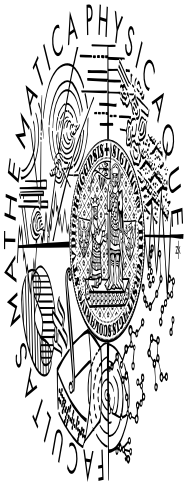

MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Martin Minárik

Distribuční modely šperků

Vedoucí bakalářské práce: doc. RNDr. Ing. Miloš Kopa, Ph.D.

Studijní program: Finanční matematika

Praha 2024

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne a výhradne s použitím citovaných prameňov, literatúry a ďalších odborných zdrojov. Táto práca nebola využita k získaniu iného alebo rovnakého titulu.

Berem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona v platnom znení, hlavne skutočnosť, že Univerzita Karlova má právo na uzavretie licenčnej zmluvy o použití tejto práce ako školského diela podľa §60 odst. 1 autorského zákona.

V ..... dňa ....

Podpis autora

Na tomto mieste by som rád poďakoval vedúcemu práce doc. RNDr. Ing. Milošovi Kopovi, Ph.D., za jeho ochotu stať sa vedúcim mojej práce a čas, ktorý mi pri jej vytváraní venoval. Jeho cenné rady a pripomienky mi pomohli usmerniť množstvo rozlietaných myšlienok, ktoré vďaka nemu našli tú správnu formu a dostali sa v uhľadenej forme do tejto práce. Obrovská vďaka patrí taktiež mojej úžasnej priateľke, ktorá bola a stále je mojou veľkou podporou. Bez jej podpory, rozveselovania, lásky a trpezlivosti si už moje prežívanie na tejto planéte viem ťažko predstaviť. Bez nej by som túto prácu takto hľadko nedokončil. Nakoniec by som chcel poďakovať svojej rodine a priateľom, ktorí stáli pri mne a vrúcne ma podporovali.

Název práce: Distribuční modely šperků

Autor: Martin Minárik

Katedra: Katedra pravděpodobnosti a matematické statistiky

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

Abstrakt: Táto práca rieši problém distribúcie šperkov pomocou teórie celočiselného programovania. V teoretickej časti sú definované základné pojmy a predstavená základná teoria celočísleného programovania. Teoretická časť sa taktiež zaoberá detailným popisom Algoritmu vetvenia a medzí (Branch and bound algorithm). Praktická časť rieši reálny problém z praxe. Cielom tejto časti je optimálne rozdeliť šperky vybranej značky na butiky danej značky. Je v nej predstavená tvorba modelu celočísleného programovania riešiaceho danú problematiku pomocou viacerých kritérií. Následne sa praktická časť venuje popisu vstupných dát a rozboru výsledkov distribúcie vstupných dát pomocou navrnutého modelu pre 3 kombinácie vstupných parametrov, ktoré jemne pozmenia navrhnutý model.

Klíčová slova: distribuční model, optimalizace, celočíselné programování

Title: Distribution models for jewelleries

Author: Martin Minárik

Department: Department of probability and mathematical statistics

Supervisor: doc. RNDr. Ing. Miloš Kopa, Ph.D., Department of probability and mathematical statistics

Abstract: This work solves jewelry distribution problem using the theory of integer programming. In theoretical part we present basic definitions and fundamental theory of integer programming. Theoretical part also presents detailed description of the Branch and bound algorithm. Practical part tackles a real world problem. The goal of the practical part is to optimaly distribute jewelry of the chosen brand to their butiques. Firstly, we present the creation of the integer programming model that solves this problem considering multiple criteria. Secondly, in practical part we present the description of the input data and also the analisis of the result of distribution of the input data using proposed model for 3 diferent combinations of input parameters.

Keywords: distribution models, optimization, integer programming

Obsah

Úvod

.

Zamestnanci na backofficoch spoločností produkujúcich a predávajúcich rozličný tovar sa pravidelne stretávajú s problémom ideálneho rozdelenia tovaru na predajne. Veľké spoločnosti disponujú algoritmami na optimálne rozdeľovanie tovaru, no u menších spoločností tomu tak spravidla byť nemusí. Tovar v menších spoločnostiach, teda častokrát rozdeľujú na predajne zamestanci na backofficoch na základe intuície, požiadavok z jednotlivých predajní a dlhoročných skúseností. Tento prístup nie je ideálny, pretože s rastom spoločnosti rapídne klesá schopnosť zamestnancov zvážiť pri efektívnom rozdeľovaní tovaru všetky faktory.
Celočíslelná optimalizácia je oblasťou matematiky, ktorá sa využíva pri riešení problémov optimálneho priraďovania. V tejto práci budeme celočíselnú optimalizáciu taktiež nazývať aj celočísleným programovaním. Celočíslené programovanie sa uplatňuje v mnohých odvetviach. Rieši napríklad logistické problémy, problémy rozvrhovania pracovných zmien, tvorbu školských rozvrhov atď. V tejto práci budeme uplatňovať poznatky z oblasti celočíselnej optimalizácie. Hlavným zdrojom informácií pre teoretickú časť bude kniha Wolsey (1998).
V teoretickej časti práce definujeme úlohu lineárneho programovania a s ňou aj úlohu celočísleného programovania. Uvedieme ich základné vlastnosti. Na záver kapitoly detailne popíšeme Algoritmus vetvenia a medzí, ktorý je vhodný na riešenie úloh celočíselného programovania.
Praktická časť bude riešiť reálny problém rozdelenia šperkov značky ALOVE, kde reálne šperky rozdelíme na butiky patriace danej značke. V tejto časti navrhneme distribučný model a jeho technickú implementáciu. Taktiež zhodnotíme výsledky distribúcie šperkov na butiky pomocou navrhnutého modelu a prediskutujeme nedostatky, a vylepšenia modelu.

1. Celočíselné a lineárne programovanie

V teoretickej kapitole sformulujeme optimalizačné úlohy, popíšeme ich vlastnosti a vzájomné prepojenia. Následne vybudujeme základnú teoriu pre popis **Algoritmu vetvenia a medzí**, pomocou ktorého môžeme riešiť úlohy celočíselného programovania.

Značenie platné pre celú prácu:

- $\boldsymbol{A}$ je matica, pre ktorú platí, že $\boldsymbol{A} \in \mathbb{R}^{m \times n}$
- $\boldsymbol{c}$ je stĺpcový vektor, pre ktorý platí, že $\boldsymbol{c} \in \mathbb{R}^n$
- $\boldsymbol{b}$ je stĺpcový vektor, pre ktorý platí, že $\boldsymbol{b} \in \mathbb{R}^m$

1.1 Predstavenie modelov

V tejto podkapitole predstavíme rôzne optimalizačné úlohy. Pre niektoré z nich budeme následne budovať základnú teóriu ich riešenia a vzájomného prepojenia.

Definícia 1 (LP)**.** *(Wolsey (1998)(kapitola 1.2))*
Pod pojmom úloha ***lineárneho programovania*** *rozumieme úlohu, nájsť vektor $\boldsymbol{x} \in \mathbb{R}^n$, ktorý rieši optimalizačný problém:*

$$\begin{aligned} \max \ & \boldsymbol{c}^T \boldsymbol{x} \\ \text{z.p. } & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \\ & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

Lineárnu funkciu $\boldsymbol{c}^T\boldsymbol{x}$ budeme nazývať ***účelová funkcia****, vektor $\boldsymbol{x}$ budeme nazývať* ***vektor rozhodovacích premenných****, vektor $\boldsymbol{c}$ budeme nazývať* ***vektor cien*** *a sústavu rovníc $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ budeme nazývať* ***obmedzujúce podmienky***

Následne pridáme obmedzenie, že niektoré rozhodovacie premenné v účelovej funkcii musia nadobúdať celočíselné hodnoty.

Definícia 2 (MIP)**.** *(Wolsey (1998)(kapitola 1.2)) Nech $\boldsymbol{G} \in \mathbb{R}^{m \times p}$ a $\boldsymbol{h} \in \mathbb{R}^p$. Potom pod úlohou* ***zmiešaného programovania*** *(anglicky Mixed integer programing) myslíme úlohu, nájsť vektory $\boldsymbol{x}$ a $\boldsymbol{y}$, ktoré riešia optimalizačný problém:*

$$\begin{aligned} \max \ & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{h}^T\boldsymbol{y} \\ \text{z.p. } & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{G}\boldsymbol{y} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \\ & \boldsymbol{y} \geq \boldsymbol{0} \\ & \boldsymbol{x} \in \mathbb{Z}^n \\ & \boldsymbol{y} \in \mathbb{R}^p. \end{aligned}$$

Ak pridáme ďalšie obmedzenie, že rozhodovacie premenné môžu nadobúdať len celočíselné hodnoty, dostaneme celočíselné programovanie (Integer programing)

Definícia 3 (IP)**.** *(Wolsey (1998)(kapitola 1.2)) Pod úlohou* ***celočísleného programovania*** *myslíme úlohu, nájsť vektor* $\boldsymbol{x}$*, ktorý rieši optimalizačný problém:*

$$\begin{aligned} \max\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{z.p. } & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \\ & \boldsymbol{x} \in \mathbb{Z}^n. \end{aligned}$$

Predstavíme ešte úlohu binárneho programovania, kde hodnoty rozhodovacích premenných môžu nadobúdať len hodnoty 0 alebo 1.

Definícia 4 (BIP)**.** *(Wolsey (1998)(kapitola 1.2)) Pod názvom úloha* ***binárneho programovania*** *budeme myslieť riešenie optimalizačnej úlohy:*

$$\begin{aligned} \max\ & \boldsymbol{c}^T\boldsymbol{x} \\ \text{z.p. } & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \\ & \boldsymbol{x} \in \{0,1\}^n. \end{aligned}$$

Definícia 5. *Pod pojmom* ***množina prípustných riešení*** *(S_α) budeme myslieť množinu hodnôt* $\boldsymbol{x}$ *vyhovujúcich* ***obmedzujúcim podmienkam*** *daného problému, zároveň patriacich do množiny prípustných hodnôt* ***rozhodovacích premenných****.* $\alpha \in \{LP, MIP, IP, BIP\}$

Poznámka. ak $\alpha = LP$ potom $S_{LP} = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$.

1.2 Vlastnosti a optimalita LP

V tejto podkapitole si ukážeme základné vlastnosti a charakteristiky optimality pre lineárne programovanie. Budeme čerpať z Dupačová (2011), Schrijver (1986) a Papadimitriou a Steiglitz (1998).

Veta 1 (Základná veta lineárneho programovania)**.** *(Dupačová (2011))*
Pri riešení úlohy ***Lineárneho programovania*** *sa stretneme len s nasledujúcimi prípadmi:*

- $S_{LP} = \emptyset$
- $S_{LP} \neq \emptyset$ & $sup_{\boldsymbol{x} \in S_{LP}} \boldsymbol{c}^T\boldsymbol{x} = \infty$
- $S^*_{LP} \neq \emptyset$.

Znakom S^*_{LP} *označujeme množinu optimálnych riešení* $\boldsymbol{x}$.

Dôkaz. Viď Dupačová (2011).

□

Vetou 1 sme popísali základné správanie lineárneho programovania. Ako ďalšie uvedieme Farkasovo lema, ktoré nám dá postačujúcu podmienku pre zistenie či má úloha prípustné riešenie alebo nie. Následne uvedieme vetu o dualite spolu s podmienkami komplementarity, ktoré nám ukážu či je nájdené riešenie optimálne.

Lema 2 (Farkasoho lema)**.** *(Schrijver (1986)(str. 90))*
Nech $\boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{b} \in \mathbb{R}^n, \boldsymbol{y} \in \mathbb{R}^m, \boldsymbol{x} \in \mathbb{R}^n$. *Potom systém* $\boldsymbol{Ax} \leq \boldsymbol{b}$ *má riešenie* $\boldsymbol{x} \geq \boldsymbol{0}$*, vtedy a len vtedy ak platí, že* $\boldsymbol{y}^T\boldsymbol{b} \geq 0$ *pre každý vektor* $\boldsymbol{y} \geq \boldsymbol{0}$ *spĺňujúci* $\boldsymbol{yA} \geq \boldsymbol{0}$.

Dôkaz. Viď Schrijver (1986)(str. 90).

□

Veta 3 (Veta o dualite)**.** *(Dupačová (2011))*
Označme $P = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{Ax} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ *a* $Q = \{y \in \mathbb{R}^m : \boldsymbol{A}^T\boldsymbol{y} \geq \boldsymbol{c}, \boldsymbol{y} \geq \boldsymbol{0}\}$.

- *Ak* $\boldsymbol{x} \in P$ *a* $\boldsymbol{y} \in Q$*, potom* $\boldsymbol{c}^T\boldsymbol{x} \leq \boldsymbol{b}^T\boldsymbol{y}$.
- *Ak* $P \neq \emptyset$ *a* $Q \neq \emptyset$*, potom*

$$\max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in P\} = \min\{\boldsymbol{b}^T\boldsymbol{y} : \boldsymbol{y} \in Q\},$$

kde $\max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in P\}$ *je úloha primárna a* $\min\{\boldsymbol{b}^T\boldsymbol{y} : \boldsymbol{y} \in Q\}$ *je ku nej úloha duálna. Úlohy spolu tvoria primárno duálny pár.*

Dôkaz. Viď Dupačová (2011).

□

Veta 4 (Podmienky komplementarity)**.** *(Papadimitriou a Steiglitz (1998)(str. 72))*

Nech $\boldsymbol{x}^* \in \mathbb{R}^n$ *je prípustné riešenie úlohy* $\max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{Ax} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ *a* $\boldsymbol{y}^* \in \mathbb{R}^m$ *je prípustné riešenie úlohy* $\min\{\boldsymbol{b}^T\boldsymbol{y} : \boldsymbol{A}^T\boldsymbol{y} \geq \boldsymbol{c}, \boldsymbol{y} \geq \boldsymbol{0}\}$. *Potom* $\boldsymbol{x}^*$, $\boldsymbol{y}^*$ *sú optimálne riešenia pre uvedené problémy práve vtedy, keď sú splnené podmienky komplementarity:*

$$\forall i \in \{1,...,m\} : y_i^* = 0 \vee \sum_{j=1}^{n} a_{i,j}x_j^* = b_i,$$
$$\forall j \in \{1,...,n\} : x_j^* = 0 \vee \sum_{i=1}^{m} a_{i,j}y_i^* = c_j.$$

Dôkaz. Viď Papadimitriou a Steiglitz (1998)(str. 72).

□

1.3 Rozdiel medzi LP a IP

V lineárnom programovaní nám Veta 1 vraví, že ak máme $S_{LP} \neq \emptyset$ & $sup_{\boldsymbol{x} \in S_{LP}} \boldsymbol{c}^T \boldsymbol{x} < \infty$, vieme nájsť optimálne riešenie. Bohužiaľ toto neplatí pre celočíselné programovanie. V nasledujúcom príklade uvedieme problém, ktorý potvrdzuje, že Veta 1 neplatí pre celočíselné programovanie.

Príklad. Riešme nasledujúcu optimalizačnú úlohu:

$$\begin{aligned} \max \ & -\sqrt{11}x + y \\ \text{z.p.} \ & -\sqrt{11}x + y \leq 0 \\ & x \geq 2 \\ & y \geq 1 \\ & x, y \in \mathbb{Z}^1. \end{aligned}$$

Po preskúmaní problému vidíme, že množina $S_{IP} \neq \emptyset$, a zároveň maximálna hodnota účelovej funkcie je obmedzená zhora viď $(-\sqrt{11}x+y \leq 0)$, teda ak by sme uvažovali, že x,$y \in \mathbb{R}^1$ namiesto x,$y \in \mathbb{Z}^1$, vedeli by sme nájsť optimum. Upravme teraz druhú podmienku na $(\frac{y}{x} \leq \sqrt{11})$. Po úprave vidíme, že optimum by bolo v bode $\sqrt{11}$, lenže $\sqrt{11}$ je iracionálne číslo, ktoré nevieme zapísať pomocou zlomku. Vieme sa ku nemu ľubovolne priblížiť, no nikdy ho nedosiahneme. Optimálne riešenie, teda neexistuje.

Poznámka. Z minulého príkladu si môžeme všimnúť, že ak by sme používali iteračný algoritmus na riešenie problému, bežal by do nekonečna bez nájdenia optima. Preto budeme potrebovať vylepšiť naše metódy pomocou zavedenia určitých medzí, viď nasledujúca podkapitola.

1.4 Vlastnosti a riešenie IP

V nasledujúcej podkapitole uvedieme vlastnosti celočíselného programovania a ukážeme ako nám znalosti o optimálnych riešeniach lineárneho programovania pomôžu nájsť riešenia celočíselného programovania ľubovoľne blízke optimálnemu.

Poznámka. Ako sme videli v príklade, v minulej podkapitole, pri riešení úloh celočíselného programovania sa môžeme dostať do situácie, že nevieme nájsť optimálne riešenie. Stáva sa to z viacerých dôvodov. Niektoré z nich sú (ako v Príklade 1.3) neexistencia optimálneho riešenia alebo naša neschopnosť nájsť optimálne riešenie v dostupnom čase. Z hľadiska časovej náročnosti ani s najlepšou technikou niekedy nie je možné nájdenie optimálneho riešenia úloh IP. Preto volíme alternatívny prístup. Snažíme sa nájsť riešenie blízke optimálnemu, s nami zvolenou toleranciou.

Majme problém $Z = \max\{\boldsymbol{c}^T \boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b},\ \boldsymbol{x} \in \mathbb{Z}^n,\ \boldsymbol{x} \geq \boldsymbol{0}\}$.

Hlavnou myšlienkou pri riešení IP bude nájsť *dolnú medzu* $\underline{Z}$, pre ktorú bude platiť $\underline{Z} \leq Z$ a *hornú medzu* $\bar{Z}$, pre ktorú bude platiť $\bar{Z} \geq Z$. Budeme sa snažiť vytvoriť algoritmus, čo dokáže nájsť klesajúcu postupnosť $\bar{Z}_1 \geq \bar{Z}_2 \geq ... \geq \bar{Z}_s \geq Z$ horných medzí a rastúcu postupnosť $\underline{Z}_1 \leq \underline{Z}_2 \leq ... \leq \underline{Z}_t \leq Z$ dolných medzí. Algoritmus sa zastaví, keď $\bar{Z}_s - \underline{Z}_t \leq \epsilon$. Konštanta ϵ predstavuje maximálnu odchylku od optimálneho riešenia, ktorú sme ochotní akceptovať.

1.4.1 Dolná medza

Každé prípustné riešenie $\boldsymbol{x} \in S_{IP}$ nám poskytne dolnú medzu $\underline{Z} = \boldsymbol{c}^T\boldsymbol{x} \leq Z$. Pre niektoré problémy bude jednoduché násť dolnú medzu na prvý pohľad, no pre niektoré nie. Pri úlohách, kde prípustné riešenie nie je zjavné si môžeme pomôcť napríklad *reštrikciou* problému definovanou naskledovne:

Definícia 6. *(Wolsey (1998))*
Problém $Z^{RE} = \max\{f(\boldsymbol{x}) : \boldsymbol{x} \in T \subseteq \mathbb{Z}^n\}$ *je reštrikciou* $Z = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S_{IP} \subseteq \mathbb{Z}^n\}$ *ak:*

- $T \subseteq S_{IP}$ *a*
- $f(\boldsymbol{x}) \leq \boldsymbol{c}^T\boldsymbol{x}$ *pre* $\forall \boldsymbol{x} \in S_{IP}$.

Nájdením optimálneho riešenia reštrikcie problému nájdeme prípustné riešenie pôvodného problému.

Reštrikciu problému vytvárame napríklad zafixovaním niektorých rozhodovacích premenných alebo pridaním ďalších podmiedok zmenšujúcich množinu prípustných riešení. Postupujeme tak, aby sme vytvorili jednoduchší problém.

Príklad reštrikcie vieme nájsť vo Wolsey (1998)(str. 35)(Príklad 2.5).

1.4.2 Horná medza

Pri hľadaní dolnej medze sme využívali reštrikciu problému. Pri hľadaní hornej medze využijeme *relaxáciu*. Základnou myšlienkou bude zjednodušit daný optimalizačný problém, ale stále zanechať vlastnosť aby jeho optimálna hodnota bola aspoň taká veľká ako Z.

Definícia 7. *(Wolsey (1998)(str. 26))*
Problém $Z^{RP} = \max\{f(\boldsymbol{x}) : \boldsymbol{x} \in T \subseteq \mathbb{R}^n\}$ *je relaxáciou problému* $Z = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S_{IP}\}$ *ak:*

- $S_{IP} \subseteq T$, *a*
- $f(\boldsymbol{x}) \geq \boldsymbol{c}^T\boldsymbol{x}$ *pre* $\forall \boldsymbol{x} \in S_{IP}$.

Dôkaz. Viď Wolsey (1998)(str. 26).

□

Otázkou už len zostáva, ako skonštruovať takéto relaxácie, a aké pekné vlastnosti nám poskytnú. Na to odpovie nasledujúca veta a ďalšie definície.

Veta 5 (vlastnosti relaxácie)**.** *(Wolsey (1998)(str. 28))*

- Ak *relaxácia* nemá prípustné riešenie, potom ani pôvodný problém nemá prípustné riešenie.
- Nech $\boldsymbol{x}^*$ je optimálne riešenie *relaxácie*. Ak $\boldsymbol{x}^* \in S_{IP}$ a $f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x}$, potom $\boldsymbol{x}^*$ je optimálne riešenie pôvodného problému IP.

Dôkaz. Viď Wolsey (1998)(str. 28). □

Definícia 8 (Lineárna relaxácia)**.** *(Wolsey (1998))*
Majme úlohu celočíselného programovania $Z = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S_{IP}\}$*, potom pod pojmom lineárna relaxácia pôvodnej úlohy označíme* $Z^{LP} = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0}\}$.

Je to vlastne rovnaká úloha až na to, že $\boldsymbol{x} \in \mathbb{R}^n$ namiesto $\boldsymbol{x} \in \mathbb{Z}^n$.

Veta 6 (Lagrangeova relaxácia)**.** *(Wolsey (1998)(str. 30))*
Nech $Z = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \in X \subseteq \mathbb{Z}^n\}$ *je úloha celočísleného programovania a nech* $Z(\boldsymbol{u}) = \max\{\boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{u}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) : \boldsymbol{x} \in X\}$*, potom* $Z(\boldsymbol{u}) \geq Z$ *pre* $\forall \boldsymbol{u} \in \mathbb{R}^m,\ \boldsymbol{u} \geq \boldsymbol{0}$

Dôkaz. Viď Wolsey (1998)(str. 30). □

Dualita a IP

V lineárnom programovaní využívame dualitu velmi často pri zisťovaní optimálneho riešenia a stanovovaní horných medzí. Použijeme, teda dualitu aj pri stanovovaní horných medzí problémov celočísleného programovania.

Definícia 9. *(Wolsey (1998)(str. 30))*
Nech $Z = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S_{IP}\}$*, potom úlohu* $Z^D = \min\{\boldsymbol{b}^T\boldsymbol{u} : \boldsymbol{u} \in S_D\}$,
kde $S_D = \{\boldsymbol{u} \in \mathbb{R}^m : \boldsymbol{A}^T\boldsymbol{u} \geq \boldsymbol{c},\ \boldsymbol{u} \geq \boldsymbol{0}\}$ *nazveme slabo duálnou ku pôvodnej úlohe ak pre* $\forall \boldsymbol{x} \in S_{IP}$ *a* $\forall \boldsymbol{u} \in S_D$ *platí:*

$$\boldsymbol{c}^T\boldsymbol{x} \leq \boldsymbol{b}^T\boldsymbol{u}.$$

V prípade, že $Z = Z^D$ *hovoríme o silno duálnej úlohe.*

Pomocou takto definovanej slabo duálnej úlohy vieme krásne nachádzať horné hranice, a to aj bez hľadania optima duálnej úlohy, stačí nám hodnota ľubovolného prípustného riešenia.

Ďalej si uvedieme, tvrdenie, ktoré nám ukáže ďalšie použitelné vlastnosti duality.

Tvrdenie 7. *(Wolsey (1998)(str.30))*

- Ak úloha Z^D z Definície 9 nemá optimálne riešenie, potom Z nemá prípustné riešenie
- Nech pre $\boldsymbol{x}^* \in S_{IP}$ *a* $\boldsymbol{u}^* \in S_D$ a platí $\boldsymbol{c}^T\boldsymbol{x}^* = \boldsymbol{b}^T\boldsymbol{u}^*$, potom $\boldsymbol{x}^*$ je optimálne riešenie pôvodnej úlohy IP a $\boldsymbol{u}^*$ je optimálne riešenie jej duálnej úlohy.

Dôkaz. Viď Wolsey (1998)(str.30).

□

Dualita nám teda dáva ďalší spôsob ako stanovovať horné medze, zisťovať či pôvodná úloha IP má vôbec riešenie a v špeciálnych prípadoch nám ukáže optimálnu hodnotu účelovej funkcie pôvodného problému. O špeciálnych prípadoch, kedy nám relaxácia IP ukáže optimálne riešenie pôvodnej úlohy IP bude nasledujúca podkapitola.

1.5 Totálna unimodularita

V minulej podkapitole sme zistili, že existujú prípady, kedy vieme nájsť optimálne riešenie úlohy celočísleného programovania z duality alebo dokonca len vyriešením lineárnej relaxácie úlohy $Z = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x} \geq \boldsymbol{0}\}$. V tejto podkapitole sa budeme venovať presne takým prípadom. Pomôže nám s tým totálna unimodularita alebo presnejšie, totálne unimodulárne matice.

Definícia 10.

- Celočíselná štvorcová matica sa nazýva *unimodulárna* ak jej determinant nadobúda jednu z hodnôt $\in \{1, -1\}$.
- Celočíselná matica, sa nazýva *totálne unimodulárna* ak determinant každej jej štvorcovej podmatice nadobúda jednu z hodnôt $\in \{1, 0, -1\}$

Poznámka. Totálne unimodulárne matice nebudú mať len determinanty štvorcových podmatíc nadobúdajúce hodnoty $\{1, 0, -1\}$, ale aj hodnoty všetkých ich prvkov budú 1, 0 alebo -1. Teda ak je matica $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ totálne unimodulárna, bude platiť $\forall i \in \{1, ..., m\}$ *a* $j \in \{1, ..., n\} : a_{i,j} \in \{1, 0, -1\}$.

Tvrdenie 8. *(Wolsey (1998)(str. 40))*
Úloha lineárneho programovania $Z^{LP} = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b},\ \boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{x} \geq \boldsymbol{0}\}$*, má celočíselné optimálne rešenie pre každý vektor* $\boldsymbol{b} \in \mathbb{Z}^n$*, pre ktorý je optimálna hodnota účelovej funkcie konečná, práve vtedy keď matica* $\boldsymbol{A}$ *je totálne unimodulárna.*

Dôkaz. Viď Wolsey (1998)(str. 40).

□

Vďaka tomuto tvrdeniu sme zistili, že používaním totálne unimodulárnych matíc, nájdeme celočíslené optimálne riešenia v lineárnom programovaní. No tu vzniká ďalšia úloha, a to overiť či je matica totálne unimodulárna. Pri väčších maticiac je overovanie totálnej unimodularity z definície obtiažne, pretože náročnosť overovania determinantov všetkých podmatíc je výpočtovo veľmi vysoká. Existujú však postačujúce podmienky na overenie totálnej unimodularity viď Papadimitriou a Steiglitz (1998)(str. 317). Na overenie totálnej unimodularity ľubovoľnej matice existuje algoritmus viď dôkaz vety z Schrijver (1986)(str.290).

1.6 Algoritmus vetvenia a medzí

V tejto podkapitole sa budeme zaoberať Algoritmom vetvenia a medzí, ktorý sa spolu s určitými ďalšími vylepšeniami používa v aktuálnych softvéroch na riešenie úloh celočíselného programovania. Povieme si na základe akých princípov funguje, a aké sú jeho možné úpravy alebo vylepšenia.

Jedným z princípov, na ktorých je algoritmus postavený je myšlienka rozdeliť komplikovaný problém na viacej menších. Potom následne riešiť dané jednoduchšie problémy. Nakoniec, ako najlepšie označit najlepšie optimálne riešenie jednej z dcérskych úloh, kde dcérska úloha vznikne z materskej zafixovaním jednej z rozhodovacích premenných na určitú hodnotu. Viacej o tvorbe dcérskych úloh v Podkapitole 1.6.2.
Uvažujme problém $Z = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S_{IP}\}$. Delenie na menšie problémy nám umožňuje tvrdenie:

Tvrdenie 9. *(Wolsey (1998)(str. 114))*
Nech $S_{IP} = S^1_{IP} \cup ... \cup S^K_{IP}$ je rozdelenie množiny pripustných riešení na menšie množiny a nech $Z^k = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S^k_{IP}\}$, pre $k = 1, ..., K$. Potom $Z = \max_k Z^k$.

Dôkaz. Viď Wolsey (1998)(str. 114).

□

Teraz sme problém rozčlenili na veľa menších, no ak by sme takto pokračovali z delením donekonečna, tak by sme náročnosť pôvodného problému akurát previedli do náročnosti riešiť obrovské množstvo malých problémov, preto budeme používať medze aby sme rozdelili problémy na tie, ktorých vyriešenie nám poskytne lepšie medze, a tie ktorých nie. Nasledujúce tvrdenie nám ukáže ako narábať s medzami a ďalšie, ako pomocou nich rozdeľovať problémy.

Tvrdenie 10. *(Wolsey (1998))*
Nech $S = S_1 \cup ... \cup S_K$ je rozdelenie množiny prípustných riešení na menšie a nech $Z^k = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S_k\}$ pre $k = 1, ..., K$, nech $\bar{Z}^k$ je horná medza pre Z^k a $\underline{Z}^k$ je dolná medza pre Z^k. Potom $\bar{Z} = \max_k \bar{Z}^k$ je horná medza pre Z a $\underline{Z} = \max_k \underline{Z}^k$ je dolná medza pre Z.

Tvrdenie 11. *(Prerezávanie podľa optimality)(Nemhauser (1998))*
Ak $\underline{Z}^k = \bar{Z}^k$ pre nejaké k, potom optimálne riešenie pre k-tý problém $Z^k = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S^k_{IP}\}$ je známe a nemusíme ďalej deliť S^k_{IP}.

Tvrdenie 12. *(Prerezávanie podľa medzí)(Nemhauser (1998))*
Ak $\bar{Z}^k < \underline{Z}^j$ pre nejaké $k \neq j$ a $k,j \in \{1, ..., K\}$, potom optimálne riešenie pre k-tý problém $Z^k = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S^k_{IP}\}$ nemôže byť nikdy optimálnym riešením pre pôvodný problém. Teda nie je potrebné ďalej deliť S^k_{IP}.

Tvrdenie 13. *(Prerezávanie podľa prípustnosti)(Nemhauser (1998))*
Ak $S^k_{IP} = \emptyset$, potom nie je potrebné ďalej deliť S^k_{IP}.

Takže takto algoritmus rozdelí počiatočný problém na veľa menších, z ktorých postupne vyberáme tie, ktoré sa oplatí riešiť, a tie ktoré nie. Postupne sa v priebehu algoritmu snažíme zmenšovať rozdiel medzi hornou a dolnou medzou.

Algoritmus končí v prípadoch:

- Vyriešime všetky dcérske úlohy a nájdeme optimálne riešenie jednej z nich s najvyššou hodnotou účelovej funkcie. To označíme za optimálne riešenie celého problému. Stane sa tak pred vypršaním časového limitu algroritmu.
- Nájdeme riešenie, ktorého hodnota účelovej funkcie bude v tolerancii ϵ, ktorá je definovaná ako: $\epsilon = \bar{Z} - \underline{Z}$, kým mi vyprší predom stanovený čas na riešenie problému. Dané riešenie označíme za $\epsilon-$optimálne.
- Nenájdeme optimálne riešenie, a taktiež do vypršania času nenájdeme ani riešenie, ktoré by bolo $\epsilon-$optimálne. V tomto prípade sa budeme musieť uspokojiť s najlepším riešením, ktoré sme našli alebo skúsiť opätovne spustiť algoritmus s väčším množstvom času na riešenie.

1.6.1 Hľadanie medzí

Nájdením lepšej spodnej medze vyradíme veľa dcerskych úloh, kde výsledkom bude zrýchlenie algoritmu. Na začiatku algoritmu stanovíme dolnú medzu (pri maximalizačnom probléme) na $-\infty$. Alternatívne riešenie (šikovnejšie) je nastaviť dolnú medzu na čo najvyššiu hodnotu nájdením, čo najlepšieho prípustného riešenia. Na to môžeme použiť metódu reštrikcie problému zo Podkapitoly 1.4.1 alebo metódu Greedy and Local search popísanú vo Wolsey (1998)(str. 35 - 37).

Na nájdenie horných medzí použijeme metódy z Podkapitoly 1.4.2. Budeme používať hlavne lineárnu relaxáciu, ktorej vyriešenie závisí od rýchlosti algoritmu riešiaceho danú úlohu lineárneho programovania. Jedným z algoritmov pre túto potrebu je duálny simplexový algoritmus, ktorý je popísaný napríklad v Papadimitriou a Steiglitz (1998).

1.6.2 Delenie na dcérske úlohy

Popísali sme základnú myšlienku algoritmu, ako riešiť a spájať riešenia dcérskych úloh, a teraz si popíšeme akým spôsobom budeme deliť zložitejšie úlohy na jednoduchšie.

Uvažujme pôvodnú úlohu $Z = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S_{IP}\}$. Riešme najprv jeho lineárnu relaxáciu. Ak jej optimálne riešenie je celočíselné, tak algoritmus končí. Ak optimálne riešenie nie je celočíselne, tak to znamená, že $\exists\ x_j^* \notin \mathbb{Z}_0^+,\ j \in \{1, ..., n\}$, kde x_j^* je j-tá zložka vektoru $\boldsymbol{x}^* \in \mathbb{R}^n$. V tom prípade rozdelím materskú úlohu na 2 dcérske:

- Úloha 1: $Z_1 = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S_{IP},\ x_j \leq \lfloor x_j^* \rfloor\}$
- Úloha 2: $Z_2 = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in S_{IP},\ x_j \geq \lfloor x_j^* \rfloor\ +\ 1\}$,

kde $\boldsymbol{x}^*$ je optimálne riešenie materskej úlohy a $\lfloor . \rfloor$ značí dolnú celú časť daného čísla. Následne postupujeme rovnako s dcérskymi úlohami až kým nenastane jedna z možností konca algoritmu popísaných vyššie v texte.

Výber premennej na vetvenie

Ak nám, ako adept na vetvetie vyjde iba jedna premenná, tak je rozhodovanie ako vetviť jednoduché. Horšie je, keď je takých premenných viacej. V tom prípade by dávalo zmysel vybrať takú, ktorej výberom sa v dcérskych úlohách zníži optimálna hodnota účelovej funkcie, čo najmenej, teda vyberáme tú, ktorá produkuje najlepšie riešenia. Pre tento účel predstavíme stratégiu vetvenia pomocou pseudostrátových faktorov (Pseudocost branching). Základom stratégie je odhadnutie pseudostrátových faktorov P_j^+ a P_j^- pre $\forall j \in \{1, ..., n\}$. Definujeme potrebné hodnoty:

- Z_{LP} je optimálna hodnota účelovej funkcie pre aktuálne riešenú úlohu.
- $\boldsymbol{x}^*$ je optimálne riešenie aktuálnej úlohy.
- $f_j = x_j^* - \lfloor x_j^* \rfloor$, kde $\lfloor x_j^* \rfloor$ je dolná celá časť hodnoty x_j^*.
- Hodnoty Z_j^- a Z_j^+ sú hodnoty účelovej funkcie ak by sme zafixovali $x_j = \lfloor x_j^* \rfloor$ alebo $x_j = \lfloor x_j^* \rfloor + 1$ a vyriešili s touto hodnotou aktuálne riešenú úlohu
- D_j^- a D_j^+ sú pseudostráty na hodnote účelovej funkcie aktuálnej úlohy pri vetvení podľa premennej x_j^* nadol a nahor. $D_j^- = f_j * P_j^-$ a $D_j^+ = (1 - f_j) * P_j^+$.

Ak sa pri vyriešení relaxácie IP dostaneme do bodu, keď riešenie x^* má viacej, ako jednu zložku neceločíselnú, spočítame pre každú takúto zložku D_j^- a D_j^+. Následne budeme vetviť podľa zložky, ktorej index získame, ako $\hat{j} = \text{argmax}_j[(D_j^- + D_j^+)]$ alebo $\hat{j} = \text{argmax}_j[\min(D_j^-,\ D_j^+)]$.
Pre výpočet D_j^- a D_j^+ potrebujeme odhadnúť pseudostrátove faktory P_j^- a P_j^+. Postup je naslednovný:

- Pred začatím riešenia počiatočnej úlohy vytvorím zoznamy $\boldsymbol{P}_j^-$ a $\boldsymbol{P}_j^+$ pre $\forall j \in \{1, ..., n\}$.
- Pri každom vetvení podľa premennej x_j priradím do zoznamu $\boldsymbol{P}_j^-$ hodnotu $\frac{Z_{LP} - Z_j^-}{f_j}$ a do zoznamu $\boldsymbol{P}_j^+$ hodnotu $\frac{Z_{LP} - Z_j^+}{1 - f_j}$.
- faktory P_j^- a P_j^+ odhadnem pri každom riešení podproblému, ako aritmetický priemer aktuálnych prvkov v zoznamoch $\boldsymbol{P}_j^-$ a $\boldsymbol{P}_j^+$.

Ako si môžeme všimnúť, faktory P_j^- a P_j^+ sme schopní odhadnúť až po zopár iteráciách algoritmu, keď v zoznamoch $\boldsymbol{P}_j^-$ a $\boldsymbol{P}_j^+$ máme už zopár prvkov. Dovtedy vetvíme podľa preferencie riešiteľa alebo podľa zložky x_j, ktorá maximalizuje funkciu $\min\{f_j, 1 - f_j\}$. Informácie o vetení pomocou pseudostrátových faktorov boli čerpané z Wolsey (1998)(str. 124).

1.6.3 Postupnosť riešenia úloh

Nasleduje problém, ako si vybrať, ktoré dcérske úlohy budeme riešit, ako prvé.

Možnosti:

- **Best first search:** Snažime sa najprv vetviť do hĺbky, teda pridávať dodatočné podmienky do predošlého problému, až kým nenájdeme celočíselné riešenie. Snažíme sa nájdeným riešením vytvoriť dobrý podklad pre vyhodenie veľkého množstva dcérskych úloh na základe Tvrdenia 12. Tento postup je navyše výhodný, pretože duálny simplexový algoritmus, ktorý sa často používa na riešenie úloh lineárneho programovania, vie po pridaní dodatočnej podmienky nájsť optimálne riešenie veľmi rýchlo. Vie sa rýchlo preoptimalizovať.

- **Best node first strategy:** Snažíme sa postupne riešiť úlohy s najvyššími hornými medzami, teda zvyšujeme svoju šancu, že nájdeme celočíselné riešenie s **najvyššou** hodnotou účelovej funkcie, čo najskôr. Touto stratégiou taktiež zaručíme, že nikdy nebudeme riešiť dcérsku úlohu, ktorá má hornú medzu nižšiu ako je optimálna hodnota účelovej funkcie celej úlohy.

V praxi sa používa kombinácia týchto prístupov. Najprv sa snažime nájsť celočíselné riešenie aby sme vyradili, čo najviac dcérskych úloh, a potom sa snažíme nájsť riešenie pomocou prístupu **Best node first strategy**.

1.6.4 Preprocesovanie problému

Preprocesovanie problému je zamerané na zlepšenie formulácie daného problému. Často sa stane, že pridaním dodatočnej podmienky výrazne zredukujeme problém. Avšak, bez dodatočného pohľadu a snahy upraviť formuláciu, a množinu prípustných riešení, zmeny neuvidíme. Preto, pomocou preprocesovania vyhodíme prebytočné premenné a prebytočné hranice množiny prípustných riešení. Celý tento proces sa dá spraviť algoritmicky. Jeho postup je popísany v nasledujúcom tvrdení.

Tvrdenie 14. *Majme množinu* $S = \{\boldsymbol{x} : a_0x_0 + \sum_{j=1}^{n} a_jx_j \leq b, l_j \leq x_j \leq k_j \text{ pre } j = 0, 1, ..., n\}$.

- *Hranice pre premenné. Ak* $a_0 > 0$, *potom:*

$$x_0 \leq \left(b - \sum_{j \geq 1 : a_j > 0} a_j l_j - \sum_{j \geq 1 : a_j < 0} a_j k_j \right) \Big/ a_0,$$

  a ak $a_0 < 0, potom$

$$x_0 \geq \left(b - \sum_{j \geq 1 : a_j > 0} a_j l_j - \sum_{j \geq 1 : a_j < 0} a_j k_j \right) \Big/ a_0.$$

- *Nadbytočnosť. Podmienka* $a_0x_0 + \sum_{j=1}^{n} a_jx_j \leq b$ *je nadbytočná ak platí:*

$$\sum_{j:a_j>0} a_j k_j \; + \sum_{j:a_j<0} a_j l_j \leq b.$$

- *Nemožnosť.* $S = \emptyset$ *ak:*

$$\sum_{j:a_j>0} a_j l_j \; + \sum_{j:a_j<0} a_j k_j > b.$$

- *Fixácia premenných. Majme maximalizačný problém v tvare:* $max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{Ax} \leq \boldsymbol{b},\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{k}\}$ *a* $\boldsymbol{l},\boldsymbol{k} \in \mathbb{R}^n,\ \boldsymbol{l},\boldsymbol{k} \geq \boldsymbol{0},$ *ak* $a_{ij} \geq 0$ *pre* $\forall i = 1, ..., m$ *a* $c_j < 0,$ *potom* $x_j = l_j$. *Naopak ak* $a_{ij} \leq 0,$ *pre* $\forall i = 1, ..., m$ *a* $c_j > 0,$ *potom* $x_j = k_j$.

Tieto obmedzenia a zmeny vyplývajú z príkladu vo Wolsey (1998)(str. 129 - 130)

Poznámka. Pri úlohe IP môžeme dokonca pritvrdiť a namiesto podmienky $l_j \leq x_j \leq k_j$, použiť podmienku $\lceil l_j \rceil \leq x_j \leq \lfloor k_j \rfloor$, kde $\lfloor . \rfloor$ značí dolnú celú časť daného čísla a $\lceil . \rceil$ značí hornú celú časť čísla.

.

2. Praktická časť

V tejto kapitole budeme riešiť reálny problém rozdeľovania kombinacie homogénnych a heterogénnych šperkov. Budeme rozdeľovať konkrétny súbor šperkov firmy ALO diamonds zo 17.10.2023. Spoločnosti produkujúce tovar pravideľne čelia problému optimálneho rozdelenia tovaru na svoje predajne. S rastom firiem vzniká potreba riešenia daného problému pomocou sofistikovaných algoritmov namiesto ručného rozdeľovania. V praktickej časti navrhneme algoritmus riešiaci danú problematiku. Uvedený algoritmus priradí šperky na butiky podľa odhadu pravdepodobnosti predaja konkrétneho šperku na konkrétnom butiku. Bude brať taktiež na zreteľ, koľko podobných šperkov sa už nachádza na danom butiku. Tento dodatok je dôležitý, pretože bez neho by sme všetky rovnaké šperky pridelili vždy na jeden butik s najlepšou predajnosťou daného modelu. Následne do algoritmu zahrnieme podmienky na základe, ktorých budeme schopní dodržať skladové obmedzenia. Nakoniec aplikujeme algoritmus pre 3 rôzne voľby vstupných parametrov a výsledky medzi sebou porovnáme.

2.1 Model

Vo formulácii modelu budeme pracovať s nasledujúcim značením:

- ID - Je súbor čísiel a písmen priradených každému šperku udávajúci jeho všetky vlastnosti. Rovnaké šperky majú rovnaké ID. Rozdielne šperky majú rozdielne ID.
- i - Index označujúci konkrétny šperk z $M = \{1, ..., m\}$.
- j - Index označujúci konkrétny butik z $N = \{1, ..., n\}$.
- $x_{i,j}$ - Hodnota bude 1 ak šperk i umiestnim na butik j, 0 inak.
- $k_{i,j}$ - Značí počet rovnakých šperkov ako šperk s indexom i nachádzajúcich sa na butiku j pred naskladňovaním.
- $h_{i,j}$ - Značí počet šperkov s vysokou podobnosťou so šperkom i nachádzajúcich sa na butiku j pred naskladňovaním. Pod pojmom šperky s vysokou podobnosťou myslíme šperky, ktoré sa líšia od šperku s indexom i iba farbou kameňa, nie farbou zlata ani tvarom.
- $s_{i,j}$ - Značí počet šperkov so strednou podobnosťou so šperkom i nachádzajúcich sa na butiku j pred naskladňovaním. Pod pojmom šperky so strednou podobnosťou myslíme šperky, ktoré sa líšia od šperku s indexom i farbou kameňa a ešte navyše aj farbou zlata, nie tvarom.
- $p_{i,j}$ - Je odhad pravdepodobnosti predaja šperku s rovnakými charakteristikami ako šperk i na butiku j za určený časový interval.
- $c_{i,j}$ - Je faktor dôležitosti šperku i pre butik j definovaný ako:

$$c_{i,j} = p_{i,j} * \frac{1}{3^{k_{i,j}} * 1.7^{h_{i,j}} * 1.5^{s_{i,j}}},$$

čísla (3, 1.7, 1.5) sú empiricky získané hodnoty, ktoré určujú, ako moc potrebujeme penalizovať butik j ak sa na ňom nachádzajú šperky s rôznymi stupňami podobností so šperkom i.

- H_i - Je množina indexov rovnakých šperkov ako je šperk s indexom i vrátane indexu i.

- L je množina indexov i. Počet jej prvkov je počtom rozdielnych šperkov. Pre každý index platí, že je jedným z indexov zo skupiny totožných šperkov. Zostrojím ju tak, že budem iterovať cez všetky indexy $i \in M$. Do množiny L pridám index i ak sa v množine L nenachádza index šperku s rovnakým ID, ako index šperku, ktorý chcem aktuálne priradiť. Teda docielime, že v množine L bude každý index patriť šperku s rôznym ID. Prvky množiny L budeme označovať l. Bude pre ne platiť $l_t \in \{l_1, \ ..., \ l_K\}$. K je počet rozdielnych šperkov.

- F_i - Je množina indexov šperkov vysokej podobnosti so šperkom i bez indexu i.

- G_i - Je množina indexov šperkov strednej podobnosti so šperkom i bez indexu i

- $y_{r,l_k,j}$ - Hodnota bude 1 ak je počet šperkov s indexami z množiny H_{l_k}, ktoré chceme naskladniť na butik j väčší, ako r. $r \in \{1, \ ..., \ 5\}$. Inak je to 0.

  Poznámka. Ako predpoklad budeme brať, že nechceme naraz rozdeľovať viacej ako 6 úplne rovnakých šperkov. Tento predpoklad vyplýva zo skúsenosti s povahou distribučných súborov značky ALOVE.

- $z_{r,l_k,j}$ - Hodnota bude 1 ak je počet šperkov s indexami z množiny F_{l_k}, ktoré chceme naskladniť na butik j väčší, ako r. $r \in \{1, \ ..., \ 8\}$. Inak je to 0.

  Poznámka. Ako predpoklad budeme brať, že nechceme naraz rozdeľovať viacej ako 9 šperkov s vysokou podobnosťou so šperkom s indexom l_k. Tento predpoklad vyplýva zo skúsenosti s povahou distribučných súborov značky ALOVE. Pre premenné $q_{r,l_k,j}$ použijeme rovnaký predpoklad.

- $q_{r,l_k,j}$ - Hodnota bude 1 ak je počet šperkov s indexami z množiny G_{l_k}, ktoré chceme naskladniť na butik j väčší, ako r. $r \in \{1, \ ..., \ 8\}$. Inak je to 0.

- d_j - Udáva počet šperkov, ktoré môžem ešte naskladniť na butik j bez prekročenia maxima skladu

Účelová funkcia:

$$\max\Big(A - B - C - D\Big).$$

Účelová funkcia bude pozostávať zo 4 častí *(A, B, C, D)*. Časť A sa bude snažiť maximalizovať dôležitosti šperkov pre dané butiky, teda súčet členov $c_{i,j}$.

$$A = \sum_{i=1}^{M} \sum_{j=1}^{N} c_{i,j} * x_{i,j}.$$

Časť B bude v maximalizačnej funkcii s mínuskom, takže funkcia sa bude snažiť minimalizovať časť B. To isté bude platiť aj pre časti C a D. Časť B bude penalizovať naskladnenie viac ako jedného kusu rovnakého šperku na rovnaký butik. Penalizácia bude narastať nelineárne s rastúcim počtom rovnakých naskladnených šperkov na rovnaký butik.

$$B = \sum_{j=1}^{N} \sum_{l=l_1}^{L} c_{l,j} * \sum_{r=1}^{5} y_{r,l,j} * \frac{r}{r+1}.$$

Príklad. Predstavme si, že chceme naskladňovať len 3 šperky s rovnakým ID. Ak by sme ich chceli naskladniť na rovnaký butik napr.(číslo 1) a index prvého šperku v dátovom súbore by bol 1, potom by mala účelová funkcia tvar max $c_{1,1} + c_{2,1} - \frac{c_{1,1}}{2} + c_{3,1} - \frac{2*c_{1,1}}{3}$, keď zvážime, že platí $c_{1,1} = c_{2,1} = c_{3,1}$, pretože sú to rovnaké šperky, bude mať účelová funkcia tvar max $c_{1,1} + \frac{c_{1,1}}{2} + \frac{c_{1,1}}{3}$.

Nelineárne správanie časti B sme schopní modelovať na základe premenných $y_{r,l,j}$ a príslušných stále sa zvyšujúcich faktorov $\frac{r}{r+1}$. Budeme teda napodobňovať priebeh funkcie $f(x) = \frac{x}{x+1}$. Prvé dve sumy nám vravia, že budeme penalizovať cez všetky butiky a cez všetky rozdielne šperky. Následne používame predpoklad, že nenaskladňujeme viacej ako 6 rovnakých šperkov, takže sa nemusíme pýtať pomocou premenných $y_{r,l,j}$ či dávame na butik viacej, ako 6 rovnakých šperkov.

Následne, potrebujeme do modelu zahrnúť aj vysoko podobné šperky so šperkom i a stredne podobné šperky so šperkom i. Potrebujeme ich zahrnúť z dôvodu vyváženia sortimentu. Stáva sa totižto, že ak je na butiku j úspešný konkrétny šperk i, tak sú na butiku úspešné aj jemu šperky podobné. Ak by sme podobné šperky do modelu nezahrnuli, mohlo by to mať za následok nadmerné naskladnenie butiku podobným sortimentom. Keďže, tovaru podobného šperku i je obmedzené množstvo, naskladnenie jeho priveľkého množstva na jeden butik implikuje nižšie alebo nulové naskladnenia daného tovaru na ostatné butiky. Nadmerné naskladnenie jedného butiku podobným tovarom, teda znižuje pestrosť sortimentu na iných butikoch, čo môže mať za následok ušlé zisky.

Časť C bude mať za úlohu penalizovať naskladnenie viacej ako jedného kusu vysoko podobného šperku so šperkom s indexom i na butik j. $i \in M$ a $j \in N$ a časť D bude mať za úlohu to isté, akurát so stredne podobnými šperkami.

$$C = \sum_{j=1}^{N} \sum_{l=l_1}^{L} \alpha * c_{l,j} * \sum_{r=1}^{8} z_{r,l,j} * \frac{r}{r+1}.$$

$$D = \sum_{j=1}^{N} \sum_{l=l_1}^{L} \beta * c_{l,j} * \sum_{r=1}^{8} q_{r,l,j} * \frac{r}{r+1}.$$

Úlohou koeficientov α a β je zaistiť nižšiu penalizáciu podobných šperkov v porovnaní s penalizáciou rovnakých šperkov. Pre tieto koeficienty bude teda platiť $1 > \alpha > \beta > 0$. Pri prvej implementácii modelu zvolíme $\alpha = \frac{3}{4}$ a $\beta = \frac{1}{2}$. Výsledná upravená účelová funkcia bude v tvare:

$$\max \sum_{j=1}^{N} \sum_{i=1}^{M} c_{i,j} * x_{i,j}$$
$$- \sum_{j=1}^{N} \sum_{l=l_1}^{L} \left(c_{l,j} * \sum_{r=1}^{5} y_{r,l,j} * \frac{r}{r+1} + c_{l,j} * \sum_{r=1}^{8} (\alpha * z_{r,l,j} + \beta * q_{r,l,j}) * \frac{r}{r+1} \right).$$

Obmedzujúce podmienky:

- Podmienka vraviaca, že každý šperk musí byť priradený presne na 1 butik:

$$\textstyle\sum_{j=1}^{n} x_{i,j} = 1, \ \forall i \in M.$$

- Podmienky zodpovedné za dodržanie skladových maxím:

$$\textstyle\sum_{i=1}^{M} x_{i,j} \leq d_j, \ \forall j \in N.$$

- Podmienky definujúce premenné $y_{r,l_k,j}$. Najprv si zadefinujeme podmienky potrebné na správne fungovanie premennej $y_{1,l_1,1}$, pre ktorú chceme aby nadobúdala hodnotu 1 ak sa na butik 1 naskladní viacej ako 1 šperk rovnaký so šperkom l_1, vrátane šperku l_1. Inak chcem aby mala premenná $y_{1,l_1,1}$ hodnotu 0. Dané správanie premennej $y_{1,l_1,1}$ docielime dvoma podmienkami:

$$\sum_{t \in H_{l_1}} x_{t,1} - 1 \leq \delta * y_{1,l_1,1},$$
$$\sum_{t \in H_{l_1}} x_{t,1} - 1 \geq -\delta * (1 - y_{1,l_1,1}) + 1.$$

  V podmienkach figuruje konštanta δ, ktorú stanovíme fixne na nejaké veľké číslo. Táto konštanta musí byť nastavená tak aby ju ľavá strana $\sum_{t \in H_{l_1}} x_{t,1} - 1$, nikdy nepresiahla. Jej hodnotu nastavíme na 1984. Prvá podmienka zabezpečuje, že premenná $y_{1,l_1,1}$ bude 1 ak je $\sum_{t \in H_{l_1}} x_{t,1} > 1$.
  Keď je $\sum_{t \in H_{l_1}} x_{t,1} \leq 1$, tak premenná $y_{1,l_1,1}$ môže nadobúdať hodnotu 0 alebo 1. Správne fungovanie dodáva až druhá pomienka, ktorá zaručuje, že ak je $\sum_{t \in H_{l_1}} x_{t,1} \leq 1$, tak premenná $y_{1,l_1,1}$ musí nadobúdať hodnotu 0. Ďalším krokom je definovať rovnako premenné y pre všetky rozdielne šperky, všetky butiky a hodnoty $r \in \{1, \ ..., \ 5\}$. Výsledné podmienky budú vyzerať nasledovne:

$$\sum_{t \in H_{l_k}} x_{t,j} - r \leq \delta * y_{r,l_k,j},$$
$$\sum_{t \in H_{l_k}} x_{t,j} - r \geq -\delta * (1 - y_{r,l_k,j}) + 1,$$
$$\forall r \in \{1, \ ..., \ 5\},$$
$$\forall l_k \in L,$$
$$\forall j \in N.$$

- Podmienky definujúce premenné $z_{r,l_k,j}$. Podmienky budú rovnaké ako podmienky definujúce premenné $y_{r,l_k,j}$ až na množinu indexov, cez ktorú sčítavame. Správanie premenných $z_{r,l_k,j}$ docielime podmienkami:

$$\sum_{t \in F_{l_k}} x_{t,j} - r \leq \delta * z_{r,l_k,j},$$
$$\sum_{t \in F_{l_k}} x_{t,j} - r \geq -\delta * (1 - z_{r,l_k,j}) + 1,$$
$$\forall r \in \{1,\ ...,\ 8\},$$
$$\forall l_k \in L,$$
$$\forall j \in N.$$

- Podmienky definujúce premenné $q_{r,l_k,j}$. Správanie premenných $q_{r,l_k,j}$ docielime podmienkami:

$$\sum_{t \in G_{l_k}} x_{t,j} - r \leq \delta * q_{r,l_k,j},$$
$$\sum_{t \in G_{l_k}} x_{t,j} - r \geq -\delta * (1 - q_{r,l_k,j}) + 1,$$
$$\forall r \in \{1,\ ...,\ 8\},$$
$$\forall l_k \in L,$$
$$\forall j \in N.$$

2.2 Predstavenie vstupných dát

Rozdeľovať budeme budeme 146 šperkov značky ALOVE, čo je jedna zo značiek patriacich spoločnosti ALO jewelry s.r.o. Uvažujeme reálne šperky, ktoré boli naskladňované dňa 17.10.2023. Pri každom šperku budeme pracovať s jeho ID. Ďalej budeme pracovať s dvoma úpravami ID, ktoré nám pomôžu rozoznať a zoskupiť podobné šperky. Navyše, použijeme údaje o type šperku, farbe zlata, farbe kameňa, počte kameňov a cenovej kategorii, ktoré budeme potrebovať na zoskupenie šperkov pre vypočítanie odhadu pravdepodobnosti predaja šperku i na butiku j. V dátovom súbore budeme mať pre každý šperk 14 riadkov, kde každý jeden riadok bude reprezentovať kombináciu konkrétneho šperku i a butiku j. Nakoniec pre každú kombináciu uvedieme počty rovnakých, vysoko podobných a stredne podobných šperkov so šperkom i aktuálne naskladnených na butiku j. Taktiež uvedieme dôležitosť butiku $c_{i,j}$. Ukážka vstupných dát je uvedená v Prílohe 1.

2.3 Riešenie

2.3.1 Použitá technika

K výpočtu bol použitý notebook Lenovo ThinkBook 15 G4 IAP so 64 - bitovým procesorom Intel Core i7-1255U 12th Gen 1.70 GHz s operačnou pamäťou 18 GB.

2.3.2 Technická implementácia modelu

Model prezentovaný v predminulej podkapitole bol riešený v programovacom jazyku Python pomocou knižnice *scipy.optimize*, z ktorej bola použitá funkcia *milp* (Mixed integer linear programming). Uvedená funkcia je vytvorená na riešenie optimalizačných úloh zmiešaného programovania a je v nej zabudovaný aj algoritmus na preprocesovanie modelu. Daná knižnica je zadarmo prístupná širokej verejnosti, vďaka čomu môže každý ľubovolne replikovať implementáciu modelu do python kódu, ktorá bude uvedená v prílohe. Pomocou vstupných parametrov bola funkcia *milp* naprogramovaná aby riešila nami zadanú úlohu binárneho programovania. Nastavenia vyzerajú nasledovne:

```
optimalizovane_hodnoty = milp(c=hodnoty_ucelovej_funkcie,
                              constraints=obmedzenia,
                              integrality=celociselnost,
                              bounds=obmedzenia_rozhodovacich_premennych,
                              options={"disp": True, "presolve": True})
```

Obr. 2.1: Classa milp

V zozname *hodnoty_ucelovej_funkcie* je umiestnený vektor cien. V premennej *obmedzenia* sú uvedené obmedzujúce podmienky použitím funkcie knižnice *scipy.optimize* s názvom *LinearConstraint*. Táto funkcia skombinuje maticu $\boldsymbol{A}$ s vektorom $\boldsymbol{b}$ a vytvorí súbor obmedzujúcich podmienok v tvare $\boldsymbol{Ax} \leq \boldsymbol{b}$, ktoré

spracováva funkcia *milp*. Premenná s názvom *celociselnost* je zoznam jednotiek o dĺžke rovnakej ako zoznam $hodnoty\_ucelovej\_funkcie$. Nastavením parametru *integrality* na samé jednotky totižto informujeme funkciu *milp*, že chceme aby každá jedna zložka vektoru rozhodovacích premenných mohla nadobúdať len celočíselné hodnoty. Premennou $obmedzenia\_rozhodovacich\_premennych$ nastavíme funkciu *milp*, tak aby mohli jednotlivé zložky vektoru rozhodovacích premenných nadobúdať iba hodnoty od 0 do 1 vrátane. Poslednými dvoma nastaveniami docielime, že zložky vektoru rozhodovacích premenných budú môcť nadobúdať iba binárne hodnoty. Nastavením "disp"na True a "presolve"na True zaistíme aby nám funkcia *milp* počas riešenia problému ukazovala vývoj riešenia problému, a aby bolo použité preprocesovanie.

2.3.3 Výsledné nastavenia modelu

Presuňme sa teraz ku výpočtu optimálneho riešenia modelu pre naše dáta, voľbu parametrov $\alpha = \frac{3}{4},\ \beta = \frac{1}{2}$ a obmedzenia skladov butikov pre butiky $j \in \{1,\ ...,\ 7\} : d_j = 100$ a $j \in \{8,\ ...,\ 14\} : d_j = 20$. Obmedzenia sme nastavili daným spôsobom z dôvodu, že vieme o butikoch $j \in \{8,\ ...,\ 14\}$, že nie sú hlavnými butikmi, kde chceme značku ALOVE distribuovať, ale iba vedľajšími, a taktiež majú menšie skladové kapacity pre danú značku, preto sú skladové obmedzenia nastavené daným spôsobom. Samotný výpočet s týmito nastaveniami trval 2,81 sekundy, no zostrojenie matice $\boldsymbol{A}$ nutnej pre výpočet zabralo asi 120 minút. Tento čas by sa dal možno skrátiť zefektivnením kódu vytvárajúceho maticu $\boldsymbol{A}$. Hodnota účelovej funkcie bola 78,63 a ku riešeniu sme sa dostali po 752 iteráciách algoritmu. Zdrojový kód aj so vstupnými dátami je uložený na GitHube: `https://github.com/Mirecmotuz/BAKALARKA_SPERKY/tree/c21cc5891593bd3889de631feaeb1191fe6d748a`.

2.3.4 Analýza efektu rôznych penalizácií na konkrétnom ID

Rozdeľovali sme 147 šperkov, z toho 109 šperkov malo rôzne ID. Úlohu rozdelenia šperkov sme okrem vstupných parametrov $\alpha = \frac{3}{4},\ \beta = \frac{1}{2}$ vyriešili aj pre parametre $\alpha = \frac{7}{8},\ \beta = \frac{5}{6}$ a $\alpha = \frac{1}{3},\ \beta = \frac{1}{4}$. Obmedzenia skladov zostali pri všetkých výpočtoch rovnaké. Riešili sme teda problém s normálnou penalizáciou podobných šperkov, vysokou penalizáciou podobných šperkov a nízkou penalizáciou podobných šperkov. Teraz sa pozrieme na tabuľku, ktorá popisuje, na aké butiky sa naskladnilo ID $274001526B.L20.B$ a všetky jeho podobné šperky pre všetky 3 rôzne nastavenia. Daná tabulka nám ukáže, ako sa prejavili rozdielne nastavenia parametrov α a β. Dané ID bolo vybrané, pretože má spomedzi celého súboru rozdeľovaných šperkov naväčsiu rozmanitosť a najväčší počet vysoko a stredne podobných šperkov. *V.pod* označuje šperky s vysokou podobnosťou so šperkom s ID $274001526B.L20.B$ a *S.pod.* označuje stredne podobné šperky s týmto ID.

Butiky	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ID	0	0	0	0	0	0	0	0	0	1	0	0	0	1
V.pod	0	0	0	0	0	0	0	1	1	0	0	0	0	0
S.pod	0	0	0	0	0	0	0	0	1	1	0	0	0	1

Tabuľka 2.1: Normálna penalizácia

Butiky	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ID	0	0	0	0	0	0	0	0	0	1	0	0	0	1
V.pod	0	0	0	0	0	0	0	1	1	0	0	0	0	0
S.pod	0	0	0	0	0	0	0	0	1	1	0	0	0	1

Tabuľka 2.2: Vysoká penalizácia

Butiky	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ID	0	0	0	0	0	0	0	0	0	1	0	0	0	1
V.pod	0	0	0	0	0	0	0	1	0	1	0	0	0	0
S.pod	0	0	0	0	0	0	0	0	1	0	0	0	0	2

Tabuľka 2.3: Nízka penalizácia

Z tabuliek vidíme, že normálna a vysoká penalizácia vyprodukovali rovnaké výsledky týkajuce sa ID 274001526$B.L20.B$ a jeho podobných šperkov. Rozdielne výsledky by sme vedeli zaznamenať až v momente, keď by sme mali väčšiu vzorku podobných šperkov na rozdelenie. Naša vzorka nebola dostatočná na ilustráciu rozdielov v týchto penalizáciach. Fakt, že sme v oboch prípadoch naskladnili na butiky 9, 10 a 14 po 2 podobné šperky je v poriadku, pretože nastavením modelu povoľujeme maličké naskladnenia podobných šperkov na rovnaké butiky. Je to z dôvodu, že veľké množstvo podobného tovaru je pre butik škodlivé, no malé množstvo je vítané z hľadiska variability. V modeli to povoľujeme pomocou toho, že penalizujeme až naskladnenia viacej ako 1 kusu. Pri tabuľke popisujúcej nízku penalizáciu naopak vidíme dôsledok nízkej penalizácie, keďže sme naskladnili už 2 kusy stredne podobného tovaru na butik 14, znamená to teda, že aj napriek penalizácii mal butik 14 stále väčšiu dôležitosť pre daný šperk ako butik 10 (kde sme daný kus naskladnili pri normálnej a vysokej penalizácii). Záverom z analýzy rozdelenia šperkov s ID 274001526$B.L20.B$ a im podobným šperkom pomocou nami prezentovaného modelu s rôznymi penalizáciami je, že môžeme vidieť dôležitosť dostatočnej penalizácie a úlohu akú penalizácia v modeli zohráva. Na posúdenie rozdielov medzi vysokou a normálnou penalizáciou by sme potrebovali väčšiu vzorku podobných šperkov.

2.3.5 Predstavenie výsledkov

Uvedieme tabulky popisujúce, ako boli rozdelené všetky šperky pre 3 rôzne penalizácie. Najprv uvedieme tabulku celkového počtu šperkov, ktoré chceme naskladniť na daný butik pri konkrétnej penalizácii, následne uvedieme tabulku, ktorá uvádza rozdelenia na konkrétne typy šperkov, potom rozdelenie podľa farby zlata, farby kameňa a nakoniec podľa príslušnosti šperku ku cenovej kategórii. *P.* bude označovať penalizáciu. *N.* bude penalizácia normálna, *V.* bude označovať vysokú penalizáciu a *Ni.* bude označovať nízku penalizáciu.

	Butiky													
P.	1	2	3	4	5	6	7	8	9	10	11	12	13	14
N.	11	15	14	6	5	0	9	9	6	15	17	2	20	18
V.	11	16	14	6	5	0	9	9	6	16	15	2	20	18
Ni.	11	15	14	6	3	0	9	13	5	15	15	2	20	19

Tabuľka 2.4: Celkové počty naskladnených šperkov na butiky

		Butiky													
P.	Typy	1	2	3	4	5	6	7	8	9	10	11	12	13	14
N.	Prsten	0	0	0	0	1	0	3	0	1	3	2	1	2	3
N.	Naus.	0	1	0	0	0	0	0	3	1	0	3	0	2	2
N.	Nahrd.	8	13	8	4	4	0	1	5	4	10	2	1	13	9
N.	Nar.	3	1	6	2	0	0	5	1	0	2	10	0	3	3
N.	Priv.	0	0	0	0	0	0	0	0	0	0	0	0	0	1
V.	Prsten	0	0	0	0	1	0	3	0	1	3	2	1	2	3
V.	Naus.	0	1	0	0	0	0	0	3	1	0	3	0	2	2
V.	Nahrd.	8	14	8	4	4	0	1	5	4	10	2	1	13	8
V.	Nar.	3	1	6	2	0	0	5	1	0	3	8	0	3	4
V.	Priv.	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Ni.	Prsten	0	0	0	0	1	0	3	0	1	3	2	1	2	3
Ni.	Naus.	0	1	0	0	0	0	0	7	1	0	1	0	0	2
Ni.	Nahrd.	8	13	8	4	2	0	1	5	3	10	2	1	15	10
Ni.	Nar.	3	1	6	2	0	0	5	1	0	2	10	0	3	3
Ni.	Priv.	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Tabuľka 2.5: Naskladnenia podľa konkrétnych typov šperkov

Naus. je skratka pre náušnice, *Nahrd.* je skratka pre náhrdelník, *Nar.* je skratka pre náramok, *Priv.* je skratka pre prívesok.

		Butiky													
P.	Farba	1	2	3	4	5	6	7	8	9	10	11	12	13	14
N.	Zlt.	0	1	8	4	4	0	0	4	2	9	6	0	16	5
N.	Ru.	1	0	5	1	0	0	4	0	3	3	4	0	0	4
N.	Bie.	10	14	1	1	1	0	5	5	1	3	7	2	4	9
V.	Zlt.	0	2	8	4	4	0	0	4	2	10	5	0	16	4
V.	Ru.	1	0	5	1	0	0	4	0	3	3	4	0	0	4
V.	Bie.	10	14	1	1	1	0	5	5	1	3	6	2	4	10
Ni.	Zlt.	0	1	8	4	2	0	0	8	2	8	4	0	16	6
Ni.	Ru.	1	0	5	1	0	0	4	0	3	3	4	0	0	4
Ni.	Bie.	10	14	1	1	1	0	5	5	0	4	7	2	4	9

Tabuľka 2.6: Naskladnenia podľa farby zlata

Zlt. je skratka pre žlté zlato, *Ru.* je skratka pre ružové zlato a *Bie.* je skratka pre biele zlato.

		Butiky													
P.	Farba	1	2	3	4	5	6	7	8	9	10	11	12	13	14
N.	Bie.	10	14	13	6	5	0	5	5	5	11	13	2	18	13
N.	Mod.	0	0	0	0	0	0	0	0	0	1	0	0	0	1
N.	Zlt.	0	0	0	0	0	0	0	0	1	0	0	0	0	0
N.	Ne.	1	1	1	0	0	0	4	4	0	3	4	0	2	4
V.	Bie.	10	15	13	6	5	0	5	5	5	12	11	2	18	13
V.	Mod.	0	0	0	0	0	0	0	0	0	1	0	0	0	1
V.	Zlt.	0	0	0	0	0	0	0	0	1	0	0	0	0	0
V.	Ne.	1	1	1	0	0	0	4	4	0	3	4	0	2	4
Ni.	Bie.	10	14	13	6	3	0	5	5	4	11	13	2	20	14
Ni.	Mod.	0	0	0	0	0	0	0	0	0	1	0	0	0	1
Ni.	Zlt.	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Ni.	Ne.	1	1	1	0	0	0	4	8	0	3	2	0	0	4

Tabuľka 2.7: Naskladnenia podľa farby kameňa

Bie. je skratka pre bielu farbu, *Mod.* je skratka pre modrú farbu, *Zlt.* je skratka pre žltú farbu a *Ne.* je skratka pre šperky, ktorých farba kameňa nebola vo vstupných dátach uvdedená.

		Butiky													
P.	Kat.	1	2	3	4	5	6	7	8	9	10	11	12	13	14
N.	C1	11	13	14	6	4	0	7	5	2	5	5	1	16	9
N.	C2	0	2	0	0	1	0	1	3	0	4	11	1	4	6
N.	C3	0	0	0	0	0	0	0	0	4	0	0	0	0	0
N.	C4	0	0	0	0	0	0	0	0	0	1	1	0	0	2
N.	C5	0	0	0	0	0	0	1	0	0	5	0	0	0	0
N.	C6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
N.	C7	0	0	0	0	0	0	0	0	0	0	0	0	0	1
V.	C1	11	14	14	6	4	0	7	5	2	5	5	1	16	8
V.	C2	0	2	0	0	1	0	1	3	0	5	9	1	4	7
V.	C3	0	0	0	0	0	0	0	0	4	0	0	0	0	0
V.	C4	0	0	0	0	0	0	0	0	0	1	1	0	0	2
V.	C5	0	0	0	0	0	0	1	0	0	5	0	0	0	0
V.	C6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
V.	C7	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Ni.	C1	11	13	14	6	2	0	7	4	2	5	5	1	18	10
Ni.	C2	0	2	0	0	1	0	1	7	0	4	9	1	2	6
Ni.	C3	0	0	0	0	0	0	0	1	3	0	0	0	0	0
Ni.	C4	0	0	0	0	0	0	0	0	0	1	1	0	0	2
Ni.	C5	0	0	0	0	0	0	1	0	0	5	0	0	0	0
Ni.	C6	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Ni.	C7	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Tabuľka 2.8: Naskladnenie podľa cenovej kategórie

2.3.6 Zhodnotenie výsledkov

Čo sa týka celkového rozmiestnenia distribuovaných šperkov na butiky, môžeme vidieť, že sú preferované butiky 1, 2, 3, 10, 11, 13, 14. U butikov 1, 2, 3 nám pridelené počty vyhovujú, no u butikov 10, 11, 13, 14 tomu už tak nie je. Dôvody sú spomenuté vyššie v Podkapitole 2.3.3. Ako môžeme vidieť, butik 13 má dokonca dostať 20 šperkov, čo je jeho hornou hranicou určenou skladovým limitom. Tento problém je možné vyriešiť znížením skladových limitov pre butiky $j \in \{8, ..., 14\}$. Nižšie uvedená tabulka udáva celkové počty rozdelených šperkov zo vstupný dát na butiky pri penalizačných faktoroch $\alpha = \frac{3}{4}$, $\beta = \frac{1}{2}$ a znížených skladových limitoch $d_j, \forall j \in \{8, ..., 14\}$ z 20 na 8. Zdrojový kód, je uvedený na GitHub účte uvedenom v Podkapitole 2.3.3. Výpočet trval 3,47 sekundy a hodnota účelovej funkcie bola 73,28.

Butik	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Počet	14	25	17	12	6	4	13	8	8	8	8	8	8	8

Tabuľka 2.9: Celkové počty naskladnených šperkov na butiky s upravenými limitami

Po preskúmaní nových výsledkov môžeme vidieť, že tento problém sa nám podarilo vyriešiť, pretože sme presunuli vysoké počty naskladnených šperkov z butikov, kde by vysoké počty mohli byť problematické, na butiky, kde budú pros-

pešné. Z ostatných tabuliek môžeme napríklad vidieť, že pri náhrdelníkoch je preferovaný butik číslo 10. Tento butik je taktiež preferovaný pri naskladňovaní šperkov cenovej kategórie C5. Ostatné informácie o rozdeleniach šperkov sú uvedené čitateľovi pre zaujímavosť.

Záver

V tejto práci sme sa zaoberali problémom optimálneho rozdelenia tovaru na predajne. V teoretickej časti sme čitateľa zoznámili s teóriou celočísleného programovania a s Algoritmom vetvenia a medzí. Uvedené teoretické poznatky sme zúročili v praktickej časti, kde sme navrhli model na rozdeľovanie šperkov, ktorý rozdeľuje šperky na základe ich predajnej úspešnosti na konkrétnych butikoch a navyše prihliada aj na množstvá podobného tovaru na daných butikoch. Model bol implementovaný pomocou programovacieho jazyku Pyhon a knižnice *scipy.optimize*, z ktorej sme na riešenie použili funkciu *milp*. Výsledky rozdelenia vstupných dát sme zanalyzovali a zistili, že šperky sa rozdelili v nadmernom množstve na butiky, kde mali byť ako doplnkový tovar. Navrhli sme riešenie pomocou zníženia skladových limitov daných butikov, s ktorým sme opäť vyriešili daný model. Výsledné rozdelenie už bolo uspokojujúce. V praktickej časti sme taktiež demonštrovali dôležitosť dostatočnej penalizácie pomocou analýzy rôznych penalizačných konštánt.

Zoznam použitej literatúry

Dupačová, J. a Lachout, P. (2011). *Úvod do optimalizace.* Matfyzpress. ISBN 0-471-28366-5.

Nemhauser, G. L. a Wolsey, L. A. (1998). *Integer and Combinatorial Optimization*. JohnWiley Sons, Inc. ISBN 0-471-82819-X.

Papadimitriou, C. H. a Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. General Publishing Company, Ltd. ISBN 0-486-40258-4.

Schrijver, A. (1986). *Theory of linear and integer programming*. JohnWiley Sons, Inc. ISBN 0-471-90854-1.

Wolsey, L. A. (1998). *Integer Programming*. JohnWiley Sons, Inc. ISBN 9781119606536.

Zoznam obrázkov

Zoznam tabuliek

Príloha 1

sperk	butik	ID	podobnost_10	podobnost_9	typ_sperku	cenkat	farba_zlata	farba_kame	kamenovost	pravdepod	typ_pravitik	ID_politik	politik_10	politik_9	poc	faktor_c
1	1	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.663969	konk	0	0	0	0	0.66397
1	2	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.401467	konk	0	0	0	0	0.40147
1	3	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.760741	konk	0	0	0	0	0.76074
1	4	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.7021	konk	0	0	0	0	0.7021
1	5	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.775877	konk	0	0	0	0	0.77588
1	6	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.318044	konk	0	0	0	0	0.31804
1	7	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.704416	konk	0	0	0	0	0.70442
1	8	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.319896	bayes	0	0	0	0	0.3199
1	9	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.34101	bayes	0	0	0	0	0.34101
1	10	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.493524	bayes	0	0	0	0	0.49352
1	11	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.436655	konk	0	0	0	0	0.43666
1	12	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.056933	konk	0	0	0	0	0.05693
1	13	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.791855	bayes	0	0	0	0	0.79186
1	14	2240032162	2240032162	224003216	Prsten	C1	Žlutá	BR bílá/čir	Jednokam	0.468045	bayes	0	0	0	0	0.46805
2	1	234002344R	234002344R	234002344	Náušnice	C2	Růžová	BR bílá/čir	Jednokam	0.855645	konk	0	0	0	1	0.57043
2	2	234002344R	234002344R	234002344	Náušnice	C2	Růžová	BR bílá/čir	Jednokam	0.631446	konk	0	0	0	0	0.63145

Obr. 2.2: Vstupné data