

Metódy celočíselného lineárneho programovania

Monika Kovárová



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav informatiky a umělé inteligence

Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Monika Kovárová
Osobní číslo: A21007
Studijní program: B0613A140020 Softwarové inženýrství
Forma studia: Kombinovaná
Téma práce: Metody celočíselného lineárního programování
Téma práce anglicky: Methods of Integer Linear Programming

Zásady pro vypracování

- Nastudujte historii, vývoj a klasifikaci metod lineárního programování.
- Formulujte postup řešení simplexovou tabulkou pro primární a duální úlohu.
- Vysvětlete aspekty celočíselné úlohy a princip řezných nadrovin.
- Vypracujte a využijte programovou podporu pro řešení celočíselných úloh LP.
- Vypracujte aplikační příklady pro využití celočíselných úloh.
- Vytvořte studijní oporu a porovnání s metodou větví a mezí.

Forma zpracování bakalářské práce: **tištěná/elektronická**
Jazyk zpracování: **Slovenština**

Seznam doporučené literatury:

1. DANTZIG, George B.. Lineárne programovanie a jeho rozvoj. 1. vyd. Bratislava: SVTL, 1966. 704 s.
2. DEMEL, Jiří. Operační výzkum [online]. 2017 [cit. 2018-03-08], učební text ČVUT. Dostupné na: <https://kix.fsv.cvut.cz/demel/ped/ov/ov.pdf>.
3. DUPAČOVÁ, Jitka, LACHOUT, Petr. Úvod do optimalizace. 1.vyd. Praha: Matfyzpress, 2011. 88s. ISBN 978-7378-176-7.
4. GRYGAROVÁ, Libuše. Úvod do lineárního programování, skripty. 1. vyd. Praha: Státní pedagogické nakladatelství, 1975.
5. HLADÍK, Milan. Celočíselné programování. 2017, text k přednášce na MFF UK. Dostupné na: https://kam.mff.cuni.cz/hladik/CP/text_cp.pdf.
6. JABLONSKÝ, Josef. Operační výzkum: kvantitativní modely pro ekonomické rozhodování. 1. vyd. Praha: Professional publishing, 2002. 323 s. ISBN 80-86419-23-1.

Vedoucí bakalářské práce: **prof. Ing. Roman Prokop, CSc.**
Ústav matematiky

Datum zadání bakalářské práce: **27. října 2024**
Termín odevzdání bakalářské práce: **2. června 2025**



L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Bakalárska práca sa zaoberá problematikou celočíselného lineárneho programovania. Prináša prehľad histórie a vývoja lineárneho programovania ako významnej oblasti operačného výskumu a matematickej optimalizácie. Popisuje typické úlohy riešené pomocou lineárneho programovania a vysvetľuje princípy simplexového algoritmu, vrátane konceptu simplexovej tabuľky. Hlavná časť práce je venovaná riešeniu celočíselných lineárnych úloh metódou Gomoryho rezov a prináša jej porovnanie s metódou vetiev a medzí. V závere sú predstavené možnosti praktického riešenia celočíselných úloh lineárneho programovania pomocou populárnych doplnkov pre Microsoft Excel, konkrétne nástrojov Solver, OpenSolver a SolverStudio.

Kľúčové slová: Celočíselné lineárne programovanie, lineárne programovanie, simplexový algoritmus, simplexová tabuľka, dualita, Gomoryho rez, metóda vetiev a medzí, Excel, Solver, Riešiteľ, OpenSolver, SolverStudio

ABSTRACT

The bachelor's thesis addresses the topic of integer linear programming. It provides an overview of the history and development of linear programming as an important area of operations research and mathematical optimization. The thesis describes typical problems solved using linear programming and explains the principles of the simplex algorithm, including the concept of the simplex tableau. The main part of the thesis focuses on solving integer linear programming problems using Gomory's cutting plane method and presents a comparison with the branch-and-bound method. The thesis introduces practical approaches to solving integer linear programming problems using popular Microsoft Excel add-ins, specifically the Solver, OpenSolver, and SolverStudio.

Key words: Integer linear programming, linear programming, simplex algorithm, simplex tableau, duality, Gomory cuts, branch and bound method, Excel, Solver, OpenSolver, SolverStudio

Na tomto mieste by som sa rada úprimne poďakovala svojmu školiteľovi, pánovi profesorovi Prokopovi, za jeho láskavý prístup, odborné vedenie, trpezlivosť a ľudský rozmer, ktorý bol pre mňa motiváciou pri písaní tejto práce.

OBSAH

ÚVOD	10
I TEORETICKÁ ČASŤ.....	11
1 LINEÁRNE PROGRAMOVANIE	12
1.1 HISTÓRIA A VÝVOJ LINEÁRNEHO PROGRAMOVANIA	12
1.2 EKONOMICKÝ A MATEMATICKÝ MODEL ÚLOH LINEÁRNEHO PROGRAMOVANIA...	14
1.3 TYPICKÉ ÚLOHY LINEÁRNEHO PROGRAMOVANIA	18
2 SIMPLEXOVÁ METÓDA	21
2.1 SIMPLEXOVÁ TABUĽKA.....	23
2.2 DUALITA ÚLOH LINEÁRNEHO PROGRAMOVANIA	28
3 CELOČÍSELNÉ LINEÁRNE PROGRAMOVANIE	32
3.1 METÓDY REZNÝCH ROVÍN.....	33
3.2 METÓDA VETIEV A MEDZÍ.....	35
4 NÁSTROJE NA RIEŠENIE ÚLOH CELOČÍSELNÉHO LINEÁRNEHO PROGRAMOVANIA	37
4.1 MODELOVACIE JAZYKY	37
4.2 OPTIMALIZAČNE SOLVERY	38
4.3 MS EXCEL A JEHO DOPLNKY	38
II PRAKTICKÁ ČASŤ	40
5 RIEŠENIE ÚLOHY METÓDOU GOMORYHO REZOV	41
5.1 POROVNANIE METÓDY GOMORYHO REZOV A METÓDY VETIEV A MEDZÍ.....	47
6 PROGRAMOVÁ PODPORA CELOČÍSELNÝCH ÚLOH.....	49
6.1 DOPLNOK RIEŠITEĽ.....	49
6.2 DOPLNOK OPENSOLVER	54
6.3 DOPLNOK STUDIO SOLVER S VYUŽITÍM PROGRAMOVACIEHO JAZYKA PYTHON....	58
ZÁVER.....	63
ZOZNAM POUŽITEJ LITERATÚRY	64
ZOZNAM TABULIEK.....	67
ZOZNAM PRÍLOH	68

ÚVOD

Optimalizácia predstavuje jednu z kľúčových oblastí moderného vedeckého a technického výskumu, ktorá nachádza uplatnenie v širokom spektre praktických problémov od priemyslu cez logistiku až po ekonomiku. Lineárne programovanie, ako významná časť optimalizácie, umožňuje efektívne riešiť úlohy, ktorých cieľom je maximalizácia alebo minimalizácia lineárnej cieľovej funkcie za splnenia lineárnych obmedzení.

V mnohých reálnych situáciách však nie je možné povoliť zmeny premenných vo forme ľubovoľných reálnych hodnôt – často sú požadované celočíselné riešenia, ktoré zodpovedajú konkrétnym jednotkám, predmetom či rozhodnutiam, ktoré nemožno deliť. Práve takéto problémy rieši celočíselné lineárne programovanie, ktoré prináša ďalšiu vrstvu náročnosti a zaujímavých matematických metód.

Táto práca sa venuje práve takýmto problémom, kde je potrebné hľadať riešenia vo forme celých čísel. Teoretická časť práce je rozdelená do štyroch kapitol. Prvá kapitola prináša prehľad histórie a vývoja lineárneho programovania ako významnej oblasti matematickej optimalizácie a operačného výskumu. V druhej kapitole sa detailne venuje simplexovej metóde, vrátane vysvetlenia princípov simplexovej tabuľky, ktorá je základným nástrojom pre riešenie lineárnych programov. Tretia kapitola je zameraná na celočíselné lineárne programovanie, kde sú predstavené hlavné metódy riešenia, ako napríklad Gomoryho rezy či metóda vetiev a medzí. Štvrtá kapitola prináša prehľad nástrojov na riešenie úloh celočíselného lineárneho programovania.

Praktická časť práce sa sústreďuje na aplikáciu metódy Gomoryho rezov na konkrétny príklad, vrátane porovnania jej efektívnosti s metódou vetiev a medzí. Zároveň predstavuje možnosti riešenia celočíselných úloh pomocou populárnych softvérových doplnkov pre Microsoft Excel, konkrétne nástrojov Solver, OpenSolver a SolverStudio.

Cieľom tejto práce je nielen teoreticky objasniť problematiku celočíselného lineárneho programovania, ale aj poskytnúť praktické návody a ukážky využitia moderných nástrojov na riešenie týchto úloh.

I TEORETICKÁ ČASŤ

1 LINEÁRNE PROGRAMOVANIE

Termín lineárne programovanie zaviedol americký matematik George Dantzig, počas svojej práce pre americké letectvo, v 40. rokoch 20. storočia. Slovo programovanie nesúviselo s písaním počítačového kódu, čo je dnes bežné chápanie tohto slova. Používalo sa v armádnych kruhoch vo význame rozvrh či plán činnosti. Cieľom lineárneho programovania bolo stanovenie optimálneho plánu a slovo lineárne nám napovedá, že tieto plány sú vymedzené lineárnymi podmienkami uvažovaných veličín a jeho kvalita sa porovnáva určitou lineárnou funkciou. [14]

Počas druhej svetovej vojny bolo lineárne programovanie využívané najmä na vojenské účely. Avšak už počas druhej svetovej vojny a bezprostredne po nej prebiehali výskumy a práce týkajúce sa problémov plánovania v poľnohospodárstve, rozloženia dopravných ciest medzi prístavmi, či prúdovania tovarov medzi odvetvami národného hospodárstva. Až do roku 1947, keď lineárne programovanie zjednotilo rozdielne oblasti tým, že dalo lineárnemu programovaniu matematickú kostru a prvú výpočtovú metódu – simplexový algoritmus. [3]

Simplexový algoritmus sa stal prvým systematickým a efektívnym spôsobom riešenia lineárneho programovania, ktoré začalo rýchlo prenikať aj do ďalších mnohých odvetví a to najmä do ekonomiky, priemyslu, dopravy a logistiky, neskôr i do energetiky, financií, telekomunikácií či IT. Postupne nasledoval aj vývoj metód lineárneho programovania a každá nová metóda prinášala vylepšenia v efektivite, viac možností i vyššiu rýchlosť pri riešení úloh.

Dnes má lineárne programovanie široké využitie všade tam, kde je potrebné efektívne rozhodovanie a optimalizácia zdrojov. I naše každodenné činnosti sú ovplyvnené snahou o ich optimalizáciu. Snažíme sa niečo minimalizovať alebo naopak maximalizovať, hľadáme extrémny, limity, či obmedzujúce faktory. Podobne sa to deje vo firmách a podnikoch, kde sa stalo lineárne programovanie súčasťou každodenného rozhodovania.

1.1 História a vývoj lineárneho programovania

Základy lineárneho programovania boli položené nezávisle na sebe na viacerých miestach a v rôznom čase. Za prvého, kto formálne formuloval problém, ktorý možno interpretovať ako problém lineárneho programovania, je považovaný francúzsky matematik a fyzik J. Fourier. Zaoberal sa riešením sústavy lineárnych nerovníc (1826) (Fourier-Motzkinova eliminácia, metódu v roku 1936 formalizoval T. Motzkin), ako aj optimalizáciou a hľadaním maxima

a minima vo fyzike (Analytická teória tepla, 1822). Jeho metódy položili matematické základy pre neskoršie algoritmy. [17]

Matematický génus a astronóm Carl Fridrich Gauss vyvinul metódy, ktoré sú dnes nevyhnutné pre riešenie lineárnych optimalizačných problémov. Gaussova eliminácia, neskôr upravená do Gaussovo-Jordanovej eliminácie, je základná metóda pre riešenie sústav lineárnych rovníc a výrazne ovplyvnila vývoj algoritmov na riešenie lineárneho programovania. [3]

Za jedného z otcov lineárneho programovania je považovaný ruský matematik L.V.Kantorovič. V roku 1939 publikoval prácu „Mathematical Methods in the Organization and Planning of Production“ v ktorej navrhol spôsob optimalizácie priemyselnej výroby. V práci použil lineárne obmedzenia, cieľovú funkciu a iteratívne metódy, podobné neskoršiemu simplexovému algoritmu. Zaviedol koncept tieňových cien, čo je základná myšlienka duality v lineárnom programovaní. Kantorovičove práce vznikali v dobe, kedy bola trhová optimalizácia v rozpore s centrálnou plánovanou ekonomikou, a tak sa jeho metódam dostalo uznania až o niekoľko desiatok rokov neskôr, keď spolu s T. Koopmansom získali Nobelovu cenu za teóriu optimálnej alokácie zdrojov. [5]

Klasický dopravný problém, (preprava tovaru z miest výroby do miest dopytu pri minimálnych nákladoch), pochádza od Franka L.Hitchcocka, ktorý vo svojom článku z roku 1941 načrtnol aj riešiteľský postup. Nezávisle od Hitchcocka sa rovnakým problémom zaoberal aj ekonóm Tjalling C. Koopmans. Preto sa tento problém nazýva Hitchcock-Koopmansov dopravný problém. [5]

Nezabudnuteľnú zásluhu na teoretickom rozvoji lineárneho programovania a vynálezcom simplexovej metódy je Američan George B.Dantzig. Vedec, profesor dopravy a operačného výskumu na Stanfordskej univerzite a hlavný matematik výskumného tímu amerického letectva v projekte SCOOP (Scientific Computation of Optimal Programs). V rokoch 1947-1955 matematicky formuloval a otestoval funkčnosť modelu lineárneho programovania a simplexovej metódy. Dantzigova práca bola v roku 2000 zaradená medzi desať najvýznamnejších algoritmov 20. storočia. [2]

Ďalšou významnou osobnosťou bol maďarsko-americký matematik John von Neumann, ktorý prispel k rôznym oblastiam matematiky, fyziky a informatiky, vrátane lineárneho programovania. Patrí k spoluvytvorcom teórie hier a v roku 1947 definoval teóriu duality.

Problematike duality a rozšíreniu lineárneho programovania do teórie hier a ekonomiky sa venoval aj kanadsko-americký matematik Albert W. Tucker. V roku 1950 sformoval koncept

Dilema väzňa, ktorý ilustruje problémy kooperácie a konfliktu v teórii hier. Spolu s Halaldom W. Kuhnom formalizovali optimálne podmienky pre nelineárne programovanie.

Po roku 1950 sa oblasti operačného výskumu a lineárneho programovania začali veľmi rýchlo rozvíjať. Vďaka novým algoritmom a teoretickým prístupom sa začiatkom 50. rokov 20. storočia začala formovať špeciálna oblasť lineárneho programovania, celočíselné lineárne programovanie. Za jeho zakladateľa je považovaný americký matematik a informatik Ralph E. Gomory, ktorý dosiahol teoretický i výpočtový pokrok riešením lineárneho problému pomocou rezných rovín. Gomoryho práca priniesla revolúciu v aplikácii lineárneho programovania na problémy, kde boli vyžadované celočíselné riešenia a táto metóda sa stala základným nástrojom v oblasti celočíselného programovania a optimalizácie [5].

Alternatívnym prístupom k riešeniu niektorých celočíselných problémov je dynamické programovanie, ktoré predstavuje techniku optimalizácie založenej na rekurzívnom delení úlohy na menšie časti. O jeho rozvoj sa zaslúžil Richard Bellman. O pár rokov neskôr, v roku 1960, bola publikovaná práca dvojice autorov Ailsy H. Land a Alison G. Doig, „An Automatic Method of Solving Discrete Programming Problems“, ktorá sa považuje za pôvod metódy vetiev a medzí – branch and bound. Táto metóda vetví problém na menšie časti v snahe eliminovať neefektívne vetvy. K ďalším osobnostiam, ktoré ovplyvnili vývoj celočíselného lineárneho programovania patrí aj Jack Edmonds, kanadský matematik a informatik, ktorý zaviedol polyhedrálnu teóriu. Metódu rezných rovín významne v 70. a 80. rokoch rozšíril matematik Egon Balas. Súčasťou moderných solverov celočíselného lineárneho programovania sú dnes algoritmy a metódy o ktoré sa zaslúžili matematici László Lovász, Martin Grötschel, či Laurenc Wolsey. [5]

1.2 Ekonomický a matematický model úloh lineárneho programovania

"Lineárne programovanie je metóda optimalizácie, ktorá sa zameriava na maximalizáciu alebo minimalizáciu lineárnej cieľovej funkcie, pričom sú splnené obmedzenia vyjadrené ako lineárne rovnice alebo nerovnice". [3]

Ako napovedá predošlá veta, lineárne programovanie patrí do oblasti matematického programovania, ktoré je jedným z odvetví optimalizácie. Z matematického hľadiska predstavuje oblasť, ktorá sa zaoberá optimalizáciou lineárnej cieľovej funkcie, pri rešpektovaní lineárnych obmedzení. K nájdeniu optimálneho riešenia problému využíva rôzne metódy a algoritmy, ktoré majú dva hlavné prvky.

1. Ciele – sú vyjadrené pomocou jednej alebo viacerých lineárnych funkcií viacerých premenných, nazývané ako účelové (cieľové, kritériálne) funkcie, ktoré sa snažíme maximalizovať (zisk, produktivitu, efektívnosť), alebo minimalizovať (náklady, odpad, spotrebu).
2. Ohraničujúce podmienky – definujú obmedzenia a podmienky, ktoré je nutné splniť, aby bolo riešenie platné. Tieto obmedzenia v praxi reprezentujú rôzne fyzické, ekonomické alebo technologické limity, ktoré je nutné rešpektovať pri hľadaní optimálneho riešenia. Matematicky sú vyjadrené lineárnymi rovnicami, nerovnicami alebo sústavou rovníc. [1]

Pri riešení úloh lineárneho programovania rozlišujeme ekonomický a matematický model. Ekonomický model úloh lineárneho programovania predstavuje slovný a číselný popis daného problému, ktorý popisuje:

- cieľ analýzy (napr. maximalizácia zisku, minimalizácia nákladov, apod.)
- procesy, ktoré majú vplyv na cieľ analýzy
- činitele ovplyvňujúce realizáciu procesov
- vzťahy medzi procesmi, činiteľmi a cieľom.

Aby bolo riešenie ekonomického modelu uskutočniteľné, je treba zostaviť odpovedajúci matematický model, ktorý je následne možné riešiť štandardnými postupmi lineárneho programovania. Matematický model má rovnakú štruktúru ako ekonomický model:

- cieľ analýzy je vyjadrený ako lineárna funkcia $z = f(x)$, ktorej extrém (minimum alebo maximum) treba nájsť.
- procesy - každému procesu z ekonomického modelu je priradená jedna premenná nazývaná štruktúrna premenná
- činiteľom ovplyvňujúcim realizáciu procesov odpovedajú v matematickom modeli lineárne rovnice či nerovnice [9], [11]

Obečne je možné matematický model úloh lineárneho programovania zapísať nasledovne:

max/min

$$z = c_1x_1 + c_2x_2 + \dots + c_nx_n$$

za podmienok

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2$$

$$\vdots \qquad \qquad \qquad \vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m$$

$$x_j \geq 0 \qquad j = 1, 2, \dots, n$$

Prvá rovnica vyjadruje účelovú funkciu. Sústava rovníc (nerovnic) popisuje obmedzujúce podmienky. Kde:

n – počet štruktúrnych premenných modelu

m – počet vlastných obmedzení

$c_j, j = 1, 2, \dots, n$ – cenový koeficient príslušnej j – tej premennej

$b_i, i = 1, 2, \dots, m$ – hodnota pravej strany príslušnému j – tému obmedzeniu

$a_{ij}, i = 1, 2, \dots, m, j = 1, 2, \dots, n$ – štruktúrny koeficient vyjadrujúci vzťah medzi i - tým činiteľom a j - tým procesom

Zápis matematického modelu je možné nahradiť aj maticovým zápisom:

max/mix

$$z = c^T x$$

za podmienok

$$Ax \leq b$$

$$x \geq 0$$

kde:

$c^T = (c_1, c_2, \dots, c_n)$ je n – zložkový riadkový vektor cenových koeficientov

$x = (x_1, x_2, \dots, x_n)^T$ je n – zložkový stĺpcový vektor štruktúrnych premenných

$A (m, n)$ je matica sústavy obmedzujúcich podmienok o rozmere $m \times n$

$b = (b_1, b_2, \dots, b_m)^T$ je m – zložkový stĺpcový vektor hodnôt pravej strany

$0 = (0, 0, \dots, 0)^T$ je n – zložkový stĺpcový nulový vektor [8], [10]

Formuláciú úlohy lineárneho programovania priblíži nasledovný príklad:

Firma Valentini vyrába štyri druhy kávových zmesí – Caffè Gusto, Caffè Gold, Caffè Dolce a Espresso. Na výrobu kávových zmesí využívajú štyri druhy kávových zŕn. Výberová Arabica, výberová Robusta, Arabica Santos a Robusta z Indie. Firma má k dispozícii 12000 kg výberovej zmesi Arabica, 5000 kg výberovej zmesi Robusta, 11000 kg kávy Arabica Santos a 1600 kg Robusty z Indie. Skladbu zmesi zobrazuje tabuľka 1. Aké je optimálne využitie zmesi, aby bol zisk firmy čo najvyšší, ak predaj 1 kg kávy Caffè Gusto prináša zisk 4 eur, 1 kg kávy Caffè Gold 5 eur, 1 kg kávy Caffè Dolce 3 eurá a predaj 1 kg kávy Espresso zisk vo výške 6 eur.

druh kávy/zloženie zmesi	Caffè Gusto	Caffè Gold	Caffè Dolce	Espresso	dostupné množstvo
výberova Arabica	70	90			12000
výberova Robusta	30	10			5000
Arabica Santos			80	90	11000
Robusta z Indie			20	10	1600

Tabuľka 1. Zloženie kávových zmesí.

Celkový zisk firmy sa dá zapísať pomocou lineárnej funkcie, ktorá sa nazýva účelová funkcia a jej cenové koeficienty:

$$z = 4x_1 + 5x_2 + 3x_3 + 6x_4$$

Na výrobu jednej jednotky zmesi Caffè Gusto sa spotrebuje 0,7 jednotky a na výrobu zmesi Caffè Gold sa použije 0,9 jednotky výberovej Arabiky. K dispozícii je 12 000 kg výberovej Arabiky. Celková spotreba výberovej Arabiky je daná súčtom oboch kávových zmesí. Ich súčet však môže byť rovný alebo menší ako jej dostupné množstvo. Vyjadrením tejto podmienky je lineárna nerovnica, ktorá sa zapíše v tvare:

$$0,7 x_1 + 0,9 x_2 \leq 12\,000$$

Obdobným spôsobom sa zapíšu ostatné nerovnice, vyjadrujúce vzťah medzi spotrebou a kapacitou kávových zŕn. Ich zápisom vznikne sústava lineárnych nerovnic. Tieto nerovnice (rovnice) sú označované ako vlastné obmedzenia úlohy lineárneho programovania. Vystupujú v nich dva druhy premenných. Štruktúrne premenné, ktoré popisujú vzťah medzi činiteľmi a procesmi, druhými premennými sú hodnoty pravej strany, ktoré definujú absolútnu úroveň činiteľov. Okrem vlastných obmedzení existuje aj podmienka nezápornosti, ktorá určuje, že rozhodovacie premenné musia byť nezáporné. Tieto premenné nemôžu nadobúdať mínusové

hodnoty, pretože v reálnych aplikáciách, ako sú výroba, distribúcia alebo financie, nemá zmysel, aby hodnoty týchto premenných boli záporné. Kompletnú podobu matematického modelu úlohy výroby kávy, nájdenia extrémnej účelovej funkcie, pri rešpektovaní vlastných obmedzení a podmienok nezápornosti, je možné zapísať nasledovne:

maximalizovať

$$z = 4x_1 + 5x_2 + 3x_3 + 6x_4$$

za podmienok

$$0,7x_1 + 0,9x_2 \leq 12\,000$$

$$0,3x_1 + 0,1x_2 \leq 5\,000$$

$$0,8x_3 + 0,9x_4 \leq 11\,000$$

$$0,2x_3 + 0,1x_4 \leq 1\,600$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0$$

1.3 Typické úlohy lineárneho programovania

Typické úlohy lineárneho programovania sú využívané v rôznych oblastiach. Príklady je možné nájsť vo výrobe, logistike, plánovaní, financovaní, ekonómii a mnohých ďalších oblastiach. Lineárne programovanie rieši širokú škálu optimalizačných problémov, pri ktorých je potrebné maximalizovať alebo minimalizovať cieľovú funkciu, pri splnení určitých lineárnych obmedzení.

Úlohy výrobného plánovania:

Úlohy výrobného plánovania sa zaoberajú otázkou, ako optimalizovať štruktúru výrobného programu. Aký sortiment v akom množstve vyrábať a aké výrobné postupy použiť, aby bola dosiahnutá maximalizácia či minimalizácia želaných ukazovateľov, ako napríklad maximalizácia zisku, minimalizácia prevádzkových nákladov, optimálne vytiaženie zamestnancov. Do úvahy treba brať všetky obmedzujúce faktory, ako limitované výrobné kapacity, technológie, požiadavky odberateľov a iné. Účelová funkcia obyčajne maximalizuje zisk a obmedzenia sú v tvare menší alebo rovný. [18]

Zmiešavací problém

Zmiešavací problém (známy aj ako problém miešania alebo mixovania) je špecifickým typom úloh v lineárnom programovaní. Zaoberá sa optimálnym zložením komponentov alebo surovín,

s cieľom namiešať takú zmes, aby boli dosiahnuté požadované vlastnosti finálneho produktu pri minimálnych nákladoch alebo maximalizácii zisku. Tento typ problému sa často vyskytuje v priemyselných odvetviach ako výroba palív, chemikálií alebo potravín. Príkladom môže byť výroba krmív pre zvieratá, s cieľom nájsť optimálnu kombináciu surovín, ktorá splní požiadavky na obsah živín a zároveň minimalizuje náklady na výrobu krmiva, alebo problém miešania palív, so snahou vyrobiť určitý typ paliva, ktorý spĺňa požiadavky na obsah chemických zložiek, pričom sa minimalizujú náklady na suroviny.

Medzi zmiešavacie problémy lineárneho programovania patria aj úlohy o výžive, nutričný problém. Jeho cieľom je určiť optimálne zloženie dennej dávky výživy jedinca, ktorá sa skladá z celej rady komponent. Tieto úlohy sú obvykle minimalizačné (čo najnižšie náklady), prípadne maximalizačné (čo najvyšší prísun vápniku). Obmedzujúce podmienky sú definované požiadavkami, ktoré musia byť splnené pri výbere potravín, ako sú kalórie, proteíny, vitamíny alebo minerály. Jedným z prvých a najznámejších problémov lineárneho programovania je diétna úloha (problém americkej gazdinej). Problém formuloval v päťdesiatych rokoch 20. storočia George B. Dantzig a išlo o jedno z prvých využití matematických metód k riešeniu reálnych problémov. Cieľom úlohy bolo navrhnúť optimálnu stravu, ktorá by splnila minimálne nutričné požiadavky na základe daných potravín a zároveň minimalizovala náklady na túto stravu. [12]

Distribučné problémy

Distribučné alebo logistické problémy sú typom lineárneho programovania, ktoré sa zameriavajú na optimalizáciu distribúcie tovaru medzi výrobcom a konečným zákazníkom, alebo distribučnými centrami. Zohrávajú dôležitú úlohu v oblasti logistiky a sú základom pre mnohé rozhodnutia v oblastiach dopravy, správy zásob a optimalizácie distribučných procesov. K distribučným problémom patrí dopravný problém (problém minimálnych nákladov), s cieľovou funkciou minimalizovať náklady na dopravu. Jedným z najznámejších a najdôležitejších problémov je problém obchodného cestujúceho (Travelling Salesman Problem). Obchodný cestujúci má navštíviť n miest, každé miesto presne raz a na záver sa má vrátiť do mesta, kde cestu začal. Cieľom je minimalizovať cestovné náklady, teda nájsť najkratšiu možnú cestu tak. Tento problém sa radí medzi NP-ťažké problémy, čo znamená, že neexistuje žiadny efektívny algoritmus, ktorý by mohol vyriešiť tento problém v polynomiálnom čase. [12]

Úlohy o delení materiálu

Problémy delenia materiálu alebo aj rezný plán, riešia, ako čo najefektívnejšie rozdeliť materiál na požadované menšie časti, s cieľom minimalizovať odpad a náklady, pri dodržiavaní obmedzení na veľkosť a množstvo rezných častí. Využitie nachádza najmä v stavebníctve, kovovýrobe, textilnom či automobilovom priemysle.

Na riešenie úloh pre menšie a stredné problémy delenia materiálu je vhodné dynamické programovanie, pre väčšie problémy s komplexnejšími požiadavkami a obmedzeniami sa osvedčujú metódy celočíselného programovania. [12], [18]

Úlohy finančného plánovania

Medzi štandardné úlohy lineárneho programovania patrí aj finančné plánovanie, alebo tiež optimalizácia portfólia. Ide o jeden z hlavných problémov v oblasti financií a investícií, zameriavajúci sa na správu a optimalizáciu investičného portfólia tak, aby bol dosiahnutý optimálny pomer medzi rizikom a výnosom. Úlohy môžu byť ako maximalizačné tak i minimalizačné. Obmedzujúce podmienky sa môžu líšiť v závislosti na cieľoch investora. Okrem rozpočtového obmedzenia a nezáporných podmienok, sú najbežnejšie podmienky týkajúce sa výnosu, rizika, likvidity, či regulačných, etických a sociálnych obmedzení. [18]

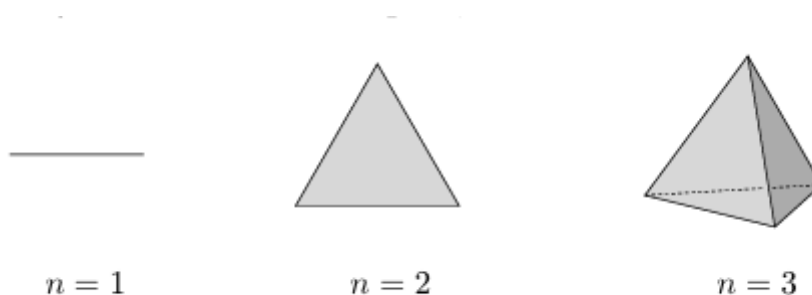
Marketingové aplikácie úloh lineárneho programovania

Lineárne programovanie je silný nástroj aj na optimalizáciu marketingových rozhodnutí. Konkrétnym príkladom je plánovanie reklamy (media selection problem), v ktorom ide o efektívne rozdelenie rozpočtu na reklamu medzi rôzne mediálne kanály, ako sú televízia, rozhlas, noviny, internet a ďalšie. Cieľom je maximalizovať úžitok z reklamy pri danom rozpočte, pričom sa využívajú rôzne ukazovatele výkonnosti, ako je napríklad gross rating point (udáva, koľkokrát sa v priemere stretne s reklamou jeden jednotlivec z cieľovej skupiny). [18]

2 SIMPLEXOVÁ METÓDA

Simplexová metóda (simplexový algoritmus), je jednou z najznámejších a najpoužívanějších metód pri riešení úloh lineárneho programovania. Prvý raz ju opísal v rokoch 1947 - 1949 americký matematik George B. Dantzig. Simplexová metóda bola pôvodne navrhnutá ako iteratívny algoritmus, ktorý sa pohybuje medzi rôznymi riešeniami v poli, ktoré zodpovedajú vrcholom polyedru, kde každý vrchol reprezentuje možné riešenie. Dantzig ukázal, že optimálne riešenie sa vždy nachádza na jednom z týchto vrcholov, a preto stačí prechádzať medzi vrcholmi, až kým nenájdeme optimálne riešenie.

Názov metódy je odvodený od slova simplex, ktorý v matematike vyjadruje základnú formu geometrického tvaru v n rozmernom priestore, množinu $n+1$ bodov (vrcholov), pričom každá dvojica vrcholov je spojená hranou.

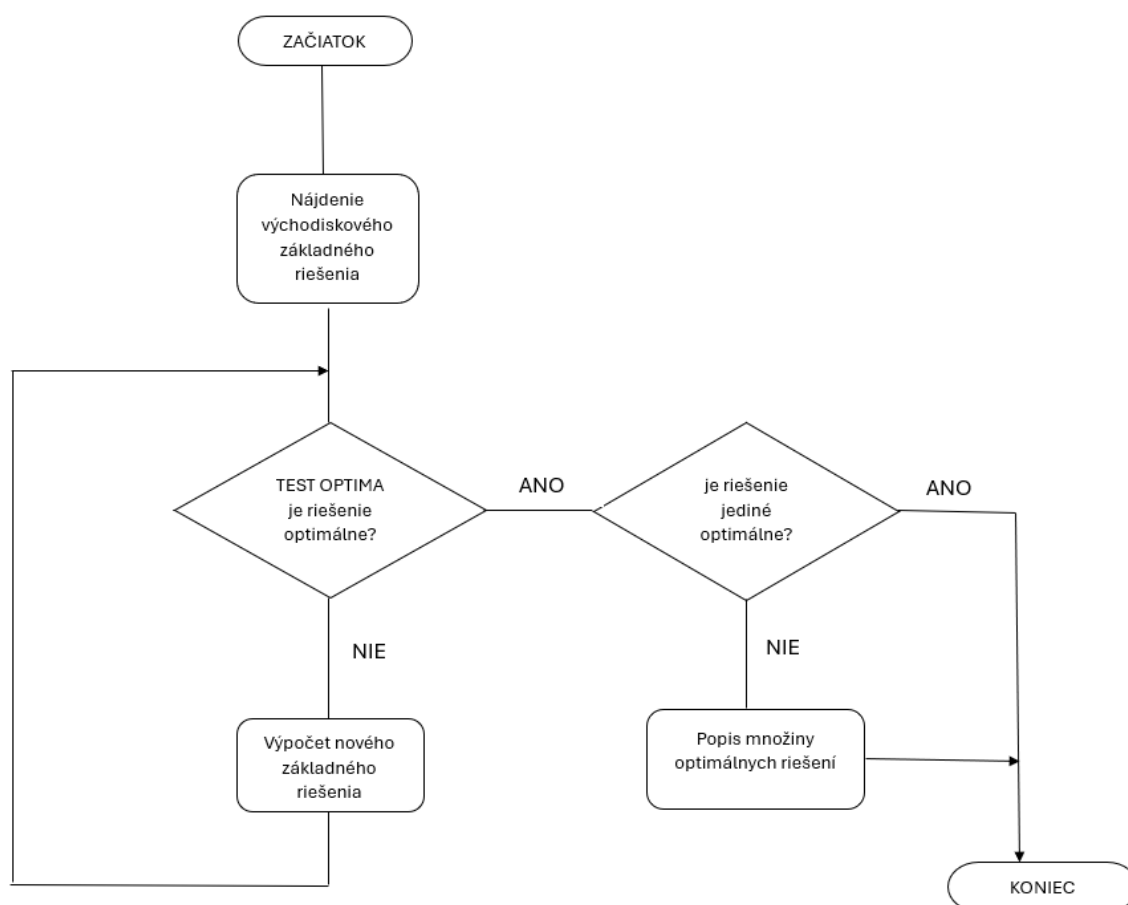


Obrázok 1. n rozmerný priestor

V kontexte simplexovej metódy predstavuje simplex polyéder - množinu možných riešení. Simplexový algoritmus iteratívne prechádza vrcholmi polyédru a hľadá optimálny bod, ktorý maximalizuje alebo minimalizuje cieľovú funkciu. Na začiatku algoritmu je zvolený ľubovoľný počiatočný vrchol, východiskový bod. V tomto bode sa zráta hodnota účelovej funkcie. Z vybraného bodu sa posúva po jeho hranách a skúma sa, ako sa vyvíjajú hodnoty účelovej funkcie v jednotlivých bodoch. Tento krok sa opakuje, pokiaľ je možné hodnotu zvyšovať. Klesajúce hodnoty napovedajú, že sa ide zlým smerom a treba zvoliť hranu, na ktorej hodnoty účelovej funkcie rastú. Ak sa natrafi na ďalší vrchol, opätovne sa zráta hodnota účelovej funkcie a hľadá sa hrana, na ktorej bude jej hodnota rásť. Ak sa cestou po hrane na žiaden vrchol nenatrafi, účelová funkcia môže nadobúdať nekonečne veľkých hodnôt. Ak sa narazí na vrchol, z ktorého nevedú žiadne hrany, na ktorých sa hodnota účelovej funkcie zvyšuje, našlo sa

optimálne riešenie úlohy. Účelová funkcia tú nadobúda maximálnu hodnotu a leží v množine prípustných bodov. [8]

Postup výpočtu simplexovou metódou sa delí na dve fázy. Prvou fázou je nájdenie východiskového základného prípustného riešenia. K tomu je potrebné, aby boli obmedzujúce podmienky v tvare sústavy lineárnych nerovníc, prevedené do tvaru sústavy lineárnych rovníc, do tzv. kanonického tvaru. V druhej fáze je dôležitým krokom testovanie optimálnosti aktuálneho riešenia, ktoré slúži na preverenie, či prvé prípustné riešenie už je alebo ešte nie je optimálne. Ak test optimality ukáže, že riešenie je optimálne, výpočet končí. V prípade, že aktuálne riešenie nie je optimálne, hľadá sa riešenie s vyššou hodnotou účelovej funkcie (maximalizačná úloha), prípadne riešenie s nižšou hodnotou účelovej funkcie (minimalizačná úloha). Test optimality sa opakuje až pokiaľ nebude nájdené riešenie optimálne. (Obrázok 2).



Obrázok 2. Schéma simplexovej metódy [8]

2.1 Simplexová tabuľka

Na riešenie úloh sa v praxi pre prehľadnosť a zjednodušenie výpočtu osvedčuje postup riešenia pomocou simplexovej tabuľky. Simplexová tabuľka predstavuje spôsob, ako systematicky organizovať a sledovať výpočty počas iterácií simplexovej metódy. Umožňuje sledovanie hodnôt premenných a koeficientov pri každej iterácii algoritmu, rýchlo a prehľadne identifikovať, ktoré premenné sú v základnom riešení (t.j. majú nenulové hodnoty) a ktoré nie, vykonávať pivotovanie z jedného základného riešenia na iné, ako aj určiť, kedy je riešenie optimálne.

Jednoduchá simplexová tabuľka zahŕňa niekoľko kľúčových častí. Jej formálna úprava je vecou dohody a nemá vplyv na výsledok výpočtu. Obecné riadky tabuľky odpovedajú obmedzeniam a stĺpce premenným.

	x_1	...	x_n	x_{n+1}	...	x_{n+m}	z	b_i
x_{n+1}	a_{11}		a_{1n}	1		0	0	b_1
...	...		...					...
...	...		...					...
x_{n+m}	a_{m1}		a_{mn}	0		1		b_m
z_j	c_1		c_n	0		0	1	0

Tabuľka 2. Schéma východiskovej simplexovej tabuľky.

Tabuľka 2 schematicky znázorňuje sústavu rovníc, kde n je počet štruktúrnych premenných a m je počet vlastných obmedzení. Stĺpce tabuľky sú označené jednotlivými premennými. Na začiatku riadku je vždy uvedená základná premenná. Premenné x_{n+1} až x_{n+m} v prvom stĺpci sú základne premenné, ich hodnota je rovná pravým stranám obmedzení b_1 až b_m . Premenné x_1 až x_n sú nezákladné premenné a sú rovné nule. V poslednom riadku sa nachádza účelová funkcia a jej koeficienty sú anulované. [12]

Na vysvetlenie algoritmu a postupu riešenia pomocou simplexovej tabuľky poslúži nasledovný príklad so zadaním:

maximalizovať

$$z = 10x_1 + 8x_2 + 12x_3$$

za podmienok

$$x_1 + x_2 + x_3 \leq 100$$

$$2x_1 + x_2 + 2x_3 \leq 150$$

$$x_1 + 3x_2 + x_3 \leq 120$$

$$x_j \geq 0, j=1, 2, 3$$

Úloha sa prevedie do kanonického tvaru a zapíše do simplexovej tabuľky. (Tabuľka 3).

Obmedzenia modelu sa vyrovnajú na rovnice.

$$x_1 + x_2 + x_3 + x_4 = 100$$

$$2x_1 + x_2 + 2x_3 + x_5 = 150$$

$$x_1 + 3x_2 + x_3 + x_6 = 120$$

$$x_j \geq 0, j=1, 2, \dots, 6$$

Premenné účelovej funkcie budú prevedené na ľavú stranu. Anulovaná účelová funkcia:

$$z - 10x_1 - 8x_2 - 12x_3 = 0$$

	x_1	x_2	x_3	x_4	x_5	x_6	b_i
x_4	1	1	1	1	0	0	100
x_5	2	1	2	0	1	0	150
x_6	1	3	1	0	0	1	120
z_j	-10	-8	-12	0	0	0	0

Tabuľka 3. Východisková simplexová tabuľka

Nezákladné premenné sú x_1 , x_2 a x_3 . Základné premenné predstavujú pridané doplnkové (sklzové) premenné x_4 , x_5 a x_6 .

Východiskové riešenie úlohy:

$$x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 100, x_5 = 150, x_6 = 120, z = 0$$

Iteračný postup simplexovej metódy začína od východiskového riešenia. Prvým krokom je test optimality a určenie vstupnej premennej, ktorá udáva kľúčový stĺpec a vystupujúcej premennej, ktorá udáva kľúčový riadok.

Vstupná premenná bude zvolená podľa hodnoty z_k , tzv. redukovanej ceny.

Pre maximalizáciu účelovej funkcie:

$$z_k = \min z_j, z_j < 0, j \in \mathbb{N}$$

Pre minimalizáciu účelovej funkcie:

$$z_k = \max z_j, z_j > 0, j \in \mathbb{N}$$

(koeficient z_j - v obecnom matematickom modeli označovaný ako c_j).

Obecne platí, že riešenie je optimálne, ak sú pri maximalizácii účelovej funkcie všetky redukované ceny nezáporné a pri minimalizácii účelovej funkcie všetky redukované ceny nekladné.

Výstupná premenná je minimálny podiel transformovaných hodnôt pravej strany (b_i) a kladných štruktúrnych koeficientov vstupujúcej premennej (kľúčový stĺpec).

$$\min \frac{\beta_i}{\alpha_{ik}}, i = 1, 2, \dots, m$$

$\alpha_{ik}, i = 1, 2, \dots, m, j = 1, 2, \dots, n$ sú transformované štruktúrne koeficienty, v obecnom matematickom modeli označené ako a_{ij}

$\beta_i, i = 1, 2, \dots, m$ sú transformované hodnoty pravej strany, v obecnom matematickom modeli označené ako b_i

$x_j, j = 1, 2, \dots, n$ sú hodnoty premenných [8]

V simplexovej tabuľke treba nájsť v riadku účelovej funkcie najmenší koeficient $-z_j$. V uvedenom príklade je najnižšia hodnota -12. Stĺpec x_3 je teda kľúčový stĺpec. Kľúčový riadok predstavuje najnižšia kladná hodnota, získaná ako podiel hodnoty b_i a hodnoty kľúčového stĺpca. Priesečníkom kľúčového stĺpca a kľúčového riadka sa získa kľúčový prvok. (Tabuľka 4).

	x_1	x_2	x_3	x_4	x_5	x_6	b_i	$\frac{\beta_i}{\alpha_{ik}}$
x_4	1	1	1	1	0	0	100	100
x_5	2	1	2	0	1	0	150	75
x_6	1	3	1	0	0	1	120	120
z_j	-10	-8	-12	0	0	0	0	

↑

←

Tabuľka 4. Simplexová tabuľka, určenie kľúčového stĺpca a riadku

Tabuľka sa prepočíta tak, aby v bola v kľúčovom stĺpci na mieste kľúčového prvku jednotka a ostatné prvky v kľúčovom stĺpci nulové.

Postup výpočtu:

- Kľúčový riadok sa vydolí kľúčovým prvkom, čím sa docielí, že na pozícii kľúčového prvku bude hodnota 1.
- Na ostatné riadky sa uplatní Gaussovu eliminačná metóda, aby boli na ostatných miesta kľúčového stĺpca nuly. To bude dosiahnuté tak, že nové hodnoty kľúčového riadku, (po delení kľúčovým prvkom), sa najskôr vynásobia vhodným číslom a následne prirátajú (odráťajú) od riadku, ktorý je upravovaný. Takto sa upraví celý riadok danej premennej.

Výsledkom je prepočítaná simplexová tabuľka. Opäť sa urobí test optimality. Ak sa v riadku účelovej funkcie nachádzajú len kladné hodnoty, bolo nájdené optimálne riešenie (maximalizačná úloha).

V uvedenom príklade sa v riadku účelovej funkcie nachádza záporná hodnota, čo naznačuje, že po prvej iterácii nebolo nájdené optimálne riešenie. V úlohe je preto potrebné pokračovať ďalšou iteráciou a zopakovať celý proces výberu kľúčového prvku a Gaussovej eliminácie.

Po druhej iterácii v riadku účelovej funkcie nenachádzajú žiadne záporné hodnoty, čo znamená, že bolo nájdené optimálne riešenie úlohy. (Tabuľka 5).

	x_1	x_2	x_3	x_4	x_5	x_6	b_i	$\frac{\beta_i}{\alpha_{ik}}$
x_4	1	1	1	1	0	0	100	100
x_5	2	1	2	0	1	0	150	75
x_6	1	3	1	0	0	1	120	120
z_j	-10	-8	-12	0	0	0	0	
			↑					
x_4	1	1	1	1	0	0	25	25
x_2	1/2	1/2	1	0	1/2	0	75	150
x_6	0	2	0	0	-1/2	1	45	-
z_j	-10	-8	0	0	6	0	900	
	↑							
x_1	1	1	1	1	0	0	25	
x_3	0	0	1	0	1/2	0	62,5	
x_6	0	0	0	0	-1/2	1/1	45	
z_j	0	0	0	0	15	0	1150	

Tabuľka 5. Výsledná simplexová tabuľka po druhej iterácii

Zo simplexovej tabuľky vyplýva, že optimálne riešenie je:

$$x_1 = 25$$

$$x_2 = 0$$

$$x_3 = 62,5$$

$$z_{max} = 1150$$

Na dosiahnutie maximálneho zisku vo výške 1150 jednotiek je najvýhodnejšie nepoužiť premennú x_2 vôbec, použiť premennú x_1 v množstve 25 jednotiek a premennú x_3 v množstve 62,5 jednotiek. Takto sa efektívne využijú dostupné zdroje tak, aby sa maximalizovala hodnota účelovej funkcie.

2.2 Dualita úloh lineárneho programovania

Každú maximalizačnú úlohu je možné pri splnení určitých pravidiel previesť na úlohu minimalizačnú a naopak. Tento vzájomný vzťah sa nazýva dualita. Ide o vzájomne symetrický matematický vzťah medzi primárnou úlohou a duálnou úlohou a spolu tvoria dvojicu duálne združených úloh. Dualita má značný význam v celej teórii lineárneho programovania a často umožňuje vhodnejšiu ekonomickú interpretáciu riešených úloh.

Každému primárnemu modelu úlohy lineárneho programovania môže byť priradený duálny model lineárneho programovania pri splnení určitých pravidiel:

- Ak je primárna úloha maximalizačná, duálna úloha bude minimalizačná a naopak.
- Počet premenných v duálnej úlohe je rovný počtu obmedzení v primárnej úlohe a počet obmedzení v duálnej úlohe je rovný počtu premenných v primárnej úlohe.
- Matica štruktúrnych koeficientov primárnej úlohy a matica štruktúrnych koeficientov duálnej úlohy sú navzájom transponované.
- Vektor pravých strán primárnej úlohy je vektorom cien duálnej úlohy a naopak.

Platí veta o reflexívnosti duality, ktorá hovorí, že duálna úloha k duálnej úlohe je pôvodná primárna úloha. Inak povedané, ak by sa aplikoval proces transformácie jednej úlohy na druhú dva krát po sebe, výsledkom by bola primárna úloha. Reflexívnosť duality umožňuje z vlastností riešenia primárnej úlohy určiť vlastnosti a riešenie duálnej úlohy a naopak. Základná veta o dualite hovorí, že ak má primárny model riešenie, má optimálne riešenie aj duálny model s rovnakou hodnotou účelovej funkcie.

V závislosti na tom, ako je zadaný primárny problém, môže sa duálny problém formulovať ako symetrický alebo nesymetrický. Ak sú všetky obmedzenia primárneho problému zadané v tvare nerovnice a podmienky nezápornosti sú definované na všetky premenné, hovorí sa o symetrickej dualite. O nesymetricky duálne združené úlohy sa jedná v prípade, ak sú všetky alebo niektoré obmedzenia primárnej úlohy v tvare rovnice. Pre duálne premenné priradené rovniciam v primárnom probléme ale nemusí platiť podmienka nezápornosti. [8], [9]

Existuje niekoľko prístupov, ako riešiť duálnu úlohu. Jednou z možností je získanie riešenia zo simplexovej tabuľky vyriešenej primárnej úlohy, pričom platí:

- Ak má primárna úloha jediné optimálne riešenie, potom má aj duálna úloha jediné optimálne riešenie a naopak
- Ak má primárna úloha alternatívne riešenia, optimálne riešenie duálnej úlohy je degenerované a naopak
- Ak primárna úloha nemá optimálne riešenie, to znamená, že má prípustné riešenia, ale účelová funkcia môže neobmedzene rásť alebo klesať, duálna úloha nemá žiadne prípustné riešenie. [12], [13]

Ďalšou možnosťou je riešenie duálnej úlohy pomocou simplexovej tabuľky, pričom sa využíva rovnaký simplexový algoritmus ako pri primárnej úlohe, avšak s úpravami, ktoré vyplývajú z duálnych podmienok.

Na počiatku riešenia sa formuluje duálny problém. Podľa pravidiel duality sa vytvorí matematický model duálnej úlohy, odvodený z primárnej úlohy.

Primárna úloha:

Maximalizovať

$$z = 10x_1 + 8x_2 + 12x_3$$

za podmienok

$$x_1 + x_2 + x_3 \leq 100$$

$$2x_1 + x_2 + 2x_3 \leq 150$$

$$x_1 + 3x_2 + x_3 \leq 120$$

$$x_j \geq 0, j=1, 2, 3$$

Ide o úlohu maximalizačnú a všetky štruktúrne obmedzujúce podmienky sú v tvare nerovností. Úloha má tri vlastné obmedzenia, preto bude mať duálna úloha tri duálne premenné s označením y_1, y_2, y_3 . Tieto premenné predstavujú cenové multiplikátory, ktoré indikujú, aký vplyv má zmena obmedzení na optimálny zisk. Ak je primárna úloha maximalizačná, duálna úloha bude formulovaná ako minimalizačná:

Minimalizovať

$$z = 100y_1 + 150y_2 + 120y_3$$

za podmienok

$$y_1 + 2y_2 + y_3 \geq 10$$

$$y_1 + y_2 + 3y_3 \geq 8$$

$$y_1 + 2y_2 + y_3 \geq 12$$

$$y_1, y_2, y_3 \geq 0$$

Anulovaná účelová funkcia má tvar:

$$z - 100y_1 - 150y_2 - 120y_3 = 0$$

Aby bolo možné použiť simplexovú metódu, úlohu musí byť prevedená do kanonického tvaru. Upraví sa nerovnosti na rovnosti, obmedzujúce podmienky sa prevedú do tvaru rovníc použitím pomocných premenných.

$$y_1 + 2y_2 + y_3 - y_4 = 10$$

$$y_1 + y_2 + 3y_3 - y_5 = 8$$

$$y_1 + 2y_2 + y_3 - y_6 = 12$$

$$y_1, y_2, y_3, y_4, y_5, y_6 \geq 0$$

Kanonický tvar predstavuje základné východiskové riešenie. Základné premenné, teda pomocné premenné musia byť kladné, preto je potrebné vynásobiť rovnice číslom mínus jeden.

$$z - 100y_1 - 150y_2 - 120y_3 = 0 \quad (\text{MIN})$$

$$-y_1 - 2y_2 - y_3 + y_4 = -10$$

$$-y_1 - y_2 - 3y_3 + y_5 = -8$$

$$-y_1 - 2y_2 - y_3 + y_6 = -12$$

$$y_1, y_2, y_3, y_4, y_5, y_6 \geq 0$$

V zostavenej simplexovej tabuľke sa stanoví kľúčový riadok. Určuje ho najmenšia hodnota pravých strán obmedzujúcich podmienok ($\min b_i$).

- ak je $b_i > 0$, riešenie je aj primárne prístupné, teda optimálne
- ak je $b_i = 0$, riešenie je optimálne, ale degenerované
- ak je $b_i < 0$, pokračuje sa ďalej vo výpočte a najnižšia hodnota $b_i < 0$ je uvažovaná ako kľúčový riadok

Kľúčový stĺpec predstavuje najnižšia hodnota, získaná ako podiel hodnoty účelovej funkcie z_j a hodnoty kľúčového riadku (uvažujú sa len záporné hodnoty). Priesečníkom kľúčového stĺpca a kľúčového riadka sa získa kľúčový prvok. V riešení sa pokračuje ďalej prepočítaním tabuľky úplnou elimináciou, rovnako ako pri primárnej úlohe. [4],[13]

3 CELOČÍSELNÉ LINEÁRNE PROGRAMOVANIE

Celočíselné lineárne programovanie (Integer Linear Programming) vzniklo ako reakcia na problémy, ktorých riešenia musia nadobúdať iba celočíselné hodnoty. Takéto problémy sa bežne vyskytujú v ekonómii, logistike, plánovaní výroby, a mnohých ďalších oblastiach, kde nie je možné použiť zlomkové riešenia. Základy celočíselného lineárneho programovania vychádzajú z klasického lineárneho programovania, ktoré bolo formálne zavedené v roku 1947 americkým matematikom Georgeom Dantzigom prostredníctvom simplexovej metódy. Hoci simplexová metóda poskytovala efektívny spôsob riešenia LP úloh, nebola priamo použiteľná na úlohy s celočíselnými obmedzeniami.

Skutočný rozmach celočíselného lineárneho programovania nastal koncom 50. rokov 20. storočia, keď Ralph E. Gomory, vyvinul prvé všeobecné metódy na riešenie celočíselných úloh. V roku 1958 predstavil Gomoryho rezné roviny, ktoré umožňovali upraviť riešenia lineárneho programovania tak, aby spĺňali celočíselnosť, čím položil základy samostatného smeru výskumu. Súčasne sa vyvíjala aj metóda vetiev a medzí, ktorú neskôr systematicky rozpracoval Egon Balas. Táto metóda sa stala kľúčovým nástrojom pri riešení rozsiahlych celočíselných problémov a je využívaná dodnes. [5]

Úlohy celočíselného programovania vychádzajú z úloh lineárneho programovania, a rovnako hľadajú optimálnu hodnotu účelovej funkcie, pomocou premenných a obmedzení zadaných v tvare rovností alebo nerovností. K podmienke nezápornosti sa pridáva i podmienka celočíselnosti a v matematickom modeli ju označujeme ako $x \in Z$.

Matematický zápis:

$$\begin{array}{ll} \text{max/min} & z = c^T x \\ \text{za podmienok} & Ax \leq b \\ & x \geq 0 \\ & x \in Z^n \end{array}$$

Premenná z označuje hodnotu účelovej funkcie, vektor c je vektor cenových koeficientov o veľkosti $1 \times n$, A matica štruktúrnych koeficientov o rozmere $m \times n$, kde m je počet vlastných obmedzení úlohy a n počet premenných, b vektor pravých strán o veľkosti $m \times 1$ a x vektor premenných modelu.

O celočíselnom lineárnom programovaní sa hovorí v prípade, ak je podmienka celočíselnosti uplatnená na všetky premenné. Ak je táto podmienka požadovaná len u niektorých premenných, ide o zmiešané celočíselné programovanie. Špeciálnymi úlohami sú binárne úlohy, ktoré musia spĺňať podmienku binarity, kde sú hodnoty premenných obmedzené len na 0 a 1. Označujú sa ako úlohy bivalentného programovania.

Úlohy celočíselného lineárneho programovania sú náročnejšie ako štandardné úlohy lineárneho programovania a na ich riešenie boli vyvinuté špeciálne metódy a algoritmy, vyhovujúce pre tento typ úloh. Každá metóda je vhodná sa pre iný typ problému a podľa charakteru je možné ich rozdeliť na metódy:

1. empirické - najčastejšie sa využíva metóda celočíselného zaokrúhľovania a enumeratívna metóda. Pri zaokrúhľovaní však býva často riešenie výrazne vzdialené od hľadaného celočíselného riešenia. V praxi je vhodné túto metódu využívať len v prípadoch, ak je chyba prijateľná.
2. exaktné – je možné ich rozdeliť do dvoch skupín, ktoré predstavujú hlavné spôsoby riešenia celočíselných úloh:
 - Metódy rezných rovín - vychádzajú z relaxácie problému a postupne zužujú oblasť možných riešení pridávaním ďalších obmedzení - rezných rovín, ktoré vylučujú neceločíselné riešenia, (Gomoryho metóda).
 - Kombinatorické metódy – riešia problém diskretným spôsobom, (metóda vetiev a medzí).
3. Špeciálne - patria sem heuristické metódy, ktoré neposkytujú zaručene optimálne riešenie, ale hľadajú riešenie blízke optimu. [4], [13]

3.1 Metódy rezných rovín

Metódy rezných rovín, (Cutting Plane method), sú vhodné pre riešenie zmiešaných i čisto celočíselných úloh. Vychádzajú z množiny prístupných riešení neceločíselnej (relaxovanej) úlohy. S použitím simplexového algoritmu sa získa optimálne riešenie. Ak je neceločíselné, slúži ako východiskové riešenie pre metódu rezných rovín. Samotná metóda následne pridáva dodatočné obmedzujúce podmienky, nazývané rezné roviny. Doplnková podmienka vymedzí priestor ohraničený rovinou, ktorá z množiny prípustných riešení odsekne podmnožinu neceločíselných riešení. V takto zúženej množine prípustných riešení sa opäť určí optimálne

riešenie a testuje sa, či už sa jedná o celočíselné riešenie alebo ešte nie. Úloha s doplnkovou obmedzujúcou podmienkou sa rieši rovnakým spôsobom ako pôvodná úloha. [13].

Rezné roviny je možné vytvoriť viacerými spôsobmi, pričom najznámejší z nich je Gomoryho algoritmus, ktorý sa vyznačuje lepšou schopnosťou orezávať prípustný priestor riešení. Gomoryho rezy využívajú dolnú celú časť reálneho čísla x , čo je najväčšie celé číslo, ktoré nie je väčšie ako x . Označujeme ho $\lfloor x \rfloor$.

Gomoryho algoritmus je možné zhrnúť do nasledovných krokov:

1. S využitím simplexovej metódy sa určí optimálne riešenie úlohy, bez ohľadu na podmienku celočíselnosti. Algoritmus odvodzuje reznú rovinu zo simplexovej tabuľky. Vychádza z takého jej i -teho riadku, ktorý má na pravej strane neceločíselnú hodnotu b_i . Zvolený i -ty riadok reprezentuje rovnica v tvare

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i$$

2. Riešenie sa rozdelí na celé a necelé časti, t.j. výsledok pravej strany b_i sa rozdelí na celočíselnú časť d_i a zlomkovú časť r_i . Platí

$$a_{ij} = \lfloor a_{ij} \rfloor + r_j b_i$$

$$b_i = \lfloor b_i \rfloor + r$$

$$b_i = d_i + r_i$$

3. Vyberie sa najväčší neceločíselný zvyšok v riadku i_0 . Zvolí sa rovnica i , v ktorej sa vyžaduje celočíselné riešenie a v ktorej je najväčšia hodnota r_i . Rozložia sa koeficienty ľavej strany na celočíselné časti d_{ij} a zlomkové časti r_{ij} .

$$a_{ij} = d_{ij} + r_{ij}$$

4. Definícia ďalšej účelovej funkcie. K rovnici vybranej v bode 3 sa vytvorí doplnková rovnica, rozšírená o novú prídavnú premennú s .

$$-r_{i1}x_1 - r_{i2}x_2 - \dots - r_{in}x_n + s = -r_i$$

5. Simplexová tabuľka sa rozšíri o riadok premennej s . Tento riadok je považovaný za kľúčový riadok. Kľúčový stĺpec predstavujú stĺpec s najnižšou hodnotou vypočítanou ako podiel z_j / r_{ij} .

$$\min |z_j / r_{ij}|$$

6. V ďalšom kroku nasleduje samotné riešenie tabuľky. Ak je riešenie celočíselné pre všetky premenné, výpočet sa končí. Ak zostanú niektoré hodnoty neceločíselné, opakuje sa postup od druhého kroku a sústava sa rozšíri o ďalšiu rovnicu s novou premennou s_2 . [13], [18]

3.2 Metóda vetiev a medzí

Metóda vetiev a medzí (Branch and Bound) patrí k najpoužívanejším kombinatorickým metódam pre celočíselné úlohy. Kľúčovú rolu v jej vývoji zohrali autorky Land a Doig, keď v roku 1960 publikovali článok, v ktorom použili princípy vetiev a medzí. O pár rokov neskôr ich prácu rozšíril o efektívne techniky britský matematik Robert J. Dakin. Samotnú metódu Branch and Bound pomenovali Little et al. v roku 1963, pri riešení problému obchodného cestujúceho. [7]

Hlavnou myšlienkou tejto metódy je „vetviť“ úlohu na niekoľko menších úloh a následne riešiť každú zvlášť. Množina všetkých prípustných riešení sa rozdelí na menšie podmnožiny (vetvy), v ktorých sa hľadá optimálne riešenie, pričom sa vyradujú nevhodné vetvy. Ak sa v danej vetve optimálne riešenie nenájde, delí na ďalšie podmnožiny. Pri výpočte sú využívané horné a dolné medze, ktoré sa v priebehu výpočtu spresňujú, v snahe priblížiť sa čo najviac k optimu. Pomocou medzí sa určuje, či sa v danej vetve nachádza lepšie či horšie riešenie a výpočet pokračuje vetvením podmnožiny s lepšou hodnotou.

Všeobecný postup metódy vetiev a medzí je možné zhrnúť v nasledovných krokoch:

1. Vypočíta sa optimálne riešenie úlohy, napríklad s použitím simplexovej metódy, pričom sa neberie do úvahy podmienka celočíselnosti. Vektor optimálneho riešenia $x^0 = (x_1, x_2, \dots, x_n)$ a hodnota účelovej funkcie z^0 . Ak je riešenie celočíselné, úloha končí. Ak riešenie nie je celočíselné, uvažuje sa hodnota riešenia za hornú medzu a celá časť tejto hodnoty ako dolná medza.
2. Pristúpi sa k vetveniu množiny prípustných riešení. Množina X^0 sa rozdelí na dve podmnožiny X^1 a X^2 . Z vektora x^0 sa vyberie ľubovoľná neceločíselná premenná, napr. x_k , ktorej hodnota vo vektore x^0 je rovná x_k^0 . Množina X^1 je rozšírená o podmienku

$$x_k \geq [x_k^0] + 1$$

Množina X^2 vznikne doplnením množiny X^2 o podmienku

$$x_k \leq [x_k^0]$$

výraz $[x_k^0]$ predstavuje celú časť z neceločíselnej hodnoty x_k^0 .

3. V oboch vetvách sa vyráta optimálne riešenie, bez uvažovania podmienky celočíselnosti. Nájdené optimálne riešenie je hornou medzou pre daný uzol.
4. Proces vetvenia pokračuje, pokiaľ nie je nájdené celočíselné riešenie a následne sa porovnáva s hornou medzou. Riešenie s najvyššou hodnotou účelovej funkcie je riešenie optimálne. Ak vo vetve nebolo nájdené prípustné celočíselné riešenie, vetvenie pokračuje ďalej. Každá z vetiev môže byť ukončená nasledovným spôsobom:
 - Vo vetve je dosiahnuté celočíselné riešenie
 - Vo vetve neexistuje žiadne prípustné riešenie
 - Vo vetve je nájdené neceločíselné riešenie, ale horná medza pre hodnotu účelovej funkcie tohoto riešenia je nižšia ako hodnota účelovej funkcie celočíselného riešenia nájdeného v niektorej z ostatných vetiev.

Po uzatvorení všetkých vetiev proces vetvenia končí. Optimálnym riešením je najlepšie nájdené celočíselné riešenie (ak existuje). [4], [6], [7]

4 NÁSTROJE NA RIEŠENIE ÚLOH CELOČÍSELNÉHO LINEÁRNEHO PROGRAMOVANIA

Riešenie úloh lineárneho programovania je vo všeobecnosti náročná a zdĺhavá činnosť. Preto boli vyvinuté mnohé programy, ktoré si v oveľa kratšom čase poradia s množstvom obmedzení, pri hľadaní optimálnej hodnoty danej úlohy. Programová ponuka je široká a výber vhodnej podpory závisí od viacerých faktorov. Či už od rozsahu a náročnosti riešeného problému, licenčných možností a dostupnosti programovej podpory, úrovne používateľa, alebo účelu a kontextu použitia.

Programová podpora pre úlohy lineárneho programovania sa delí na dve hlavné skupiny. Na modelovacie nástroje (jazyky a rozhrania) a solvery (riešiče).

4.1 Modelovacie jazyky

Modelovacie nástroje slúžia na vytváranie a formulovanie optimalizačných modelov. S ich pomocou môže používateľ definovať rozhodovacie premenné, množinu obmedzení a cieľovú funkciu, ktorú chce maximalizovať alebo minimalizovať. Modelovacie jazyky majú vlastnú syntax a sú navrhnuté špeciálne pre optimalizáciu. Úlohou týchto nástrojov je zjednodušiť zápis modelu a poskytnúť užívateľské rozhranie pre solver. Rozlišujú sa uzatvorené systémy, kde sa spája modelovací jazyk a vlastným integrovaným solverom. Väčšinou ide o komerčné softvéry, menej otvorené iným jazykom či prostrediam. Príkladom je modelovací jazyk LINGO s implementovaným riešiteľom Lindo. FICO Xpress Optimization s riešiteľom Xpress, napísaný v Mosel jazyku. Alebo IBM ILOG CPLEX Optimization Studio, skrátene CPLEX Optimization Studio, prostredie vyvinuté k solveru CPLEX, s modelovacím jazykom OPL, Optimization Programming Language. Ďalšou možnosťou sú modelovacie jazyky nezávislé od solvera. Poskytujú len modelovací jazyk alebo rozhranie a vďaka ich väčšej flexibilitě pri voľbe solvera, takzvané otvorené systémy. Patrí sem GAMS (General Algebraic Modeling System), AMPL (A Mathematical Programming Language) alebo AIMMs (Advanced Interactive Multidimensional Modeling System).

K modelovacím nástrojom patria aj programovacie knižnice, ktoré umožňujú písať kódy optimalizačných modelov v bežných programovacích jazykoch. Najuniverzálnejším jazykom v optimalizácii je Python, ktorý podporuje väčšinu solverov a poskytuje množstvo dostupných knižníc ako napríklad PuLP, Pyomo, OR tools. Využívaný je aj jazyk C/C++, najmä pre náročné úlohy, pre svoj vysoký výkon a rýchlosť výpočtov, vhodný na vývoj solverov. Pre

pokročilé optimalizačné úlohy sa využíva moderný programovací jazyk Julia, ktorý kombinuje výhody viacerých jazykov. Je navrhnutý práve pre optimalizáciu a numerickú matematiku a podobne ako Python, poskytuje jednoduchú syntax a výkonnosťou je porovnateľný s C/C++. Pre svoju stabilitu a robustnosť je vhodným a často používaným aj jazyk Java. Podporuje solvery Gurobi či CPLEX a často sa využíva pre podnikové aplikácie s integrovanou optimalizáciou.

4.2 Optimalizačne solvery

Optimalizačný solver alebo riešič je softvérový nástroj, ktorý pomocou výpočtových algoritmov rieši zadanú úlohu. Solver sa dá volať cez modelovací jazyk (napr. AMPL, JuMP, Pyomo), alebo sa používa priamo cez API (Application Programming Interface), rozhranie, ktoré umožňuje komunikáciu priamo pomocou programovacieho jazyka. I pri solveroch treba zvažovať náročnosť a účel riešenej úlohy, ako aj dostupnosť samotného solvera. Medzi najvýkonnejšie open-source solvery patrí CBC (Coin-or Branch and Cut), vhodný pre menšie až stredné úlohy. Na náročnejšie úlohy je vhodný výkonný open-source solver SCIP (Solving Constraint Integer Programs), ktorý sa špecializuje na úlohy celočíselného programovania. Najpoužívanejšie a najvýkonnejšie komerčné solvery sú Gurobi, vhodné pre zložité úlohy, CPLEX, alebo Xpress. CPLEX patrí k najspoľahlivejším a na trhu najdlhšie využívaným optimalizačným solverom. Rovnako ako aj Gurobi, je podporovaný mnohými modelovacími a programovacími jazykmi. Ich používanie však vyžaduje istú mieru znalostí.

4.3 MS Excel a jeho doplnky

Ľahko dostupný nástroj na riešenie úloh celočíselného lineárneho programovania predstavuje Microsoft Excel. Ponúka jednoduché riešenie optimalizačných úloh pomocou solvera v podobe doplnku Riešiteľ. Vzhľadom na svoje možnosti i výkonnosť sa neodporúča pre rozsiahle a náročné úlohy, no jeho využitie poskytuje firmám a manažérom zrýchlenie zložitých výpočtov, či podklad k optimálnemu rozhodnutiu. Zároveň je veľmi jednoduchý pre pochopenie problematiky riešenia úloh a to vďaka možnosti analyzovať jednotlivé kroky riešenia.

Doplnok Riešiteľ je uzatvorený optimalizačný systém s vlastným solverom. Nevyužíva žiaden modelovací jazyk, ale pracuje priamo s bunkami Excelu. Je vhodný pre výuku a menšie úlohy, nakoľko sú jeho možnosti obmedzené. Horná hranica pre počet premenných je 200 a limit pre počet obmedzujúcich podmienok je 600. Z tohoto počtu je 400 rezervovaných pre dolné a horné

medze premenných. Ostatných obmedzujúcich podmienok, vrátane podmienok celočíselnosti, môže byť maximálne 200. [12].

Rozšírením doplnku Riešiteľ je doplnok OpenSolver, ktorý umožňuje riešenie väčších modelov, bez limitov na premenné. Vďaka podpore solveru CBC (Coin-or branch and cut) využíva na riešenie celočíselných úloh algoritmy vetiev i medzí i rezné roviny.

Najflexibilnejší doplnok pre MS Excel je SolverStudio. Nepoužíva Excel Solver, ale spája Excel s výkonnými open-source alebo komerčnými solvermi. Rozširuje schopnosti Excelu a umožňuje používanie modelovacích jazykov a externých solverov priamo v prostredí Excelu.

II PRAKTICKÁ ČASŤ

5 RIEŠENIE ÚLOHY METÓDOU GOMORYHO REZOV

Táto kapitola je zameraná na praktickú aplikáciu metódy Gomoryho rezov, ktorá predstavuje jeden z klasických prístupov riešenia celočíselných lineárnych úloh. Ide o iteratívny postup, ktorým sa k optimálnemu celočíselnému riešeniu dospeje pridávaním dodatočných obmedzení do simplexovej tabuľky. V nasledujúcej časti bude riešený konkrétny príklad, na ktorom bude názorne ukázaný celý priebeh tejto metódy od počiatočného riešenia až po dosiahnutie optimálneho celočíselného výsledku.

Je daná úloha:

Maximalizovať

$$z = 9x_1 + 10x_2$$

za podmienok

$$2x_1 + 5x_2 \leq 15$$

$$x_1 + \leq 3$$

$$x_1, x_2 \geq 0, x \in Z$$

Úloha bude najprv prevedená na kanonický tvar doplnením o sklzové premenné x_3 a x_4 . Podmienka celočíselnosti bude odstránená, aby bolo možné aplikovať simplexovú metódu.

Kanonický tvar úlohy:

$$z = 9x_1 + 10x_2$$

$$2x_1 + 5x_2 + x_3 = 15$$

$$x_1 + x_4 = 3$$

$$x_1, x_2, x_3, x_4 \geq 0$$

Premenné účelovej funkcie budú prevedené na ľavú stranu. Anulovaná účelová funkcia:

$$z - 9x_1 - 10x_2 = 0$$

Relaxovanú úlohu je teraz možné vložiť do simplexovej tabuľky (Tabuľka 6) a riešiť primárnym simplexovým algoritmom.

	x_1	x_2	x_3	x_4	b_i
x_3	2	5	1	0	15
x_4	1	0	0	1	3
z_j	-9	-10	0	0	0

Tabuľka 6. Východisková simplexová tabuľka

Prvým krokom je určenie kľúčového stĺpca v riadku účelovej funkcie, ktorým je najmenší koeficient $-z_j$, teda hodnota -10. Stĺpec x_2 je kľúčový stĺpec. Kľúčový riadok sa vypočíta ako podiel hodnoty b_i a hodnoty kľúčového stĺpca a najnižšia kladná hodnota predstavuje kľúčový riadok. Priesečníkom kľúčového stĺpca a kľúčového riadka bol získaný kľúčový prvok. Na mieste kľúčového prvku je potrebné získať hodnotu 1, preto sa celý riadok vydelený hodnotou kľúčového prvku. Na ostatné riadky treba použiť Gaussovu eliminačnú metódu tak, aby boli na ostatných miestach v kľúčovom stĺpci nuly. Po prepočítaní sa opakuje test optimality. Ak je riešenie celočíselné, bolo nájdené optimálne riešenie. V opačnom prípade sa pokračuje ďalšou iteráciou.

Na získanie optimálneho riešenia daného príkladu boli potrebné dve iterácie. Výsledok zobrazuje prepočítaná simplexová tabuľka 7.

	x_1	x_2	x_3	x_4	b_i	$\frac{\beta_i}{\alpha_{ik}}$
x_3	2	5	1	0	15	15/5=3
x_4	1	0	0	1	3	-
z_j	-9	-10	0	0	0	
		↑				
x_2	2/5	1	1/5	0	3	3/0,4=7,5
x_4	1	0	0	1	3	3/1=3
z_j	-5	0	2	0	30	
	↑					
x_2	0	1	1/5	-2/5	9/5	
x_1	1	0	0	1	3	
z_j	0	0	2	5	45	

Tabuľka 7. Prepočítaná simplexová tabuľka.

Výsledné optimálne riešenie:

$$\tilde{x} = (3, 9/5)^T$$

$$\tilde{z} = 45$$

Z výsledkov simplexovej tabuľky je možné vyčítať, že relaxácia problému je optimalizovaná, no získané riešenie nie je platné pre celočíselný problém. Preto nie je optimálnym riešením pôvodnej úlohy a výpočet bude pokračovať použitím Gomoryho metódy, ktorá pridá nové obmedzenie (tzv. Gomoryho rez), ktoré odstráni toto neceločíselné riešenie.

Gomoryho algoritmus odvodzuje reznú rovinu z riadku, ktorý má na pravej strane neceločíselnú hodnotu b_i . (Tabuľka 8). Ak sa na pravej strane nachádza viac riadkov s neceločíselnou hodnotou, vyberie sa riadok s najväčšou neceločíselnou časťou.

	x_1	x_2	x_3	x_4	b_i
x_2	0	1	1/5	-2/5	9/5
x_1	1	0	0	1	3
z_j	0	0	2	5	45

Tabuľka 8. Riešenie simplexovej tabuľky a výber riadku pre Gomoryho rez

Z vyriešenej tabuľky simplexovej metódy bolo získané riešenie:

$$x_2 = \frac{9}{5}, \quad x_1 = 3, \quad x_3 = 0, \quad x_4 = 0$$

$$f_{max} = 45$$

Riešenie obsahuje neceločíselnú hodnotu $x_2 = \frac{9}{5}$

Analýza riadku s neceločíselnou hodnotou x_2 :

$$x_2 = \frac{9}{5} - \frac{1}{5}x_3 - \frac{2}{5}x_4$$

Výpočet celočíselných zvyškov:

$$\frac{9}{5} - \left\lfloor \frac{9}{5} \right\rfloor = \frac{9}{5} - 1 = \frac{4}{5}$$

Celočíselná časť pravej strany je 1, zvyšok (neceločíselná časť) je $\frac{4}{5}$

Štandardný tvar rovnice pre odvodenie Gomoryho rezu:

$$x_2 - \frac{1}{5}x_3 - \frac{2}{5}x_4 = 1 + \frac{4}{5}$$

Zo štandardného tvaru sa odvodí riadok (Gomoryho rez), ktorý sa pridá ako nové obmedzenie do pôvodnej úlohy. Je potrebné zaviesť pomocnú premennú, napríklad s_l . Táto premenná slúži na to, aby mohol byť Gomoryho rez spracovaný simplexovou tabuľkou. Rozširuje sústavu obmedzení, pričom pôvodná účelová funkcia ostáva nezmenená.

Gomoryho rez je zostavený z neceločíselných častí koeficientov premenných v riadku základnej premennej x_2 , ktorá má neceločíselné riešenie:

$$x_3 = \frac{1}{5}, \quad \{x_3\} = \frac{1}{5} - \left\lfloor \frac{1}{5} \right\rfloor = \frac{1}{5}$$

$$x_4 = -\frac{2}{5}, \quad \{x_4\} = -\frac{2}{5} - \left\lfloor -\frac{2}{5} \right\rfloor = -\frac{2}{5} - (-1) = \frac{3}{5}$$

$$b_i = \frac{9}{5}, \quad \{b_i\} = \frac{9}{5} - \left\lfloor \frac{9}{5} \right\rfloor = \frac{9}{5} - 1 = \frac{4}{5}$$

Gomoryho rez po algebrickej úprave:

$$-\frac{1}{5}x_3 - \frac{3}{5}x_4 + s_1 = -\frac{4}{5}$$

Ďalším krokom je zostavenie východiskového tvaru simplexovej tabuľky, ktorá sa rozšíri o nový riadok zodpovedajúci Gomoryho rezu a o stĺpec pre pomocnú premennú s_l . Výpočet pokračuje pomocou duálnej metódy. V tabuľke sa určí kľúčový riadok podľa najzápornejšej hodnoty a kľúčový stĺpec podľa najmensej hodnoty nekladných podielov. (Tabuľka 9).

	x_1	x_2	x_3	x_4	s_1	b_i
x_2	0	1	1/5	- 2/5	0	9/5
x_1	1	0	0	1	0	3
s_1	0	0	- 1/5	- 3/5	1	- 4/5
z_j	0	0	2	5	0	45
x_2	0	1	1/3	0	- 2/5	7/3
x_1	1	0	- 1/3	0	5/3	5/3
x_4	0	0	1/3	1	- 5/3	4/3
z_j	0	0	1/3	0	25/3	115/3

Tabuľka 9. Simplexová tabuľka po prvej iterácii

Z tabuľky po prvej iterácii je zrejmé, že riešenie stále obsahuje neceločíselné hodnoty. Preto je potrebné pridať ďalší Gomoryho rez. Opäť sa vyberá základný riadok, ktorého pravá strana (b_i) má najväčšiu (neceločíselnú) zlomkovú časť.

Zlomková časť premenných:

$$x_2 = \frac{7}{3} = 2\frac{1}{3}, \quad \text{zlomková časť} = \frac{1}{3}$$

$$x_1 = \frac{5}{3} = 1\frac{2}{3}, \quad \text{zlomková časť} = \frac{2}{3}$$

$$x_4 = \frac{4}{3} = 1\frac{1}{3}, \quad \text{zlomková časť} = \frac{1}{3}$$

Najväčšiu zlomkovú časť má riadok s premennou x_1 a bude vybraný pre Gomoryho rez.

Analýza riadku vychádzajúca zo simplexovej tabuľky (Tabuľka 9) s najväčšou neceločíselnou hodnotou, premenná x_1 :

$$x_1 = \frac{5}{3} + \frac{1}{3}x_3 - \frac{5}{3}s_1$$

Nový riadok Gomoryho rezu bude opäť zostavený z neceločíselných častí koeficientov premenných v riadku základnej premennej s najväčšou neceločíselnou hodnotou pravej strany.

$$x_3 = -\frac{1}{3}, \quad \{x_3\} = -\frac{1}{3} - \lfloor -\frac{1}{3} \rfloor = -\frac{1}{3} - (-1) = \frac{2}{3}$$

$$x_4 = 0,$$

$$s_i = \frac{5}{3}, \quad \{s_i\} = \frac{5}{3} - \lfloor \frac{5}{3} \rfloor = \frac{5}{3} - 1 = \frac{2}{3}$$

$$b_i = \frac{5}{3}, \quad \{b_i\} = \frac{5}{3} - 1 = \frac{2}{3}$$

Gomoryho rez po algebrickéj úprave:

$$-\frac{2}{3}x_3 - 0x_4 - \frac{2}{3}s_1 + s_2 = -\frac{2}{3}$$

Do simplexovej tabuľky bude doplnený nový riadok vychádzajúci z Gomoryho rezu. Ostatné riadky tabuľky, vrátane účelovej funkcie, ostávajú nezmenené. Tabuľka sa rozšíri o nový stĺpec pre pomocnú premennú, ktorá umožní spracovanie nového obmedzenia v ďalších iteráciách. Výpočet pokračuje ďalšou simplexovou iteráciou. (Tabuľka 10).

	x_1	x_2	x_3	x_4	s_1	s_2	b_i
x_2	0	1	1/3	0	-2/3	0	7/3
x_1	1	0	-1/3	0	5/3	0	5/3
x_4	0	0	1/3	1	-5/3	0	4/3
s_2	0	0	-2/3	0	-2/3	1	-2/3
z_j	0	0	1/3	0	25/3	0	115/3
x_2	0	1	0	0	-1	-2/3	2
x_1	1	0	0	0	2	5/3	2
x_4	0	0	0	1	-2	-5/3	1
x_3	0	0	1	0	1	-2/3	1
z_j	0	0	0	0	8	25/3	38

Tabuľka 10. Doplnený riadok o nové obmedzenie. Druhá iterácia a výsledné riešenie

Tabuľka zároveň zobrazuje výsledné riešenie, ktoré je celočíselné a teda optimálne.

Optimálne celočíselné riešenie je $x_1=2$, $x_2=2$, s maximálnou hodnotou $z=38$.

Metóda Gomoryho rezov je založená na pridávaní lineárnych obmedzení (rezov), ktoré vychádzajú z necelých hodnôt a slúžia na postupné vylučovanie necelých riešení, až kým sa nájde optimálne celočíselné riešenie.

Na prvý pohľad sa môže javiť ako jednoduchý spôsob, ako získať celočíselné riešenie, zaokrúhlenie riešenia. Čo však môže spôsobiť zaokrúhľovanie? Ak by sa riešenie relaxovanej úlohy zaokrúhlilo, riešenie by nemuselo byť prístupné. Napríklad ak $x_1 = 3$, $x_2 = 1.8$ zaokrúhlime na celé čísla $x_1 = 3$, $x_2 = 2$, bude porušené obmedzenie $2x_1 + 5x_2 \leq 15$, pretože $2 \cdot 3 + 5 \cdot 2 = 16$, riešenie nie je prípustné. Pri zaokrúhľovaní hrozí strata optimality, a takýto prístup môže viesť k veľkým chybám v porovnaní s optimálnou hodnotou účelovej funkcie. Presné metódy ako Gomoryho rezy postupne eliminujú necelé hodnoty a udržujú riešenie v rámci obmedzení.

5.1 Porovnanie metódy Gomoryho rezov a metódy vetiev a medzí

Princíp metódy vetiev a medzí (Branch and Bound) spočíva v systematickom rozdeľovaní pôvodnej celočíselnej úlohy na menšie, ľahšie riešiteľné podúlohy, vetvy. Tieto menšie úlohy vznikajú postupným zavádzaním nových ohraničení na premenné, čím sa vytvára strom úloh lineárneho programovania. Tieto úlohy sa riešia samostatne a ich riešenia sa následne posudzujú z hľadiska celočíselnosti a optimality. Ak riešenie nie je celočíselné, ale ešte stále má potenciál byť súčasťou optimálneho riešenia, vetví sa ďalej. Naopak, ak je hodnota účelovej funkcie horšia než doposiaľ známe celočíselné riešenie, daná vetva sa odreže z vetviaceho stromu a už sa v nej nepokračuje.

Metóda Gomoryho rezných rovín (Gomory's Cutting Plane Method) využíva iný prístup. Najprv sa rieši relaxovaná úloha (bez požiadavky na celočíselnosť premenných) pomocou simplexovného algoritmu. Ak výsledné riešenie nie je celočíselné, z vyriešenej simplexovej tabuľky sa vyberie riadok, ktorého pravá strana má najväčšiu neceločíselnú hodnotu (zlomkovú časť). Na základe tohto riadku sa sformuluje nové lineárne obmedzenie, Gomoryho rez. Toto obmedzenie vylúči aktuálne neceločíselné riešenie, ktoré je optimálne pre relaxovanú úlohu, ale zároveň ponechá všetky možné celočíselné riešenia. Gomoryho rez sa pridá do simplexovej tabuľky ako nový riadok a zároveň sa zavedie ďalšia pomocná premenná. Výsledkom je nová lineárna úloha, ktorú možno znova riešiť pomocou simplexovej alebo duálnej simplexovej metódy. Po každom pridaní rezu sa hľadá nové riešenie. Ak je opäť neceločíselné, pridáva sa ďalší rez až kým sa nenájde optimálne riešenie, ktoré je zároveň celočíselné.

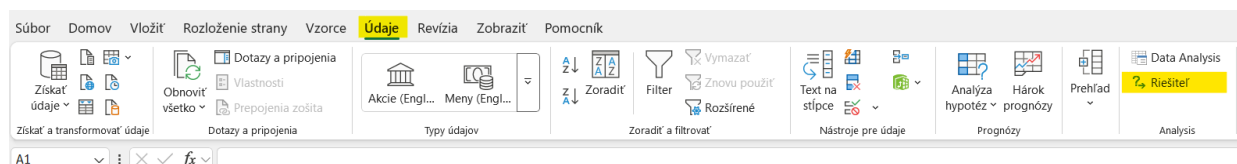
Obe metody mají svoje výhody i nevýhody. Metóda vetiev a medzí je veľmi spoľahlivá, no často výpočtovo náročná, najmä pri veľkom počte vetiev. Čo vedie k výraznému zvýšeniu časovej a pamäťovej záťaže pri riešení rozsiahlych problémov. Gomoryho rezy môžu pomocou rezov systematicky zúžiť priestor možných riešení. Avšak, v praxi môžu tieto metódy naraziť na problémy degenerácie či numerickej nestability. V praxi sa preto často využíva kombinácia oboch metód, a ich implementácia v rámci moderných solverov (napr. CBC, CPLEX, Gurobi).

6 PROGRAMOVÁ PODPORA CELOČÍSELNÝCH ÚLOH

6.1 Doplnok Riešiteľ

Prvý Riešiteľ sa objavil vo verzii Excel 3.0 z roku 1990, avšak riešil len základné úlohy lineárneho programovania. Od verzie Excel 97 pribudla limitovaná podpora celočíselného programovania. Moderného a skutočne výkonného Riešiteľa, ako ho poznáme dnes, s plnou podporou celočíselného programovania, priniesol Excel 2010. Doplnok Riešiteľ má vo verzii 2010 vylepšené používateľské rozhranie (Evolučný riešiteľ), a prináša plnohodnotnú podporu pre úlohy celočíselného programovania, s možnosťou výberu algoritmov a možnosťou označiť konkrétne rozhodovacie premenné ako celočíselné (int).

Doplnok je nutné najskôr aktivovať. V Exceli, v ľavom hornom rohu, je potrebné kliknúť na *Súbor* a následne na *Možnosti*. Po otvorení karty *Zobrazenie a správa doplnkov balíka Microsoft Office* kliknúť na *Doplnky* a následne v spodnej časti *Spravovať* vybrať *Doplnky programu Excel* a potvrdiť tlačidlom *Spustiť*. Po spustení sa zobrazí nové okno, v ktorom treba vybrať *Doplnok Riešiteľ* a potvrdiť *OK*. Riešiteľ je teraz aktivovaný, čo sa dá ľahko skontrolovať na karte *Údaje*. V sekcii *Analysis* je teraz zobrazený Riešiteľ. (Obrázok 3).



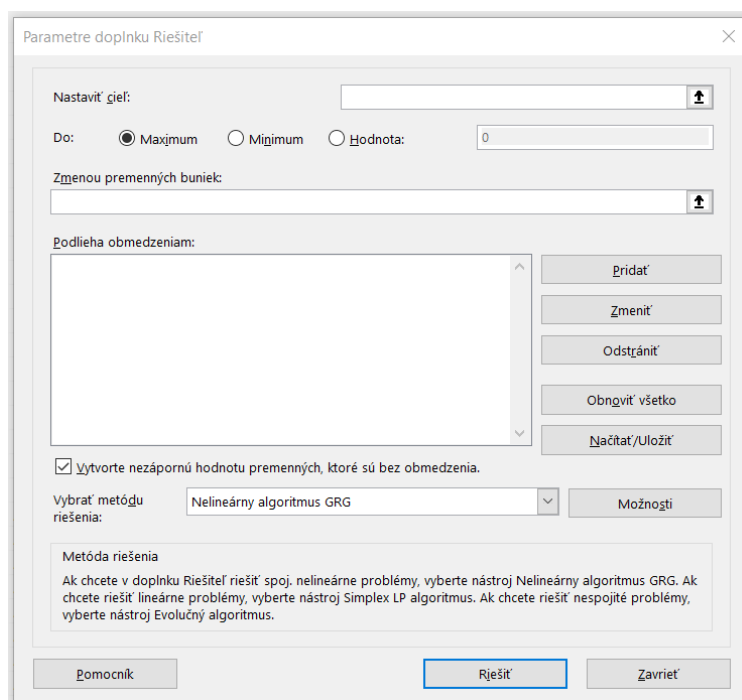
Obrázok 3. Zobrazenie doplnku Riešiteľ na páse kariet v Exceli.

Po kliknutí na príkaz Riešiteľ sa zobrazí úvodné okno s parametrami doplnku Riešiteľ. Veľmi dôležité sú nasledujúce polia:

- *Nastaviť cieľ* - do tohto poľa sa zadáva hodnota bunky, v ktorej má byť zobrazený požadovaný výsledok.
- *Do*: Podľa typu riešenej úlohy treba vybrať správnu možnosť. Ak je riešená maximalizačná úloha, označiť možnosť *Maximum*, ak sa jedná o úlohu minimalizačnú, označiť variantu *Minimum*.
- *Zmenou premenných buniek* – Sem sa zadávajú bunky, ktoré bude program meniť, aby bolo dosiahnuté požadované riešenie. Na rozdiel od cieľovej bunky, ktorá môže byť len jedna, premenných buniek môže byť zadaných

viacero. Aby sa dalo nájsť riešenie, musí existovať vzťah medzi premennými bunkami a cieľovou bunkou.

- *Podlieha obmedzeniam* – v tomto okne budú zobrazené všetky obmedzujúce podmienky. Tie budú vložené pomocou tlačidla *Pridať*, ktoré sa nachádza vpravo vedľa okna obmedzení a po jeho kliknutí sa zobrazí dodatočné okno, kde budú zadefinované obmedzujúce podmienky.
- *Vybrať metódu riešenia* – Riešiteľ ponúka tri metódy riešenia. Nelineárny algoritmus, Simplexovú lineárne programovanie a Evolučný algoritmus. Na riešenie úloh celočíselného lineárneho programovania je dôležité vybrať možnosť *Simplex LP algoritmus*



Obrázok 4. Parametre doplnku Riešiteľ

Jednotlivé parametre Riešiteľa budú naznačené na jednoduchom príklade z oblasti výrobného plánovania.

Slovné zadanie úlohy: Firma vyrába dva typy výrobkov, A a B. Zisk z predaja výrobku A predstavuje 90 €, zisk z predaja výrobku B je 70 €. Na výrobu oboch výrobkov podnik spotrebovávajú dva druhy surovín, S1 a S2. Na výrobu výrobku A sú potrebné 2 jednotky suroviny S1 a 5 jednotiek suroviny S2. Na výrobu výrobku B sa spotrebuje 5 jednotiek suroviny

S1 a 4 jednotky suroviny S2. K dispozícii je obmedzené množstvo surovín. Zásoba suroviny S1 je 1200 jednotiek a zásoba suroviny S2 predstavuje 1300 jednotiek. Koľko kusov každého výrobku je potrebné vyrobiť, aby bol maximalizovaný celkový zisk?

Z danej úlohy je jednoznačné, že sa jedná o maximalizačnú úlohu. K dispozícii je obmedzené množstvo surovín a vyrobený počet výrobkov musí byť nezáporný a celočíselný.

Matematický model úlohy:

Maximalizovať

$$z = 90x_1 + 70x_2$$

za podmienok

$$4x_1 + 5x_2 \leq 1200$$

$$5x_1 + 2x_2 \leq 1300$$

$$x_1, x_2 \geq 0, x \in Z$$

Východisková Excel tabuľka:

	A	B	C
1	Parametre	Výrobok A	Výrobok B
2	Zisk/kus (€)	90	70
3	Surovina S1	4	5
4	Surovina S2	5	2
5	Optimálny počet vyrobených kusov		
6			
7	Celkový zisk	0	
8	Spotreba zásob S1	0	
9	Spotreba zásob S2	0	

Obrázok 5. Východisková tabuľka

V piatom riadku tabuľky sa nachádza *Optimálny počet vyrobených kusov* a zatiaľ neobsahuje žiadne hodnoty pre výrobok A ani B. V riadku číslo sedem je zobrazený *Celkový zisk*, taktiež s nulovou hodnotou. Ich hodnoty budú získané výpočtom úlohy pomocou Riešiteľa. K samotnému výpočtu je potrebné nastavenie parametrov v okne *Parametre doplnku Riešiteľ*. Do okna *Nastaviť cieľ* bude vložená bunka pre celkový zisk. Optimálny počet vyrobených

kusov bude získaný zápisom buniek v tvare B5:C5 do *Zmenou premenných buniek*. Prehľad o spotrebe zásob S1 a S2 prinesie riadok osem a deväť. Pre správny výpočet je však potrebné do jednotlivých buniek doplniť vhodné vzorce, ktoré zohľadnia príslušné vstupné hodnoty a podmienky.

$$\text{Celkový zisk} = B2*B5 + C2*C5$$

$$\text{Spotreba zásob S1} = B3*B5 + C3*C5$$

$$\text{Spotreba zásob S2} = B4*B5 + C4*C5.$$

Obmedzujúce podmienky je nutné jednotlivito vložiť v okne *Pridať obmedzenie*, ktoré sa otvorí po kliknutí na tlačidlo *Pridať*. Postupne treba zadať všetky obmedzenia, ktoré sa následne zobrazia v okne *Podlieha obmedzeniam*. Pri riešení celočíselných úloh je potrebné zadať aj podmienku celočíselnosti, aby sa predišlo neceločíselnému riešeniu. (Obrázok 6). Pre zabezpečenie najlepšieho celočíselného riešenia existuje možnosť nastavenia celočíselnej odchýlky v dialógovom okne *Možnosti*. Ak sú všetky podmienky zadane, treba spustiť riešenie kliknutím na tlačidlo *Riešiť*.

	A	B	C
1	Parametre	Výrobok A	Výrobok B
2	Zisk/kus (€)	90	70
3	Surovina S1	4	5
4	Surovina S2	5	2
5	Optimálny počet vyrobených kusov	241,1764706	47,05882353
6			
7	Celkový zisk	25000	
8	Spotreba zásob S1	1200	
9	Spotreba zásob S2	1300	
10			
11	Pridať obmedzenie Odkaz na bunku: \$B\$5:\$E\$16 ↑ Obmedzenie: int celočíslný OK Pridať Zrušiť		
12			
13			
14			
15			
16			

Obrázok 6. Pridanie podmienky celočíselnosti

Výsledky riešenia sú zobrazené v tabuľke. (Tabuľka 11) Z tabuľky je možné vyčítať, že maximálny celkový zisk je výške 24.980 € a bude dosiahnutý pri výrobe 241 kusov výrobku

A a 47 kusov výrobku B. Pričom sa spotrebuje z dostupného množstva zásob suroviny S1 1199 jednotiek z dostupných 1200 jednotiek a 1299 jednotiek z 1300 jednotiek zo zásob suroviny S2. Dialógové okno *Výsledok doplnku Riešiteľ* zároveň informuje, že Riešiteľ našiel riešenie a všetky podmienky a obmedzenia optimálnosti sú splnené. V rovnakom okne sa nachádzajú aj *Zostavy*. Po zakliknutí *Prehľadových zostáv* pribudne na spodnej strane nový hárok s odpoveďou. (Tabuľka 12).

	A	B	C
1	Parametre	Výrobok A	Výrobok B
2	Zisk/kus (€)	90	70
3	Surovina S1	4	5
4	Surovina S2	5	2
5	Optimálny počet vyrobených kusov	241	47
6			
7	Celkový zisk	24980	
8	Spotreba zásob S1	1199	
9	Spotreba zásob S2	1299	

Tabuľka 11. Výsledok riešenia

Doplnok Riešiteľ v Exceli neumožňuje používateľovi zvoliť konkrétnu metódu na riešenie úloh lineárneho celočíselného programovania. Zároveň neposkytuje žiadny výstup, ktorý by jednoznačne uvádzal, aký algoritmus bol pri riešení použitý. Táto informácia nie je explicitne uvedená ani vo výsledkoch optimalizácie. Solver využíva metódu vetiev a medzí v kombinácii so Simplexovým algoritmom, pričom sám automaticky rozhoduje o ich použití podľa charakteru danej úloh.

Microsoft Excel 16.0 Zostava s odpoveďou

Hárok: [Zošit1]Hárok1

Zostava vytvorená: 18. 5. 2025 13:32:27

Výsledok Doplnok Riešiteľ našiel riešenie. Všetky obmedzenia a podmienky optimálnosti sú splnené.

Nástroj Riešiteľ

Možnosti doplnku Riešiteľ

Cieľová bunka (Maximum)

Bunka	Názov	Pôvodná hodnota	Výsledná hodnota
\$B\$7	Celkový zisk Výrobok A	24930	24980

Premenné bunky

Bunka	Názov	Pôvodná hodnota	Výsledná hodnota	Celočíselné
\$B\$5	Optimálny počet vyrobených kusov Výrobok A	242	241	Celočíselné
\$C\$5	Optimálny počet vyrobených kusov Výrobok B	45	47	Celočíselné

Obmedzenia

Bunka	Názov	Hodnota bunky	Vzorec	Stav	Odchýlka
\$B\$8	Spotreba zásob S1 Výrobok A	1199	\$B\$8<=1200	Neplatí	1
\$B\$9	Spotreba zásob S2 Výrobok A	1299	\$B\$9<=1300	Neplatí	1
\$B\$5=Celočíselné					
\$C\$5=Celočíselné					

Tabuľka 12. Zostava s odpoveďou

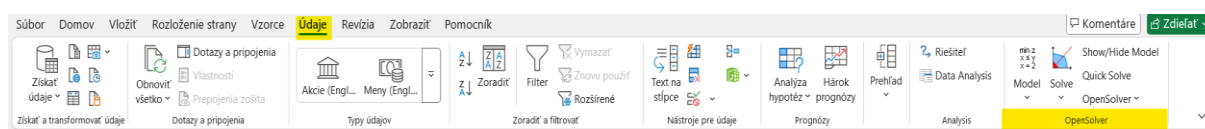
6.2 Doplnok OpenSolver

Open Solver je open-source optimalizačný nástroj, založený na VBA (Visual Basic for Application), ktorý rozširuje doplnok Excel Riešiteľ. Umožňuje riešenie väčších a zložitejších problémov a oproti Riešiteľovi nemá obmedzený počet rozhodovacích premenných. Využíva výkonné solvery ako je CBC alebo Gurobi, ktoré umožňujú efektívnejšie riešenie úloh pomocou algoritmov rezných rovín či vetiev a medzí.

Podobne ako Excel Riešiteľ je OpenSolver doplnok, ktorý je treba stiahnuť a nainštalovať do Excelu. Je dostupný v dvoch verziách: jednoduchšej OpenSolver Linear, a v pokročilej verzii OpenSolver Advanced (Non-Linear), ktorá podporuje viac solverov vrátane ne-lineárnych. Obe verzie sú dostupné pre Windows a Mac. Na Windows je podporovaná s Excelom 2007 a novším. Na Mac OS s Excelom pre Mac 2011 a akoukoľvek verzou OS X novšou než 10.7. [16]

Na riešenie príkladu je použitá verzia 2.9.4 Beta Advanced, dostupná na oficiálnej stránke <https://opensolver.org>, kde sa nachádzajú všetky potrebné informácie o inštalácii a fungovaní OpenSolveru. Inštalačný súbor je vo formáte .zip. a obsah komprimovaného súboru treba extrahovať. Rovnako ako pri aktivácii Riešiteľa, je potrebné OpenSolver v Exceli najskôr

aktivovať kliknutím na *Súbor* → *Možnosti* → *Doplnky* → *Spravovať*, vybrať v okne možnosť *Doplnky programu Excel* → *Spustiť*, čím sa otvorí nové okno, kde stačí zakliknúť OpenSolver a potvrdiť výber kliknutím na *OK*. Môže sa stať, že pred samotnou aktiváciou doplnku bude potrebné upraviť nastavenia dôveryhodnosti a pridať cestu k extrahovanému priečinku v centre dôveryhodnosti. V takom prípade je potrebné opäť prejsť do ponuky *Súbor* → *Možnosti* → *centrum dôveryhodnosti* → *nastavenie centra dôveryhodnosti* → *pridať nové umiestnenie* a pomocou tlačidla *prehľadávať* vybrať priečinok, do ktorého bol OpenSolver extrahovaný. Tento výber je potrebné potvrdiť kliknutím na *OK*. Následne sa OpenSolver zobrazí na karte *Údaje*. (Obrázok 7).



Obrázok 7. Zobrazenie doplnku OpenSolver na páse kariet v Exceli

Používanie OpenSolveru je veľmi podobné ako použitie Excel Riešiteľa. Základným rozdielom je, že OpenSolver oddeľuje proces modelovania od procesu riešenia.

Príklad: Dopravná úloha, minimalizácia nákladov pekárne.

Firma Chlebič má na Slovensku 4 pekárne, v Trnave, Ružomberku, Poprade a Zvolene, v ktorých okrem chleba a pečiva vyrába i vianočky. Výrobná kapacita pekární je 3600, 2900, 3200 a 2800 vianočiek denne. Vianočky sú dovážané do podnikovej predajne v meste Bratislava, Nitra, Žilina, Banská Bystrica a Košice v množstve 3000, 2400 2300, 2100 a 2700 kusov denne. Distribučné náklady medzi pekárňami a predajňami sú vypočítané na 1 vianočku a sú zobrazené v Tabuľke 13. Úlohou je vyriešiť rozvoz vianočky z dodávateľských miest do predajní tak, aby boli minimalizované celkové náklady.

	Bratislava	Nitra	Žilina	Banská Bystrica	Košice	Výrobná kapacita
Trnava	0,6	0,5	1,6	1,7	4	3600
Ružomberok	2,7	1,8	0,7	0,5	1,9	2900
Poprad	3,4	2,5	1,4	1,2	1,2	3200
Zvolen	2	1,1	1,1	0,2	2,2	2800
Požadované množstvo	3000	2400	2300	2100	2700	12500

Tabuľka 13. Distribučné náklady

Matematický model úlohy:

Minimalizovať

$$\begin{aligned} Z = & 0,6x_{11} + 0,5x_{12} + 1,6x_{13} + 1,7x_{14} + 4x_{15} + 2,7x_{21} + 1,8x_{22} \\ & + 0,7x_{23} + 0,5x_{24} + 1,9x_{25} + 3,4x_{31} + 2,5x_{32} + 1,4x_{33} + 1,2x_{34} \\ & + 1,2x_{35} + 2x_{41} + 1,1x_{42} + 1,1x_{43} + 0,2x_{44} + 2,2x_{45} \end{aligned}$$

za podmienok

$$0,6x_{11} + 0,5x_{12} + 1,6x_{13} + 1,7x_{14} + 4x_{15} \leq 3600$$

$$2,7x_{21} + 1,8x_{22} + 0,7x_{23} + 0,5x_{24} + 1,9x_{25} \leq 2900$$

$$3,4x_{31} + 2,5x_{32} + 1,4x_{33} + 1,2x_{34} + 1,2x_{35} \leq 3200$$

$$2x_{41} + 1,1x_{42} + 1,1x_{43} + 0,2x_{44} + 2,2x_{45} \leq 2800$$

Požiadavky predajní

$$x_{11} + x_{21} + x_{31} + x_{41} = 3000$$

$$x_{12} + x_{22} + x_{32} + x_{42} = 2400$$

$$x_{13} + x_{23} + x_{33} + x_{43} = 2300$$

$$x_{14} + x_{24} + x_{34} + x_{44} = 2100$$

$$x_{15} + x_{25} + x_{35} + x_{45} = 2700$$

$$x_{ij} \geq 0, \quad x \in Z$$

$$i = 1, 2, 3, 4, \quad j = 1, 2, 3, 4, 5$$

Všetky dostupné údaje je potrebné zapísať do prehľadnej tabuľky a zároveň pripraviť priestor pre hodnoty, ktoré budú vypočítané doplnkom OpenSolver. Do príslušných buniek je taktiež potrebné vložiť vzorce pre výpočet cieľovej funkcie, výrobné kapacity a dopytovaného množstva. (Tabuľka 14).

Cieľová funkcia =SUMPRODUCT(B2:F5; B11:F14)

Výrobné kapacity:

G11 =SUM(B11:F11), G12 =SUM(B12:F12), G13 =SUM(B13:F13), G14 =SUM(B14:F14)

Súčty dopytu jednotlivých predajní: B15 =SUM(B11:B14), C15 =SUM(C11:C14), D15 =SUM(D11:D14), E15 =SUM(E11:E14), F15 =SUM(F11:F14).

	A	B	C	D	E	F	G
1	Výrobňa/Predajňa	Bratislava	Nitra	Žilina	Banská Bystrica	Košice	Výrobná kapacita
2	Trnava	0,6	0,5	1,6	1,7	4	3600
3	Ružomberok	2,7	1,8	0,7	0,5	1,9	2900
4	Poprad	3,4	2,5	1,4	1,2	1,2	3200
5	Zvolen	2	1,1	1,1	0,2	2,2	2800
6	Požadované množstvo	3000	2400	2300	2100	2700	
7							
8							
9							
10	Výrobňa/Predajňa	Bratislava	Nitra	Žilina	Banská Bystrica	Košice	
11	Trnava						0
12	Ružomberok	$0 \leq$					$\geq$ 0
13	Poprad						0
14	Zvolen						0
15	Súčet (dopyt)	=	0	0	0	0	0
16	Požadované množstvo	3000	2400	2300	2100	2700	
17							
18							
19	min Cieľová funkcia						
20		0					

Tabuľka 14. Východisková tabuľka pre riešenie úlohy pomocou doplnku OpenSolver

Po vytvorení východiskovej tabuľky so zadanými vzorcami je potrebné vytvoriť model pomocou doplnku OpenSolver, kliknutím na možnosť *Model*. Následne sa otvorí dialógové okno, kde je nutné zadať do Objective Cell bunku cieľovej funkcie a vybrať, či ide o úlohu maximalizačnú alebo minimalizačnú. Okno Variable Cells je určené na zadanie rozsahu buniek, ktoré predstavujú rozhodovacie premenné, teda bunky, ktoré treba optimalizovať. Ďalším krokom je zadanie obmedzení do okna Constraints. Podmienku nezápornosti je možné pridať do obmedzení alebo zakliknúť nastavenie premenných na nezáporné hodnoty (*Make unconstrained variable cells non-negative*), toto nastavenie automaticky zabezpečí, že solver nedovolí záporné hodnoty. Taktiež je vhodné nastaviť v možnostiach toleranciu metódy vetiev a medzí na nulu, aby bolo docielené čo najpresnejšie riešenie (*Options* → *Branch and Bound Tolerance*). Po zadaní všetkých potrebných údajov sa model uloží kliknutím na *Save Model*. Údaje v tabuľke Excelu zatiaľ zostanú nezmenené, ešte neobsahujú vypočítané hodnoty, no definované obmedzenia a cieľová funkcia sú už súčasťou modelu pripraveného na optimalizáciu. Na vyriešenie úlohy je potrebné kliknúť na tlačidlo *Solve* na karte OpenSolver. Optimálne riešenie sa zobraní v pripravenej tabuľke v Exceli. (Tabuľka 15).

	A	B	C	D	E	F	G
1	Výrobná/Predajňa	Bratislava	Nitra	Žilina	Banská Bystrica	Košice	Výrobná kapacita
2	Trnava	0,6	0,5	1,6	1,7	4	3600
3	Ružomberok	2,7	1,8	0,7	0,5	1,9	2900
4	Poprad	3,4	2,5	1,4	1,2	1,2	3200
5	Zvolen	2	1,1	1,1	0,2	2,2	2800
6							
7	Požadované množstvo	3000	2400	2300	2100	2700	
8							
9	Trnava	3000	600	0	0	0	3600
10	Ružomberok	0	0	1800	1100	0	2900
11	Poprad	0	0	500	0	2700	3200
12	Zvolen	0	1800	0	1000	0	2800
13		= 3000	2400	2300	2100	2700	
14							
15	Cieľová funkcia						
16	min	10030					

Tabuľka 15. Optimálne riešenie úlohy

Z tabuľky je možné vyčítať, koľko jednotiek má ísť z danej pekárne do predajne, aby bol uspokojený dopyt predajní, dodržané výrobné kapacity pekární a minimalizované náklady na prepravu. Zároveň sú v poli cieľovej funkcie uvedené minimálne celkové prepravné náklady.

OpenSolver využil na riešenie úlohy predvolený solver CBC (Coin-or Branch and Cut), ktorý používa na riešenie celočíselných úloh kombináciu metód vetiev a medzí a rezných rovín. Rovnako ako pri doplnku Riešiteľ, používateľ nemá možnosť výberu konkrétnej metódy. Solver ju volí sám podľa typu úlohy. Výstupom úlohy je vyplnená tabuľka v Exceli.

Podľa logu OpenSolvera bola na riešenie úlohy použitá metóda CBC, tolerancia optimality bola nastavená na 0, teda riešenie je globálne optimálne. Na nájdenie riešenia bolo potrebných 8 iterácií a výpočet trval 0.002 sekundy. Solver ukončil výpočet bez chyby.

6.3 Doplnok StudioSolver s využitím programovacieho jazyka Python

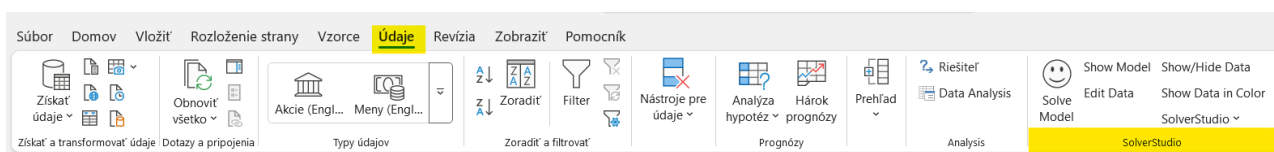
SolverStudio je doplnok pre Microsoft Excel (od verzie 2007 a novších), ktorý umožňuje vytvárať a riešiť optimalizačné modely priamo v prostredí Excelu s využitím pokročilých modelovacích jazykov. Vďaka tomuto nástroju môžu používatelia formulovať modely, prepojiť ich s tabuľkovými údajmi a získať riešenia bez potreby opustiť Excel. SolverStudio využíva na riešenie lineárnych a celočíselných optimalizačných úloh predovšetkým knižnicu PuLP a Pyomo. Obe sú open-source knižnice v jazyku Python, ktorý umožňuje jednoduché a

flexibilné formulovanie optimalizačných modelov. Súčasťou inštalácie je aj solver CBC, ktorý sa často využíva na riešenie celočíselných lineárnych problémov. Okrem PuLP podporuje SolverStudio aj ďalšie modelovacie a simulačné jazyky AMPL, GMPL, GAMS či SimPy a je kompatibilné s viacerými známymi solvermi, vrátane CPLEX, Gurobi a GLPK. Niektoré z týchto solverov si však vyžadujú dodatočnú inštaláciu zo strany používateľa. Väčšina jazykov používaných v SolverStudio sú písané v Pythone a využívajú jeho syntax a štruktúru.

Hlavnou výhodou SolverStudio je, že celý proces, od formulácie modelu, cez načítanie údajov z Excelu, až po interpretáciu výsledkov, prebieha priamo v rámci Excelového prostredia. Model je možné uložiť priamo do excelového súboru a prepojiť ho s konkrétnymi bunkami tabuľky ako parametre, konštanty alebo premenné. Po spustení modelu SolverStudio automaticky vypočíta riešenie a výsledky vloží späť do pracovného hárka.

Inštalčný balík SolverStudio spolu s podrobnými informáciami o možnostiach, inštalácii a používaní je dostupný na oficiálnej stránke <https://solverstudio.org/download-install/>. Balík na stiahnutie obsahuje všetky potrebné komponenty, vrátane IronPython, PuLP, štandardného Pythonu (CPython), GNU Linear Programming Kit (GLPK) a podporných súborov pre AMPL a GAMS. [19]

Pred inštaláciou doplnku SolverStudio je potrebné rozbaľiť stiahnutý komprimovaný súbor. Potom v rozbalenom priečinku spustiť inštalčný súbor *Setup.exe* a pre dokončenie inštalácie kliknúť na *Install*. Po úspešnom nainštalovaní sa doplnok SolverStudio objaví na karte *Údaje* v Exceli a je pripravený na použitie. (Obrázok 8).



Obrázok 8. Zobrazenie SolverStudia na karte Údaje v Exceli

Na analýzu funkčnosti doplnku SolverStudio bude použitý rovnaký príklad ako pri doplnku OpenSolver, čo zároveň overí správnosť riešenia. Cieľom riešenej úlohy je nájsť optimálne rozdelenie dodávok vianočiek z viacerých pekární do predajní v rôznych mestách tak, aby sa minimalizovali celkové prepravné náklady. Dodávky pritom musia rešpektovať kapacity jednotlivých pekární a zároveň uspokojiť presný dopyt každej predajne. Zároveň je požadované, aby dodávané množstvá boli celočíselné.

Riešenie úlohy začína prípravou vstupných údajov v Exceli. Najskôr je potrebné vytvoriť tabuľku, ktorá bude obsahovať prepravné náklady z každej pekárne do každej predajne. Súčasťou tabuľky by mali byť aj kapacity jednotlivých pekární (umiestnené v stĺpci) a dopyt v jednotlivých predajniach (umiestnený v riadku). Okrem toho je vhodné pripraviť aj prázdnu tabuľku, do ktorej sa po vyriešení úlohy automaticky prenesú výsledné optimálne hodnoty dodávok. Zároveň budú v bunke J1 vyčíslené minimálne náklady, teda hodnota účelovej funkcie. (Tabuľka 16)

	A	B	C	D	E	F	G	H	I	J
1		Bratislava	Nitra	Žilina	Banská Bystrica	Košice	kapacita		Minimálne náklady:	
2	Trnava	0,6	0,5	1,6	1,7	4	3600			
3	Ružomberok	2,7	1,8	0,7	0,5	1,9	2900			
4	Poprad	3,4	2,5	1,4	1,2	1,2	3200			
5	Zvolen	2	1,1	1,1	0,2	2,2	2800			
6	Dopyt	3000	2400	2300	2100	2700				
7										
8										
9										
10										
11										
12										
13										

Tabuľka 16. Príklad č. Východiskové riešenie

Po príprave dát v Exceli nasleduje samotný proces modelovania optimalizačnej úlohy v SolverStudio, ktoré je dostupné na karte Údaje v Exceli. V okne SolverStudio je potrebné kliknúť na možnosť *Show model*, čím sa otvorí textový editor, ktorý okrem priestoru na zápis kódu ponúka aj menu s možnosťami *File*, *Edit* a *Language*. V ponuke *Language* je možný výber modelovacieho jazyka a solvera, v ktorom bude úloha riešená.

Na riešenie úlohy bol zvolený Python (external), ktorý umožňuje spúšťať Python kódy mimo samotného Excelu, pričom je možné čítať a zapisovať dáta priamo z Excelových buniek pomocou knižníc ako xlwings. Podmienkou je mať v počítači nainštalovaný Python a následne doinštalovať potrebné knižnice PuLP na formuláciu a riešenie optimalizačných úloh a xlwings na prepojenie s Excelom. Inštaláciu týchto knižníc je možné vykonať cez príkazový riadok zadáním príkazu: `pip install pulp xlwings`.

Do textového editora bol následne zapísaný Python skript, ktorý postupne vykonáva:

- načítanie vstupných dát (náklady, kapacity, dopyt) z predpripravených tabuliek v Exceli pomocou knižnice xlwings,

- definíciu premenných predstavujúcich množstvá dodávok medzi jednotlivými pekárňami a predajňami ako celočíselné premenné,
- nastavenie cieľovej funkcie, ktorej cieľom je minimalizovať celkové prepravné náklady,
- definovanie obmedzení zabezpečujúcich rešpektovanie výrobných kapacít pekární a zároveň uspokojenie dopytu v predajniach. (Príloha 1)

Po napísaní skriptu sa výpočet spustí kliknutím na *Solve model* v ponuke SolverStudio. Tým sa aktivuje optimalizačný solvera CBC, ktorý vykoná výpočet a nájde jej optimálne riešenie. Po ukončení výpočtu sa výsledky zapíšu do pripravenej tabuľky v Exceli.

Tabuľka s riešením obsahuje optimálne množstvá dodávok z jednotlivých pekární, celkové náklady, dodané množstvo do jednotlivých predajní ako aj minimálne náklady na rozvoz tovaru. (Tabuľka 17).

Po spustení a vyriešení modelu bol dostupný podrobný log riešenia. Podľa tohto výstupu solver okamžite identifikoval optimálne riešenie pomocou heuristiky greedy equality, bez potreby akýchkoľvek iterácií alebo vetvenia. Výsledná hodnota cieľovej funkcie bola 10 030, teda rovnaká ako pri riešení pomocou optimalizačného doplnku OpenSolver. Výpočet úlohy bol ukončený za menej než 0,01 sekundy bez akýchkoľvek chýb.

	A	B	C	D	E	F	G	H	I	J
1		Bratislava	Nitra	Žilina	Banská Bystrica	Košice	kapacita		Minimálne náklady:	10030
2	Trnava	0,6	0,5	1,6	1,7	4	3600			
3	Ružomberok	2,7	1,8	0,7	0,5	1,9	2900			
4	Poprad	3,4	2,5	1,4	1,2	1,2	3200			
5	Zvolen	2	1,1	1,1	0,2	2,2	2800			
6	Dopyt	3000	2400	2300	2100	2700				
7										
8		Bratislava	Nitra	Žilina	Banská Bystrica	Košice				
9	Trnava	3000	600				3600			
10	Ružomberok			2300	600		2900			
11	Poprad				500	2700	3200			
12	Zvolen		1800		1000		2800			
13	Dodané	3000	2400	2300	2100	2700				

Tabuľka 17. Výsledné riešenie úlohy.

OpenSolver i SolverStudio dospeli k rovnakému riešeniu s rovnakou hodnotou cieľovej funkcie 10 030, no SolverStudio v kombinácii s knižnicou PuLP poskytuje väčšiu flexibilitu, možnosť modelovania zložitejších úloh a automatické programové spracovanie výstupov. Naopak,

OpenSolver je jednoduchší na použití, nevyžaduje programovanie a je vhodný pre používateľov s minimálnymi znalosťami programovania.

ZÁVER

Bakalárska práca priniesla prehľad problematiky celočíselného lineárneho programovania, a to z teoretického aj praktického hľadiska. V teoretickej časti bola predstavená história a vývoj lineárneho programovania, základné princípy a algoritmy simplexovej metódy, ako aj špecifiká celočíselného programovania vrátane metód rezov a metódy vetiev a medzí.

V praktickej časti bola úspešne implementovaná metóda Gomoryho rezov, ktorá bola zároveň porovnaná s metódou vetiev a medzí. Ďalej boli predstavené a zhodnotené možnosti riešenia celočíselných lineárnych programov pomocou moderných softvérových nástrojov dostupných ako doplnky pre Microsoft Excel – konkrétne Solver, OpenSolver a SolverStudio. Ukázalo sa, že tieto nástroje poskytujú užitočnú podporu pri praktickom riešení úloh lineárneho i celočíselného lineárneho programovania. Analýza softvérových doplnkov pre Microsoft Excel potvrdila, že tieto nástroje výrazne zjednodušujú prácu s celočíselnými úlohami a zvyšujú dostupnosť optimalizačných metód aj pre používateľov bez hlbokých matematických znalostí. Ich flexibilita a možnosť programovania, napríklad prostredníctvom jazyka Python v SolverStudio, otvárajú nové možnosti pre pokročilé modelovanie a automatizáciu riešenia.

Výsledky práce ukazujú, že celočíselné lineárne programovanie predstavuje náročnú, no veľmi dôležitú oblasť optimalizácie so širokým spektrom aplikácií v praxi. Prehľad a porovnanie teoretických metód spolu s ich praktickými implementáciami poskytuje užitočný základ pre ďalšie štúdium či nasadenie v reálnych projektoch.

V budúcnosti by bolo zaujímavé preskúmať aj ďalšie softvérové nástroje a metódy, ktoré by mohli pomôcť pri riešení ešte zložitejších celočíselných úloh. Tiež by bolo prínosné aplikovať tieto metódy na reálne väčšie problémy, aby sa overila ich praktická použiteľnosť a efektivita.

ZOZNAM POUŽITEJ LITERATÚRY

- [1] BEREŽNÝ, Štefan a KRAVECOVÁ, Daniela. *Lineárne programovanie*. Košice: Katedra matematiky a teoretickej informatiky Fakulta elektrotechniky a informatiky Technická univerzita v Košiciach, 2012. ISBN 978-80-553-0910-1.
- [2] BIOGRAPHICAL MEMOIRS. [Online]. Dostupné z: <https://www.nasonline.org/wp-content/uploads/2024/06/dantzig-george-b.pdf>. [cit. 2025-03-01].
- [3] DANTZIG, George B. *Lineárne programovanie a jeho rozvoj*. Edícia ekonomickej literatúry (Alfa). Bratislava: Slovenské vydavateľstvo technickej literatúry, 1966.
- [4] DEMEL, Jiří. *Operační výzkum* [online]. 2017 [cit. 2025-04-05], učební text ČVUT. Dostupné na: <https://kix.fsv.cvut.cz/~demel/ped/ov/ov.pdf>.
- [5] GASS, Saul I. a ASSAD, Arjang A. *An annotated timeline of operations research: an informal history*. International series in operations research & management science. Boston: Kluwer Academic Publishers, 2004. ISBN 1-4020-8112-X.
- [6] GOMORY, Ralph E. Outline of an algorithm for integer solutions to linear programs: Bulletin of the American Mathematical Society, 64. Online. In: . 1958, s. 275-278. Dostupné z: <https://www.ams.org/journals/bull/1958-64-05/S0002-9904-1958-10224-4/>. [cit. 2025-03-04].
- [7] HLADÍK, Milan. *Celočíselné programování: text k přednášce*. Univerzita Karlova v Praze Praha, 2017, 90 s. Online. Dostupné z: https://kam.mff.cuni.cz/~hladik/CP/text_cp.pdf. [cit. 2025-05-02].
- [8] HRABEC, Dušan. *Matematika v bezpečnostních technologiích: AP3MB*. UTB, Fakulta aplikované informatiky, Ústav matematiky.
- [9] JABLONSKÝ, Josef. *Operační výzkum*. Praha: Vysoká škola ekonomická, 1996. ISBN 8070790318
- [10] KORDA, Benedikt. *Učebnice lineárního programování*. Praha: Státní nakladatelství technické literatury, 1962.
- [11] KOŘENÁŘ, Václav a LAGOVÁ, Milada. *Optimalizační metody*. V Praze: Oeconomica, 2003. ISBN 8024506092
- [12] LAGOVÁ, Milada a JABLONSKÝ, Josef. *Lineární modely*. Vyd. 3. Praha: Oeconomica, 2014. ISBN 978-80-245-2020-9.
- [13] MÁČA, Jaromír a LEITNER, Bohuš. *Operační analýza I*. 2 vyd. Žilina: Žilinská univerzita v Žiline, Fakulta speciálneho inžinierstva, detašované pracovisko Košice (Žilina), 2002. ISBN 8088829399.
- [14] MATOUŠEK, Jiří. *Lineární programování: Úvod pro informatiky*. Online. In: . Verze

27/VI/2006. Dostupné z: <https://iti.mff.cuni.cz/series/2006/311.pdf>. [cit. 2025-02-28].

[15] MICROSOFT CORPORATION. *Microsoft Excel 365* [softvér]. Redmond, WA: Microsoft, 2024. Prístupné cez predplatné Microsoft 365

[16] OpenSolver [online]. [s.l.]: OpenSolver.org [cit. 2025-05-24]. Dostupné z: <https://opensolver.org>

[17] PLESNÍK, Ján, DUPAČOVÁ, Jitka, VLACH, Milan, MASARYK, Ján a ŠIMKOVIC, Ján. *Lineárne programovanie: Metódy a aplikácie*. 2. vyd. Bratislava: Alfa, 1972. 397 s.

[18] PROKOP, Roman. *Metody optimalizace*. Projekt: CZ.02.2.69/0.0/0.0/16_018/0002381. 2016 - 2020. Rozvoj výzkumně zaměřených studijních programů na FAI.

[19] SolverStudio [online]. [s.l.]: SolverStudio.org, [bez dátumu] [cit. 2025-05-25]. Dostupné z: <https://solverstudio.org/>

ZOZNAM OBRÁZKOV

Obrázok 1. n rozmerný priestor.....	21
Obrázok 2. Schéma simplexovej metódy [8]	22
Obrázok 3. Zobrazenie doplnku Riešiteľ na páse kariet v Exceli.	49
Obrázok 4. Parametre doplnku Riešiteľ	50
Obrázok 5. Východisková tabuľka.....	51
Obrázok 6. Pridanie podmienky celočíselnosti	52
Obrázok 7. Zobrazenie doplnku OpenSolver na páse kariet v Exceli.....	55
Obrázok 8. Zobrazenie SolverStudia na karte Údaje v Exceli	59

ZOZNAM TABULIEK

Tabuľka 1. Zloženie kávových zmesí.	17
Tabuľka 2. Schéma východiskovej simplexovej tabuľky.	23
Tabuľka 3. Východisková simplexová tabuľka	24
Tabuľka 4. Simplexová tabuľka, určenie kľúčového stĺpca a riadku.....	26
Tabuľka 5. Výsledná simplexová tabuľka po druhej iterácii	27
Tabuľka 6. Východisková simplexová tabuľka	42
Tabuľka 7. Prepočítaná simplexová tabuľka.....	42
Tabuľka 8. Riešenie simplexovej tabuľky a výber riadku pre Gomoryho rez	43
Tabuľka 9. Simplexová tabuľka po prvej iterácii.....	45
Tabuľka 10. Doplnený riadok o nové obmedzenie. Druhá iterácia a výsledné riešenie	46
Tabuľka 11. Výsledok riešenia.....	53
Tabuľka 12. Zostava s odpoveďou.....	54
Tabuľka 13. Distribučné náklady	55
Tabuľka 14. Východisková tabuľka pre riešenie úlohy pomocou doplnku OpenSolver	57
Tabuľka 15. Optimálne riešenie úlohy	58
Tabuľka 16. Príklad č. Východiskové riešenie	60
Tabuľka 17. Výsledné riešenie úlohy.....	61

ZOZNAM PRÍLOH

Príloha 1. Python skript.....	69
-------------------------------	----

Príloha 1. Python skript

```
import xlwings as xw

from pulp import LpProblem, LpMinimize, LpVariable, lpSum, LpInteger,
LpStatus

# Prístup k excel tabulke
wb = xw.books.active
sheet = wb.sheets.active

# Nazvy miest pekarni a predajni
factories = ['Trnava', 'Ružomberok', 'Poprad', 'Zvolen']
stores = ['Bratislava', 'Nitra', 'Žilina', 'Banská Bystrica',
'Košice']

# Nacitanie nakladov z buniek B2:F5
costs = {}
for i, factory in enumerate(factories):
    for j, store in enumerate(stores):
        costs[(factory, store)] = sheet.range((2 + i, 2 + j)).value

# Vyrobné kapacity pekarni nacistane z buniek G2:G5
capacities = {}
for i, factory in enumerate(factories):
    capacities[factory] = sheet.range((2 + i, 7)).value

# Dopyt predajni nacistany z buniek B6:F6
demands = {}
for j, store in enumerate(stores):
    demands[store] = sheet.range((6, 2 + j)).value

# Vytvorenie LP modelu
model = LpProblem("Minimalizacia_nakladov", LpMinimize)

# Definovanie podmienky nezapornosti a celociselnosti
```

```

x = LpVariable.dicts("x", (factories, stores), lowBound=0,
cat=LpInteger)

# Cielova funkcia, minimalizacia celkovych nakladov
model += lpSum(costs[(i, j)] * x[i][j] for i in factories for j in
stores)

# Kapacitne obmedzenia
for i in factories:
    model += lpSum(x[i][j] for j in stores) <= capacities[i],
f"Kapacita_{i}"

# Dopytove obmedzenia
for j in stores:
    model += lpSum(x[i][j] for i in factories) == demands[j],
f"Dopyt_{j}"

# Riesenie modelu
model.solve()

# Vystup riesenia. Zapis hodnot do tabulky zacina v riadku 8, stlpec
B
start_row = 8
start_col = 2

# Vytvorenie hlavicky miest predajni v tabulke s vysledkami
for j, store in enumerate(stores):
    sheet.range((start_row, start_col + j)).value = store

# Zapis vysledkov do tabulky pre hodnoty > 0, inak zostane bunka
prazdna
for i, factory in enumerate(factories):
    sheet.range((start_row + 1 + i, start_col - 1)).value = factory
    for j, store in enumerate(stores):
        value = x[factory][store].varValue
        if value and value > 0:

```

```

        sheet.range((start_row + 1 + i, start_col + j)).value =
int(value)
    else:
        sheet.range((start_row + 1 + i, start_col + j)).value =
"""

# Zapis vyrobeneho mozstva/kapacit pekarni do stlpca G9:G12
for i, factory in enumerate(factories):
    sheet.range((9 + i, 7)).value = capacities[factory]

# Zapis dodaneho mnozstva/dopyt pekarni
sum_row = start_row + 1 + len(factories)
sheet.range((sum_row, start_col - 1)).value = "Dodané"
for j in range(len(stores)):
    total_delivery = sum(x[f][stores[j]].varValue or 0 for f in
factories)
    sheet.range((sum_row, start_col + j)).value = int(total_delivery)

# Nastavenie pozície výstupu cielovej funkcie, stĺpec I, riadok 1
output_col = 9
summary_row = 1

# Vypis cielovej funkcie, celkových minimalnych nakladov
sheet.range((summary_row, output_col)).value = "Minimálne náklady:"
sheet.range((summary_row, output_col + 1)).value =
model.objective.value()

```