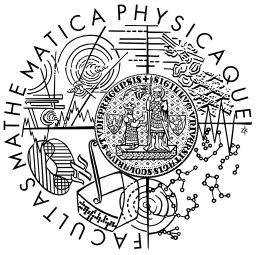


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Bc. Ivona Vlčková

Algoritmy pro řešení stochastických dvoustupňových úloh

Katedra pravděpodobnosti a matematické statistiky

Vedoucí diplomové práce:	doc. RNDr. Ing. Miloš Kopa, Ph.D.
Studijní program:	Matematika
Studijní obor:	Pravděpodobnost, matematická statistika a ekonometrie

Praha 2017

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Na tomto místě bych ráda poděkovala svému konzultantovi RNDr. Václavovi Kozmíkovi, Ph.D. za odborné vedení, rady a neocenitelnou pomoc poskytnutou během přípravy této práce. Děkuji také svému vedoucímu doc. RNDr. Ing. Milošovi Kopovi za jeho ochotu a čas, který mi věnoval během konzultací. Dále mé díky patří Dr. Sebastianovi Vitalimu, Ph.D., který mi pomohl propojit GAMS a Matlab. V neposlední řadě bych chtěla poděkovat své rodině a přátelům za jejich podporu během celého studia.

Název práce: Algoritmy pro řešení stochastických dvoustupňových úloh

Autor: Bc. Ivona Vlčková

Katedra: Katedra pravděpodobnosti a matematické statistiky

Vedoucí diplomové práce: doc. RNDr. Ing. Miloš Kopa, Ph.D., Katedra pravděpodobnosti a matematické statistiky

Abstrakt: V této práci se zabýváme algoritmy pro řešení dvoustupňových stochastických úloh. V první kapitole jsou uvedeny teoretické vlastnosti účelové funkce a množiny omezení, jenž jsou nezbytné pro pochopení fungování algoritmů. V závěru jsou diskutovány podmínky optimality. Druhá kapitola se zabývá algoritmy pro řešení úloh s lineární účelovou funkcí. V první části je podrobně vysvětlena základní forma L-shaped algoritmu. Druhá část je věnována algoritmu stochastické dekompozice včetně popisu regularizované verze. Poslední kapitola práce slouží k praktickému porovnání uvedených algoritmů na třech aplikačních příkladech. Každá úloha je nejprve teoreticky popsána a poté vyřešena oběma uvedenými algoritmy.

Klíčová slova: dvoustupňové stochastické programování, L-shaped algoritmus, stochastická dekompozice

Title: Algorithms for solving two-stage stochastic programs

Author: Bc. Ivona Vlčková

Department: Department of Probability and Mathematical Statistics

Supervisor: doc. RNDr. Ing. Miloš Kopa, Ph.D., Department of Probability and Mathematical Statistics

Abstract: The thesis deals with the algorithms for two-stage stochastic programs. The first chapter considers the basic properties and theory. Specifically, we introduce the properites of the feasibility region and the objective function. Further, optimality conditions are discussed. In the second chapter we present algoritms which can be used to solve two-stage linear programs with fixed recourse. In the first section the basic L-shaped method is described in detail. The second section provides an explanation of the Stochastic Decomposition algorithm with the inclusion of a regularization term. The last chapter presents computational results. Three practical examples are provided both with a brief description of the problem and solutions by the studied algorithms.

Keywords: two-stage stochastic programming, L-shaped method, Stochastic Decomposition

Obsah

Úvod

V této práci je naším cílem popsat algoritmy pro řešení dvoustupňových stochastických úloh. Tyto stupně se vztahují ke dvěma časovým okamžikům, současnosti a budoucnosti. Klasickým příkladem dvoustupňových úloh je problém prodavače novin. Ten se musí v současnosti rozhodnout, kolik novin na svůj stánek nakoupí, aniž by věděl, jaká bude další den po novinách poptávka. Dvoustupňové úlohy sice nedají odpověď na otázku, kolik zákazníků na stánek přijde, ale hledají optimální řešení ve střední hodnotě vzhledem k pravděpodobnostnímu rozdělení poptávky. Místo střední hodnoty lze v úlohách v účelové funkci uvažovat také různé míry rizika.

V první kapitole se budeme věnovat formulaci dvoustupňové úlohy a vlastnostem účelové funkce ve druhém stupni. Dále si ukážeme vlastnosti množiny omezení, duální reprezentaci úlohy a uvedeme si podmínky optimality. Tyto teoretické poznatky využijeme při popisu fungování algoritmů v další kapitole.

V druhé kapitole se zaměříme na algoritmy, které řeší stochastické lineární úlohy s pevnou kompenzací. V práci se budeme zabývat úlohami s konečným počtem realizací náhodného vektoru, které lze řešit také deterministickým ekvivalentem. Pokud je těchto realizací mnoho, stává se klasická deterministická úloha velkou a obtížně řešitelnou. Při implementaci algoritmů je proto vhodné vzít v potaz speciální strukturu konkrétní úlohy. Pokud tyto struktury ignorujeme, dochází často k neefektivnímu hledání optimálního řešení zejména z hlediska časové náročnosti. Efektivním způsobem je složitou úlohu rozdělit na více menších podúloh, které lze řešit snadněji. Mezi dekompoziční algoritmy, které tuto techniku využívají, patří L-shaped algoritmus, jehož základní forma je podrobně popsána v první části kapitoly. Tento algoritmus je deterministický, neboť řeší úlohu pro konkrétní množinu scénářů. V další části kapitoly se zabýváme algoritmem stochastické dekompozice, který si realizace náhodného vektoru generuje postupně. Pokud je rozdělení náhodného vektoru známo, nepotřebuje mít konkrétní množinu scénářů, na nichž úlohu spočítá. Naopak si v každé iteraci generuje vždy pouze jednu realizaci náhodného vektoru, postupně zpřesňuje odhad účelové funkce a je zastaven na základě ukončovacích kritérií. Tato kritéria jsou složitější než v případě deterministických algoritmů a základní myšlenky, jak algoritmus stochastické dekompozice zastavit, si uvedeme na konci kapitoly.

Poslední kapitola je věnována praktické části. Prezentuje výsledky získané aplikací obou algoritmů na třech různých příkladech. Kromě zmíněného problému prodavače novin se diskutuje také jeho rozšíření v úloze květinářky. Její páteční rozhodnutí ovlivňuje sobotní i nedělní prodej růží. Tato úloha obsahuje dva budoucí okamžiky, je tedy obecně třístupňová. Ukážeme si, jak ji lze převést na dvoustupňovou, aby byla studovanými algoritmy řešitelná. Dále se zabýváme nalezením optimálního portfolia, které minimalizuje riziko z investice. Jedná se o „mean-risk“ model, ve kterém riziko představuje CVaR. Všechny příklady jsou nejprve popsány, poté jsou na nich algoritmy srovnány a diskutovány jejich výhody a nevýhody.

V příloze jsou stručně uvedeny definice a věty, na které odkazujeme především v první teoretické kapitole.

1. Základní vlastnosti dvoustupňových stochastických úloh

V této kapitole se budeme zabývat vlastnostmi dvoustupňových stochastických úloh. V prvních třech částech, kde se uvádí formulace úlohy a vlastnosti množin omezení a účelové funkce, je koncepce převzata z [4]. V poslední části zabývající se podmínkami optimality, se čerpá především z [17]. Důraz je kladen na teorii subdiferenciálů, která bude využita v následujících kapitolách.

Celá teorie předpokládá základní znalosti z teorie optimalizace, kde můžeme odkázat např. na studijní materiál k přednášce Teorie optimalizace na MFF UK [14]. Mezi další zdroje, které byly při budování této teorie použity, uveďme [11] a také přehledně zpracovanou teorii v českém jazyce v první kapitole diplomové práce [2].

1.1 Formulace problému

Nejprve uvedeme formální zápis dvoustupňové stochastické úlohy. V celé této práci budeme uvažovat úlohy s pevnou maticí W (angl. fixed recourse):

$$\begin{aligned} \min_{x} \quad & \phi = c^T x + \mathsf{E}_{\boldsymbol{\xi}}[\min q(\omega)^T y(\omega)] \\ \text{za podmínek} \quad & Ax = b \\ & T(\omega)x + Wy(\omega) = h(\omega) \text{ s. j.} \\ & x \geq 0, \quad y(\omega) \geq 0 \text{ s. j.}, \end{aligned} \tag{1.1}$$

kde pro upřesnění dimenzí uvádíme rozměry vektorů a matic, tedy $x \in \mathbb{R}^{n_1}$, $c \in \mathbb{R}^{n_1}$, $b \in \mathbb{R}^{m_1}$, $A \in \mathbb{R}^{m_1 \times n_1}$. Dále pro každý náhodný jev $\omega \in \Omega : y(\omega) \in \mathbb{R}^{n_2}$, $q(\omega) \in \mathbb{R}^{n_2}$, $h(\omega) \in \mathbb{R}^{m_2}$, $T(\omega) \in \mathbb{R}^{m_2 \times n_1}$, $W \in \mathbb{R}^{m_2 \times n_2}$.

Náhodnou složku problému můžeme tedy celkově zapsat do jednoho vektoru $\xi^T(\omega) = (q(\omega)^T, h(\omega)^T, T_{1.}(\omega), \ldots, T_{m_2.}(\omega))$, jenž má $N = n_2 + m_2 + (m_2 \times n_1)$ složek a kde $T_{i.}(\omega)$ představuje i-tý řádek matice $T(\omega)$. Dimenze jsou takto označeny zcela záměrně, neboť indexy označují stupeň, ke kterému se dané vektory a matice pojí, přičemž matice $T(\omega)$ je maticí, která oba stupně spojuje.

Dále $\mathsf{E}_{\boldsymbol{\xi}}$ představuje střední hodnotu vzhledem k náhodnému vektoru $\boldsymbol{\xi}$ s nosičem $\Xi \subset \mathbb{R}^N$. Nosič náhodného vektoru je nejmenší uzavřená množina taková, že platí $\mathsf{P}(\boldsymbol{\xi} \in \Xi) = 1$. Navíc všechny podmínky problému (1.1) platí skoro jistě.

Ekvivalentní deterministická úloha

Hlavní myšlenkou dvoustupňových úloh je najít optimální řešení takové, že minimalizuje fixní náklady v prvním stupni plus očekávánou hodnotu v druhém stupni úlohy. Druhý stupeň nám tak určuje v těchto úlohách onu stochastiku, neboť se jedná o realizace v budoucnosti. Tyto realizace se odvíjí na základě rozhodnutí vykonaného v prvním stupni a náhodě, jež je v současnosti neznámá.

Pro úlohu (1.1) můžeme definovat ekvivalentní deterministickou úlohu jako

$$\begin{aligned} \min_{x} \quad & \phi = c^T x + \mathcal{Q}(x) \\ \text{za podmínek} \quad & Ax = b \\ & x \geq 0, \end{aligned} \tag{1.2}$$

kde optimální řešení problému druhého stupně je

$$\mathcal{Q}(x) = \mathsf{E}_{\boldsymbol{\xi}}\, Q(x, \boldsymbol{\xi}) \tag{1.3}$$

a pro dané ω definujeme

$$Q(x, \xi(\omega)) = \min_{y} \{q(\omega)^T y |\, Wy = h(\omega) - T(\omega)x, y \geq 0\}. \tag{1.4}$$

V celé práci budeme psát tučné $\boldsymbol{\xi}$, pokud se bude jednat o náhodnou veličinu, a netučné ξ, pokud půjde o její realizaci $\xi(\omega)$. Argument ω budeme často vynechávat.

Pokud fixujeme x a minimalizujeme úlohu ve druhém stupni, potom označíme problém jako:

- neomezený, pokud $Q(x,\xi) = -\infty$,
- nepřípustný, pokud $Q(x,\xi) = \infty$.

Je důležité rozlišit, zda má náhodná veličina spojité, nebo diskrétní rozdělení. V této práci uvažujme situaci, kdy je $\boldsymbol{\xi}$ konečná diskrétní náhodná veličina, tedy $\boldsymbol{\xi} \in \Xi$, kde Ξ je konečná množina. Nechť $\boldsymbol{\xi}$ nabývá hodnot $\xi_1, \ldots, \xi_K$ postupně s pravděpodobnostmi $p_1, \ldots, p_K$, potom můžeme střední hodnotu (1.3) zapsat jako váženou sumu $Q(x,\xi_k)$, tedy

$$\mathcal{Q}(x) = \sum_{k=1}^{K} p_k Q(x,\xi_k).$$

Ve výpočtu položme $+\infty + (-\infty) = +\infty$. Tímto dodefinováním mluvíme o konzervativním přístupu, neboť se zamítne jakékoliv rozhodnutí v prvním stupni úlohy, které by mohlo vést k tomu, že by úloha v druhém stupni nebyla definována. Rozhodnutí x se tak zamítne, ačkoliv existují takové realizace náhodného vektoru, pro něž by úloha ve druhém stupni, tedy i celá úloha (1.2), neomezeně klesala.

Abychom dospěli k diskrétnímu rozdělení s konečnou množinou stavů, lze budoucí náhodné veličiny aproximovat pomocí scénářů. Jednotlivé scénáře můžeme generovat, nebo za ně považujeme napozorované historické hodnoty. Pokud máme konečný počet scénářů vyjadřujících budoucnost a všechny úlohy v druhém stupni jsou lineární, lze sestavit ekvivalentní lineární deterministickou úlohu. Index scénáře budeme značit dolním indexem. Původní problém (1.1) lze tak přepsat do deterministického ekvivalentu následujícím způsobem

$$\begin{aligned} \min_{x, y_1, \ldots, y_K} \quad & \phi = c^T x + \sum_{k=1}^{K} p_k q_k^T y_k \\ \text{za podmínek} \quad & Ax = b \\ & T_k x + W y_k = h_k, \quad k = 1, \ldots, K \\ & x \geq 0, \quad y_k \geq 0, \quad k = 1, \ldots, K. \end{aligned} \tag{1.5}$$

Zde součet $+\infty + (-\infty)$ není potřeba dodefinovat, neboť pro $x \in K_1$ musí platit podmínky pro každé k. Pokud je problém nepřípustný pro nějaké k, t.j. $Q(x,\xi_k) = +\infty$, potom je nepřípustný i celý problém (1.5) a optimální hodnota je automaticky $+\infty$.

1.2 Množiny omezení

V této podkapitole se budeme zabývat vlastnostmi množin omezení, nejprve si definujme množiny přípustných řešení pro první a druhý stupeň úlohy jako

$$K_1 = \{x | Ax = b, x \geq 0\},$$

$$K_2 = \{x | \mathcal{Q}(x) < +\infty\}.$$

Tato definice nám umožňuje zapsat výše uvedenou deterministickou úlohu (1.2) pomocí množin K_1 a K_2 následujícím způsobem:

$$\begin{array}{rl} \min\limits_x & c^T x + \mathcal{Q}(x) \\ \text{za podmínek} & Ax = b \\ & x \geq 0 \end{array} \iff \begin{array}{rl} \min\limits_x & c^T x + \mathcal{Q}(x) \\ \text{za podmínek} & x \in K_1 \cap K_2. \end{array}$$

Chceme navíc definovat taková x, která budou přípustná v druhém stupni, aniž bychom museli počítat hodnotu $\mathcal{Q}(x)$.

Definujme

- $K_2(\xi) = \{x | Q(x,\xi) < +\infty\}$, množinu takových rozhodnutí x, pro než je $Q(x,\xi)$ přípustné,

- $K_2^P = \{x | \forall\, \xi \in \Xi \text{ existuje } y \geq 0 : Wy = h(\xi) - T(\xi)x\} = \bigcap_{\xi \in \Xi} K_2(\xi)$, množinu takových rozhodnutí x, že přípustné řešení v druhém stupni lze nalézt pro $\forall\, \xi \in \Xi$.

Poznamenejme, že rozhodnutí y se pro různé hodnoty ξ může lišit.

Tvrzení 1. *[4]*

a) Pro všechna $\xi \in \Xi$ je množina $K_2(\xi)$ uzavřená konvexní polyedrická množina, proto také množina K_2^P je uzavřená a konvexní.

b) Pokud Ξ je konečná množina, pak je K_2^P navíc polyedrická a platí $K_2^P = K_2$.

Důkaz. a)Pro každé ξ je množina $K_2(\xi)$ definována pomocí lineárních omezení (rovnosti a neostré nerovnosti), tedy se jedná o uzavřenou konvexní polyedrickou množinu. Spočetný průnik konvexních množin je opět konvexní množina, což dohromady stačí k důkazu první částí tvrzení.

b) Dané x je prvkem množiny K_2, jestliže funkce $\mathcal{Q}(x)$ je shora omezená. $\mathcal{Q}(x)$ je součet konečně mnoha hodnot $Q(x,\xi)$ vážených pomocí nezáporných pravděpodobností. Díky zavedené konvenci $+\infty + (-\infty) = +\infty$ je tedy $\mathcal{Q}(x)$ shora

omezená, pokud každá $Q(x,\xi)$ je shora omezená. To znamená, že x je prvkem množiny $K_2(\xi)$ pro $\forall\xi$, neboli x patří do množiny K_2^P. Naopak, pokud je x prvkem K_2^P, potom je $Q(x,\xi)$ shora omezená pro všechny hodnoty, což implikuje, že $\mathcal{Q}(x)$ je konečná a x je tak prvkem K_2.

□

V druhé části tvrzení 1 se předpokládalo diskrétní rozdělení $\boldsymbol{\xi}$ na konečné množině, pro spojité rozdělení $\boldsymbol{\xi}$ obecně vztah $K_2^P = K_2$ neplatí. Ukážeme ilustrační protipříklad, uvedený v [4], jenž nesplňuje tento předpoklad ani předpoklad na pevnou matici W.

Předpokládejme, že funkce ve druhém stupni je definována jako

$$Q(x,\xi) = \min_y\{y|\xi y = 1 - x, y \geq 0\},$$

kde $\boldsymbol{\xi}$ má trojúhelníkové rozdělení na [0,1], konkrétně $P(\boldsymbol{\xi} \leq u) = u^2$. Tedy matice W je rozměru 1×1 a je jediná náhodná, $W(\omega) = \xi$.

Pro $\forall\xi \in (0,1]$ optimální řešení $Q(x,\xi)$ je $y = \frac{1-x}{\xi}, y \geq 0$, takže

$$K_2(\xi) = \{x|x \leq 1\}$$

a

$$Q(x,\xi) = \frac{1-x}{\xi}, \text{ pro } x \leq 1.$$

Nechť $\xi = 0$. Pokud neplatí $x = 1$, potom neexistuje žádné y splňující $0 \cdot y = 1 - x$. Tedy

$$K_2(0) = \{x|x = 1\}.$$

Tedy pro $x \neq 1$ by $Q(x,0)$ nabývalo hodnoty $+\infty$. Jelikož pravděpodobnost toho, že $\boldsymbol{\xi} = 0$, je nulová, dodefinujeme $Q(x,0) = 0$, což odpovídá konvenci $0 \cdot \infty = 0$. Proto

$$K_2^P = \{x|x = 1\} \cap \{x|x \leq 1\} = \{x|x = 1\},$$

zatímco

$$\mathcal{Q}(x) = \int_0^1 \frac{1-x}{\xi} \cdot 2\xi d\xi = 2(1-x) \text{ pro } \forall x \leq 1,$$

tedy $K_2 = \{x|x \leq 1\}$ a $K_2^P \subsetneq K_2$. Rozdíl mezi množinami je způsoben tím, že bod není v množině K_2^P, jakmile je nepřípustným řešením pro nějaké ξ nehledě na rozdělení $\boldsymbol{\xi}$. Naopak K_2 neřeší nepřípustnost, které se nabývá s nulovou pravděpodobností.

Je třeba říct, že tato situace nastane zřídka, pokud je matice W pevná, což je předpoklad, který v celé práci uvažujeme za splněný. K další nesnázi může dojít, pokud jsou všechny hodnoty $Q(x,\xi)$ shora omezené s pravděpodobností 1, avšak $\mathcal{Q}(x)$, tedy střední hodnota $Q(x,\boldsymbol{\xi})$, je neomezená.

Jelikož v další části práce budeme uvažovat hlavně diskrétně rozdělený náhodný vektor $\boldsymbol{\xi}$, závěry týkající se spojitě rozděleného $\boldsymbol{\xi}$ zde nebudeme uvádět, lze je nalézt včetně důkazů např. v [4]. Poznamenejme, že potom závěry týkající se rovnosti množin $K_2 = K_2^P$ či jejich specifických vlastností budou platit jen při splnění dalších podmínek. Pro spojitě rozdělený náhodný vektor či diskrétně rozdělený s nekonečným nosičem, je jedním z hlavních předpokladů konečnost

jeho druhých momentů. Uvedeme si pouze lemma, kde konečnost druhých momentů vede ke konečnosti účelové funkce $\mathcal{Q}(x)$. Jejími vlastnostmi se budeme zabývat v následující části.

Lemma 2. *[4] Pokud $\boldsymbol{\xi}$ má konečné druhé momenty, potom platí*

$$P(\omega|Q(x,\boldsymbol{\xi}) < \infty) = 1 \Rightarrow \mathcal{Q}(x) < \infty.$$

1.3 Vlastnosti účelové funkce ve druhém stupni

V této kapitole si uvedeme vlastnosti účelové funkce ve druhém stupni. Nejprve připomeňme, že v celé práci předpokládáme, že matice omezení W nezávisí na realizaci ξ. Říkáme pak, že problém (1.2) má **pevnou kompenzaci**.

Než přejdeme k obecným vlastnostem $\mathcal{Q}(x)$, uvedeme si speciální případy úlohy (1.2), které usnadňují počítačovou implementaci jednotlivých algoritmů. Nespornou výhodou je například situace, kdy všechna rozhodnutí x, která jsou přípustná pro úlohu v prvním stupni, mají přípustné řešení i ve druhém stupni.

Definice 1. *[4] Řekneme, že dvoustupňový problém (1.2) má **relativně úplnou kompenzaci**, pokud pro všechna $x \in K_1$ a skoro všechna ξ existuje $y \geq 0$ splňující $Wy = h(\xi) - T(\xi)x$, tedy $K_1 \subset K_2$. Problém (1.2) má **úplnou kompenzaci**, pokud pro každé χ existuje $y \geq 0$ takové, že platí $Wy = \chi$.*

Výhody relativně úplné kompenzace budeme diskutovat v další kapitole. Speciálním případem relativně úplné kompenzace je úplná kompenzace, kterou lze často identifikovat ze struktury matice W. Úplná kompenzace se dá zapsat také jako pos $W = \mathbb{R}^{m_2}$.

Speciálním případem úplné kompenzace je **jednoduchá kompenzace**, kdy $W = [I, -I]$, stejně tak $\boldsymbol{y} = (\boldsymbol{y}^+, \boldsymbol{y}^-)$ a $\boldsymbol{q} = (\boldsymbol{q}^+, \boldsymbol{q}^-)$. Potom lze úloha (1.4) zapsat následujícím způsobem:

$$\begin{aligned} Q(x,\xi) = \min_{y^+,y^-} \quad & (q^+)^T y^+ + (q^-)^T y^- \\ \text{za podmínek} \quad & y^+ - y^- = h(\xi) - T(\xi)x \\ & y^+, y^- \geq 0. \end{aligned} \tag{1.6}$$

Pro jednoduchou kompenzaci navíc platí následující tvrzení o konečnosti $\mathcal{Q}(x)$:

Tvrzení 3. *[4] Nechť dvoustupňová stochastická úloha (1.1) je přípustná, má jednoduchou kompenzaci a nechť $\boldsymbol{\xi}$ má konečné druhé momenty. Potom $\mathcal{Q}(x)$ je konečná, právě když platí $\boldsymbol{q}_i^+ + \boldsymbol{q}_i^- \geq 0$ skoro jistě.*

Důkaz. Nechť pro nějaké i platí, že $q_i^+(\omega) + q_i^-(\omega) < 0$ pro $\forall\, \omega \in \Omega_1 \subset \Omega$, kde $\mathsf{P}(\Omega_1) > 0$. Pro přípustné y platí, že také bod $(y_i^+ + \Delta, y_i^- + \Delta)$ je pro $\Delta \geq 0$ přípustný. Potom dostáváme, že pro všechna přípustná x úlohy (1.1) a všechna $\omega \in \Omega_1$ platí, že pro rostoucí $\Delta \geq 0$ účelová funkce klesá:

$$q_i^+(y_i^+ + \Delta) + q_i^-(y_i^- + \Delta) = q_i^+ y_i^+ + q_i^- y_i^- + \underbrace{\Delta}_{\to\infty}\underbrace{(q_i^+ + q_i^-)}_{<0}.$$

Tedy funkce $\mathcal{Q}(x)$ není konečná.

Nechť $\boldsymbol{q}_i^+ + \boldsymbol{q}_i^- \geq 0$ skoro jistě, potom

$$Q(x,\xi(\omega)) = \sum_{i=1}^{m_2} [q_i^+(h_i - T_{i.}x)^+ + q_i^-(-h_i + T_{i.}x)^+],$$

kde výraz $(x)^+$ znamená

$$(x)^+ = \begin{cases} 0 & \text{pokud } x < 0, \\ x & \text{pokud } x \geq 0. \end{cases}$$

Problém je totiž separovatelný a každé $y_i^+(\omega)$ a $y_i^-(\omega)$ je závislé na znaménku $h_i(\omega) - T_{i.}(\omega)x$, tj. pokud

- $(h_i(\omega) - T_{i.}(\omega)x) \geq 0 \Rightarrow y_i^+ = h_i(\omega) - T_{i.}(\omega)x,\ y_i^- = 0,$
- $(h_i(\omega) - T_{i.}(\omega)x) \leq 0 \Rightarrow y_i^+ = 0,\ y_i^- = T_{i.}(\omega)x - h_i(\omega).$

Protože $Q(x, \xi(\omega))$ je konečná pro skoro všechna ω, pomocí lemmatu 2 je tvrzení dokázáno.

□

Dříve než přejdeme ke konkrétním vlastnostem funkce $Q(x,\xi)$, uvedeme si zde duální reprezentaci druhého stupně úlohy (1.1). Primární úloha ve druhém stupni má tvar:

$$\begin{aligned} \min_y \quad & q^T y \\ \text{za podmínek} \quad & Wy = h - Tx \\ & y \geq 0. \end{aligned} \tag{1.7}$$

K ní duální úloha je pak tvaru:

$$\begin{aligned} \max_\pi \quad & \pi^T(h - Tx) \\ \text{za podmínek} \quad & W^T\pi \leq q \\ & \pi \in \mathbb{R}^{m_2}. \end{aligned} \tag{1.8}$$

Z duality plyne, že pokud má primární i duální úloha alespoň jedno přípustné řešení, potom mají obě úlohy i optimální řešení a optimální hodnoty jejich účelových funkcí se rovnají.

Množina přípustných řešení Π definovaná nerovnostmi $W^T\pi \leq q$ duální úlohy (1.8) nezávisí na x, navíc při fixovaném ξ je tato množina průnikem konečně mnoha uzavřených poloprostorů, tedy se jedná o konvexní polyedrickou množinu. Množina $\Pi(q,W) = \{\pi | W^T\pi \leq q\}$ je navíc uzavřená a pokud je neprázdná, pak při fixním ξ platí pro libovolné x, že $Q(x,\xi) > -\infty$. Naopak pokud je množina prázdná, potom je primární problém neomezený.

Poznámka. Pokud má problém (1.1) relativně úplnou kompenzaci, potom uvažujme přípustnou bázi L primární úlohy (1.7), viz definice P.1. Pokud máme pevnou matici W, je podle definice P.2 pro fixní ξ vektor $\pi(L) = \left((W_L)^T\right)^{-1} q_L(\xi)$ duální bazické řešení. Pokud tedy platí $q_L(\xi)^T(W_L)^{-1}W - q^T(\xi) \leq 0^T$, je příslušné duální bazické řešení také přípustné řešení úlohy (1.8) a pro libovolné x je pro fixní ξ funkce $Q(x,\xi)$ konečná.

Položme

$$s_q(\chi) = \inf\{q^T y : Wy = \chi, y \geq 0\}.$$

Jestliže $\Pi(q,W) \neq \emptyset$, potom platí

$$s_q(\chi) = \sup_{\pi \in \Pi(q,W)} \pi^T \chi,$$

tedy $s_q(\cdot)$ je opěrnou funkcí množiny $\Pi(q,W)$. Pokud je tedy množina $\Pi(q,W)$ neprázdná, potom je funkce $s_q(\cdot)$ polyedrická. V následující definici si připomeňme, kdy je funkce polyedrická.

Definice 2. *[19] Funkce $g : \mathbb{R}^m \to \overline{\mathbb{R}}$ je polyedrická, pokud je její epigraf konvexní, uzavřená polyedrická množina a $g(x)$ je konečná alespoň v jednom bodě x (tedy $g(x)$ je vlastní). Jinými slovy můžeme také říct, že funkce je polyedrická právě tehdy, pokud je vlastní, konvexní, zdola spojitá, její doména je konvexní uzavřená polyedrická množina a $g(\cdot)$ je na ní po částech lineární.*

Je vidět, že platí rovnost $Q(x,\xi) = s_q(h - Tx)$ a navíc doména funkce $s_q(\cdot)$ je rovna pozitivnímu obalu matice W, tedy

$$\text{dom } Q(\cdot,\xi) = \{x | h - Tx \in \text{pos } W\}.$$

Následující tvrzení, popisuje vlastnosti funkce $Q(\cdot,\xi)$.

Tvrzení 4. *[19] Pokud je množina $\Pi(q,W)$ neprázdná a problém (1.7) má přípustné řešení alespoň pro jedno x, potom je funkce $Q(\cdot,\xi)$ polyedrická.*

Důkaz. Protože platí rovnost $Q(x,\xi) = s_q(h - Tx)$, vlastnosti funkce $Q(\cdot,\xi)$ plynou z vlastností funkce $s_q(\cdot)$.

□

Konvexita funkce $Q(x,\xi)$ bude v další části práce pro použití algoritmů jedním ze stěžejních předpokladů, proto si nyní tuto vlastnost dokážeme z definice, za předpokladu, že $Q(x,\xi)$ nenabývá hodnoty $-\infty$, tedy problém (1.2) není neomezený. Poznamenejme, že o konvexitě mluvíme vzhledem k x a k parametrům (h,T), v parametru q je naopak funkce $Q(x,\xi)$ konkávní [4].

K důkazu konvexity je třeba ukázat, že $f(\chi) = \min_y\{q^T y | Wy = \chi\}$ je konvexní funkce v χ. Uvažujme dva vektory $\chi_1 \neq \chi_2$ a nějakou konvexní kombinaci $\chi_\lambda = \lambda\chi_1 + (1-\lambda)\chi_2$, $\lambda \in (0,1)$.

Nechť y_1^* a y_2^* jsou optimální řešení úloh $\min_y\{q^T y | Wy = \chi\}$ pro $\chi = \chi_1$ a $\chi = \chi_2$. Potom $\lambda y_1^* + (1-\lambda)y_2^* = y$ je přípustné řešení úlohy $\min_y\{q^T y | Wy = \chi_\lambda\}$neboť

- $y \geq 0$, protože $y_1^* \geq 0, y_2^* \geq 0$ a $\lambda \in [0,1]$.

- $Wy = \chi_\lambda$, protože $W(\lambda y_1^* + (1-\lambda)y_2^*) = \lambda \underbrace{Wy_1^*}_{\chi_1} + (1-\lambda)\underbrace{Wy_2^*}_{\chi_2} = \chi_\lambda$.

Nechť y_λ^* je optimální řešení úlohy $\min_y\{q^T y|Wy = \chi_\lambda\}$. Potom platí

$$f(\chi_\lambda) = q^T y_\lambda^* \leq q^T(\lambda y_1^* + (1-\lambda)y_2^*) = \lambda q^T y_1^* + (1-\lambda)q^T y_2^* = \lambda f(\chi_1) + (1-\lambda)f(\chi_2),$$

kde první nerovnost plyne z optimality. To dokazuje konvexitu účelové funkce ve druhém stupni v (h,T) a v x pro všechna $x \in K = K_1 \cap K_2$.

Podobně pro konkávitu v q je třeba ukázat, že funkce $f(q) = q^T y^*$ je konkávní. Nechť y_1^*, y_2^* a y_λ^* jsou optimální řešení úloh $\min_y\{q^T y|Wy = \chi\}$ pro $q = q_1, q = q_2$ a $q_\lambda = \lambda q_1 + (1-\lambda)q_2$. Potom platí

$$\begin{aligned} q_1^T y_1^* &\leq q_1^T y_\lambda^* \quad [\lambda] \\ q_2^T y_2^* &\leq q_2^T y_\lambda^* \quad [1-\lambda]. \end{aligned}$$

Tedy dohromady $\lambda \underbrace{q_1 y_1^*}_{f(q_1)} + (1-\lambda)\underbrace{q_2 y_2^*}_{f(q_2)} \leq \underbrace{(\lambda q_1^* + (1-\lambda)q_2^*)y_\lambda^*}_{f(\lambda q_1 + (1-\lambda)q_2)}$.

Popsané vlastnosti účelové funkce ve druhém stupni budou využity v následující kapitole v algoritmech určených pro řešení dvoustupňových stochastických úloh tvaru (1.1). Poznamenejme, že v případě, že bychom znali rozdělení $Q(x,\xi)$, bylo by snadné nalézt také $\mathcal{Q}(x)$ či jiný funkcionál, než je střední hodnota. Tzv. distribuční problém, tedy nalezení samotné distribuce, však není triviální a dále se jím zabývat nebudeme. Některé metody nalezení rozdělení lze nalézt např. v [21]. My si dále ukážeme vlastnosti funkce $\mathcal{Q}(x)$, funkcionálu střední hodnoty. Poznamenejme, že v následujících tvrzeních většinou předpokládáme konečné druhé momenty. V celém textu však pracujeme zejména s náhodným vektorem $\boldsymbol{\xi}$, jenž má diskrétní rozdělení s konečným nosičem Ξ, a tedy je tento předpoklad splněn automaticky. Navíc opět předpokládejme, že pro jednotlivé funkce platí $Q(x,\xi) > -\infty$.

Tvrzení 5. *[4] Pro stochastický problém (1.2), kde $\boldsymbol{\xi}$ má konečné druhé momenty, platí*

1. *$\mathcal{Q}(x)$ je Lipschitzovsky konvexní a vlastní.*

2. *Pokud Ξ je konečná množina, potom $\mathcal{Q}(x)$ je po částech lineární.*

3. *Pokud $\boldsymbol{\xi}$ má absolutně spojité rozdělení, $\mathcal{Q}(x)$ je diferencovatelná na K_2.*

Důkaz.
1. Konvexita plyne přímo z definice. Uvažujme body $x_1, x_2 \in K_2$ a $\lambda \in [0,1]$. Potom z konvexity funkce $Q(x,\xi)$ dostáváme

$$\mathcal{Q}(\lambda x_1 + (1-\lambda)x_2) = \mathsf{E}\, Q(\lambda x_1 + (1-\lambda x_2), \boldsymbol{\xi}) \leq \mathsf{E}\,[\lambda Q(x_1, \boldsymbol{\xi}) + (1-\lambda)Q(x_2, \boldsymbol{\xi})].$$

Tím je konvexita funkce $\mathcal{Q}(x)$ dokázána, jelikož

$$\mathsf{E}\,[\lambda Q(x_1, \boldsymbol{\xi}) + (1-\lambda)Q(x_2, \boldsymbol{\xi})] = \lambda \mathcal{Q}(x_1) + (1-\lambda)\mathcal{Q}(x_2).$$

Důkaz Lipschitzovské vlastnosti, tedy:

$$\exists\, M \geq 0 :\| \mathcal{Q}(x_2) - \mathcal{Q}(x_1) \| \leq M \| x_2 - x_1 \| \ \forall\, x_1, x_2 \in K_2,$$

je uveden např. v [17].

Funkce $\mathcal{Q}(x)$ je z definice K_2 shora omezená. Z předpokladu $Q(x,\xi) > -\infty$ a konečnosti druhých momentů je také na doméně omezená zdola, tedy je na K_2 vlastní.

2. Nechť Ξ je konečná množina nabývající hodnot $\xi_1, \ldots, \xi_K$ s příslušnými pravděpodobnostmi, tedy $\mathcal{Q}(x) = \sum_{k=1}^{K} p_k Q(x,\xi_k)$. Každá funkce $Q(x,\xi_k)$ je po částech lineární, tím pádem $\mathcal{Q}(x)$ je také po částech lineární, neboť je součtem (váženým pravděpodobnostmi) konečně mnoha po částech lineárních funkcí.

3. Diferencovatelnost pro spojité rozdělení $\boldsymbol{\xi}$ je dokázáno např. v [10].

□

Tvrzení 5 nám ukazuje rozdíl mezi vlastnostmi účelových funkcí pro diskrétní rozdělení s konečným nosičem a pro spojité rozdělení. Diskrétní rozdělení charakterizuje $\mathcal{Q}(x)$ jako po částech lineární funkci, tedy takovou, jež má body zlomu, ve kterých může nabývat svého optima. Pro spojité rozdělení si můžeme laicky představit, že těchto zlomů je nekonečně mnoho, tedy účelová funkce je hladká v tom smyslu, že je na množině K_2 diferencovatelná. Z toho také plyne, že zatímco v případě spojitého rozdělení bychom se zabývali teorií gradientů, pro diskrétně rozdělené $\boldsymbol{\xi}$ není funkce $\mathcal{Q}(x)$ v bodech zlomů diferencovatelná, tedy bude nutné použít teorii subgradientů, kterou si přiblížíme v následující kapitole.

1.4 Podmínky optimality

Tvrzení 6. *[4] Nechť má úloha (1.2) konečné optimální řešení. Řešení $x^* \in K_1$ je optimálním řešením úlohy (1.2) tehdy a jen tehdy, pokud existují $\lambda^* \in \mathbb{R}^{m_1}$, $\mu^* \in \mathbb{R}^{n_1}_+, \mu^{*T} x^* = 0$ tak, že platí*

$$-c + A^T \lambda^* + \mu^* \in \partial \mathcal{Q}(x^*). \tag{1.9}$$

Důkaz. Z teorie konvexního programování, tedy v případě, že optimalizujeme konvexní funkci přes konvexní množinu, nám nutná a postačující podmínka říká, že účelová funkce $c^T x + \mathcal{Q}(x)$ má v x^* globální minimum na K_1 právě tehdy, když existuje subgradient η v bodě x^*, tak že platí $\eta^T(x - x^*) \geq 0$ pro $\forall\, x \in K_1$. Množinu takových subgradientů $\{\eta | \eta^T(x - x^*) \geq 0 \text{ pro } \forall\, x \in K_1\}$ lze podle [4] vyjádřit ve tvaru $\{\eta | \eta = A^T\lambda + \mu, \text{ pro nějaké } \mu \geq 0,\ \mu^T x^* = 0\}$. Tedy nutnou a postačující podmínkou je, aby průnik množin $\partial(c^T x^* + \mathcal{Q}(x^*)) = c + \partial\mathcal{Q}(x^*)$ a $\{\eta | \eta = A^T\lambda + \mu, \text{ pro nějaké } \mu \geq 0, \mu^T x^* = 0\}$ byl neprázdný. Tím je také dokázáno tvrzení.

□

Poznámka. Pokud by rozdělení bylo absolutně spojité, z tvrzení 5 plyne, že $\mathcal{Q}(x)$ je diferencovatelná, tedy místo $\partial\mathcal{Q}(x^*)$ bychom dosadili $\nabla\mathcal{Q}(x^*)$ a vztah (1.9) nahradili pomocí gradientů:

$$c + \nabla\mathcal{Q}(x^*) = A^T\lambda^* + \mu^*.$$

Z tvrzení 6 plyne, že k nalezení optima je nutný subdiferenciál funkce $\mathcal{Q}(x)$. Následující tvrzení nám říká, jak lze tento subdiferenciál získat pomocí subdiferenciálu jednotlivých funkcí $Q(x,\xi)$, pokud má náhodný vektor $\boldsymbol{\xi}$ konečný nosič.

Tvrzení 7. *[19] Nechť je Ξ konečná množina a funkce $\mathcal{Q}(\cdot) = \mathsf{E}[Q(\cdot,\xi)]$ má konečnou hodnotu alespoň v jednom bodě $\overline{x}$. Potom je funkce $\mathcal{Q}(\cdot)$ polyedrická a pro každé $x_0 \in \operatorname{dom}\mathcal{Q}$ platí*

$$\partial\mathcal{Q}(x_0) = \sum_{k=1}^{K} p_k \partial Q(x_0,\xi_k).$$

Důkaz. Protože $\mathcal{Q}(\overline{x})$ je konečná, automaticky musí také pro všechna k být funkce $Q(\overline{x},\xi_k)$ konečné. Z toho plyne, že $\Pi(q_k,W_k)$ jsou neprázdné, což podle tvrzení 4 znamená, že všechny $Q(\cdot,\xi_k)$ jsou polyedrické. Tedy $\mathcal{Q}$ je také polyedrická, neboť je lineární kombinací polyedrických funkcí s kladnými koeficienty.
Jelikož všechny množiny $\Pi(q_k,W_k)$ jsou neprázdné, pro $x_0 \in \operatorname{dom}\mathcal{Q}$ platí, že $\mathcal{Q}(x_0)$ je konečná. Tedy lze použít Moreau-Rockafellarovu větu P.5, která dokazuje tvrzení, neboť všechny funkce $p_kQ(\cdot,\xi_k)$ jsou polyedrické.

□

Na závěr nám tedy chybí vyjádřit subdiferenciál funkce $Q(x,\xi)$. Zde se využije teorie dvojice duálních úloh související s pojmem sdružené funkce a teorií Fenchelovy duality.

Tvrzení 8. *[19] Nechť pro nějaké $x \in \mathbb{R}^{n_1}$ a $\xi \in \Xi$ je $Q(x,\xi)$ konečná. Potom $Q(\cdot,\xi)$ je subdiferencovatelná v x a platí*

$$\partial Q(x,\xi) = -T^T D(x,\xi),$$

kde

$$D(x,\xi) = \arg\max_{\pi\in\Pi(q,W)} \pi^T(h - Tx).$$

Důkaz. Jelikož $\mathcal{Q}(x,\xi)$ je konečná, z duality plyne, že $\Pi(q,W)$ je neprázdná.
Platí tedy, že s_q je polyedrická a jsou tak splněny předpoklady důsledku P.7 a pro $\chi_0 = h - Tx_0$ platí

$$\partial s_{q,W}(\chi_0) = \arg\max_{\pi}\{\pi^T\chi_0 - s^*_{q,W}(\pi)\},$$

kde s^*_q je sdružená funkce k funkci $s_{q,W}$, viz definice P.8.

Pro funkci

$$f(\pi) = \begin{cases} 0 & \pi \in \Pi(q,W) \\ +\infty & \text{jinak} \end{cases}$$

platí $s_{q,W} = f^*$, neboť z definice sdružené funkce dostaneme

$$f^*(\chi) = \sup_{\pi\in\mathbb{R}^{m_2}} \{\pi^T\chi - f(\pi)\}$$

a funkce $s_q(\chi) = \inf\{q^T y : Wy = \chi, y \geq 0\}$ je rovna

$$s_q(\chi) = \begin{cases} \sup_{\pi \in \Pi(q,W)} \pi^T \chi & \pi \in \Pi(q,W) \\ -\infty & \text{jinak.} \end{cases}$$

Funkce f je vlastní, navíc také konvexní a zdola polospojitá, neboť množina $\Pi(q,W)$ je konvexní a uzavřená. Podle věty P.6 $f = \text{lsc}\, f = f^{**} = s_q^*$. Tedy platí

$$\partial s_q(\chi_0) = \arg\max_{\pi}\{\pi^T \chi_0 - f(\pi)\} = \arg\max_{\pi \in \Pi(q,W)}\{\pi^T \chi_0\}.$$

Z rovnosti $Q(x,\xi) = s_q(h - Tx)$ a použitím řetízkového pravidla dostaneme

$$\partial Q(x_0,\xi) = -T^T \arg\max_{\pi \in \Pi(q,W)}\{\pi^T (h - Tx_0)\}.$$

□

2. Algoritmy

2.1 Bendersova dekompozice

Prvním algoritmem, kterým se budeme zabývat, je Bendersova dekompozice, jež je založena na vnější aproximaci recourse funkce pomocí optimálních řezů. Bendersova dekompozice nese název po svém tvůrci J. F. Bendersovi, jenž v roce 1962 vyřešil mixed-integer úlohu. V té pokaždé nějakou celočíselnou proměnnou x zafixoval a řešil druhou úlohu tak, že x zde nebyla proměnná, nýbrž konstanta [3]. Účelem této techniky je rozdělit velkou složitou úlohu na dvě či více podúloh, které lze řešit snadněji. Bendersova dekompozice tak patří mezi strategie, které jsou v anglické literatuře označovány jako „divide-and-conquer“, tedy rozdělit a zdolat.

Předpokladem pro použití Bendersovy dekompozice je, že matice omezení je tvaru M. Pokud bychom se zabývali duální úlohou, nepracovalo by se s maticí omezení tvaru M, ale její transpozicí M^T.

$$M = \begin{bmatrix} A & & & \\ T_1 & W & & \\ \vdots & & \ddots & \\ T_K & & & W \end{bmatrix} \qquad M^T = \begin{bmatrix} A^T & T_1^T & \dots & T_K^T \\ & W^T & & \\ & & \ddots & \\ & & & W^T \end{bmatrix}$$

Pro duální úlohu by se naopak použila Dantzig-Wolfeho dekompozice, která je postavená na vnitřní linearizaci a je popsána v [5]. My se budeme v této práci zabývat primární úlohou a vnější linearizací recourse funkce. Proto budeme v této kapitole pracovat s úlohami, jejichž extenzivní forma má níže uvedenou blokovou strukturu.

$$\begin{array}{lccccccccc} \min_{x,y_1,\dots,y_K} & c^T x & + & \sum_{k=1}^{K} p_k q_k^T y_k & & & & & & \\ \text{za podmínek} & Ax & & & & & & & = & b \\ & T_1 x & + & W y_1 & & & & & = & h_1 \\ & \vdots & & & & \ddots & & & \vdots & \\ & T_K x & & & & & + & W y_K & = & h_K \\ & x \geq 0 & & y_k \geq 0 & & \forall k & & & & \end{array}$$

Bendersova dekompozice, jak už bylo zmíněno výše, byla původně využita k řešení mixed-integer úlohy. Tato metoda byla rozšířená dále také na stochastické úlohy, kde je nutno hlídat přípustnost řešení, díky autorům Van Slyke a Wets, kteří v roce 1969 přišli s tzv. L-shaped metodou [20]. Ta dostala svůj název na základě struktury matice M, protože pokud zvolíme

$$\overline{T} = \begin{bmatrix} T_1 \\ \vdots \\ T_K \end{bmatrix} \qquad \overline{W} = \begin{bmatrix} W & & \\ & \ddots & \\ & & W \end{bmatrix},$$

bude matice M mít jeden nulový blok a zbylé nenulové bloky vytvoří písmeno L (angl. L-shaped matrix form):

$$M = \begin{bmatrix} A & \mathbf{0} \\ \overline{T} & \overline{W} \end{bmatrix}.$$

2.1.1 L-shaped algoritmus

Předtím, než si uvedeme samotný algoritmus, je vhodné si uvědomit, že řešení úlohy (1.2), tedy

$$\begin{aligned} \min_{x} \quad & \phi = c^T x + \mathcal{Q}(x) \\ \text{za podmínky} \quad & x \in K_1 \cap K_2 \end{aligned}$$

je ekvivalentní s

$$\begin{aligned} \min_{x,\theta} \quad & \phi = c^T x + \theta \\ \text{za podmínek} \quad & \mathcal{Q}(x) \leq \theta \\ & x \in K_1 \cap K_2. \end{aligned} \tag{2.1}$$

Funkci $\mathcal{Q}(x)$ sice neumíme zapsat explicitně, díky tvrzení 4 však víme, že se jedná o polyedrickou funkci. To znamená, že ji na její doméně můžeme zapsat jako maximum konečného množství lineárních funkcí a můžeme tak uvažovat vnější linearizaci pomocí opěrných funkcí. Algoritmus umí postupně generovat opěrné funkce, nazývané jako tzv. řezy optimality, za předpokladu, že bod získaný optimalizací hlavní úlohy (2.2)-(2.4) bude přípustný také pro druhý stupeň stochastické úlohy.

Druhý stupeň stochastické úlohy algoritmus rozkládá do K subproblémů, kde každý subproblém přísluší jedné realizaci ξ_k náhodného vektoru $\boldsymbol{\xi}$. Pokud je alespoň pro jednu realizaci subproblém nepřípustný, algoritmus vygenerovaný bod odřízne a vygeneruje novou podmínku, tzv. řez přípustnosti. Využívá vlastnosti z tvrzení 1, že množina K_2, tedy doména funkce $\mathcal{Q}$, je polyedrická. Díky této vlastnosti lze množinu K_2 zapsat pomocí systému nerovností, kterými jsou postupně vytvořené řezy přípustnosti. Poznamenejme, že pokud se jedná o problém s relativně úplnou kompenzací, plyne automaticky, že v každém iteračním kroku je optimální řešení (x^ν,θ^ν) přípustným bodem, tedy $x^\nu \in K_2$, díky čemuž je možné druhý krok algoritmu přeskočit úplně.

Nyní přejdeme k samotnému algoritmu, jehož zápis je převzat z [4]. Hlavní problém algoritmu bude odpovídat úloze (2.1), kde místo podmínky $\mathcal{Q}(x) \leq \theta$, budeme mít podmínky s vygenerovanými řezy optimality.

Krok 0: Nechť $r = s = \nu = 0$, kde r a s představují počet řezů přípustnosti a řezů optimality.

Krok 1: Nechť $\nu = \nu + 1$, tedy zvyš pořadí iterace o 1 a vyřeš hlavní úlohu:

$$\begin{aligned} \min_{x,\theta} \quad & \phi = c^T x + \theta && (2.2) \\ \text{za podmínek} \quad & Ax = b \\ & D_l x \geq d_l, \quad l = 1,\ldots,r && (2.3) \\ & E_l x + \theta \geq e_l, \quad l = 1,\ldots,s && (2.4) \\ & x \geq 0, \theta \in \mathbb{R} \end{aligned}$$

Označ (x^ν, θ^ν) optimální řešení výše uvedené úlohy. Pokud úloha neobsahuje žádnou podmínku typu (2.4), potom polož $\theta^\nu = -\infty$ a pro výpočet optimální hodnoty x^ν ji neuvažuj.

Krok 2: Zjisti, zda $x^\nu \in K_2$, tedy pro $k = 1, \ldots, K$ vyřeš úlohu lineárního programování:

$$\begin{aligned} \min_{v^+, v^-} \quad & w'_k = e^T v^+ + e^T v^- \\ \text{za podmínek} \quad & Wy + Iv^+ - Iv^- = h_k - T_k x^\nu \mid \sigma^\nu \\ & y \geq 0,\, v^+ \geq 0,\, v^- \geq 0, \end{aligned} \tag{2.5}$$

kde $e^T = (1, \ldots, 1)$, dokud pro nějaké k není optimální hodnota $w'_k > 0$. V tomto případě nechť σ^ν je příslušný duální multiplikátor a definuj

$$D_{r+1} = (\sigma^\nu)^T T_k$$

a

$$d_{r+1} = (\sigma^\nu)^T h_k,$$

čímž se vytvoří nový řez přípustnosti (angl. feasibility cut), tedy nové omezení typu (2.3). Dále polož $r = r + 1$, přidej toto omezení do množiny podmínek (2.3) a vrať se na *Krok 1.* Pokud pro všechna k je $w'_k = 0$, přejdi na *Krok 3.*
Krok 3: Pro $k = 1, \ldots, K$ vyřeš úlohu lineárního programování:

$$\begin{aligned} \min_{y_k} \quad & w_k = q_k^T y_k \\ \text{za podmínek} \quad & W y_k = h_k - T_k x^\nu \mid \pi_k^\nu \\ & y_k \geq 0. \end{aligned} \tag{2.6}$$

Nechť π_k^ν je příslušný duální multiplikátor problému k typu (2.6). Definuj

$$E_{s+1} = \sum_{k=1}^{K} p_k (\pi_k^\nu)^T T_k$$

a

$$e_{s+1} = \sum_{k=1}^{K} p_k (\pi_k^\nu)^T h_k.$$

Nechť $w^\nu = e_{s+1} - E_{s+1} x^\nu$. Pokud $\theta^\nu \geq w^\nu$, ukonči algoritmus, x^ν je optimální řešení. Jinak polož $s = s + 1$ a přidej další omezení do množiny podmínek (2.4) a vrať se na *Krok 1.*

Nyní si podrobně rozebereme kroky výše popsaného algoritmu, v němž je postupně v každé iteraci přidáno jedno z možných typů omezení:

- řezy přípustnosti určující množinu $\{\boldsymbol{x} | \mathcal{Q}(\boldsymbol{x}) < \infty\}$, podmínka typu (2.3),
- optimální řezy, které lineárně aproximují $\mathcal{Q}$ na její doméně, podmínka typu (2.4).

Algoritmus v každém iteračním kroku nejprve najde optimální řešení hlavního problému (x^ν, θ^ν) a zjistí, jestli je přípustné i v druhém stupni. Pokud není, přidá v druhém kroku do hlavní úlohy nový řez přípustnosti. Pokud je přípustné, vyřeší K subproblémů, pro každý subproblém získá řešení y_k^ν a poté pomocí ukončovacího kritéria zjistí, zda je bod $(x^\nu, y_1^\nu, \ldots, y_K^\nu)$ bodem optima. Pokud je, ukončí algoritmus, v opačném případě přidá do hlavní úlohy nový řez optimality.

Řezy přípustnosti

Nyní si ukážeme, jak lze pomocí řezů přípustnosti vytvořit množinu přípustných řešení ve druhém stupni, tedy takovou množinu, aby platilo $x \in K_2$.

Definice $x \in K_2$ je ekvivalentní se zápisem

$$x \in \{x | \text{ pro } \forall\, k = 1, \ldots, K, \exists\, y_k \geq 0 \text{ splňující } W y_k = h_k - T_k x\}.$$

To také znamená, že

$$h_k - T_k x \in \text{pos } W \text{ pro } k = 1, \ldots, K.$$

V druhém kroku L-shaped algoritmu je řešena úloha (2.5), k ní duální úloha je tvaru

$$\begin{aligned} \max_{\sigma} \quad & \sigma^T (h_k - T_k x) \\ \text{za podmínek} \quad & \sigma^T W \leq 0 \\ & \sigma \leq e \\ & -\sigma \leq e. \end{aligned} \tag{2.7}$$

Úloha (2.5) slouží k testování, zda $h_k - T_k x \in \text{pos } W$ pro $\forall\, k = 1, \ldots, K$. Pokud tomu tak není, pak pro nějaké $k = 1, \ldots, K$ platí, že $h_k - T_k x^\nu \notin \text{pos } W$. Jelikož pos W je konvexní uzavřená množina, podle věty 3 musí existovat nadrovina, pomocí níž lze od ní bod $h_k - T_k x^\nu$ ostře oddělit.

Nejprve si vysvětlíme, proč tato nadrovina musí splňovat $\sigma^T \chi \leq 0$ pro všechna $\chi \in \text{pos } W$ a zároveň $\sigma^T (h_k - T_k x^\nu) > 0$. Dále si ukážeme, že tyto vlastnosti vektoru nadroviny σ splňuje duální multiplikátor σ^ν úlohy (2.5).Vše vychází ze známé Farkasovy věty, kterou si zde pro úplnost uvedeme.

Tvrzení 9 (Farkasova věta)**.** *[6] Soustava $Wy = \chi$ má nezáporné řešení tehdy a jen tehdy, když pro každé σ, které splňuje $W^T \sigma \geq 0$, platí $\chi^T \sigma \geq 0$.*

Změnou znaménka σ můžeme podmínku přepsat na $W^T \sigma \leq 0 \Rightarrow \chi^T \sigma \leq 0$. Množinu všech σ splňujících $W^T \sigma \leq 0$ můžeme přepsat do tvaru polárního kužele následujícím způsobem:

$$\{\sigma | W^T \sigma \leq 0\} = \{\sigma | \sigma^T W y \leq 0 \text{ pro } \forall\, y \geq 0\} = \{\sigma | \sigma^T \chi \leq 0 \text{ pro } \forall\, \chi \in \text{pos } W\}.$$

Předpokládejme, že v ν-tém iteračním kroku nalezne algoritmus nepřípustný bod x^ν. Mějme takové σ splňující $\sigma^T \chi \leq 0$ pro $\forall\, \chi \in \text{pos } W$, což je ekvivalentní s $\sigma^T W \leq 0$. Pro nepřípustný bod x^ν musí pro nějaké k platit nerovnost $\sigma^T (h_k - T_k x^\nu) > 0$, neboť tento bod nesplňuje podmínku z Farkasovy věty. Přidáním podmínky

$$\sigma^T (h_k - T_k x) \leq 0 \tag{2.8}$$

tak oddělíme bod x^ν a zároveň zajistíme, že všechna přípustná řešení úlohy budou i s tímto novým omezením zachována.

Výše požadované vlastnosti na σ splňuje duální multiplikátor σ^ν úlohy (2.5). Z duality totiž plyne, že pokud pro nějaké $k = 1, \ldots, K$ je w'_k kladné, z (2.7) dostáváme $(\sigma^\nu)^T (h_k - T_k x) > 0$. Navíc je také splněna nerovnost $(\sigma^\nu)^T W \leq 0$, neboť σ^ν je optimální duální multiplikátor, tedy musí být i přípustným řešením

duální úlohy. Z (2.7) vidíme, že optimální duální multiplikátor musí splňovat $(\sigma^\nu)^T W \leq 0$ a $-e \leq \sigma^\nu \leq e$, kde druhý výraz vyjadřuje pouze normalizaci.

Proto nutnou podmínkou pro $x \in K_2$ je nerovnost $(\sigma^\nu)^T(h_k - T_k x) \leq 0$. Těchto podmínek typu (2.3) je navíc konečně mnoho, neboť je pouze konečně mnoho optimálních bází úlohy (2.5).

Na závěr si ukážeme souvislost s množinou krajních směrů duální úlohy (1.8). Pokud je množina přípustných řešení $\Pi(q,W)$, definovaná nerovnostmi $W^T\pi \leq q$, neprázdná a navíc také omezená, potom ji můžeme vyjádřit pomocí množiny krajních bodů $\{\pi_1, \ldots \pi_s\}$ a směrů $\{\sigma_1, \ldots, \sigma_r\}$. Lze totiž ukázat, že každá konvexní polyedrická množina, jež je neprázdná a neobsahuje přímku, lze zapsat jako součet konvexního obalu množiny všech krajních bodů a nezáporného obalu množiny všech krajních směrů množiny $\Pi(q,W)$ [6]. Tedy každý bod π množiny $\Pi(q,W)$ lze zapsat následujícím způsobem:

$$\begin{aligned} \pi &= \sum_{l=1}^{s} \lambda_l \pi_l + \sum_{l=1}^{r} \mu_l \sigma_l \\ \sum_{l=1}^{s} \lambda_l &= 1 \\ \lambda_l &\geq 0, \quad l = 1, \ldots, s \\ \mu_l &\geq 0, \quad l = 1, \ldots, r. \end{aligned}$$

Z duality plyne, že pokud je duální úloha (1.8) neomezená, potom primární úloha (1.7) nemá přípustné řešení. Nyní předpokládejme, že existuje nějaký krajní směr σ_l, pro který platí

$$(\sigma_l)^T(h_k - T_k x) > 0. \tag{2.9}$$

Jelikož pro $\forall\, \pi \in \Pi$ a $\forall\, \mu > 0$ platí $\pi + \mu\sigma_l \in \Pi$, účelová funkce duální úlohy pro k-tý subproblém (2.6) v případě, že platí podmínka (2.9), roste do plus nekonečna, neboť

$$\left(\underbrace{\pi^T(h_k - T_k x)}_{c \in \mathbb{R}} + \underbrace{\mu}_{\to +\infty} \underbrace{(\sigma_l)^T(h_k - T_k x)}_{>0} \right).$$

Účelem je tedy zamezit tomu, aby duální úloha v následném třetím kroku byla neomezená a získat tak řešení, které je přípustné pro primární úlohu. Toho docílíme tím, že hlavní úlohu algoritmu podmíníme dodatečnými podmínkami (řezy přípustnosti) ve tvaru (2.8)

$$(\sigma_l)^T(h_k - T_k x) \leq 0, \quad l = 1, \ldots, r,$$

kde σ_l představují krajní směry množiny $\Pi(q,W)$.

Optimální řezy

Ve třetím kroku L-shaped algoritmu je pro přípustný bod x^ν získaný z 1. kroku řešena pro každé $k = 1, \ldots, K$ úloha (2.6), čímž dostaneme optimální duální multiplikátory π_k^ν. Z duality plyne, že pro každé $k = 1, \ldots, K$ je optimální řešení primární úlohy rovno optimálnímu řešení duální úlohy, tedy

$$Q(x^\nu, \xi_k) = (\pi_k^\nu)^T(h_k - T_k x^\nu). \tag{2.10}$$

Navíc díky tomu, že je $Q(x, \xi_k)$ konvexní funkce, platí pro ni nerovnost

$$Q(x, \xi_k) \geq (\pi_k^\nu)^T(h_k) - (\pi_k^\nu)^T T_k x, \tag{2.11}$$

neboť z vlastnosti subdiferenciálu z definice P.3 dostaneme

$$Q(x) \geq Q(x^\nu) + [\partial Q(x^\nu)]^T(x - x^\nu)$$

a díky tvrzení 8 víme, že pro každé $k = 1, \ldots, K$ platí

$$\partial Q(x^\nu, \xi_k) = -(\pi_k^\nu)^T T_k.$$

Následným dosazením získáme požadovanou nerovnost (2.11)

$$Q(x,\xi_k) \geq (\pi_k^\nu)^T(h_k - T_k x^\nu) - (\pi_k^\nu)^T T_k(x - x^\nu) = (\pi_k^\nu)^T h_k - (\pi_k^\nu)^T T_k x.$$

Aplikací střední hodnoty na (2.10) poté získáme následující vztah

$$\mathcal{Q}(x^\nu) = \mathsf{E}\left[(\boldsymbol{\pi}^\nu)^T(\boldsymbol{h} - \boldsymbol{T}x^\nu)\right] = \sum_{k=1}^{K} p_k(\pi_k^\nu)^T(h_k - T_k x^\nu)$$

a aplikací na (2.11) dostaneme

$$\mathcal{Q}(x) \geq \mathsf{E}\left[(\boldsymbol{\pi}^\nu)^T(\boldsymbol{h} - \boldsymbol{T}x)\right] = \sum_{k=1}^{K} p_k(\pi_k^\nu)^T h_k - \sum_{k=1}^{K} p_k(\pi_k^\nu)^T T_k x. \tag{2.12}$$

Z výše uvedené nerovnosti plyne, že optimální řezy lineárně aproximují funkci $\mathcal{Q}(x)$ na její doméně. Pravá strana (2.12), jež odpovídá řezům optimality (2.4), je dolním odhadem funkce $\mathcal{Q}(x)$. Navíc v bodě x^ν mají stejnou funkční hodnotu, tedy se jedná o opěrnou funkci.

Dále vidíme, že funkce $\mathcal{Q}(x)$ je omezená shora z úlohy (2.1) nerovností $\mathcal{Q}(x) \leq \theta$ a zdola podmínkou (2.12). Pokud se toto horní a dolní omezení sobě rovná, nachází se algoritmus v optimu. A tedy, optimum je nalezeno právě tehdy, když $\theta^\nu = \mathsf{E}\,(\boldsymbol{\pi}^\nu)^T(\boldsymbol{h} - \boldsymbol{T}x^\nu)$, což odpovídá kritériu k ukončení algoritmu v kroku 3.

To znamená, že v každém iteračním kroku je buď podmínka $\theta^\nu \geq \mathcal{Q}(x^\nu)$ splněna, tedy je nalezeno optimální řešení, nebo naopak naopak $\theta^\nu < \mathcal{Q}(x^\nu)$. Poté je třeba hlavní úlohu řešenou v kroku 1 dodatečně omezit, tedy pomocí duálních multiplikátorů π_k^ν příslušících k x^ν přidat novou podmínku (řez) typu (2.4). Konkrétně, pro k-tý subproblém (2.6) platí, že optimální hodnota účelové funkce je $\pi_k^\nu(h_k - T_k x^\nu) = q_k(L)W_L^{-1}(h_k - T_k x^\nu)$, kde L značí podle definice P.1 optimální bázi. V každé iteraci je tedy $k-$tý multiplikátor určen optimální bází $k-$tého subproblému ($\pi_k^\nu = q_k(L)W_L^{-1}$), kterých je konečně mnoho. Algoritmus tudíž vytváří pouze konečně mnoho řezů optimality, neboť je pouze konečně mnoho kombinací K multiplikátorů, kdy každý může nabývat pouze konečně mnoha hodnot.

L-shaped algoritmus tak vytváří pouze konečně mnoho řezů přípustnosti a konečně mnoho řezů optimality, tedy konverguje v konečném čase. Pokud optimální řešení existuje, tak jej také nalezne.

2.2 Stochastická dekompozice

V této kapitole se budeme zabývat algoritmem stochastické dekompozice. Jak již název napovídá, tento algoritmus bude využívat dekompozičních vlastností a stochastických aproximací. Dekompozicí jsme se zabývali v předchozí kapitole při popisu L-shaped algoritmu, který využívá Bendersovu dekompozici na úlohy stochastického programování. Připomeňme, že základní forma L-shaped algoritmu staví mimo jiné na tom, že náhodná veličina $\boldsymbol{\xi}$ je diskrétní nabývající K hodnot a v případě, že je spojitá, tak ji lze zdiskretizovat. V každé iteraci musí algoritmus řešit všech K subproblémů, jenž přísluší daným scénářům. Při velkém množství scénářů pak může být řešení problému časově náročné.

Naopak, stochastické aproximace, mezi než se řadí stochastická kvazi-gradient metoda (SQG), pracují s diskrétními i spojitými náhodnými veličinami stejným způsobem. V každé iteraci používají pouze jeden prvek náhodného výběru, který může být generován jak z diskrétního tak spojitého rozdělení. Jejich nevýhodou je, že neposkytují v průběhu algoritmu odhad hodnoty účelové funkce a navíc je obtížné správně nastavit délku kroku a implementovat ukončovací kritéria.

Stochastická dekompozice tyto dvě metody propojuje. Umí využít speciální struktury úlohy tak, jako to dělá L-shaped algoritmus, a tedy iterativně aproximuje účelovou funkci pomocí lineárních řezů. Zároveň používá ideu stochastických aproximací, která v každé iteraci počítá pouze s jednou realizací náhodného výběru, nikoliv se všemi scénáři.

Za vznikem této metody stojí autoři Higle a Sen, kteří v roce 1991 vydali článek, v němž představili koncept a dokázali důležitá teoretická tvrzení. Ukázali v něm také konvergenci vygenerovaných řešení v podposloupnosti k optimálnímu řešení úlohy skoro jistě [7]. Tito autoři o pět let později vydali také knihu specializující se pouze na stochastickou dekompozici, ve které navíc podrobně uvádějí výhody regularizovaného hlavního problému a zabývají se hlouběji ukončovacími kritérii [8]. Metoda stochastické dekompozice je stručně a přehledně shrnuta v [12].

Nejprve si uvedeme formální zápis dvoustupňové stochastické úlohy, se kterou budeme v této kapitole pracovat:

$$\begin{aligned} \min_{x} \quad & \phi(x) \equiv c^T x + \mathcal{Q}(x) \\ \text{za podmínek} \quad & Ax = b \\ & x \geq 0. \end{aligned} \tag{2.13}$$

Dále

$$\mathcal{Q}(x) = \int Q(x,\xi) f(\xi) d\xi,$$

kde f vyjadřuje hustotu $\boldsymbol{\xi}$ a

$$\begin{aligned} Q(x,\xi) &= \min_{y}\{q^T y | Wy = h(\xi) - T(\xi)x, y \geq 0\} \\ &= \max_{\pi}\{\pi^T (h(\xi) - T(\xi)x) | W^T \pi \leq q\}. \end{aligned}$$

Druhá rovnost nastane za předpokladu, že množina duálně přípustných řešení Π je neprázdná, což v této kapitole budeme předpokládat.

Funkce $\mathcal{Q}(x) = \mathsf{E}\,[Q(x,\boldsymbol{\xi})]$ je aproximována na základě náhodného výběru, který je vygenerován do $\nu-$té iterace $(\xi_1,\ldots,\xi_\nu)$ pomocí průměru účelových funkcí $Q(x,\xi_k)$

$$\mathcal{Q}_\nu(x) = \frac{1}{\nu}\sum_{k=1}^{\nu} Q(x,\xi_k).$$

Jak si uvedeme posléze, samotná funkce $\mathcal{Q}_\nu(x)$ je aproximována pomocí postupně vytvořených řezů do $\nu-$té iterace. Stejně jako v L-shaped algoritmu lze pak hlavní úlohu algoritmu (M^ν), která je počítána v $\nu-$té iteraci, zapsat s využitím pomocné proměnné θ.

(M^ν) :

$$\begin{aligned} \min_{x,\theta} \quad & \phi_\nu(x) \equiv c^T x + \theta \\ \text{za podmínek} \quad & Ax = b \\ & x \geq 0 \\ & \theta \geq e_k^\nu - E_k^\nu x, \quad k = 1,\ldots,\nu. \end{aligned} \tag{2.14}$$

Pro přidání řezu se v ν-té iteraci základního algoritmu pro x^ν nalezené v hlavním problému v předchozí iteraci musí vyřešit subproblém pro nové ξ_ν.

(S^ν) :

$$\begin{aligned} \min_{y} \quad & q^T y \\ \text{za podmínek} \quad & Wy = h(\xi_\nu) - T(\xi_\nu)x \\ & y \geq 0. \end{aligned} \tag{2.15}$$

V případě rozšířeného algoritmu je potřeba tento subproblém vyřešit také pro kandidáta řešení (angl. incumbent solution) $\overline{x}^{\nu-1}$, což vysvětlíme později.

Připomeňme, že L-shaped algoritmus počítá v každé iteraci všech K subproblémů (kde každý přísluší jednomu scénáři), pomocí nichž postupně vytváří řezy, kterými aproximuje funkci $\mathcal{Q}(x)$. Naopak, algoritmus stochastické dekompozice generuje realizace náhodné veličiny $\boldsymbol{\xi}$ postupně. V každé iteraci počítá pouze jeden subproblém (v případě rozšířeného algoritmu dva subproblémy) tvaru (2.15), pomocí něhož přidá nový řez do hlavní úlohy (2.14). Na základě tohoto náhodného výběru aproximuje funkci $\mathcal{Q}_\nu(x)$, která je odhadem $\mathcal{Q}(x)$.

Předpoklady

Aby bylo možné použít stochastickou dekompozici, musí úloha splňovat následující předpoklady [7]:

A1: K_1 a Ξ jsou neprázdné kompaktní množiny.

A2: Π je neprázdná kompaktní konvexní polyedrická množina.

A3: Pro $\forall\, x \in X$ platí $Q(x,\boldsymbol{\xi}) \geq 0$ skoro jistě.

Množiny K_1, Ξ a Π jsou definovány stejně jako v předchozí části. Připomeňme, že K_1 je množina přípustných řešení v prvním stupni úlohy, Ξ je nosičem náhodného vektoru $\boldsymbol{\xi}$ a Π je množina duálně přípustných řešení.

Předpoklady A1-A3 pak implikují, že $\exists\, C < \infty$ takové, že

$$0 \leq Q(x,\xi) = \max_{\pi}\{\pi^T(h(\xi) - T(\xi)x) | W^T\pi \leq q\} < C$$

pro všechna $(x,\xi) \in K_1 \times \Xi$. Tedy primární úloha má úplnou kompenzaci a algoritmus v hlavní úloze (2.14) vytváří pouze řezy optimality, neboť řezy přípustnosti při těchto předpokladech nejsou potřeba.

Dále se předpokládá fixní vektor $q(\xi) \equiv q$ pro $\forall\, \xi \in \Xi$. Teorii, jež toto poslední omezení nevyžaduje, lze nalézt v [9].

2.2.1 Algoritmus stochastické dekompozice (SD)

Nyní si uvedeme algoritmus SD, jehož kroky si následně rozebereme podrobně. Algoritmus popsaný níže je převzat z knihy [8]. Vychází ze základního algoritmu stochastické dekompozice [7] a využívá podposloupnosti kandidátů řešení. Je uveden bez ukončovacích kritérií, která jsou diskutována v závěru kapitoly.
Krok 0: Nechť $\nu = 0$, $V_0 = \emptyset$, $\xi_0 = \mathsf{E}\,[\boldsymbol{\xi}]$,

$$x^1 = \arg\min_x\{c^Tx + Q(x,\xi_0) | x \in K_1\},$$

$\overline{x}^0 = x^1$, $i_0 = 0$ a zvol fixní $r \in (0,1)$.
Krok 1: Zvyš iteraci o jedna, $\nu = \nu + 1$. Vygeneruj nové ξ_ν (ξ_k, $k = 1,\ldots,\nu$ jsou generovány náhodně).
Krok 2: Vyřeš dva subproblémy (S^ν), jeden pro x^ν a druhý pro kandidáta řešení $\overline{x}^{\nu-1}$. Přidej příslušné duální multiplikátory do množiny

$$V_\nu = V_{\nu-1} \cup \{\pi(x^\nu,\xi_\nu), \pi(\overline{x}^{\nu-1},\xi_\nu)\}, \tag{2.16}$$

kde $\pi(x,\xi) \in \arg\max_\pi\{\pi^T(h(\xi) - T(\xi)x) | W^T\pi \leq q\}$.
Krok 3: Definuj $\phi_\nu(x)$, novou aproximaci funkce $\phi(x) = c^Tx + \mathcal{Q}(x)$.
a) Sestroj koeficienty nového řezu vytvořeného v $\nu-$té iteraci, který bude přidán do hlavního problému (M^ν) na základě maximalizační funkce přes množinu duálních multiplikátorů V_ν vytvořenou v předchozím kroku (2.16).

$$E^\nu_\nu = \frac{1}{\nu}\sum_{k=1}^{\nu}(\pi^\nu_k)^T T(\xi_k), \quad e^\nu_\nu = \frac{1}{\nu}\sum_{k=1}^{\nu}(\pi^\nu_k)^T h(\xi_k),$$

kde $\pi^\nu_k \in \arg\max_\pi\{\pi^T(h(\xi_k) - T(\xi_k)x^\nu) | \pi \in V_\nu\}$.
b) Opět na základě množiny duálních multiplikátorů V_ν sestroj koeficienty nového řezu indexovaného $i_{\nu-1}$, tedy koeficienty řezu pro kandidáta řešení, který bude přidán do hlavního problému (M^ν).

$$E^\nu_{k_{\nu-1}} = \frac{1}{\nu}\sum_{k=1}^{\nu}(\overline{\pi}_k)^T T(\xi_k), \quad e^\nu_{k_{\nu-1}} = \frac{1}{\nu}\sum_{k=1}^{\nu}(\overline{\pi}_k)^T h(\xi_k),$$

kde $\overline{\pi}_k \in \arg\max_\pi\{\pi^T(h(\xi_k) - T(\xi_k)\overline{x}^{\nu-1}) | \pi \in V_\nu\}$.
c) Obnov předchozí řezy tak, aby splňovaly, že jsou dolní aproximací funkce $\mathcal{Q}_\nu$.

$$E^\nu_k = \frac{\nu-1}{\nu}E^{\nu-1}_k, \quad e^\nu_k = \frac{\nu-1}{\nu}e^{\nu-1}_k, \quad k \notin \{i_{\nu-1},\nu\}.$$

Krok 4: Otestuj, zda je x^ν novým kandidátem řešení. Pokud

$$\phi_\nu(x^\nu) - \phi_\nu(\overline{x}^{\nu-1}) < r[\phi_{\nu-1}(x^\nu) - \phi_{\nu-1}(\overline{x}^{\nu-1})],$$

potom $\overline{x}^\nu = x^\nu$, $i_\nu = \nu$. Jinak $\overline{x}^\nu = \overline{x}^{\nu-1}$, $i_\nu = i_{\nu-1}$.
Krok 5: Vyřeš hlavní problém (M^ν) a získej $x^{\nu+1}$. Vrať se na *Krok 1.*

Generování nových řezů

Jednou z hlavních myšlenek stochastické dekompozice je fakt, že pro všechny $\pi \in \Pi$ a pro $\forall\, x \in K_1$ a $\forall\, \xi \in \Xi$ platí

$$Q(x,\xi) \geq \pi^T[h(\xi) - T(\xi)x].$$

Algoritmus nejprve sestrojí v bodě x^ν opěrnou nadrovinu k funkci $Q(x,\xi_\nu) = \max_\pi\{\pi^T(h(\xi_\nu) - T(\xi_\nu)x) | W^T\pi \leq q\}$ pomocí optimálního duálního multiplikátoru π^ν_ν, který vloží do množiny V_ν. Množina V_ν značí množinu všech optimálních duálních multiplikátorů získaných do iterace ν.

Zavedení množiny V_ν vede k následnému zrychlení výpočtu, neboť algoritmus odhaduje v bodě x^ν funkce $Q(x,\xi_k)$ pro $k = 1,\ldots,\nu-1$ pomocí

$$\max_\pi\{\pi^T(h(\xi_k) - T(\xi_k)x^\nu) | \pi \in V_\nu\},$$

čímž se pro každé $k = 1,\ldots,\nu$ získá multiplikátor π^ν_k. Jelikož platí $V_\nu \subset \Pi(q,W)$, jsou tyto odhady dolními odhady funkcí $Q(x,\xi_k)$ pro $k = 1,\ldots,\nu-1$.

Pomocí takto získaných duálních multiplikátorů $\{\pi^\nu_k\}^\nu_{k=1}$ se sestrojí nový řez. Funkce $\mathcal{Q}_\nu(x)$, jež je samotná odhadem funkce $\mathcal{Q}(x)$, je pak aproximována zdola

$$\mathcal{Q}_\nu(x) = \frac{1}{\nu}\sum_{k=1}^{\nu} Q(x,\xi_k) \geq \frac{1}{\nu}\sum_{k=1}^{\nu}(\pi^\nu_k)^T[h(\xi_k) - T(\xi_k)x].$$

Obnovení starých řezů

Zde využijeme předpokladu A3, tj. $Q(x,\boldsymbol{\xi}) \geq 0$ skoro jistě. Z toho získáme

$$\begin{aligned}\mathcal{Q}_\nu(x) &= \frac{1}{\nu}\sum_{k=1}^{\nu} Q(x,\xi_k) = \frac{\nu-1}{\nu}\left\{\frac{1}{\nu-1}\sum_{k=1}^{\nu-1} Q(x,\xi_k)\right\} + \frac{1}{\nu}Q(x,\xi_k)\\ &\geq \frac{\nu-1}{\nu}\left\{\frac{1}{\nu-1}\sum_{k=1}^{\nu-1} Q(x,\xi_k)\right\} = \frac{\nu-1}{\nu}\mathcal{Q}_{\nu-1}(x).\end{aligned}$$

Tedy, aby také předchozí řezy byly vhodnými dolními odhady funkce $\mathcal{Q}_\nu(x)$, je třeba je vynásobit výrazem $\frac{\nu-1}{\nu}$. Poznamenejme, že pokud by dolní mez funkce $Q(x,\boldsymbol{\xi})$ byla $L \neq 0$, je nutné algoritmus upravit, což budeme diskutovat v praktické části.

Kandidáti řešení

V článku [7], ve kterém byl poprvé představen algoritmus stochastické dekompozice, jsou popsány teoretické vlastnosti společně s jejich důkazy. Tyto důkazy

lze ve čtenářsky přívětivější formě nalézt také v [8]. V této práci si uvedeme jen nejdůležitější závěry.

Za výše zmíněných předpokladů platí, že stochastická dekompozice generuje v hlavní úloze takovou posloupnost bodů, že existuje její podposloupnost x^{ν_j} taková, že všechny hromadné body této podposloupnosti řeší problém (2.13) skoro jistě [7].

Ve většině deterministických algoritmů hodnota účelové funkce v nalezených bodech postupně klesá. Toto stochastické algoritmy nemohou zajistit, v každé iteraci totiž dochází ke změně odhadu účelové funkce, který je ovlivněn novým pozorováním. Autoři za účelem stabilizace řešení zavedli posloupnost tzv. kandidátů řešení $\overline{x}^k$, která tvoří podposloupnost nalezených řešení. V případě, že je odhadovaná hodnota účelové funkce v nalezeném řešení „dostatečně malá“, stává se toto řešení novým kandidátem řešení. Zároveň zavedli posloupnost indexů i_k vyjadřujících pořadí iterace, ve které byl nalezen kandidát řešení $\overline{x}^k$. V [8] je popsáno, že ač dochází ke změně kandidáta řešení konečně či nekonečně mnohokrát, lze nalézt podposloupnost kandidátů řešení, jejíž hromadné body řeší problém (2.13) skoro jistě. Nyní si ukážeme, jak lze algoritmus rozšířit, aby jednoduše detekoval tuto podposloupnost kandidátů řešení.

Aby bylo možné body x^ν a $\overline{x}^{\nu-1}$ vhodně porovnat, je nutné nalézt nový optimální multiplikátor také v posledním kandidátovi řešení $\overline{x}^{\nu-1}$. V druhém kroku algoritmu SD proto dochází k rozšíření množiny duálních multiplikátorů V_ν nejen o duální multiplikátor v nově nalezeném bodě x^ν, ale také v kandidátovi řešení $\overline{x}^{\nu-1}$. Pomocí rozšířené množiny V_ν se pak vytvoří aproximující řezy aktuálního odhadu účelové funkce.

K testování, zda je odhadovaná hodnota účelové funkce v nalezeném řešení „dostatečně menší“ než v bodě posledního kandidáta řešení, tedy zda by mělo být řešení x^ν novým kandidátem řešením, slouží nerovnost čtvrtého kroku algoritmu

$$\phi_\nu(x^\nu) - \phi_\nu(\overline{x}^{\nu-1}) < r[\phi_{\nu-1}(x^\nu) - \phi_{\nu-1}(\overline{x}^{\nu-1})].$$

Rozdíl na pravé straně vyjadřuje aproximaci toho, co bychom získali, jestliže by se řešení x^ν stalo novým kandidátem řešení. Funkce $\phi_{\nu-1}(x)$, která je odhadem funkce $\phi(x)$ v iteraci $(\nu-1)$, ji však nemusí vhodně aproximovat v bodě x^ν. To je nalezeno až jako její minimum, z čehož navíc plyne $\phi_{\nu-1}(x^\nu) - \phi_{\nu-1}(\overline{x}^{\nu-1}) \leq 0$. Na druhou stranu, funkce ϕ_ν byla aproximována řezy, které byly vytvořené na okolí bodu x^ν, v tomto bodě by proto měla odhadovat funkci $\phi(x)$ správně. Zároveň odhad $\phi_\nu(\overline{x}^{\nu-1})$ by měl být přibližně stejně dobrým odhadem $\phi(\overline{x}^{\nu-1})$ jako odhad $\phi_{\nu-1}(\overline{x}^{\nu-1})$.

Levá strana tak vyjadřuje reálný rozdíl a pravá strana $(r * 100)\%$ rozdíl, který očekáváme, přičemž r označuje míru, kterou požadujeme. Pokud je r blízko nule, potom je téměř každé x^ν novým kandidátem řešení. Pokud je naopak blízké jedné, dochází ke změně pouze zřídka. V této práci budeme v praktické části volit pokaždé $r = 0.5$.

Regularizovaný hlavní problém

V této části si uvedeme modifikaci hlavní úlohy algoritmu SD, která vede ke stabilizaci nalezených řešení a snížení výpočetní složitosti hlavního problému. Připomeňme, že v každé iteraci je přidán nový řez a všechny předchozí jsou

obnoveny. Stejně jako v deterministickém L-shaped algoritmu s každou iterací roste počet aproximujících řezů o jedna. L-shaped algoritmus však od začátku pracuje se všemi scénáři, tedy k nalezení optima potřebuje menší počet iterací. Počet řezů vytvořených SD algoritmem bývá podstatně větší než počet, který je vygenerovaný L-shaped algoritmem. Některé tyto řezy jsou plně nahraditelné ostatními, tedy jejich zachování zbytečně navyšuje časovou náročnost výpočtu hlavní úlohy. Z toho důvodu je vhodné uvažovat nad jejich eliminací. K odstranění řezů však musí docházet sofistikovaně, aby se předešlo situaci, kdy je smazán řez, který ve skutečnosti zbytečný není. Potom by tvrzení, která se o existenci těchto řezů opírají, neplatila. Nemohli bychom např. tvrdit, že vygenerovaná posloupnost bodů má hromadné body, které jsou optimálními skoro jistě.

Díky posloupnosti kandidátů řešení lze hlavní problém v 5. kroku SD algoritmu nahradit tzv. regularizovaným hlavním problémem, který umožňuje omezit shora počet řezů. Lze ukázat, že pokud $x \in \mathbb{R}^{n_1}$, potom regularizovaný hlavní problém obsahuje maximálně $n_1 + 3$ řezů [8]. Množinu indexů řezů obsažených v $\nu-$té iteraci označme J_ν. Platí, že $J_\nu \subset \{1, \ldots, \nu\}$.

Regularizovaný hlavní problém bude mít v účelové funkci kromě funkce

$$\phi_\nu(x) = c^T x + \max_x \{e_k^\nu - E_k^\nu x | k \in J_\nu\}$$

také člen

$$\rho_\nu(x) = \frac{1}{2} \parallel x - \overline{x}^\nu \parallel^2 .$$

Pokud budeme uvažovat množinu K_1 pomocí nerovností $K_1 = \{x | Ax \leq b\}$, potom regularizovaný hlavní problém (RM^ν) ekvivalentně zapíšeme jako

$$\min_{x \in K_1} \phi_\nu(x) + \rho_\nu(x).$$

Problém (RM^ν) můžeme dále zapsat pomocí proměnné θ ve tvaru

$$\begin{aligned} \min_{x,\theta} \quad & \theta + \rho_\nu(x) \\ \text{za podmínek} \quad & \theta \geq e_k^\nu + (c - E_k^\nu)x \quad \forall\, k \in J_\nu \\ & Ax \leq b. \end{aligned} \tag{2.17}$$

Vlastnostmi této úlohy se dále nebudeme zabývat. Uvedeme pouze nutné podmínky optimality pro úlohu (2.17) uvedené v [8], abychom ukázali, které řezy v regularizovaném hlavním problému zůstanou. Nechť μ představuje řádkový vektor Lagrangeových multiplikátorů, které postupně přísluší podmínkám $Ax \leq b$. Stejně pro $\forall\, k \in J_\nu$ označme Lagrangeovy multiplikátory λ_k příslušné podmínkám na θ. Následující podmínky jsou poté nutnými podmínkami optima pro úlohu (2.17) :

$$\begin{aligned} Ax - b &\leq 0 \\ -\theta + e_k^\nu + (c - E_k^\nu)x &\leq 0 \quad \forall\, k \in J_\nu \\ \mu(Ax - b) &= 0 \\ \lambda_k[-\theta + e_k^\nu + (c - E_k^\nu)x] &= 0 \quad \forall\, k \in J_\nu \\ \mu A + \sum_{k \in J_\nu} \lambda_k (c - E_k^\nu) + (x - \overline{x}^\nu)^T &= 0 \\ \sum_{k \in J_\nu} \lambda_k = 1,\, \lambda \geq 0, \mu &\geq 0. \end{aligned} \tag{2.18}$$

Nechť dvojice řešení $(x^{\nu+1}, \theta^{\nu+1})$ představuje řešení (2.18) získané v 5. kroku SD algoritmu. Dále označme příslušné optimální Lagrangeovy multiplikátory jako (λ^ν, μ^ν). Potom můžeme definovat indexovou množinu

$$J_{\nu+1} = \{k \in J_\nu | \lambda_k^\nu > 0\} \cup \{i_{\nu+1}, \nu+1\}.$$

Tedy v regularizovaném hlavním problému bude nově vytvořený řez, aktualizovaný řez v kandidátovi řešení a budou ponechány pouze aktivní řezy, tedy takové jejichž multiplikátor je kladný, což z (2.18) znamená, že

$$\theta = e_k^\nu + (c - E_k^\nu)x.$$

Zápis duální úlohy k (2.18) lze nalézt v [8] nebo v [9], kde jsou navíc uvedené další vlastnosti této dvojice úloh. Na základě těchto vlastností lze ukázat, že počet multiplikátorů splňujících $\lambda_k^\nu > 0$ je maximálně $n + 1$ [9].

V [8] je dokázáno, že takto regularizovaný SD algoritmus má stejné asymptotické vlastnosti jako původní, který v hlavní úloze uvažoval všechny vytvořené řezy. Snížení počtu řezů v hlavní úloze lze tedy shora omezit počtem $n_1 + 3$.

Ukončovací kritéria

Na závěr si ukážeme, jak lze algoritmus SD ukončit. Zatímco deterministické algoritmy označí vygenerovaný bod za optimum, jakmile se přesně definovaná horní a dolní mez sobě s danou přesností rovnají, stochastické algoritmy takto lehce identifikovatelná ukončovací kritéria nemají. Ve stochastických algoritmech je např. nutné tyto meze statisticky odhadnout. V [8] jsou popsané různé přístupy, jak lze SD algoritmus zastavit, jež jsou založené na out-of-sample scénářích nebo na in-sample scénářích s využitím bootstrapu. My si ukážeme ukončovací kritéria, která jsou odvozená na základě asymptotických vlastností a jež byla uvedena již v prvotním článku [7].

Tato kritéria vycházejí z předpokladu, že lze algoritmus ukončit, pokud se hodnota odhadů účelových funkcí v kandidátech řešení stabilizuje. Aktuální odhad se porovná s průměrnou hodnotou odhadů účelových funkcí. Pro tyto účely definujme následující průměrné hodnoty

$$\gamma^\nu = \frac{1}{\nu} \sum_{k=1}^{\nu} \mathcal{Q}_\nu(\overline{x}^\nu),$$
$$\overline{\gamma}^\nu = \frac{1}{m_\nu} \sum_{n=1}^{m_\nu} \mathcal{Q}_{\nu_n}(\overline{x}^{\nu_n}),$$

kde $\{\nu_n\}_{n=1}^{m_\nu}$ označuje posloupnost iterací, ve kterých došlo ke změně kandidáta řešení během prvních ν iterací, a m_ν jejich počet. Pokud je ν dostatečně velké, lze teoreticky ukázat, že alespoň jeden z výše uvedných průměrů konverguje k optimální hodnotě účelové funkce skoro jistě. Pro $\varepsilon > 0$ lze potom definovat následující ukončovací kritéria:

- Jestliže je kandidát řešení stejný pro dostatečně velký počet iterací, potom ukonči algoritmus, pokud

$$\frac{|\ \mathcal{Q}_\nu(\overline{x}^\nu) - \gamma^{\nu-1}\ |}{\gamma^{\nu-1}} < \varepsilon. \tag{2.19}$$

- Jestliže v některé z posledních iterací došlo ke změně kandidáta řešení, potom ukonči algoritmus, pokud

$$\frac{\mid \mathcal{Q}_\nu(\overline{x}^\nu) - \overline{\gamma}^{\nu-1} \mid}{\overline{\gamma}^{\nu-1}} < \varepsilon. \tag{2.20}$$

Tato kritéria závisejí na volbě uživatele, jenž určuje, kdy je počet iterací dostatečně velký či ε dostatečně malé. Jak si ukážeme v praktické části, toto rozhodnutí velkou měrou ovlivňuje, které řešení bude nalezeno jako optimální. Na závěr uveďme, že tato kritéria lze uvažovat pouze při splnění všech uvedených předpokladů pro použití algoritmu SD. Jejich zřejmou modifikaci v případě porušení předpokladu A3, který požaduje nezápornost funkce $Q(x,\boldsymbol{\xi})$, si rovněž uvedeme v praktické části.

2.3 Možná rozšíření a další algoritmy

V první části této kapitoly jsme si podrobně rozebrali základní verzi L-shaped algoritmu pro úlohu lineárního programování. Tento algoritmus ovšem nabízí několik možných rozšíření, která by jej měla zrychlovat. Mezi taková patří např. jeho multicut verze, při níž je v jedné iteraci přidáváno více řezů a ne pouze jediný. Z multicut L-shaped metody poté vychází regularizovaná dekompozice. Ta podobně jako v případě regularizovaného algoritmu SD přidává do účelové funkce kvadratický člen a vynechává neaktivní řezy.

V případě nelineárního programování je možné L-shaped algoritmus rozšířit také do polyedrického či kvadratického případu. Pokud však množina přípustných řešení není polyedrická, nelze pro takové úlohy L-shaped algoritmus použít. Tento problém vyřešili autoři Wets a Rockafellar, kteří na tyto úlohy aplikovali Progressive hedging algoritmus. Jeho základní myšlenkou je místo proměnné x v účelové funkci uvažovat K proměnných x_k a úlohu řešit s pomocí relaxované podmínky neanticipativity.

Rozšíření L-shaped algoritmu do nelineárních případů stejně jako stručný popis Progressive hedging algoritmu jsou uvedeny v práci [2]. Všechny úlohy, které budou v následující kapitole popsány, jsou však úlohy lineárního programování. Pro jejich vyřešení proto využijeme algoritmy, které byly v této kapitole podrobně popsány. Pro zjednodušení budeme v praktické části porovnávat výsledky základní verze L-shaped algoritmu a algoritmu SD bez regularizace.

3. Aplikace algoritmů

V této kapitole si na příkladech ukážeme aplikaci popsaných algoritmů. Každý příklad nejprve teoreticky popíšeme a poté úlohy formulujeme jako dvoustupňové problémy. Ty následně vyřešíme pomocí L-shaped algoritmu a algoritmu stochastické dekompozice.

L-shaped algoritmus byl naprogramován v algebraickém modelovacím jazyku GAMS (GAMS Development Corporation, 2016), který slouží k efektivnímu řešení optimalizačních úloh. Naopak, pro algoritmus SD, ve kterém je nutné náhodně generovat realizace náhodného vektoru, postupně ukládat duální multiplikátory a provádět dynamické změny optimálních řezů, byl pro implementaci použit skriptovací programovací jazyk MATLAB (MATLAB and Optimization Toolbox Release 2017a, The MathWorks, Inc.). Spouštět GAMS pro řešení úloh lineárního programování v průběhu algoritmu SD z MATLABu se neukázalo pro naše úlohy jako časově výhodná varianta. Jelikož jednotlivé subproblémy i hlavní problém algoritmu jsou v MATLABu řešitelné bez problémů, rozhodli jsme se celý algoritmus SD spouštět v něm.

Pro vyhodnocování rychlosti daných algoritmů jsme se zaměřili na porovnání počtu úloh lineárního programování, které tyto algoritmy musí provést. Tento způsob považujeme za lepší měřítko než naměření času potřebného k výpočtu (v sekundách/minutách), jenž velkou měrou závisí na tom, jak efektivně byl daný algoritmus naprogramován. Časové údaje by v našem případě navíc vypovídaly spíše o srovnání výkonnosti MATLABu a GAMSu než o samotných algoritmech.

V případě ukončovacích kritérií pro algoritmus SD jsme se zaměřili na výpočet stability hodnoty účelové funkce v kandidátech řešení na základě kritéria (2.19). Jelikož jsme omezovali maximální počet iterací (např. 1000), po který jsme chtěli algoritmus nechat spuštěný, toto ukončovací kritérium jsme modifikovali. Průměrnou hodnotu účelové funkce jsme nepočítali od prvních iterací, ve kterých je hodnota účelové funkce zpravidla velmi odlišná od optimální hodnoty, ale pouze v několika posledních iteracích. Odchylku od tohoto průměru jsme spočetli pouze za podmínky, že se algoritmus nacházel alespoň v minimálně stanovené iteraci a kandidát řešení zůstal v posledních iteracích neměnný. Jelikož algoritmus SD konverguje k optimálnímu řešení až po dostatečném počtu iterací, který může být mnohem větší než 1000, jsou tato ukončovací kritéria slabší než ta popsána v předchozí kapitole, a nemusí proto pokaždé vést k nalezení optimálního řešení. Algoritmus SD byl vždy spuštěn několikrát s různými počátečními semínky. Za optimální hodnotu, která je prezentována ve výsledcích, pak byl brán průměr z nalezených optimálních řešení v jednotlivých bězích. Minimální počet iterací byl volen většinou v závislosti na počtu scénářů použitých v L-shaped algoritmu, maximální počet naopak tak, aby byl každý běh ukončen v rozumném časovém horizontu (v jednotkách až desítkách minut).

První aplikační příklad byl spočítán na základě reálných finančních dat, pro další dva příklady jsme data simulovali. Zdrojové kódy obou algoritmů jsou přiložené na CD včetně souboru *napoveda.pdf*, který obsahuje krátkou nápovědu k použití přiložených souborů.

3.1 Minimalizace CVaRu

První aplikační ukázkou představených algoritmů je příklad z finančního prostředí. Jedná se o „mean-risk“ model, který zjišťuje optimální volbu portfolia na základě zisku a rizika. Vychází z původního Markowitzova modelu, jenž byl představen pozdějším nositelem Nobelovy ceny za ekonomii H. Markowitzem v 50. letech 20. století. Ten ve své práci [15] uvažoval za míru rizika rozptyl. My budeme v následující kapitole minimalizovat CVaR za podmínky, že výnos portfolia dosáhne alespoň námi určené hranice. Pro účely této aplikace zavedeme následující značení:

x	vektor vah jednotlivých akcií v portfoliu
N	počet možných akcií
$\boldsymbol{R}$	náh. veličina reprezentující zisk, zisk portfolia s váhami x označme $\boldsymbol{R}(x)$
r_0	minimální požadovaný výnos
$\boldsymbol{L}$	náh. vel. reprezentující ztrátu, ztrátu portfolia s váhami x označme $\boldsymbol{L}(x)$
$\mathcal{A}$	množina náhodných veličin $\boldsymbol{L}$

Náhodné veličiny budeme uvažovat se zprava spojitou distribuční funkcí, které jsou definované na měřitelném prostoru $(\Omega, \mathcal{F})$ s hodnotami v $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$. Náhodná veličina $\boldsymbol{L}$ reprezentuje ztrátu tím způsobem, že pokud je $\boldsymbol{L}(\omega) > 0$ pro nějaké $\omega \in \Omega$, potom říkáme, že se utrpěla ztráta. Pokud $\boldsymbol{L}(\omega) < 0$, potom bylo dosaženo zisku. Tedy uvažujeme vztah, že záporný výnos je roven ztrátě, neboli $\boldsymbol{L}(\omega) = -\boldsymbol{R}(\omega)$. Dále budeme argument ω vynechávat.

Výnos celého portfolia se získá pomocí výnosů jednotlivých akcií obsažených v portfoliu následujícím způsobem:

$$\boldsymbol{R}(x) = \sum_{i=1}^{N} \boldsymbol{R}_i x_i.$$

Ztráta celého portfolia se získá stejným způsobem. Ztrátu využijeme při definování měr rizika, zisk portfolia naopak chápeme jako funkci výnosu. Zaveďme funkcionál vyjadřující očekávaný výnos portfolia s váhami x jako $\rho_x \equiv \rho(\boldsymbol{R}(x))$ a funkcionál pro riziko portfolia s váhami x jako $V_x \equiv V(\boldsymbol{L}(x))$. Poté můžeme definovat eficientní portfolio následovně.

Definice 3 (Eficientní portfolio)**.** *Portfolio s váhami x^* je eficientní vzhledem k výnosu ρ a riziku V, jestliže neexistuje žádné jiné x takové, že $\sum_{i=1}^{N} x_i = 1$, pro které platí: $\rho_x \geq \rho_{x^*}$ a zároveň $V_x \leq V_{x^*}$, kde alespoň jedna nerovnost je splněna ostře.*

Eficientní portfolio získáme např. pomocí optimalizační úlohy, jež v účelové funkci minimalizuje riziko V, a požadovaný zisk portfolia je v podmínce:

$$\begin{aligned} \min_{x} \quad & V_x \\ \text{za podmínek} \quad & \rho_x \geq r_0 \\ & \sum_{i=1}^{N} x_i = 1 \\ & x_i \in \mathbb{R}, \quad i = 1, \ldots, N. \end{aligned} \tag{3.1}$$

V zápise optimalizační úlohy (3.1) se uvažuje prostředí, ve kterém jsou povoleny tzv. prodeje nakrátko, tedy prodeje akcií, které nevlastníme. Ty lze zakázat přidáním podmínky

$$x_i \geq 0, \quad i = 1, \ldots, N.$$

Podmínky pro x včetně zakázaných prodejů nakrátko budeme dále zkráceně zapisovat jako $x \in X$.

Nyní si uvedeme definice měr rizika, které budeme v této kapitole v „mean-risk“ modelu uvažovat.

Definice 4. *Nechť $\boldsymbol{L} \in \mathcal{A}$ a nechť je dána míra spolehlivosti $\alpha \in (0,1)$, potom*

$$\mathrm{VaR}_\alpha(\boldsymbol{L}) = \inf\{l \in \mathbb{R} : P(\boldsymbol{L} > l) \leq 1 - \alpha\}$$

se nazývá Value-at-Risk (hodnota v riziku) na hladině α.

VaR udává ztrátu, jež s pravděpodobností $(1-\alpha)$ nebude překročena. Obvykle se α volí velké, blízko 1, např. v rámci Solvency II (regulatorika pojišťoven) je $\alpha = 0.995$. Nevýhodou této míry rizika je, že se nezabývá ztrátami, které jsou vetší než tato hranice, ačkoliv tyto ztráty mohou nabývat vysokých hodnot. Výši ztrát, jež mohou nastat po překročení $\mathrm{VaR}_\alpha(\boldsymbol{L})$, zohledňuje CVaR.

$$\mathrm{CVaR}_\alpha(\boldsymbol{L}) = \mathsf{E}\,[L | L > \mathrm{VaR}_\alpha(\boldsymbol{L})].$$

CVaR_α tedy vyjadřuje průměrnou ztrátu z $100(1-\alpha)$ % nejhorších ztrát portfolia. V [16] autoři ukázali ekvivalentní definici, kterou si zde uvedeme a budeme s ní dále pracovat.

Definice 5. *Nechť $\boldsymbol{L} \in \mathcal{A}$ a nechť je dána míra spolehlivosti $\alpha \in (0,1)$, potom*

$$\mathrm{CVaR}_\alpha(\boldsymbol{L}) = \inf_{u \in \mathbb{R}} \{u + \frac{1}{1-\alpha} \mathsf{E} \max(0, \boldsymbol{L} - u)]\}$$

se nazývá Conditional-Value-at-Risk (podmíněný VaR) na hladině α.

Při výpočtu CVaRu je navíc vypočtena také hodnota VaR, kterou v předešlé definici vyjadřuje optimální hodnota proměnné u. Mezi další výhody CVaRu oproti klasickému VaRu je, že se jedná o tzv. koherentní míru rizika.

Definice 6. *Zobrazení $V : \mathcal{A} \to \mathbb{R}$ se nazývá koherentní míra rizika, pokud pro každou ztrátu $\boldsymbol{L} \in \mathcal{A}$ a $\boldsymbol{M} \in \mathcal{A}$ splňuje následující vlastnosti:*

1. *Translační ekvivariance: $V(\boldsymbol{L} + c) = V(\boldsymbol{L}) + c$ pro $\forall\, c \in \mathbb{R}$.*
2. *Pozitivní homogenita: $V(0) = 0$ a $V(\lambda \boldsymbol{L}) = \lambda V(\boldsymbol{L})$ pro $\forall\, \lambda \geq 0$.*
3. *Subaditivita: $V(\boldsymbol{L} + \boldsymbol{M}) \leq V(\boldsymbol{L}) + V(\boldsymbol{M})$.*
4. *Monotonie: $\boldsymbol{L}(\omega) \geq \boldsymbol{M}(\omega), \forall\, \omega \in \Omega \Rightarrow V(\boldsymbol{L}) \geq V(\boldsymbol{M})$.*

Subaditivita a pozitivní homogenita pak zajistí konvexitu. Proto v případě „mean-risk“ modelu se bude jednat o optimalizační úlohu s konvexní účelovou funkcí.

Nejprve si uvedeme zápis optimalizační úlohy minimalizace CVaRu při stanoveném očekávaném výnosu r_o ve formě lineárního programování. Pro účely takto formulované úlohy si zavedeme novou proměnnou z, která bude představovat výraz $\max(0, \boldsymbol{L} - u)$ v definici CVaRu. Pro tuto proměnnou pak musí platit, že je nezáporná a v případě, že je ztráta $\boldsymbol{L}$ větší než VaR, který zde představuje proměnná u, je rovna jejich rozdílu. Toho docílíme tím, že proměnnou z budeme minimalizovat a zároveň ji zdola omezíme rozdílem $\boldsymbol{L} - u = -\boldsymbol{R} - u$.

Uvažujme navíc, že pro každou akcii v portfoliu máme K různých scénářů (tyto scénáře mohou být např. napozorované historické výnosy všech N uvažovaných akcií). Vektorem r_k značíme N-rozměrný vektor výnosů možných akcií v portfoliu pro scénář k. Dále p_k značí pravděpodobnost, že daný scénář k nastane, a součet těchto pravděpodobností přes všech K scénářů je roven 1. Speciálním případem jsou stejně pravděpodobné scénáře, tedy $p_k = \frac{1}{K}$.

Úloha lineárního programování pro minimalizaci CVaRu při daném očekávaném výnosu:

$$\begin{aligned} \min_{x,u,z_k} \quad & u + \frac{1}{(1-\alpha)} \sum_{k=1}^{K} p_k z_k \\ \text{za podmínek} \quad & z_k \geq -x^T r_k - u, \quad k = 1, \ldots, K \\ & z_k \geq 0, \quad k = 1, \ldots, K \\ & \sum_{k=1}^{K} p_k x^T r_k \geq r_0 \\ & x \in X. \end{aligned} \tag{3.2}$$

První dvě omezení přísluší CVaRu, třetí podmínka představuje požadovaný očekávaný výnos portfolia vyjádřený na základě K scénářů.

3.1.1 Dekompoziční algoritmy

Při použití dekompozičních algoritmů budeme vycházet z úlohy lineárního programování (3.2), nyní je však nutné si uvědomit, co je primární a co sekundární úloha. Rozhodnutí, které musíme učinit v současnosti a spadá tak do primární úlohy, je volba akcií, které budou obsaženy v portfoliu, a jejich váhy. Další proměnná, jež nebude ovlivněná náhodou v budoucnosti a spadá rovněž do primární úlohy, je hodnota v riziku u.

Primární úloha:

$$\begin{aligned} \min_{x,u} \quad & u + \frac{1}{(1-\alpha)} \sum_{k=1}^{K} p_k Q_k(x,u) \\ \text{za podmínek} \quad & \sum_{k=1}^{K} p_k x^T r_k \geq r_0 \\ & x \in X. \end{aligned} \tag{3.3}$$

Na základě náhodných výnosů, které jsou reprezentovány scénáři r_k, následně řešíme v případě L-shaped algoritmu K dílčích subproblémů při daném (x,u):

$$\begin{aligned} Q_k(x,u) = \min_{y_k} \quad & y_k \\ \text{za podmínek} \quad & y_k \geq -x^T r_k - u \\ & y_k \geq 0. \end{aligned} \tag{3.4}$$

Nyní si uvedeme duální úlohu k problému (3.4), která je tvaru

$$\begin{aligned} \max_{\pi_k} \quad & \pi_k(-x^T r_k - u) \\ \text{za podmínek} \quad & 0 \leq \pi_k \leq 1. \end{aligned} \tag{3.5}$$

Úloha (3.4) není třeba optimalizovat, při daném (x,u) stačí pro každý subproblém pouze porovnat, zda je $-x^T r_k - u \geq 0$, či nikoliv. Podle toho je $y_k = -x^T r_k - u$, resp. $y_k = 0$. Analogicky také získáme duální multiplikátory z úlohy (3.5) jako $\pi_k = 1$, resp. $\pi_k = 0$. Na základě této vlastnosti vidíme, že lze v každém iteračním kroku uvažovat indexovou množinu $\mathcal{J}^\nu \subset \{1,\ldots,K\}$ takovou, že

$$\pi_k^\nu = \begin{cases} 1 & k \in \mathcal{J}^\nu, \\ 0 & k \notin \mathcal{J}^\nu. \end{cases}$$

Dále má problém (3.4) úplnou kompenzaci, tedy druhý krok L-shaped algoritmu, který kontroluje přípustnost, lze vynechat. Poté můžeme sestavit hlavní problém algoritmu následujícím způsobem

$$\begin{aligned} \min_{x,u,\theta} \quad & u + \frac{1}{(1-\alpha)}\theta \\ \text{za podmínek} \quad & \sum_{k=1}^{K} p_k x^T r_k \geq r_0 \\ & x \in X \\ & E_l x + F_l u + \theta \geq e_l, \quad l = 1,\ldots,\nu-1. \end{aligned} \tag{3.6}$$

Výsledkem této optimalizační úlohy je $(N+2)-$rozměrný vektor $(x_1,\ldots,x_N,u,\theta)^T$. Koeficienty E_l přísluší váhám x a koeficienty F_l přísluší hodnotě v riziku u. Jelikož v úloze (3.4) je $T_k = (r_k^T,1)$ a $h_k = 0$ pro každé $k = 1,\ldots,K$, získáváme koeficienty u řezu v ν-té iteraci ve tvaru

$$\begin{aligned} E_\nu^T &= \sum_{k=1}^{K} p_k \pi_k^\nu r_k = \sum_{k\in\mathcal{J}^\nu} p_k r_k, \\ F_\nu^T &= \sum_{k=1}^{K} p_k \pi_k^\nu = \sum_{k\in\mathcal{J}^\nu} p_k, \\ e_\nu^T &= 0. \end{aligned}$$

Myšlenku formulovat úlohu pomocí indexové množiny přinesly autoři Künzi-Bay a Mayer, kteří v [13] uvedli ekvivalentní formulaci úlohy (3.3)-(3.4). Takto získáme řezy optimality přidané do úlohy (3.6) v ν-té iteraci ve tvaru

$$x^T \sum_{k\in\mathcal{J}^\nu} p_k r_k + u \sum_{k\in\mathcal{J}^\nu} p_k + \theta \geq 0.$$

Pokud bychom v posledním řádku hlavní úlohy L-shaped algoritmu (3.6) přidali ke koeficientům řezů navíc horní index $\nu - 1$, tj.

$$E_l^{\nu-1}x + F_l^{\nu-1}u + \theta \geq 0, \quad l = 1, \ldots, \nu - 1,$$

potom bychom dostali zápis hlavní úlohy algoritmu stochastické dekompozice. Pro připomenutí, dolní index vyjadřuje iteraci, v níž byl pomocí algoritmu SD řez vytvořen. Horní index poté vyjadřuje pořadí iterace, ve které byly všechny předchozí řezy obnoveny a ve které byly vytvořené řezy pro poslední nalezené (x,u) z hlavní úlohy a posledního kandidáta řešení. V algoritmu SD se tedy řeší v každé iteraci dva subproblémy typu (3.4), jeden pro váhy portfolia x^ν a hodnotu v riziku u^ν a druhý pro kandidáta řešení s váhami $\overline{x}^{\nu-1}$ a hodnotou v riziku $\overline{u}^{\nu-1}$. Tím se získají koeficienty řezů (E_ν^ν, F_ν^ν), resp. $(E_{k_{\nu-1}}^\nu, F_{k_{\nu-1}}^\nu)$.

Je zřejmé, že účelová funkce ve druhém stupni vyjadřující ztrátu nad VaR je nezáporná, tedy také třetí předpoklad pro použití algoritmu SD je splněn a lze jej použít bez omezení.

3.1.2 Numerické výsledky

K testování algoritmů jsme použili reálná data dostupná z [1], jež obsahují měsíční výnosy reprezentativních portfolií tvořené firmami na třech amerických burzách (NYSE, NASDAQ, AMEX). Tyto firmy jsou rozděleny do decilů pomocí tržní kapitalizace, přičemž v každém decilu je vytvořeno portfolio z firem, které do něj spadají, opět pomocí tržní kapitalizace. Do těchto decilů jsou firmy rozdělovány každoročně v červnu, tedy meziročně se počet firem v jednotlivých decilech může měnit. Zároveň může být po roce firma na základě tržního vývoje přeřazená z jednoho decilu do jiného. Do těchto portfolií se reálně nedá investovat, slouží však jako ukázkový zdroj dat k testování a porovnání výsledků obou algoritmů.

Pro prezentaci výsledků jsme se rozhodli použít dvacetiletou historii, tedy 240 scénářů výnosů, od října 1996 do září 2016 (pro desetiletou či padesátiletou historii jsou závěry podobné).

Následující tabulka shrnuje průměrné výnosy všech deseti portfolií a jejich směrodatné odchylky. Tyto hodnoty jsou uvedené v procentech, tedy výnos 5 % znamená, že z jedné investované jednotky jmění na začátku období bude na konci období 1.05 jednotek jmění. Naopak ztráta 5 %, tj. výnos −5 % znamená na konci investovaného období 0.95 jednotek jmění.

Tabulka 3.1: Průměrné výnosy a jejich směrodatné odchylky [%]

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
Výnosy	1.0014	0.9961	1.0334	0.8883	0.9472	0.9215	0.9755	0.9549	0.9140	0.6939
StD	6.3905	6.7995	6.2267	5.9258	5.8874	5.3295	5.2316	5.2692	4.6809	4.4223

Na základě hodnot uvedených v tabulce 3.1 jsme volili minimální výnos $r_0 = 1.01$ %, tedy je požadováno investovat tak, aby portfolio bylo na konci období ziskové a vyplatilo se riziko spojené s touto investicí podstupovat. Tedy je požadováno, aby jmění při investování jedné jednotky na počátku bylo na konci období v průměru 1.0101 jednotek. Z tabulky 3.1 lze vidět, že bude nutné část majetku investovat do třetího portfolia, protože je jediné, které má průměrný

výnos alespoň 1.01 %. Z důvodů diverzifikace portfolia pak bude potřeba část jmění investovat také do jiného portfolia x_i, které je sice méně výnosné, avšak zároveň méně rizikové. Riziko v tabulce vyjadřují směrodatné odchylky, pro portfolio s větší směrodatnou odchylkou platí, že jeho výnosy/ztráty vykazují větší volatilitu. Mezi portfolii x_i jsou navíc značné korelace. V naší úloze však není riziko vyjádřeno kovarianční maticí, nýbrž CVaRem na hladině $\alpha = 0.95$.

Pomocí Bendersovy dekompozice bylo nalezeno optimum úlohy (když rozdíl mezi horní a dolní mezí byl menší než 0.000001) po 20 nalezených řezech optimality. K tomu tedy bylo zapotřebí vyřešit $20 * 240$ subproblémů a 20 hlavních problémů, celkem tedy 4820 úloh lineárního programování. Výsledky algoritmu se shodovaly s výsledky úlohy ve tvaru (3.2).

Optimální váhy portfolia

$$x^* = (0, 0, 0.5957, 0, 0, 0, 0.4043, 0, 0, 0), \tag{3.7}$$

optimální hodnota v riziku

$$u^* = 8.4406\,\% \tag{3.8}$$

a optimální účelová funkce

$$\mathrm{CVaR}_{0.95} = 12.3999\,\%. \tag{3.9}$$

Tedy aby byl zajištěn požadovaný výnos 1.01 %, je optimální z hlediska minimálního CVaRu investovat do třetího portfolia 0.5957 jednotek a zbylých 0.4043 investovat do 7. portfolia. S pravděpodobností 0.95 pak ztráta nebude vetší než 8.44 %. V případě, že bude větší než tato hraniční hodnota, potom bude v průměru 12.4 %.

Nyní si ukážeme výsledky získané algoritmem stochastické dekompozice, který byl spuštěn na 30 nezávislých bězích (tj. pro 30 různých počátečních semínek). Generovalo se z 240 scénářů, které byly použité pro výpočet pomocí Bendersovy dekompozice. Jak už bylo řečeno v předchozí kapitole, pokud není algoritmus SD ukončen na základě složitých statistických kritérií, musí uživatel zadat minimální počet iterací, jenž musí algoritmus proběhnout. Pokud je voleno ukončovací kritérium na základě nerovnosti (2.19), je potřeba také zadat, kdy je pro uživatele účelová funkce dostatečně stabilní. Dále musí uživatel určit minimální počet iterací, během nichž je požadováno, aby kandidát řešení zůstal neměnný.

Pro znázornění výsledků jsme zvolili dvě různá ukončovací kritéria, která se shodovala v podmínce minimálního počtu iterací 240 (počet volen na základě počtu scénářů použitých pro Bendersovu dekompozici) a v podmínce pro ustálenost kandidáta řešení po alespoň 20 iterací. Nejprve jsme volili maximální počet iterací 500, pokud se účelová funkce v kandidátovi řešení $\mathcal{Q}_\nu(\overline{x}^\nu)$ od průměru v posledních dvaceti iteracích $\gamma_{20}^{\nu-1}$ lišila o méně než 0.005 násobek tohoto průměru. Tedy $\frac{\mathcal{Q}_\nu(\overline{x}^\nu) - \gamma_{20}^{\nu-1}}{\gamma_{20}^{\nu-1}} < 0.005$. Poté jsme použili striktnější ukončovací kritérium s odchylkou od tohoto průměru o méně než 0.0001 s větším možným maximálním počtem iterací (2000).

Tabulka 3.2: Průměrné váhy portfolia, které jsou nenulové, a jejich směrodatné odchylky pro dvě různá ukončovací kritéria

	x_1	x_3	x_7	x_9
Váhy 1	0.0042	0.5990	0.3920	0.0049
StD 1	0.0230	0.0226	0.0396	0.0185
Váhy 2	0.0000	0.5957	0.4043	0.0000
StD 2	0.0000	0.0000	0.0000	0.0000

Z tabulky 3.2 vidíme, že algoritmus nenachází v každém běhu při toleranci $\varepsilon = 0.005$ optimální řešení (3.7). Z celkového počtu 30 běhů nalezlo jiné řešení ve 3 případech. Přirozeně, se striktnějšími ukončovacími kritérii a s větším možným maximálním počtem iterací jsou řešení přesnější, jak ukazuje druhá část tabulky.

Následující tabulka shrnuje výsledky pro hodnotu v riziku. Nalezené řešení hodnoty VaR je méně přesné v porovnání s výše uvedenými váhami, které jsou kromě vytvořených řezů navíc omezené dalšími podmínkami v prvním stupni. Tabulka dále uvádí průměrnou hodnotu účelové funkce v době, kdy byl algoritmus ukončen, a pořadí iterace, v níž k ukončení došlo. V závorce je udán počet běhů, které nebyly ukončeny na základě ukončovacích kritérií, nýbrž maximálním počtem povolených iterací. Zkratka StD (standard deviation), stejně jako v předchozí tabulce, označuje směrodatnou odchylku.

Tabulka 3.3: Průměrné optimální hodnoty VaR, CVaR, průměrný počet iterací algoritmu a jejich směrodatné odchylky pro dvě různá ukončovací kritéria

	VaR	CVaR	Iterace
Průměr 1	8.3888	12.1386	392 (9)*
StD 1	0.6456	1.2320	97
Průměr 2	8.4096	12.2867	973 (0)*
StD 2	0.4378	0.9565	388

* Počet běhů z 30 celkem ukončených maximálním počtem iterací.

Z tabulky 3.3 vidíme, že hodnota v riziku při menší toleranci ($\varepsilon < 0.0001$) je k optimální hodnotě (3.8) blíž, stejně tak odhad účelové funkce se blíží k optimální hodnotě (3.9). Při daných ukončovacích kritériích však nejsou optimální hodnoty nalezeny. Tento výsledek potvrzuje, že je algoritmus stochastické dekompozice vyvinut především k nalezení optimálního řešení x^*. Hodnotu účelové funkce odhaduje na základě vygenerovaného výběru scénářů, tedy pro 500, resp. 2000 scénářů se pro různá semínka liší. K ustálení odhadu účelové funkce by bylo zapotřebí větší počet iterací. V případě, že byl počet iterací omezen na 500, byl v 9 případech algoritmus zastaven tímto maximálním počtem iterací, nikoliv splněnými ukončovacími kritérii. Pokud uvažujeme druhé ukončovací kritérium, které zastavilo algoritmus v optimálním řešení x^* pokaždé, bylo potřeba průměrně vypočítat 973*2 subproblémů a 973 hlavních úloh algoritmu. Algoritmus

SD (2919 úloh) tak provedl průměrně téměř o 2000 méně výpočtů úloh lineárního programování než L-shaped algoritmus (4820 úloh).

Na základě výše uvedených výsledků můžeme pro tuto úlohu shrnout, že pokud je algoritmus SD ukončen po nedostatečném počtu iterací, nenalezne pokaždé správné optimální hodnoty. Při vhodných ukončovacích kritériích však algoritmus SD dojde v úloze minimalizace CVaRu k optimálním váhám rychleji (vyřeší méně úloh lineárního programování) než L-shaped algoritmus. Tato rychlost je však kompenzována tím, že algoritmus SD poskytuje pouze odhad hodnoty účelové funkce pro dané scénáře. Na základě historických výnosů a nalezených vah lze však poté snadno spočítat tuto hodnotu přesně. Pro spojitě rozdělené výnosy by navíc také L-shaped algoritmus hodnotu účelové funkce pouze odhadoval. Důležitým faktem tak zůstává, že pro algoritmus SD je potřeba vhodně nastavit ukončovací kritéria tak, aby neběžel příliš dlouho a přitom nalézal správná řešení.

3.2 Problém prodavače novin

Klasickým příkladem, který využívá stochastického programování, je problém prodavače novin (v anglicky psané literatuře jej najdeme pod názvem *Newsvendor problem* nebo *Newsboy problem*). Tato úloha má za cíl vyřešit, jaký počet výtisků má prodavač novin u dodavatele nakoupit, aniž ví, jaká bude následující den po novinách poptávka. Ceny, za které noviny nakupuje a prodává, jsou fixní. V klasické úloze jsou neprodané noviny vyhozeny. My si zde uvedeme variantu úlohy, kdy prodavač může neprodané výtisky dodavateli vrátit a jeho ztráta z neprodaných novin se tím sníží [4].

V této úloze si zavedeme následující značení:

x	počet nakoupených novin za jednotkovou cenu c
$y(\boldsymbol{\xi})$	počet prodaných novin za jednotkovou cenu p
$w(\boldsymbol{\xi})$	počet vrácených novin za jednotkovou cenu r

V praxi předpokládáme $p > c > r$, tedy prodejní cena je vyšší než nákupní. Zároveň cena vrácených novin dodavateli je mnohem nižší než ta, za níž byly noviny nakoupeny. Počet novin, který lze u dodavatele nakoupit, je navíc omezen horním limitem u, což je také jediné omezení v prvním stupni. Tímto dostáváme zápis dvoustupňové úlohy

$$\begin{aligned} \min_{x} \quad & cx + \mathcal{Q}(x) \\ \text{za podmínek} \quad & 0 \leq x \leq u, \end{aligned} \tag{3.10}$$

kde

$$\mathcal{Q}(x) = \mathsf{E}_{\boldsymbol{\xi}}\, Q(x,\boldsymbol{\xi})$$

a

$$\begin{aligned} Q(x,\boldsymbol{\xi}) = \min_{y,w} \quad & -py(\boldsymbol{\xi}) - rw(\boldsymbol{\xi}) \\ \text{za podmínek} \quad & y(\boldsymbol{\xi}) \leq \boldsymbol{\xi} \\ & y(\boldsymbol{\xi}) + w(\boldsymbol{\xi}) \leq x \\ & y(\boldsymbol{\xi}), w(\boldsymbol{\xi}) \geq 0. \end{aligned} \tag{3.11}$$

Úloha je sestavená jako minimalizační, v účelové funkci se minimalizuje ztráta, ke které nákupem novin dojde. Záporná hodnota této účelové funkce znamená zisk prodavače novin poté, co si z tržby a obnosu z vrácení novin odečte náklady, které vynaložil na nákup novin. Je zřejmé, že účelová funkce bude maximálně nula, pokud žádné noviny nenakoupí. Funkce $-\mathcal{Q}(x)$ vyjadřuje očekávaný zisk prodavače novin z prodeje a vrácení novin, pokud se rozhodne pro nákup x kusů novin. Jednotlivé funkce $-Q(x,\boldsymbol{\xi})$ vyjadřují zisk z prodeje a vrácení novin při poptávce $\boldsymbol{\xi}$.

Je zřejmé, že se prodavač snaží uspokojit celou poptávku, může však prodat maximálně tolik novin, kolik má na stánku, tedy optimální počet prodaných novin je roven

$$y^*(\boldsymbol{\xi}) = \min(\boldsymbol{\xi}, x).$$

Prodavač navíc vrátí všechny neprodané noviny, pokud však poptávka přesáhne počet nakoupených novin, nevrátí zpět nic, tedy optimální počet vracených novin je roven

$$w^*(\boldsymbol{\xi}) = \max(x - \boldsymbol{\xi}, 0).$$

Očekávaná hodnota účelové funkce ve druhém stupni lze pak zapsat následovně

$$\mathcal{Q}(x) = \mathsf{E}_{\boldsymbol{\xi}}[-p\min(\boldsymbol{\xi}, x) - r\max(x - \boldsymbol{\xi}, 0)]. \tag{3.12}$$

Pokud bychom předpokládali spojité rozdělení náhodné poptávky $\boldsymbol{\xi}$ s konečnými druhými momenty, potom z tvrzení 5 bodu 3 víme, že funkce $\mathcal{Q}(x)$ je diferencovatelná. Optimem této úlohy je potom řešení rovnice $c + \mathcal{Q}'(x) = 0$. Pro $0 \leq x \leq u$, pokud $c + \mathcal{Q}'(x) > 0$, potom $x^* = 0$ a pokud $c + \mathcal{Q}'(x) < 0$, potom $x^* = u$. Výraz $\mathcal{Q}'(x)$ vyjadřuje první derivaci funkce $\mathcal{Q}(x)$ v bodě x. Z (3.12) lze funkci $\mathcal{Q}(x)$ spočítat analyticky

$$\begin{aligned} \mathcal{Q}(x) &= \int_{-\infty}^{x} (-p\xi - r(x-\xi))\, dF(\xi) + \int_{x}^{\infty} -px\, dF(\xi) \\ &= -(p-r)\int_{-\infty}^{x} \xi\, dF(\xi) - rx\, F(x) - px(1 - F(x)), \end{aligned}$$

kde $F(\boldsymbol{\xi})$ je distribuční funkce $\boldsymbol{\xi}$. Integrováním pomocí per partes dostáváme

$$\int_{-\infty}^{x} \xi\, dF(\xi) = xF(x) - \int_{-\infty}^{x} F(x).$$

Z toho plyne rovnost

$$\mathcal{Q}(x) = -px + (p-r)\int_{-\infty}^{x} F(\xi)d\xi$$

a následně

$$\mathcal{Q}'(x) = -p + (p-r)F(x),$$

tedy optimální řešení je

$$x^* = \begin{cases} 0 & \text{pokud } \frac{p-c}{p-r} < F(0), \\ u & \text{pokud } \frac{p-c}{p-r} > F(u), \\ F^{-1}(\frac{p-c}{p-r}) & \text{jinak.} \end{cases}$$

$F^{-1}(\alpha)$ vyjadřuje $\alpha-$kvantil rozdělení F. Pokud je F spojitá funkce, potom $x = F^{-1}(\alpha)$ lze ekvivalentně zapsat jako $\alpha = F(x)$. Jelikož předpokládáme, že poptávka je nezáporná ($F(0) = 0$) a prodejní cena je vyšší než nákupní, nestane se, že by optimální řešení bylo $x^* = 0$ a k prodeji novin by vůbec nedošlo.

Poznámka. Výše jsme ukázali přesné analytické řešení. Pokud by nás však zajímala pouze otázka, zda je úloha přípustná ve druhém stupni, pomocí tvrzení 3 lze ukázat, že je funkce $\mathcal{Q}(x)$ konečná. Aby ovšem bylo průkazné, že problém prodavače novin má jednoduchou kompenzaci, je potřeba úlohu (3.10) přepsat do tvaru

$$\begin{aligned} \min_{x} \quad & (c-p)x + \mathcal{Q}(x) \\ \text{za podmínek} \quad & 0 \leq x \leq u, \end{aligned} \tag{3.13}$$

kde

$$\mathcal{Q}(x) = \mathsf{E}_{\boldsymbol{\xi}}\, Q(x, \boldsymbol{\xi})$$

a

$$\begin{aligned} Q(x,\boldsymbol{\xi}) = \min_{y^+, y^-} \quad & 0y^+(\boldsymbol{\xi}) + (p-r)y^-(\boldsymbol{\xi}) \\ \text{za podmínek} \quad & y^+(\boldsymbol{\xi}) - y^-(\boldsymbol{\xi}) = \boldsymbol{\xi} - x \\ & y^+(\boldsymbol{\xi}), y^-(\boldsymbol{\xi}) \geq 0. \end{aligned} \tag{3.14}$$

Výraz $(p-r)y^-(\boldsymbol{\xi})$ vyjadřuje korekční člen. Prodavač doufá v prodej všech x novin, které od dodavatele nakoupil, pokud je však poptávka nižší, je po realizaci této náhodné veličiny $\boldsymbol{\xi}$ potřeba účelovou funkci „upravit“. V případě minimalizační úlohy, kdy záporná hodnota vyjadřuje zisk, je třeba účelovou funkci navýšit o očekávanou avšak nenaplněnou tržbu sníženou o částku získanou z vrácených novin.

Vidíme, že q^+, q^- z tvrzení 3 je $q^+ = 0, q^- = p - r$, tedy výraz $q^+ + q^- = p - r > 0$ při uvedeném reálném předpokladu. Podle tvrzení 3 je tak problém (3.13) v druhém stupni přípustný.

3.2.1 Dekompoziční algoritmy

Podobně jako úloha minimalizace CVaRu, také úloha prodavače novin lze vypočítat jako úloha lineárního programování bez využití dekompozičních vlastností. Takto sepsanou úlohu GAMS dokáže vyřešit velmi rychle. My pro účely aplikace L-shaped algoritmu a algoritmu SD budeme primárně pracovat se zápisem úlohy ve tvaru (3.10)-(3.11).

Nejprve předpokládejme řešení úlohy pomocí L-shaped algoritmu na základě K scénářů poptávky, tedy střední hodnotu aproximujeme pomocí výběrového průměru vypočtených funkcí $Q_k(x, \boldsymbol{\xi} = \xi_k)$ tvaru (3.11). Ke každé z nich sestavíme duální úlohu

$$\begin{aligned} \max_{\pi_{k,1},\pi_{k,2}} \quad & \xi_k\pi_{k,1} + x\pi_{k,2} \\ \text{za podmínek} \quad & \pi_{k,1} + \pi_{k,2} \leq -p \\ & \pi_{k,2} \leq -r - p \\ & \pi_{k,1}, \pi_{k,2} \leq 0, \end{aligned} \tag{3.15}$$

kde $\pi_{k,i}$ označuje $i-$tý prvek vektoru duálních multiplikátorů π_k, který přísluší $k-$tému subproblému. Zde opět stačí pouze vyhodnotit, zda je daná poptávka menší, nebo větší než x získané z hlavního problému:

- Pokud $\xi_k \leq x$, potom $\pi_k = (-p + r, -r)^T$.
- Pokud $\xi_k > x$, potom $\pi_k = (0, -p)^T$.

Pro scénáře s pravděpodobnostmi p_k a vektory $h_k = (\xi_k, 0)^T$ a $T_k = (0, -1)^T$ můžeme v $\nu-$té iteraci sestrojit řezy ve tvaru

$$\begin{aligned} E_\nu &= \sum_{k=1}^{K} p_k(\pi_k^\nu)^T T_k = \sum_{k:\,\xi_k \leq x^\nu} p_k r + \sum_{k:\,\xi_k > x^\nu} p_k p, \\ e_\nu &= \sum_{k=1}^{K} p_k(\pi_k^\nu)^T h_k = \sum_{k:\,\xi_k \leq x^\nu} p_k(-p + r)\xi_k. \end{aligned}$$

Poté dostáváme hlavní problém L-shaped algoritmu ve tvaru

$$\begin{aligned} \min_{x,\theta} \quad & cx + \theta \\ \text{za podmínek} \quad & 0 \leq x \leq u \\ & x \geq 0 \\ & \theta \geq e_l - E_l x, \quad l = 1, \ldots, \nu. \end{aligned} \tag{3.16}$$

Problém má relativně úplnou kompenzaci, není proto potřeba generovat řezy přípustnosti, algoritmus vytváří pouze řezy optimality. Pokud bychom stejně jako pro úlohu minimalizace CVaRu v posledním řádku (3.16) hlavní úlohy L-shaped algoritmu přidali ke koeficientům řezů navíc horní index ν, tj.

$$\theta \geq e_l^\nu - E_l^\nu x, \quad l = 1, \ldots, \nu,$$

potom bychom dostali zápis hlavní úlohy algoritmu stochastické dekompozice. V algoritmu SD se v každé iteraci řeší dva subproblémy typu (3.15), jeden pro x^ν a druhý pro kandidáta řešení $\overline{x}^{\nu-1}$. Tím se získají koeficienty řezů (E_ν^ν, e_ν^ν), resp. $(E_{k_{\nu-1}}^\nu, e_{k_{\nu-1}}^\nu)$.

Než přejdeme k samotné implementační části, poznamenejme, že počet zákazníků reálně nemá spojité rozdělení. Naopak předpokládáme, že rozdělení je diskrétní, nabývající pouze nezáporných celých čísel s konečným nosičem. V tomto

případě nelze použít podmínky optimality tak, že položíme první derivaci účelové funkce rovnu nule. Jelikož víme, že optimální řešení bude konečné, můžeme naopak použít tvrzení 6.

Pro optimální x^* musí platit, že existuje nezáporné μ^* splňující $\mu^* x^* = 0$ takové, že

$$-c + \mu^* \in \partial\mathcal{Q}(x^*).$$

Předpokládejme, že počet zákazníků má diskrétní rozdělení G nabývající K hodnot s pravděpodobnostmi $p_k, k = 1,\ldots,K$. Potom lze podle tvrzení 7 vyjádřit subdiferenciál $\partial\mathcal{Q}(x^*)$ jako váženou sumu subdiferenciálů jednotlivých účelových funkcí. Podle tvrzení 8 lze tyto subdiferenciály vyjádřit pro účelovou funkci Q_k jako skalární součin $-T_k\pi_k$. Tedy $\partial\mathcal{Q}(x^*)$ je přesně záporně vzatá hodnota vyjádřeného koeficientu řezu E v bodě x^*, tedy

$$\begin{aligned}\partial\mathcal{Q}(x^*) &= -\sum_{k:\,\xi_k \le x^*} p_k r - \sum_{k:\,\xi_k > x^*} p_k p \\ &= -r\,G(x^*) - p\,(1 - G(x^*)).\end{aligned}$$

Pokud se vyplatí noviny nakoupit, tj. $x^* > 0$, potom z podmínky optimality plyne $\mu^* = 0$ a řešení je $G^{-1}(\frac{p-c}{p-r})$. Toto řešení je omezené horním limitem u. Řešení je naopak nulové pokud $\frac{p-c}{p-r} \le G(0)$. Celkově tak pro tuto úlohu umíme vyjádřit řešení analyticky ve tvaru

$$x^* = \begin{cases} 0 & \text{pokud } \frac{p-c}{p-r} \le G(0), \\ u & \text{pokud } \frac{p-c}{p-r} > G(u), \\ G^{-1}(\frac{p-c}{p-r}) & \text{jinak,} \end{cases} \tag{3.17}$$

kde $G^{-1}(\alpha)$ vyjadřuje $\alpha-$kvantil diskrétního rozdělení G. Z výše uvedeného vidíme, že pokud budeme rozdělení aproximovat pomocí scénářů, potom pro daný výběr scénářů bude řešením úlohy výběrový $\alpha-$kvantil.

3.2.2 Implementace a numerické výsledky

Pro tuto úlohu předpokládejme, že se počet zákazníků nejčastěji pohybuje mezi 70 a 130. Pro tyto účely byly pseudonáhodně vygenerovány scénáře poptávky ξ_k z normálního rozdělení se střední hodnotou 100 a rozptylem 100, $\mathcal{N}$(100,100). Jelikož tyto scénáře mají představovat počet zákazníků, byla vygenerovaná čísla následně zaokrouhlená na celá. Z přepsané úlohy (3.13) vidíme, že matice omezení je z definice P.9 totálně unimodulární, tudíž pro celočíselné pravé strany ξ_k je vektor řešení (x,y,w) celočíselný [18]. Optimální počet novin x, který má prodavač nakoupit, bude proto při takto vygenerovaných scénářích celé číslo.

Omezení pro počet novin bylo voleno 140 kusů, nákupní cena $c = 20$, prodejní $p = 25$ a cena za vrácení neprodaných kusů novin $r = 5$. Nalezené řešení je potom 0.25−kvantil tohoto diskrétního rozdělení (zaokrouhleného normálního rozdělení $\mathcal{N}$(100,100)), tedy $x^* = 93$.

Z implementačního hlediska je důležité si uvědomit, že ačkoliv jsou předpoklady A1 a A2 pro použití stochastické dekompozice splněny, předpoklad A3, který požaduje nezápornost funkce $Q(x,\boldsymbol{\xi})$, je porušen. Jak již bylo zmíněno v druhé kapitole, při obnovování řezů je tento předpoklad stěžejní. Lze jej však

obejít. Chceme-li zajistit, aby předchozí řezy byly dolními odhady aproximace účelové funkce $\mathcal{Q}_\nu$, kromě vynásobení absolutního členu daných přímek výrazem $\frac{\nu-1}{\nu}$ stačí navíc přičíst konstantu $\frac{L}{\nu}$, kde L vyjadřuje dolní mez, tedy $Q(x,\xi) \geq L$ pro všechna $(x,\xi) \in K_1 \times \Xi$. Tím se přímka posune ve směru osy y o tuto zápornou konstantu.

Dále je potřeba modifikovat ukončovací kritéria. Jelikož je funkce $Q(x,\xi) \leq 0$ pro všechna $(x,\xi) \in K_1 \times \Xi$, změnili jsme ukončovací kritérium (2.19) na

$$\frac{\mid \mathcal{Q}_\nu(\overline{x}^\nu) - \gamma^{\nu-1} \mid}{|\gamma^{\nu-1}|} < \varepsilon. \tag{3.18}$$

Algoritmus SD byl použit jak pro výpočet úlohy (3.10)-(3.11), ozn. NB1, tak pro výpočet přepsané úlohy (3.13)-(3.14), ozn. NB2. Dolní mez pro úlohu NB1 byla volena -140*p. Počet novin je omezen shora $x \leq 140$, v případě scénáře s vysokou poptávkou (140 a více) je optimální vektor ve druhém stupni roven $(y,w) = (140,0)$, tedy optimální hodnota může být nejméně -140*p. Úloha NB2 splňuje všechny předpoklady a nebylo pro ni nutné řezy upravovat o zápornou konstantu. Dále byla úloha spočtena také pomocí algoritmu s regularizovaným hlavním problémem na základě úlohy (3.10)-(3.11), ozn. NBr.

Při použití algoritmu SD bylo vždy použito ukončovací kritérium na základě ustálenosti kandidáta řešení po 20 iterací a ustálenost účelové funkce s odchylkou $\varepsilon = 0.0001$. Maximální počet iterací byl zvolen 1000 a nebyl nikdy překročen, ukončovací kritéria fungovala pro všechny úlohy a všechny běhy.

Následující tabulka 3.4 shrnuje průměrné výsledky a směrodatné odchylky vypočtené na základě 30 běhů. V každém běhu bylo vygenerováno vždy 100 scénářů poptávky.

Tabulka 3.4: Výsledky úlohy prodavače novin

	Počet novin	Zisk	Iterace
Průměr Benders	93.067	436.953	7.967
StD Benders	1.311	7.352	0.556
Průměr NB1	93.167	437.451	218
StD NB1	1.085	4.983	100
Průměr NB2	93.200	437.705	155
StD NB2	0.997	5.779	45
Průměr NBr	93.325	437.677	235
StD NBr	0.871	5.117	77

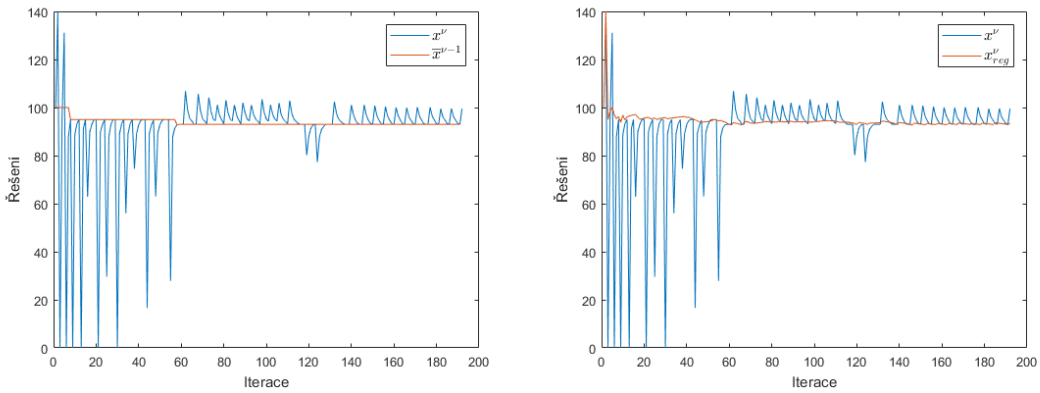
Z tabulky 3.4 vidíme, že L-shaped algoritmus vytvořil v průměru 8 řezů optimality, než došel k optimálnímu řešení. V každém běhu tak bylo L-shaped algoritmem průměrně spočteno přibližně 800 úloh lineárního programování. Algoritmus SD při minimálním požadovaném počtů běhů 100 byl ukončen průměrně po cca 200 iteracích. Vyřešil tak celkově přibližně 600 úloh lineárního programování, tedy méně než L-shaped algoritmus.

Optimální řešení této úlohy je $x^* = 93$ a zisk roven 436.432 (záporně vzata optimální účelová funkce). Z tabulky 3.4 vidíme, že výsledky pro L-shaped algoritmus a algoritmus SD jsou srovnatelné. Pokud porovnáme průměrné hodnoty,

lepších výsledků dosahuje L-shaped algoritmus, jeho hodnota optimálního řešení i účelové funkce je blíž ke správnému řešení. Na druhou stranu, algoritmus SD má menší směrodatnou odchylku nalezených řešení.

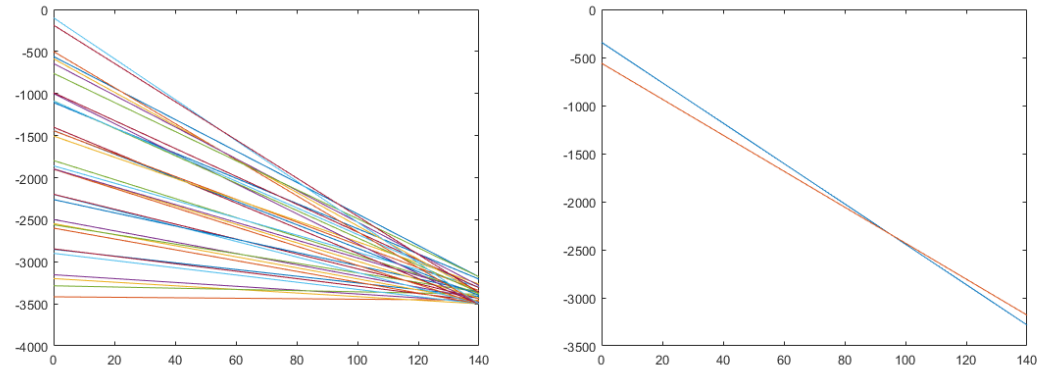
Regularizovaná verze algoritmu SD sice nalézá v průběhu algoritmu stabilnější řešení, nicméně z tabulky 3.4 vidíme, že vyžaduje větší počet iterací. Autoři v [8] uvedli, že na základě jejich výpočetní studie SD algoritmus s regularizovaným problémem, který obsahuje kvadratický člen $\rho_\nu(x) = \frac{1}{2} \parallel x - \overline{x}^\nu \parallel^2$, generuje posloupnost bodů konvergující k optimu pomalu. Namísto toho doporučují uvažovat parametrickou regularizaci $\rho_\nu(x) = \frac{\sigma_\nu}{2} \parallel x - \overline{x}^\nu \parallel^2$ a člen σ_ν měnit dynamicky. Tato modifikace je v [8] diskutována podrobněji .

Na závěr tohoto příkladu si zde názorně ilustrujeme stabilitu kandidátů řešení z klasického algoritmu SD v porovnání s posloupností řešení nalezených v jednotlivých iteracích. Dále porovnáme rozdíl mezi stabilitou řešení nalezených klasickým algoritmem SD a řešeními získanými pomocí regularizovaného hlavního problému. Ukážeme také počet řezů v hlavním problému klasického a regularizovaného SD algoritmu.



(a) Nalezená řešení a kandidáti řešení (b) SD a regularizovaná SD

Obrázek 3.1: Stabilita řešení



(a) Řezy přípustnosti pro SD (b) Aktivní řezy přípustnosti pro reg. SD

Obrázek 3.2: Srovnání počtu řezů pro SD a regularizovanou SD po 35 iteracích

Na obrázku 3.1a vidíme, že posloupnost kandidátů řešení se velmi rychle ustálí na optimální hodnotě daného běhu. Tyto hodnoty vykazují podstatně menší volatilitu v porovnání s posloupností řešení nalezených klasickým SD algoritmem v jednotlivých iteracích. Stabilnější řešení rovněž dává algoritmus SD s regularizovaným hlavním problémem, jak vidíme na obrázku 3.1b.

Obrázky 3.2 znázorňují řezy optimality po 35 iteracích algoritmu SD v porovnání s algoritmem SD s regularizovaným hlavním problémem. Na obrázku 3.2a vidíme, že algoritmus SD při ponechání všech předchozích řezů zbytečně navyšuje složitost hlavního problému, neboť většina z nich je neaktivním omezením a je možné je v hlavní úloze neuvažovat. Naopak na obrázku 3.2b je ukázáno, že v případě použití regularizované hlavní úlohy jsou po 35 iteracích zachovány pouze dva aktivní řezy.

3.3 Problém květinářky

Posledním aplikačním příkladem je problém květinářky, který je rozšířením klasického problému prodavače novin. Květinářka se rozhoduje, kolik nakoupí růží na víkendový prodej. Na rozdíl od prodavače novin se rozhoduje v pátek na dva dny dopředu, neboť se předpokládá, že lze při dobrých skladovacích podmínkách prodávat tyto růže za nejen v sobotu, ale také v neděli. V případě, že je poptávka v sobotu vysoká, může se květinářka rozhodnout přikoupit další množství růží na neděli.

Pro úlohu květinářky si zavedeme následující značení:

x	počet nakoupených novin v pátek
$z(\boldsymbol{\xi_1})$	počet nakoupených novin v sobotu po prvním dni prodeje
$s(\boldsymbol{\xi_1})$	počet skladovaných růží po prvním dni prodeje
$w(\boldsymbol{\xi_1}, \boldsymbol{\xi_2})$	počet neprodaných (vyhozených) růží po druhém dni prodeje
c	nákupní cena
p	prodejní cena

Náhodné veličiny $\boldsymbol{\xi_1}$, resp. $\boldsymbol{\xi_2}$ označují poptávku první, resp. druhý den. Počet uskladněných růží stejně jako počet nakoupených růží druhý den závisí na poptávce první den, zatímco počet růží, které se po víkendovém prodeji vyhodí, závisí na celkovém počtu zákazníků, tedy na obou náhodných veličinách.

Poptávka první den určí počet růží, které se uskladní na další den:

$$x - s(\boldsymbol{\xi_1}) \leq \boldsymbol{\xi_1}.$$

Počet růží, které se nakonec neprodají, je určen navíc také nedělní poptávkou:

$$z(\boldsymbol{\xi_1}) + s(\boldsymbol{\xi_1}) - w(\boldsymbol{\xi_1}, \boldsymbol{\xi_2}) \leq \boldsymbol{\xi_2}.$$

Ekvivalentní zápis úlohy prodavače novin (3.13)-(3.14) nám poslouží jako vodítko k sestavení úlohy květinářky, která je obecně třístupňová. Květinářka věří, že jí počet nakoupených růží v pátek postačí k uspokojení víkendové poptávky

a že všechny nakoupené růže prodá. Očekává proto, že její zisk bude $(p-c)x$. Pokud poptávka bude první prodejní den vysoká, dokoupí další růže na neděli a její očekávaný zisk se tímto nákupem navýší. Dojde ke korekci, tj. navýšení očekávaného zisku, o $(p-c)z(\xi_1)$ v závislosti na počtu zákazníků první den. Pokud květinářce po víkendovém prodeji některé růže zůstanou, bude muset svůj zisk naopak snížit o tržbu z jejich očekávaného prodeje. Tato korekce nastává po realizaci náhodného vektoru (ξ_1, ξ_2), tj. po realizaci obou poptávek.

Nechť máme K_1 scénářů počtu zákazníků první den s pravděpodobnostmi $p_k, k = 1, \ldots, K_1$ a K_2 scénářů počtu zákazníků druhý den s pravděpodobnostmi $p_l, l = 1, \ldots, K_2$. Pro jednoduchost uvažujme případ, že poptávka první den neovlivňuje poptávku následující den, budeme tedy předpokládat mezistupňovou nezávislost. Každý uzel scénářového stromu, který bude příslušet poptávce první den, se bude větvit stejným způsobem (bude mí stejných K_2 následníků). Úlohu květinářky poté formulujeme deterministickým ekvivalentem následujícím způsobem, kde jsou střední hodnoty nahrazeny váženými průměry přes příslušné scénáře

$$\begin{aligned}
\max_{x, z_k, w_{k,l}} \quad & (p-c)x + \sum_{k=1}^{K_1} p_k[(p-c)z_k - p\sum_{l=1}^{K_2} p_l w_{k,l}] \\
\text{za podmínek} \quad & x - s_k \leq \xi_{1,k} && k = 1, \ldots, K_1 \\
& z_k + s_k - w_{k,l} \leq \xi_{2,l} && k = 1, \ldots, K_1,\ l = 1, \ldots, K_2 \\
& (x_1, z_k, s_k, w_{k,l}) \geq 0 && k = 1, \ldots, K_1,\ l = 1, \ldots, K_2.
\end{aligned}$$

3.3.1 Dekompoziční algoritmy

Aby bylo možné použít L-shaped algoritmus a algoritmus SD, přeformulujeme úlohu do tvaru (1.2)-(1.4). Maximalizační úlohu je třeba převést na minimalizační (aby se účelové funkce rovnaly, je nutné účelovou funkci úlohy (3.19) vynásobit konstantou -1) a třetí stupeň vyřešit v rámci účelové funkce ve druhém stupni deterministickým ekvivalentem, čímž dostaneme následující formulaci:

$$\begin{aligned}
\min_{x} \quad & (c-p)x + \sum_{k=1}^{K_1} p_k Q_k(x, \xi_{1,k}) \\
\text{za podmínek} \quad & 0 \leq x \leq u,
\end{aligned} \tag{3.19}$$

kde na základě poptávky $\xi_{1,k}$ řešíme $k-$tý subproblém

$$\begin{aligned}
Q_k(x, \xi_{1,k}) = \min_{z_k, s_k, w_{k,l}} \quad & (c-p)z_k + p\sum_{l=1}^{K_2} p_l w_{k,l} \\
\text{za podmínek} \quad & -s_k \leq \xi_{1,k} - x, \\
& z_k + s_k - w_{k,l} \leq \xi_{2,l} \quad l = 1, \ldots, K_2, \\
& (z_k, s_k, w_{k,l}) \geq 0 \quad l = 1, \ldots, K_2,
\end{aligned} \tag{3.20}$$

který má celkem $(K_2 + 2)$ proměnných. Pokud označíme vektor proměnných ve druhém stupni y_k, potom $y_k = (z_k, s_k, w_{k,1}, \ldots, w_{k,K_2})$.

Vyřešením $k-$tého subproblému (3.20) získáme duální multiplikátory (η_k, π_k), kde η_k přísluší první podmínce určující počet uskladněných růží a K_2-rozměrný vektor π_k přísluší podmínkám, které určují počet vyhozených růží $w_{k,l}$. Každý subproblém (3.20) řeší ve druhém stupni pro nezáporné proměnné celkem K_2+1 podmínek, tedy vektory pravých stran omezení (tj. $h_k - T_k x$) jsou $(K_2+1)-$rozměrné. Vektor h_k se liší mezi jednotlivými subproblémy pouze v prvním řádku, $h_k = (\xi_{1,k}, \xi_2^T)^T$, kde $\xi_2^T = (\xi_{2,1}, \ldots, \xi_{2,K_2})$. Vektor T_k je shodný pro všechny subproblémy, $T_k = (1, 0, \ldots, 0)^T$. Řezy v $\nu-$té iteraci pro scénáře poptávky v 1. dni prodeje s pravděpodobnostmi p_k poté můžeme zapsat ve tvaru:

$$\begin{aligned} E_\nu &= \sum_{k=1}^{K_1} p_k \eta_k^\nu \\ e_\nu &= \sum_{k=1}^{K_1} p_k \eta_k^\nu \, \xi_{1,k} + \sum_{k=1}^{K_1} p_k (\pi_k^\nu)^T \, \xi_2. \end{aligned}$$

Tyto řezy jsou následně přidány do hlavní úlohy L-shaped algoritmu

$$\begin{aligned} \min_{x,\theta} \quad & (c-p)x + \theta \\ \text{za podmínek} \quad & 0 \le x \le u \\ & \theta \ge e_l - E_l x, \quad l = 1, \ldots, \nu. \end{aligned} \tag{3.21}$$

Problém má úplnou kompenzaci, není proto potřeba generovat řezy přípustnosti, algoritmus vytváří pouze řezy optimality. Pokud bychom stejně jako pro předchozí úlohy v posledním řádku (3.21) hlavní úlohy L-shaped algoritmu přidali ke koeficientům řezů navíc horní index ν, tj.

$$\theta \ge e_l^\nu - E_l^\nu x, \quad l = 1, \ldots, \nu,$$

potom bychom dostali zápis hlavní úlohy algoritmu stochastické dekompozice. V algoritmu SD se v každé iteraci řeší pro nově vygenerované $\xi_{1,\nu}$ dva subproblémy typu (3.20), jeden pro x^ν a druhý pro kandidáta řešení $\overline{x}^{\nu-1}$. Tím se získají koeficienty řezů (E_ν^ν, e_ν^ν), resp. $(E_{k_{\nu-1}}^\nu, e_{k_{\nu-1}}^\nu)$.

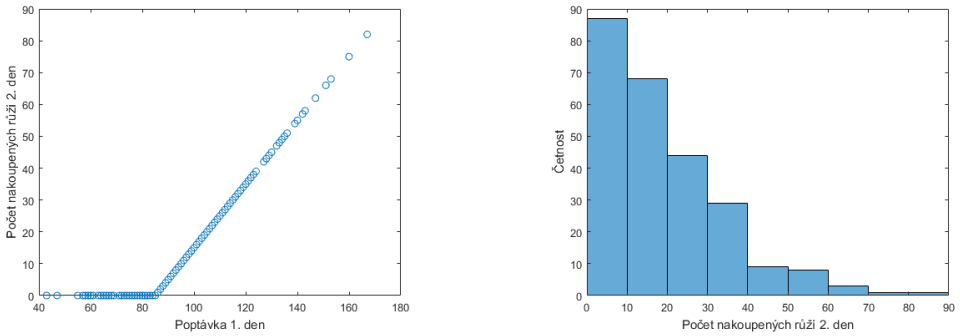
3.3.2 Implementace a numerické výsledky

Pro naše účely budeme nejprve uvedenou úlohy květinářky lehce modifikovat. Pokud bychom úlohu nechali ve tvaru popsaném výše, nedostávali bychom pro libovolné volby parametrů jednoznačné řešení. Pokud chceme porovnávat řešení, která dostaneme z různých algoritmů, je vhodné úlohu upravit tak, aby řešení jednoznačné bylo. Mezi možnými modifikacemi, mezi které můžeme zařadit např. nenulové skladovací náklady, jsme se rozhodli pro volbu různých cen nakoupených růží. Prodavačka nakoupí první den růže za cenu c_1, druhý den si může dokoupit libovolné množství růží v závislosti na poptávce první den avšak za cenu c_2, která bude vyšší než původní cena c_1. Poté bude řešení úlohy jednoznačné.

Pro tuto úlohu předpokládejme, že průměrný počet zákazníků první den je 100 s větší směrodatnou odchylkou 20, tedy se pohybuje přibližně mezi 40 a 160. Druhý den je poptávka stabilnější a počet zákazníků se pohybuje mezi 60 a 120.

Pro tyto účely byly pseudonáhodně vygenerovány scénáře poptávky $\xi_{1,k}$ z normálního rozdělení $\mathcal{N}(100{,}400)$ a scénáře poptávky $\xi_{2,l}$ z normálního rozdělení $\mathcal{N}(90{,}100)$. Jelikož tyto scénáře mají představovat počet zákazníků, byla vygenerovaná čísla následně zaokrouhlená na celá. Omezení pro počet růží nakoupených první den bylo voleno 250 kusů, nákupní cena první den $c_1 = 40$, cena druhý den $c_2 = 45$ a prodejní cena $p = 60$.

Na následujících obrázcích si ilustrujeme řešení jednotlivých subproblémů (3.20) v závislosti na poptávce 1. den. Pro tyto účely bylo vygenerováno 250 scénářů poptávky 1. den a 250 scénářů poptávky 2. den. Optimální hodnota, spočítaná L-shaped algoritmem, je pro takto sestavenou úlohu $x^* = 168$.



(a) Závislost počtu růží na poptávce (b) Histogram optimálního počtu růží

Obrázek 3.3: Výsledky pro optimální počet růží nakoupených druhý den

Na grafu 3.3a vidíme závislost optimálního řešení z^* na poptávce první den. Pokud květinářka první den prodá všechny nakoupené růže, tj. nezůstane jí po 1. dni prodeje žádná růže k uskladnění, je optimální řešení počtů růží, které má další den nakoupit, rovno $(\frac{p-c_2}{p})-$kvantilu rozdělení poptávky druhý den. Tato rovnost vychází z odvození (3.17), které jsme udělali v úloze prodavače novin. V našem případě se jedná o stejné rozdělení jako u prodavače novin, nyní je však střední hodnota rovna 90. Posuneme-li výsledek lineárně o 10, dostáváme $z^* = 83$. Dále, pokud je např. poptávka první den $\xi_k = 100$, je $z_k^* = 83 - (168 - 100)$. Květinářka po prvním dni ke zbylým růžím dokoupí tolik růží, aby jich další den měla na stánku 83. Pokud je naopak první den poptávka nízká a květinářce zůstane 83 nebo více růží, potom žádnou další nekoupí. Závislost počtu růží na poptávce první den pak lze pro naši úlohu zapsat následujícím vztahem

$$z_k = \min(83 - (x^* - \xi_k)^+; 0).$$

Obrázek 3.3b zobrazuje histogram, tedy četnosti nakoupeného počtu růží druhý den. Jelikož jsou funkcí poptávky, která má rozdělení, jež vychází z normálního, je také rozdělení počtu nakoupený růží 2. den přibližné normální, useknuté v nule.

Pro tuto úlohu stejně jako pro úlohu prodavače novin jsou první dva předpoklady pro použití algoritmu SD splněny. Předpoklad A3, který požaduje nezápornost funkce $Q(x,\boldsymbol{\xi})$, je porušen. Z předchozí diskuze dostáváme, že účelová funkce ve druhém stupni je minimální, pokud je velká poptávka a květinářka

nakoupí v sobotu 83 růží. Hodnota $(c - p) * 83$ tedy funkci zdola omezí, neboť druhý člen účelové funkce je kladný. Tuto mez budeme uvažovat v případě, že také výběrový 1/4-kvantil vygenerovaných scénářů poptávky druhý den je roven 83. Z důvodů porušeného předpokladu A3 jsme také modifikovali ukončovací kritéria. Stejně jako v úloze prodavače novin jsme ve jmenovateli přidali absolutní hodnotu a použili kritérium (3.18).

Nejprve jsme L-shaped algoritmus a algoritmus SD porovnali pro pevný počet scénářů. Následující tabulka udává hodnoty získané algoritmem SD pro úlohu s 250 vygenerovanými poptávkami 1. den a 2. den. Pro tuto množinu scénářů, kterou jsme generovali z nového počátečního semínka, je optimální řešení získané L-shaped algoritmem $x^* = 169$ a zisk 3449.32. Jelikož algoritmy dekomponují úlohu pouze pro poptávku 1. den, rozhodli jsme se v tomto případě ponechat pevných 250 scénářů poptávky 2. den. Pro srovnání jsme použili různá ukončovací kritéria. Nejprve jsme z 250 scénářů poptávky 1. den generovali minimálně 100 scénářů a počet možných iterací omezili na 1000. Stejně jako pro úlohu prodavače novin jsme volili ustálenost účelové funkce přes 20 posledních hodnot kandidátů řešení, kteří v těchto iteracích museli být stejní. Odchylka byla volena 0.0001. Výběrový 1/4 kvantil popisovaných scénářů poptávky druhý den je 83, proto jsme volili dolní mez $(c - p) * 83$. Výsledky jsme označili FG1 (83). Dále jsme řešili úlohu s prudentním omezením $(c-p)*200$ a pro první ukončovací kritérium jsme tyto výsledky označili FG1 (200). Nakonec jsme pro dolní mez $(c - p) * 83$ změnili požadavek na minimální počet 500 iterací a ustálenost účelové funkce se kontrolovala přes posledních 100 nezměněných kandidátů řešení. Výsledky jsme označili FG2 (83). Algoritmus SD byl spuštěn vždy pro 30 různých počátečních semínek.

Tabulka 3.5: Výsledky pro úlohy květinářky-generování z 250 scénářů

	Počet růží 1. den	Zisk	Iterace
Průměr FG1 (83)	168.897	3446.428	238 (0)*
StD FG1 (83)	1.915	33.231	87
Průměr FG1 (200)	168.715	3445.364	213 (0)*
StD FG1 (200)	2.071	34.302	82
Průměr FG2 (83)	169.100	3447.746	686 (5)*
StD FG2 (83)	1.242	16.597	172

* Počet běhů z 30 celkem ukončených maximálním počtem iterací.

Z tabulky 3.5 opět vidíme, že s přísnějšími ukončovacími kritérii se průměrné optimální hodnoty získané SD algoritmem blíží ke správnému řešení. Srovnáním průměrných optimálních řešení a odhadů účelových funkcí pro dvě různé volby dolní meze dostáváme, že pro těsnější volbu dolní meze algoritmus SD odhaduje správné optimální hodnoty přesněji. Dále vidíme, že v pěti případech z 30 byla druhá ukončovací kritéria zastavená maximálním počtem iterací, tedy nefungovala pokaždé správně. V tomto případě je to způsobené především nízkým počtem maximálních iterací.

Pokud bychom napozorovali pouze 250 scénářů poptávky a její rozdělení

bychom neuměli odhadnout, je pro výpočet takové úlohy vhodnější použít L-shaped algoritmus. Ten je přesný a na rozdíl od algoritmu SD jej stačí spustit pouze jednou. Další jeho výhodou je, že na rozdíl od algoritmu SD poskytuje pro jednotlivé scénáře poptávky 1. den také hodnotu optimálního řešení počtu růží druhý den, jak bylo ukázáno na obrázku 3.3. L-shaped algoritmus vygeneruje 10 řezů optimality, tedy spočítá pouze 250*10 úloh lineárního programování.

Pokud zvolíme stejné požadavky na minimální (500) a maximální (1000) počet iterací při generování z 1500 scénářů, budou tyto požadavky velmi omezující a odhad optimálního řešení a hodnoty účelové funkce budou horší než v případě generování z 250 scénářů. S větším počtem scénářů by tak přirozeně měl růst požadavek na minimální počet iterací algoritmu SD, totéž platí pro volbu maximální hodnoty. S počtem scénářů lineárně roste náročnost výpočtu L-shaped algoritmem. Pro 15000 scénářů odpovídá jeho náročnost přibližně 50000 iteracím algoritmu SD. Pro takto velké množství scénářů jsme však na běžném notebooku nebyli schopni algoritmy porovnat, neboť úlohy byly výpočetně náročné. Z napozorovaných výsledků soudíme, že pokud chceme úlohu spočítat pro menší počet scénářů, je vhodnější použít přesný L-shaped algoritmus. Pro větší počet scénářů bychom naopak upřednostnili použití SD algoritmu. Jeho výpočet by měl být méně náročný a z teorie navíc očekáváme, že pro velký počet scénářů také nalezne odhad optimálního řešení s dostatečně malou odchylkou od správného řešení. Určit, pro kolik scénářů je vhodnější tuto úlohu spočítat L-shaped algoritmem a pro kolik naopak získá výhodu algoritmus SD by vyžadovalo efektivnější implementaci a také spouštění programů současně na více výkonnějších počítačích.

Pro pevnou množinu scénářů jsem se pokusili úlohu vyřešit také deterministickým ekvivalentem. Pro 250 scénářů bylo optimální řešení nalezeno rychleji než L-shaped algoritmem. Pro 5000 scénářů však tuto úlohu, na rozdíl od L-shaped algoritmu, z kapacitních důvodů neuměl vypočítat. Pro 5000x5000 scénářů se najednou hledá optimální řešení cca 25 milionů proměnných, což při sestavování úlohy bylo na běžném notebooku bez použití dekompozice nemožné. L-shaped algoritmus naopak pro stejných 5000x5000 scénářů nalezl optimum za necelých 9 hodin.

Nyní si ukážeme, jakých výsledků dosahují oba algoritmy, pokud generujeme scénáře náhodně ze známého rozdělení. Pro L-shaped algoritmus jsme generovali 100 scénářů pro poptávku první den a 100 scénářů pro poptávku druhý den. Algoritmus byl spuštěn pro 5 různých počátečních semínek určených pro poptávku první den a 5 dalších pro poptávku druhý den, celkově tedy pro 25 běhů.

Při použití algoritmu SD, označ. jako FGs, jsme generovali ze stejných počátečních semínek. Pro poptávku 2. den jsme použili pokaždé stejných 100 pevných scénářů jako pro L-shaped algoritmus. Scénáře poptávky 1. den jsme generovali ze zaokrouhleného $\mathcal{N}(100{,}400)$, přičemž jsme požadovali minimálně stejný počet scénářů (tj. minimálně 100 iterací). Algoritmus byl ukončen, pokud se hodnota účelové funkce v kandidátovi řešení od průměru spočítaného na posledních 20 neměnných kandidátech lišila s odchylkou menší než 0.0001. Jelikož pro 100 scénářů poptávky 2. den nemusí být výběrový 1/4-kvantil vždy roven 83, zvolili jsme pro jistotu dolní mez $(c - p) * 90$.

Tabulka 3.6: Výsledky Úlohy květinářky

	Počet růží 1. den	Zisk	Iterace
Průměr Benders	168.360	3452.561	10.080
StD Benders	2.464	53.205	0.572
Průměr FGs	169.080	3470.266	240
StD FGs	1.681	37.669	104

Z tabulky 3.6 vidíme, že v případě použití L-shaped algoritmu bylo v každém běhu vygenerováno přibližně 10 řezů optimality, tedy spočteno 1000 subproblémů. Směrodatná odchylka řešení je pro L-shaped algoritmus velká, neboť popsat zaokrouhlené rozdělení $\mathcal{N}$(100,400), resp. zaokrouhlené $\mathcal{N}$(90,100) pouze 100 scénáři není dostačující. Na základě různých množin scénářů se pak optimální řešení značně liší. Při daných ukončovacích kritériích je také řešení nalezené SD algoritmem citlivé na volbu počátečního semínka. Pro větší počet scénářů použitých v L-shaped algoritmu se směrodatná odchylka snižuje, stejně tak se snižuje pro algoritmus SD s přísnějším požadavkem na minimální počet proběhnutých iterací. Z tabulky 3.6 nemůžeme určit, který algoritmus je pro tuto úlohu lepší, k vhodnému porovnání obou algoritmů je potřeba, aby byl L-shaped algoritmus spuštěn několikrát pro velký počet scénářů a algoritmus SD s přísnými ukončovacími kritérii.

Pro jednoduchost jsme v této práci uvažovali ukončovací kritéria odvozená na základě asymptotických vlastností, která vyžadovala určit, co je „vhodný“ minimální počet iterací a co „dostatečný“ počet iterací, kdy má kandidát řešení zůstat neměnný. Tato kritéria mají výhodu jednoduché implementace, jak je však ukázáno v [8], statistická kritéria jsou mnohem vhodnějším nástrojem na ukončení SD algoritmu. Na základě statistických testů zjistí, jestli vygenerování nového scénáře ovlivní odhad optimálního řešení nalezený v dané iteraci. Důsledkem je, že algoritmus SD sám určí, co je dostatečný počet iterací k tomu, aby bylo nalezené řešení dostatečně blízko správnému řešení. Naopak pro L-shaped algoritmus je nutné určit počet scénářů, na kterých bude spočítán. Pokud zvolíme nedostatečný počet scénářů, bude odchylka řešení nezanedbatelná a bude nutné scénářů vygenerovat více, přičemž pro K scénářů se vypočítá přibližně 10*K úloh lineárního programování. Výhoda algoritmu SD by se proto měla nejvíc projevit v úlohách, kde je náhodný vektor spojitý nebo je jeho rozptyl stejně jako v této úloze velký.

Závěr

Celá práce se věnovala dvoustupňovým stochastickým úlohám s lineární účelovou funkcí a polyedrickou množinou omezení. V první kapitole jsme formulovali nejdůležitější tvrzení. Následně jsme podrobně rozebrali fungování L-shaped algoritmu a algoritmu stochastické dekompozice (SD), kterými lze tyto úlohy řešit. V praktické části jsme oba algoritmy testovali na třech příkladech.

V úloze hledání optimálního portfolia s minimální hodnotou CVaRu nalezl SD algoritmus při vhodně zvolených ukončovacích kritériích v každém běhu přesné řešení vypočtené L-shaped algoritmem. Hodnotu v riziku a účelovou funkci při použitých ukončovacích kritériích algoritmus SD pouze aproximoval. Na základě pozorovaných scénářů a nalezených optimálních vah lze však aproximovaná hodnota v riziku a optimální hodnota účelové funkce spočítat přesně. Již pro menší počet scénářů je rozdíl mezi náročností obou algoritmů znatelný, SD algoritmus dojde k optimálním váhám rychleji. Pro velký počet scénářů (např. dvacetiletou historii denních výnosů), kdy časová náročnost výpočtu značně poroste, bychom proto doporučili použít algoritmus SD. Pokud bychom byli schopni rozdělení výnosů odhadnout, potom při předpokladu spojitého rozdělení má algoritmus SD proti L-shaped algoritmu nespornou výhodu. L-shaped algoritmus by musel několikrát vygenerovat velké množství scénářů (např. 50000), aby dobře aproximoval odhadnuté rozdělení a průměrné řešení těchto aproximativních úloh se dalo považovat za dostatečně blízké správnému optimu. Algoritmus SD umí tuto úlohu řešit stejně dobře, ať generuje scénáře ze spojitého nebo diskrétního rozdělení.

V úloze prodavače novin jsme kromě srovnání algoritmů ilustrovali také stabilitu řešení a zachování pouze aktivních řezů v případě použití regularizovaného algoritmu SD. Poprvé jsme diskutovali modifikaci algoritmu SD, pokud není splněn předpoklad na nezápornou účelovou funkci ve druhém stupni. Navrhli jsme dva možné přístupy, kterými lze porušený předpoklad napravit. Při známém rozdělení jsme pro L-shaped algoritmus 30x generovali 100 scénářů a pro algoritmy SD jsme 30x spustili generátor se stejnými počátečními semínky. Výsledky obou algoritmů byly srovnatelné. L-shaped algoritmus se v průměru od teoreticky odvozeného správného řešení lišil méně než algoritmus SD, jehož nalezená řešení však vykazovala větší stabilitu. Je třeba říct, že stabilita L-shaped algoritmu je závislá na počtu scénářů, který bude vygenerován. Pro větší počet scénářů budou nalezená řešení stabilnější avšak časová náročnost lineárně stoupne.

Problémem květinářky jsme uzavřeli aplikační část této práce. Opět jsme řešili volbu dolní meze při porušeném předpokladu nezápornosti účelových funkcí ve druhém stupni pro jednotlivé poptávky první den. Ukázali jsme, že v případě striktního omezení maximálního počtu iterací může nepřesná volba meze ovlivnit řešení nalezené algoritmem SD. Pro přesněji odvozenou dolní mez dával algoritmus SD lepší odhad optimálního řešení. Pokud bychom počítali úlohu pro rozumný počet scénářů, je vhodnější použít přesný L-shaped, neboť algoritmus SD pro malý počet iterací řešení pouze aproximuje. Další výhodou L-shaped algoritmu je, že poskytuje informaci o optimálním počtu růží nakoupených druhý den v závislosti na poptávce první den. Pro větší počet scénářů bychom naopak upřednostnili použití SD algoritmu. Jeho výpočet by totiž měl být méně náročný a z teorie navíc očekáváme, že pro velký počet scénářů také nalezne odhad op-

timálního řešení s dostatečně malou odchylkou od správného řešení. Určit, pro kolik scénářů je vhodnější tuto úlohu (stejně jako úlohu prodavače novin) spočítat L-shaped algoritmem a pro kolik naopak získá výhodu algoritmus SD, by vyžadovalo efektivnější implementaci a také spouštění programů současně na více výkonnějších počítačích.

Možným rozšířením práce je tak testovat algoritmy na složitějších a větších úlohách, pro které jsou primárně určeny. Kromě základních algoritmů by se mohlo jejich fungování otestovat také pro jejich regularizované verze. Pro algoritmus SD je navíc vhodné naprogramovat sice složitější avšak exaktnější ukončovací kritéria založená na statistických testech, která jsou uvedená v [8]. Algoritmus SD totiž může dojít ke špatným výsledkům z důvodů nevhodně nastavených ukončovacích kritérií, jak jsme si v této práci také ukázali.

Seznam použité literatury

[1] Portfolios formed on size. `http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html#Research`.

[2] L. Adam. Nelinearity v úlohách stochastického programování: aplikace na řízení rizika. Diplomová práce, Univerzita Karlova v Praze, Praha, 2011.

[3] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[4] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, New York, NY, USA, 1997.

[5] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[6] J. Dupačová and P. Lachout. *Úvod do optimalizace*. První vydání. MATFYZ-PRESS, Praha, 2011.

[7] J. L. Higle and S. Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of operations research*, 16(3):650–669, 1991.

[8] J. L. Higle and S. Sen. *Stochastic decomposition: A Statistical Method for Large Scale Stochastic Linear Programming*. Kluwer Academic Publishers, 1996.

[9] J. L. Higle and S. Sen. Statistical approximations for stochastic linear programming problems. *Annals of Operations Research*, 85(0), 1999.

[10] P. Kall. *Stochastic Linear Programming*. Springer, Berlin, 1976.

[11] P. Kall and J. Mayer. *Stochastic Linear Programming*. Springer, New York, NY, 2005.

[12] P. Kall and S. W. Wallace. *Stochastic Programming*. John Wiley & Sons, 1994.

[13] A. Künzi-Bay and J. Mayer. Computational aspects of minimizing conditional value-at-risk. *Computational Management Science*, 3(1):3–27, 2006.

[14] P. Lachout. Teorie optimalizace. Pracovní text k přednášce Teorie optimalizace na MFF UK. Dostupné z `http://www.karlin.mff.cuni.cz/~lachout/Vyuka/T-Optima/161127-OptimizationTheory.pdf`.

[15] H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[16] R. T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance*, 26(7):1443–1471, 2002.

[17] A. Ruszczyński and A. Shapiro. Optimality and duality in stochastic programming. *Handbooks in Operations Research and Management Science*, 10:65–139, 2003.

[18] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, Chichester, 1986.

[19] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming.* SIAM, Philadelphia, PA, USA, 2009.

[20] R. M. Van Slyke and R. 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.

[21] R. J.-B. Wets. Stochastic multipliers, induced feasibility and nonanticipativity in stochastic programming. *Stochastic programming, Proc. int. Conf, Oxford 1974, 137-146*, 1980.

Přílohy

Základní definice a věty

Níže uvedené definice a věty lze nalézt v [6], [14], [17] a [18].

Definice P. 1. *Nechť máme úlohu LP ve standardním tvaru*

$$\min\{c^T x : Ax = b, x \geq 0\},\ \textit{kde } b \in \mathbb{R}^m, c \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}. \tag{P.1}$$

Uvažujme, že matice $A \in \mathbb{R}^{m\times n}$ má plnou řádkovou hodnost. Vyberme nějakou množinu indexů proměnných $L \subset \{1,2,\ldots,n\}$ tak, aby obsahovala právě m různých položek. Vezměme příslušné sloupce matice A a sestavme z nich čtvercovou matici $A_L = (A_{j,i}, j \in \{1,2,\ldots,m\}, i \in L)$.

- *Pokud je matice A_L regulární, pak říkáme, že L je* ***báze*** *úlohy (P.1).*

- *Když L je báze úlohy (P.1), pak k ní přísluší $x(L) \in \mathbb{R}^n$, které je jednoznačně určené podmínkami $Ax(L) = b$ a $x(L)_i = 0$ pro všechna $i \notin L$. Bod $x(L)$ nazýváme* ***bazické řešení*** *příslušné k bázi L.*

- *Nechť L je báze úlohy (P.1) a bazické řešení $x(L)$ je nezáporné. Pak mluvíme o* ***přípustné bázi****.*

- *Nechť L je báze úlohy (P.1) a příslušné bazické řešení $x(L)$ je optimálním řešením úlohy (P.1). Pak mluvíme o* ***optimální bázi****.*

- *Říkáme, že x je* ***bazické řešení*** *úlohy (P.1), jestliže existuje báze L tak, že $x = x(L)$.*

- *Říkáme, že x je* ***přípustné bazické řešení*** *úlohy (P.1), jestliže existuje přípustná báze L tak, že $x = x(L)$.*

- *Říkáme, že x je* ***optimální bazické řešení*** *úlohy (P.1), jestliže existuje optimální báze L tak, že $x = x(L)$.*

Lemma P. 1. *Nechť $L \subset \{1,2,\ldots,n\}$ je báze úlohy (P.1). Potom příslušné bazické řešení je určeno jednoznačně a je tvaru $x(L)_L = (A_L)^{-1}b, x(L)_i = 0$ pro $i \notin L$.*

Z další teorie LP a ze Základní věty LP pak vychází, že má-li úloha (P.1) optimální řešení, pak je vždy můžeme najít mezi přípustnými bazickými řešeními. Přípustná bazická řešení jsou navíc krajními body množiny přípustných řešení.

Definice P. 2. *Pro úlohu (P.1) a bázi L definujeme $y(L) \in \mathbb{R}^m$, které je jednoznačně určené podmínkou $(A_L)^T y(L) = c_L$. Tedy $y(L) = \left((A_L)^T\right)^{-1} c_L$. Vektor $y(L)$ nazýváme* ***duální bazické řešení*** *příslušné k bázi L.*

Definice P. 3. *Nechť $D \subset \mathbb{R}^n$ je množina a $f : D \to \mathbb{R}$ je konvexní funkce. Řekneme, že f má v bodě $x \in D$ subgradient $\eta \in \mathbb{R}^n$, jestliže platí*

$$f(y) - f(x) \geq \langle \eta, y - x \rangle \text{ pro každé } y \in D.$$

Množinu všech subgradientů v bodě x budeme nazývat subdiferenciál funkce f v bodě x a budeme ji značit $\partial f(x)$.

Věta P. 2. *Nechť $D \subset \mathbb{R}^n, x^* \in D$ a $f : D \to \mathbb{R}$ je konvexní funkce. Pak x^* je bod globálního minima funkce f na množině D tehdy a jen tehdy, když $0 \in \partial f(x^*)$.*

Věta P. 3 (Oddělitelnost bodu a konvexní množiny)**.** *Nechť $K \subset \mathbb{R}^n, K \neq \emptyset$ je konvexní množina a $x \notin \text{clo}(K)$. Pak existuje $\gamma \in \mathbb{R}^n, \gamma \neq 0$ tak, že*

$$\inf\{\langle \gamma, y \rangle : y \in K\} = \min\{\langle \gamma, y \rangle : y \in \text{clo}(K)\} > \langle \gamma, x \rangle.$$

Věta P. 4 (O opěrné nadrovině)**.** *Nechť $K \subset \mathbb{R}^n, K \neq \emptyset$ je konvexní množina a $y \in \partial(K)$. Pak existuje $\gamma \in \mathbb{R}^n, \gamma \neq 0$ tak, že $\inf\{\langle \gamma, x \rangle : x \in K\} = \langle \gamma, y \rangle$.*

Definice P. 4. *Když $D \subset \mathbb{R}^n$, pak symbolem* pos D *označujeme nejmenší konvexní kužel obsahující množinu D. Množinu* pos D *nazýváme nezáporný obal množiny D a můžeme ji zapsat následujícím způsobem*

$$\text{pos } D = \{\sum_{s \in I} \lambda(s) s : \lambda(s) \geq 0 \; \forall \, s \in I, I \subset D \text{ konečná}\}.$$

Definice P. 5. *Pro rozšířenou reálnou funkci $f : R^n \to \overline{\mathbb{R}}$ definujeme její doménu*

$$\text{Dom } f = \{x : f(x) < +\infty, x \in \mathbb{R}^n\}.$$

Budeme říkat, že f je vlastní, jestliže Dom $f \neq 0$ *a $f(x) > -\infty$ pro každé $x \in \mathbb{R}^n$.*

Definice P. 6. *Pro funkci $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ definujeme její epigraf*

$$\text{epi}(f) = \{(x, \alpha) : f(x) \leq \alpha, x \in \mathbb{R}^n, \alpha \in \mathbb{R}\}.$$

Definice P. 7. *Rozšířená reálná funkce $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ je zdola polospojitá funkce (ozn. lsc=lower-semicontinuous) v bodě x_0, jestliže*

$$\liminf_{x \to x_0} f(x) \geq f(x_0).$$

Řekneme, že funkce f je zdola polospojitá, pokud je lsc v každém bodě $x \in \mathbb{R}^n$.

Lze ukázat, že funkce f je lsc právě tehdy, pokud její epigraf je uzavřená podmnožina $\mathbb{R}^n \times \mathbb{R}$.

Věta P. 5 (Moreau-Rockefellar)**.** *Nechť pro $i = 1, \ldots, m$ jsou $f_i : \mathbb{R}^n \to \overline{\mathbb{R}}$ konvexní vlastní funkce a $f = f_1 + \ldots + f_m$. Pokud pro x_0 jsou $f_i(x_0)$ konečné, pak*

$$\partial f_1(x_0) + \ldots + \partial f_m(x_0) \subset \partial f(x_0).$$

Rovnost nastane, pokud je splněna alespoň jedna z následujících podmínek

1. $\bigcap_{i=1}^m \text{rint}(\text{Dom } f_i)$ *je neprázdná,*

2. *pro nějaké k jsou $f_1, \ldots, f_k$ polyedrické funkce a průnik množin $\bigcap_{i=1}^{k} \text{Dom } f_i$ a $\bigcap_{i=k+1}^{m} \text{rint}(\text{Dom } f_i)$ je neprázdný,*

3. *existuje $\overline{x} \in \text{Dom } f_m$ takový, že pro $\forall\, i = 1, \ldots, m-1$ platí $\overline{x} \in \text{int}(\text{Dom } f_i)$.*

Speciálně, pokud jsou všechny funkce $f_1, \ldots, f_m$ polyedrické, potom rovnost platí automaticky i bez splnění dalších podmínek.

Definice P. 8. *Pro $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ definujeme sdruženou funkci jako*

$$f^*(z) = \sup_{x \in \mathbb{R}^n} \{z^T x - f(x)\}.$$

Věta P. 6 (Fenchel-Moreau)**.** *Nechť $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ je vlastní konvexní funkce. Potom $f^{**} = \text{lsc } f$, kde $\text{lsc } f$ je největší zdola polospojitá funkce, která je menší než nebo rovna f.*

Pro vlastní a konvexní funkci f platí rovnost $f = f^{**}$, právě když je f zdola polospojitá.

Důsledek P. 7. Nechť f je konvexní, vlastní a zdola polospojitá funkce. Pokud $f(x) < +\infty$, pak platí

$$\partial f(x) = \arg\max_z \{z^T x - f^*(z)\}.$$

Definice P. 9. *Řekneme, že matice A je totálně unimodulární, pokud každý její subdeterminant nabývá hodnoty 0, +1 nebo -1.*

Konkrétně, každý prvek totálně unimodulární matice je 0, +1 nebo -1.