elements and m subsets. In the first stage some subsets have to be chosen at a first-stage cost. In the second stage, the subset of elements to cover is revealed. If this set is not yet covered by the subsets chosen in the first stage, additional subsets can be purchased at second-stage costs.
The authors of [SS06b] prove that if the corresponding deterministic set cover problem can be approximated with ratio ρ, then the stochastic set cover problem can be approximated with a worst case ratio of 2ρ. More precisely, the authors show that the integrality gap of the stochastic set cover problem is at most twice the integrality gap of the deterministic set cover problem. It follows that the stochastic set cover problem is $2\ln n$-approximable in case of a number of scenarios polynomial in the input size.

7) Stochastic Vertex Cover ([GPRS04],[IKMM04],[RS06],[SS06b]): In this two-stage problem one wants to cover a random subset of edges of a given graph by choosing a

minimum cardinality vertex set. All vertices have the same first-stage cost. Vertices purchased in the second stage are of a constant factor more expensive.
The authors of [IKMM04] propose a constant factor approximation based on the following idea: Vertices with a high degree have a high probability to be incident to an edge that has to be covered in the second stage. The algorithm greedily determines in the first stage a maximal matching such that the vertex degrees induced by this matching are bounded by a certain constant. Then the vertices whose induced degree attains this bound are kept as first-stage vertices.

8) Stochastic Steiner Tree/Forest ([GPRS04],[IKMM04]): Given a graph whose vertices are independently active with a given probability. The aim is to construct a Steiner Tree that connects the active vertices. In the first stage, while the active vertex set is still unknown, one can purchase edges at a first-stage cost. The second-stage costs are assumed to differ by a constant factor λ from the first-stage costs and the aim is to minimize the expected overall cost.
Among other variants of the problem, the authors of [IKMM04] study the case where the graph is metric and the probability of a vertex to be active is at least $\frac{1}{k\lambda}$ with $k \geq 1$. Then they show that a minimum spanning tree of the initial graph is a k-approximation of the optimal solution.

9) Stochastic Shortest Path with random sink ([RS06]): Given a graph with (first-stage) edge costs, a fix source node and a set of scenarios consisting of a sink node, a cost-factor (by which the first-stage edge cost is multiplied in this scenario) and a probability. In the first stage edges can be purchased at their first-stage cost. In the second stage, the sink node has come to be known and additional edges can be purchased in order to link source and sink by a minimum cost path.
The authors of [RS06] show that in any optimal solution the edges purchased in the first-stage induce a tree that contains the source. They conclude that the stochastic shortest path problem with random sink is equivalent to the so called tree-star network design problem. The tree-star network design problem consists in finding a tree that minimizes the sum of tree cost and total cost of the shortest paths from the tree to a set of demand nodes. It has been shown previously that this deterministic problem is 5-approximable, which is therefore also true for the stochastic shortest path problem with random sink.

10) Stochastic Minimum Spanning Tree ([DRS05]): The stochastic minimum spanning tree problem is generally defined as follows: Given a graph where each edge has a deterministic first- and a random second-stage cost. The decision maker can choose some edges in the first stage and has to complete its choice in the second stage to obtain a spanning tree.
In ([DRS05]) the authors study in particular two cases for the random second-stage costs: Either the number of possible outcomes in the second stage is assumed to be finite and known, or we are given a black-box from which we can draw independent

samples of the unknown distribution. The authors propose to approximate these problems using randomized algorithms: They first solve a linear relaxation. The decision of which first- and second-stage variables to set to 1 is then made by independent Bernoulli trials: Each first- and second-stage variable is independently set to one with a success probability that equals the corresponding solution component of the relaxation.
In case of finitely many, known scenarios, they obtain an approximation ratio of order $\log n + \log K$ (with high probability) where n is the number of vertices of the graph and K is the number of scenarios. The running time of the algorithm is polynomial in n and K.
In case of the black-box assumption the randomized procedure explained above is preceded by an independent drawing of a set of scenarios whose size is polynomial in n and a value λ defined as

$$\lambda := \max_{1 \leq i \leq K, e \in E} \left\{ \frac{c_e^i}{c_e^0}, \frac{c_e^0}{c_e^i} \right\}$$

Here, for an edge e c_e^0 is the first- and c_e^i the second-stage cost (in scenario i). The algorithm has an approximation ratio of order $\log n + \log \lambda$ (with high probability) and a running time polynomial in n and λ.

To conclude this subsection on approximation algorithms for two-stage problems, we want to outline some properties of two-stage problems that, in some cases, make the problem easier to approximate than in general:
First of all there is the assumption that the second-stage problem has a number of second-stage scenarios that is polynomial in the input size. This assumption allows in some cases for an approximation algorithm that mainly applies a known approximation algorithm for the deterministic variant of the problem to all second-stage scenarios.
Furthermore, in some cases assuming that the second-stage cost (resp. reward) is a fix multiple (resp. fraction) of the first stage cost (resp. reward) seems to make the problem easier.
Last but not least, Gupta et al. present in [GPRS04] a method called *boosted sampling* that makes use of an approximation algorithm for the deterministic problem to create an algorithm for the two-stage variant. Contrary to the aforementioned idea to apply such an algorithm to all second-stage scenarios, the algorithm of Gupta et al. does only require a black-box to draw samples. Their approach can, however, only be applied to problems that are subadditive. This means mainly that the union of two optimal solutions for two different scenarios gives an optimal solution for the combination of these scenarios. Moreover, the approach needs the existence of a so called *cost-sharing function*. The ratio of the obtained approximation algorithm then depends on both the ratio of the used approximation algorithm for the deterministic problem as well as the "quality" of the cost-sharing function (for details see [GPRS04]). Examples of problems that satisfy the needed conditions are stochastic steiner tree, facility location and vertex cover.

4.1.5 Solving Two-Stage Knapsack Problems

Apart from the work presented in this thesis, no references can be found that explicitly treat linear Two-Stage Knapsack problems. However, Lisser et al. ([GLLH10],[LLH09]) studied quadratic knapsack problems. They propose upper bounds by using linear and semi-definite relaxations.
Another studied problem that is close to Two-Stage Knapsack problems is the Stochastic Single Node Provision problem studied in [DST03]. The main difference is that the second-stage decision variables are continuous and that the weights uniformly equal 1. This difference allows the authors to solve the problem approximately by using an LP rounding-down heuristic with performance guarantee.
In the first part of this chapter we will propose upper and lower bounds for the Two-Stage Knapsack problem with normally distributed weights. The upper bounds are obtained by solving a continuous relaxation of the treated problem. In fact, this continuous relaxation can be shown to be a Simple Recourse Knapsack problem with chance-constraint and results from chapters 2 and 3 can be applied. The proposed lower bounds replace the exact evaluation of the objective function and are based on a study of the second-stage problem. The best possible lower bound that can be obtained by the proposed method is then determined in a B&B framework.
In the second part of this chapter we focus on the Two-Stage Knapsack problem with finite number of second-stage scenarios. We show that this problem is equivalent to the multiply-constrained (or multi-dimensional vector) knapsack problem with uniform capacities. Therefore, (non-)approximation results for the latter problem are valid for the Two-Stage Knapsack problem with a finite number of scenarios, as well. Moreover, we show that neither assuming a polynomial number of scenarios nor a linear dependency of first- and second-stage rewards make the problem easier to approximate, as it was the case for some of the problems discussed in section 4.1.4. Concerning the approximation algorithm for Two-Stage problems presented in [GPRS04], it is easy to see that Two-Stage Knapsack problems with random weights do not satisfy the condition of subadditivity, as it is not even clear what a combination of two scenarios looks like. And even in the case where two scenarios differ only slightly, the union of the corresponding optimal solutions might be highly infeasible.

4.2 The Two-Stage Knapsack Problem with normal weight distributions ($TSKP$)

4.2.1 Mathematical formulation and properties

We consider a stochastic knapsack problem of the following form: Given a knapsack with fixed weight capacity $c > 0$ as well as a set of n items. Each item has a weight that is not known in the first stage but comes to be known before the second-stage decision has to be made. Therefore, we handle the weights as random variables and assume that weight χ_i of item i is independently normally distributed with mean $\mu_i > 0$ and standard deviation

σ_i. Furthermore, each item has a fixed reward per weight unit $r_i > 0$.
In the first stage, items can be placed in the knapsack. However, if the outcome of the total weight of these items should exceed the capacity, some of these first-stage items have to be removed in the second stage. Therefore, we restrict the percentage of cases where the first-stage items lead to an overload by introducing a chance-constraint in the first stage.
At the beginning of the second stage the weights of all items are revealed. In case of an overload, items have to be removed and a penalty d has to be paid for each unit of weight that is unpacked. Our objective is thus to minimize the total penalty. If we decide to remove item i, the corresponding second-stage decision variable y_i^- is set to 1. In this case, all components of y^+ can automatically be set to 0. If the capacity constraint is respected, items can be added if and only if this does not lead to an overload (second-stage items). If we decide to add item i, we set $y_i^+ = 1$. In this second case, all components of y^- can be set to 0. It is easy to see that this model only makes sense if the reward per weight unit $\overline{r}_i$ of an item that is added in the second stage is strictly smaller than r_i. The aim is to maximize the expected total reward of the knapsack:

The Two-Stage Knapsack Problem with chance-constraint ($TSKP$)

$$
\begin{aligned}
(TSKP) \quad \max_{x\in\{0,1\}^n} \quad & \mathbb{E}\left[\sum_{i=1}^{n} r_i\chi_i x_i\right] + \mathbb{E}\left[\mathcal{Q}(x,\chi)\right] \\
\text{s.t.} \quad & \mathbb{P}\{\sum_{i=1}^{n}\chi_i x_i \le c\} \ge p. && \text{(4.4a)}\\
& \mathcal{Q}(x,\chi) = \max_{y^+,y^-\in\{0,1\}^n} \sum_{i=1}^{n}\overline{r}_i\chi_i y_i^+ - d\sum_{i=1}^{n}\chi_i y_i^- && \text{(4.4b)}\\
\text{s.t.} \quad & y_k^+ \le 1 - x_k \quad \forall\, k = 1,\dots,n, && \text{(4.4c)}\\
& y_k^- \le x_k \quad \forall\, k = 1,\dots,n, && \text{(4.4d)}\\
& y_k^+ \le \mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n}\chi_i x_i)\cdot y_k^+ \quad \forall\, k = 1,\dots,n, && \text{(4.4e)}\\
& y_k^- \le \mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n}\chi_i x_i - c)\cdot y_k^- \quad \forall\, k = 1,\dots,n, && \text{(4.4f)}\\
& \sum_{i=1}^{n}(x_i + y_i^+ - y_i^-)\chi_i \le c. && \text{(4.4g)}
\end{aligned}
$$

where $\overline{r}_i < r_i$ for all $i = 1,\dots,n$, $d \ge \max_{i\in\{1,\dots,n\}} r_i$ and $p \in [0.5, 1]$ is the prescribed probability.
Constraints (4.4e) and (4.4f) assure that items can only be added in the second stage in case the first-stage items do not lead to an overload, and they can only be removed in case of an overload. Therefore, the above setting does not allow an exchange of items in the second stage. These $2n$ nonlinear constraints could be replaced by the $n^2 - n$ following

linear constraints:

$$y_i^+ + y_j^- \leq 1 \quad \forall i, j \in \{1, \ldots, n\}, i \neq j$$

However, we chose the nonlinear problem formulation (4.4) in order to be able to rewrite the second-stage problem as a linear combination of two independent problems, one treating the second-stage decision variables y_i^+, the other the decision variables y_i^- (see subsection 4.2.1, Property 3).
Throughout, we assume that the zero-vector is not an optimal first-stage solution (i.e. that at least one item is put in the knapsack in the first stage) and that the probability that the total weight of all n items is smaller or equal than c is negligible.

Properties of the $TSKP$

The following properties are mainly used in section 4.2.3 to calculate lower bounds on the objective function value for a given first-stage solution.

Property 1: The two-stage problem (4.4) is a *relatively complete recourse* problem, i.e. for every feasible first-stage decision there exists a feasible second-stage decision.

Property 2: Given a first-stage decision and a realization of χ, solving the second-stage problem means solving a deterministic knapsack problem:
In case of an underload, the capacity of the "second-stage knapsack" equals the amount of capacity that is still unused after the first stage. The set of items that one can choose from contains all items that have not been added in the first stage.
In case of an overload, the second-stage knapsack is the initial knapsack. However, the choice can only be made among the first-stage items.
It is thus clear how to solve the second-stage problem once the first-stage decision is made and the actual item weights are revealed. However, no method to practically compute the expected second-stage solution value for a given first-stage decision is known, mainly due to the assumption that the item weights follow continuous distributions. This implies, that for a given first-stage solution, we are not able to compute the corresponding objective function value, either. Instead, one can compute a lower bound (see section 4.2.3). This lower bound can, on the one hand, be helpful in order to get an idea of the quality of the chosen decision. On the other hand, it can replace the exact evaluation of the objective function value in an appropriate framework in order to search for good overall lower bounds (see section 4.2.4).

Property 3: The random variable $\mathcal{Q}(x,\chi)$ can be written as the linear combination of two random variables $\mathcal{Q}_1(x,\chi)$ and $\mathcal{Q}_2(x,\chi)$:

$$\mathcal{Q}_1(x,\chi) = \max_{y^+\in\{0,1\}^n} \sum_{i=1}^{n} \overline{r}_i y_i^+ \chi_i \tag{4.5a}$$

$$\text{s.t.} \quad y_k^+ \leq 1 - x_k \quad \forall\, k = 1,\ldots,n, \tag{4.5b}$$

$$y_k^+ \leq \mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} x_i\chi_i) \cdot y_k^+ \quad \forall\, k = 1,\ldots,n, \tag{4.5c}$$

$$\mathbb{1}_{\mathbb{R}^+}(c - \sum x_i\chi_i) \sum_{i=1}^{n} (x_i + y_i^+)\chi_i \leq c. \tag{4.5d}$$

and

$$\mathcal{Q}_2(x,\chi) = \min_{y^-\in\{0,1\}^n} \sum_{i=1}^{n} y_i^- \chi_i \tag{4.6a}$$

$$\text{s.t.} \quad y_k^- \leq x_k \quad \forall\, k = 1,\ldots,n, \tag{4.6b}$$

$$y_k^- \leq \mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n} x_i\chi_i - c) \cdot y_k^- \quad \forall\, k = 1,\ldots,n, \tag{4.6c}$$

$$\mathbb{1}_{\mathbb{R}^+}(\sum x_i\chi_i - c) \sum_{i=1}^{n} (x_i - y_i^-)\chi_i \leq c. \tag{4.6d}$$

It follows $\mathbb{E}[\mathcal{Q}(x,\chi)] = \mathbb{E}[\mathcal{Q}_1(x,\chi)] - d \cdot \mathbb{E}[\mathcal{Q}_2(x,\chi)]$.

Property 4: Let $\widetilde{x}$ be a given feasible first-stage solution. If $\widetilde{x}$ leads to an underload, i.e. if for the realization $\hat{\chi}$ of χ we have $\sum_{i=1}^{n} \hat{\chi}_i \widetilde{x}_i \leq c$, it follows $\mathcal{Q}_2(\widetilde{x},\hat{\chi}) = 0$. On the contrary, in the case of an overload, we have $\mathcal{Q}_1(\widetilde{x},\hat{\chi}) = 0$.

4.2.2 Computing upper bounds on the optimal solution value of the $TSKP$

Given a feasible first-stage solution $\widetilde{x}$, the expectation of the second-stage solution value can be bounded (from above) by

$$\mathbb{E}[\mathcal{Q}(\widetilde{x},\chi)] \leq \overline{r}_{max} \cdot \mathbb{E}\left[[c - \sum_{i=1}^{n} \widetilde{x}_i\chi_i]^+\right] - d \cdot \mathbb{E}\left[[\sum_{i=1}^{n} \widetilde{x}_i\chi_i - c]^+\right]$$

where $[x]^+ := \max(0,x) = x \cdot \mathbb{1}_{\mathbb{R}^+}(x)$ $(x \in \mathbb{R})$ and $\overline{r}_{max} = \max\{\overline{r}_i | i \in \{1,\ldots,n\}\}$. The right hand side of the inequality is in fact the expectation of the optimal second-stage

solution value in case of continuous second-stage variables and a uniform second-stage reward $\overline{r}_{max}$. More precisely, if y_i^+ can take any value between 0 and 1, we could fill the knapsack up to the capacity in case of an underload and the expectation of $\mathcal{Q}_1(\widetilde{x}, \chi)$ would be at most $\overline{r}_{max} \cdot \mathbb{E}\left[[c - \sum_{i=1}^n \widetilde{x}_i \chi_i]^+\right]$. And if $y_i^- \in [0,1]$, we would have to pay exactly d times the overload as additional costs in case the capacity is not respected, i.e. $\mathbb{E}[\mathcal{Q}_2(\widetilde{x}, \chi)] = d \cdot \mathbb{E}\left[[\sum_{i=1}^n \widetilde{x}_i \chi_i - c]^+\right]$. An upper bound on the optimal solution value of the $TSKP$ (4.4) is thus given by the optimal solution value of the following simple recourse problem:

$$(SRKP) \quad \max_{x \in \{0,1\}^n \setminus \{\mathbb{O}_n\}} \quad \mathbb{E}\left[\sum_{i=1}^n r_i x_i \chi_i\right] + \overline{r}_{max} \cdot \mathbb{E}\left[[c - \sum_{i=1}^n x_i \chi_i]^+\right] - d \cdot \mathbb{E}\left[[\sum_{i=1}^n x_i \chi_i - c]^+\right]$$

$$\text{s.t.} \quad \mathbb{P}\{\sum_{i=1}^n x_i \chi_i \leq c\} \geq p. \tag{4.7a}$$

We can solve the $SRKP$ (4.7) approximately by the methods proposed in chapters 2 and 3: The basic idea is to solve the corresponding Lagrangian relaxation using a Stochastic Arrow-Hurwicz algorithm. In chapter 2 it is shown how to approximate the gradient of the function $\mathbb{E}\left[[\sum_{i=1}^n x_i \chi_i - c]^+\right]$ using the Approximation by Convolution method. Alternatives to this approach are to use a stochastic subgradient method (see the conclusion of chapter 2) as well as the Integration by Parts and Finite Differences approaches presented in chapter 3.
In order to handle the chance-constraint using a Lagrange multiplier, let us rewrite the constraint as follows:

$$p - \mathbb{P}\{\sum_{i=1}^n x_i \chi_i \leq c\} \leq 0$$

Unfortunately, the left hand side of this inequality is generally not convex. However, this can be achieved by further reformulating the constraint (see e.g. [LVBL98]):

$$p - \mathbb{P}\{\sum_{i=1}^n x_i \chi_i \leq c\} \leq 0 \Longleftrightarrow \sum_{i=1}^n x_i \chi_i + F^{-1}(p) \|\Sigma^{1/2} x\| - c \leq 0 \tag{4.8}$$

Here Σ denotes the diagonal covariance matrix of the random vector χ. As $p \geq 0.5$ and therefore $F^{-1}(p) \geq 0$, the obtained inequality is a Second Order Cone constraint with a convex left hand side function whose gradient on $\{0,1\}^n \setminus \{\mathbb{O}_n\}$ can easily be computed. By replacing the initial chance- by this Second Order Cone constraint, we thus get a well defined, convex Lagrangian problem. The problem has an expected value objective function and can be solved using a stochastic gradient algorithm as described in chapters 2.7 and 3.18.

4.2.3 Computing lower bounds on the optimal solution value of the $TSKP$

In this section we propose lower bounds on the overall solution value for a given fixed first-stage decision $\widetilde{x}$. More precisely, we bound $\mathbb{E}[Q_1(\widetilde{x}, \chi)]$ (from below) as well as $\mathbb{E}[Q_2(\widetilde{x}, \chi)]$

(from above) in order to get a lower bound on the expected second-stage solution value $\mathbb{E}[Q(\widetilde{x}, \chi)]$.
Let in the following S denote the index set of those items that have been chosen in the first stage and $\overline{S}$ its complement (i.e. $S = \{i \in \{1, \dots n\} | \widetilde{x}_i = 1\}$ and $\overline{S} = \{i \in \{1, \dots n\} | \widetilde{x}_i = 0\}$) and define $\overline{r}_{min} = \min\{\overline{r}_i | i \in \{1, \dots, n\}\}$.

Lower bound on the expectation of $\mathcal{Q}_1$

We first show the following:

Proposition 4.3.

$$\mathbb{E}[\mathcal{Q}_1(\tilde{x}, \chi)] > \overline{r}_{min} \cdot \left(\mathbb{E}\left[[c - \sum_{i=1}^{n} \tilde{x}_i \chi_i]^+ \right] - \widetilde{p} \cdot \mathbb{E}\left[\max_{i \in \overline{S}} \chi_i \right] \right)$$

where $\widetilde{p} = \mathbb{P}\{\sum_{i=1}^{n} \tilde{x}_i \chi_i \leq c\}$ *denotes the probability of an underload.*

Proof. Let us rewrite the optimal solution of the (partial) second-stage problem $\mathcal{Q}_1(\tilde{x}, \hat{\chi})$ for a realization $\hat{\chi}$ of χ as

$$\mathcal{Q}_1(\tilde{x}, \hat{\chi}) = \sum_{i=1}^{n} \overline{r}_i (y^+)_i^* \hat{\chi}_i \geq \overline{r}_{min} \cdot \sum_{i=1}^{n} (y^+)_i^* \hat{\chi}_i \tag{4.9}$$

$$= \overline{r}_{min} \cdot \left((c - \sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i) + \left(\sum_{i=1}^{n} (y^+)_i^* \hat{\chi}_i - (c - \sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i) \right) \right) \tag{4.10}$$

where $(y^+)^*$ is the (unknown) optimal second-stage solution vector that depends on $\tilde{x}$ as well as $\hat{\chi}$.
In case of an underload, $(c - \sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i) - \sum_{i=1}^{n} (y^+)_i^* \hat{\chi}_i = c - \sum_{i=1}^{n} (\tilde{x}_i + (y^+)_i^*) \hat{\chi}_i$ is the difference between the capacity and the final weight of the knapsack. Note that this quantity figures negatively in (4.10). This amount can be bounded independently of the second-stage solution $(y^+)^*$, as in the worst case we have to keep a weight of $\max_{i \in \overline{S}} \hat{\chi}_i - \epsilon$ of the knapsack unused (where $\epsilon > 0$):

$$(c - \sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i) - \sum_{i=1}^{n} (y^+)_i^* \hat{\chi}_i < \max_{i \in \overline{S}} \hat{\chi}_i \tag{4.11}$$

It follows

$$\mathcal{Q}_1(\tilde{x}, \hat{\chi}) > \overline{r}_{min} \left((c - \sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i) - \max_{i \in \overline{S}} \hat{\chi}_i \right) \tag{4.12}$$

Whenever we are in the case on an overload, $\mathcal{Q}_1$ is zero. We thus get:

$$\mathbb{E}[\mathcal{Q}_1(\tilde{x}, \chi)] > \overline{r}_{min} \left(\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) \left((c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) - \max_{i \in \overline{S}} \chi_i \right) \right] \right)$$

$$= \overline{r}_{min} \left(\mathbb{E}\left[[c - \sum_{i=1}^{n} \tilde{x}_i \chi_i]^+ \right] - \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) \cdot \max_{i \in \overline{S}} \chi_i \right] \right)$$

In the case of independently normally distributed items, $\mathbb{E}\left[[c - \sum_{i=1}^{n} \tilde{x}_i \chi_i]^+\right]$ has a deterministic equivalent formulation that is a linear combination of the density function and CDF of the standard normal distribution. It can therefore be computed exactly (see e.g. [CB98] or chapter 2). So let us see how to further bound $\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) \cdot \max_{i \in \overline{S}} \chi_i\right]$:

Claim 4.4.

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) \cdot \max_{i \in \overline{S}} \chi_i\right] \leq \widetilde{p} \cdot \mathbb{E}[\max_{i \in \overline{S}} \chi_i]$$

Proof of the claim: It is easy to see that

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) \cdot \max_{i \in \overline{S}} \chi_i\right] + \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n} \tilde{x}_i \chi_i - c) \cdot \max_{i \in \overline{S}} \chi_i\right] = \mathbb{E}\left[\max_{i \in \overline{S}} \chi_i\right]$$

so either

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) \cdot \max_{i \in \overline{S}} \chi_i\right] \leq \widetilde{p} \cdot \mathbb{E}[\max_{i \in \overline{S}} \chi_i]$$

or

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n} \tilde{x}_i \chi_i - c) \cdot \max_{i \in \overline{S}} \chi_i\right] \leq (1 - \widetilde{p}) \cdot \mathbb{E}[\max_{i \in \overline{S}} \chi_i]$$

and if one of the two inequalities is satisfied with equality, the second would be satisfied with equality, as well. We further have

$$\mathbb{E}[\max_{i \in \overline{S}} \chi_i | \sum_{i=1}^{n} \tilde{x}_i \chi_i \leq c] \leq \mathbb{E}[\max_{i \in \overline{S}} \chi_i | \sum_{i=1}^{n} \tilde{x}_i \chi_i \geq c]$$

which excludes the case that

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n} \tilde{x}_i \chi_i - c) \cdot \max_{i \in \overline{S}} \chi_i\right] < (1 - \widetilde{p}) \cdot \mathbb{E}[\max_{i \in \overline{S}} \chi_i] \quad \text{and}$$

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) \cdot \max_{i \in \overline{S}} \chi_i\right] > \widetilde{p} \cdot \mathbb{E}[\max_{i \in \overline{S}} \chi_i]$$

Thus,

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) \cdot \max_{i \in \overline{S}} \chi_i\right] \leq \widetilde{p} \cdot \mathbb{E}[\max_{i \in \overline{S}} \chi_i]$$

q.e.d.

This concludes the proof. □

Unfortunately, (even) in the case of independently normally distributed random variables, $\mathbb{E}[\max_{i\in\overline{S}}\chi_i]$ is not easily computable. We thus bound it by discretization of the probability space of the random variable $\max_{i\in\overline{S}}\chi_i$:
Let us define a new random variable $\chi^{\overline{S}}_{max} := \max_{i\in\overline{S}}\chi_i$. Let Φ_i be the CDF of χ_i, and $\Phi^{\overline{S}}_{max}$ the CDF of $\chi^{\overline{S}}_{max}$. Then one can easily show that $\Phi^{\overline{S}}_{max} = \prod_{i\in\overline{S}}\Phi_i$.
Furthermore, if there exists a $\beta < \infty$ such that $\mathbb{P}\{\chi^{\overline{S}}_{max} \in (-\infty, \beta]\} = 1$, we can bound $\mathbb{E}\left[\chi^{\overline{S}}_{max}\right]$ by splitting the interval $(-\infty, \beta]$ into K disjoint intervals (K scenarios) $(\alpha_k, \beta_k]$, $k = 1, \ldots, K$, and it follows

$$\mathbb{E}\left[\chi^{\overline{S}}_{max}\right] \leq \sum_{k=1}^{K} \beta_k \mathbb{P}\{\max_{i\in\overline{S}}\chi_i \in (\alpha_k, \beta_k]\} = \sum_{k=1}^{K} \beta_k(\Phi^{\overline{S}}_{max}(\beta_k) - \Phi^{\overline{S}}_{max}(\alpha_k))$$

Unfortunately, in the case of normally distributed weights, (theoretically) no such β exists. However, as for every ϵ there exists a β such that $\mathbb{P}\{\exists i \in \overline{S}|\chi_i \geq \beta\} \leq \epsilon$, we can approximate the upper bound on $\mathbb{E}\left[\chi^{\overline{S}}_{max}\right]$ by defining β in such a way that $\mathbb{P}\{\exists i \in \overline{S}|\chi_i \geq \beta\}$ is (sufficiently) small. From a numerical point of view, ϵ can even be chosen to be zero as we generally deal with distributions that have small variances (compared to the mean) and whose density functions go to zero very fast. So we finally get:

Corollary 4.5. *Let α_k, β_k $(k = 1, \ldots, K)$ be defined as above. Then*

$$\mathbb{E}[\mathcal{Q}_1(\tilde{x}, \chi)] > \overline{r}_{min}\left(\mathbb{E}\left[[c - \sum_{i=1}^{n}\tilde{x}_i\chi_i]^+\right] - \widetilde{p}\cdot\sum_{k=1}^{K}\beta_k(\Phi^{\overline{S}}_{max}(\beta_k) - \Phi^{\overline{S}}_{max}(\alpha_k))\right)$$

where $\widetilde{p} = \mathbb{P}\{\sum_{i=1}^{n}\tilde{x}_i\chi_i \leq c\}$ denotes the probability of an underload.

□

Remark 4.6. *As the item weights have been assumed to be independently normally distributed, $X := \sum_{i=1}^{n}\tilde{x}_i\chi_i$ is normally distributed with mean $\sum_{i=1}^{n}\tilde{x}_i\mu_i$ and standard deviation $\sqrt{\sum_{i=1}^{n}\tilde{x}_i^2\sigma_i^2}$. We are thus able to compute $\widetilde{p} = \mathbb{P}\{X \leq c\}$ as well as $\mathbb{E}\left[[c - \sum_{i=1}^{n}\tilde{x}_i\chi_i]^+\right] = \mathbb{E}\left[[c - X]^+\right]$.*

Upper bound on the expectation of $\mathcal{Q}_2$

Similar to the approach presented in the previous subsection, we first bound $\mathbb{E}[Q_2(\tilde{x}, \chi)]$ depending on the expected maximum first-stage item weight $\mathbb{E}[\max_{i\in S}\chi_i]$. However, in the case of an overload no claim such as Claim (4.4) can be proved.

Proposition 4.7.

$$\mathbb{E}[\mathcal{Q}_2(\tilde{x}, \chi)] < \mathbb{E}\left[[\sum_{i=1}^{n}\tilde{x}_i\chi_i - c]^+\right] + \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n}\tilde{x}_i\chi_i - c)\cdot\max_{i\in S}\chi_i\right]$$

Proof. The proof is similar to the proof of Proposition 4.3: For a realization $\hat{\chi}$ of χ, let $(y^-)^* = (y^-)^*(\tilde{x}, \hat{\chi})$ be the corresponding optimal second-stage solution. It follows

$$\begin{aligned}\mathcal{Q}_2(\tilde{x}, \hat{\chi}) &= \sum_{i=1}^{n} \hat{\chi}_i (y^-)_i^* \\ &= (\sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i - c) + \left(\sum_{i=1}^{n} (y^-)_i^* \hat{\chi}_i - (\sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i - c) \right)\end{aligned}$$

$\sum_{i=1}^{n} (y^-)_i^* \hat{\chi}_i - (\sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i - c)$ is the amount of weight that we remove from the knapsack in addition to the overweight. This amount can be once more bounded independently of the second-stage solution $(y^-)^*$, as in the worst case the knapsack weight might fall $(\max_{i \in S} \hat{\chi}_i - \epsilon)$ under the capacity due to the removal of items in the second stage (where $\epsilon > 0$). Thus

$$\sum_{i=1}^{n} (y^-)_i^* \hat{\chi}_i - (\sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i - c) < \max_{i \in S} \hat{\chi}_i \tag{4.13}$$

and it follows

$$\mathcal{Q}_2(\tilde{x}, \hat{\chi}) < (\sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i - c) + \max_{i \in S} \hat{\chi}_i \tag{4.14}$$

Whenever we are in the case of an underload, $\mathcal{Q}_2$ is zero:

$$\mathbb{E}[\mathcal{Q}_2(\tilde{x}, \chi)] < \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n} \tilde{x}_i \chi_i - c) \left((\sum_{i=1}^{n} \tilde{x}_i \chi_i - c) + \max_{i \in S} \chi_i \right) \right]$$

□

We finally bound $\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n} \tilde{x}_i \chi_i - c) \cdot \max_{i \in S} \chi_i\right]$ by discretization in the same way as $\mathbb{E}\left[\max_{i \in \overline{S}} \chi_i\right]$ has been bounded in subsection 4.2.3: under the condition that $\sum_{i \in S} \chi_i \geq c$, we have $\max_{i \in S} \chi_i \geq c/|S|$. So let α_k^c, β_k^c $(k = 1, \ldots, K^c)$ such that $\cup_{k=1}^{K^c} (\alpha_k^c, \beta_k^c\,] = (\frac{c}{|S|}, \beta\,]$ (if $\frac{c}{|S|} \geq \beta$ we define $K^c = 0$). It follows

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n} \tilde{x}_i \chi_i - c) \cdot \max_{i \in S} \chi_i \right] \leq \sum_{k=1}^{K^c} \beta_k^c (\Phi_{max}^S(\beta_k^c) - \Phi_{max}^S(\alpha_k^c))$$

which leads to the following corollary:

Corollary 4.8. *Let α_k^c, β_k^c $(k = 1, \ldots, K^c)$ be defined as above. Then*

$$\mathbb{E}[\mathcal{Q}_2(\tilde{x}, \chi)] < \mathbb{E}\left[[\sum_{i=1}^{n} \tilde{x}_i \chi_i - c]^+ \right] + \sum_{k=1}^{K^c} \beta_k^c (\Phi_{max}^S(\beta_k^c) - \Phi_{max}^S(\alpha_k^c))$$

□

Lower bound on the overall solution value

Combining the results from the previous subsections (corollaries 4.5 and 4.8) and as

$$\mathbb{E}\left[[c - \sum_{i=1}^{n} x_i\chi_i]^+\right] = \mathbb{E}\left[[\sum_{i=1}^{n} x_i\chi_i - c]^+\right] - \mathbb{E}\left[\sum_{i=1}^{n} x_i\chi_i - c\right]$$

we get the following lower bound on the overall solution for a given feasible first-stage vector $\tilde{x}$:

Proposition 4.9. *Let α_k,β_k, $(k = 1, \ldots, K)$, α_k^c,β_k^c $(k = 1, \ldots, K^c)$ be defined as in the previous two subsections. Let $\tilde{x}$ be a feasible first-stage solution. Then the following lower bound on the corresponding overall solution value holds:*

$$\mathbb{E}\left[\sum_{i=1}^{n} r_i\chi_i\tilde{x}_i\right] + \mathbb{E}\left[\mathcal{Q}(\tilde{x}, \chi)\right] > \sum_{i=1}^{n}(r_i - \overline{r}_{min})\mu_i\tilde{x}_i + (\overline{r}_{min} - d) \cdot \mathbb{E}\left[[\sum_{i=1}^{n} \tilde{x}_i\chi_i - c]^+\right] + \overline{r}_{min} \cdot \left(c - \widetilde{p} \cdot \sum_{k=1}^{K} \beta_k(\Phi^{\overline{S}}_{max}(\beta_k) - \Phi^{\overline{S}}_{max}(\alpha_k))\right) - d \cdot \sum_{k=1}^{K^c} \beta_k^c(\Phi^{S}_{max}(\beta_k^c) - \Phi^{S}_{max}(\alpha_k^c))$$

where $\widetilde{p} = \mathbb{P}\{\sum_{i=1}^{n} \tilde{x}_i\chi_i \leq c\}$ denotes the probability of an underload.

□

Lower bound on the optimal solution value of the $TSKP$ in the case of similar items

In order to tighten the lower bounds, we make the assumption that the item weights are similar in the following sense:

Assumption 4.10. *The probability of an item to have more than twice the size of another item is zero.*

In the case of normally distributed items Assumption 4.10 is theoretically never true. However, in many practical problems the probability that the weight of an item varies by 100% is practically zero. Hence, whenever we treat items whose means are similar, Assumption 4.10 is practically true.

Lower bound on the expectation of $\mathcal{Q}_1$ in the case of similar items

In the previous sections we bounded the optimal second-stage solution value depending on the expectation of the maximum item weight among all the items that had not been added in the first stage. In the case of similar items, this lower bound can be improved due to the following lemma:

Lemma 4.11. *Let $\overline{r}_i = \overline{r}_j$ for all $i, j \in \{1, \dots, n\}$. Then, under Assumption 4.10, it follows*

$$(c - \sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i) - \sum_{i=1}^{n} (y^+)_i^* \hat{\chi}_i < \min_{i \in \overline{S}} \hat{\chi}_i$$

Proof. If the first-stage items lead to an overload, the left hand side of the inequality is negative and the lemma is true for positive item weights. So let us suppose that we are in the case of an underload. Let T be the set of indexes such that $i \in T$ if and only if $\tilde{x}_i = (y^+)_i^* = 0$.

It is clear that $(c - \sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i) - \sum_{i=1}^{n} (y^+)_i^* \hat{\chi}_i < \min_{i \in T} \hat{\chi}_i$ as otherwise we could have added another item in the second stage. It follows that if (at least) one of the items with minimum weight in $\overline{S}$ has not been added in the second stage, we are done, as in this case $\min_{i \in T} \hat{\chi}_i = \min_{i \in \overline{S}} \hat{\chi}_i$. So let $\min_{i \in \overline{S}} \hat{\chi}_i < \min_{i \in T} \hat{\chi}_i$ and define $j := \arg\min_{i \in \overline{S}} \hat{\chi}_i$ as well as $j' := \arg\min_{i \in T} \hat{\chi}_i$. It follows that $(c - \sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i) - \sum_{i=1}^{n} (y^+)_i^* \hat{\chi}_i + \hat{\chi}_j < \hat{\chi}_{j'}$ as otherwise we could have added item j' instead of item j which would have increased the optimal objective function (here we use the assumption that $\overline{r}_i = \overline{r}_j$ for all $i, j \in \{1, \dots, n\}$). We consequently get

$$(c - \sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i) - \sum_{i=1}^{n} (y^+)_i^* \hat{\chi}_i < \hat{\chi}_{j'} - \min_{i \in \overline{S}} \hat{\chi}_i \leq 2 \min_{i \in \overline{S}} \hat{\chi}_i - \min_{i \in \overline{S}} \hat{\chi}_i = \min_{i \in \overline{S}} \hat{\chi}_i$$

□

Using this lemma we can prove the following lower bound on $\mathbb{E}[\mathcal{Q}_1(\tilde{x}, \chi)]$ in case of similar items:

Proposition 4.12. *Under Assumption 4.10, the following holds:*

$$\mathbb{E}[\mathcal{Q}_1(\tilde{x}, \chi)] > \overline{r}_{min} \cdot \left(\mathbb{E}\left[[c - \sum_{i=1}^{n} \tilde{x}_i \chi_i]^+ \right] - \widetilde{p} \cdot \frac{\sum_{i \in \overline{S}} \mu_i}{|\overline{S}|} \right)$$

where $\widetilde{p}$ denotes the probability of an underload.

Proof. First of all we replace the second-stage rewards $\overline{r}_i$ $(i = 1, \dots, n)$ by $\overline{r}_{min}$. This replacement allows us to apply Lemma 4.11 and we get:

$$\begin{aligned} \mathbb{E}[\mathcal{Q}_1(\tilde{x}, \chi)] &> \overline{r}_{min} \cdot \left(\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) \left((c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) - \min_{i \in \overline{S}} \chi_i \right) \right] \right) \\ &= \overline{r}_{min} \cdot \left(\mathbb{E}\left[[c - \sum_{i=1}^{n} \tilde{x}_i \chi_i]^+ \right] - \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) \cdot \min_{i \in \overline{S}} \chi_i \right] \right) \end{aligned}$$

The following claim is similar to Claim 4.4 (and so is its proof):

Claim 4.13. *Under Assumption 4.10, the following holds:*

$$\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(c - \sum_{i=1}^{n} \tilde{x}_i \chi_i) \cdot \min_{i \in \overline{S}} \chi_i\right] \leq \widetilde{p} \cdot \mathbb{E}\left[\min_{i \in \overline{S}} \chi_i\right]$$

where $\widetilde{p}$ denotes the probability of an underload.

□

It follows:

$$\mathbb{E}[\mathcal{Q}_1(\tilde{x}, \chi)] > \overline{r}_{min} \cdot \left(\mathbb{E}\left[[c - \sum_{i=1}^{n} \tilde{x}_i \chi_i]^+\right] - \widetilde{p} \cdot \mathbb{E}\left[\min_{i \in \overline{S}} \chi_i\right]\right)$$

For any set $\overline{S} \subseteq \{1, \ldots, n\}$, $\mathbb{E}\left[\min_{i \in \overline{S}} \chi_i\right]$ can easily be bounded, as the weight of the lightest item among all items in $\overline{S}$ is at most $1/|\overline{S}|$ of the total weight of the items in $\overline{S}$:

$$\min_{i \in \overline{S}} \chi_i \leq \frac{\sum_{i \in \overline{S}} \chi_i}{|\overline{S}|} \tag{4.15}$$

It follows

$$\mathbb{E}[\min_{i \in \overline{S}} \chi_i] \leq \frac{\mathbb{E}[\sum_{i \in \overline{S}} \chi_i]}{|\overline{S}|} = \frac{\sum_{i \in \overline{S}} \mu_i}{|\overline{S}|}$$

which ends the proof. □

The previous proposition can be generalized as follows:

Proposition 4.14. *If the probability for an item to have more than k times the size of another item is zero ($k \geq 2$), it follows*

$$\mathbb{E}[\mathcal{Q}_1(\tilde{x}, \chi)] > \overline{r}_{min} \cdot \left(\mathbb{E}\left[[c - \sum_{i=1}^{n} \tilde{x}_i \chi_i]^+\right] - \widetilde{p} \cdot (k-1) \cdot \frac{\sum_{i \in \overline{S}} \mu_i}{|\overline{S}|}\right)$$

where $\widetilde{p}$ denotes the probability of an underload.

□

Upper bound on the expectation of $\mathcal{Q}_2$ in the case of similar items

The following lemma is the counterpart of Lemma 4.11 in the underload case. The proof is similar:

Lemma 4.15. *Under Assumption 4.10, the following holds:*

$$\sum_{i=1}^{n} (y^-)_i^* \hat{\chi}_i - (\sum_{i=1}^{n} \tilde{x}_i \hat{\chi}_i - c) < \min_{i \in S} \hat{\chi}_i$$

□

Proposition 4.16. *Under Assumption 4.10, the following holds:*

$$\mathbb{E}[\mathcal{Q}_2(\tilde{x},\chi)] < (1+\frac{1}{|S|})\mathbb{E}\left[[\sum_{i=1}^{n}\tilde{x}_i\chi_i - c]^+\right] + \frac{c\cdot(1-\tilde{p})}{|S|}$$

where $\widetilde{p}$ denotes the probability of an underload.

Proof. Due to Lemma 4.15 we have

$$\mathbb{E}[\mathcal{Q}_2(\tilde{x},\chi)] < \mathbb{E}\left[[\sum_{i=1}^{n}\tilde{x}_i\chi_i - c]^+\right] + \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n}\tilde{x}_i\chi_i - c)\cdot\min_{i\in S}\chi_i\right]$$

Using inequality (4.15) we further have

$$\begin{aligned}\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n}\tilde{x}_i\chi_i - c)\cdot\min_{i\in S}\chi_i\right] &\leq \mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n}\tilde{x}_i\chi_i - c)\cdot\frac{\sum_{i=1}^{n}\tilde{x}_i\chi_i}{|S|}\right]\\ &= \frac{1}{|S|}\left(\mathbb{E}\left[[\sum_{i=1}^{n}\tilde{x}_i\chi_i - c]^+\right] + c\cdot(1-\tilde{p})\right)\end{aligned}$$

□

Proposition 4.16 can be generalized as follows:

Proposition 4.17. *If the probability for an item to have more than k times the size of another item is zero (for $k \geq 2$), it follows*

$$\mathbb{E}[\mathcal{Q}_2(\tilde{x},\chi)] < (1+\frac{k-1}{|S|})\mathbb{E}\left[[\sum_{i=1}^{n}\tilde{x}_i\chi_i - c]^+\right] + \frac{c\cdot(1-\tilde{p})\cdot(k-1)}{|S|}$$

where $\widetilde{p}$ denotes the probability of an underload. □

Lower bound on the expectation of $\mathcal{Q}$ in the case of similar items

Combining the previous results from this subsection, we get the following lower bounds on the expectation of the optimal second-stage solution value $Q(x,\chi)$:
From Propositions 4.12 and 4.16 it follows:

Proposition 4.18. *Let $\tilde{x}$ be a feasible first-stage solution. Under Assumption 4.10, the following upper bound on the corresponding overall solution value holds*

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{n}r_i\chi_i\tilde{x}_i\right] + \mathbb{E}\left[\mathcal{Q}(\tilde{x},\chi)\right] \geq \sum_{i=1}^{n}r_i\mu_i\tilde{x}_i + \overline{r}_{min}\cdot\left(\mathbb{E}\left[[c-\sum_{i=1}^{n}\tilde{x}_i\chi_i]^+\right] - \widetilde{p}\cdot\frac{\sum_{i\in\overline{S}}\mu_i}{|\overline{S}|}\right)\\ - d\cdot\left(\left(1+\frac{1}{|S|}\right)\mathbb{E}\left[[\sum_{i=1}^{n}\tilde{x}_i\chi_i - c]^+\right] + \frac{c\cdot(1-\tilde{p})}{|S|}\right)\end{aligned}$$

where $\widetilde{p}$ denotes the probability of an underload. □

From Propositions 4.14 and 4.17 we get:

Proposition 4.19. *Let $\tilde{x}$ be a feasible first-stage solution. If the probability for an item to have more than k times the size of another item is zero (for $k \geq 2$), the following upper bound on the corresponding overall solution value holds:*

$$\mathbb{E}\left[\sum_{i=1}^{n} r_i \chi_i \tilde{x}_i\right] + \mathbb{E}\left[\mathcal{Q}(\tilde{x}, \chi)\right] \geq \sum_{i=1}^{n} r_i \mu_i \tilde{x}_i + \overline{r}_{min} \cdot \left(\mathbb{E}\left[[c - \sum_{i=1}^{n} \tilde{x}_i \chi_i]^+ \right] - \widetilde{p} \cdot (k-1) \cdot \frac{\sum_{i \in \overline{S}} \mu_i}{|\overline{S}|} \right)$$
$$- d \cdot \left(\left(1 + \frac{k-1}{|S|}\right) \mathbb{E}\left[[\sum_{i=1}^{n} \tilde{x}_i \chi_i - c]^+ \right] + \frac{c \cdot (1 - \tilde{p}) \cdot (k-1)}{|S|} \right)$$

where $\widetilde{p}$ denotes the probability of an underload.

□

4.2.4 Branch-and-Bound Algorithm

In the previous section we proposed methods to calculate a lower bound on the objective function value for a given first-stage solution $\widetilde{x}$. However, for a bad choice of $\widetilde{x}$ this lower bound might be far from the optimal overall solution value.
In order to search the first-stage solution space for a best possible lower bound on the overall problem (4.4), we choose a B&B framework involving the bounds proposed above. Concerning the computation of the upper bounds that serve to evaluate subtrees, we apply the following policy:
In [CB98], Cohn and Barnhart solve the $SRKP$ (4.7) without the chance-constraint (4.7a), i.e. the basic *Simple Recourse Knapsack problem (basic SRKP)* (or *Static Stochastic Knapsack problem*). They propose very simple and fast computable upper bounds for the basic $SRKP$ that serve them to prune valueless subtrees in a B&B framework (see also the numerical results of chapter 2). We use their upper bounds for the same purpose as relaxing the chance-constraint gives us an upper bound on the $SRKP$ and thus also on the initial problem. If their upper bounds are not tight enough to prune the subtree that we are currently evaluating, we solve the $SRKP$ as proposed in section 4.2.2.
As described in [CB98] and in chapter 2 the first step of the B&B algorithm is to rank the items following their value of r_i/σ_i^2 which defines the binary search tree. The full framework is presented in Algorithm 4.2.1.

4.2.5 Numerical results

The aim of this section is twofold: First of all we want to give the reader an idea of the relative gap between the proposed upper bound and the best possible lower bound that one can get by using the lower bounds proposed in section 4.2.3. The latter is obtained by applying the B&B algorithm given in the previous section. The second aim is to analyze the quality of the B&B algorithm itself, i.e. the CPU-time it needs to find the best

Branch-and-Bound Algorithm

1. Rank the items by their value of r_i/σ_i^2. This ranking defines the binary tree with the highest ranked item at the root.
2. Set $INF \leftarrow 0$. Define L as the set of first-stage decisions to explore. Set $L \leftarrow \emptyset$.
3. Plunge the tree to find the first first-stage decision x: Beginning at the root of the tree, add the current item if and only if the lower bound on the overall objective function value increases and the chance-constraint is still satisfied after adding the item. Calculate a lower bound $LB(x)$ on the first-stage objective function. Set $INF \leftarrow LB(x)$. Add the found first-stage decision x to L. Set the associated upper bound $UB(x)$ to infinity.
4. **If** $L = \emptyset$, go to step 8.
   Else select $x \in L$ such that $x = \arg\max_{x \in L} LB(x)$. Go to step 5.
5. **If** $UB(x) > INF$, go to step 6.
   Else remove x from L. Go to step 4.
6. **If** there is no accepted item left in the selected first-stage decision x such that the subtree defined by rejecting this item has not already been plunged or rejected, remove x from L. Go back to step 4.
   Else, following our ranking, choose the first accepted item such that the subtree defined by rejecting this item has not already been plunged or rejected. Calculate an upper bound UB on this subtree. Go to step 7.
7. **If** $UB \leq INF$, reject the subtree, go to step 6.
   Else plunge the subtree as described in step 3 to find a new first-stage decision $\widetilde{x}$. Calculate $LB(\widetilde{x})$ and set $UB(\widetilde{x}) \leftarrow UB$. Add $\widetilde{x}$ together with the values $UB(\widetilde{x})$ and $LB(\widetilde{x})$ to L. If $LB(\widetilde{x}) > INF$, set $INF \leftarrow LB(\widetilde{x})$.
   Go to step 4.
8. The current value INF is the best lower bound for problem (4.4) that can be obtained by the proposed method(s).

Algorithm 4.2.1

possible lower bound and the number of upper bounds it has to calculate. Obviously, these subjects are linked as the better the quality of the upper and lower bounds, the more subtrees are pruned during the B&B algorithm and the less upper bounds have to be computed.
We tested our method on 50 randomly generated instances for each case and each dimension. In the general case the weight means were generated uniformly between 100 and 1000. For the case of similar items, we chose a normal distribution with mean 225 and variance 25 to generate the weight means (see [CB98]). We tested our method on the problem variant with chance-constraint in the first stage and fixed the prescribed probability p to 95%. The tests were carried out on an Intel PC with 2MB RAM and we allowed a maximum average CPU-time of $1h$. In the following tables, * indicates that an average CPU-time of more than $1h$ would have been needed to treat all instances of this

dimension.
The following analysis of the numerical tests is divided in three parts: In the first part, we study the case where, in the second stage, items can be added only. When solving the instances with similar items we compare the general lower bounds with the lower bounds proposed particularly for this case. In the second part, we study the problem variant where items can be rejected only. The last part contains the numerical results for the problem setting where items can be added in case of an underload and have to be removed in case of an overload.
The following tables present for each set of instances the minimal gap (between upper and best lower bound found; second column) and the maximal gap (third column) recorded as well as the average gap (fourth column) and the standard deviation of the gaps (fifth column). Moreover, the tables contain the average CPU-time needed by the B&B algorithm (sixth column) as well as the average number of nodes considered during the B&B algorithm (i.e. the number of computed upper bounds per run; seventh column).
For a detailed study of the stochastic gradient algorithm used to solve the continuous problem (4.7) see the numerical results of chapters 2 and 3.

Items can only be added in the second stage

	General items					
n	**Minimal Gap**	**Maximal Gap**	**Average Gap**	**Standard Deviat.**	**CPU-time (sec)**	**# considered nodes**
15	3.18%	31.09%	11.41%	4.98%	0.1	29
20	2.43%	21.51%	9.35%	4.94%	0.16	38
30	2.12%	14.87%	7.02%	3.48%	0.40	70
50	1.48%	16.50%	6.82%	3.36%	55.00	6693
75	1.08%	11.02%	6.49%	2.14%	12.36	970
100	2.07%	14.29%	6.69%	2.41%	59.00	3501
150	2.30%	10.99%	6.61%	2.06%	928.02	36807
250	*	*	*	*	*	*

Table 4.1: Numerical results for the $TSKP$: Items can only be added in the second stage (general items)

Table 4.1 shows the results of our method for non-similar items. First of all we observe that up to 30 items the gaps improve visibly while their diminution is less obvious for a greater number of items.
Furthermore, we notice that the number of nodes considered during the B&B algorithm depends not only on n, but also on the hardness of each particular instance: While for instances with 75 and 100 items the B&B algorithm considers on average 970 and 3501 nodes, respectively, it computes on average 6693 upper bounds to solve the instances of

size 50 due to one single, hard instance (230370 nodes).

When applying the general lower bounds to instances of similar items (Table 4.2) we observe much smaller relative gaps and standard deviations compared to the general instances. In addition, the minimal and the average gap are continuously decreasing for increasing n. As a result of the tighter bounds, the B&B algorithm needs to compute less upper bounds. As the computation of upper bounds using a stochastic gradient algorithm is the most time consuming part of Algorithm 4.2.1 (see chapter 2), the corresponding CPU-times are, as well, significantly smaller. Given a maximum average CPU-time of $1h$, we are thus able to treat instances of more than 2000 items, contrary to 150 items in the general case.

When comparing the performance of the algorithm that uses the general method to compute lower bounds with that proposed for the special case of similar items, we notice that the latter performs only slightly better due to insignificantly better lower bounds.

Items can only be rejected in the second stage

When solving instances with non-similar weight means (see Table 4.3), we first of all remark the high average relative gap between upper and lower bound for small size items. We assume that the dimension of the gap is essentially due to the lower bound whose quality depends mainly on the estimation of $\mathbb{E}\left[\mathbb{1}_{\mathbb{R}^+}(\sum_{i=1}^{n} \widetilde{x}_i\chi_i - c) \cdot \max_{i\in S} \chi_i\right]$ (see subsection 4.2.3). However, contrary to the case where items can only be added, the average gap is continuously decreasing with increasing n and for medium size instances it is even smaller than in the former case. We are thus able to solve instances with up to 150 items, as well.

When applying the general method to compute lower bounds on instances with similar items (Table 4.4), the gaps are less important for instances of small size. However, unlike all the cases studied before, the average gap as well as the minimum gap seem to increase with n (at least within the dimensions tested).

Contrary to the case where items can only be added in the second stage, we observe an important improvement of the gap when replacing the general method to compute lower bounds by the particular method for similar items. In terms of computing time, we are now able to solve problems with up to 1000 items (and more) in less than $1h$ (on average). However, a drawback of our approach is revealed: To compute upper bounds, we use a stochastic gradient method that only computes approximate solution values of the relaxed version of our problem. More precisely, the value obtained might be slightly smaller than the optimum of the relaxation. Consequently, in case of large size instances with a very small integrality gap, we cannot be sure if the upper bound computed is in fact an upper bound on our problem. If in addition the best lower bound obtained is close to the optimal solution value, we might have the strange situation that the computed upper bound is smaller than the computed lower bound, which was the case for some of the instances with 500 and 1000 items (Table 4.4).

	Similar items, general lower bounds					
n	**Minimal Gap**	**Maximal Gap**	**Average Gap**	**Standard Deviat.**	**CPU-time (sec)**	**# considered nodes**
15	2.64%	18.86%	7.11%	3.38%	0.06	14
20	1.87%	12.26%	5.46%	2.41%	0.08	17
30	1.58%	6.29%	3.05%	0.96%	0.16	25
50	0.92%	8.37%	2.09%	1.15%	0.44	47
75	0.72%	1.85%	1.28%	0.34%	0.96	73
100	0.57%	5.17%	1.07%	0.85%	1.70	97
150	0.41%	0.93%	0.65%	0.17%	4.40	170
250	0.29%	0.63%	0.43%	0.10%	14.28	335
500	0.17%	0.33%	0.25%	0.05%	68.80	809
1000	0.08%	0.19%	0.14%	0.02%	308.44	1776
2000	0.06%	0.11%	0.09%	0.01%	1927.06	5586
3000	*	*	*	*	*	*
	Similar items, particular lower bounds					
n	**Minimal Gap**	**Maximal Gap**	**Average Gap**	**Standard Deviat.**	**CPU-time (sec)**	**# considered nodes**
15	2.39%	18.49%	6.67%	3.38%	0.06	13
20	1.73%	11.95%	5.22%	2.38%	0.06	15
30	1.42%	6.05%	2.86%	0.95%	0.14	23
50	0.83%	8.23%	1.96%	1.15%	0.42	45
75	0.60%	1.77%	1.20%	0.34%	0.88	67
100	0.50%	5.10%	1.01%	0.84%	1.62	93
150	0.37%	0.89%	0.60%	0.17%	3.96	154
250	0.26%	0.55%	0.43%	0.06%	14.24	314
500	0.16%	0.32%	0.23%	0.04%	62.86	747
1000	0.09%	0.17%	0.13%	0.02%	276.78	1633
2000	0.06%	0.11%	0.08%	0.01%	1534.18	4477
3000	*	*	*	*	*	*

Table 4.2: Numerical results for the $TSKP$: Items can only be added in the second stage (similar items)

	General items					
n	**Minimal Gap**	**Maximal Gap**	**Average Gap**	**Standard Deviat.**	**CPU-time (sec)**	**# considered nodes**
15	17.77%	42.44%	27.79%	5.63%	0.30	93
20	14.18%	43.36%	23.67%	7.31%	0.84	215
30	10.15%	36.81%	21.57%	7.52%	10.44	1887
50	5.85%	23.93%	9.61%	3.95%	47.76	5408
75	4.10%	13.56%	5.97%	1.86%	142.98	11223
100	3.17%	6.53%	4.19%	0.72%	291.10	17244
150	2.13%	3.31%	2.69%	0.30%	2273.92	90672
250	*	*	*	*	*	*

Table 4.3: Numerical results for the $TSKP$: Items can only be rejected in the second stage (general items)

Items can be added or rejected in the second stage

Concerning the evaluation of the expected second-stage solution, this case is the most difficult among the three problem variants. Therefore, it is not surprising that the gaps in the case where items can be added or rejected in the second stage are slightly higher than in the previous cases (see Tables 4.5 and 4.6).
Concerning the instances of similar items, we observe once more a significant improvement of the results when replacing the general method by the particular one and the results are of nearly the same quality as in the case where items can be added only. In fact, the latter is what one could expect due to the chance-constraint, as in only 5% of the cases items have to be rejected in the second stage.

Recapitulation of the numerical results

To summarize the above results, we assume that whenever items can be rejected in the second stage, the lower bounds obtained by the general method are of a rather poor quality.
Furthermore, we notice that in this case the particular method for similar items significantly improves the gap between upper and lower bound, the number of considered nodes and the CPU-time compared to the general method. If, in addition, we allow items to be added in case of an underload, we are able to treat instances of more than 2000 items in an average CPU-time of less than $30min$.
In contrary, in the case where items can only be added in the second stage the general method performs on instances with similar items nearly as good as the particular one. We deduce that Claim 4.4 and Claim 4.13 are important factors in the computation of the general lower bounds.

	Similar items, general lower bounds					
n	**Minimal Gap**	**Maximal Gap**	**Average Gap**	**Standard Deviat.**	**CPU-time (sec)**	**# considered nodes**
15	8.32%	25.71%	14.84%	3.65%	0.08	27
20	6.71%	24.26%	14.42%	3.87%	0.30	77
30	9.18%	22.16%	14.99%	2.60%	2.76	495
50	10.85%	21.33%	14.85%	2.16%	69.98	7863
75	13.03%	21.22%	15.74%	2.12%	3325.820	249275
100	*	*	*	*	*	*
	Similar items, particular lower bounds					
n	**Minimal Gap**	**Maximal Gap**	**Average Gap**	**Standard Deviat.**	**CPU-time (sec)**	**# considered nodes**
15	0.53%	8.14%	3.82%	2.32%	0.02	7
20	0.29%	7.26%	2.61%	1.78%	0.06	12
30	0.30%	3.72%	1.72%	1.03%	0.12	19
50	0.22%	2.23%	1.02%	0.62%	0.34	36
75	0.16%	1.41%	0.69%	0.38%	0.72	53
100	0.10%	1.03%	0.40%	0.27%	1.16	66
150	0.05%	0.60%	0.26%	0.18%	2.62	101
250	0.01%	0.35%	0.13%	0.11%	7.22	170
500	*-0.03%*	*0.17%*	*0.04%*	*0.05%*	*25.16*	*300*
1000	*-0.06%*	*0.08%*	*0.00%*	*0.03%*	*98.68*	*590*

Table 4.4: Numerical results for the $TSKP$: Items can only be rejected in the second stage (similar items)

	General items					
n	**Minimal Gap**	**Maximal Gap**	**Average Gap**	**Standard Deviat.**	**CPU-time (sec)**	**# considered nodes**
15	19.67%	45.12%	28.63%	5.33%	0.72	223
20	17.66%	42.91%	27.15%	2.57%	5.86	1441
30	12.08%	32.08%	23.43%	5.43%	478.62	75234
50	*	*	*	*	*	*

Table 4.5: Numerical results for the $TSKP$: Items can be added or rejected in the second stage (general items)

	Similar items, general lower bounds					
n	**Minimal Gap**	**Maximal Gap**	**Average Gap**	**Standard Deviat.**	**CPU-time (sec)**	**# considered nodes**
15	9.58%	25.05%	15.36%	3.16%	0.14	43
20	9.38%	22.38%	14.55%	2.68%	0.52	133
30	9.32%	19.23%	13.62%	1.90%	3.48	621
50	9.64%	18.10%	13.13%	1.78%	113.26	12791
75	*	*	*	*	*	*
	Similar items, particular lower bounds					
n	**Minimal Gap**	**Maximal Gap**	**Average Gap**	**Standard Deviat.**	**CPU-time (sec)**	**# considered nodes**
15	2.92%	18.80%	6.87%	3.33%	0.04	13
20	1.90%	11.95%	5.30%	2.34%	0.06	16
30	1.59%	6.05%	2.88%	0.94%	0.14	24
50	1.06%	8.23%	1.99%	1.15%	0.42	45
75	0.78%	1.76%	1.23%	0.31%	0.90	69
100	0.62%	5.24%	1.04%	0.85%	1.62	98
150	0.45%	0.96%	0.63%	0.15%	4.32	170
250	0.30%	0.61%	0.42%	0.09%	13.68	324
500	0.19%	0.34%	0.24%	0.04%	63.48	755
1000	0.11%	0.17%	0.14%	0.02%	303.40	1797
2000	0.06%	0.11%	0.09%	0.01%	1739.30	5104
3000	*	*	*	*	*	*

Table 4.6: Numerical results for the $TSKP$: Items can be added or rejected in the second stage (similar items)

4.3 The Two-Stage Knapsack Problem with discrete weight distributions (TSK^D)

4.3.1 Mathematical formulation and properties

The Two-Stage Knapsack problem studied in this section is similar to that studied in section 4.2: The capacity c of the knapsack is assumed to be deterministic while the item weights are random. Contrary to the previous section we assume that weight χ_i of item i is discretely distributed with realizations $\chi_i^1, \ldots, \chi_i^K$ in the K scenarios with nonzero probabilities $p^1, \ldots, p^K$. All weights are assumed to be strictly positive.
In this section we assume for instance that the first-stage rewards $r_i > 0$ are independent of the weights. The case of weight dependent rewards and penalties will be treated in subsection 4.3.3. The second-stage rewards $\overline{r}_i > 0$ are once more assumed to be strictly smaller than the first stage rewards. In case item i is removed in the second stage, an item specific, fix penalty d_i has to be paid that is strictly greater than the reward that we have obtained in the first stage. The definition of the decision vectors is as in section 4.2. Contrary to the previous section we allow an exchange of items in the second stage, i.e. items can be removed in case of an underload and can be added in case of an overload (after having removed some other items). The resulting Two-Stage Knapsack problem with discrete weight distributions TSK^D can be formulated as follows:

The Two-Stage Knapsack Problem with discretely distributed weights (TSK^D)

$$
\begin{aligned}
(TSK^D) \quad \max_{x\in\{0,1\}^n} \quad & \sum_{i=1}^{n} r_i x_i + \sum_{k=1}^{K} p^k \mathcal{Q}(x, \chi^k) \\
\text{s.t.} \quad & \mathcal{Q}(x,\chi) = \max_{y^+,y^-\in\{0,1\}^n} \sum_{i=1}^{n} \overline{r}_i y_i^+ - \sum_{i=1}^{n} d_i y_i^- && \text{(4.16a)} \\
& \text{s.t. } y_i^+ \le 1 - x_i, \quad \forall\, i = 1,\ldots,n, && \text{(4.16b)} \\
& \qquad y_i^- \le x_i, \quad \forall\, i = 1,\ldots,n, && \text{(4.16c)} \\
& \qquad \sum_{i=1}^{n} (x_i + y_i^+ - y_i^-)\chi_i \le c. && \text{(4.16d)}
\end{aligned}
$$

Properties of the TSK^D

Property 1: The two-stage problem (4.16) is a *relatively complete recourse* problem, i.e. for every feasible first-stage decision there exists a feasible second-stage decision.

Property 2: Given a first-stage decision and a realization of χ, solving the second-stage problem means solving a deterministic knapsack problem: Let $S \subseteq \{1,\ldots,n\}$ be the index

set of the first-stage items, and $\overline{S} := \{1, \dots, n\} \setminus S$. Let us define a new second-stage decision vector $z \in \{0,1\}^n$: for $i \in S$ $z_i = 1$ indicates that item i is kept in the second stage and for $j \in \overline{S}$ we set $z_j = 1$ if item j is a second-stage item. Then, in scenario k the second-stage problem consists in solving the following problem:

$$\begin{aligned} \max_{z \in \{0,1\}^n} \quad & \sum_{i \in S} (r_i z_i - d_i(1 - z_i)) + \sum_{j \in \overline{S}} \overline{r}_j z_j \\ \text{s.t.} \quad & \sum_{i=1}^{n} z_i \chi_i^k \leq c. \end{aligned} \tag{4.17a}$$

Defining $C := -\sum_{i \in S} d_i$, the objective can be rewritten as

$$\max \quad \sum_{i \in S} (r_i + d_i) z_i + \sum_{j \in \overline{S}} \overline{r}_j z_j + C$$

We therefore obtain a knapsack problem with reward $r_i + d_i$ for the first-stage items (if kept) and reward $\overline{r}_j$ for an added second-stage item.

Property 3: The TSK^D has a deterministic equivalent reformulation as a combinatorial optimization problem with linear objetctive and constraints: By introducing K copies of both the second-stage decision vector y^+ and the second-stage decision vector y^- (denoted $(y^+)^k$ and $(y^-)^k$, $k \in \{1, \dots, K\}$, respectively) and treating the second-stage constraints for each second-stage scenario separately, one obtains the following reformulation:

$$\max \quad \sum_{i=1}^{n} r_i x_i + \sum_{k=1}^{K} p^k \left(\sum_{i=1}^{n} \overline{r}_i (y^+)_i^k - \sum_{i=1}^{n} d_i (y^-)_i^k \right)$$

$$\text{s.t.} \quad (y^+)_i^k \leq 1 - x_i, \quad \forall\, i = 1, \dots, n, \forall\, k = 1, \dots, K, \tag{4.18a}$$

$$(y^-)_i^k \leq x_i, \quad \forall\, i = 1, \dots, n, \forall\, k = 1, \dots, K, \tag{4.18b}$$

$$\sum_{i=1}^{n} (x_i + (y^+)_i^k - (y^-)_i^k) \chi_i^k \leq c, \quad \forall\, k = 1, \dots, K, \tag{4.18c}$$

$$x \in \{0,1\}^n, \tag{4.18d}$$

$$(y^+)^k, (y^-)^k \in \{0,1\}^n \quad \forall\, k = 1, \dots, K. \tag{4.18e}$$

Property 4: Let $AddTSK^D$ denote the variant of TSK^D (4.16) where, in the second stage, items can only be added. The following proposition shows that this problem is a special case of the general TSK^D (4.16):

Proposition 4.20. *For any instance of $AddTSK^D$ there exists an instance of TSK^D with identical optimal solution value and such that an optimal solution of the TSK^D instance is optimal solution of the $AddTSK^D$ instance, and vice versa.*

Remark 4.21. *Before proving the above proposition, let us remark that $AddTSK^D$ is not a relatively complete recourse problem, as there exist first-stage decisions that make the second-stage problem (in one or more scenarios) infeasible. For these solutions we define the corresponding objective function value to be $-\infty$. Such a solution is thus clearly not optimal as there always exists a solution of an instance of the $AddTSK^D$ with objective function value greater or equal than 0.*
The problem of second-stage infeasibility could be arranged by adding K capacity constraints to the first stage.

Proof of Proposition 4.20. Let an instance of $AddTSK^D$ be given. We construct a corresponding instance of TSK^D as follows: The n-dimensional parameter vectors $r, \overline{r}, \chi^k$, the probabilities p^k and the capacity c are carried over to the new problem. For all $i = 1, \ldots, n$ the penalty d_i is set to $\max_{k\in\{1,\ldots,K\}} \frac{r_i}{p^k} + 1$.
Let x^* be an optimal solution of the obtained instance of TSK^D and let $(y^-)^k = (y^-)^k(x)$ be a corresponding optimal second-stage solution for y^- in scenario k. Assume that there exists $\widetilde{k} \in \{1, \ldots, K\}$ and $i \in \{1, \ldots, n\}$ such that $(y^-)_i^{\widetilde{k}} = 1$ (i.e. item i has been added in the first stage and rejected in scenario $\widetilde{k}$ in the second stage). Then, item i contributes at most

$$r_i - p^{\widetilde{k}} d_i = r_i - p^{\widetilde{k}}(\max_k \frac{r_i}{p^k} + 1) \leq r_i - r_i - p^{\widetilde{k}} < 0$$

to the objective function value. We are thus able to increase the objective function value by setting $x_i = 0$, a contradiction to the assumption that x^* is optimal. It follows that for any optimal (first-stage) solution of the constructed TSK^D instance all corresponding optimal second-stage solutions are such that $(y^-)_i^{\widetilde{k}} = 0$ for all $k \in \{1, \ldots, K\}$ and $i \in \{1, \ldots, n\}$. An optimal solution of the TSK^D is therefore feasible for the corresponding $AddTSK^D$, with identical objective function value. As any solution of $AddTSK^D$ is, in turn, feasible for the TSK^D instance (with identical objective function value), the proposition is proved. □

Property 5: In the TSK^D (4.16) the knapsack capacity is assumed to be deterministic, i.e. identical in all scenarios. However, the formulation where the capacities are, as well, scenario dependent (with a finite number of outcomes) is in fact equivalent to the TSK^D (4.16) as we could simply multiply the capacity constraint by an appropriate factor in each of the K scenarios. Note that we use here that all outcomes of the capacity are known and that their number is finite. In the case where the knapsack capacity is bounded from above but has an infinite number of possible outcomes, an equivalent reformulation with uniform (deterministic) capacity can still be obtained by introducing an additional item and setting the uniform capacity to a value strictly greater than the capacity's upper bound. The random weight of the additional item is defined as the difference between the newly defined, deterministic capacity and the initial, random capacity. Its outcomes are thus always strictly positive. The reward of the additional item is defined in a way that any optimal first-stage solution must contain this item.

4.3.2 Equivalence of the $AddTSK^D$ and the $MCKP$

Formulation of the $AddTSK^D$ as an $MCKP$

The multiply-constrained knapsack problem $MCKP$ (sometimes also called multi-dimensional vector knapsack problem) with uniform capacities can be defined as:

$$
\begin{aligned}
(MCKP) \quad \max \quad & \sum_{i=1}^{\widetilde{n}} \widetilde{r}_i \widetilde{x}_i \\
\text{s.t.} \quad & \sum_{i=1}^{\widetilde{n}} \widetilde{x}_i \widetilde{w}_i^j \leq \widetilde{c} \quad \forall\, j = 1, \ldots, m, && \text{(4.19a)} \\
& \widetilde{x} \in \{0,1\}^{\widetilde{n}}. && \text{(4.19b)}
\end{aligned}
$$

where $\widetilde{w}_i^j \geq 0$ and $\widetilde{c}, \widetilde{r}_i > 0$ for all $i \in \{1, \ldots, \widetilde{n}\}$, $j \in \{1, \ldots, m\}$.
In the following $MCKP$ stands for the multiply-constrained knapsack problem with uniform capacities. Note, however, that as long as the capacities are required to be strictly positive, any multiply-constrained knapsack problem with non-uniform capacities can be trivially reformulated as an $MCKP$ by multiplying the capacity constraints with a corresponding factor.
The $AddTSK^D$ can be stated as follows:

$$
\begin{aligned}
\max_{x \in \{0,1\}^n} \quad & \sum_{i=1}^{n} r_i x_i + \sum_{k=1}^{K} p^k \sum_{i=1}^{n} \overline{r}_i (y_i^+)^k \\
\text{s.t.} \quad & (y_i^+)^k + x_i \leq 1 \quad \forall\, i = 1, \ldots, n, \forall\, k = 1, \ldots, K, && \text{(4.20a)} \\
& \sum_{i=1}^{n} (x_i + (y_i^+)^k) \chi_i^k \leq c. && \text{(4.20b)}
\end{aligned}
$$

Problem (4.20) is clearly a multiply-constrained knapsack problem (with non-uniform capacities) with strictly positive rewards and capacities and nonnegative weights. Multiplying constraints (4.20a) by $c > 0$ would turn it in an $MCKP$.

Solving an instance of the $MCKP$ via an instance of the $AddTSK^D$

If we do not allow the rewards to be zero or negative, it is in general not possible to equivalently reformulate the $MCKP$ as an $AddTSK^D$. Nevertheless, it is possible to obtain an optimal solution for an instance of the $MCKP$ by solving an instance of the $AddTSK^D$. This is the subject of the next proposition:

Proposition 4.22. *Let an instance of the MCKP be given. Then, there exists an instance of the AddTSK^D such that an optimal (first-stage) solution of the latter is optimal solution to the former.*

Proof. W.l.o.g. we assume all parameters of the given instance of the $MCKP$ (i.e. $\widetilde{r}_i, \widetilde{w}_i^j, \widetilde{c}$ for all $i = 1, \ldots, \widetilde{n}$, $j = 1, \ldots, m$) to be integer and construct a corresponding instance of the $AddTSK^D$ as follows:
The knapsack capacity is $\widetilde{c}$. The first-stage reward of item i is $\widetilde{r}_i$. There are m second-stage scenarios. The weight of item i in scenario j is set to $\widetilde{w}_i^j$. The probabilities of the scenarios are uniformly set to $\frac{1}{m}$ and the second-stage rewards are fixed at $\frac{1}{\widetilde{n}+1}$ where $\widetilde{n}$ is the number of available items for the given instance of $MCKP$.
Let x^* be an optimal solution of the constructed instance of $AddTSK^D$. Let v^* be the corresponding solution value. It is easy to see that $\lfloor v^* \rfloor$ is the gain provided by the items added in the first stage, as the first-stage solution value is always integer and the contribution of adding items in the second stage is at most

$$\sum_{j=1}^{m} \frac{1}{m} \sum_{i=1}^{\widetilde{n}} \frac{1}{\widetilde{n}+1} = \frac{\widetilde{n}}{\widetilde{n}+1}$$

which is strictly smaller than 1. Moreover, the vector x^* is feasible for the initial instance of the $MCKP$ with objective function value $\lfloor v^* \rfloor$ as for the $AddTSK^D$ any *optimal* first-stage solution is always second-stage feasible. $\lfloor v^* \rfloor$ is thus a lower bound on the optimal solution value of the given $MCKP$ instance.
Let us assume that there exists a solution $\widetilde{x}^*$ of the given $MCKP$ instance with objective function value $\widetilde{v}^* > \lfloor v^* \rfloor$. As the optimal solution of the $MCKP$ instance is integer, we especially have $\widetilde{v}^* \geq \lfloor v^* \rfloor + 1 > v^*$. As $\widetilde{x}^*$ is a feasible first-stage solution for the constructed instance of $AddTSK^D$ (in particular, it respects the capacity constraint in all scenarios), we have a contradiction.
It follows that any optimal first-stage solution of the constructed $AddTSK^D$ instance is optimal for the given $MCKP$ instance. Moreover, the optimal solution value of the latter is given by the integer part of the optimal solution value of the former. □

4.3.3 Non-approximability results for the TSK^D and some special cases

The general TSK^D and the $AddTSK^D$

It has been shown in [LY99] that the multiply-constrained knapsack problem with non-uniform capacities does not have a constant-factor approximation algorithm unless $\mathcal{P} = \mathcal{NP}$. The authors prove this by a reduction from the maximum clique problem that can be formulated as a multiply-constrained knapsack problem with capacity 1 in each constraint. Their proof is thus directly applicable to the $MCKP$ as well. The same result would be obtained by using the above remark that the $MCKP$ and multiply-constrained knapsack problem with non-uniform capacities are, in fact, equivalent problems.
Based on this, we can now show the following:

Theorem 4.23. *There exists no constant-factor approximation algorithm for the TSK^D, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Assume that $\mathcal{P} \neq \mathcal{NP}$ and that there exists an algorithm $\mathcal{A}$ and a constant $0 \leq \alpha < 1$ such that $\mathcal{A}$ finds, in polynomial time, a solution to any instance of the TSK^D with worst case ratio α. W.l.o.g. we assume that $\alpha = \frac{1}{M}$ for an $M \in \mathbb{N}$.
Let an instance of the $MCKP$ be given. Like in the proof of Proposition 4.22 we assume all parameters to be integer. Moreover, we multiply the objective function by M such that any optimal solution of the given $MCKP$ instance becomes a multiple of M.
As shown in subsection 4.3.2 we can find an optimal solution and the corresponding solution value of this $MCKP$ instance by solving a corresponding instance of the $AddTSK^D$. The $AddTSK^D$ can in turn be equivalently reformulated as an instance of the TSK^D. Applying algorithm $\mathcal{A}$ to the latter gives us therefore an approximate solution of the constructed instance of the $AddTSK^D$ with worst case ratio α.
Let $v^{\mathcal{A}}$ be the obtained solution value and v^* the optimal solution value of the $AddTSK^D$, i.e. we have $\frac{v^{\mathcal{A}}}{v^*} \geq \alpha$. Recall that by the construction in the proof of Proposition 4.22 the integer part of the optimal solution value of the $AddTSK^D$ gives us the optimal solution value of the initial $MCKP$. It follows that $\lfloor v^* \rfloor$ is a multiple of M. We have:

$$\begin{aligned}
\lfloor v^{\mathcal{A}} \rfloor + (v^{\mathcal{A}} \mod \mathbb{Z}) &= v^{\mathcal{A}} \\
&\geq \alpha v^* \\
&= \frac{1}{M}(\lfloor v^* \rfloor + (v^* \mod \mathbb{Z})) \\
&= \lfloor \frac{v^*}{M} \rfloor + \frac{1}{M}(v^* \mod \mathbb{Z})) \\
&\Longrightarrow \lfloor v^{\mathcal{A}} \rfloor \geq \frac{1}{M} \lfloor v^* \rfloor
\end{aligned}$$

Solving the TSK^D by algorithm $\mathcal{A}$ thus gives us a solution of the constructed instance of $AddTSK^D$ whose objective function value is at least an α fraction of the optimal solution value of the initial $MCKP$ instance. As all problem reformulations can be made in a number of steps polynomial in the input size of the given $MCKP$ instance, we get a contradiction to the fact that the $MCKP$ cannot be approximated to a constant factor unless $\mathcal{P} \neq \mathcal{NP}$. This terminates the proof. □

From the proof of the previous theorem we immediately get the following corollary:

Corollary 4.24. *There exists no constant-factor approximation algorithm for the $AddTSK^D$, unless $\mathcal{P} = \mathcal{NP}$.*

As the TSK^D defined in this chapter is a special case of the Two-Stage Knapsack problem with scenario dependent capacities, we also have:

Corollary 4.25. *There exists no constant-factor approximation algorithm for the Two-Stage Knapsack problem with scenario dependent capacities.*

The special case where items can only be rejected in the second stage

Let $RejTSK^D$ be the variant of the TSK^D (4.16) where, in the second stage, items can only be rejected. Similar to the case of the $AddTSK^D$ (see subsection 4.3.2) it can be shown that any instance of the $RejTSK^D$ can be solved via a corresponding instance of the $MCKP$ and vice versa.

Proposition 4.26. *Any instance of the $RejTSK^D$ can be equivalently reformulated as an instance of the $MCKP$.*

Proof. Let an instance of the $RejTSK^D$ be given. First of all we redefine the first and second-stage decision variables:

- First-stage decision vector $\widetilde{x}$: $\widetilde{x}_i = 1$ if and only if item i *is not* added in the first stage
- Second-stage decision vector $(\widetilde{y})^k$: $(\widetilde{y})^k_i = 1$ if and only if item i has been added in the first stage and is kept in the second stage in scenario k (i.e. if and only if item i is in the knapsack after the second-stage choice has been made)

There are three possible cases for an item i in scenario k:

- Item i is added in the first and kept in the second stage, i.e. $\widetilde{x}_i = 0$ and $(\widetilde{y})^k_i = 1$. In this case item i contributes r_i to the total gain.
- Item i is added in the first and rejected in the second stage, i.e. $\widetilde{x}_i = (\widetilde{y})^k_i = 0$. In this case item i contributes $-(d_i - r_i)$ to the total gain (in scenario k).
- Item i is not added in the first stage, i.e. $x_i = 1$ and $(\widetilde{y})^k_i = 0$. In this case item i does not contribute at all to the total gain (in any scenario).

Remark that $x_i = (\widetilde{y})^k_i = 1$ is not possible.

Based on these observations, the $RejTSK^D$ instance can be reformulated as follows:

$$
\begin{aligned}
(TSK_2^D) \quad \max \quad & \sum_{k=1}^{K} p^k \left(\sum_{i=1}^{n} r_i \widetilde{y}_i^k - \sum_{i=1}^{n} (d_i - r_i)(1 - \widetilde{x}_i - \widetilde{y}_i^k) \right) \\
\text{s.t.} \quad & \sum_{i=1}^{n} \widetilde{y}_i^k \chi_i^k \leq c \quad \forall\, k = 1, \ldots, K, && \text{(4.21a)} \\
& \widetilde{x}_i + \widetilde{y}_i^k \leq 1 \quad \forall\, i = 1, \ldots, n, \forall\, k = 1, \ldots, K, && \text{(4.21b)} \\
& \widetilde{x} \in \{0,1\}^n, && \text{(4.21c)} \\
& \widetilde{y}^k \in \{0,1\}^n \quad \forall\, k = 1, \ldots, K. && \text{(4.21d)}
\end{aligned}
$$

After removing the constant term in the objective function and multiplying the $n \cdot K$ constraints (4.21b) by c, we obtain the following multiply-constrained knapsack problem with nonnegative weights, strictly positive rewards and uniform, strictly positive capacities:

$$
\begin{aligned}
(MCKPR) \quad \max \quad & \sum_{k=1}^{K} p^k \left(\sum_{i=1}^{n} r_i \widetilde{y}_i^k + \sum_{i=1}^{n} (d_i - r_i)(\widetilde{x}_i + \widetilde{y}_i^k) \right) \\
\text{s.t.} \quad & \sum_{i=1}^{n} \widetilde{y}_i^k \chi_i^k \leq c, \quad \forall\, k = 1, \ldots, K, && \text{(4.22a)} \\
& c\widetilde{x}_i + c\widetilde{y}_i^k \leq c, \quad \forall\, i = 1, \ldots, n,\ \forall\, k = 1, \ldots, K, && \text{(4.22b)} \\
& \widetilde{x} \in \{0,1\}^n, && \text{(4.22c)} \\
& \widetilde{y}^k \in \{0,1\}^n \qquad \forall\, k = 1, \ldots, K. && \text{(4.22d)}
\end{aligned}
$$

□

Contrary to the $AddTSK^D$ the $RejTSK^D$ is even equivalent to the $MCKP$ as any instance of the $MCKP$ can, as well, be equivalently reformulated an instance of the $RejTSK^D$: It is sufficient to set the second-stage penalties high enough, i.e. such that rejecting an item in the second stage is never optimal:

Proposition 4.27. *Any instance of the $MCKP$ can be equivalently reformulated an instance of the $RejTSK^D$.*

Let an instance of the $MCKP$ (4.19) be given. We construct an equivalent instance of the $RejTSK^D$ having the following parameters:

- $p^k = \frac{1}{m}$ for all $k = 1, \ldots, m$
- $r_i := \widetilde{r}_i$ for all $i = 1, \ldots, \widetilde{n}$
- $d_i^k := m \cdot r_i + 1$ for all $i = 1, \ldots, \widetilde{n}$ and $k = 1, \ldots, m$
- $w_i^k := \widetilde{w}_i^k$ for all $i = 1, \ldots, \widetilde{n}$ and $k = 1, \ldots, m$
- $c = \widetilde{c}$

Let us denote $EqTSK^D$ the obtained instance of the $RejTSK^D$. Clearly, any solution of the initial $MCKP$ instance is feasible first-stage solution for the $EqTSK^D$ with identical (overall) objective function value (as no item needs to be rejected in any scenario). In turn, any first-stage solution of the $EqTSK^D$ satisfies the constraints of the $MCKP$ if no item needs to be rejected in any of the K second-stage scenarios. The objective function values are once more the same. It thus remains to prove the following claim:

Claim 4.28. *Let $(x_1^*, \ldots, x_{\widetilde{n}}^*)$ be an optimal first-stage solution of the $EqTSK^D$ and let $(y^1)^*, \ldots, (y^{\widetilde{n}})^*$ be a corresponding optimal second-stage solution. Then $(y^k)_i^* = 0$ for all $i = 1, \ldots, \widetilde{n}$ and $k = 1, \ldots, m$.*

Proof of the claim. Let v^* be the (corresponding) optimal solution value of the $EqTSK^D$. Suppose there exists $i \in \{1, \dots, \widetilde{n}\}$ and at least one scenario h with $(y^h)_i^* = 1$. In this case item i contributes at most $r_i - \frac{1}{m}(m \cdot r_i + 1) = -\frac{1}{m}$ to the total expected gain. Therefore, by not adding item i in the first stage we obtain a solution with objective function value at least $v^* + \frac{1}{m}$, a contradiction. □

As a direct corollary of proposition 4.27 as well as the non-approximability of the $MCKP$ we obtain:

Corollary 4.29. *There exists no constant-factor approximation algorithm for the $RejTSK^D$, unless $\mathcal{P} = \mathcal{NP}$.*

Remark 4.30. *To finish this subsection, we would like to remark the following:*
(i) As long as we require the second-stage rewards to be strictly positive, the $RejTSK^D$ is not a special case of the TSK^D: In general, we cannot reformulate an instance of the $RejTSK^D$ as an instance of the TSK^D with the same solution value, as this is the case for the $AddTSK^D$. However, by combining the results of this subsection with those of the previous subsections (or by proving it with a proof similar to that of Proposition 4.22) one can show that the optimal solution of an instance of $RejTSK^D$ can be obtained by solving a corresponding instance of TSK^D.
(ii) Nevertheless, the proof of Theorem 4.23 can be rewritten in the following way: Any instance of the $MCKP$ can be equivalently reformulated as an instance of the $RejTSK^D$. In turn, the solution of an instance of the $RejTSK^D$ can be obtained by solving a corresponding instance of the TSK^D.

The TSK^D with weight dependent rewards and penalties

As the actual weights of the items are not known in advance, one might assume that the rewards and penalties depend on the outcomes of the weights, i.e. instead of a reward r_i for item i, we are given the reward per weight unit (and/or the penalty per weight unit). One might wonder, if this variant is easier to solve as the rewards and penalties per weight unit of an item do not depend on the scenario.
Let us denote by V_1TSK^D (first variant of the TSK^D) the following problem:

$$
\begin{aligned}
(V_1TSK^D) \quad \max_{x \in \{0,1\}^n} \quad & \sum_{k=1}^{K} p^k \sum_{i=1}^{n} r_i \chi_i^k x_i + \sum_{k=1}^{K} p^k \mathcal{Q}(x, \chi^k) \\
\text{s.t.} \quad & \mathcal{Q}(x, \chi) = \max_{y^+, y^- \in \{0,1\}^n} \sum_{i=1}^{n} \overline{r}_i \chi_i y_i^+ - \sum_{i=1}^{n} d_i \chi_i y_i^- && \text{(4.23a)} \\
& \text{s.t. } y_i^+ \leq 1 - x_i \quad \forall\, i = 1, \dots, n, && \text{(4.23b)} \\
& \qquad y_i^- \leq x_i \quad \forall\, i = 1, \dots, n, && \text{(4.23c)} \\
& \qquad \sum_{i=1}^{n} (x_i + y_i^+ - y_i^-) \chi_i \leq c. && \text{(4.23d)}
\end{aligned}
$$

Theorem 4.31. *The V_1TSK^D does not admit a constant-factor approximation algorithm unless $\mathcal{P} = \mathcal{NP}$.*

Proof. The idea of the proof is the same as in case of Theorem 4.23:
First of all, the special variant of the V_1TSK^D where items can only be added in the second stage ($Add V_1TSK^D$) can be equivalently reformulated as an instance if V_1TSK^D by choosing the second-stage penalties per weight unit d_i sufficiently large.
We will now see how to solve an instance of $MCKP$ by solving a corresponding instance of the $Add V_1TSK^D$: First of all we recall that the solution of an instance of $MCKP$ can be obtained by solving a corresponding instance of $AddTSK^D$ (see subsection 4.3.2). Let r_i and $\overline{r}_i$ be the first and second-stage reward of item i in this $AddTSK^D$ instance, respectively. We compute factors $a_1, \ldots, a_m$ such that for all items i of the $AddTSK^D$ instance we have

$$r_i = \sum_{k=1}^{m} \frac{1}{m} a_k \chi_i^k$$

A possible choice is of course $a_k = \frac{r_i}{\chi_i^k}$. In the same way we reformulate the second-stage rewards $\overline{r}_i$ as

$$\overline{r}_i = \sum_{k=1}^{m} \frac{1}{m} b_k \chi_i^k$$

The so obtained reformulation of the $AddTSK^D$ is clearly an instance of $Add V_1TSK^D$. Like in the proof of Theorem 4.23 the existence of a constant-factor approximation algorithm for the V_1TSK^D would thus imply the existence of such an algorithm for the $MCKP$. □

The TSK^D with reward dependent second-stage rewards and penalties

For some TSPs, the special case where the second-stage costs are multiples of the first-stage rewards with uniform factor for all items is easier to solve than the general case (see subsection 4.1.4).

Theorem 4.32. *Let V_2TSK^D denote the variant of the TSK^D such that there exist $a < 1$ and $b > 1$ with $\overline{r}_i = a \cdot r_i$ and $d_i = b \cdot r_i$ for all $i \in \{1, \ldots, n\}$. Then the existence of a constant-factor approximation algorithm for V_2TSK^D would imply $\mathcal{P} = \mathcal{NP}$.*

Proof. The proof is again similar to that of Theorem 4.23, i.e. we show that the solution of an instance of the $MCKP$ can be obtained by solving a corresponding instance of the V_2TSK^D: Let once more $Add V_2TSK^D$ denote the variant of the V_2TSK^D where, in the second stage, items can only be added. By choosing b sufficiently large, an instance of the $Add V_2TSK^D$ can be equivalently formulated as an instance of the V_2TSK^D.
In addition, an instance of the $MCKP$ can be solved via an instance of the $Add V_2TSK^D$ in a similar way to that shown in the proof of Proposition 4.22. The factor a for the

second-stage rewards could for example be chosen as

$$a = \frac{1}{n \cdot \max_i r_i + 1} \Longrightarrow \sum_{j=1}^{m} \frac{1}{m} \sum_i \overline{r}_i = \sum_i a \cdot r_i \leq \frac{n \cdot \max_i r_i}{n \cdot \max_i r_i + 1} < 1$$

□

4.3.4 Final Remark

The proofs of non-approximability leave the question unanswered if there might exist a lower bound on the second-stage rewards and/or an upper bound on the second-stage penalties such that an instance that respects these bounds admits a constant-factor approximation algorithm. For example, it is clear that if we would allow the second-stage rewards to be equal to the first-stage rewards, and if in addition the number of scenarios is polynomial in the input size, a simple $\frac{1}{2}$-approximation algorithm is the following: In the first stage, no item is added at all and to every second-stage problem we apply the well known $\frac{1}{2}$-approximation algorithm for deterministic knapsack problems. As we cannot increase the gain by choosing items in the first-stage, the approximation ratio of this algorithm is $\frac{1}{2}$, as well. In the same way, if the second-stage penalties are all equal to the first-stage rewards, we obtain a similar constant-factor approximation algorithm by adding all items in the first stage.
Especially in the case where the second-stage rewards are a fix fraction and the second stage penalties a fix multiple of the first-stage rewards, one might wonder if there exists a lower bound $\hat{a}$ on the reward factor $a < 1$ or an upper bound $\hat{b} > 1$ on the penalty factor b such that for instances of the V_2TSK^D with $a \geq \widetilde{a}$ and $b \leq \widetilde{b}$ there exists a constant-factor approximation algorithm.

4.4 Concluding remarks and future work

In this chapter we presented results for the Two-Stage Knapsack problem with random weights.
In the first part we studied the case where the random weights are independently normally distributed. We introduced a chance-constraint in the first stage in order to restrict the probability of an overload, but all methods and bounds proposed are still valid if the problem contains no such constraint. As the weights are assumed to be continuously distributed, no practical method to exactly evaluate the expectation of the second-stage solution value for a given first stage decision is known. Instead, we propose to compute lower bounds on this expectation and we use a B&B framework in order to find first-stage solutions that provide best possible lower bounds. For comparison, upper bounds are computed by solving a continuous version of the problem using a stochastic gradient algorithm. Special interest is given to the case where the items have similar weight means. Numerical tests have shown that our method provides bounds with small relative gaps for medium size instances with similar items and even near-optimal lower bounds for large

size instances with similar items. However, for non-similar items the gaps are rather high, especially when we allow items to be rejected in the second stage. Further work will therefore primarily consist in improving the quality of the general bounds.
Furthermore, when searching for good lower bounds, one could replace the exhaustive B&B framework by a heuristic, or stop the B&B as soon as the upper bound and best lower bound found are sufficiantly close.
In order to evaluate the quality of the upper and lower bounds, an idea could be to apply an SAA method to the initial problem, i.e. to approximate the optimal solution by discretization of the continuous sample space and solving the corresponding deterministic equivalent combinatorial linear problem. However, this approach is only practicable if already a relatively small sample is sufficient to obtain good approximations.
The second part of the chapter was dedicated to the Two-Stage Knapsack problem with finite number of second-stage scenarios. In this case, the problem can be equivalently reformulated as a linear programming problem with binary decision variables and solved with common algorithms or algorithms adapted to the special shape of the problem. However, solving the Two-Stage Knapsack problem exactly naturally becomes intractable in case of a large number of scenarios. We therefore studied the approximability of the problem. We showed that Two-Stage Knapsack problems with finite number of scenarios are in fact equivalent to multiply-constrained knapsack problems. More precisely, a solution of an instance of the former can be obtained by solving an instance of the latter, and vice versa. All approximation results for the multiply-constrained knapsack problem therefore also apply to the Two-Stage Knapsack problem. It follows, that the Two-Stage Knapsack problem does not admit a constant-factor approximation algorithm, unless $\mathcal{P} = \mathcal{NP}$. We showed, that the same holds true for the special variants where items can only be added or rejected in the second stage, where the rewards and penalties depend linearly on the item weights or where the second-stage rewards and penalties are a constant fraction or multiple of the first-stage rewards, respectively.
An open question is if there exists a lower bound on the second-stage rewards and/or an upper bound on the second-stage penalties such that an instance that respects these bounds admits a constant-factor approximation algorithm. Furthermore, there might exist approximation algorithms for the Two-Stage Knapsack problem with discretely distributed weights whose approximation ratio is not constant, but decreases e.g. with the input size.
For a fixed number of constraints, the multiply-constrained knapsack problem admits a $PTAS$ (though no $FPTAS$) whose running time is exponential in the number of constraints (see e.g. [LY99]). This running time can be tractable for multiply-constrained knapsack problems as often the number of constraints is much smaller than the number of items. When reformulating the Two-Stage Knapsack problem studied in this chapter as a multiply-constrained knapsack problem, the number of constraints is, however, strictly greater than the number of items. Nevertheless, further work might consist in studying if some of the ideas of this $PTAS$ might be used to solve the Two-Stage Knapsack problem (e.g. with fixed number of scenarios) approximately and to compare the performance of this algorithm with exact solution methods.

In the case of two-stage decision problems, the decision making process is composed of two decisions: One while the random parameters are still unknown and one once their actual values have been revealed. The decision maker, however, is in most cases identical in both stages and, more importantly, the aim is the same. In the case of bilevel programs, as studied in the next chapter, the decision also consists of two parts. The main difference is however that two different decision makers with generally different (or even conflictive) aims are involved. As one of the decisions is made with respect to the second and in general not vice-versa, one speaks of "levels" rather than "stages".
Basically, bilevel programming problems are deterministic, i.e. one assumes that it is possible to predict the decision of the second party by solving a deterministic optimization problem. In the variant that we will study in the next chapter we assume that not all of the parameters are given deterministically. This is once more modeled by introducing a chance-constraint. Like in the second part of this chapter, only a finite number of possible scenarios will be assumed. Nevertheless, the obtained problem turns out to be generally intractable when solved exactly and we thus propose to solve a Lagrangian relaxation in order to provide bounds on the optimal solution value.

5 The Stochastic Bilevel Problem with Knapsack Chance-Constraint

5.1 Introduction

5.1.1 Bilevel Problems

Bilevel problems might arise wherever decisions have to be made by two dependent parties. Each party aims to maximize its gain (or minimize its cost) that might depend on the decision of the second party.
Bilevel models are chosen whenever a hierarchy is established between the two decision makers. In a bilevel programming problem this hierarchy is reflected by introducing the second party's (or *follower*'s) decision problem as being part of the first party's (or *leader*'s) constraints. As a consequence, the follower's decision does not depend on the leader's constraints, while the leader has to make his decision in a way that the follower's problem admits a finite solution. The leader's decision problem is therefore often called *upper level* problem, while the follower's decision is made at the *lower level*.
In general, a bilevel programming problem can be stated as follows:

$$\begin{aligned}
(\mathrm{BP}) \quad & \max_{\substack{x\in\mathbb{R}^{n_x}\\ y\in\mathbb{R}^{n_y}}} \quad F(x,y) && (5.1a)\\
& \text{s.t. } G(x,y) \leq 0, && (5.1b)\\
& \quad y \in \arg\max_{y\in\mathbb{R}^{n_y}} f(x,y), && (5.1c)\\
& \quad \text{s.t. } g(x,y) \leq 0. && (5.1d)
\end{aligned}$$

where x is the upper level decision vector of dimension n_x (*leader's decision vector*) and y is the lower level decision vector of dimension n_y (*follower's decision vector*).
Bilevel programming problems have their origin in the so called *Stackelberg game* (see [Sta52]): Stackelberg observed that it is not uncommon in economics that a certain monopoly is given to one party (the leader) which allows it to make its decision previous to the other parties (the followers). Naturally, the followers have to make their decisions depending on the leader's decisions. Stackelberg games are thus the counterpart of Nash games where all players make their decision at the same time (see [Zha09] for a study on the relations between Stackelberg and Nash games).
The first papers treating bilevel programming problems appeared in the seventies ([BM73]) where the problem was initially described as "Mathematical Program with Optimization

Problems in the Constraints". The first reference that used the notation "bilevel" was that of Candler and Norton [CN77] published in 1977.
For a fixed upper level decision x let us define the corresponding *lower level feasible set* as

$$\Omega(x) := \{y \in \mathbb{R}^{n_y} | g(x,y) \leq 0\}$$

Based on this feasible set, one can define the *rational reaction set* for a given upper level decision x (i.e. the set of optimal lower level decisions for a given upper level decision) as

$$R(x) := \{y \in \mathbb{R}^{n_y} | y \in \arg\max\{f(x,\hat{y}) | \hat{y} \in \Omega(x)\}\}$$

The most important set for the understanding of bilevel programming problems is the so called *induced region*, i.e. the actual feasible set of the bilevel problem:

$$\mathcal{IR} := \{(x,y) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} | G(x,y) \leq 0, y \in R(x)\}$$

This set is usually nonconvex and might even be disconnected or empty.
Most of the references on bilevel programming problems treat so called *Convex Bilevel problems* (CBL). CBL's are bilevel problems where the follower's problem is a convex optimization problem. Note that this property does not imply the convexity of the induced region. A great advantage of CBLs is, however, that they can often be equivalently reformulated as a single level optimization problem by making use of the Karush-Kuhn-Tucker conditions for the follower's problem. Clearly, to apply this approach some additional regularity condition(s) need to be satisfied (see for example [Dem02]).
The most simple variant of bilevel problems is the general linear bilevel problem ($GLBP$). In this problem both objective functions depend linearly on the decision vectors, all constraints are linear and the decision variables are continuous. However, even this simplest problem variant is strongly $\mathcal{NP}$-hard (by a reduction from the linear minmax problem, see [FP01],[HJS92]) and many combinatorial optimization can be equivalently rewritten as a $GLBP$ ([MS05]). The $GLBP$ can be stated as follows:

$$\begin{aligned}
(\text{GLBP}) \quad & \max_{x \in \mathbb{R}^{n_x}} \quad c_1^t x + d_1^t y && (5.2a)\\
& \text{s.t. } A^1 x + B^1 y \leq b^1, && (5.2b)\\
& \quad x \geq 0, && (5.2c)\\
& \quad y \in \arg\max_{y \in \mathbb{R}^{n_y}} c_2^t x + d_2^t y, && (5.2d)\\
& \quad \text{s.t. } A^2 x + B^2 y \leq b^2, && (5.2e)\\
& \quad\quad y \geq 0. && (5.2f)
\end{aligned}$$

The $GLBP$ is probably the most studied variant of bilevel programming problems. It can be shown that the $GLBP$ is an optimization problem that consists of maximizing the linear function $c_1^t x + d_1^t y$ over a piecewise linear constraint region (see [Bar84]). It is thus clear that the $GLBP$ is generally nonconvex.
Another particular and well-studied variant of bilevel programming problems are *linear-quadratic bilevel problems* that consist of a linear upper and (convex) quadratic lower level

problem.
More details about bilevel programming can be found in the books by Shimizu et al. [SIB97], Bard [Bar98], the most recent book on bilevel programming by Stephan Dempe [Dem02] as well as in the several surveys on bilevel programming, e.g. [VC94], [Dem03] or [CMS07].

5.1.2 Stochastic Bilevel Problems and Bilevel Problems with Knapsack Constraint

In this chapter we will consider a linear bilevel problem that, in the upper level, contains a knapsack chance-constraint of the following form:

$$\mathbb{P}\{w^t(\chi)x \leq s(\chi)\} \geq (1-\alpha) \tag{5.3}$$

where χ is a random vector.
Single level problems with knapsack chance-constraint have been studied in chapter 3. The interpretation of the chance-constraint is the same in both chapters: We are given a (deterministic or random) resource $s(\chi)$ (for example an available capacity) as well as a set of demands $w^t(\chi)_i$ (for example items to pack in a knapsack or routing demands in a network). Due to the randomness of the demands (and eventually the resource) we cannot be sure if our decision x will respect the knapsack constraint $w^t(\chi^k)x \leq s(\chi^k)$ in case χ has the outcome χ^k. The assumption that a violation of the knapsack constraint is acceptable in at most a fraction α of all possible scenarios can, however, be modeled by the chance-constraint (5.3). Introducing a chance-constraint in the upper level of a bilevel problem thus models the case where the decision of the leader has to be made under uncertainty of certain demands (for example of the follower). To the best of our knowledge this is the first work on a chance-constrained bilevel problem or a stochastic bilevel problem with knapsack constraint and random parameters in the leader's problem.
Dempe and Richter [DR00] studied in their paper a deterministic bilevel problem with knapsack constraint. In their version of the problem, the decision of the leader consists in choosing the (one-dimensional) right hand side of the knapsack constraint (i.e. the capacity of the knapsack). Depending on this value, the follower has to solve a common knapsack problem. Both capacity and lower level decision appear linearly in the leader's objective function. The difficulty of this problem lies in the fact that even a slight perturbation of the right hand side of a knapsack constraint in an optimization problem can lead to an optimal solution that is arbitrary far from any optimal solution of the initial problem. Recently Brotcorne et al. [BHM09] studied the same problem. They show that by using a dynamic programming approach the bilevel knapsack problem can be solved with a surprisingly not much greater time requirement than needed to solve a common single level knapsack problem.
The study of stochastic bilevel problems is rather recent and only few references are available that treat this special case of stochastic optimization problems.
In 1997 Patriksson and Wynter wrote a technical report on "Stochastic nonlinear bilevel programming" [PW97]. In their work, they consider the follower's problem to contain

random parameters in the objective. The problem is modeled as a two-stage problem. In [PW99] the authors extended their work to the more general case of "Stochastic mathematical programs with equilibrium constraints". They study the feasibility of the lower level problem as well as convexity properties.
A complementary point of view is taken in [Wer04]. Instead of extending a deterministic bilevel problem to a stochastic one by assuming some of the parameters to be uncertain (*stochastic bilevel program*), the authors point out that the follower's decision can be seen as a special kind of uncertainty to the leader's problem (*stochastic program with bilevel structure*). Additional random parameters might occur in the upper and/or lower level problem. As an example, they study the case where the upper level problem is a two-stage problem whose second stage is, however, independent of the follower's decision. The lower level problem is a single-stage stochastic optimization problem with expected value objective function.
A very recent publication concerning stochastic bilevel problems is that of Özaltin et al. [OPS10]. The authors solve a stochastic version of the bilevel knapsack problem aforementioned (see [DR00],[BHM09]). As in the works by Patriksson and Wynter the authors of [OPS10] assume the uncertainty to occur in the lower level problem. More precisely, they assume that the right hand side of the knapsack constraint in the lower level does not only depend on the leader's decision but also on a random vector. Both upper and lower level decision variables are required to be binary.

5.1.3 Solving Bilevel Problems

Solution algorithms for bilevel problems can be divided in two classes: On the one hand, there are those algorithms that are designed to find the global optimum of the problem. Most of the problems that can be solved with these algorithms have a linear upper level and linear or convex quadratic lower level problem. On the other hand, there are those algorithms that can be used for bilevel programming problems with nonconvex lower level problem, but that might only find a local optimum.
For the special case of the $GLBP$ one can show that the set of optimal solutions (if nonempty) contains at least one vertex of the polyhedron $\{(x,y) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} | A^1x + B^1y \leq b^1, A^2x + B^2y \leq b^2\}$ (see [Sav89]). This property gave rise to several vertex enumeration algorithms such as presented in [CT82] and [BK84]. Note that linear-quadratic bilevel problems do not have this property.
If the lower level problem is convex and satisfies some suitable regularity conditions, the bilevel problem can be reformulated as an equivalent single level problem. The added complementary constraints, however, make the obtained problem hard to solve. To overcome this problem, a B&B algorithm might be considered that branches on these constraints: While in the left subtree an additional constraint is added that forces one of the lower level constraints to be satisfied with equality, one sets the corresponding dual variable to zero in the right subtree. One of the first references where this idea is applied to bilevel programming problems is [BF82]. While the authors of this reference only treat the case of linear bilevel problems, the method has been adapted to the quadratic ([Bar88]) and

linear-quadratic ([BM90]) case in subsequent publications.
Another type of algorithm for bilevel problems that is based on the single level reformulation is the complementary pivot algorithm. Complementary pivot algorithms are known from linear programming and serve to solve linear complementary problems. The name of the algorithm directly indicates its functioning: Complementary pivot algorithms are pivot algorithms like the simplex algorithm. However, the exchange of basic variables follows a special structure: The leaving variable is always the complement of the entering variable. Applying a complementary pivot algorithm to the single level reformulation of the $GLCP$ (that has been obtained by adding the primal, dual and complementary slackness constraints of the follower's problem to the upper level) is thus a natural idea. The first such algorithm for bilevel programming was proposed by Bialas et al. [BKS80] in 1980. However, the authors admitted in a subsequent paper that the proposed algorithm might not find an exact solution of the problem in all cases (see [BK84] and the given (counter-)example in [BAB90]). Later Júdice and Faustino proposed an algorithm based on the same principle of complementary pivoting for linear ([JF88],[JF88]) and linear-quadratic bilevel problems ([JF94]).
Contrary to the above algorithms, descent and penalty function methods are generally only capable of finding local optima or stationary points.
The problem of applying descent methods to bilevel problems lies in the $\mathcal{NP}$-hardness of determining the steepest decent direction, even in case of linear bilevel programming problems (see [OZ95]).
Some of the decent direction approaches proposed in the literature rely on the idea of computing a gradient (or subgradient) of the so called *implicit function* $y(x)$, that, for the BP, is defined as

$$y(x) = \arg\max\{f(x, \hat{y}) | \hat{y} \in \Omega(x)\}$$

Knowing the gradient of the implicit function, the gradient of the objective function of the BP is given by

$$\nabla_x F(x, y(x)) = \nabla_x F(x, y) + \nabla_y F(x, y) \nabla_x y(x)$$

An approximation for this gradient has been proposed in [KL90].
The main idea of penalty function methods is to include the upper level constraints into the upper level objective function and the lower level constraints into the lower level objective function using an exact penalty factor (see e.g. [IA92]). Variants work with the Karush-Kuhn-Tucker single level reformulation ([LHZ01]) or use the duality gap of the lower level problem as penalty function in the upper level objective (see for example [WA93] or [MZ96]).

5.1.4 Solving Stochastic Bilevel Problems

As abovementioned, only few references on stochastic bilevel programs are available. In [PW99] the authors study a nonlinear stochastic bilevel problem with uncertainties in the lower level problem that are handled with a two-stage model. The authors point out that

the problem is in general nonconvex and thus propose a heuristic gradient descent based algorithm that searches for local minima.
In [Wer04] the authors present a method to solve a stochastic bilevel problem with recourse in the upper level. They first replace the lower level problem by adding the corresponding Karush-Kuhn-Tucker optimality conditions to the upper level program (under the needed assumptions). The obtained problem is then solved using a variant of the Stochastic Arrow-Hurwicz algorithm (see chapters 2 and 3): The Karush-Kuhn-Tucker conditions are included in the objective by introducing Lagrange multipliers. The so obtained Lagrangian dual (with recourse) is then solved using a stochastic gradient algorithm. Upper level feasibility is assured using a projection method. The second-stage problem is solved at each iteration for an independently drawn sample of the random variables.
In [OPS10] the authors solve a stochastic bilevel problem with knapsack constraint. Contrary to our model, the knapsack constraint is part of the lower level problem and only the capacity is assumed to be random (and dependent on the leader's decision), not the item weights. Moreover, the knapsack constraint is the only constraint in the lower level. The authors first develop a branch-and-backtrack algorithm that evaluates the lower level solution for a given right hand side. Their first approach to solve the overall problem is to evaluate the lower level problem for all the possible outcomes of the knapsack capacity. As an alternative for this exhaustive search they propose a branch-and-cut algorithm based on the L-shaped algorithm for stochastic problems with integer recourse and binary first-stage decisions by Laporte and Louveau ([LL93]) (see also the introduction of chapter 4). The authors show that using their branch-and-backtrack algorithm to solve lower level problems is much more efficient than using a commercial solver for linear integer problems.
Throughout this chapter, we assume the probability distributions of the random variables to be discrete. In case of continuous probability distributions, our assumption and the resulting problem reformulations might nevertheless be helpful as one could approximate the probability space by generating a finite number of representative scenarios. From a theoretical point of view, assuming a finite sample space has the advantage that the initial chance-constrained bilevel problem can be reformulated as a deterministic equivalent problem by treating the constraints for every scenario separately. In our case this leads to a mixed integer linear bilevel problem ($MILBP$).
$MILBP$s have been studied in [VSJ96] and the authors show that in case the integer variables only appear in the upper level, the problem can be equivalently reformulated as a linear bilevel problem (LBP). In [FAM81] the authors show how to further convert an LBP into a single level, bilinear program. We apply both techniques to our problem. However, our aim is not to solve the resulting quadratic problem exactly, which could for example be achieved by reformulating the problem as an MIP (see [FAM81]) or by using other quadratic programming methods. Instead, we propose upper bounds by relaxing the bilinear constraints in the objective function. This results in a bilinear minmax problem of the following form: The constraints are linear and separable, i.e. there exists a partition of the variables that also partitions the constraints into two disjoint sets. When fixing one of these sets of variables, the resulting problem is either a linear minimization,

or maximization problem.
Bilinear problems of this type have been studied in the literature (see [HT96]). However, to the best of our knowledge all these studies concerned pure minimization (resp. maximization) problems known as *bilinear programming problems with disjoint (or separable) constraints*. Due to the structure of these problems it is obvious why the decomposition of the initial problem is the basis of most solution methods. By simply iteratively solving the two subproblems (or the primal of one and the dual of the other) one, however, cannot ensure global convergence. The most common approaches to guarantee that the algorithm converges towards a global maximum are cutting plane ([Kon76],[SS80]) and B&B ([Fal73],[AKF83]) methods (for a survey see e.g. [AK90] or [FV94]). A more detailed overview will be given in section 5.8.1.
As in our case one of the subproblems is a maximization, while the other is a minimization problem, we propose a minmax scheme that alternately solves the two problems. The convergence to the global optimum of the bilinear minmax problem is ensured by continuously introducing new cuts to both problems. To the best of our knowledge, such an algorithm has not yet been proposed in the literature.
There has been an extensive research on minmax problems. A paper that treats a minmax problem with a similar structure to our problem is [Pan93]: As in our case, the studied minmax problem has separable linear constraints and a quadratic objective function. The solution procedure proposed is a finite iteration method that, at each iteration, solves a quadratic subproblem. Although we cannot guarantee that our algorithm finds an optimal solution in a finite number of iterations (we can only guarantee an ϵ-optimal solution after finitely many steps), its advantages are the simple structure and the fact that we have to solve two linear programming problems per iteration, instead of one quadratic.

5.2 Mathematical formulation and an illustrative example

The problem that we study in this chapter has the following form: In the upper level the leader wants to maximize its benefits that are linear in the upper as well as lower level decision variables. The upper level constraint set contains linear constraints as well as one single knapsack chance-constraint. While the former depend on both upper and lower level variables, the latter only depends on the upper level decision variables. The methods and results presented in this chapter are however easily adaptable to the case where the chance-constraint also contains the follower's decision variables. In addition to the coefficients that multiply the decision variables in the knapsack constraint ("weights"), we assume the right hand side ("capacity") to be random, as well. All random variables depend on a random vector χ that is assumed to only have a finite number of possible

outcomes.

$$
\begin{aligned}
(\text{SLBP}) \quad & \max_{x} \quad c_1^t x + d_1^t y && (5.4a)\\
& \text{s.t. } A^1 x + B^1 y \leq b^1, && (5.4b)\\
& \quad \mathbb{P}\{w^t(\chi) x \leq s(\chi)\} \geq (1-\alpha), && (5.4c)\\
& \quad \mathbb{O}_{n_x} \leq x \leq \mathbb{1}_{n_x}, && (5.4d)\\
& \quad y \in \arg\max_{y} c_2^t x + d_2^t y, && (5.4e)\\
& \quad \text{s.t. } A^2 x + B^2 y \leq b^2, && (5.4f)\\
& \quad\quad y \geq 0. && (5.4g)
\end{aligned}
$$

where $c_1, c_2, \in \mathbb{R}^{n_x}$, $d_1, d_2 \in \mathbb{R}^{n_y}$, $A^1 \in \mathbb{R}^{m_1 \times n_x}$, $B^1 \in \mathbb{R}^{m_1 \times n_y}$, $b^1 \in \mathbb{R}^{m_1}$, $A^2 \in \mathbb{R}^{m_2 \times n_x}$, $B^2 \in \mathbb{R}^{m_2 \times n_y}$, $b^2 \in \mathbb{R}^{m_2}$ and $0 < \alpha \leq 1$.
Let us now illustrate this general model with a real-world application: A mobile phone or internet provider grants capacities on a network to multiple customers, who maximize their profit (5.4e) while routing their demand on the provider's network (5.4f). The follower's maximization reduces to choosing between the leader and the competition. Each client is entitled to a given bandwidth, which is modeled by x. The leader charges clients so as to obtain an optimal trade-off between the number of clients and prices (5.4a). To do so, the leader must take into account the prices of the competition but also the capacity $s(\chi)$ of its network. However, typically, customers use only a fraction $w(\chi)$ of the granted capacity. Therefore it is often the case that the network is not fully utilized. To further optimize the load of the network, it is possible to use stochastic knapsack constraints (5.4c) to allow an *overbooked* network with a given risk of *overload*.

5.3 From the $SLBP$ to the (Deterministic Equivalent) Linear Bilevel Problem (LBP)

Let $\chi_1, ..., \chi_K$ denote the K outcomes of the discretely distributed random vector χ and define $p^k := \mathbb{P}\{\chi = \chi_k\}$ with

$$\sum_{k=1}^{K} p_k = 1, \quad p_k > 0.$$

For each scenario χ_k $(k = 1, \ldots, K)$ we introduce an auxiliary binary variable z_k as follows:

$$z_k = \begin{cases} 0 & \text{if the scenario is considered} \\ 1 & \text{otherwise} \end{cases}$$

We shall simplify the notations by defining for all $k = 1, \ldots, K$:

$$w_k := w\left(\chi_k\right), \; s_k := s\left(\chi_k\right), \; w^k := \left(w_1^k, ..., w_n^k\right)$$

For all $k = 1, \dots, K$, we define M_k such that

$$M_k := \sum_{i=1}^{n_x} w_i^k - s_k$$

Thus, the $SLBP$ can be reformulated as the following mixed integer optimization problem:

$$\begin{aligned}
(\text{MILBP}) \quad & \max_{x,z} \quad c_1^t x + d_1^t y \\
& \text{s.t. } A^1 x + B^1 y \leq b^1, && (5.5a)\\
& \quad w_k^t x \leq s_k + M_k z_k \quad \forall\, k = 1, \dots, K, && (5.5b)\\
& \quad p^t z \leq \alpha, && (5.5c)\\
& \quad \mathbb{0}_{n_x} \leq x \leq \mathbb{1}_{n_x}, && (5.5d)\\
& \quad z \in \{0,1\}^K, && (5.5e)\\
& \quad y \in \arg\max_y c_2^t x + d_2^t y, && (5.5f)\\
& \quad \text{s.t. } A^2 x + B^2 y \leq b^2, && (5.5g)\\
& \qquad y \geq 0. && (5.5h)
\end{aligned}$$

Constraints (5.5b) ensure that, if scenario χ_k is not covered (i.e. $z_k = 1$), then the adopted strategy x does not have to respect the knapsack constraint for this scenario. However, as per constraint (5.5c), the probability of occurrence of the uncovered scenarios must be below the risk α.
Note that

$$w_k^t x \leq s_k + M z_k \quad \forall\, k = 1, \dots, K$$

with $M = \max_{k=1,\dots,K} M_k$ is a more frequent manner than (5.5b) to rewrite knapsack chance-constraints. However, allowing the M_k values to be different for constraints (5.5b) yields tighter LP-relaxations (see also [LLH09]).
As shown in [VSJ96], we can now reformulate the mixed integer bilevel problem (5.5) as

a linear bilevel problem:

$$
\begin{aligned}
\text{(LBP)} \quad & \max_{x,z} \quad c_1^t x + d_1^t y \\
& \text{s.t. } A^1 x + B^1 y \leq b^1, && \text{(5.6a)} \\
& \quad w_k^t x \leq s_k + M_k z_k \quad \forall\, k = 1, \ldots, K, && \text{(5.6b)} \\
& \quad p^t z \leq \alpha, && \text{(5.6c)} \\
& \quad \mathbb{0}_{n_x} \leq x \leq \mathbb{1}_{n_x}, && \text{(5.6d)} \\
& \quad \mathbb{0}_K \leq z \leq \mathbb{1}_K, && \text{(5.6e)} \\
& \quad v = 0, && \text{(5.6f)} \\
& \quad (y, v) \in \arg\max_{y,v} c_2^t x + d_2^t y + (\mathbb{1}_K)^t v, && \text{(5.6g)} \\
& \quad \text{s.t. } A^2 x + B^2 y \leq b^2, && \text{(5.6h)} \\
& \qquad v \leq z, && \text{(5.6i)} \\
& \qquad v \leq \mathbb{1}_K - z, && \text{(5.6j)} \\
& \qquad y \geq 0. && \text{(5.6k)}
\end{aligned}
$$

where $\dim(v) = \dim(z) = K$. The term $(\mathbb{1}_K)^t v$ in the lower level objective function forces v to be equal to $\min(1_K - z, z)$ (see Proposition 5.2). Note that the vector $\mathbb{1}_K$ could be replaced by any vector with strictly positive components.

Definition 5.1. *We denote $(\tilde{x}, \tilde{z}, \tilde{y})$ (resp. $(\tilde{x}, \tilde{z}, \tilde{y}, \tilde{v})$) a feasible solution for problem (5.5) (resp. problem (5.6)) if all upper and lower level constraints are satisfied. A rational solution of problem (5.5) (resp. (5.6)) is a feasible solution such that $\tilde{y}$ (resp. $(\tilde{y}, \tilde{v})$) is optimal for the lower level problem with parameters $\tilde{x}$ and $\tilde{z}$.*

Proposition 5.2 (see Proposition 3.2. of [AHJS97])**.**

1.) *Let (x^*, z^*, y^*, v^*) be a rational optimal solution of the LBP (5.6).*
Then $v^* = 0$ and (x^*, z^*, y^*) is a rational optimal solution of the MILBP (5.5).

2.) *Let (x^*, z^*, y^*) be a rational optimal solution of the MILBP (5.5).*
Then $(x^*, z^*, y^*, 0)$ is a rational optimal solution of the LBP (5.6).

5.4 From the LBP to the Global Linear Complementarity Problem ($GLCP$)

We will now continue the transformation process by reformulating the LBP as a single level quadratic problem as described in [AHJS97]. The idea is to replace the lower level problem by a set of constraints that contain (i) the initial constraints of the lower level problem and (ii) the complementary slackness conditions of the lower level problem. The latter ensures that an optimal solution of the obtained single level problem is also optimal for the LBP. The decision vectors of the new problem are both the decision vectors of

the upper and lower level problems as well as the dual variables of the latter.
Let us first state the dual of the follower's problem (5.6g)-(5.6k):

$$\begin{aligned}
(\text{DFP}) \min_{\lambda,\mu_1,\mu_2} \quad & \lambda^t(b^2 - A^2x) + \mu_1 z + \mu_2(\mathbb{1}_K - z), \\
\text{s.t.} \quad & (B^2)^t\lambda \geq d_2, && (5.7a)\\
& \mathbb{I}_K\mu_1 + \mathbb{I}_K\mu_2 \geq \mathbb{1}_K, && (5.7b)\\
& \lambda, \mu_1, \mu_2 \geq 0. && (5.7c)
\end{aligned}$$

where $\lambda \in \mathbb{R}^{m_2}$ (resp. $\mu_1 \in \mathbb{R}^K$, $\mu_2 \in \mathbb{R}^K$) is the dual decision vector associated with (5.6h) (resp. (5.6i), (5.6j)). We also need the corresponding complementary slackness conditions to ensure the optimality of the DFP:

$$\begin{aligned}
\lambda^t(b^2 - A^2x - B^2y) = 0 \qquad & \qquad y^t((B^2)^t\lambda - d_2) = 0 \\
\mu_1^t(z - v) = 0 \qquad & \qquad v^t(\mathbb{I}_K\mu_1 + \mathbb{I}_K\mu_2 - \mathbb{1}_K) = 0 \\
\mu_2^t(\mathbb{1}_K - z - v) = 0 \qquad &
\end{aligned}$$

We obtain the following equivalent Global Linear Complementarity problem ($GLCP$) which is no longer a bilevel problem [AHJS97]:

$$\begin{aligned}
(\text{GLCP}) \quad \max_{x,y,z,\lambda,\mu_1,\mu_2} \quad & c_1^t x + d_1^t y \\
\text{s.t.} \quad & A^1x + B^1y \leq b^1, && (5.8a)\\
& w_k^t x \leq s_k + M_k z_k \quad \forall\, k = 1, \ldots, K, && (5.8b)\\
& p^t z \leq \alpha, && (5.8c)\\
& A^2x + B^2y \leq b^2, && (5.8d)\\
& (B^2)^t\lambda \geq d_2, && (5.8e)\\
& \mathbb{I}_K\mu_1 + \mathbb{I}_K\mu_2 \geq \mathbb{1}_K, && (5.8f)\\
& \lambda^t(b^2 - A^2x - B^2y) = 0, && (5.8g)\\
& \mu_1^t z = 0, && (5.8h)\\
& \mu_2^t(\mathbb{1}_K - z) = 0, && (5.8i)\\
& y^t((B^2)^t\lambda - d_2) = 0, && (5.8j)\\
& \mathbb{O}_{n_x} \leq x \leq \mathbb{1}_{n_x}, \mathbb{O}_K \leq z \leq \mathbb{1}_K, && (5.8k)\\
& y, \lambda, \mu_1, \mu_2 \geq 0. && (5.8l)
\end{aligned}$$

Note that in this formulation the decision variable v has been eliminated due to the fact that $v = 0$.
All reformulations are equivalent so far, i.e. by solving the quadratic problem (5.8) we get an optimal solution to the initial stochastic bilevel problem (5.5) (provided that the probability space is discrete). Solving a generally nonconvex problem such as (5.8) directly is hard. Instead, we propose a method to compute upper bounds by relaxing it into a linear minmax problem.

5.5 Calculating upper bounds

We relax the quadratic terms (5.8g), (5.8h), (5.8i) and (5.8j) of the $GLCP$ into the objective function:

$$\mathcal{L}(x, y, z, \lambda, \mu_1, \mu_2) = c_1^t x + d_1^t y + \lambda^t(b^2 - A^2 x - B^2 y) + \mu_1^t z + \mu_2^t(\mathbb{1}_K - z) + y^t((B^2)^t\lambda - d_2)$$

Then the Lagrangian relaxation of the $GLCP$ (5.8) becomes

$$\begin{aligned}
(\text{LGN}) \quad \min_{\lambda,\mu_1,\mu_2} \max_{x,y,z} \quad & \mathcal{L}(x, y, z, \lambda, \mu_1, \mu_2) \\
\text{s.t.} \quad & A^1 x + B^1 y \leq b^1, && (5.9a)\\
& w_k^t x \leq s_k + M_k z_k \quad \forall\, k = 1, \ldots, K, && (5.9b)\\
& p^t z \leq \alpha, && (5.9c)\\
& A^2 x + B^2 y \leq b^2, && (5.9d)\\
& (B^2)^t \lambda \geq d_2, && (5.9e)\\
& \mathbb{I}_K \mu_1 + \mathbb{I}_K \mu_2 \geq \mathbb{1}_K, && (5.9f)\\
& \mathbb{O}_{n_x} \leq x \leq \mathbb{1}_{n_x}, \mathbb{O}_K \leq z \leq \mathbb{1}_K, && (5.9g)\\
& y, \lambda, \mu_1, \mu_2 \geq 0. && (5.9h)
\end{aligned}$$

Proposition 5.3. *Let $(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$ be an optimal solution of the LGN. Then $\mathcal{L}(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$ is an upper bound on the optimal solution value of the GLCP (5.8).*

Proof. Let $(\widetilde{x}, \widetilde{y}, \widetilde{z}, \widetilde{\lambda}, \widetilde{\mu}_1, \widetilde{\mu}_2)$ denote an optimal solution of the $GLCP$ (5.8). As

$$\begin{aligned}
b^2 - A^2\widetilde{x} - B^2\widetilde{y} &\geq 0\\
(B^2)^t\lambda^* - d_2 &\geq 0\\
\widetilde{y}, \lambda^*, \mu_1^*, \mu_2^* &\geq 0\\
0 \leq \widetilde{z} &\leq 1
\end{aligned}$$

we have

$$c_1^t\widetilde{x} + d_1^t\widetilde{y} \leq \mathcal{L}(\widetilde{x}, \widetilde{y}, \widetilde{z}, \lambda^*, \mu_1^*, \mu_2^*)$$

and as $(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$ is optimal for the LGN we also have

$$\mathcal{L}(\widetilde{x}, \widetilde{y}, \widetilde{z}, \lambda^*, \mu_1^*, \mu_2^*) \leq \mathcal{L}(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$$

□

The LGN has the nice property that by fixing either the primal or the dual variables, we obtain a linear problem. This property gives rise to the idea to decompose the LGN and to apply an iterative minmax scheme to practically solve the relaxation. More precisely, at iteration $N \geq 1$ we solve the following two linear problems:

- The Lagrangian subproblem LGNs(N), maximized over the primal variables.
- Problem LGNd(N), which is mainly composed of the DFP (5.7) (with additional constraints).

At each iteration of the scheme, an auxiliary constraint is added to both problems in order to enforce the convergence of their optimal solution values towards the optimal solution value of the LGN. Remark that the so obtained decrease (resp. increase) of the objective function value of the $LGNs$ (resp. $LGNd$) is only monotonic. In section 5.6 this matter is further discussed and we propose a method to obtain even strict convergence at each iteration.
The iteration process stops when $\beta - \gamma < \delta$ or $(\beta - \gamma)/\beta < \epsilon$ for small $\delta > 0$ and $\epsilon > 0$.

(LGNs(N))

$$\max_{\beta,x,y,z} \quad \beta$$

$$\begin{aligned}
\text{s.t. } & A^1x + B^1y \leq b^1, \\
& \beta \leq \mathcal{L}(x, y, z, \lambda^q, \mu_1^q, \mu_2^q) \\
& \qquad \forall\, q = 0, \ldots, N-1, && \text{(5.10a)} \\
& w_k^t x \leq s_k + M_k z_k \\
& \qquad \forall\, k = 1, \ldots, K, && \text{(5.10b)} \\
& p^t z \leq \alpha, && \text{(5.10c)} \\
& A^2x + B^2y \leq b^2, && \text{(5.10d)} \\
& \mathbb{0}_{n_x} \leq x \leq \mathbb{1}_{n_x}, && \text{(5.10e)} \\
& \mathbb{0}_K \leq z \leq \mathbb{1}_K, && \text{(5.10f)} \\
& y \geq 0. && \text{(5.10g)}
\end{aligned}$$

(LGNd(N))

$$\min_{\gamma,\lambda,\mu_1,\mu_2} \quad \gamma$$

$$\begin{aligned}
\text{s.t. } & \gamma \geq \mathcal{L}(x^q, y^q, z^q, \lambda, \mu_1, \mu_2) \\
& \qquad \forall\, q = 1, \ldots, N, && \text{(5.11a)} \\
& (B^2)^t \lambda \geq d_2, && \text{(5.11b)} \\
& \mathbb{I}_K \mu_1 + \mathbb{I}_K \mu_2 \geq \mathbb{1}_K, && \text{(5.11c)} \\
& \lambda, \mu_1, \mu_2 \geq 0. && \text{(5.11d)}
\end{aligned}$$

where $N \geq 1$ is the iteration number, (x^q, y^q, z^q) is an optimal solution of problem $LGNs(q)$ $(q = 1, \ldots, N)$, $(\lambda^q, \mu_1^q, \mu_2^q)$ is feasible for the LGN if $q = 0$ and it is an optimal solution of problem $LGNd(q)$ if $q \geq 1$. In the reminder of this chapter we will use the following notation:

N1: $X \subseteq \mathbb{R}_+^{n_x+n_y+K}$ denotes the set of triples (x, y, z) feasible for the LGN

N2: $\Lambda \subseteq \mathbb{R}_+^{m_2+2K}$ denotes the set of triples (λ, μ_1, μ_2) feasible for the LGN

N3: (β^N, x^N, y^N, z^N) denotes an optimal solution of problem $LGNs(N)$

N4: $(\gamma^N, \lambda^N, \mu_1^N, \mu_2^N)$ denotes an optimal solution of problem $LGNd(N)$

Using the first two notations, problems (5.10) and (5.11) of the iterative minmax scheme can be stated equivalently as

$$(LGNs(N)) \qquad \max_{(x,y,z)\in X} \min_{q\in\{0,\dots,N-1\}} \mathcal{L}(x,y,z,\lambda^q,\mu_1^q,\mu_2^q)$$

$$(LGNd(N)) \qquad \min_{(\lambda,\mu_1,\mu_2)\in\Lambda} \max_{q\in\{1,\dots,N\}} \mathcal{L}(x^q,y^q,z^q,\lambda,\mu_1,\mu_2)$$

We directly get the following properties:

P1: $\beta^N = \max_{(x,y,z)\in X} \min_{q\in\{0,\dots,N-1\}} \quad \mathcal{L}(x,y,z,\lambda^q,\mu_1^q,\mu_2^q)$

P2: $\beta^N = \min_{q\in\{0,\dots,N-1\}} \mathcal{L}(x^N,y^N,z^N,\lambda^q,\mu_1^q,\mu_2^q)$

P3: $\exists\, j_N \in \{0\dots N-1\}$ s.t. $\beta^N = (x^N,y^N,z^N,\lambda^{j_N},\mu_1^{j_N},\mu_2^{j_N})$

P4: $\gamma^N = \min_{(\lambda,\mu_1,\mu_2)\in\Lambda} \max_{q\in\{1,\dots,N\}} \quad \mathcal{L}(x^q,y^q,z^q,\lambda,\mu_1,\mu_2)$

P5: $\gamma^N = \max_{q\in\{1,\dots,N\}} \mathcal{L}(x^q,y^q,z^q,\lambda^N,\mu_1^N,\mu_2^N)$

P6: $\exists\, i_N \in \{1,\dots,N\}$ s.t. $\gamma^N = \mathcal{L}(x^{i_N},y^{i_N},z^{i_N},\lambda^N,\mu_1^N,\mu_2^N)$

In the following, we will keep the notations j_N and i_N introduced in **P3** and **P6**.

5.5.1 Proving upper and lower bounds

Problems $LGNs(N)$ and $LGNd(N)$ provide upper and lower bounds on the minmax relaxation (5.9), respectively:

Lemma 5.4. *Let $N \geq 1$. Then γ^N is a lower bound on the optimal solution value of the LGN.*

Proof. As $(x^q,y^q,z^q) \in X$ for all $q = 1,\dots,N$, we have for all $(\lambda,\mu_1,\mu_2) \in \Lambda$

$$\max_{(x,y,z)\in X} \mathcal{L}(x,y,z,\lambda,\mu_1,\mu_2) \geq \max_{q=1,\dots,N} \mathcal{L}(x^q,y^q,z^q,\lambda,\mu_1,\mu_2)$$

It follows

$$\min_{(\lambda,\mu_1,\mu_2)\in\Lambda} \max_{(x,y,z)\in X} \mathcal{L}(x,y,z,\lambda,\mu_1,\mu_2) \geq \min_{(\lambda,\mu_1,\mu_2)\in\Lambda} \max_{q=1,\dots,N} \mathcal{L}(x^q,y^q,z^q,\lambda,\mu_1,\mu_2) = \gamma^N$$

which proves the lemma. □

Lemma 5.5. *Let $N \geq 1$. Then β^N is an upper bound on the optimal solution value of the LGN.*

Proof. As $\mathcal{L}(x, y, z, \cdot, \cdot, \cdot)$ is linear for all $(x, y, z) \in X$ and $\mathcal{L}(\cdot, \cdot, \cdot, \lambda, \mu_1, \mu_2)$ is linear for all $(\lambda, \mu_1, \mu_2) \in \Lambda$ and X as well as Λ are compact convex sets, it follows by von Neumanns's minimax theorem ([vN28],[Sio58]) that

$$\min_{(\lambda,\mu_1,\mu_2)\in\Lambda} \max_{(x,y,z)\in X} \mathcal{L}((x, y, z, \lambda, \mu_1, \mu_2) = \max_{(x,y,z)\in X} \min_{(\lambda,\mu_1,\mu_2)\in\Lambda} \mathcal{L}(x, y, z, \lambda, \mu_1, \mu_2)$$

As

$$\beta^N = \max_{(x,y,z)\in X} \min_{q=0,\ldots,N-1} \mathcal{L}(x, y, z, \lambda^q, \mu_1^q, \mu_2^q) \geq \max_{(x,y,z)\in X} \min_{(\lambda,\mu_1,\mu_2)\in\Lambda} \mathcal{L}(x, y, z, \lambda, \mu_1, \mu_2)$$

the lemma is proved. □

As a direct consequence of Proposition 5.3 and Lemma 5.5 we get that the optimal solution value of problem $LGNs(N)$ not only provides an upper bound on the optimal solution value of the LGN but also on the optimal solution value of the $GLCP$ (5.8) and thus of the initial $SLBP$ (5.4) in case of discretely distributed random variables:

Corollary 5.6. *Let $N \geq 1$. Then β^N is an upper bound on the optimal solution value of the GLCP (5.8).*

5.5.2 Stopping criteria

We define an absolute and a relative stopping criterion. Given $\delta > 0$ and $\epsilon > 0$, the iterative scheme stops if the absolute error is less than δ, i.e.

$$\beta^N - \gamma^N < \delta \text{ or } \beta^N - \gamma^{N-1} < \delta$$

or if the relative error is less than ϵ, i.e.

$$\frac{(\beta^N - \gamma^N)}{\beta^N} < \epsilon \text{ or } \frac{(\beta^N - \gamma^{N-1})}{\beta^N} < \epsilon$$

But does this case automatically arise when one of the problems have found an optimal solution? Or, more precisely, can we immediately detect that an optimal solution has been found? The answer is yes:

Lemma 5.7. *Let $N \geq 0$ and suppose that γ^N is the optimal solution value of the LGN. Then β^{N+1} is also optimal with corresponding solution vector $(x^{N+1}, y^{N+1}, z^{N+1}, \lambda^N, \mu_1^N, \mu_2^N)$.*

Proof. If γ^N is optimal for the LGN, we have

$$(\lambda^N, \mu_1^N, \mu_2^N) = \arg \min_{(\lambda,\mu_1,\mu_2)\in\Lambda} \left(\max_{(x,y,z)\in X} \mathcal{L}(x, y, z, \lambda, \mu_1, \mu_2) \right)$$

and

$$\gamma^N = \max_{(x,y,z)\in X} \mathcal{L}(x, y, z, \lambda^N, \mu_1^N, \mu_2^N)$$

As $\beta^{N+1} \leq \mathcal{L}(x^{N+1}, y^{N+1}, z^{N+1}, \lambda^q, \mu_1^q, \mu_2^q)$ for all $q \leq N$ by constraints (5.10a), we get

$$\beta^{N+1} \leq \max_{(x,y,z)\in X} \mathcal{L}(x, y, z, \lambda^N, \mu_1^N, \mu_2^N) = \gamma^N$$

It has been shown in Lemmata 5.4 and 5.5 that γ^N is a LB and β^{N+1} an UB on the optimal solution value of the LGN. It follows $\beta^{N+1} = \gamma^N$ which terminates the proof. □

Lemma 5.8. *Let $N \geq 0$ and suppose that β^N is the optimal solution value of the LGN. Then γ^N is also optimal with corresponding solution vector $(x^N, y^N, z^N, \lambda^N, \mu_1^N, \mu_2^N)$.*

Proof. The proof is similar to the previous one. □

Therefore, we know directly if one of the given solution values β^N or γ^N (and the corresponding solution vector) is optimal for the Lagrangian relaxation.

5.5.3 Convergence of the algorithm

In this subsection we will study the convergence of the minmax scheme. We will therefore suppose that the $GLCP$ (5.8) has a finite optimal solution value.

As at each iteration we add a new constraint to both problems, the following two lemmata follow directly:

Lemma 5.9. *Let $N \geq 0$. Then $\beta^{N+1} \leq \beta^N$.*

□

Lemma 5.10. *Let $N \geq 1$. Then $\gamma^{N+1} \geq \gamma^N$.*

□

Lemmata 5.11 and 5.12 show that the minmax scheme is acyclic, i.e. at each iteration we either find an optimal solution of the LGN or generate two new feasible triples (x, y, z) and (λ, μ_1, μ_2):

Lemma 5.11. *Let $N \geq 2$. Then either $(\lambda^N, \mu_1^N, \mu_2^N) \neq (\lambda^h, \mu_1^h, \mu_2^h)$ for all $1 \leq h < N$ or γ^N is the optimal solution value of the LGN.*

Proof. Let us assume that $(\lambda^N, \mu_1^N, \mu_2^N) = (\lambda^h, \mu_1^h, \mu_2^h)$ for a $h < N$. Then

$$\begin{aligned}
\gamma^N &= \max_{q\in\{1,\dots,N\}} \mathcal{L}(x^q, y^q, z^q, \lambda^N, \mu_1^N, \mu_2^N) \\
&\geq \mathcal{L}(x^{h+1}, y^{h+1}, z^{h+1}, \lambda^N, \mu_1^N, \mu_2^N) \\
&= \mathcal{L}(x^{h+1}, y^{h+1}, z^{h+1}, \lambda^h, \mu_1^h, \mu_2^h) \\
&\geq \min_{q\in\{0,\dots,h\}} \mathcal{L}(x^{h+1}, y^{h+1}, z^{h+1}, \lambda^q, \mu_1^q, \mu_2^q) = \beta^{h+1}
\end{aligned}$$

As γ^N is a lower bound on the LGN and β^{h+1} is an upper bound, it follows that $\gamma^N = \beta^{h+1}$ and γ^N (resp. β^{h+1}) is thus the optimal solution value of the Lagrangian relaxation. □

Lemma 5.12. *Let $N \geq 1$. Then either $(x^N, y^N, z^N) \neq (x^h, y^h, z^h)$ for all $0 \leq h < N$ or β^N is the optimal solution value of the LGN.*

Proof. The proof is similar to the previous one. □

5.6 Modified iterative minmax scheme

As the LGN is continuous, it might theoretically be possible that we get stuck on a nonoptimal solution value for infinitely many iterations, i.e. that there exists an $N^0 \geq 2$ such that either

$\mathcal{L}(x^{N^0}, y^{N^0}, z^{N^0}, \lambda^{N^0-1}, \mu_1^{N^0-1}, \mu_2^{N^0-1})$ is not optimal solution value of the LGN,
and for all $N \geq N^0$
$\mathcal{L}(x^N, y^N, z^N, \lambda^{N-1}, \mu_1^{N-1}, \mu_2^{N-1}) = \mathcal{L}(x^{N^0}, y^{N^0}, z^{N^0}, \lambda^{N^0-1}, \mu_1^{N^0-1}, \mu_2^{N^0-1})$

or

$\mathcal{L}(x^{N^0}, y^{N^0}, z^{N^0}, \lambda^{N^0}, \mu_1^{N^0}, \mu_2^{N^0})$ is not optimal solution value of the LGN,
and for all $N \geq N^0$
$\mathcal{L}(x^N, y^N, z^N, \lambda^N, \mu_1^N, \mu_2^N) = \mathcal{L}(x^{N^0}, y^{N^0}, z^{N^0}, \lambda^{N^0}, \mu_1^{N^0}, \mu_2^{N^0})$.

However, by Lemmata 5.7 and 5.8 we can immediately detect such a case as whenever we have $\beta^N = \beta^{N+1}$ ($\gamma^N = \gamma^{N+1}$) but $\gamma^N \neq \beta^{N+1}$ we can conclude that β^N and β^{N+1} (γ^N and γ^{N+1}) are nonoptimal. In order to handle such cases, we propose the following modified minmax scheme: Whenever the case $\beta^N = \beta^{N+1}$ ($\gamma^N = \gamma^{N+1}$) arises we assure that the next upper (lower) bound produced is better than the previous one by at least δ by adding the following constraint:

$$\beta \leq \beta^N - \delta \quad \text{(respectively} \quad \gamma \geq \gamma^N + \delta \text{)} \tag{5.14}$$

Here δ is the absolute error chosen for our stopping criterion. So whenever adding a constraint of type (5.14) leads to an infeasible problem, we know that the last upper (lower) bound found has at most a difference of δ to the optimal solution value and we can immediately stop the iterations.
Let $LGNs(\mathbf{x}, \mathbf{y}, \mathbf{z}, \lambda, \mu_1, \mu_2, UB)$ be the function that solves subproblem (5.10). If the corresponding problem has a feasible solution, the function returns the optimal solution value. Otherwise, it returns a predefined value NF. Here (λ, μ_1, μ_2) are parameters that add a new constraint of the form (5.10a). This constraint is kept in the following calls of

LGNs.
The function will store the solution vectors in the variables (x, y, z). If the problem has no feasible solution, (x, y, z) keep their input values. The parameter UB defines an additional constraint of type (5.14). This constraint is only kept for this single iteration. If $UB = \infty$ no constraint of type (5.14) is added.
In the same manner (just changing the roles of parameters and variables and the sign of the added constraint) we define $LGNd(x, y, z, \lambda, \mu_{\mathbf{1}}, \mu_{\mathbf{2}}, LB)$ with $LB = -\infty$ meaning that no additional constraint is added.
The proposed modified minmax scheme is stated in Algorithm 5.6.1. The variables UB^* and LB^* store the best upper and lower bound found so far while UB and LB are the bounds which we make the next calls of our problem solving functions with. tmp_{UB} and tmp_{LB} serve to store the return of these calls. The stopping criteria defined in section 5.5.2 are also used for this modified scheme. The following proposition is a corollary of the aforementioned results and the modification proposed in this section:

Proposition 5.13. *Algorithm 5.6.1 finds, after finitely many iterations, either a feasible solution for the LGN* (5.9) *with error ratio* ϵ *or a* δ*-optimal solution.*

□

5.7 Numerical experiments[1]

We conducted computational experiments to test the convergence of the scheme proposed in section 5.6 data randomly generated to match the general model studied in this chapter. As the exact solutions of the test instances generated are not known and cannot, in general, be practically obtained using common commercial solvers, we restricted our study to the convergence of the proposed minmax scheme.
The iterative minmax scheme has been implemented in C++. Linear programs are solved with Cplex 11[2].

5.7.1 Data Generation

The instances used were randomly generated following the LBP model. Note that this is the same as generating data for the initial model $SLBP$ directly since both problems are equivalent under the assumption that Ω is finite.
The parameters of the data generation are the number of variables (resp. constraints) for the upper-level problem n_x (resp. m_1), the number of variables (resp. constraints) for the lower-level problem n_y (resp. m_2), the number of scenarios K and the risk α.
In order to generate a bounded polyhedron, we used the method described in [HT89] to

[1] The numerical experiments and results presented in this section have been conducted by Pierre Le Bodic in the course of a joint work with Janny Leung and Abdel Lisser ([KBLL10]).
[2] Computation times have been measured on *NEC Express5800 120Rh (4*2GHz,3GB)* servers, with two single-processed schemes simultaneously running.

set the components of A, B and b: For every row except the last one, the elements of A and B are uniformly drawn in $[-1, 1]$. The components of the last row are uniformly chosen in $[0, 1]$. b is computed in the following manner:

$$b_i = \sum_{j=1}^{n} A_{ij} + B_{ij} + \rho,\ i = 1, \dots, m$$

where ρ is uniformly chosen in $[0, 2]$. This ensures that the polyhedron defined by both sets of constraints (5.6a) and (5.6h) is nonempty and bounded.
Constraints (5.6b) are generated as follows: For every scenario k, the components of w_k are uniformly chosen in $[0, 1]$. Let $W_k = w_k^t \mathbb{1}_{n_x}$. s_k is uniformly generated in $[\frac{1}{2}W_k, W_k]$. The upper bound W_k is chosen such that scenario k is not necessarily respected. The lower bound $\frac{1}{2}W_k$ is arbitrarily chosen so that scenario k is not too restrictive. As presented in section 5.3 one can set $M_k = W_k - s_k$ so that $z_k = 1$ disables the constraint.
The vector p of constraint (5.6c) respects:

$$\begin{cases} 0 \leq p_k \leq 1,\ k = 1, \dots, K \\ p^t \mathbb{1}_K = 1 \end{cases}$$

i.e. p_k is the probability of scenario k. The coefficients of the objective functions, i.e. the components of c_i and d_i $(i = 1, 2)$, are uniformly generated in $[0, 10]$.
This generation procedure ensures that the feasible region of the LBP instance generated is bounded, but it does not guarantee nonemptiness.

5.7.2 Numerical Results

The absolute (resp. relative) tolerance is $\delta = 10^{-8}$ (resp. $\epsilon = 10^{-5}$). The maximum relative error ϵ has obviously a significant influence on the number of iterations needed to stop the scheme. For example, in Figure 5.7.1 the convergence of the bound values iteratively produced by the minmax scheme on one of the randomly generated test instances (with $n_x = n_y = K = 100$) is shown. The scheme stops after solving 33 linear programs, i.e. at the 17^{th} iteration. After 10 iterations, the relative error amounts to 0.14%. Upon reaching the stopping criteria, the relative error is 0.01%. This convergence behavior is recurrent.

The obtained numerical results are shown in table 5.1. Results are only presented for $m_1 = n_x$ and $m_2 = n_y$. For each (K, n_x, n_y) triple, the scheme has been tested on 50 instances. For each triple, the minimum, average, maximum time (in seconds) and number of linear programs solved ($LGNs$ and $LGNd$ added) are provided.
Not surprisingly, the time needed to reach the stopping criteria increases both with the size of the upper-level and lower-level problems, n_x and n_y. The computing time as well as the number of linear problems solved does, however, not vary with the number of scenarios K. Also, a very interesting phenomenon arises once n_y becomes larger than n_x: The average number of iterations decreases as n_y increases (see for example the first four

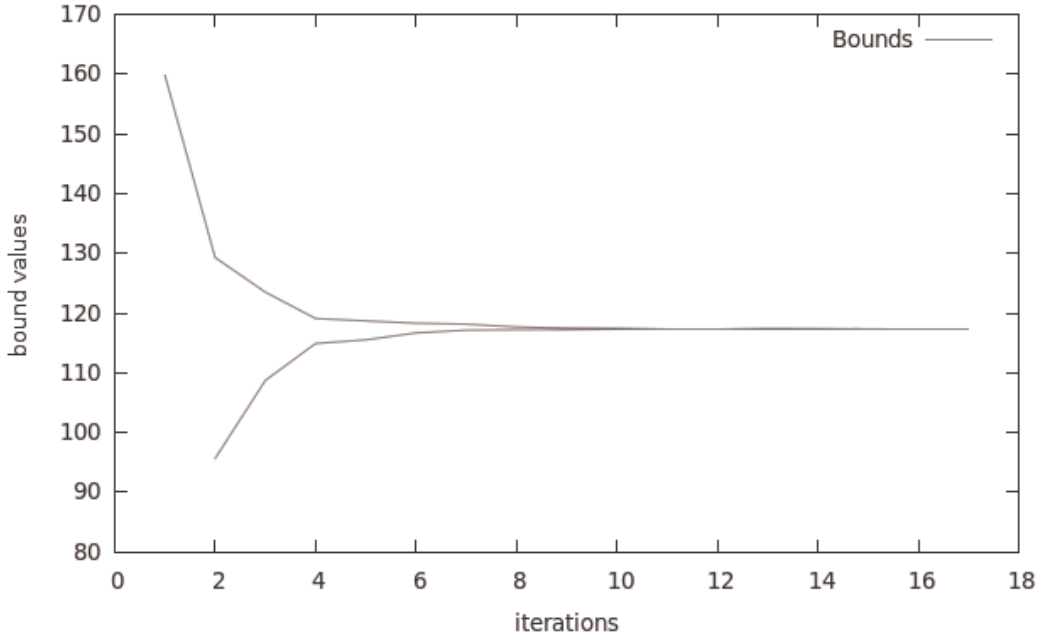

Figure 5.7.1: Convergence of the bound values for the $SLBP$ iteratively produced by the minmax scheme

valued rows of column Avg #LP). A possible explanation is that the bigger the lower-level problem, the larger the Lagrangian $\mathcal{L}(x, y, z, \lambda, \mu_1, \mu_2)$, and therefore the deeper each cut. Clearly, deeper cuts decrease the number of iterations needed to reach the stopping criteria.
The numerical tests confirm the theoretical results obtained in section 5.5, i.e. the iterative scheme reaches a near-optimal solution of the LGN after finitely many steps. However, the quality of the bounds produced by the minmax scheme remains to evaluate.

5.8 Improved bounds

The LGN (5.9) is a relaxation of the initial $SLBP$ and solving this problem will in general only give an upper bound on the optimal solution value of the $SLBP$. Unfortunately, some tests on particular instances (where the optimal solution values were known) have shown that these upper bounds can be quite far from the optimum. The following result shows that, in fact, a relaxation of the $GLCP$ where one simply removes the quadratic constraints (5.8g)-(5.8j) (hereafter called *Relaxed* $GLCP$ (RGLCP)) would generally give a better bound than the LGN:

Proposition 5.14. *Let* $(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$ *and* $(\tilde{x}, \tilde{y}, \tilde{z}, \tilde{\lambda}, \tilde{\mu}_1, \tilde{\mu}_2)$ *be optimal solutions of the LGN (5.9) and the RGLCP, respectively. Then*

$$c_1^t \tilde{x} + d_1^t \tilde{y} \leq \mathcal{L}(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$$

Proof. As $(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$ is optimal for the *LGN* (5.9), we have

$$\begin{aligned}\mathcal{L}(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*) &= \max_{x,y,z \in X} \quad \mathcal{L}(x, y, z, \lambda^*, \mu_1^*, \mu_2^*) \\ &\geq \mathcal{L}(x, y, z, \lambda^*, \mu_1^*, \mu_2^*) \quad \forall\, (x, y, z) \in X\end{aligned}$$

where X is the set of feasible triples (x, y, z) for problem *LGN* (5.9). As problems *LGN* and *RGLCP* have the same constraints, $(\tilde{x}, \tilde{y}, \tilde{z}) \in X$. We get

$$\mathcal{L}(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*) \geq \mathcal{L}(\tilde{x}, \tilde{y}, \tilde{z}, \lambda^*, \mu_1^*, \mu_2^*) \geq c_1^t \tilde{x} + d_1^t \tilde{y}$$

where the last inequality follows from

$$(\lambda^*)^t (b^2 - A^2 \tilde{x} - B^2 \tilde{y}) + (\mu_1^*)^t z + (\mu_2^*)^t (\mathbb{1}_K - \tilde{z}) + \tilde{y}^t ((B^2)^t \lambda^* - d_2) \geq 0$$

□

Due to this observation, we propose a second relaxation where the quadratic terms are relaxed in the objective function with a minus sign and the resulting problem is maximized over both the primal as well as dual variables. Let us define a second Lagrangian relaxation function $\mathcal{L}^-$ as

$$\mathcal{L}^-(x, y, z, \lambda, \mu_1, \mu_2) = c_1^t x + d_1^t y - \lambda^t (b^2 - A^2 x - B^2 y) - \mu_1^t z - \mu_2^t (\mathbb{1}_K - z) - y^t ((B^2)^t \lambda - d_2)$$

This gives us the following relaxation of the *GLCP* (5.8):

$$\begin{aligned}\text{(LGN-)} \quad \max_{x,y,z,\lambda,\mu_1,\mu_2} \quad & \mathcal{L}^-(x, y, z, \lambda, \mu_1, \mu_2) \\ \text{s.t.} \quad & A^1 x + B^1 y \leq b^1, && \text{(5.15a)} \\ & w_k^t x \leq s_k + M_k z_k \quad \forall\, k = 1, \ldots, K, && \text{(5.15b)} \\ & p^t z \leq \alpha, && \text{(5.15c)} \\ & A^2 x + B^2 y \leq b^2, && \text{(5.15d)} \\ & (B^2)^t \lambda \geq d_2, && \text{(5.15e)} \\ & \mathbb{I}_K \mu_1 + \mathbb{I}_K \mu_2 \geq \mathbb{1}_K, && \text{(5.15f)} \\ & \mathbb{O}_{n_x} \leq x \leq \mathbb{1}_{n_x}, \mathbb{O}_K \leq z \leq \mathbb{1}_K, && \text{(5.15g)} \\ & y, \lambda, \mu_1, \mu_2 \geq 0. && \text{(5.15h)}\end{aligned}$$

Proposition 5.15. *Let $(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$ be an optimal solution of the LGN. Then $\mathcal{L}^-(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$ is an upper bound on the optimal solution value of the GLCP (5.8).*

Proof. Let $(\widetilde{x}, \widetilde{y}, \widetilde{z}, \widetilde{\lambda}, \widetilde{\mu}_1, \widetilde{\mu}_2)$ denote an optimal solution of the *GLCP* (5.8) and let *OPT* denote the optimal solution value of (5.8). $(\widetilde{x}, \widetilde{y}, \widetilde{z}, \widetilde{\lambda}, \widetilde{\mu}_1, \widetilde{\mu}_2)$ is feasible for the relaxation (5.15) with corresponding objective function value $c_1^t \widetilde{x} + d_1^t \widetilde{y}$. It follows that for the optimal solution value OPT_{rel} of (5.15) we have

$$OPT_{rel} \geq c_1^t \widetilde{x} + d_1^t \widetilde{y} = OPT$$

which terminates the proof. □

Moreover, it can be shown that the upper bound obtained by solving the LGN is better than that obtained by solving the simple linear relaxation $RGLCP$ of the $GLCP$:

Proposition 5.16. *Let $(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$ be an optimal solution of the relaxation (5.15). Then $\mathcal{L}^-(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*) \leq OPT_{lrel}$ where OPT_{lrel} denotes the optimal solution value of the linear relaxation RGLCP.*

Proof. Let $(x^{rel}, y^{rel}, z^{rel}, \lambda^{rel}, \mu_1^{rel}, \mu_2^{rel})$ be an optimal solution of the $RGLCP$. As $(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$ is feasible for the $RGLCP$, it follows

$$OPT_{lrel} = c_1^t x^{rel} + d_1^t y^{rel} \geq c_1^t x^* + d_1^t y^*$$

Furthermore, we have

$$\begin{aligned} b^2 - A^2 x^* - B^2 y^* &\geq 0 \\ (B^2)^t \lambda^* - d_2 &\geq 0 \\ y^*, \lambda^*, \mu_1^*, \mu_2^* &\geq 0 \\ 0 \leq z^* &\leq 1 \\ \Rightarrow -(\lambda^*)^t (b^2 - A^2 x^* - B^2 y) - (\mu_1^*)^t z - (\mu_2^*)^t (\mathbb{1}_K - z) - (y^*)^t ((B^2)^t \lambda^* - d_2) &\leq 0 \end{aligned}$$

It follows

$$OPT_{lrel} \geq c_1^t x^* + d_1^t y^* \geq \mathcal{L}^-(x^*, y^*, z^*, \lambda^*, \mu_1^*, \mu_2^*)$$

□

The $LGN-$ has a structure similar to that of the LGN (5.9):

- The objective function of the $LGN-$ is bilinear and the constraints are linear.
- When fixing either the primal decision vectors x, y, z or the dual decision vectors λ, μ_1, μ_2, the obtained problem is linear (i.e. the bilinear summands in the objective always contain one primal and one dual variable).
- The constraints are separable in the sense that no constraint contains both primal and dual variables.

Contrary to the LGN the $LGN-$ is, however, a pure maximization problem. The iterative scheme presented in section 5.5 is therefore not a candidate to solve this new relaxation. In the next subsection we will give an extended review of methods that can be used to solve this second relaxation. Future work will consist of studying the tractability of these algorithms to solve the $LGN-$ and/or to adapt one or more of them to our special case.

5.8.1 Methods to solve Bilinear Optimization Problems with separable, linear constraints

There have been numerous attempts to solve separably constrained bilinear problems of the form

$$\begin{aligned} (\text{SCBP}) \quad & \max_{x,y} \quad c^T x + x^T Q\, y + d^T y \\ & \text{s.t. } x \in X, y \in Y. \end{aligned} \tag{5.16a}$$

where c, d are given vectors, A is a given matrix and X and Y are given polyhedra. A natural idea to solve such a problem is to iteratively find a solution of the two subproblems

$$\begin{aligned} (SCBP_1) \quad & \max_{x} \quad c^T x + x^T Q\, y^k \\ & \text{s.t. } x \in X. \end{aligned} \tag{5.17a}$$

and

$$\begin{aligned} (SCBP_2) \quad & \max_{y} \quad (x^k)^T Q\, y + d^T y \\ & \text{s.t. } y \in Y. \end{aligned} \tag{5.18a}$$

where x^k and y^k are the last obtained optimal solutions of the $SCBP_1$ or $SCBP_2$, respectively. Konno [Kon76] studied this idea and argued that, in case of finite polyhedra X and Y, the algorithm finds a Karush-Kuhn-Tucker point of the initial problem after a finite number of iterations. The author extended its algorithm using Ritter-cutting planes in order to obtain an ϵ-optimal solution. Although the final algorithm is proven to stop with such a solution, no demonstration is given that the algorithm effectively terminates after a finite number of steps.
Konno presented in this same paper a theorem that is the basis for another approach to solve separably constrained bilinear problems, namely vertex enumeration of the polyhedron X and/or Y (see for example [CF70]):

Theorem 5.17. *[Kon76] Assume that X and Y are nonempty and bounded. Then the $SCBP$ has an optimal solution (x^*, y^*) such that x^* is a vertex of the polyhedron X and y^* is a vertex of the polyhedron Y.*

By using the dual of either the $SCBP_1$ or the $SCBP_2$, the $SCBP$ can be reformulated as a programming problem with piecewise linear objective function (see for example [AAJS01]): Suppose that $X = \{x \geq 0 | Ax \leq a\}$ and $Y = \{y \geq 0 | By \leq b\}$. Then the $SCBP$ can be equivalently formulated as either

$$\begin{aligned} (SCBP_{d1}) \quad & \max_{x} \quad c^T x + \min_{u} b^T u \\ & \text{s.t. } Ax \leq a, && (5.19a) \\ & \qquad B\, u \geq d + Q^T\, x, && (5.19b) \\ & \qquad x, u \geq 0. && (5.19c) \end{aligned}$$

or

$$
\begin{aligned}
(SCBP_{d2}) \quad & \max_{y} \quad d^T y + \min_{u} a^T u \\
& \text{s.t. } By \leq b, && (5.20a)\\
& \quad A\,u \geq c + Q\,y, && (5.20b)\\
& \quad y, u \geq 0. && (5.20c)
\end{aligned}
$$

In [GU77] the authors reformulate the $SCBP_{d1}$ as a maximization problem over a generally nonconvex, but connected feasible set. The authors propose a cutting plane algorithm that explores the set of vertices of X. Contrary to the case of the iterative cutting plane algorithm by Konno, the authors of [GU77] show that their algorithm always terminates with an optimal (not just ϵ-optimal) solution, but finite termination is not guaranteed, either.
The first exact cutting plane algorithm for the $SCBP$ with finite convergence guarantee was proposed by Sherali and Shetty [SS80]. The algorithm makes use of a bunch of previously developed results on cutting plane algorithms. Among others, they improve a previously proposed algorithm by Vaish and Shetty ([VS77]) that did not yet guarantee finite convergence, by adding so called disjunctive cuts.
There have also been attempts to solve the $SCBP$ using a B&B algorithm. In fact, the B&B algorithm by Falk ([Fal73]) created for a more general minmax problem was the first algorithm for linear bilevel programming problems that was proven to terminate with an optimal solution in a finite number of steps. Bounding is done by solving pure maximization subproblems and the branching idea is to keep certain variables out of the basis until a solution is found that is also optimal for the initial minmax problem. Other B&B algorithms branch on the complementary slackness conditions of $SCBP_{d1}$ and $SCBP_{d2}$, an idea that is often used to solve problems with a structure similar to that of linear complementary problems (see also section 5.1.3 where this B&B method has been presented as an approach to directly solve linear bilevel problems). These as well as other B&B approaches (e.g. [AKF83]) are, however, in general intractable for large size instances and beat by the abovementioned cutting plane algorithm by Sherali and Shetty. As a consequence, most recent work on the solution of the $SCBP$ focused on the search for better cutting plane algorithms. In [AAJS01] the authors combine a cutting plane algorithm (using concavity cuts such as in the early works [Kon76] and [GU77]) with a B&B procedure. The algorithm is however slowed down by the fact that concavity cuts are added for both subproblems (5.19) and (5.20). Similarly, the algorithm by Sherali and Shetty suffers from the computational expensive generation of the disjunctive cuts. A very recent work by Ding and Al-Khayyal thus aimed to decrease the computational load of the added cuts by tightening the bounding (see [DAK07]).
A notable algorithm is that presented in [YK91] for bilinear problems where the rank of the matrix Q is two or three. The authors show that solving a so called two rank bilinear problem is in general not more difficult than solving three linear problems having the same constraints as the initial bilinear problem.

5.9 Concluding remarks and future work

In this chapter we studied a novel stochastic bilevel problem with knapsack chance-constraint. To solve this problem, we proposed to transform the initial problem into an equivalent quadratic problem based on the assumption of a discrete probability space. The obtained quadratic (or more precisely bilinear) problem was further relaxed into a linear minmax problem. An iterative approach with convergence guarantee applied to the latter allowed to find upper bounds on the optimal solution value of the initial stochastic bilevel problem.
Numerical experiments confirmed that the proposed iterative method converges in finitely many iterations to an ϵ-optimal solution. We however remarked that on some instances the obtained upper bound is relatively far from the optimum. We therefore proposed improved bounds by relaxing the quadratic problem not in a minmax, but a maximization problem. Methods to solve such bilinear maximization problems can be found in the literature and a selection of such algorithms was presented in this chapter. Future work will naturally consist in testing these algorithms on our special problem. It is clear that to solve the bilinear maximization problem obtained, some of the presented approaches might be more suitable than others. Especially, it might be fruitful to exploit the special structure of the problem, as for example some of the constraints are reflected in the objective.
For instance we only proposed a method to obtain an upper bound on the optimal solution value of the initial stochastic bilevel problem. It remains to study how to solve the problem exactly (for example by adapting one of the existing algorithms for linear bilevel programming problems) and to compare such an approach to the method proposed in this chapter (concerning computing time / tractability as well as the quality of our upper bounds). As an example, one could use an exact penalty method to include the bilinear terms in the objective (as proposed in [FP01]) instead of relaxing them.
In this chapter we assumed a finite number of possible outcomes for the random vector χ. Another interesting variant would be obtained by making the assumption that the weights as well as the capacity are independently normally distributed, as done for example in chapters 3 and 4. It is well known that in such a case the chance-constraint defines a convex set and can be deterministically reformulated. Consequently, methods to solve bilevel problems with convex upper and linear lower level problem could be applied to solve the problem. Moreover, one could extend the model to more general (convex) functions, in particular to the well studied case of a quadratic lower level problem.
Another possible approach to solve the stochastic bilevel problem with knapsack chance-constraint in the case of continuous probability distributions (or a number of scenarios that is too big to allow a deterministic equivalent reformulation as presented in this chapter) could be to use a stochastic gradient algorithm as proposed in [Wer04]: First, the initial problem is reformulated as a quadratic single level problem with knapsack chance-constraint using the reformulation method presented in this chapter. Then, the bilinear constraints are included in the objective by introducing *additional* Lagrange multipliers. The obtained problem is solved using a Stochastic Arrow-Hurwicz algorithm (see chap-

ter 3). Feasibility concerning the remaining linear constraints is ensured by a matrix projection at each iteration.

Iterative minmax Algorithm

Require: $\exists\,(x, y, z, \lambda, \mu_1, \mu_2)$ feasible for the LGN
Require: $\delta > 0, \epsilon > 0$
  $UB \leftarrow \infty$, $UB^* \leftarrow \infty$
  $LB \leftarrow -\infty$, $LB^* \leftarrow -\infty$
  (λ, μ_1, μ_2) feasible for the LGN
  loop
    $\mathrm{tmp}_{UB} \leftarrow LGNs(\mathbf{x}, \mathbf{y}, \mathbf{z}, \lambda, \mu_1, \mu_2, UB)$
    if $\mathrm{tmp}_{UB} = \mathrm{NF}$ **then**
      $LB^* = UB$
      BREAK LOOP
    else
      if $\mathrm{tmp}_{UB} = UB^*$ **then**
        $UB \leftarrow \mathrm{tmp}_{UB} - \delta$
      else
        $UB^* \leftarrow \mathrm{tmp}_{UB}$
        $UB \leftarrow \infty$
        if $UB^* - LB^* < \delta$ or $(UB^* - LB^*)/UB^* < \epsilon$ **then**
          BREAK LOOP
        end if
      end if
    end if
    $\mathrm{tmp}_{LB} \leftarrow LGNd(x, y, z, \lambda, \mu_1, \mu_2, LB)$
    if $\mathrm{tmp}_{LB} = \mathrm{NF}$ **then**
      $UB^* = LB$
      BREAK LOOP
    else
      if $\mathrm{tmp}_{LB} = LB^*$ **then**
        $LB \leftarrow \mathrm{tmp}_{LB} + \delta$
      else
        $LB^* \leftarrow \mathrm{tmp}_{LB}$
        $LB \leftarrow -\infty$
        if $UB^* - LB^* < \delta$ or $(UB^* - LB^*)/UB^* < \epsilon$ **then**
          BREAK LOOP
        end if
      end if
    end if
  end loop
  return LB^* and/or UB^*

Algorithm 5.6.1: The iterative minmax scheme with convergence guarantee.

Parameters			**Time (s)**			**#LP**		
K	n_x	n_y	Min	Avg	Max	Min	Avg	Max
20	50	50	0	0	1	17	29	57
		100	0	0	1	18	32	44
		500	11	13	17	17	24	32
		1000	113	137	172	16	20	27
	100	50	0	0	1	18	31	48
		100	0	0	2	23	49	84
		500	18	21	25	23	34	52
		1000	152	180	213	20	28	37
	500	50	5	7	10	13	26	43
		100	10	13	17	22	36	51
		500	95	123	179	46	81	177
		1000	431	549	795	53	82	116
	1000	50	46	61	76	13	25	37
		100	73	89	108	21	31	43
		500	293	383	458	34	57	89
		1000	854	1211	1549	32	75	129
50	50	50	0	0	1	15	28	56
		100	0	0	1	17	31	49
		500	13	15	18	18	25	31
		1000	115	134	166	15	20	26
	100	50	0	0	1	23	32	53
		100	0	0	2	27	44	73
		500	19	23	29	26	36	56
		1000	149	179	212	20	28	41
	500	50	6	8	12	13	28	47
		100	11	14	19	24	36	58
		500	89	127	199	39	80	213
		1000	446	535	652	47	80	151
	1000	50	50	64	78	10	26	41
		100	78	93	114	23	32	43
		500	277	389	629	25	56	89
		1000	638	1259	2235	10	77	113
100	50	50	0	0	1	19	28	45
		100	0	0	1	18	30	44
		500	13	15	19	16	23	32
		1000	115	140	166	16	20	26
	100	50	0	0	1	17	32	49
		100	1	1	2	29	42	79
		500	19	23	29	23	33	46
		1000	156	180	213	21	27	37
	500	50	7	9	14	15	27	45
		100	11	16	22	19	36	61
		500	102	130	169	50	77	151
		1000	485	548	720	53	79	163
	1000	50	50	67	92	17	26	40
		100	79	97	126	19	33	47
		500	303	405	760	37	56	77
		1000	867	1220	1528	33	77	122

Table 5.1: Numerical results for the $SLBP$: Convergence results of the iterative scheme

6 Conclusion

Stochastic optimization problems are generally hard to solve. In only some few cases introducing stochasticity does not considerably change the difficulty of the problem. In the rest of the cases, assuming uncertainty in one or more parameters significantly increases the time or space requirement to solve the problem exactly or approximately. As an example, we have seen in the second part of chapter 4 that the knapsack problem, that is weakly $\mathcal{NP}$-hard in the deterministic setting, becomes strongly $\mathcal{NP}$-hard in case of a stochastic two-stage setting with random, discretely distributed weights. Moreover, the deterministic knapsack problem has a very simple constant-factor approximation algorithm, whereas the Two-Stage Knapsack problem was shown to be most likely non-approximable within a constant factor.
Due to the hardness of stochastic optimization problems, it is not unusual that the conducted studies focus on the computation of good and relatively fast computable upper and/or lower bounds, for example by solving a relaxed version of the initial problem.
In chapter 2 we proposed a method to solve the Simple Recourse Knapsack problem with normally distributed weights. We have seen that the relaxed, i.e. continuous version of the problem can be solved approximately in a rather efficient way by applying a stochastic gradient algorithm. When comparing this approach with the solution of the deterministic knapsack problem, whose relaxation can be solved by a very simple greedy algorithm, one, however, gets an idea of the increase in difficulty that the introduction of uncertainties entails. It is thus not surprising that, for instance, we are only able to solve the (combinatorial) Simple Recourse Knapsack problem with up to 100 items in an average computing time of less than $1h$.
In chapter 3 we studied the Chance-Constrained Knapsack problem. Once again we applied a stochastic gradient type algorithm to solve the continuous relaxation of the problem. We encountered some problems that are based on the special structure of the problem. Two of these problems prevented the algorithm from converging. They could be solved by investigating in a deeper analysis of the problem's structure and the used solution method. The third problem concerned our choice to solve the Chance-Constrained Knapsack problem with a stochastic Arrow-Hurwicz algorithm. While the initial problem can be shown to be a convex optimization problem (more precisely the maximization of a linear objective function over a convex feasible set), the objective function of the chosen Lagrangian relaxation is unfortunately not concave on the whole feasible region. Although this was not an issue for the convergence of the algorithm on the test instances generated, a wiser choice of the Lagrangian relaxation would result in a concave objective of the Lagrangian relaxation. This idea has been applied in chapter 4. Nevertheless, we have seen that we can solve the relaxed as well as combinatorial Chance-Constrained Knapsack

problem with the proposed method with more or less the same efficiency as the Simple Recourse Knapsack problem in chapter 2.
The first part of chapter 4 was dedicated to the Two-Stage Knapsack problem with normally distributed weights. When searching for a method to solve this problem, one encounters a difficulty that is typical for stochastic optimization problems but that has not yet been an issue in the previous two chapters: the question of how to evaluate the objective function. In the case of the Two-Stage Knapsack problem with normally distributed weights (as well as for many stochastic optimization problems with continuously distributed random variables), the evaluation of the objective function involves the computation of a multi-dimensional integral whose integrand and domain of integration is, in addition, not given analytically. In the literature, this problem is often avoided by assuming a discrete distribution for the random parameters. We instead proposed lower bounds on the objective function value that are designed to replace the exact evaluation of the objective function in a framework such as a branch-and-bound algorithm. We also showed, that the continuous relaxation of the problem is in fact a Simple Recourse Knapsack problem with chance-constraint, which allowed us to solve this relaxation using methods from the previous two chapters. For the particular case of similar item weights, we were able to solve the problem to near optimality for up to 2000 items in about 30 minutes average computing time.
In the second part of chapter 4 we studied the Two-Stage Knapsack problem under the assumption of discretely distributed weights. In this case, the problem can be reformulated as a linear programming problem with binary variables and could thus theoretically be solved with common algorithms for linear combinatorial optimization. The problem is (as with many deterministic reformulations of stochastic optimization problems) that solving the obtained deterministic problem to optimality is generally only tractable in case the random variables only admit a very small number of realizations. We therefore studied the problem under the aspect of approximability. While some combinatorial optimization problems are not much more difficult to approximate than their deterministic counterpart, the Two-Stage Knapsack problem turned out to not even admit a constant-factor approximation algorithm (unless $\mathcal{P} = \mathcal{NP}$). This result has been obtained by a reduction from the well-studied multiply-constrained (or multi-dimensional) knapsack problem. Moreover, even some special variants of the Two-Stage Knapsack problem that, on the first sight, seem to be easier to solve than the general problem were shown to be non-approximable within a constant factor.
The stochastic optimization problem studied in the last chapter of this thesis is different from those in the previous chapters in the sense that its deterministic counterpart is already very hard to solve: While the stochastic knapsack problems treated in chapters 2-4 were based on the weakly $\mathcal{NP}$-hard deterministic knapsack problem, we studied in the last chapter a stochastic variant of a bilevel problem with knapsack constraint. Bilevel problems can be shown to be strongly $\mathcal{NP}$-hard even in the apparently simplest case of linear objectives and constraints and continuous decision variables. Contrary to the previous chapters, the main effort made to solve the proposed stochastic bilevel problem therefore consisted in solving the underlying deterministic problem (or, more precisely, computing

upper bounds on its optimal solution value). The problem studied in chapter 5 is thus an example for a stochastic optimization problem that is not "much" harder to solve than its deterministic counterpart simply by the fact that the underlying deterministic problem is already extremely difficult.

6.1 Future Work

Some concrete ideas for future work concerning the models and methods proposed in this thesis have already been presented in the respective chapter conclusions. Here we want to discuss some more general, open questions concerning the thesis globally.

In this thesis we assumed for the random variables of each of the treated problems a particular underlying probability distribution. Future work will thus first of all consist in extending the obtained results and proposed methods to a larger class of distributions.
In the first two chapters we assumed the item weights of the studied stochastic knapsack problems to be independently normally distributed. The great advantage of this assumption is that by using the available value tables for normally distributed variables we are able to easily evaluate the objective or constraint functions. We used this in the proposed branch-and-bound framework in order to obtain lower bounds as well as in the stochastic gradient algorithm in order to determine the best feasible solution found. Our solution method can therefore be easily extended to other distributions that allow for a simple evaluation of the involved functions, but also to the case of a black-box assumption: The evaluation of the objective function in the branch-and-bound tree could be replaced by either lower bounds on the exact objective function value (as done in section 4 for the Two-Stage Knapsack problem) or by approximations of the objective function value using sampling. Concerning the stochastic gradient algorithm, the numerical convergence tests have shown, that it might be an alternative to stop the algorithm when the variance of the produced solutions becomes sufficiently small. Remember that the algorithm itself does only make use of a black-box for the random variables and theoretically no evaluation of the objective is needed during the algorithm. Moreover, the sampling done during the stochastic gradient algorithm could be used to approximate the solution values for the obtained solution vectors.
In the first part of chapter 4 we used the assumption of a normal distribution to propose lower bounds on the objective function value of the Two-Stage Knapsack problem for a given first-stage solution. Mainly, we took advantage of the property that the CDF of a random variable defined as the maximum over a set of normally distributed parameters can be easily computed. Such a property might be difficult to obtain for other distributions. It is, however, not clear how to evaluate the expectation of such a random variable in case of normal distributions, which might in contrary be possible for different distributions. Therefore, the approach of chapter 4 is probably not extendable to a general class of distributions, but similar bounds might be obtained for other (continuous) distributions.
Concerning the non-approximation result proved in the second part of chapter 4, it would

be interesting to see what happens if the underlying distribution is assumed to be continuous or if we make a black-box assumption. Approximating the problem by applying a Sample Average Approach is clearly not useful, as the obtained problem is a Two-Stage Knapsack problem with discretely distributed weights that has been shown to be non-approximable within a constant factor.
For the Stochastic Bilevel problem studied in chapter 5 some ideas to solve the problem in case of continuously distributed random variables have already been given at the end of the chapter (see section 5.9).

In chapters 3, 4 and 5 the treated problems contain a knapsack chance-constraint. Here, future research might consist in extending the models to the case of a joint chance-constraint, i.e. the case where several constraints shall be jointly satisfied with a probability greater or equal to a given threshold. Note that in case of discretely distributed parameters and binary (or bounded) decision variables, such a constraint can be replaced by a set of deterministically equivalent constraints as done in chapter 5 with the simple chance-constraint (see also the introduction of chapter 3). In case of continuous distributions, the question is mainly how to evaluate the probability function.
An open problem is how to handle additional *deterministic* constraints that for example represent a network architecture or additional (deterministic) resource/demand relations. A possible answer to this question has already been discussed in the conclusion of chapter 2. More details can be found in [KL10b] where we propose a projected gradient algorithm to solve the Simple Recourse Shortest Path problem. A still open question concerning the deterministic projected gradient method however prevented us for instance from obtaining numerical results and evaluating the quality of our randomized algorithm.

An important stochastic optimization problem type that has not at all been studied in this thesis is the multi-stage model. It is clear that in many practical cases informations about the random variables are not revealed in one, but several stages. When extending the Two-Stage Knapsack problem to a Multi-Stage Knapsack problem the first question that has to be answered is what decision can or has to be made in each stage. Can an item that has been rejected in one stage be re-added in a subsequent stage? Has an item, once it is chosen, to be kept in subsequent stages? Is it realistic to assume that we know in advance which item weights are revealed in which stage? Open questions concerning the solution of such a problem are: Can the problem be decomposed into a sequence of Two-Stage problems? Can the continuous relaxation be reformulated as a sort of Simple Recourse Knapsack problem as this is the case for the corresponding two-stage problem? Can the lower bounds proposed in this thesis be extended to the multi-stage case?
As a concrete example, imagine that the item weights come to be known over several stage and the reward per weight unit of each item is strictly decreasing from one stage to the next. Items can be chosen while their weight is still unknown or in one of the stages after the actual weight has come to be known. An added item can be rejected once its weight is revealed. However, it cannot be re-added any more. Of course, rejecting an item entails a penalty that is strictly greater than the reward obtained for adding the item. Then a (sim-

ple) lower bound can be computed by solving the following Two-Stage Knapsack problem: One assumes that all weights are revealed in the second stage and that the item's rewards per weight unit equal the reward per weight unit received in the last stage. Instead of solving the problem exactly, the lower bounds proposed in this thesis could be applied. Similarly, an upper bound is given by the solution of the Two-Stage Knapsack problem where the second-stage rewards equal the second-stage rewards of the multi-stage problem.

Last but not least, it remains naturally to study how to (efficiently) solve the Two-Stage Knapsack problems of chapter 4 and the Stochastic Bilevel problem with knapsack constraint of chapter 5 to optimality. Although many algorithms have been proposed for both types of problems (i.e. two-stage and quadratic problems), it seems promising to exploit the special structure of the problems in order to obtain more efficient algorithms.

List of Figures

List of Tables

Bibliography

[AAJS01] Stéphane Alarie, Charles Audet, Brigitte Jaumard, and Gilles Savard. Concavity cuts for disjoint bilinear programming. *Mathematical Programming*, 90(2):373–398, 2001.

[ACVA07] Laetitia Andrieu, Guy Cohen, and Felisa Vázquez-Abad. Stochastic programming with probability constraints. `http://hal.inria.fr/hal-00166149_v1` (Accessed 17 August 2010), 2007.

[AG09] Semra Ağralı and Joseph Geunes. A single-resource allocation problem with Poisson resource requirements. *Optimization Letters*, 3(4):559–571, 2009.

[AHJS97] Charles Audet, Pierre Hansen, Brigitte Jaumard, and Gilles Savard. Links between linear bilevel and mixed 0-1 programming problems. *Journal of Optimization Theory and Applications*, 93(2):273–300, 1997.

[AK90] Faiz Al-Khayyal. Jointly constrained bilinear programs and related problems: An overview. *Computers & Mathematics with Applications*, 19(11):53–62, 1990.

[AKF83] Faiz Al-Khayyal and James E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

[Alb03] Susanne Albers. Online algorithms: a survey. *Mathematical Programming*, 97:3–26, 2003.

[And04] Laetitia Andrieu. *Optimization sous contrainte en probabilité*. PhD thesis, Ecole Nationale des Ponts et Chaussées, 2004.

[ARGK98] Norbert Ascheuer, Jörg Rambau, Martin Grötschel, and Sven O. Krumke. Combinatorial online optimization. In *Operations Research Proceedings*, 1998.

[AS02] Shabbir Ahmed and Alexander Shapiro. The sample average approximation method for stochastic programs with integer recourse. Technical report, School of Industrial & Systems Engineering, Georgia Institute of Technology, 2002. `http://www2.isye.gatech.edu/~sahmed/saasip.pdf`.

[ATS03] Shabbir Ahmed, Mohit Tawarmalani, and Nikolaos V. Sahinidis. A finite branch and bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100(2), 2003.

[BAB90] Omar Ben-Ayed and Charles E. Blair. Computational difficulties of bilevel linear programming. *Operations Research*, 38(3):556–560, 1990.

[Bar84] Jonathan F. Bard. Optimality conditions for the bilevel programming problem. *Naval Research Logistics*, 31(1):13–26, 1984.

[Bar88] Jonathan F. Bard. Convex two-level optimization. *Mathematical Programming*, 40(1):15–27, 1988.

[Bar98] Jonathan F. Bard. Practical bilevel optimization: Algorithms and applications. In *Nonconvex Optimization and its Applications*, volume 30. Kluwer Academic Publishers, 1998.

[Bea61] E. Martin L. Beale. The use of quadratic programming in stochastic linear programming. *Rand Corporation Report*, P-2404-1, 1961.

[BF82] Jonathan F. Bard and James E. Falk. An explicit solution to the multilevel programming problem. *Computers & Operations Research*, 9(1):77–100, 1982.

[BHM09] Luce Brotcorne, Saïd Hanafi, and Raïd Mansi. A dynamic programming algorithm for the bilevel knapsack problem. *Operations Research Letters*, 37(3):215–218, 2009.

[BK84] Wayne F. Bialas and Mark H. Karwan. Two-level linear programming. *Management Science*, 30(8):1004–1020, 1984.

[BKS80] Wayne F. Bilias, Mark H. Karwan, and J. Shaw. A parametric complementary pivot approach for two-level linear programming. Technical report, State University of New York at Buffalo, Operations Research Program, 1980.

[BL88] John R. Birge and François V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392, 1988.

[BM73] Jerome Bracken and James T. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1):37–44, 1973.

[BM90] Jonathan F. Bard and James T. Moore. A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing*, 11(2):281–292, 1990.

[BR02] Patrizia Beraldi and Andrzej Ruszczyński. A branch and bound method for stochastic integer problems under probabilistic constraints. *Optimization Methods and Software*, 17(3):359–382, 2002.

[BS07] Hans-Georg Beyer and Bernhard Sendhoff. Robust optimization - a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196:3190–3218, 2007.

[CAAK06] Myun-Seok Cheon, Shabbir Ahmed, and Faiz Al-Khayyal. A branch-reduce-cut algorithm for the global optimization of probabilistically constrained linear programs. *Mathematical Programming*, 108(2):617–634, 2006.

[Car10a] Claus C. Carøe. *Decomposition in Stochastic Integer Programming.* PhD thesis, Institute for Mathematical Sciences, Dept. of Operations Research, University of Copenhagen, Denmark, 2010.

[Car10b] Pierre Carpentier. Méthodes numériques en optimisation stochastique. `http://www.ensta.fr/~pcarpent/MNOS/` (Accessed 17 August 2010), 2010.

[CB98] Amy Cohn and Cynthia 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.

[CC59] Abraham Charnes and William W. Cooper. Chance-constrained programming. *Management Science*, 6(1):73–79, 1959.

[CC95] Jean-Christophe Culioli and Guy Cohen. Optimisation stochastique sous contraintes en espérance. *Comptes rendus de l'Académie des sciences, Paris, Série I*, 320(6):753–758, 1995.

[CC04] Marco C. Campi and Giuseppe C. Calafiore. *Multiple Participant Decision Making (eds.: J. Andrysek, M. Karny and J. Kracik)*, chapter Decision making in an uncertain environment: The scenario based optimization approach, pages 99–111. Advanced Knowledge International, 2004.

[CCS58] Abraham Charnes, William W. Cooper, and Gifford H. Symonds. Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil. *Management Science*, 4(3):235–263, 1958.

[CF70] A. Victor Cabot and Richard L. Francis. Solving certain nonconvex quadratic minimization problems by ranking the extreme points. *Operations Research*, 18(1):82–86, 1970.

[CMS07] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, 2007.

[CN77] Wilfred Candler and Roger D. Norton. Multilevel programming. Technical report, World Bank Development Research Center, Washington, D.C., 1977.

[CS99] Claus C. Carøe and Rüdiger Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24:37–45, 1999.

[CSW93] Robert L. Carraway, Robert L. Schmidt, and Lawrence R. Weatherford. An algorithm for maximizing target achievement in the stochastic knapsack problem with normal returns. *Naval research logistics*, 40(2):161–173, 1993.

[CT82] Wilfred Candler and Robert J. Townsley. A linear two level programming problem. *Computers and Operations Research*, 9(1):59–76, 1982.

[CT98] Claus C. Carøe and Jørgen Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83(3):451–464, 1998.

[DAK07] Xiaosong Ding and Faiz Al-Khayyal. Accelerating convergence of cutting plane algorithms for disjoint bilinear programming. *Journal of Global Optimization*, 38(3):421–436, 2007.

[Dan55] George Dantzig. Linear programming under uncertainty. *Management Science*, 1:197–206, 1955.

[Dem02] Stephan Dempe. Foundations of bilevel programming. In *Nonconvex Optimization and its Applications*, volume 61. Kluwer Academic Publishers, 2002.

[Dem03] Stephan Dempe. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Optimization: A Journal of Mathematical Programming and Operations Research*, 52(3):333–359, 2003.

[DM61] George Dantzig and Albert Madansky. On the solution of two-stage linear programs under uncertainty. In *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 165–176. University of California, Berkeley, 1961.

[DPR00] Darinka Dentcheva, András Prékopa, and Andrzej Ruszczyński. Concavity and efficient points of discrete distributions in probabilistic programming. *Mathematical Programming*, 89(1):55–77, 2000.

[DR00] Stephan Dempe and K. Richter. Bilevel programming with knapsack constraints. *Central European Journal of Operations Research*, 8(2):93–107, 2000.

[Dra72] Mihai Dragomirescu. An algorithm for the minimum-risk problem of stochastic programming. *Operations Research*, 20(1):154 – 164, 1972.

[DRS05] Kedar Dhamdhere, R. Ravi, and Mohit Singh. On two-stage stochastic minimum spanning trees. *Lecture Notes in Computer Science*, 3509:321–334, 2005.

[DST03] Shane Dye, Leen Stougie, and Asgeir Tomasgard. The stochastic single resource service-provision problem. *Naval Research Logistics*, 50(8):869–887, 2003.

[DW60] George Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[EN67] Yuri M. Ermol'ev and Z. V. Nekrylova. The method of stochastic gradients and its applications. In *The Theory of Optimal Solutions [in Russian]*, volume 1. IK AN UkrSSR, Kiev, 1967.

[ENW95] Yuri M. Ermoliev, Vladimir I. Norkin, and Roger J-B. Wets. The minimization of semicontinuous functions: Mollifier subgradients. *SIAM Journal on Control and Optimization*, 33(1):149–167, 1995.

[Fal73] James E. Falk. A linear max-min problem. *Mathematical Programming*, 5(1):169–188, 1973.

[FAM81] José Fortuny-Amat and Bruce McCarl. A representation and economic interpretation of a two-level programming problem. *Journal of the Operational Research Society*, 32:783–792, 1981.

[Fe98] Amos Fiat and Gerhard J. Woeginger (eds.). Online algorithms: The state of the art. In *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[FLLP08] Bernard Fortz, Martine Labbé, François V. Louveaux, and Michael Poss. A non linear approach to the stochastic knapsack problem with recourse. In *VI ALIO/EURO Workshop on Applied Combinatorial Optimization*, 2008.

[FP01] Christodoulos A. Floudas and Panos M. Pardalos. *Encyclopedia of Optimization*, volume 6, chapter Bilevel linear programming: Complexity, equivalence to mimax, concave programs. Kluwer Academic Publishers, 2001.

[FV94] Christodoulos A. Floudas and Vishy Visweswaran. *Handbook of Global Optimization (eds.: R. Horst and P.M. Pardalos)*, chapter Quadratic optimization, pages 217–270. Kluwer Academic Publishers, 1994.

[GI99] Ashish Goel and Piotr Indyk. Stochastic load balancing and related problems. In *40th Annual Symposium on Foundations of Computer Science*, pages 579–586, 1999.

[GLLH10] Alexei A. Gaivoronski, Abdel Lisser, Rafael Lopez, and Xu Hu. Knapsack problem with probability constraints. *Journal of Global Optimization (Online First)*, 2010. `http://dx.doi.org/10.1007/s10898-010-9566-0` (Accessed 17 August 2010).

[GPRS04] Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. Boosted sampling: Approximation algorithms for stochastic optimization. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 417–426, 2004.

[GR09] Vineet Goyal and R. Ravi. Chance constrained knapsack problem with random item sizes. `http://www.mit.edu/~goyalv/stoch{_}knapsack.pdf` (Accessed 17 August 2010), 2009.

[GU77] Giorgio Gallo and Aydin Ülkücü. Bilinear programming: An exact algorithm. *Mathematical Programming*, 12(1):173–194, 1977.

[HB06] Pascal Van Hentenryck and Russell Bent. *Online Stochastic Combinatorial Optimization*. MIT Press, 2006.

[HJS92] Pierre Hansen, Brigitte Jaumard, and Gilles Savard. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1194–1217, 1992.

[HS91] Julia L. Higle and Suvrajeet Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research*, 16(3):650–669, 1991.

[HS96] Julia L. Higle and Suvrajeet Sen. Stochastic decomposition: A statistical method for large scale stochastic linear programming. In *Nonconvex Optimization and its Applications*, volume 8. Kluwer Academic Publishers, 1996.

[HT89] Reiner Horst and Nguyen Van Thoai. Modification, implementation and comparison of three algorithms for globally solving linearly constrained concave minimization problems. *Computing*, 42:271–289, 1989.

[HT96] Reiner Horst and Hoang Tuy. *Global Optimization: Deterministic Approaches*. Springer-Verlag, 3. edition, 1996.

[HUL93a] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. Convex analysis and minimization algorithms: Part 1: Fundamentals. In *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1993.

[HUL93b] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. Convex analysis and minimization algorithms: Part 2: Advanced theory and bundle methods. In *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1993.

[HvdV06] Willem K. Klein Haneveld and Maarten H. van der Vlerk. Integrated chance constraints: Reduced forms and an algorithm. *Computational Management Science*, 3(4):245–269, 2006.

[IA92] Yo Ishizuka and Eitaro Aiyoshi. Double penalty method for bilevel optimization problems. *Annals of Operations Research*, 34(1):73–88, 1992.

[IKMM04] Nicole Immorlica, David Karger, Maria Minkoff, and Vahab S. Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms, Session 8B*, pages 691–700, 2004.

[JF88] Joaquim J. Júdice and Ana M. Faustino. The solution of the linear bilevel programming problem by using the linear complementarity problem. *Investigação Operacional*, 8:77–95, 1988.

[JF94] Joaquim J. Júdice and Ana M. Faustino. The linear-quadratic bilevel programming problem. *Information Systems and Operational Research*, 32:87–98, 1994.

[Kal74] Peter Kall. Approximations to stochastic programs with complete fixed recourse. *Numerische Mathematik*, 22:333–339, 1974.

[Kal76] Peter Kall. *Stochastic Linear Programming*. Springer-Verlag, 1976.

[Kal94] Peter Kall. Solution methods in stochastic programming. In Jacques Henry and Jean-Pierre Yvon, editors, *System Modelling and Optimization*, volume 197 of *Lecture Notes in Control and Information Sciences*, pages 1–22. Springer Berlin / Heidelberg, 1994.

[KBLL09] Stefanie Kosuch, Pierre Le Bodic, Janny Leung, and Abdel Lisser. On a stochastic bilevel programming problem with knapsack constraints. In *Proceedings of the International Network Optimization Conference*, 2009.

[KBLL10] Stefanie Kosuch, Pierre Le Bodic, Janny Leung, and Abdel Lisser. On a stochastic bilevel programming problem with knapsack constraints. *Networks*, Accepted for publication, 2010.

[KHSvdV95] Willem K. Klein Haneveld, Leen Stougie, and Maarten H. van der Vlerk. On the convex hull of the simple integer recourse objective function. *Annals of Operations Research*, 56(1):209–224, 1995.

[KHSvdV96] Willem K. Klein Haneveld, Leen Stougie, and Maarten H. van der Vlerk. An algorithm for the construction of convex hulls in simple integer recourse programming. *Annals of Operations Research*, 64(1):67–81, 1996.

[KHSvdV06] Willem K. Klein Haneveld, Leen Stougie, and Maarten H. van der Vlerk. Simple integer recourse models: convexity and convex approximations. *Mathematical Programming*, 108(2):435–473, 2006.

[KHvdV94] Willem K. Klein Haneveld and Maarten H. van der Vlerk. On the expected value function of a simple integer recourse problem with random technology matrix. *Journal of Computational and Applied Mathematics*, 56:45–53, 1994.

[KKMU07] Irit Katriel, Claire Kenyon-Mathieu, and Eli Upfal. Commitment under uncertainty: Two-stage stochastic matching problems. *Lecture Notes in Computer Science*, 4596:171–182, 2007.

[KL90] Charles D. Kolstad and Leon S. Lasdon. Derivative estimation and computational experience with large bilevel mathematical programs. *Journal of Optimization Theory and Applications*, 65:485–499, 1990.

[KL09] Stefanie Kosuch and Abdel Lisser. On a two-stage stochastic knapsack problem with probabilistic constraint. In *Proceedings of the 8th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, 2009.

[KL10a] S. Kosuch and A. Lisser. On two-stage stochastic knapsack problems. *Discrete Applied Mathematics*, In Press, Corrected Proof, 2010. `http://www.sciencedirect.com/science/article/B6TYW-507DJM2-1/2/5e433a7491637bdf8e4a610ff4df45f2` (Accessed 17 August 2010).

[KL10b] Stefanie Kosuch and Abdel Lisser. Stochastic shortest path problem with delay excess penalty. *Electronic Notes in Discrete Mathematics*, 36(1):511–518, 2010.

[KL10c] Stefanie Kosuch and Abdel Lisser. Upper bounds for the 0-1 stochastic knapsack problem and a b&b algorithm. *Annals of Operations Research*, 176(1):77–93, 2010.

[KLL10] Stefanie Kosuch, Marc Letournel, and Abdel Lisser. On a stochastic knapsack problem. In *Proceedings of the 9th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, 2010.

[Klo09] Olivier Klopfenstein. Tractable algorithms for chance-constrained combinatorial problems. *RAIRO-Operations Research*, 43:157–187, 2009.

[Klo10] Olivier Klopfenstein. Solving chance-constrained combinatorial problems to optimality. *Computational Optimization and Applications*, 45(3):607–638, 2010.

[KM93] Peter Kall and Janos Mayer. Slp-ior: On the design of a workbench for testing slp codes. *Revista Investigacion Operacional*, 14:148–161, 1993.

[KM05] Peter Kall and Janos Mayer. *Stochastic Linear Programming: Models, Theory, and Computation*. Kluwer Academic Publishers, 2005.

[KN08] Olivier Klopfenstein and Dritan Nace. A robust approach to the chance-constrained knapsack problem. *Operations Research Letters*, 36(5):628–632, 2008.

[Kol67] Peter J. Kolesar. A branch and bound algorithm for the knapsack problem. *Management Science*, 13(9):723–735, 1967.

[Kon76] Hiroshi Konno. A cutting plane algorithm for solving bilinear programs. *Mathematical Programming*, 11(1):14–27, 1976.

[KRT97] Jon Kleinberg, Yuval Rabani, and Éva Tardos. Allocating bandwidth for bursty connections. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 664–673, 1997.

[KS06] Nan Kong and Andrew J. Schaefer. A factor $\frac{1}{2}$ approximation algorithm for two-stage stochastic matching problems. *European Journal of Operational Research*, 172(3):740–746, 2006.

[KSH02] Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[KW52] Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[KW94] Peter Kall and Stein W. Wallace. *Stochastic Programming*. John Wiley & Sons, Chichester, 1994.

[KY03] Harold J. Kushner and George Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer-Verlag, 2. edition, 2003.

[Lag98] Manuel Laguna. Applying robust optimization to capacity expansion of one location in telecommunications with demand uncertainty. *Management Science*, 44(11):101–110, 1998.

[LHZ01] G. S. Liu, J. Y. Han, and J. Z. Zhang. Exact penalty functions for convex bilevel programming problems. *Journal of Optimization Theory and Applications*, 110(3):621–643, 2001.

[LL93] Gilbert Laporte and François V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993.

[LLH09] Abdel Lisser, Rafael Lopez, and Xu Hu. Stochastic quadratic knapsack with recourse. In *INOC 2009*, 2009.

[LLM92] Gilbert Laporte, François V. Louveaux, and Hélène Mercure. The vehicle routing problem with stochastic travel times. *Transportation Science*, 26(3):161–170, 1992.

[LLS05] Constantino M. Lagoa, Xiang Li, and Mario Sznaier. Probabilistically constrained linear programs and risk-adjusted controller design. *SIAM Journal on Optimization*, 15(3):938–951, 2005.

[Lop10] Rafael Lopez. *Stochastic Quadratic Knapsack Problems and Semidefinite Programming*. PhD thesis, LRI, Université Paris Sud, France, 2010.

[LVBL98] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

[LvdV93] François V. Louveaux and Maarten H. van der Vlerk. Stochastic programming with simple integer recourse. *Mathematical Programming*, 61:301–325, 1993.

[LY98] Pierre L'Écuyer and George Yin. Budget dependent convergence rate of stochastic approximation. *SIAM Journal on Optimization*, 8(1):217–247, 1998.

[LY99] Zhang Li'ang and Zhang Yin. Approximation for knapsack problems with multiple constraints. *Journal of Computer Science and Technology*, 14(4):289–297, 1999.

[May79] Janos Mayer. *Survey of Mathematical Programming (ed.: A. Prékopa)*, volume 2, chapter A nonlinear programming method for the solution of a stochastic programming model of A. Prékopa, pages 129–139. North Holland Publishing Co., 1979.

[May88] Janos Mayer. Probabilistic constrained programming: A reduced gradient algorithm implemented on pc. Technical Report WP-88-39, International Institute for Applied Systems Analysis (IIASA), Austria, 1988.

[MOS10] MOSEK ApS. *The MOSEK optimization tools manual. Version 6.0 (Revision 61)*, 1998-2010. `http://www.optirisk-systems.com/manuals/MOSEK.pdf` (Accessed 17 August 2010).

[MS05] Patrice Marcotte and Gilles Savard. *Graph Theory and Combinatorial Optimization*, chapter Bilevel programming: A combinatorial perspective, pages 191–217. Springer US, 2005.

[MT77] Silvano Martello and Paolo Toth. An upper bound for the zero-one knapsack problem and a branch and bound algorithm. *European Journal of Operational Research*, 1(3):169–175, 1977.

[MW65] Bruce L. Miller and Harvey M. Wagner. Chance constrained programming with joint constraints. *Operations Research*, 13(6):930–945, 1965.

[MW97] David P. Morton and R. Kevin Wood. *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search*, chapter On a stochastic knapsack problem and generalizations, pages 149–168. Kluwer Academic Publishers, 1997.

[MZ96] Patrice Marcotte and Dao Li Zhu. Exact and inexact penalty methods for the generalized bilevel programming problem. *Mathematical Programming*, 74(2):141–157, 1996.

[Nem03] Arkadi Nemirovski. On tractable approximations of randomly perturbed convex constraints. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, volume 3, pages 2419–2422, 2003.

[NH76] Mikhail Borisovich Nevel'son and Rafail Zalmanovich Has'minskii. Stochastic approximation and recursive estimation. In *Translations of Mathematical Monographs*. American Mathematical Society, 1976.

[NS04] Arkadi Nemirovski and Alexander Shapiro. *Probabilistic and Randomized Methods for Design under Uncertainty (eds.: G. Calafiore and F. Dabbene)*, chapter Scenario approximation of chance constraints, pages 3–48. Springer London, 2004.

[NS06] Arkadi Nemirovski and Alexander Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2006.

[NS07] Lewis Ntaimo and Suvrajeet Sen. A branch-and-cut algorithm for two-stage stochastic mixed-binary programs with continuous first-stage variables. *Int. J. Comput. Sci. Eng.*, 3(3):232–241, 2007.

[NS08] Lewis Ntaimo and Suvrajeet Sen. A comparative study of decomposition algorithms for stochastic combinatorial optimization. *Computational Optimization and Applications*, 40(3):299–319, 2008.

[NT08] Lewis Ntaimo and Matthew W. Tanner. Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs. *Journal of Global Optimization*, 41(3):365–384, 2008.

[NW86] Larry Nazareth and Roger J-B. Wets. Algorithms for stochastic programs: The case of non-stochastic tenders. *Mathematical Programming Studies*, 28:1–28, 1986.

[Ols76] Paul Olsen. Discretizations of multistage stochastic programming problems. *Mathematical Programming Studies*, 6, 1976.

[OPS10] Osman Y. Özaltin, Oleg A. Prokopyev, and Andrew J. Schaefer. The bilevel knapsack problem with stochastic right-hand sides. *Operations Research Letters*, 38(4):328–333, 2010.

[OZ95] Jiri Outrata and Jochem Zowe. A numerical approach to optimization problems with variational inequality constraints. *Mathematical Programming*, 68:105–130, 1995.

[Pan93] V. M. Panin. Finite algorithm to find the saddle point of a quadratic function subject to linear constraints. *Cybernetics and Systems Analysis*, 29(1):35–46, 1993.

[PFaS96] Erica L. Plambeck, Bor-Ruey Fu, and Stephen M. Robinson andRajah Suri. Sample-path optimization of convex stochastic performance functions. *Mathematical Programming*, 75(2):137–176, 1996.

[Pol90] Boris T. Polyak. New method of stochastic approximation type. *Automation and Remote Control*, 51:937–946, 1990.

[Pré70] András Prékopa. On probabilistic constrained programming. In *Proceedings of the Princeton Symposium on Mathematical Programming*, pages 113–138. Princeton University Press, 1970.

[Pré72] András Prékopa. A class of stochastic programming decision problems. *Mathematische Operationsforschung und Statistik*, 3(5):349–354, 1972.

[Pré90] András Prékopa. Dual method for the solution of a one-stage stochastic programming problem with random rhs obeying a discrete probability distribution. *Mathematical Methods of Operations Research*, 34(6):441–461, 1990.

[Pré95] András Prékopa. *Stochastic Programming.* Kluwer Academic Publishers, 1995.

[PVB98] András Prékopa, Béla Vizvári, and Tamás Badics. *New Trends in Mathematical Programming (eds.: F. Giannessia et al.)*, chapter Programming under probabilistic constraint with discrete random variables, pages 235–255. Kluwer Academic Publishers, 1998.

[PW97] Michael Patriksson and Laura Wynter. Stochastic nonlinear bilevel programming. Technical report, PRiSM, Université de Versailles Saint-Quentin-en-Yvelines, 1997.

[PW99] Michael Patriksson and Laura Wynter. Stochastic mathematical programs with equilibrium constraints. *Operations Research Letters*, 25(4):159–167, 1999.

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[RR02] Vicente Rico-Ramirez. Two-stage stochastic linear programming: A tutorial. *SIAG/OPT News-and-Views Newsletter*, 13(1):8–14, 2002.

[RS06] R. Ravi and Amitabh Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *Mathematical Programming*, 108(1):97–114, 2006.

[Rus86] Andrzej Ruszczyński. A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming*, 35:309–333, 1986.

[Rus93] Andrzej Ruszczyński. Regularized decomposition of stochastic programs: Algorithmic techniques and numerical results. Technical Report WP-93-21, International Institute for Applied Systems Analysis (IIASA), Austria, 1993. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.3955&rep=rep1&type=pdf` (Accessed 17 August 2010).

[Rus02] Andrzej Ruszczyński. Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. *Mathematical Programming*, 93(2):195–215, 2002.

[Sav89] Gilles Savard. *Contributions à la programmation mathématique a deux niveaux*. PhD thesis, Ecole Polytechnique, Université de Montréal, 1989.

[Sch93] Rüdiger Schultz. Continuity properties of expectation functions in stochastic integer programming. *Mathematics of Operations Research*, 18(3):578–589, 1993.

[Sch95] Rüdiger Schultz. On structure and stability in stochastic programs with random technology matrix and complete integer recourse. *Mathematical Programming*, 70:73–89, 1995.

[SDR09] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. Lectures on stochastic programming: Modeling and theory. In *MPS/SIAM Series on Optimization*, volume 9. SIAM-Society for Industrial and Applied Mathematics, 2009.

[SH98] Alexander Shapiro and Tito Homem-de-Mello. A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming*, 81(3):301–325, 1998.

[SH05] Suvrajeet Sen and Julia L. Higle. The C^3 theorem and a D^2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104(1):1–20, 2005.

[Sha96] Alexander Shapiro. Simulation-based optimization: Convergence analysis and statistical inference. *Stochastic Models*, 12(3):425–454, 1996.

[Sha07] Alexander Shapiro. Stochastic programming approach to optimization under uncertainty. *Mathematical Programming*, 112(1):183–220, 2007.

[SIB97] Kiyotaka Shimizu, Yo Ishizuka, and Jonathan F. Bard. *Nondifferentiable and two-level mathematical programming.* Kluwer Academic Publishers, 1997.

[Sio58] Maurice Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 81(1):171–176, 1958.

[SS80] Hanif D. Sherali and C. M. Shetty. A finitely convergent algorithm for bilinear programming problems using polar cuts and disjunctive face cuts. *Mathematical Programming*, 19(1):14–31, 1980.

[SS04] Chaitanya Swamy and David B. Shmoys. The sample average approximation method for 2-stage stochastic optimization. `http://www.math.uwaterloo.ca/~cswamy/papers/SAAproof.pdf` (2008 version; Accessed 17 August 2010), 2004.

[SS06a] Suvrajeet Sen and Hanif D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223, 2006.

[SS06b] David B. Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *Journal of the ACM*, 53(6):978–1012, 2006.

[SS06c] Chaitanya Swamy and David B. Shmoys. Algorithms column: Approximation algorithms for 2-stage stochastic optimization problems. *ACM SIGACT News*, 37(1):33–46, 2006.

[SS07] David B. Shmoys and Mauro Sozio. Approximation algorithms for 2-stage stochastic scheduling problems. *Lecture Notes in Computer Science*, 4513:145–157, 2007.

[SSL07] Xiaoling Sun, Hongbo Sheng, and Duan Li. An exact algorithm for 0-1 polynomial knapsack problems. *Journal of Industrial and Management Optimization*, 3(2):223–232, 2007.

[SSvdV98] Rüdiger Schultz, Leen Stougie, and Maarten H. van der Vlerk. Solving stochastic programs with integer recourse by enumeration: a framework using gröbner basis reductions. *Math. Program.*, 83(2):229–252, 1998.

[Sta52] Heinrich Stackelberg. *The theory of market economy.* Oxford: Oxford University Press, 1952.

[SvdV03] Leen Stougie and Maarten H. van der Vlerk. Approximation in stochastic integer programming. Technical Report 03A14, University of Groningen, Research Institute SOM, 2003.

[SW69] Richard Van Slyke and Roger J-B. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663, 1969.

[SZH09] Suvrajeet Sen, Zhihong Zhou, and Kai Huang. Enhancements of two-stage stochastic decomposition. *Computers & Operations Research*, 36(8):2434–2439, 2009.

[Tam76] Ebu Tamm. The quasiconvexity of the probability function and the quantile function. *Eesti NSV Teaduste Akademia Toimetised (News of the Estonian Academy of Sciences) Füüs. Mat.*, 25(2):141–145, 1976.

[Tam77] Ebu Tamm. On g-concave functions and probability measures. *Eesti NSV Teaduste Akademia Toimetised (News of the Estonian Academy of Sciences) Füüs. Mat.*, 26(4):376–379, 1977.

[VAK+03] Bram Verweij, Shabbir Ahmed, Anton J. Kleywegt, George Nemhauser, and Alexander Shapiro. The sample average approximation method applied to stochastic routing problems: A computational study. *Computational Optimization and Applications*, 24:289–333, 2003.

[VC94] Luís N. Vicente and Paul H. Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global Optimization*, 5(3):291–306, 1994.

[vdV95] Maarten H. van der Vlerk. *Stochastic Programming with Integer Recourse.* PhD thesis, University of Groningen, The Netherlands, 1995.

[vdV04] Maarten H. van der Vlerk. Convex approximations for complete integer recourse models. *Mathematical Programming*, 99(2):297–310, 2004.

[vN28] John von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[VS77] Harish Vaish and C. M. Shetty. A cutting plane algorithm for the bilinear programming problem. *Naval Research Logistics*, 24(1):83–94, 1977.

[VSJ96] Luís N. Vicente, Gilles Savard, and Joaquim J. Júdice. Discrete linear bilevel programming problem. *Journal of Optimization Theory and Applications*, 89(3):597–614, 1996.

[WA93] D. J. White and G. Anandalingam. A penalty function approach for solving bi-level linear programs. *Journal of Global Optimization*, 3(4):397–419, 1993.

[Wer04] Adrian S. Werner. *Bilevel stochastic programming problems: analysis and application to telecommunications*. PhD thesis, Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, 2004.

[Wet66] Roger Wets. Programming under uncertainty: The complete problem. *Probability Theory and Related Fields*, 4(4):316–339, 1966.

[Wet83] Roger J-B. Wets. Solving stochastic programs with simple recourse. *Stochastics*, 10:219–242, 1983.

[Wol80] Richard D. Wollmer. Two stage linear programming under uncertainty with 0-1 integer first stage variables. *Mathematical Programming*, 19(1):279–288, 1980.

[WW67] David W. Walkup and Roger J.-B. Wets. Stochastic programs with recourse. *SIAM. Journal on Applied Mathematics*, 15(5):1299–1314, 1967.

[WZ98] Thomas Winter and Uwe T. Zimmermann. Discrete online and real-time optimization. In *Proceedings of the 15th IFIP World Computer Congress, Budapest/Vienna*, 1998. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.3.2549&rep=rep1&type=pdf`.

[YK91] Yasutoshi Yajima and Hiroshi Konno. Efficient algorithms for solving rank two and rank three bilinear programming problems. *Journal of Global Optimization*, 1(2):155–171, 1991.

[Zha09] Xiaoyan Zhang. A note on Stackelberg games and Nash games. *International Journal of Business and Management*, 1(5), 2009.