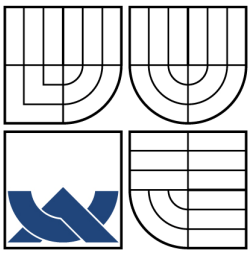
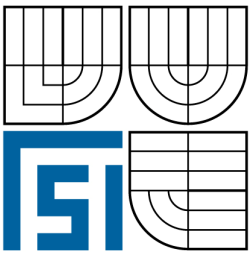


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV METROLOGIE A ZKUŠEBNICTVÍ

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF METROLOGY AND QUALITY ASSURANCE TESTING

OPTIMALIZACE NÁKUPU HUTNÍHO MATERIÁLU

OPTIMIZATION OF METALLURGICAL MATERIAL PURCHASE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. DARIA ZORINA
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. ALOIS FIALA, CSc.
SUPERVISOR

BRNO 2009

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav metrologie a zkušebnictví
Akademický rok: 2008/09

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Zorina Daria

který/která studuje v **magisterském studijním programu**

obor: **Metrologie a řízení jakosti (3911T032)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Optimalizace nákupu hutního materiálu

v anglickém jazyce:

Optimization of metallurgical material purchase

Stručná charakteristika problematiky úkolu:

1. Definování cílů práce.
2. Analýza problému a literární rešerše.
3. Návrh řešení.
4. Ověření návrhu.
5. Diskuse výsledků.
6. Závěry a doporučení.

Cíle diplomové práce:

Zmapování procesů v určené oblasti
Vytvoření matematického modelu
Vytvoření kriteriální rovnice
Odvození optimálního řešení

Seznam odborné literatury:

Basl, J., Tůma, M., Glasl, v.: Modelování a optimalizace podnikových procesů. ZČU v Plzni, Plzeň, 2002

Fiala, A.: Automatizované systémy řízení slévárenských procesů. Skriptum, VUT v Brně, Brno, 1986

Vedoucí diplomové práce:doc. Ing. Alois Fiala, CSc.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2008/09.

V Brně, dne 20.11.2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA STROJNÍHO INŽENÝRSTVÍ DĚKANÁT

doc. Ing. Leoš Bumbálek, Ph.D.
Ředitel ústavu

doc. RNDr. Miroslav Doupovec, CSc.
Děkan fakulty

ABSTRAKT

Tato diplomová práce se zabývá optimalizací procesu nákupu a dělení hutníno materiálu s použitím řešení úlohy o optimálním dělení materiálu, jednou z úloh lineárního programování. Cílem při řešení této úlohy je minimalizace odpadu a minimalizace nákladů na nákup výchozího materiálu. Návrh řešení je určen pro podmínky společnosti KULIČKOVÉ ŠROUBY KUŘIM,a.s.

Základem práce je cyklus PDCA Edwardsa Deminga **(„**plánuj-dělej-kontroluj-jednej“**)** jako základní způsob pro efektivní řešení a zlepšování výrobních aktivit, procesů a systému.

Klíčová slova

Management, nákup materiálu, náklady, lineární programování, optimalizace

ANNOTATION

This diploma thesis deals with optimization of process metallurgical material purchase and cutting with use of the problem solving of optimal material dividing, which is one of the linear programming problems. The aim of this problem solving is to minimize waste and to minimize the purchase costs for base material. The solution of this problem has the practical application at the company KULICKOVE SROUBY KURIM,a.s.

This work is based on Deming Cycle (“Plan-Do-Check-Act”), as the basic method of solution efficiency and operations, processes and systems improvement.

Key words

Management, material procurement, charges, linear programming, optimization

BIBLIOGRAFICKÁ CITACE

ZORINA, D. *Optimalizace nákupu hutního materiálu.* Brno: Výsoké učení technické v Brně, Fakulta strojního inženýrství, 2009. 65 s. Vedoucí diplomové práce doc. Ing. Alois Fiala, CSc.

Prohlášení

Prohlašuji, že jsem diplomovou práci na téma «Optimalizace nákupu hutního materiálu» vypracovala samostatně s použitím odborné literatury a pramenů, uvedených na seznamu, který tvoří přílohu této práce.

Datum …….

Daria Zorina

Poděkování

Děkuji tímto doc. Ing. Aloisu Fialovi, Ing. Martinu Bajerovi, MBA, CSc., RNDr. Jiří Dvořákovi, CSc. za cenné připomínky a rady při vypracování diplomové práce.

OBSAH

Abstrakt …… 4
Prohlášení …… 5
Poděkování ……... 6
Obsah ……... 7
Úvod …… 8
1 POPIS SPOLEČNOSTI «KULIČKOVÉ ŠROUBY KUŘIM, a.s.» ……. 9
1.1 Profil společnosti …… 9
1.2 Kuličkové šrouby ……... 9
2 TEORETICKÝ ZÁKLAD …… 11
2.1 Analýza problému …….. 11
2.2 Lineární programování a simplexová metoda …… 15
2.3 Dvoufázová simplexová metoda …… 22
2.4 Metody řešení úloh celočíselného programování …….. 23
2.5 Řešení úloh lineárního programování v MS EXCEL …… 24
3 NÁVRH ŘEŠENÍ ……. 29
3.1 Zadání ……. 29
3.2 Minimalizace odpadů ……. 31
3.2.1 Vypracování řezných plánů …… 31
3.2.2 Optimalizace v MS Excel ……... 37
3.3 Použití programu «Optimizer» ……... 43
3.3.1 Popis programu ……... 43
3.3.2 Příručka k používání programu «Optimizer» ……. 43
4 ANALÝZA VÝSLEDKŮ ……. 46
Závěr …… 47
Seznam použitých zdrojů ……. 48
Příloha A – Algoritmus programu «Optimizer» …….. 49
Příloha B – Text programu «Optimizer» …… 51

ÚVOD

V současné době věnují spotřebitelé stále větší pozornost kvalitě výrobků a dodacím lhůtám. Společnosti, které nemohou zajistit požadovanou úroveň kvality a mají vysoké náklady, vystavují svůj byznys značnému nebezpečí. Pro splnění požadavků spotřebitelů a rovněž pro svou konkurenceschopnost musí hledat vedení společností co nejvíce úsporné cesty nepřetržitého zlepšení kvality a efektivnosti výrobků. Manažerové si uvědomují, že řízení jakosti výrobků (služeb), jehož základem je plánování, evidenci, analýza a audit, je stěžejním základem jejich prosperity.

Jedním ze způsobů užitečného řízení nákladů je optimalizace – ať už v podobě stanovení optimální ceny, množství nebo vytížení výrobních kapacit a plné využitelnosti výrobních zdrojů. Pro řešení těchto problémů se používají mimo jiné modely lineárního programování, využívající k nalezení optimálního řešení tzv. simplexovou metodu.

Problematika řešená v této práci je zaměřena na optimální dělení materiálu, tzv. řezný problém, který patří k typickým úlohám lineárního programování.

Kromě toho je optimalizace procesu zaměřena i na nákup materiálu, který musí přinášet ekonomické výhody. V práci jsou zohledněny a optimálizovány ekonomické náklady, s ohledem na aktuální údaje a vstupy. Výsledky práce jsou vizualizovány a představeny ve formě pohodlného uživatelského rozhraní, které umožňuje zadávat vstupní údaje a dostávat optimální výsledky bez odborných znalosti v oblasti programování.

1 POPIS SPOLEČNOSTI «KULIČKOVÉ ŠROUBY KUŘIM, a.s.»

1.1 Profil společnosti

Kuličkové šrouby se v Kuřimi vyrábí od počátku 70. let, do roku 1996 v rámci firmy TOS KUŘIM. V roce 1996 vzniká privatizací nová společnost TOS KUŘIM–KŠ, s.r.o. a v roce 1997 dochází ke změně názvu firmy na KULIČKOVÉ ŠROUBY KUŘIM. Od 1.1.2001 došlo k transformaci na akciovou společnost. Od 1.1.2005 je majoritním vlastníkem KSK česká obchodní společnost ALTA, a.s. se sídlem v Brně, Štefánikova 41, č.p.110.

Kuličkové šrouby jsou vyráběny ve třídách přesnosti IT1, IT3 a IT5 dle norem ISO a DIN a splňují nejnáročnější požadavky konstrukcí obráběcích strojů, automatických linek, jednoúčelových strojů spočívající na technických parametrech tuhosti, trvanlivosti, vysoké účinnosti a plynulého chodu.

Pro konvenční obráběcí stroje a užití v jiných průmyslových odvětvích vyrábí a dodává společnost KULIČKOVÉ ŠROUBY KUŘIM trapézové šrouby odpovídající normě ISO a ČSN.

Dalším produktem společnosti jsou lineární valivá vedení s válcovými vodícími tyčemi. Lineární valivá vedení se vyznačují vysokou únosností, přesností, malými pasivními odpory. Pro lineární valivá vedení jsou používána lineární ložiska zahraničních výrobců.

Vysoká úroveň systému jakosti byla potvrzena auditem zavedeného systému jakosti podle standardu ISO 9001 certifikovaného společností RW TÜV Nord Cert. [1]

1.2 Kuličkové šrouby

Kuličkové šrouby jsou konstrukční prvky pohybových ústrojí převádějící s vysokou účinností rotační pohyb na přímočarý, vyznačující se vysokou tuhostí, přesností a trvanlivostí.

Kuličkové šrouby vyžadují přesné a tuhé uložení s rovnoběžností kuličkového šroubu a vodících ploch do 0,02 mm/1000 mm, rovněž uložení maticové jednotky musí zajišťovat její kolmost k podélné ose šroubu do 0,02 mm/1000 mm.

Maticové jednotky mohou být zatěžovány pouze v axiálním směru. U dlouhých a štíhlých kuličkových šroubů musí být konstrukcí pohybového ústrojí vhodně eliminován průhyb hřídele vzniklý jeho hmotností. [1]

Konstrukční provedení maticivých jednotek:

Nepředepnutá válcová matice bez příruby

Nepředepnutá válcová matice s přírubou

Předepnutá dvojice válcových matic bez příruby

Předepnutá dvojice matic s přírubou

Předepnutá matice s přírubou

Předepnutá dvouchodá rychloběžná matice s přírubou

Předepnutá čtyřchodá rychloběžná matice s přírubou

Nepředepnutá matice s přírubou pro válcovaný hřídel

Předepnutá dvojice matic v kostce

Kuličkový šroub s integrovaným servomotorem

Kuličkové šrouby jsou vyráběny ve 3 třídách přesnosti, přičemž úchylky stoupání odpovídají normě ISO, jak je uvedeno v následující tabulce: [1]

Tab.1.1 Třídy přesnosti kuličkových šroubů

ÚCHYLKA STOUPÁNÍ NA DÉLCE ZÁVITU	**Broušený závit**			**Válcovaný závit**	
	IT 1	**IT 3**	**IT 5**	**T5**	**T5**
300 mm	**0,006**	**0,012**	**0,023**	**0,023**	**0,052**

S ohledem na technologické možnosti jsou v závislosti na třídě přesnosti stanovena následující omezení závitových délek: [1]

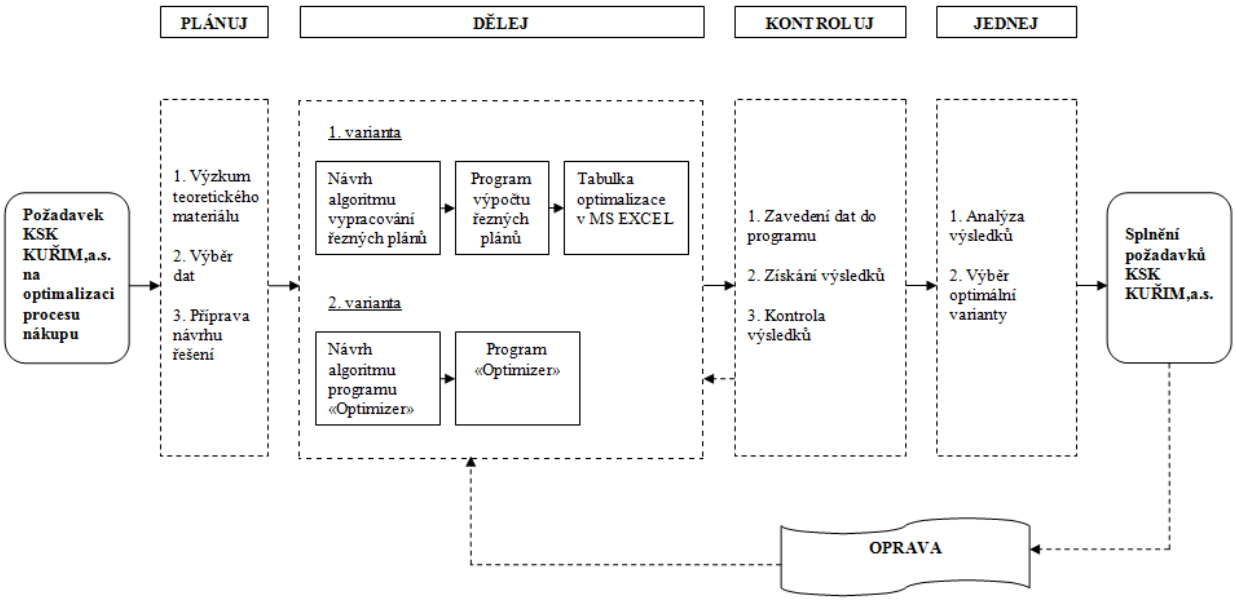
Tab.1.2 Délky závitu kuličkových šroubů

TŘÍDA PŘESNOSTI		JMENOVITÝ PRŮMĚR ZÁVITU [mm]										
		12	16	20	25	32	40	50	63	80	100	125
BROUŠENÝ ZÁVIT	IT 1	200	250	250	350	700	1 200	1 500	2 000	2 500	3 000	3 000
	IT 3	250	300	320	500	1 100	1 800	2 500	3 000	3 500	4 000	4 000
	IT 5	300	350	500	800	1 500	2 500	3 500	4 000	4 500	5 000	4 000
VÁLCOVANÝ ZÁVIT	IT 5				4 000	5 000	6 000	6 000	6 000	6 000		
	IT 7				4 000	5 000	6 000	6 000	6 000	6 000		

2 TEORETICKÝ ZÁKLAD

2.1 Analýza problému

Pro lepší představu o postupu řešení problému, zobrazíme model řešení graficky. Na základě je cyklus PDCA Edwardsa Deminga („plánuj-dělej-kontroluj-jednej"), viz obrázek 2.1.



Použité značky:

→ - akce

⇢ - informace

Obr. 2.1 Model řešení problému

Každý krok tohoto cyklu obsahuje konkrétní akci procesu řešení problému. Pro začátek je nutno prozkoumat teoretický materiál pro daný problém. Potom pokračujeme podle modelu, který odráží i zpětnou vazbu od podniku ve formě informace o splnění požadavků a další návrhy na zlepšení.

Cílem aktivity každého obchodního podniku je zisk, který může zajistit jeho další vývoj. Ziskovost je nejen hlavním cílem, ale také jako hlavní podmínka pracovní aktivity podniku, jako výsledek jeho činnosti, efektivnosti jeho aktivit pro zajištění potřeb odběratelů ve formě zboží či služeb dle existující poptávky.

Nejdůležitějšími faktory, určujícími zisk, je:

- zavedení inovací
- schopnost podstoupit riziko (riziko jak zdroj zisku)
- racionální použití prostředků a zdrojů
- vyváženost optimálních objemů činnosti (výběr takového měřítka podniku, který umožňuje zajistit optimální rentabilitu).

Zavedení inovací jako zdroj zisku předpokládá výrobu (prodej) nového zboží (služby) větší kvality, osvojení nového trhu, organizačně-správní inovace, osvojení nových zdrojů dodávek zboží.

Z těchto důvodů byla společnost KULIČKOVÉ ŠROUBY KUŘIM,a.s. postavena před problém optimalizace nákupu hutního materiálu a jeho řezání pro výrobní proces okružování kuličkových šroubů. Především budou posouzeny podmínky práce.

Podnik uvedl do provozu nové technologické vybavení – stroj pro rotační frézování závitu «Leistritz PW160». Pro frézování závitu používají kalené a broušené ocelové tyče (Cf53 a 42CrMo4) o průměru 40, 63, 80 a 100 mm, 6,5 a 8,5 metrů dlouhé. Řezaná délka polotovaru je u jednotlivých výrobků rozdílná podle požadavku zákazníka.

Z rozdílných požadavků vzniká problém efektivního nákupu nutného množství požadovaného materiálu v přijatelných cenách, který řeší nákupní logistika.

Hlavním cílem nákupní logistiky je zajištění potřeb výroby materiálem s maximálně možnou ekonomickou efektivností.

Systém výroby je charakterizován:

- počtem dodavatelů
- rovnoprávností partnerů
- důležitost smluvích závazků na nakupování zboží

- svobodou tvorby cen
- konkurencí dodavatelů a zákazníků
- ekonomickou odpovědností stran
- iniciativou, samostatností a podnikavostí prodejce a zákazníků.

Existují následující funkce nákupní logistiky:

- určování potřeby materiálových zdrojů
- získání a oceňování návrhů
- výběr dodavatelů
- sladění ceny a smluvních podmínek
- formování zakázek
- kontrola množství a lhůt dodávek
- vstupní kontrola a rozmístění materiálových zdrojů na skladu.

Pro efektivní fungování logistiky nakupování je nutno sestavit plán nakupování, který řeší následující úlohy:

- určování potřeby materiálů
- určování metody nakupování
- sladění ceny a smluvních podmínek
- vedení monitoringu a kontroly počtu, kvality a lhůt dodávek
- organizace rozmístění zboží na skladě

Pro vyřešení problému optimalizace procesu nákupu hutního materiálu je na začátku nutné stanovit nejvíce problémový faktor, účinkující na proces a potom definovat metodu jeho inovace.

V současné době existuje dostatek postupů řízení jakosti pro hodnocení a analýzy procesů. Velice široké uplatnění má tak zvaný Ishikawův diagram. Tento diagram důvodů a následků je prostředek, který umožňuje vyjádřit tyto vztahy v jednoduché a dostupné formě. Je to nástroj, který umožňuje odhalit nejpodstatnější faktory (příčiny), ovlivňující konečný důsledek (následek).

Veškeré možné příčiny jsou klasifikovány na základě principu <5M>: [2]

1. Man (člověk) – příčiny, spojené s lidským činitelem
2. Machines (stroje, zařízení) - příčiny, spojené s zařízením
3. Materials (materialy) - příčiny, spojené s materiály
4. Methods (metody) - příčiny, spojené s technologií práce, organizací procesů

5. Measurements (měření) **-** příčiny, spojené se způsoby měření

Zkoumaný jev se zobrazuje z pravé strany schématu jako kořen stromovitého diagramu. Horizontálně od kořene diagramu do levého kraje listu se nanáší centrální osa diagramu, podobná kmenu stromu. K centrální ose diagramu Ishikawy se přidává pět větví, každá z nich odpovídá svému typu příčin, svému <M>.

Dále na každé větvi zvlášť, jako na ose, se kreslí jednotlivé «malé větve», každá z nich reprezentuje zvláštní příčinu ve svém typu. Jestliže by se pokračovalo tak dál, dostaneme strom, který odráží příčiny problému.

Faktory, které mohou působit na proces nákupu materiálu:

Člověk (1):

- 1a kompetentnost
- 1b rychlost realizace zakázky
- 1c zájem o práci
- 1d únava

Stroje (2):

- 2a existence nutného zařízení pro provedení vstupní kontroly
- 2b kalibrační zařízení pro provedení vstupní kontroly
- 2c zařízení pro skladování materiálu

Materiály (3):

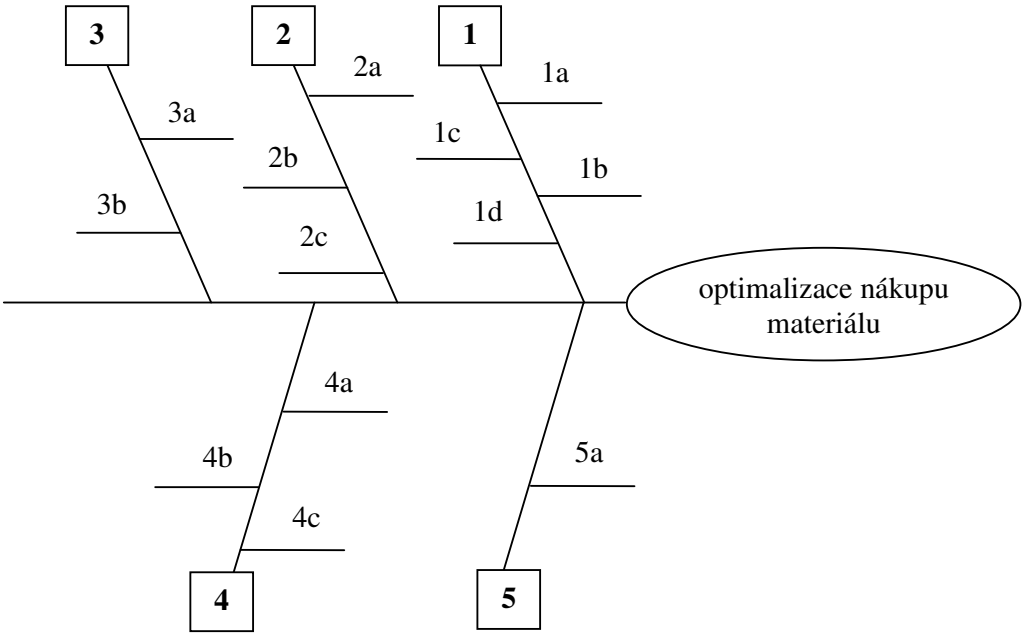
- 3a ceny materiálu
- 3b jakost materiálu

Metody (4):

- 4a technologie vstupní kontroly
- 4b metodika optimálního řezání počáteční délky
- 4c výpočet nutného materiálu

Měření (5):

- 5a výpočty



Obr. 2.2 Ishikawův diagram

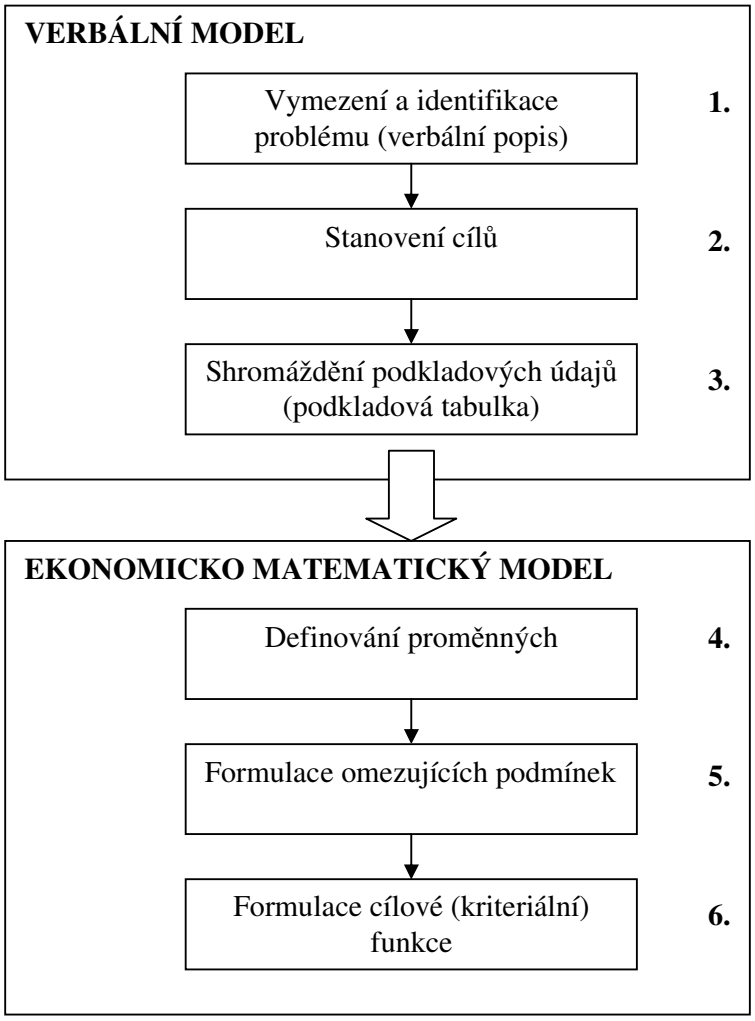
Poté, co je diagram vytvořen, následuje krok rozložení příčin podle stupně jejich důležitosti.

Výsledkem analýzy příčin byly následující závěry: nezpracována metodika optimálního řezání a výběr počáteční délky polotovaru, kvůli čemuž má podnik vysoké ekonomické náklady na nákup materiálu a vysokou míru odpadu.

Stejně tak je problém optimalizace nákupu materiálu obsažen i ve zpracování řezného plánu s cílem snížení ekonomických nákladů a jeho vizualizace pro další využití podnikem.

2.2 Lineární programování a simplexová metoda

V poslední době se stále silněji zdůrazňuje potřeba uplatňovat vědecké objektivní metody na všech úsecích řízení. Ekonomické vztahy jsou tak složité, že i na poměrně úzkém úseku rozhodování by se mohl dopouštět vážných chyb každy, kdo by chtěl vycházet pouze z praktických zkušeností a objektivnosti. Dnes už není vážnější překážkou ani problém, kde a jak zvládnout velký objem výpočetních prácí, jež se obvykle vyskytnou v souvislosti s užívíním matematiky. Některé matematické metody jsou tak jednoduché, že mohou účinně prospět i podniku, který nemá mezi svými zaměstnanci žádného specialistu v oboru matematiky. Platí to zejména o metodách lineárního programování, jež pro tuto vlastnost dosáhly v posledních letech velkého rozšíření v hospodářské praxi ve všech průmyslově vyspělých státech.



Obr. 2.3 Postup při formulaci modelu LP [3]

Simplexová metoda je nejznámější a nejefektivnější metoda pro řešění všech úloh linearního programování (LP). Je to metoda obecná, nebot' umožňuje řešit každý rozhodovací problém formulovaný jako model LP, a iterační, nebot' výpočet postupuje po krocích.

Lze jí řešit všechny modely LP upravené do kanonického tvaru. Model LP je v kanonickém (též bazickém) tvaru, jestliže jeho matice soustavy obsahuje jako svou submatici úplnou jednotkovou matici řádu *m*, tj. po přerovnání sloupců matice soustavy lze psát: [3]

$$A = (\tilde{A}|E), \tag{2.1}$$

kde A je matice soustavy,

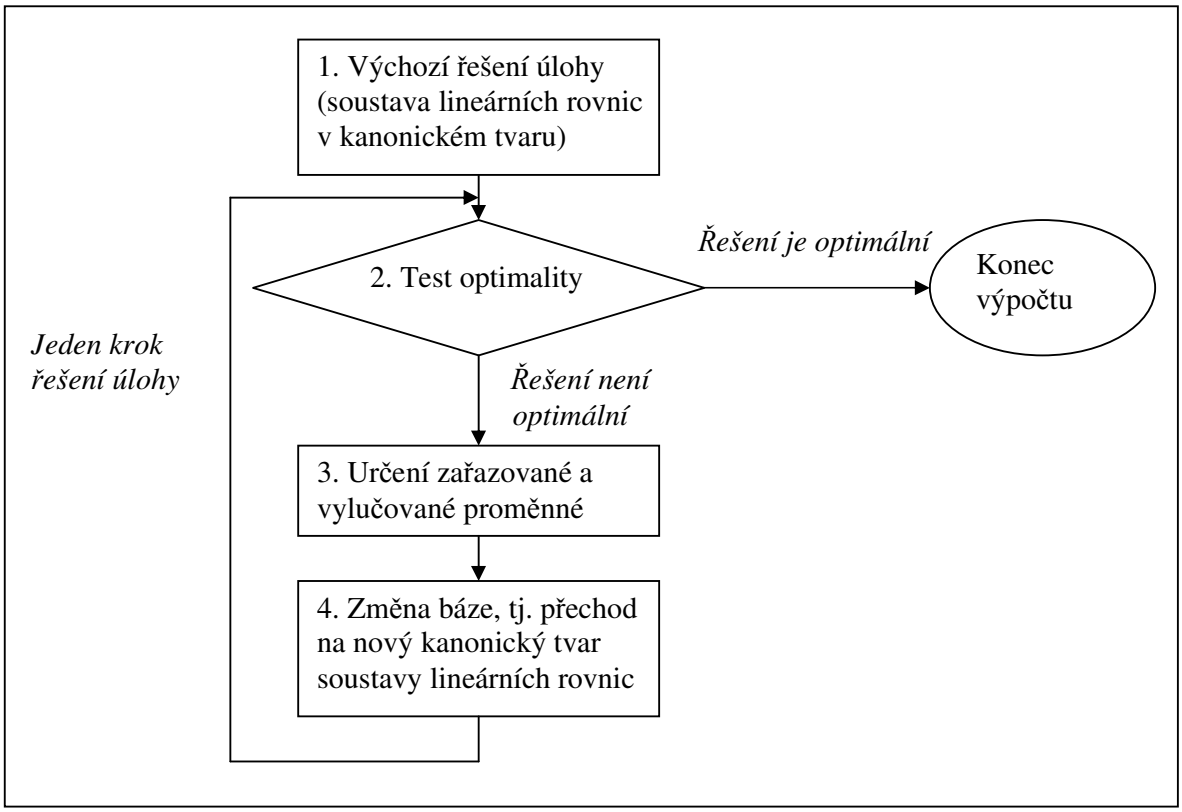
$\tilde{A}$ je čast matice soustavy po vyřazení sloupců jednotkových vektorů,

E je jednotková matice řádu *m.*

V principu jde o eliminační metodu pro řešení soustavy m lineárních rovnic o n proměnných, kde $m < n$, tj. počet rovnic je menší než počet neznámých, která je doplněna o vhodná kritéria ukazující, jak lze dojít k optimálnímu řešení.

Simplexová metoda byla vyvinuta v r. 1947 B. Dantzigem. Objev algoritmu simplexové metody společně s rozvojem výpočetní techniky v létech 1947-1956 byl rozhodujícím mezníkem pro rozvoj metod operačního výzkumu a významně ovlivnil i rozvoj některých matematických disciplín.

Na obrázku 2.4 je uveden postup při aplikaci simplexové metody. Algoritmus předpokládá postupné provádění následujících operací:



Obr. 2.4 Vývojový diagram simplexové metody

Simplexová metoda je založená na Gaussově – Jordanově metodě úplné eliminace. Nejdříve se definuje tzv. základní řešení, které je potom v iteračních krocích zlepšováno. Cílem každé další iterace je nalézt takové řešení, aby se hodnota účelové funkce zvýšila nebo alespoň nezměnila, příp. snížila (v závislosti na typu úlohy – maximalizační, příp. minimalizační úloha). Touto metodou by se mělo podařit nalézt optimální řešení dané úlohy lineárního programování.

Optimalizačním problémem budeme rozumět úlohu nalezení optimuma (maxima nebo minima) funkce:

$$z = f(x_1, x_2, \ldots, x_n) \tag{2.2}$$

za podmínek

$$g_i\,(x_1, x_2, \ldots, x_n) \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} 0;\ i=1,2, \ldots, m \tag{2.3}$$

a pro ekonomické úlohy je třeba předpokládat, že

$$x_1 \geq 0,\ x_2 \geq 0,\ \ldots,\ x_n \geq 0 \tag{2.4}$$

Najít optimum funkce (2.2) za podmínek (2.3) a (2.4) znamená určit hodnoty proměnných $x_1, x_2, \ldots, x_n$ tak, aby splňovaly dané podmínky a zároveň optimalizovaly hodnotu funkce (2.2).

Rozepsaný tvar bude vypadat takto: [4]

maximalizovat (minimalizovat)

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

za podmínek

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m \end{aligned}$$

a nechť dále platí předpoklad

$$\begin{gathered} x_j \geq 0,\ j=1, 2, \ldots, n \\ b_i \geq 0,\ i=1, 2, \ldots, m \end{gathered}$$

kde n – počet strukturních proměnných modelu

m – počet vlastních omezení modelu

c_j – cenový koeficient příslušející j–té proměnné

b_i – hodnota pravé strany příslušející i–tému vlastnímu omezení

a_{ij} – strukturní koeficient vyjadřující vztah mezi i–tým omezením a j–tou proměnnou

Celkový postup při formulaci modelu LP je uveden tabulce 2.1

Tab.2.1 Postup při formulaci modelu LP [3]

Postup formulace úlohy	Postup tvorby modelu úlohy
1. Popis procesů, např. vektory $\begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} \cdots \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix}$	Jde o popis jednotlivých, izolovaných procesů. Neuvažujeme zatím ještě vůbec vztahy mezi všemi procesy. Neuvažujeme o možné úrovni procesů. Modelujeme tedy jednotlivé procesy.
2. Zavedení neznámých x_i a vymezení jejich vlastností (nazápornosti $x_j \geq 0$)	Modelujeme sice ještě jednotlivé procesy, ale začínáme uvažovat o jejich možné velikosti. Popisem $x_j \geq 0$ charakterizujeme velikost procesů stále ještě velmi neurčitě.
3. Formulace podmínek plynoucích z úlohy $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$ $a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$ …… $a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m$	Začínáme modelovat vztahy mezi všemi procesy úlohy a charakterizujeme je soustavou nerovnic a později soustavou rovnic.
4. Popis přípustných řešení: nezáporná x_j na množině řešení soustavy lineárních rovnic (krok 3)	Shrnutím vlastností neznámých a omezení či požadavků vyplývajících z úlohy získáme model přípustných řešení. Tento model zahrnuje řešení matematicky přípustná, technicky a technologicky uskutečnitelná, ekonomicky výhodná i nevýhodná.
5. Formulace účelové funkce $z = c_1x_1 + \ldots + c_nx_n$	Matematicky vyjadřujeme hledisko pro posuzování výhodnosti popsaných přípustných řešení.

Ze základní věty lineárního programování: „Jestliže má úloha lineárního programování optimální řešení, potom má také optimální řešení základní," [5] lze vyvodit, že optimální řešení úlohy nalezneme mezi konečným počtem základních řešení. Počet základních řešení se dá spočítat pomocí kombinačního čísla

$$\binom{m+n}{m} = \frac{(m+n)!}{m!*n!} \tag{2.5}$$

Mezi těmito základními řešeními se nacházejí základní přípustná řešení, kterých je tedy také konečný počet. Mezi těmito je pak postupným „prohledáváním" množiny nalezeno optimální řešení s nejlepší hodnotou účelové funkce.

Nejdříve je třeba ukázat, jak se dostaneme k tzv. výchozímu základnímu řešení: Omezení modelu se musí nejprve upravit na tzv. kanonický tvar, kdy soustava daných rovnic obsahuje jednotkovou submatici koeficientů. Toho se dosáhne vyrovnáním nerovnic na rovnice pomocí tzv. přídatných proměnných y_i ($i = 1 \ldots n$). U nerovnic typu menší nebo rovno je pro vyrovnání přičtena kladná přídavná proměnná k levé straně omezení. Po této úpravě

má soustava stejný počet proměnných s jednotkovým vektorem, jako je počet rovnic. Účelová funkce se převede do anulovaného tvaru, kdy se všechny proměnné nacházejí na levé straně.

Pro snazší a přehlednější výpočet je dobré si úlohu lineárního programování přepsat do tzv. simplexové tabulky.

Řádky tabulky odpovídají jednotlivým omezením úlohy, sloupce značí proměnné úlohy. Do předposledního sloupce se zapíší hodnoty pravých stran omezení, poslední sloupec obvykle bývá využit pro hodnotu t (o hodnotě t bude řeč dále). Pro zápis koeficientů účelové funkce se použije poslední řádek.

V první části tabulky se nacházejí proměnné, které označujeme jako základní. Jejich hodnoty jsou ve výchozím řešení rovny pravým stranám omezení $b_1 \ldots b_m$. Ostatní proměnné jsou nezákladní a jsou rovny nule.

V případě maximalizační úlohy jsou koeficienty v řádce účelové funkce z_j záporné. Udávají, o kolik se zvýší hodnota účelové funkce v případě změny hodnoty proměnné x_j o jednu jednotku.

S tímto upraveným modelem se tedy postupuje řešením jednotlivých iterací až k hledanému řešení. Každá iterace se dá rozdělit do tří částí [5]:

1. Test optima a určení vstupující proměnné

V řádce účelové funkce se nalezne při řešení maximalizační úlohy nejmenší koeficient z_j

$$g=\min_j(z_j)=z_k \qquad \text{pro } j = 1, 2, \ldots, m+n, \tag{2.6}$$

při řešení minimalizační úlohy se hledá největší koeficient z_j

$$g=\max_j(z_j)=z_k \qquad \text{pro } j = 1, 2, \ldots, m+n. \tag{2.7}$$

Odůvodnění pro hledání příslušného g lze nalézt v anulovaném tvaru účelové funkce ve výchozím základním řešení úlohy lineárního programování:

$$z + z_1x_1 + z_2x_2 + \ldots + z_{n+m} = 0, \tag{2.8}$$

kde z_j je koeficient účelové funkce u proměnné x_j. Ve výchozím řešení jsou koeficienty z_j rovny záporným hodnotám cenových koeficientů u příslušných proměnných.

Pokud položíme všechny strukturní proměnné rovny nule, budou hodnoty přídatných proměnných rovny pravým stranám omezení a základní výchozí řešení bude vypadat takto:

$$x^{(0)} = (0, 0, 0, 0, \ldots b_1, b_2, \ldots b_m). \tag{2.9}$$

Po dosazení základního výchozího řešení (2.9) do anulovaného tvaru účelové funkce (2.8) dostaneme tvar účelové funkce

$$z = 0 - z_1 * 0 - z_2 * 0 - \ldots - z_n * 0 - 0 * x_{n+1} - \ldots - 0 * x_{n+m} = 0. \qquad (2.10)$$

Z uvedeného tvaru účelové funkce ve výchozím základním řešení (2.10) lze odvodit, že pokud je koeficient $z_j > 0$, hodnota účelové funkce klesne, a naopak pokud je $z_j < 0$, hodnota účelové funkce vzroste.

Po vyhledání příslušného koeficientu z_j se testuje optimalita základního řešení: Jestliže bylo podle (2.6) nalezeno takové g, že $g \geq 0$, je v případě řešení maximalizační úlohy základní řešení i řešením optimálním, protože hodnotu účelové funkce už zlepšit nelze. Pro minimalizační úlohu toto platí, pokud je nalezeno podle (2.7) $g \leq 0$.

Pokud je nalezené $g < 0$ (v případě minimalizace $g > 0$), označí se k-tý sloupec jako klíčový a proměnná x_k jako proměnná vstupující.

2. Určení vystupující proměnné

Vystupující proměnná se najde pomocí hodnot t:

$t = b_i / a_{ik}$ pro $a_{ik} > 0$

kde k je index klíčového sloupce a pro $i = 1, 2, \ldots, m$.

Podle $t = \min (b_i / a_{ik})$ se nalezne klíčový řádek, který se označí indexem q, a základní proměnná v tomto řádku je určena jako vystupující proměnná.

Optimální řešení neexistuje, jsou–li všechny koeficienty v klíčovém sloupci $a_{ik} \leq 0$. V takovém případě má řešená úloha neomezenou hodnotu účelové funkce.

3. Transformace řešení

Prvek, který leží zároveň v klíčovém řádku i klíčovém sloupci se nazývá klíčový prvek a má označení a_{qk}. Tímto prvkem se vydělí klíčový řádek:

$$a_{qj}^{(s+1)} = a_{qj}^{(s)} / a_{qk}^{(s)}$$

$$b_q^{(s+1)} = b_j^{(s)} / a_{qk}^{(s)}$$

kde $j = 1, 2, \ldots, n+m$, s je index iterace, k je index klíčového sloupce, q je index klíčového řádku.

Transformace ostatních prvků tabulky se provede následně:

$a_{ij}^{(s+1)}=a_{ij}^{(s)}-a_{ik}^{(s)}a_{qj}^{(s+1)}$ kde $i = 1, 2, \ldots, m+1$

$j = 1, 2, \ldots, n+m+1$, pro $i \neq q$

$b_i^{(s+1)}=b_j^{(s)}-a_{ik}^{(s)}b_q^{(s)}$ kde $i = 1, 2, \ldots, m$.

Zvýší se index iterace na $s+1$ a výpočet se opakuje až do nalezení optimálního řešení.

Popsaná metoda se někdy též označuje jako jednofázová simplexová metoda.

2.3 Dvoufázová simplexová metoda

Již zmíněná jednofázová simplexová metoda se dá použít pouze pro matematický model, který obsahuje nerovnosti typu menší nebo rovno. Pokud se ovšem v modelu vyskytují opačné nerovnosti nebo dokonce rovnice, musí se použít tzv. dvoufázová simplexová metoda.

V takovém případě po vyrovnání nerovností soustavy vlastních omezení na rovnice pomocí přídatných proměnných neobsahuje soustava jednotkovou submatici potřebnou pro další výpočet. Pro vytvoření této submatice se musí použít další typ proměnné, tzv. pomocná (někdy též umělá) proměnná y_i (pro $i = m+1 \ldots p$, kde m je počet vlastních omezení modelu, p je počet omezení modelu typu větší nebo rovno a rovnic). Pomocná proměnná se přidá k těm omezením, kde chybí přídatná proměnná s jednotkovým vektorem. Vznikne tak tzv. rozšířená soustava rovnic. Původní soustavě rovnic se rovná v případě, že všechny přidané pomocné proměnné se rovnají nule. To znamená, že v první fázi dvoufázové simplexové metody je potřeba „vynulovat“ pomocné proměnné. Vynulování těchto proměnných lze učinit dvěma způsoby:

- zavedením prohibitivních cen
- použitím pomocné účelové funkce

Prohibitivní ceny

Tento postup spočívá v přiřazení pomocné proměnné tzv. prohibitivní ceny. V případě maximalizační úlohy dáme pomocné proměnné hodně velké záporné číslo, v případě minimalizace hodně velké kladné číslo. Dále se pak úloha řeší jednofázovou simplexovou metodou.

Při určování prohibitivních cen se ovšem musí postupovat velmi opatrně. Pokud zvolíme prohibitivní cenu podobnou cenovým koeficientům v zadané úloze, nemusí se podařit

vynulovat danou pomocnou proměnnou. Při velmi vysoké prohibitivní ceně výpočet sice uspěje, bude však značně náročný.

Popis dvoufázové simplexové metody lze najít v [6].

2.4 Metody řešení úloh celočíselného programování

Jako úloha celočíselného lineárního programování se označuje taková úloha, která kromě vlastních omezení a podmínek nezápornosti obsahuje také tzv. Podmínku celočíselnosti, tj. podmínka, která zabezpečuje, aby všechny nebo některé proměnné nabývaly pouze celočíselných hodnot. Toto omezení ve většině případů bezprostředně vychází z ekonomického modelu daného problému. Vyjadřuje-li například proměnná nedělitelné výrobky, počty použitých dopravních prostředků nebo počty opakování určitých technologických procesů, potom je zřejmé, že její hodnoty musí být celočíselné.

Kromě obecných celočíselných podmínek se mohou v úlohách lineárního programování vyskytovat podmínky, které požadují, aby proměnné nabývaly pouze hodnot 0 nebo 1. Takové proměnné se označují jako bivalentní proměnné. Úlohy celočíselného programování, které obsahují pouze bivalentní proměnné, se nazývají úlohami bivalentního programování.

Úlohy celočíselného programování se dělí buď na úlohy s obecnými podmínkami celočíselnosti a na bivalentní úlohy, nebo z jiného hlediska na úlohy ryze celočíselné (kdy podmínky celočíselnosti musí splňovat všechny proměnné obsažené v modelu) a na úlohy smíšeně celočíselné (takové úlohy obsahují jednak proměnné vázané podmínkou celočíselnosti, a jednak proměnné spojité).

Pro řešení celočíselných úloh lineárního programování nelze použít standardní simplexovou metodu vzhledem k tomu, že ta poskytuje obecné řešení neceločíselné.

Proto byly navrženy speciální algoritmy pro řešení těchto úloh.

Tyto algoritmy dělíme do několika skupin podle jejich charakteru:

1. **Metody řezných (sečných) nadrovin** jsou vhodné pro řešení ryze i smíšeně celočíselných úloh, nehodí se však pro řešení úloh bivalentních. Nejdříve je optimální řešení úlohy získáno běžnou simplexovou metodou bez zahrnutí podmínek celočíselnosti. V další fázi výpočtu je přidáno k modelu dodatečné omezení, které „odřízne“ podmnožinu z množiny přípustných řešení úlohy. Tato podmnožina však

nesmí obsahovat žádné celočíselné přípustné řešení dané úlohy. Pro takto zúženou množinu je opět vypočteno optimální řešení. Po konečném počtu kroků vede tento postup k získání optimálního řešení celočíselné úlohy.

2. **Kombinatorické metody** vycházejí z optimálního řešení úlohy lineárního programování získaného pomocí simplexové metody. Množina přípustných řešení se postupně „prohledává“, přičemž se vypouští podmnožiny, které nadále nejsou efektivní. Nejznámější jsou různé varianty metody „větví a mezí“.
3. **Metody dekompoziční** řeší úlohy, ve kterých jsou podmínky celočíselnosti kladeny jen na některé proměnné (smíšeně celočíselné úlohy), rozkladem na dvě části – s podmínkami celočíselnosti a bez podmínek celočíselnosti.
4. **Metody grupové** řeší úlohy lineárního programování s podmínkami celočíselnosti na grupách.
5. **Heuristické metody** řeší obzvláště obtížné úlohy ryze i smíšeně celočíselné přibližnými postupy, které dávají relativně dobré řešení, ale nezaručují jeho optimalitu (např. přiřazovací problém, problém obchodního cestujícího atd.)

Kromě těchto algoritmů či modifikací simplexové metody jsou samozřejmě k dispozici různé programové systémy pro řešení úloh jak lineárního, tak i nelineárního programování. Jako zástupce můžeme uvést doplněk Řešitel tabulkového kalkulátoru MS Excel, MPSX, XA [7], LINDO a systém na podporu modelování LINGO.

2.5 Řešení úloh lineárního programování v MS EXCEL

V následujícím textu stručně popíšeme pouze základní úkony nutné k vyřešení úlohy LP v MS Excel. Detailnější informace a popisy dalších možností řešení optimalizačních úloh v Excelu. [8]

Nejprve musíme do listu zadat hodnoty neřiditelných veličin, tj. prvky matice **A** a vektorů **b** a **c** (vektor **b** je vhodné zapsat jako sloupec, kdežto vektor **c** jako řádek) s případným popisem (význam, jednotky).

Dále je nutné vyhradit buňky pro vektor rozhodovacích proměnných (opět jako řádek). Tyto buňky nemusíme vyplňovat (pak jsou tam implicitně hodnoty 0) nebo je můžeme vyplnit nějakými nenulovými hodnotami (např. hodnotami 1). Druhá možnost je vhodnější,

protože tyto buňky pak vstupují spolu s buňkami neřiditelných veličin do nějakých vzorců a při použití nenulových hodnot máme možnost kontroly těchto vzorců.

Ještě je třeba vyhradit buňku pro výsledkovou veličinu *z*. Tato veličina je obecně dána vzorcem

$$z = \sum_{j=1}^{n} c_j x_j$$

a tento vzorec také musíme zadat do příslušné buňky. Uvedený výraz vlastně představuje skalární součin vektorů **c** a **x** a k tomu je vhodné využít funkci SOUČIN.SKALÁRNÍ, která se nachází v repertoáru matematických funkcí Excelu.

Nyní máme na ploše listu zachyceny všechny veličiny matematického modelu a mohli bychom tedy tento model přepsat do Excelu. Abychom si však vytvoření modelu v Excelu usnadnili, vyhradíme na ploše další buňky pro hodnoty levých stran vlastních omezení, které jsou dány výrazy

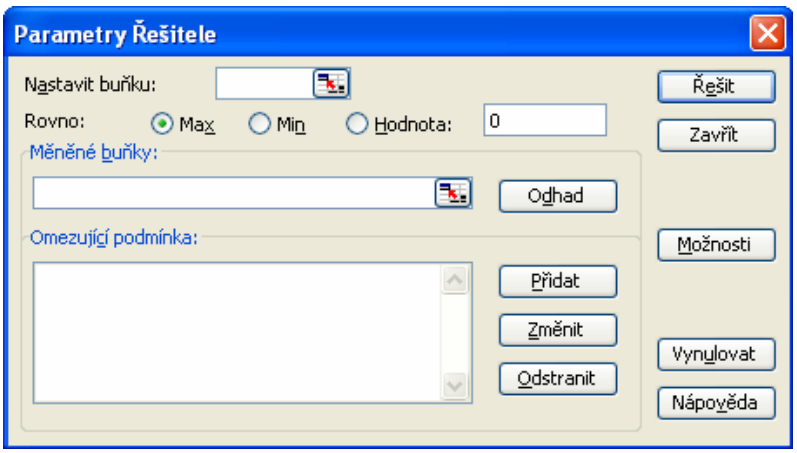
$$\sum_{j=1}^{n} a_{ij} x_j , \qquad i = 1, \ldots, m$$

Tyto výrazy zadáme do příslušných buněk opět pomocí funkce SOUČIN.SKALÁRNÍ, protože *i*-tý výraz je vlastně skalární součin *i*-tého řádku matice **A** a vektoru **x**.

Když máme připraveny všechny potřebné veličiny, zavoláme nástroj zvaný **Řešitel** (**Solver**), který se nachází v nabídce **Nástroje**. Pokud jej v této nabídce nenajdeme, musíme jej zaškrtnout v podnabídce **Doplňky**. Jestliže tam není, pak to znamená, že nemáme k dispozici kompletní verzi Excelu.

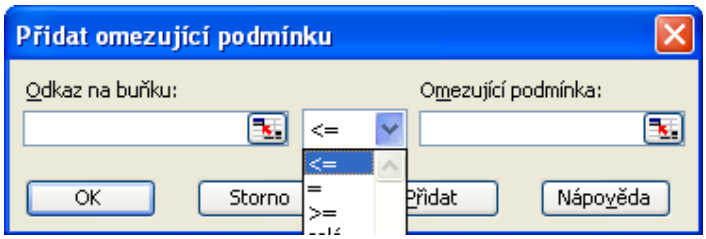
Po zavolání Řešitele se nám objeví okno **Parametry Řešitele**, ve kterém musíme zadat:

- která buňka reprezentuje účelovou funkci (pole **Nastavit buňku**),
- zda chceme dosáhnout jejího maxima, minima nebo nějaké zadané hodnoty,
- které buňky odpovídají rozhodovacím proměnným (pole **Měněné buňky**),
- omezující podmínky.



Obr. 2.5 Parametry Řešitele

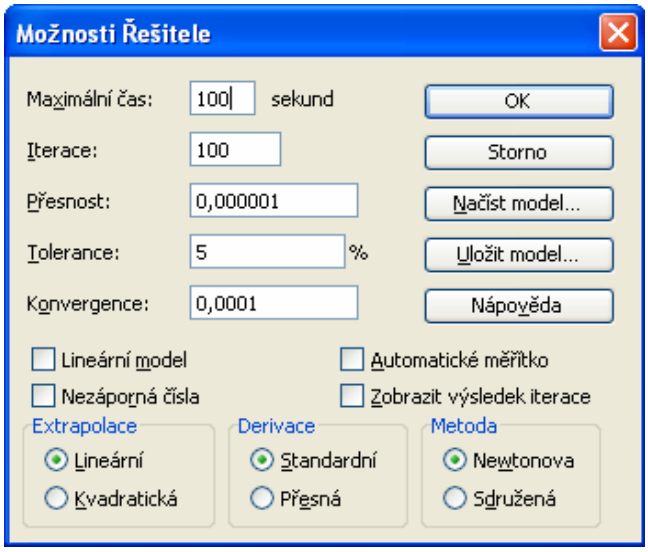
Omezující podmínky se zadávají pomocí tlačítka **Přidat**. Po kliknutí na toto tlačítko se otevře okno **Přidat omezující podmínku** se třemi poli. Do pole **Odkaz na buňku** se musí zadat adresa buňky odpovídající levé straně omezující podmínky (tuto adresu nemusíme zadávat textově, stačí na ni kliknout myší). V prostředním poli zvolíme příslušný symbol nerovnosti nebo rovnice a do pole **Omezující podmínka** zadáme odkaz na buňku odpovídající pravé straně omezující podmínky. Do tohoto pole můžeme zadat i konkrétní hodnotu, ale to není příliš vhodné, protože to ztěžuje případné budoucí úpravy modelu. Chceme-li přidávat další podmínku, klikneme na tlačítko **Přidat** (aktuální podmínka se uloží a objeví se okno pro přidání další podmínky). Chceme-li zadávání omezujících podmínek ukončit, klikneme na tlačítko **OK**, čímž se vrátíme do okna **Parametry Řešitele**. Pomocí jednoho okna **Přidat omezující podmínku** můžeme zadat celou skupinu omezujících podmínek, které mají stejný symbol nerovnosti nebo rovnice (buňky odpovídající levým a pravým stranám těchto podmínek musejí ovšem na ploše listu vytvářet souvislé skupiny). Do pole **Odkaz na buňku** zadáme skupinu buněk obsahujících levé strany omezujících podmínek (zadání se provede přejetím myši přes tyto buňky) a do pole **Omezující podmínka** zadáme buňky reprezentující příslušné pravé strany.



Obr. 2.6 Omezující podmínky

Podmínky nezápornosti můžeme zadat rovněž pomocí okna **Přidat omezující podmínku** s tím, že do pole **Omezující podmínka** zadáme hodnotu 0. Pokud jsou ale podmínky nezápornosti vztaženy na všechny rozhodovací proměnné, je výhodnější tyto podmínky zadat v okně **Možnosti Řešitele**.

Než dáme pokyn k řešení, je třeba zkontrolovat a případně upravit nastavení možností řešitele. Po kliknutí na příslušné tlačítko v okně **Parametry Řešitele** se objeví okno **Možnosti Řešitele**, ve kterém je možno nastavit různé parametry týkající se řešení lineárních, nelineárních a celočíselných úloh (některé volby jsou zde implicitně nastaveny a můžeme je v případě potřeby upravit). Zaměřme se pouze na základní volby týkající se lineárních úloh.

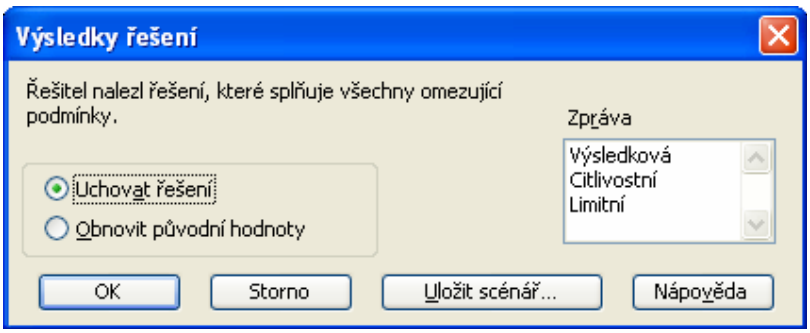


Obr. 2.7 Možnosti Řešitele

Pomocí polí **Maximální čas** a **Iterace** můžeme omezit dobu trvání výpočtu. Jestliže do vyčerpání zadané doby nebo stanoveného počtu iterací není nalezeno optimální řešení (nebo se nezjistí, že úloha žádné přípustné či optimální řešení nemá), tak výpočet skončí. Údaj v poli **Přesnost** určuje, s jakou přesností se bude omezující podmínka považovat za splněnou. Pole **Tolerance** se týká úloh s podmínkami celočíselnosti proměnných a pole **Konvergence** se používá u nelineárních problémů. S nelineárními problémy souvisejí také volby **Extrapolace**, **Derivace** a **Metody**. Jestliže řešíme úlohu LP, musíme zatrhnout volbu **Lineární model**. Pokud bychom to neudělali, úloha by se neřešila simplexovou metodou, ale metodou pro řešení nelineárních úloh a nezískali bychom tak podrobné údaje pro citlivostní analýzu, jako v případě použití simplexové metody. Zaškrtnutí políčka **Nezáporná čísla** způsobí, že bude nastavena dolní mez 0 pro všechny rozhodovací proměnné (měnitelné buňky), pro které jsme hodnotu dolní meze nezadali při zadávání omezujících podmínek.

Zaškrtnutím políčka **Automatické měřítko** se aktivuje automatická úprava měřítka v případech, kdy se výrazně liší velikosti vstupních údajů. Jestliže zaškrtneme políčko **Zobrazit výsledek iterace**, výpočet se po každé iteraci přeruší. Po kliknutí tlačítko **Uložit model** se zobrazí dialogové okno, ve kterém můžeme zadat umístění, do nějž můžeme model uložit. Toto tlačítko se používá pouze v případech, kdy s listem chceme uložit více než jeden model. Jestliže pracujeme pouze s jedním modelem, pak je tento model uložen automaticky. Kliknutí na tlačítko **Načíst model** zobrazí dialogové okno, ve kterém můžeme zadat odkaz na model, který chceme načíst.

Kliknutím tlačítko **OK** potvrdíme nastavení možností řešitele, čímž se vrátíme do okna **Parametry Řešitele** . Výpočet pak spustíme kliknutím tlačítko **Řešit**. Po ukončení procesu řešení se objeví okno **Výsledky řešení**. V případě, že bylo nalezeno optimální řešení, objeví se zde text „Řešitel nalezl řešení, které splňuje všechny omezující podmínky“ a vypočtené hodnoty rozhodovacích proměnných a účelové funkce se objeví v příslušných buňkách na ploše listu. Tyto hodnoty máme možnost uchovat nebo se vrátit k původním hodnotám a dále máme možnost požádat o vygenerování výsledkové, citlivostní a limitní zprávy. Jestliže úloha nemá konečné optimální řešení, v okně **Výsledky řešení** se objeví text „Hodnota nastavované buňky není konvergentní“. Neexistence přípustného řešení je signalizována textem „Řešitel nenalezl vhodné řešení“.



Obr. 2.8 Výsledky řešení

3 NÁVRH ŘEŠENÍ

3.1 Zadání

Přizpusobit optimalizační systém dělení materiálu pro podmínky výroby v KS-Kuřim, t.zn. vytvořit nejvhodnější kombinaci vyráběných šroubů včetně určení délky výchozího materiálu.

Dáno:

1. Různé průměry řezaných tyčí (výrobní plán)
2. Různé délky základního materiálu (6500mm a 8500mm)
3. Při jednom nastavení stroje lze vyrábět šrouby různého stoupání (t; mm), ale pouze jednoho průměru kuličky D_w (profil závitu)
4. Podstatná pro vyhodnocení odpadu je cena zbytku materiálu (různá cena tyče/m při různých délkách výchozího materiálu)

Tab.3.1 Výrobní plán

Ø80	**L celk.**	**ks**	**D_w**	**L řezaná**
ZÁŘÍ	4520	8	7,144	4770
	3150	12	7,144	3400
	3150	12	7,144	3400
	5140	2	10,319	5490
	4140	1	10,319	4490
ŘÍJEN	3150	12	7,144	3400
	4517	10	7,144	4600
	2589	2	6,35	2770
	3713	1	12,7	3728
	3245	1	12,7	3565
	5482	1	12,7	5575
LISTOPAD	5017	6	7,144	5171
	5017	6	7,144	5171
	4520	8	7,144	4770
	4517	12	7,144	4600
	2302	2	7,144	2574
	4520	8	7,144	4770
	4515	1	7,938	4610
	4070	4	12,7	4530
PROSINEC	5120	2	7,144	5522
	4970	1	7,938	5503
	4312	3	12,7	4781
	4277	2	12,7	4454

	2575	2	12,7	2889
	4421	1	12,7	4874
	4452	1	12,7	4846
	3813	2	12,7	4265
	3813	1	12,7	4265
	3813	1	12,7	4265
LEDEN	3150	12	7,144	3400
	4517	10	7,144	4675
	5129	3	7,144	5522
	2050	2	7,144	2590
	3170	1	7,144	3560
	2235	1	7,144	2710
	1700	1	7,144	2225
	4857	2	7,144	5072
	2030	2	7,938	2510
	2340	1	7,938	2825
	4483	2	7,938	4780
	4812	3	12,7	5092
	3520	2	12,7	3990
ÚNOR	5017	6	7,144	5171
	4017	6	7,144	4347
	3150	12	7,144	3400
	4520	8	7,144	4770
	1918	1	7,144	2350
	2030	1	7,938	2510
	3113	1	7,938	3410
	4452	1	12,7	4660
	6196	1	12,7	6580
	3520	2	12,7	3990
	4590	2	12,7	5060

Tab.3.2 Ceny

Průměr	**Délka**	**EUR/m**	**Celá týč €**
Ø 80	6500	86,81	564,3
Ø 80	8500	118,35	1006

Máme úlohu zpracování řezného plánu s cílem snížení ekonomických nákladů. Ekonomický problém, kterým se úloha o optimálním dělení materiálu zabývá, spočívá v co nejefektivnějším rozdělení daného původního materiálu na zadané části v rámci daných podmínek a omezení. Nejčastěji se jedná o prkna nebo kovové trubky (dále potom provazy, tyče atd.). Existuje mnoho způsobů a kombinací, jak docílit daných požadavků na počet nebo rozměry jednotlivých kusů. Úkolem při řešení těchto úloh je nalézt nejlepší kombinaci

způsobů řezání při splnění zadaných omezení. Jako hlavní požadavek pro optimalizaci se v klasických úlohách o dělení materiálu udává minimalizace vzniklého odpadu nebo minimalizace počtu rozřezaných prken, příp. tyčí. V našem případě hlavní požadavek pro optimalizaci je minimalizace nákladů při nakupování materiálu.

Požadavek na celočíselnost řešení vyplývá z ekonomické interpretace proměnných. Těmi jsou v těchto úlohách různé způsoby řezání. Pro sestavení matematického modelu je třeba nejprve vytvořit tzv. řezný plán (řezné schéma). Do přehledné tabulky se zapíší do sloupců požadované rozměry a do řádků možné způsoby dělení původního materiálu. Jako další řádek tabulky se může přidat informace o zbytku z řezání, odpadu, kdy největší možný odpad každého způsobu řezání je menší než nejmenší požadovaný kus.

3.2 Minimalizace odpadů

3.2.1 Vypracování řezných plánů

Předpokládejme, že existuje n přípustných řezných plánů. Zaveďme následující označení:

a_{ij} – počet přířezů i-tého druhu získaných rozřezáním výchozí tyče podle j-tého řezného plánu

b_i – požadovaný počet přířezů i-tého

c_i – délka odpadu vzniklého rozřezáním výchozí tyče podle j-tého řezného plánu

x_j – počet výchozích tyčí rozřezáných podle j-tého řezného plánu

Omezující podmínky:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i=1,2,\ldots,m \tag{3.1}$$

$$x_j \geq 0,\ x_j \in Z, \qquad j=1,2,\ldots,n \tag{3.2}$$

kde Z označuje množinu celých čísel.

Učelová funkce pro tento problém může být zkonstruována dvěma způsoby:

1) Minimalizace celkové množství odpadu

$$z = \sum_{j=1}^{n} c_j x_j \rightarrow \min \tag{3.3}$$

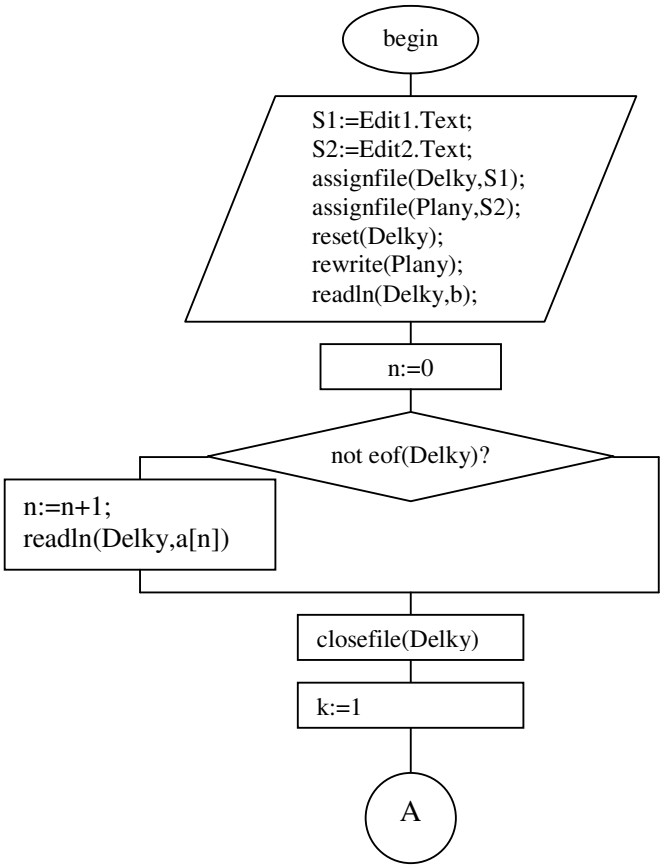
2) Minimalizace celkového počtu rozřezaných výchozích tyčí

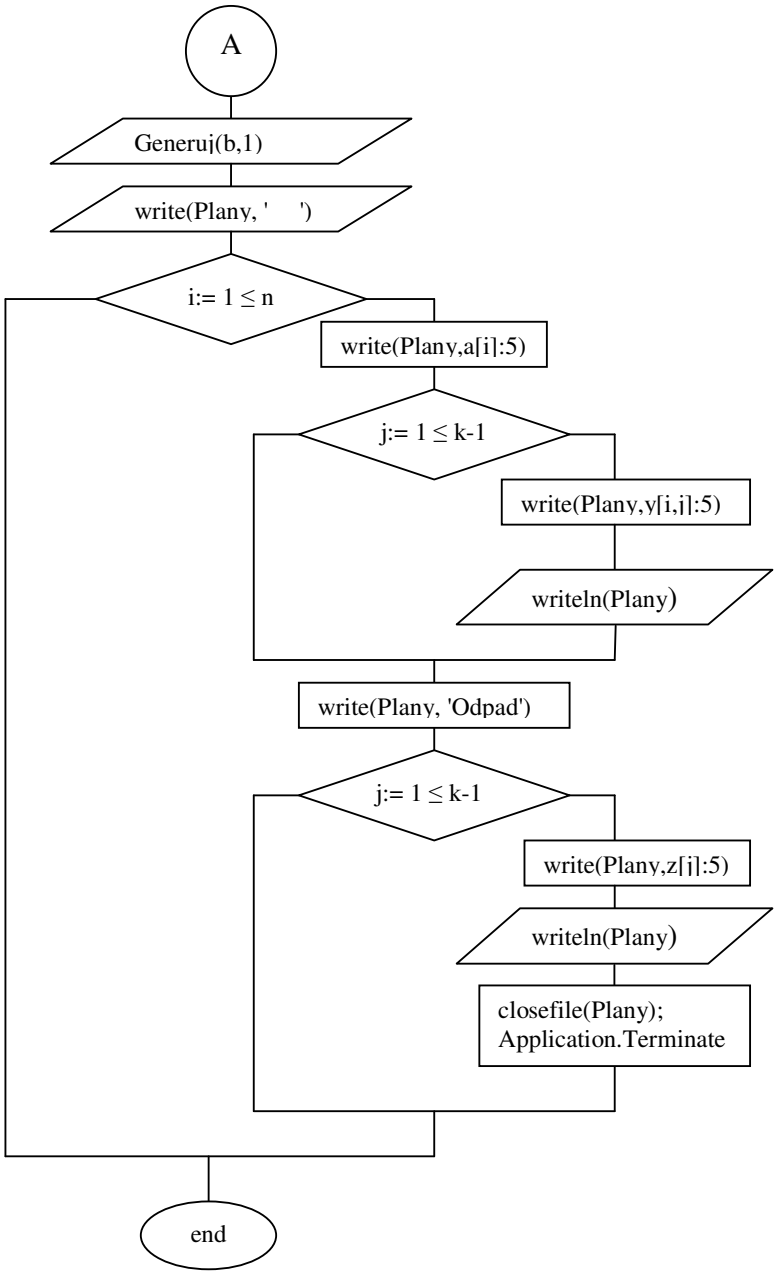
$$z = \sum_{j=1}^{n} x_j \rightarrow \min \qquad (3.4)$$

Než začneme sestavovat model, musíme nejdříve zjistit všechny možné přípustné způsoby rozřezání výchozí tyče na požadované délky.

To je možno udělat jednoduchým způsobem výpočtu ručně, ale rychlejším a přesnějším bude způsob výpočtu řezného plánu pomocí vypracování v prostředí Delphi. [9]

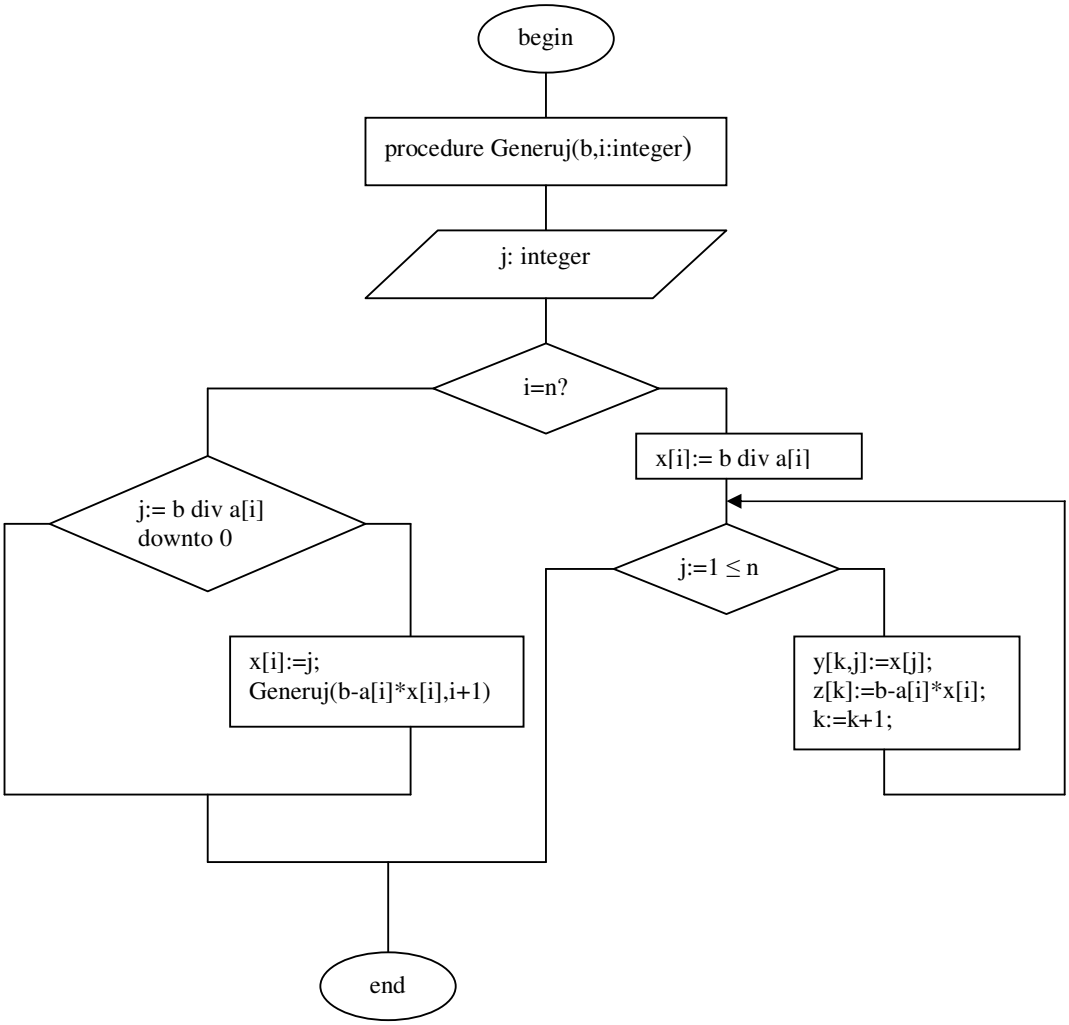
Text programu se píše programovacím jazykem Delphi, který je následníkem jazyka Pascal. Algoritmus programu tedy vypadá takto:





Obr. 3.1 Algoritmus programu na vypracování řezných plánů

Pro vypracování nejrůznějších možných řezných plánů byla napsána speciální procedura, která má název “Generuj”. Umožňuje opakovat generaci řezných plánů s různými údaji (viz obrázek 3.2).



Obr. 3.2 Algoritmus procedury "Generuj"

S použitím uvedených algoritmů byl sestaven takový program.

```
Unit Unit2;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  Tform1 = class(Tform)
    Edit1: Tedit;
    Edit2: Tedit;
```

Obr. 3.3 Program výpočtu řezných plánů

```
    Label1: Tlabel;
      Label2: Tlabel;
      Button1: Tbutton;
      procedure Button1Click(Sender: Tobject);
    private
      { Private declarations }
    public
      { Public declarations }
    end;
var
  Form1: Tform1;
implementation
{$R *.dfm}
procedure Tform1.Button1Click(Sender: Tobject);
   var  i,j,k,b,n:integer;
        S1,S2:string;
        Delky, Plany: TextFile;
        a,x: array [1..50] of integer;
        y: array [1..50,1..500] of integer;
        z: array [1..500] of integer;
   procedure Generuj(b,i:integer);
        var j: integer;
        begin
           if i=n then
              begin
                 x[i]:= b div a[i];
                 for j:=1 to n do
                    y[k,j]:=x[j];
                 z[k]:=b-a[i]*x[i];
                 k:=k+1;
              end
           else
              for j:= b div a[i] downto 0 do
                 begin
                    x[i]:=j;
                    Generuj(b-a[i]*x[i],i+1);
                 end;
        end;
begin
    S1:=Edit1.Text;
    S2:=Edit2.Text;
    assignfile(Delky,S1);
    assignfile(Plany,S2);
    reset(Delky);
    rewrite(Plany);
    readln(Delky,b);
   n:=0;
```

Obr. 3.3 Program výpočtu řezných plánů (pokračování)

```
        while not eof(Delky) do
            begin
              n:=n+1;
              readln(Delky,a[n])
            end;
          closefile(Delky);
          k:=1;
          Generuj(b,1);
          write(Plany, ,     ,);
          for j:= 1 to k-1 do write(Plany, j:5);
          writeln(Plany);
          for i:= 1 to n do
            begin
              write(Plany,a[i]:5);
              for j:= 1 to k-1 do
                 write(Plany,y[i,j]:5);
              writeln(Plany);
            end;
          write(Plany, ,Odpad‘);
          for j:= 1 to k-1 do write(Plany,z[j]:5);
          writeln(Plany);
          closefile(Plany);
          Application.Terminate;
        end;
end.
```

Obr. 3.3 Program výpočtu řezných plánů (pokračování)

Uživatelský interface je dialogové okno, kam uživatel zadává vstupní údaje. To bude soubor dat, který uživatel uchovává ve formátu «.txt». Prvním údajem je délka výchozí tyče a pak následují požadované řezné délky. Každý údaj je na zvláštním řádku. Data budeme vybírat z určitého výrobního plánu za jeden měsíc (například listopad), viz tabulku 3.1.

1. <Delky.listopad 6500>

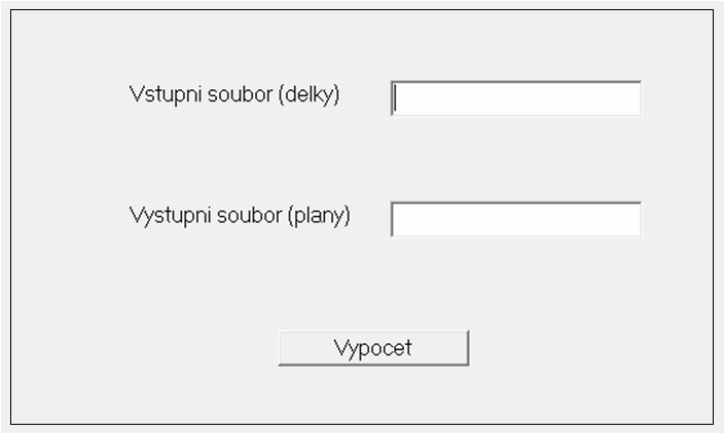
```
6500
5171
4770
4600
2574
```

2. <Delky.listopad 8500>

```
8500
5171
4770
4600
2574
```

Potom výše uvedené soubory vkládame do interface <Vstupni soubor (delky)>. V tomto poli píšeme <Delky.listopad 6500.txt>.

V poli <Vystupni soubor (plany)> musíme uvést název souboru (.txt), kam bude uložena výsledná data. Například: <Plany.listopad 6500.txt>



Obr. 3.4 Uživatelský interface

Výsledk bude uložen do výše uvedeného souboru. Uvidíme tam tabulku, kde první sloupec obsahuje požadované délky a další sloupce pak jednotlivé řezné plány označené číslem plánu. V posledním řádku tabulky jsou odpovídající odpady.

1. <Plany.listopad 6500>

```
         1    2    3    4
 5171    1    0    0    0
 4770    0    1    0    0
 4600    0    0    1    0
 2574    0    0    0    2
Odpad 1329 1730 1900 1352
```

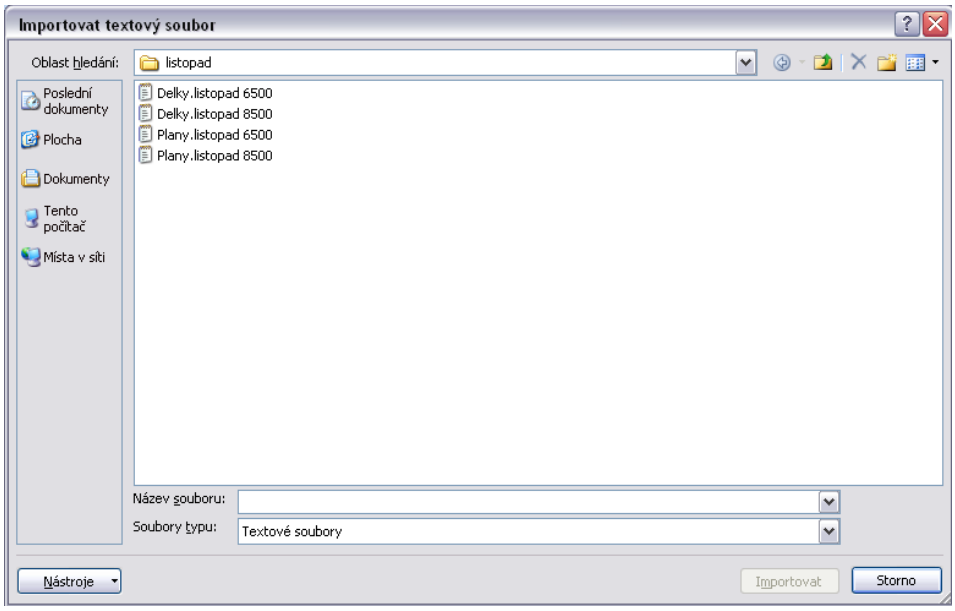
2. <Plany.listopad 8500>

```
         1    2    3    4
 5171    1    0    0    0
 4770    0    1    0    0
 4600    0    0    1    0
 2574    1    1    1    3
Odpad  755 1156 1326  778
```

3.2.2 Optimalizace v MS Excel

Získané údaje je nutno importovat do Excelu a také svázat s tabulkou optimalizace (viz bod 2.5) pro získání výsledků optimálního řezání.

Když máme připraveny všechny potřebné hodnoty, zavoláme složku zvanou **Z textu**, která se nachází v nabídce **Data**.

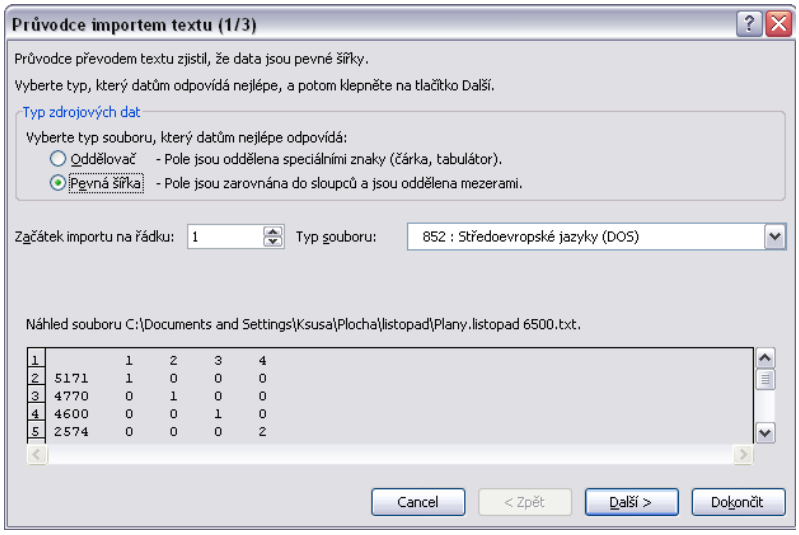


Obr. 3.5 Importování textového souboru

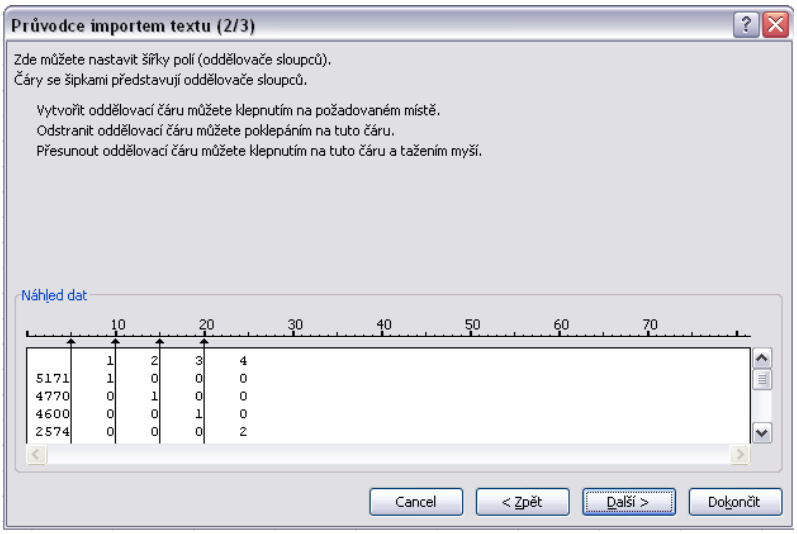
V **Oblasti hledání** najdeme umístění souborů dat na počítači a potom vybereme ten který chceme importovat do Excelu. V našem případě to bude <Plany.listopad 6500> a <Plany.listopad 8500>.

Omezující podmínky se zadávají pomocí **Průvodce importem textu**. V něm musíme nastavit Typ zdrojových dat, Začátek importu na řádku, Typ souboru, Formát dat ve spoupcích a Umíštění dat.

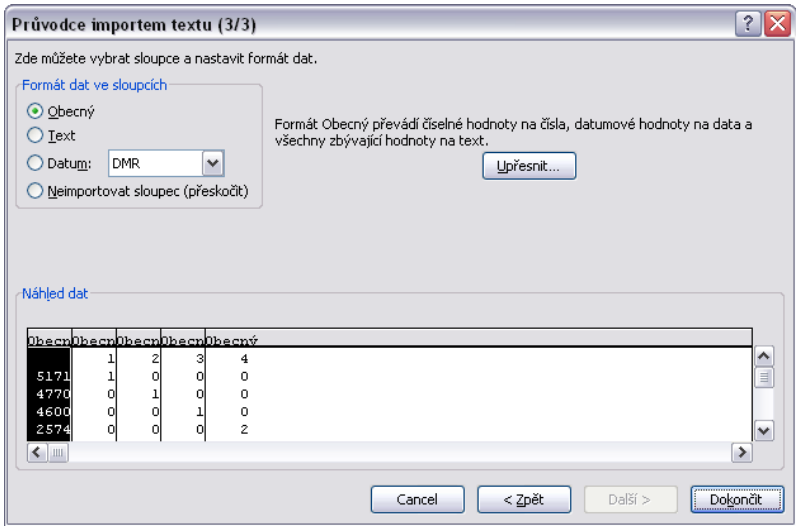
a)



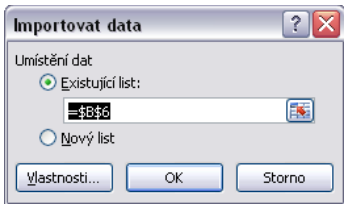
b)



c)



d)



Obr. 3.6 a)-d) Průvodce importem textu

Soubory, vytvořené takovým způsobem a umístěné do tabulky optimalizace, budou vypadat takto:

Požadované typy tyčí		Nakupovaná délka tyče								Počet tyčí k výrobě	počet
		6500 mm				8500 mm					
		Řezný plán č.									
		1	2	3	4	5	6	7	8		
Tyč 1	5171 mm	1	0	0	0	1	0	0	0	12	12
Tyč 2	4770 mm	0	1	0	0	0	1	0	0	16	16
Tyč 3	4600 mm	0	0	1	0	0	0	1	0	12	12
Tyč 4	2574 mm	0	0	0	2	1	1	1	3	2	2
Odpad	[mm]	1329	1730	1900	1352	755	1156	1326	778	Celkem	65280

Řezné plány	x1	x2	x3	x4	x5	x6	x7	x8
Použití řez. plánů	12	16	10	0	0	0	2	0

Obr. 3.7 Tabulka optimalizace

Výsledná zpráva obsahuje původní a vypočtené hodnoty účelové funkce (nastavovaná buňka) a rozhodovacích proměnných (změněné buňky). Dále podává informace o omezujících podmínkách. Jsou uvedeny výsledné hodnoty levých stran omezujících podmínek, vzorce popisující tyto podmínky, informace o stavu splnění těchto podmínek a odchylky hodnot pravých stran od výsledných hodnot levých stran. Stav **Platí** znamená „platí jako rovnice“, stav **Neplatí** znamená „neplatí jako rovnice“ (tj. podmínka je splněna jako ostrá nerovnost). Vidíme, že všechna omezení jsou splněna jako nerovnost, tj. výrobní kapacity jsou využity se zbytkem.

Nastavovaná buňka (Min)

Buňka	Název	Původní hodnota	Konečná hodnota
$L$11	Celkem počet	0	65280

Měněné buňky

Buňka	Název	Původní hodnota	Konečná hodnota
$C$14	Použití řez. plánů x1	0	12
$D$14	Použití řez. plánů x2	0	16
$E$14	Použití řez. plánů x3	0	10
$F$14	Použití řez. plánů x4	0	0
$G$14	Použití řez. plánů x5	0	0
$H$14	Použití řez. plánů x6	0	0
$I$14	Použití řez. plánů x7	0	2
$J$14	Použití řez. plánů x8	0	0

Omezující podmínky

Buňka	Název	Hodnota buňky	Vzorec	Stav	Odchylka
$L$7	5171 mm počet	12	$L$7=$K$7	Neplatí	0
$L$8	4770 mm počet	16	$L$8=$K$8	Neplatí	0
$L$9	4600 mm počet	12	$L$9=$K$9	Neplatí	0
$L$10	2574 mm počet	2	$L$10=$K$10	Neplatí	0

Obr. 3.8 Výsledná zpráva

Citlivostní zpráva poskytuje informace o citlivosti řešení na malé změny v koeficientech účelové funkce a v pravých stranách omezujících podmínek.

V části **Změněné buňky** jsou uvedeny konečné hodnoty rozhodovacích proměnných, snížené náklady, cílové koeficienty (to jsou koeficienty účelové funkce) a povolené nárůsty a poklesy těchto koeficientů. Hodnota sníženého nákladu říká, o kolik by se změnila hodnota účelové funkce, pokud by se příslušná proměnná stala bázickou.

V části **Omezující podmínky** se nacházejí konečné hodnoty levých stran omezujících podmínek, stínové ceny (to jsou optimální hodnoty duálních proměnných), hodnoty pravých stran omezujících podmínek a povolené nárůsty a poklesy těchto pravých stran. Připomeňme, že stínová cena uvádí, o kolik by se změnila optimální hodnota účelové funkce, kdybychom hodnotu odpovídající pravé strany zvýšili o jednotku (pokud ovšem by se tato změna nacházela v rámci povoleného nárůstu).

Povolený nárůst a pokles cílového koeficientu nebo pravé strany udávají, v jakém rozsahu se může příslušný údaj změnit při neměnnosti všech ostatních údajů, aniž by došlo ke změně optimální báze. Tento nárůst a pokles tedy společně určují interval stability cílového koeficientu nebo pravé strany. Meze tohoto intervalu určíme tak, že pokles odečteme od zadané hodnoty příslušného údaje a nárůst k této hodnotě přičteme. Přitom hodnota 1E+30 reprezentuje nekonečno.

Měněné buňky

Buňka	Název	Konečná hodnota	Snížené náklady	Cílový koeficient	Povolený nárůst	Povolený pokles
$C$14	Použití řez. plánů x1	12	0	1329	0	1E+30
$D$14	Použití řez. plánů x2	16	0	1730	0	1E+30
$E$14	Použití řez. plánů x3	10	0	1900	1E+30	0
$F$14	Použití řez. plánů x4	0	2500	1352	1E+30	2500
$G$14	Použití řez. plánů x5	0	0	755	1E+30	0
$H$14	Použití řez. plánů x6	0	0	1156	1E+30	0
$I$14	Použití řez. plánů x7	2	0	1326	0	1E+30
$J$14	Použití řez. plánů x8	0	2500	778	1E+30	2500

Omezující podmínky

Buňka	Název	Konečná hodnota	Stínová cena	Omezující podmínka Pravá strana	Povolený nárůst	Povolený pokles
$L$7	5171 mm počet	12	1329	12	1E+30	12
$L$8	4770 mm počet	16	1730	16	1E+30	16
$L$9	4600 mm počet	12	1900	12	1E+30	10
$L$10	2574 mm počet	2	-574	2	10	2

Obr. 3.9 Citlivostní zpráva

Limitní zpráva zobrazuje cílovou buňku (buňku s hodnotou účelové funkce) a změněné buňky (buňky s hodnotami rozhodovacích proměnných) spolu s výslednými

hodnotami těchto buněk. Dále tato zpráva obsahuje dolní a horní meze a cílové hodnoty. Dolní mez představuje nejnižší hodnotu měnitelné buňky při konstantních hodnotách ostatních měnitelných buněk a splněných omezujících podmínkách, kdežto horní mez představuje nejvyšší hodnotu této buňky. Cílový výsledek reprezentuje odpovídající hodnotu účelové funkce.

Buňka	Název cílové buňky	Hodnota
$L$11	Celkem počet	65280

Buňka	Název měněné buňky	Hodnota	Dolní mez	Cílový výsledek	Horní mez	Cílový výsledek
$C$14	Použití řez. plánů x1	12	12	65280	12	65280
$D$14	Použití řez. plánů x2	16	16	65280	16	65280
$E$14	Použití řez. plánů x3	10	10	65280	10	65280
$F$14	Použití řez. plánů x4	0	0	65280	0	65280
$G$14	Použití řez. plánů x5	0	0	65280	0	65280
$H$14	Použití řez. plánů x6	0	0	65280	0	65280
$I$14	Použití řez. plánů x7	2	2	65280	2	65280
$J$14	Použití řez. plánů x8	0	0	65280	0	65280

Obr. 3.10 Limitní zpráva

Podle výsledků se dá usoudit, že z hlediska šetření používaného materiálu a rovněž množství odpadů se doporučuje vyrábět požadované tyče podle plánu:

Tab.3.3 Výsledky minimalizace vzniklého odpadu

Tyč	Počet	
	6500 mm	8500 mm
5171 mm	12	-
4770 mm	16	-
4600 mm	10	2
2574 mm	-	2
Celkem pro nákup	38	2
Náklady celkem	**23455,4€**	

3.3 Použití programu «Optimizer»

3.3.1 Popis programu

Když obdržíme výsledky optimalizace z hlediska minimalizace odpadů, je třeba se utvrdit v tom, že to je opravdu nejúspornější varianta. Prověříme výsledky, porovnáme je s výsledky zvláštního programu «Optimizer» pro minimalizace nákladů na nákup materiálu.

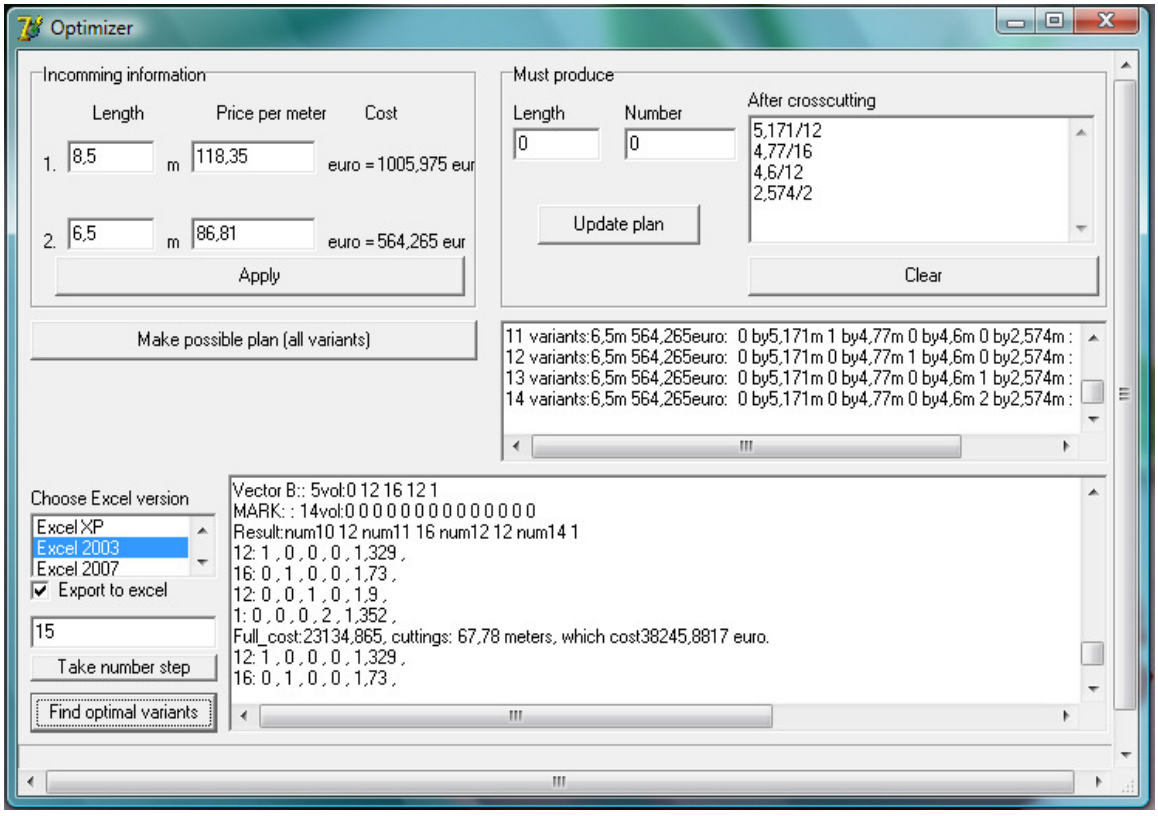
Program je psaný v programovacím jazyku Delphi. Pro lepší názornost výsledky programu lze integrovat do MS Excel.

Algoritmus programu je uveden v příloze A.

Text programu je uveden v příloze B.

3.3.2 Příručka k používání programu «Optimizer»

a) Pro spuštění programu najdeme soubor <Project1.exe> a otevřeme ho. Po zavedení programu uvidíme uživatelský interface, viz obrázek 3.11.



Obr. 3.11 Uživatelský interface programu «Optimizer»

Dále je třeba načíst data:

b) V poli <Incomming information>, což znamená <Vkládaná data> uvádíme:

- <Lenght> základní délku (8,5 m a 6,5 m)
- <Price per meter> cenu za metr.

c) V poli <Must produce>, zadáme výrobní plan a uvádíme:

- <Lenght> požadovanou délku (například 4,2)
- <Number> požadovaný počet.

Požadovaných délek může být samozřejmě několik - podle výrobního plánu. Když stiskneme tlačítko <Update plan> v okně <After crosscutting>, uvidíme celý výrobní plán. Tlačítko <Clear> vymaže toto pole.

d) Vytvořit veškeré možné plány pomůže tlačítko <Make possible plan (all variants)>.

e) Vytvořit optimální variantu pomůže tlačítko <Find optimal variants>.

f) Optimalní varianta je uvedena v okně <Find optimal variants> ve formě řádku <Result:num10 12 num11 16 num12 12 num14 1 > a rovněž ve formě zvláštního řádku dole v uživatelském interface. Což znamená, že řezný plán č.10 <num10> má být použit 12 krát, plán č.11 <num11> má být použit 16 krát a t.d.

Pro lepší názornost výsledků v programu existuje možnost veškeré plány integrovat do MS Excel. Proto je nutno zaškrtnout políčko <Export do excel> a také v seznamu <Choose Excel version> nastavit používanou verzi MS Excel.

Dostaneme následující tabulku s výsledky, viz obrázek 3.12.

Incomming Informations						
Length,m	Price per meter,euro	Cost,euro				
8,5	118,35	1005,975				
6,5	86,81	564,265				
Must produce						
Length,m	Number,piece					
5,171	12					
4,77	16					
4,6	12					
2,574	2					
Variants of cuttings						
Number of variants	from,meter	5,171m	4,77m	4,6m	2,574m	Cuttings
1	1005,975	1	0	0	0	3,329
2	1005,975	0	1	0	0	3,73
3	1005,975	0	0	1	0	3,9
4	1005,975	0	0	0	1	5,926
5	1005,975	1	0	0	1	0,755
6	1005,975	0	1	0	1	1,156
7	1005,975	0	0	1	1	1,326
8	1005,975	0	0	0	2	3,352
9	1005,975	0	0	0	3	0,778
10	564,265	1	0	0	0	1,329
11	564,265	0	1	0	0	1,73
12	564,265	0	0	1	0	1,9
13	564,265	0	0	0	1	3,926
14	564,265	0	0	0	2	1,352
Optimal variant Variants of cuttings						
Number of variants	Piece					
10	12					
11	16					
12	12					
14	1					
Cost of variant cuttings, euro	Cuttings,piece					
23134,865	67,78					

Obr. 3.12 Výsledky programu «Optimizer»

g) v tabulce můžeme najít všichna požadovaná data:

- <Variants of cuttings> - veškeré možné řezné plány,
- <Cuttings> - Množství odpadů,
- <Cost of variant cuttings, euro> - celkové nákupní náklady,
- <Cuttings, piece> - celkové množství odpadů.

4 ANALÝZA VÝSLEDKŮ

Pro optimalizaci nákupu materiálu byly aplikovány dvě metody. Pomocí první metody (řešení úloh lineárního programování v MS EXCEL) se podařilo minimalizovat počet odpadů od řezání tyčí. Použití druhé metody umožnilo minimalizovat náklady na nákup materiálu.

Pro názornost zobrazíme výsledky obou metod ve formě tabulky, viz tabulku 4.1

Tab.4.1 Analýza výsledků

Název metody	**Náklady na nákup (€)**	**Odpad**		
		% po optimalizace	Celkový počet (m)	Hodnota (€)
Lineární programování v MS EXCEL	**23455,4**	24,7	65,28	5673,4
Program «Optimizer»	**23134,865**	25,67	67,78	5883,9

Z tabulky je zřejmé, že program «Optimizer» počítá optimální variantu řezání základních tyčí pro snížení nákladů na nákup materiálu, ale počet odpadů je přesto dost velký. Přesto však toto není hlavní faktor volby metody optimalizace. A to z důvodu, že se odpady dále zpracovávají.

ZÁVĚR

Tato diplomová práce řeší problém optimalizace nákupu hutního materiálu pro potřebu společnosti KULIČKOVÉ ŠROUBY KUŘIM,a.s.

Pro realizaci tohoto úkolu byla použita simplexová metoda jako základ řešení úloh lineárního programování o optimálním dělení materiálu.

V teoretické části byla provedena analýza problému, detailně popsána simplexová metoda a matematický model řešení a také základní kroky nutné k vyřešení úlohy LP v MS Excel.

Jádrem této práce je vypracování všech možných přípustných způsobů rozřezání výchozí tyče na požadováné délky a jejich optimální kombinace z hlediska minimalizace nákladů na nákup materiálu. Proto jsou použity dvě efektivní metody: řešitel MS EXCEL pro minimalizace odpadů a speciálně vytvořený program «Optimizer» pro minimalizaci celkových nákladů na nákup materialu. Výsledky obou metod jsou rozebrány a je rozhodnuto o nutnosti použití programu «Optimizer» v konkrétních podmínkách podniku. Tento program má pohodlný uživatelský interface, který umožňuje zadávat požadované údaje a dostávat optimální výsledky bez speciálních znalostí v oblasti programování. Pro zjednodušení práce s programem byla napsána podrobná metodika pro jeho použití.

Zpracováním daného tématu jsem si rozšířila vědomosti o problematice optimalizace. Využití této úlohy lineárního programování v praktickém životě je velmi široké. V oblasti lineárních modelů se může jednat např. o dílnu zpracovávající dřevo, dále pak má své využití např. u dělení lan, tyčí, dělení papíru v rolích, příp. látky.

V příloze uvádím algoritmus a text programu «Optimizer».

SEZNAM POUŽITÝCH ZDROJŮ

[1] KULIČKOVÉ ŠROUBY KUŘIM.: *Katalog výrobků*, 2007. 46 s. V. 2007-01.

[2] BASOVSKIJ, L., PROTASJEV, V.: *Řízení jakosti*. Moskva: INFRA-M, 2003. 212 s. ISBN 5-16-001222-2.

[3] ZÍSKAL, J., BERÁNKOVÁ, M., HOUŠKA, M.: *Lineární programování. I.* Praha: Reprografické studio REF ČSU v Praze, 2006. 70 s. ISBN 80-213-1313-7.

[4] HALÍK, R., HÝBL, J., KODYM, M.: *Ekonomika spotřebního průmyslu*. Praha: MIR, novinářské závody, 1966. 180 s. 06-013-66.

[5] LAUBER, J., JABLONSKÝ, J.: *Programy pro matematické modelování. I.* Praha: VŠE, 1997. 233 s. ISBN 80-7079-296-5.

[6] *Dvoufázová simplexová metoda* [online] (2005). Dostupné z WWW: <http://www1.osu.cz/studium/mopv2/simplex/2_faze.htm>

[7] MAŇAŠ, M., LAUBER, J., PELIKÁN, J., HAVELKA, S., LAGOVÁ, M.: *Matematické metody v ekonomice*. Praha: SNTL-nakladatelství technické literatury, 1991. s. 169-181. ISBN 80-7079-157-8.

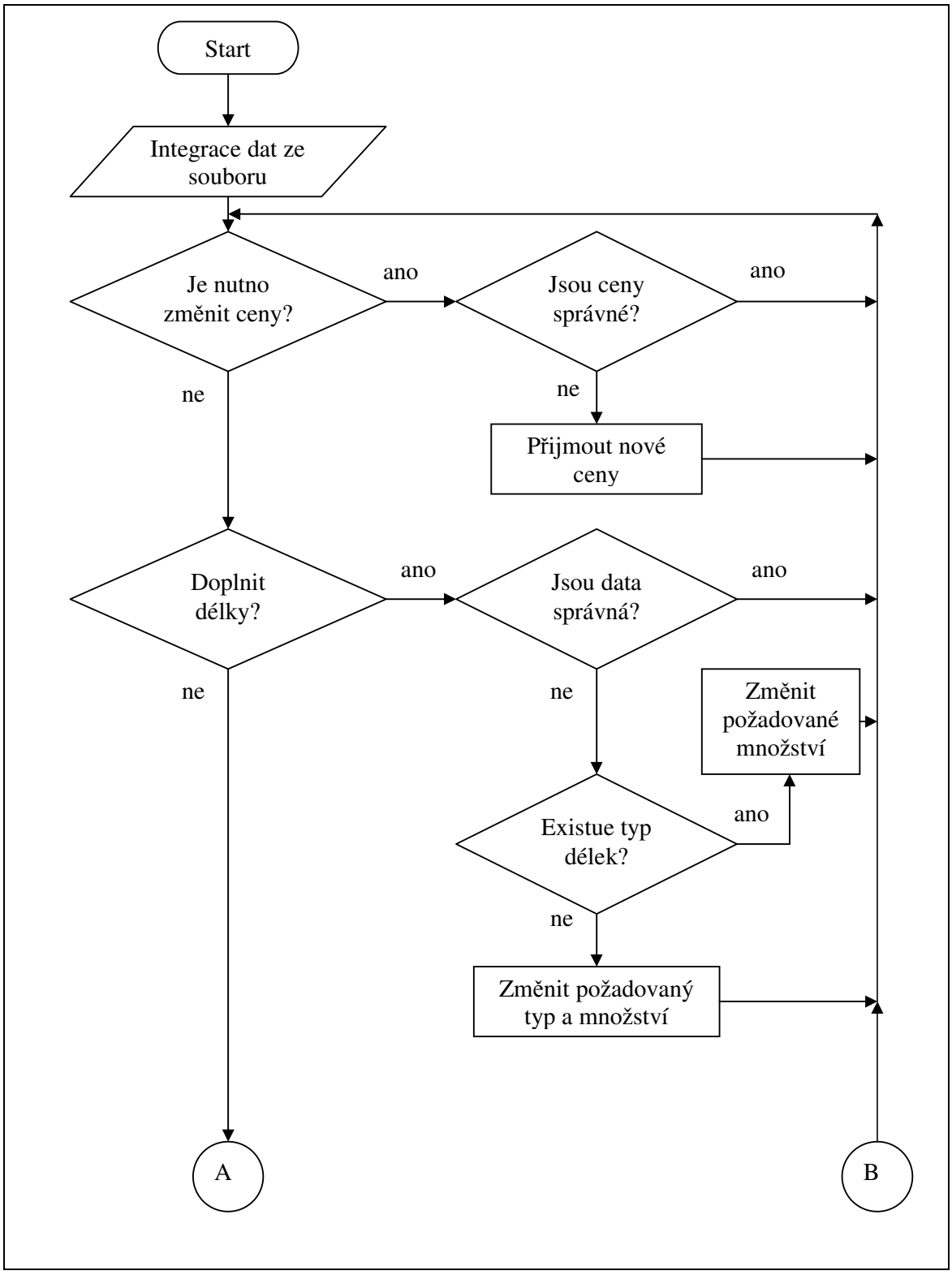
[8] DVOŘAK, J.: *Řešení úloh linearního programování v MS Excel.* Metodická pomůcka. Brno.

[9] ARCHANGELSKIJ, A.: *Jazyk Pascal a základy programování v Delphi*. Moskva: Vydavatelství «BINOM», 2008. 496 s. ISBN 978-5-9518-0241-5.

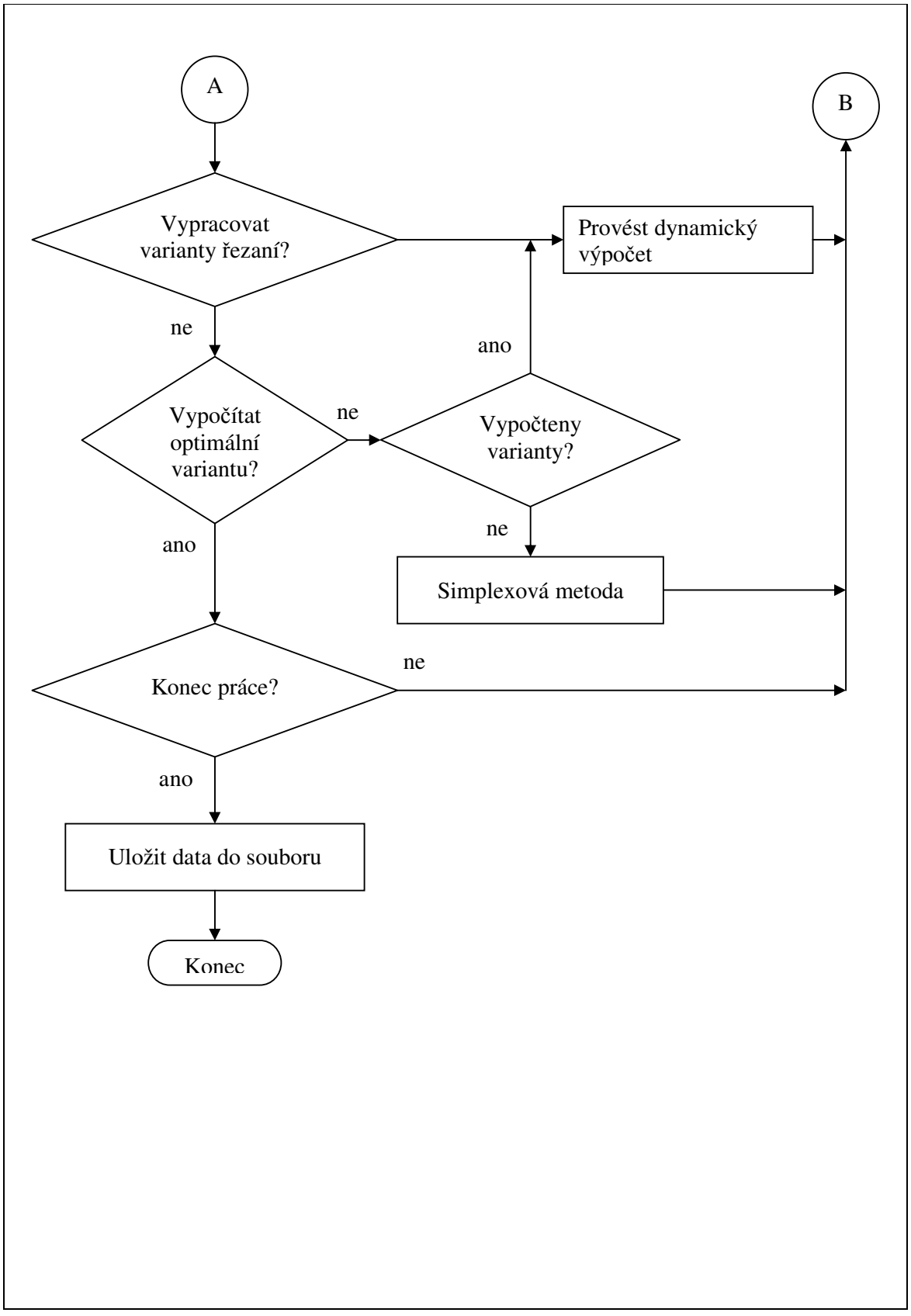
[10] BASL, J., TŮMA, M., GLASL, V.: *Modelování a optimalizace podnikových procesů*. Plzeň: ZČU v Plzni. 2002. 140 s. ISBN 80-7082-936-2.

[11] FIALA, A.: *Automatizované systémy řízení slévárenských procesů*. Skriptum. Brno: VUT v Brně, 1986.

Příloha A – Algoritmus programu «Optimizer»



Obr. A.1 Algoritmus programu «Optimizer»



Obr. A.1 Algoritmus programu «Optimizer» (pokračování)

Příloha B – Text programu «Optimizer»

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls;

type
  plan = record
      dlina:extended;
      kolich:integer;
      cost:extended;
  end;
  TForm1 = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Edit1: TEdit;
    Label6: TLabel;
    Edit2: TEdit;
    Label7: TLabel;
    Edit3: TEdit;
    Label8: TLabel;
    Edit4: TEdit;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    GroupBox2: TGroupBox;
    Memo1: TMemo;
    Edit5: TEdit;
    Edit6: TEdit;
    Label12: TLabel;
    Label13: TLabel;
    Button1: TButton;
    Button2: TButton;
    Label14: TLabel;
    Button3: TButton;
    Button4: TButton;
    Memo2: TMemo;
    Button5: TButton;
    ProgressBar1: TProgressBar;
```

Obr. B.1 Text programu «Optimizer»

```
    StatusBar1: TStatusBar;
    Memo3: TMemo;
    Button6: TButton;
    Edit7: TEdit;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  dlina1,cena1,cena2,dlina2: extended;
  full_plan: array of plan;
  counter: array of integer;
  cost:extended;
  raspil:array of array of extended;
  flag3:boolean;
  max_step:integer;
implementation

{$R *.dfm}
procedure log_ar(sender:Tmemo;ar:array of extended;name:string);
var
st:string;
ii:integer;
begin
  st:=name+': '+inttostr(length(ar))+'vol:';
for ii:=0 to length(ar)-1 do
st:=st+floattostr(ar[ii])+' ';
sender.Lines.Add(st);
end;

procedure log_ar_int(sender:Tmemo;ar:array of integer;name:string);
var
st:string;
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
ii:integer;
begin
  st:=name+': '+inttostr(length(ar))+'vol:';
for ii:=0 to length(ar)-1 do
st:=st+inttostr(ar[ii])+' ';
sender.Lines.Add(st);
end;

procedure log(sender:Tmemo;st:string);
begin
  sender.Lines.Add(st);
end;


function next_count(count:longint;ogr:array of integer;pos:longint):integer;
var
len,ii:integer;
it:integer;
begin
len:=length(ogr);
for ii:=0 to length(ogr)-1 do begin
    if pos=ii then next_count:=count mod ogr[ii];
    count:=count div ogr[ii];
end;
end;

function max_fun():integer;
begin
 max_fun:=2;
end;

function fun(st:string;i:integer):extended;
begin
 if st='dlina' then begin
    if i=1 then fun:=dlina1;
    if i=2 then fun:=dlina2
 end;
 if st='cena' then begin
    if i=1 then fun:=cena1;
    if i=2 then fun:=cena2;
 end;
end;

procedure init_counter(len:integer);
var
  temp_i:integer;
begin
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
  setlength(counter,len);
  for temp_i:=0 to len-1 do
    counter[temp_i]:=0;
end;

procedure next_(ogr:array of integer;count:longint);
label
l1;
var
len,temp_i:integer;
begin
 len:=length(counter)-1;
 for temp_i:=len downto 0 do begin
  counter[temp_i]:=count mod ogr[temp_i];
  count:=count div ogr[temp_i];
 end;
end;
procedure result_();
var
ii,jj:integer;
begin
end;

procedure make_default(input:boolean;output:boolean);
begin
if (input)and(not(output)) then begin
 dlina1:=8; cena1:=11.5;
 dlina2:=5; cena2:=8.36;
end;
if (output)and(not(input)) then begin
setlength(full_plan,3);

full_plan[0].dlina:=1.5;
full_plan[0].kolich:=300;
full_plan[1].dlina:=2;
full_plan[1].kolich:=200;
full_plan[2].dlina:=3;
full_plan[2].kolich:=100;
end;

if input and output then begin
   dlina1:=8; cena1:=11.5;
end;
if (not(input)) and (not(output)) then begin
   dlina2:=5; cena2:=8.36;
end;
end;
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
procedure fill(parametr:string);
var
temp_i:integer;
begin
  if parametr='memo' then begin
      form1.Memo1.Clear;
      if length(full_plan)>0 then
      for temp_i:=0 to length(full_plan)-1 do
form1.Memo1.Lines.Add(floattostr(full_plan[temp_i].dlina)+'/'+inttostr(full_plan[temp_i].kol
ich) );
      end;
  if parametr='label' then begin
      form1.Edit1.Text:=floattostr(dlina1);
      form1.Edit2.Text:=floattostr(cena1);
      form1.Edit3.Text:=floattostr(dlina2);
      form1.Edit4.Text:=floattostr(cena2);
      form1.Label10.Caption:=floattostr(dlina1*cena1)+' eur';
      form1.Label11.Caption:=floattostr(dlina2*cena2)+' eur';
  end;
end;

procedure dell(pos:integer);
var
temp_i:integer;
begin
   if pos<length(full_plan)-1 then
   for temp_i:=pos to length(full_plan)-2 do begin
       full_plan[temp_i]:=full_plan[temp_i+1];
   end;
   setlength(full_plan,length(full_plan)-1);
   showmessage('DELL:'+inttostr(length(full_plan)));
end;


procedure sortplan(tosort:array of plan);
label
l1;
var
min_e,max_e,temp_ex:extended;
pos_from,pos_max,temp_i,temp_j,temp_int:integer;
sorted:boolean;
temp_plan:plan;
sorted_plan:array of plan;
max_i,position:integer;
arint:array of integer;
begin
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
 l1:
 for temp_i:=0 to length(full_plan)-1 do begin
    for temp_j:=temp_i to length(full_plan)-1 do begin
       if (temp_i<>temp_j)and (full_plan[temp_i].dlina=full_plan[temp_j].dlina) then
         begin
           dell(temp_j); goto l1;
         end;
    end;
 end;

 repeat
    max_e:=0;
    sorted:=false; position:=0;
    for temp_i:=0 to length(full_plan)-1 do begin
       if (full_plan[temp_i].dlina>max_e) then begin
         max_e:=full_plan[temp_i].dlina;
         temp_plan:=full_plan[temp_i];
         position:=temp_i;
       end;
    end;
      if max_e>0 then begin
       setlength(sorted_plan,length(sorted_plan)+1);
       sorted_plan[length(sorted_plan)-1]:=temp_plan;
       full_plan[position].dlina:=0; end;
'+floattostr(temp_plan.dlina));
 until max_e=0;
showmessage('sortplan:sorted_plan'+inttostr(length(full_plan))+inttostr(length(sorted_plan)));
 setlength(full_plan,length(sorted_plan));
 for temp_i:=0 to length(full_plan)-1 do
   full_plan[temp_i]:=sorted_plan[temp_i];
(full_plan[temp_i].dlina=full_plan[temp_i+1].dlina)and(temp_i+pos_max<length(full_plan)-
2) then
end;

procedure updplan(len:extended;kol:integer);
var
flag:boolean;
temp_i:integer;
begin
   flag:=true;
   for temp_i:=0 to length(full_plan)-1 do
     if full_plan[temp_i].dlina=len then begin
       full_plan[temp_i].kolich:=kol;
       flag:=false;
     end;
if flag then begin
   setlength(full_plan,length(full_plan)+1);
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
    full_plan[length(full_plan)-1].dlina:=len;
    full_plan[length(full_plan)-1].kolich:=kol;
    end;
    sortplan(full_plan);
end;

procedure last_check();
label
l1;
var
temp_i:integer;
begin
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
var
f:file of extended;
file_plan:file of plan;
ee,eee:extended;
ii,temp_i:integer;
begin
assignfile(file_plan,'ghbdtn.ini');
rewrite(file_plan);
for temp_i:=0 to length(full_plan)-1 do
write(file_plan,full_plan[temp_i]);
closefile(file_plan);

assignfile(f,'gjrf.ini');
rewrite(f);
for ii:=1 to max_fun do begin
ee:=fun('dlina',ii); eee:=fun('cena',ii);
write(f,ee,eee); end;
closefile(f);
halt;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
form1.Memo1.Clear;
setlength(full_plan,0);
end;


procedure TForm1.FormCreate(Sender: TObject);
var
f:file of extended;
file_plan:file of plan;
temp_i:integer;
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
if temp_len=0 then begin make_default(false,true);
                    end else begin
   reset(file_plan);
   setlength(full_plan,temp_len);
   for temp_i:=0 to temp_len-1 do
      read(file_plan,full_plan[temp_i]);
end;
closefile(file_plan);
end
else make_default(false,true);
fill('memo');
if fileExists('gjrf.ini') then begin
  assignfile(f,'gjrf.ini');
  reset(f); read(f,dlina1,cena1,dlina2,cena2);
  closefile(f);
end else
  begin make_default(true,false);
  end;
if dlina1<=0 then make_default(true,true);
if dlina2<=0 then make_default(false,false);
fill('label');
end;

procedure TForm1.Button1Click(Sender: TObject);
var
dlina,kolvo:boolean;
e:extended;
i:integer;
begin

 dlina:=trystrtofloat(form1.Edit5.Text,e);
 kolvo:=trystrtoint(form1.Edit6.Text,i);
 if trystrtofloat(form1.edit5.text,e) then e:=strtofloat(form1.edit5.text) else
 showmessage('Program can not read length. For example: 12,34');
 if trystrtoint(form1.edit6.text,i) then i:=strtoint(form1.edit6.text)
 else showmessage('Program can not read volume. Must be integer. For example: 3');
 if (dlina) and (kolvo) and (e>0) and (i>0) then updplan(e,i) else showmessage('Length and
kolicestvo more than zero.');
 fill('memo');
end;

procedure TForm1.Button3Click(Sender: TObject);
var
first,second,apply:boolean;
te,temp_e:extended;
ti,temp_i:extended;
begin
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
first:=false;
second:=false;
apply:=true;
if trystrtofloat(form1.Edit1.Text,temp_e) then temp_e:=strtofloat(form1.Edit1.Text)
else showmessage('Can not read length of first doska');
if trystrtofloat(form1.Edit2.Text,temp_i) then temp_i:=strtofloat(form1.Edit2.Text)
else showmessage('Can not read price of first doska');
if (temp_i>0)and(temp_e>0) then begin dlina1:=temp_e; cena1:=temp_i; showmessage('First
X was changed'); first:=true; end
else begin apply:=false; showmessage('Length and price must be more than zero!') end;
if trystrtofloat(form1.Edit3.Text,temp_e) then temp_e:=strtofloat(form1.Edit3.Text)
else showmessage('Can not read length of second doska');
if trystrtofloat(form1.Edit4.Text,temp_i) then temp_i:=strtofloat(form1.Edit4.Text)
else showmessage('Can not read price of second doska');
if (temp_i>0)and(temp_e>0) then begin dlina2:=temp_e; cena2:=temp_i; second:=true;
showmessage('Second X was changed'); end
else begin apply:=false; showmessage('Length and price must be more than zero!') end;
fill('label');
end;

procedure TForm1.Button4Click(Sender: TObject);
var
oo,jj,ii:integer;
len:integer;
max_count:longint;
count:longint;
num_iter:array of integer;
temp_ar:array of extended;
temp_dlina:extended;
st:string;
begin
form1.Memo2.Clear;

len:=length(full_plan);

setlength(raspil,1);
setlength(raspil[0],len);
setlength(num_iter,len);
setlength(temp_ar,len);

for oo:=1 to max_fun do begin
max_count:=1;
for ii:=0 to len-1 do begin
   max_count:=max_count * (trunc(fun('dlina',oo) / full_plan[ii].dlina) + 1);
   raspil[0][ii]:=full_plan[ii].dlina;
   num_iter[ii]:= trunc(fun('dlina',oo) / full_plan[ii].dlina) + 1;
end;
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
count:=1;
for ii:=count to max_count do begin
   for jj:=0 to len-1 do begin
      temp_ar[jj]:=next_count(count,num_iter,jj);
   end;

   temp_dlina:=0;
   for jj:=0 to len-1 do
     temp_dlina:=temp_dlina+(temp_ar[jj]*raspil[0][jj]);
   if (temp_dlina>0) and (temp_dlina<=fun('dlina',oo)) then begin
     setlength(raspil,length(raspil)+1);
     setlength(raspil[length(raspil)-1],len+2);
     raspil[length(raspil)-1][0]:=fun('cena',oo)*fun('dlina',oo);
     raspil[length(raspil)-1][length(raspil[length(raspil)-1])-1]:=fun('dlina',oo)-temp_dlina;
     for jj:=1 to len do
       raspil[length(raspil)-1][jj]:=temp_ar[jj-1];
       begin st:=inttostr(num_var)+' variants:'+floattostr(fun('dlina',oo))+'m
'+floattostr(raspil[length(raspil)-1][0])+'euro: ';
       for jj:=0 to len-1 do begin
        st:=st+' '+floattostr(raspil[length(raspil)-1][jj+1])+' by'+floattostr(raspil[0][jj])+'m';
       end;
       st:=st+' : '+floattostr(raspil[length(raspil)-1][length(raspil[length(raspil)-1])-1])+'m by
ends';
       log(memo2,st);
       inc(num_var);
       end;
   end;
inc(count);

end;
end;
  st:='Program found '+inttostr(length(raspil)-1)+' variants';
  form1.StatusBar1.Panels[1].Text:=st;
end;

procedure iter();
var
 temp_i:integer;
begin
end;

procedure TForm1.Button5Click(Sender: TObject);
label
l1,l2,l3;
var
kk,jj,ii:integer;
st:string;
pos_vvod,pos_vivod,num_vivod,num_vvod,len:integer;
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
min,max:extended;
flag1,flag2{,flag3}:boolean;
step:integer;
ostatki:extended;
full_cost:extended;
ostatki_cost:extended;
{basis_minus:array of array of extended;}
  a_temp:array of array of extended;
  ocenki,b_temp:array of extended;
  basis:array of array of extended;
  a:array of array of extended;
  b:array of extended;
  nums:array of integer;
  minus_nums:array of integer;

begin
if length(raspil)<=0 then form1.Button4Click(form1.Button5);
len:=length(raspil[length(raspil)-1]);
setlength(a,len-1);
form1.memo3.clear;
log(memo3,'At first:');
log(memo3,'Matrix A');
for ii:=0 to length(a)-1 do begin
    setlength(a[ii],length(raspil)-1);
    for jj:=0 to length(raspil)-2 do begin
        a[ii][jj]:=raspil[jj+1][ii];
    end;
    log_ar(memo3,a[ii],'Line'+inttostr(ii+1));
end;
showmessage('trace -6');
setlength(b,length(full_plan)+1);
for ii:=0 to length(full_plan)-1 do  {was 1- length(full_plan)+1}
b[ii+1]:=full_plan[ii-1].kolich;
if length(full_plan)>0 then b[0]:=0;
step:=0;
flag3:=false;
flag2:=false;
log_ar(memo3,b,'Vecrot B');
showmessage('trace -5');
log(memo3,'Pover of basis: '+inttostr(length(a)));
len:=length(a);
setlength(basis,len,len);
basis[0][0]:=-1;
for ii:=1 to len-1 do
basis[ii][0]:=0;
setlength(nums,len-1);
showmessage('trace -4');
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
for kk:=1 to len-1 do begin
for ii:=0 to length(a[0])-1 do
  if a[kk][ii]=1 then begin
     flag1:=true;
     for jj:=1 to length(a)-1 do
        if (jj<>kk)and(a[jj][ii]>0) then flag1:=false;
     if flag1=true then begin
     for jj:=0 to length(a)-1 do
        basis[jj][kk]:=a[jj][ii];
     nums[kk-1]:=ii+1;
     flag2:=true;
     end;
  end;
end;
showmessage('trace -3');

LOG_ar_int(memo3,nums,'nums');
LOG(memo3,'first basis:');
for ii:=0 to length(basis)-1 do begin
LOG_ar(memo3,basis[ii],'basis '+inttostr(ii));
end;

{st:=inttostr(length(a))+'* ';
for ii:=0 to length(a)-1 do
st:=st+inttostr(length(a[ii]))+' ';
showmessage(st);}
setlength(a_temp,length(a),length(a[0]));
setlength(b_temp,length(b));
for ii:=0 to length(a)-1 do begin
{  setlength(a_temp[ii],length(a[ii]));}
  for jj:=0 to length(a[ii])-1 do begin
     a_temp[ii][jj]:=0;
     for kk:=0 to length(a)-1 do begin
     a_temp[ii][jj]:=a_temp[ii][jj]+a[kk][jj]*basis[ii][kk];
     end;
  end;
  b_temp[ii]:=0;
  for jj:=0 to length(a)-1 do begin
     b_temp[ii]:=b_temp[ii]+b[ii]*basis[ii][jj];
  end;
end;

setlength(ocenki,length(b)-1);
flag2:=false;
flag3:=false;
showmessage('trace -2');
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
repeat
inc(step);
if(step>=max_step) then begin
  st:='Result:';
  flag2:=true;
  flag3:=true;
  for ii:=1 to length(b)-1 do
     st:=st+'num'+inttostr(nums[ii-1])+' '+inttostr(round(b[ii]))+' ';
  form1.StatusBar1.Panels[0].Text:=st+'overstep';
  form1.Memo3.Lines.Add(st);
end;

LOG(memo3,'Step '+inttostr(step)+':');
LOG(memo3,'Matrix A');
showmessage('trace -1');
for ii:=0 to length(a)-1 do begin
        b[ii]:=b_temp[ii];
    for jj:=0 to length(a[ii])-1 do begin
        a[ii][jj]:=a_temp[ii][jj];
    end;
    LOG_ar(memo3,a[ii],'Line'+inttostr(ii+1));

end;
showmessage('trace 0');
LOG_ar(memo3,b,'Vector B:');
LOG_ar(memo3,a[0],'MARK: ');
flag1:=true;
for ii:=0 to length(a[0])-1 do
  if a[0][ii]>0 then flag1:=false;
if (flag1) then begin
  st:='Result:';

  for ii:=1 to length(b)-1 do
     st:=st+'num'+inttostr(nums[ii-1])+' '+inttostr(round(b[ii]))+' ';
  form1.Memo3.Lines.Add(st);
  flag3:=true;
  if not(flag2) then form1.StatusBar1.Panels[0].Text:=st+'Optimal';
  flag2:=true;
  full_cost:=0;
  ostatki:=0;
  ostatki_cost:=0;
for ii:=0 to length(nums)-1 do begin
  st:=inttostr(round(b[ii+1]))+': ';
  full_cost:=full_cost+raspil[nums[ii]][0]*round(b[ii+1]);
  ostatki:=ostatki+raspil[nums[ii]][length(raspil[nums[ii]])-1]*round(b[ii+1]);
ostatki_cost:=ostatki_cost+raspil[nums[ii]][length(raspil[nums[ii]])-
1]*raspil[nums[ii]][0]*round(b[ii+1]);
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
for jj:=0 to length(raspil[nums[ii]])-2 do begin
  st:=st+floattostr(raspil[nums[ii]][jj+1])+' '{+floattostr(full_plan[jj].dlina)}+', ' ;
     end;
  log(memo3,st);
  end;
  st:='Full_cost:'+floattostr(full_cost)+', cuttings: '+floattostr(ostatki)+' meters, which
cost'+floattostr(ostatki_cost)+ ' euro.';
  log(memo3,st);
end;
showmessage('Trace1');
max:=0;
num_vvod:=0;
st:=' Vector A vvod';
for ii:=0 to length(a[0])-1 do
  if a[0][ii]>max then begin max:=a[0][ii]; num_vvod:=ii+1; pos_vvod:=ii;{pos столбца L}
end;

if max=0 then begin
   st:='Result:';
  for ii:=1 to length(b)-1 do
     st:=st+'num'+inttostr(nums[ii-1])+' '+inttostr(round(b[ii]))+' ';
  if not(flag2) then form1.StatusBar1.Panels[0].Text:=st+'But no max';
  flag2:=true;
  end;
LOG(memo3,'Inside variant: '+inttostr(pos_vvod+1));

st:='';
for ii:=0 to (length(a)-1) do begin
showmessage(inttostr(ii)+' '+inttostr(pos_vvod)+' '{+floattostr(a[ii][pos_vvod])});
{st:=st+' '+floattostr(a[ii][pos_vvod]);}
end;

for ii:=0 to length(ocenki)-1 do begin
if a[ii+1][pos_vvod]>0 then ocenki[ii]:=b[ii+1]/a[ii+1][pos_vvod] else ocenki[ii]:=0;
end;
LOG_ar(memo3,ocenki,'GRADES');

for ii:=0 to length(ocenki)-1 do
if ocenki[ii]>0 then
begin min:=ocenki[ii];
pos_vivod:=ii+1;
num_vivod:=nums[ii]; end;
showmessage('trace2.1');
for ii:=0 to length(ocenki)-1 do begin
   if (min>=ocenki[ii])and(ocenki[ii]>0) then begin
      num_vivod:=nums[ii];
      pos_vivod:=ii+1;
      nums[ii]:=num_vvod;
```

Obr. B.1 Text programu «Optimizer» (pokračování)

```
     min:=ocenki[ii];
   end;
end;
log(memo3,'Ouside variant: '+inttostr(num_vivod));
Log(memo3,'Position: '+inttostr(pos_vvod)+'col '+inttostr(pos_vivod)+' row');
showmessage('Trace3');
for ii:=0 to length(a)-1 do begin
if ii=pos_vivod then begin
b_temp[pos_vivod]:=b[pos_vivod]/a[pos_vivod][pos_vvod];
showmessage(floattostr(b_temp[pos_vivod])+'='+floattostr(b[pos_vivod])+'/'+floattostr(a[pos
_vivod][pos_vvod]));
for jj:=0 to length(a[pos_vivod])-1 do begin
  a_temp[pos_vivod][jj]:=a[pos_vivod][jj]/a[pos_vivod][pos_vvod];
  if a_temp[pos_vivod][jj]<0.0001 then a_temp[pos_vivod][jj]:=0;
end;
end else begin
    b_temp[ii]:=b[ii]-(a[ii][pos_vvod]*(b[pos_vivod]/a[pos_vivod][pos_vvod]));
  showmessage(floattostr(b_temp[ii])+'='+floattostr(b[ii])+'-
'+floattostr(a[ii][pos_vvod])+'*'+floattostr(b[pos_vivod])+'/'+floattostr(a[pos_vivod][pos_vvo
d]));
for jj:=0 to length(a[pos_vivod])-1 do begin
a_temp[ii][jj]:=a[ii][jj]-a[ii][pos_vvod]*(a[pos_vivod][jj]/a[pos_vivod][pos_vvod]);
  if a_temp[ii][jj]<0.0001 then a_temp[ii][jj]:=0;
end;
end;
end;
until {not}(flag2);
{l3:}
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
if trystrtoint(form1.Edit7.Text,max_step)=false then
begin
  showmessage('Can not take number of steps! Make default');
  max_step:=15;
  form1.Edit7.Text:=inttostr(max_step);
end else  begin
showmessage('Take number of step is '+form1.Edit7.Text);
max_step:=strtoint(form1.edit7.text); end;//if

end;

end.
```

Obr. B.1 Text programu «Optimizer» (pokračování)