ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

PROVOZNĚ EKONOMICKÁ FAKULTA

KATEDRA SYSTÉMOVÉHO INŽENÝRSTVÍ

DIPLOMOVÁ PRÁCE

Moduly pro řešení okružního dopravního problému v MS-Excel v Jazyce Visual Basic for Applications

Vedoucí diplomové práce: RNDr. Petr Kučera

Autor diplomové práce: Bc. Jan Rydval

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Moduly pro řešení okružního dopravního problému v MS-Excel v Jazyce Visual Basic for Applications" jsem vypracoval samostatně pod vedením vedoucího diplomové práce RNDr. Petra Kučery a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15. dubna 2009 _____

Poděkování

Rád bych touto cestou poděkoval panu RNDr. Petru Kučerovi a pracovníkům firmy Alfasped Logistik s.r.o. za odborné vedení, rady a připomínky při zpracování této diplomové práce.

MODULY PRO ŘEŠENÍ OKRUŽNÍHO DOPRAVNÍHO PROBLÉMU V MS-EXCEL V JAZYCE VISUAL BASIC FOR APPLICATIONS

MODULES FOR SOLVING THE TRAVELLING SALESMAN PROBLEM IN MS-EXCEL IN VISUAL BASIC FOR APPLICATIONS

SOUHRN

Tato diplomová práce se věnuje problematice okružního dopravního problému. Zabývá se základní charakteristikou tohoto problému, přehledem metod řešení a rozborem základních algoritmů řešení.

Hlavní úkolem je tvorba modulů pro řešení okružního dopravního problému a jeho úloh ve VBA dle algoritmů metody nejbližšího souseda a Vogelovy metody. Je zde popsán jejich kód a uživatelské rozhraní pro ovládání aplikace pro zadávání vstupních dat. Poté je celá aplikace odzkoušena v prostředí reálné firmy, což je popsáno v případové studii.

Protože tato práce obsahuje praktický příklad řešení daného problému, je zde čtenář seznámen i s možností aplikace MS Excelu a jeho programovacího jazyka Visual Basic for Applications (VBA).

KLÍČOVÁ SLOVA

Dopravní logistika, Okružní dopravní problém, Mayerova metoda, metoda nejbližšího souseda, Vogelova metoda, Visual Basic for Applications

SUMMARY

This diploma thesis considers optimization of the Travelling Salesman problem. It considers basic charakteristic of this problem, methods for solving it and a list of basic algorithms of this solving. Because a practical case is inclusive of these thesis the reader is familiarised with the possibilities of application Excel and programing language Visual Basic for Applications.

The main content is creating of modules for solving of the Travelling Salesman Problem in accordance with algorithms of nearest neighbour method and of Vogel approximation method (loss method). There is described their code and user´s interface for inputting of basic data. Then the application is used in a real firm.

KEY WORDS

Transport Logistic, Travelling Salesman Problem, Mayer method, nearest neighbour method, Vogel approximation method (loss method), Visual Basic for Applications

Obsah

1 Úvod

V dnešní dynamicky se rozvíjející společnosti hraje doprava velmi významnou roli. A jako s každou oblastí života ve společnosti souvisí i s dopravou financování a náklady na dopravu. V dopravní logistice se můžeme často setkat s problémem realizace optimálního okružního spojení. S tímto problémem se můžeme setkat nejčastěji při rozvozu různých druhů zboží a materiálů, svozu komunálního odpadu či čištění vozovek, ale i například v městské hromadné dopravě při jejím plánování, optimalizaci a správě a mnoha dalších situacích. Touto problematikou a její optimalizací se zabývá v teoretické úrovni teorie grafů a v praktickém pojetí logistické metody resp. metody dopravní logistiky.

Tato diplomová práce se tedy zabývá jednou z oblastí řešení a optimalizace v dopravní logistice, a to využitím operační a systémové analýzy a jejích metod, konkrétně jedné z mnoha druhů dopravních optimalizačních úloh – Okružním dopravním problémem, který je rovněž znám pod pojmem Problém obchodního cestujícího (Travelling Salesman Problem - TSP).

Ruční písemné řešení těchto úloh je sice také možné, ale při větším počtu míst dané okružní trasy je velmi náročné a zdlouhavé. Program MS Excel přináší vhodné prostředí pro zadávání potřebných vstupních dat, programování vhodných algoritmů pro řešení okružního dopravního problému i ke koncovému výstupu dosažených výsledků.

A protože tyto metody nalézají uplatnění nejen logistických společnostech a dopravní logistice samotné, ale i v dalších oblastech lidské společnosti, uvedeme si v této práci, jak se dá problém obchodního cestujícího řešit pomocí počítače a jeho softwarového vybavení. Ukážeme si různé metody řešení tohoto problému a pomocí vytvořené aplikace v MS Excel i jeho praktické řešení, a to nejen teoreticky, ale i, jak bude v případové studii této práce ukázáno, v reálném prostředí skutečného podniku.

2 Cíl práce a metodika

2.1 Cíl práce

Jakoukoliv optimalizační úlohu je vždy možné řešit ručně, za využití příslušných znalostí metod a postupů. Tento postup je však velmi náročný a nezaručuje nám potřebnou jistotu přesnosti a bezchybnosti a dále již od určité hranice nelze tyto úlohy řešit bez pomocí výpočetní techniky.

A právě to je hlavním cílem této práce. Vytvořit vlastní programové moduly pro řešení úlohy okružního dopravního problému. Moduly jsou vytvořeného pomocí programovacího jazyka Visual Basic for Applications.

Dílčími cíli jsou pak seznámení čtenáře s danou problematikou okružního dopravního problému a jeho zařazení v dopravní logistice i s využívanými metodami a jejich algoritmy k řešení tohoto problému. Dále seznámit čtenáře s prostředím MS Excel a vysvětlení některých základních pojmů a principů, které jsou s touto problematikou úzce spjaty.

Dalším cílem je zpracování případové studie. Jde o reálnou úlohu vycházející z reálné situace firmy zabývající se vnitrostátním i mezinárodním zasilatelstvím. Cílem je ukázat, jaké problémy je možno řešit za pomocí programového vybavení a s jakou úspěšností.

Práce je tedy rozdělena na dvě velké části rozdělené do kapitol a podkapitol. V první je formou literární rešerše objasněna daná problematika a seznámení s prostředím MS Excel a jeho programovacího jazyka Visual Basic for Applications a v druhé části se nachází samotné programování modulů k řešení úloh a jejich praktického využití v reálné firmě.

2.2 Metodika

K dosažení zvolených cílů byl zvolen následující postup:

Jako první je zpracováno samotné uvedení do problematiky, a to v souvislosti zařazení okružního dopravního problému do dopravní logistiky, tím jsou získány informace a podklady pro další pokračování v tématu. Zdrojem jsou zejména odborné publikace, skripta vysokých škol a též dnes všudypřítomný internet a jeho stránky zabývající se danou problematikou. Možnosti řešení okružního dopravního problému včetně popisu jednotlivých metod a algoritmů jsou taktéž součástí literární rešerše. Následuje popis prostředí, ve kterém je možno vytvářet nejrůznější moduly a uživatelské formuláře pro řešení optimalizačních úloh. Zejména možnosti tabulkových procesorů a programovacího prostředí, které se k němu váže.

Dalším krokem je samotná programová tvorba, možnosti jejího využití a ovládání programu.

A na závěr je vyřešena vlastní případová studie a celkové zhodnocení přínosu programových modulů pro řešení okružního dopravního problému.

3 Okružní dopravní problém, význam a metody

3.1 Úvod do problematiky dopravní logistiky

3.1.1 Logistika jako pojem

Při vysvětlování pojmu logistika z běžně dostupných slovníků, můžeme zjistit, že slovo logistika je již velmi staré, které postupně nabývalo různých významů. Naučný slovník z roku 1931 pod pojmem logistika uvádí: Ve starověku až do r. 1600 praktické počítání s číslicemi, na rozdíl od aritmetiky, vědecké nauky o číslech. Vieta zavedl r. 1591 výraz logistica numerosa pro počítání číslicemi a logistica speciosa na počítání pomocí písmen. Kromě toho nazývá se tak i algoritmická neb algebraická logika. [1]

Příruční slovník naučný z roku 1964 uvádí pod heslem logistika: [řec.] 1. název pro matematickou logiku; 2. novopozitivistický výklad matematické logiky. [2]

A konečně Všeobecná encyklopedie z rolu 1997 uvádí: 1. hosp. činnost zaměřená na organizačně tech. Zajišťování přísunu optimálního množství např. materiálních prvků (surovin, materiálu, polotovarů, hotových výrobků), pohybu lidí, přenosu informací, a to ve správný čas, na správné místo a s přiměřenými náklady; 2. log. Viz logika matematická; 3. vojenství jedna z nejvýzn. kategorií souč. vojenství. Široký soubor činností a institucí (jednotek, velitelství) v -> armádách, kterých se vyvíjel jako podpora hl. úsilí vojska (administrativou, zásobováním, službami zajišťujícími péči o osoby, výzbroj, výstroj, techniku apod.). [3]

Mnohem větší rozšíření nalezla logistika v oblast vojenství. Již byzantský císař Leontos VI. (886 – 911) charakterizoval logistiku: Předmětem logistiky je mužstvo zaplatit, příslušně vyzbrojit a vybavit ochranou i municí, včas a důsledně se postarat o jeho potřeby a každou akci v polním tažení příslušně připravit, tzn. Vypočítat prostor a čas, správně ohodnotit terén z hlediska pohybu vojsk i v případě nutnosti jejich rozdělení.[4] V této větě je specifikován a náplň logistiky, která musí zvládnout pohyby lidí, pohyby materiálu a to tak, aby se příslušný objekt nacházel na potřebném místě v potřebném čase.

Vytvořením racionálních a dobře fungujících přepravních řetězců pro zásobování zbraněmi a municí, proviantem i výstrojí bylo proto neobyčejně důležitým úkolem, při kterém bylo vždy třeba překonat veliké vzdálenosti a zejména za II. světové války, kdy rozsah materiálních toků představoval obrovská kvanta materiálu, doznala logistika svého maximálního rozšíření. [5]

Velice důležité pro pozdější uplatnění logistiky i v civilním sektoru se ukázalo úspěšné uplatnění logistiky včetně jí využívaného matematického aparátu umožňujícího účinně řešit problém zásob, dopravní a rozmisťovací problémy a další, ke kterým došlo

[1] *Nový velký ilustrovaný slovník naučný*. Praha: Gutenberg, 1931, sv. XII.
[2] *Příruční slovník naučný*. Praha: Nakladatelství Československé akademie věd, 1964, díl II.
[3] *Všeobecná encyklopedie ve čtyřech svazcích*. Praha: Nakladatelský dům OP, 1997, díl 2
[4] KORTSCHAK, B. H. *Úvod do logistiky (Co je logistika?).* 2. české vyd. Praha: Babtex, 1995, ISBN 80-85816-06-7
[5] SIXTA, J. a MAČÁT, V. *Logistika - teorie a praxe*. 1. vyd. Brno: CP Books, 2005, ISBN80-251-0573-3

za II. světové války při přípravě a provádění operací spojeneckých vojsk na západní frontě.

Za úsvitu 6. června 1944 se britští a američtí vojáci v rámci obrovské obojživelné operace začínají vyloďovat na pěti normandských plážích. Více než rok trvající přípravy vyvrcholily v zahájení operace Overlord.[6]

Na obrázku 1 - 3.1.1 Vylodění spojeneckých vojsk v Normandii je patrná dokonalá synchronizace organizování dílčích procesů vylodění a následné invaze, což je v souhrnu logistický přesunu vojsk. A právě takto uplatněná vojenská logistika vedla po válce k rozšíření logistiky na řešení analogických problémů v civilní sféře. Vznikla tak hospodářská logistika s řadou účelových aplikací, nejčastěji jako podniková logistika.

Obrázek 1 - 3.1.1 Vylodění spojeneckých vojsk v Normandii[7]

3.1.2 Definice logistiky

Nejprve si na začátku této kapitoly uveďme několik názorů na logistiku, které nám říkají že, logistika je:

Systém tvorby, řízení, regulace a vlastního průběhu materiálového toku, energií, informací a přemisťování osob.

Řízený hmotný tok výrobních a oběhových procesů v odvětvích národního hospodářství a mezi nimi s cílem největší efektivnosti. [8]

Z definice je jasný systémový aspekt celé problematiky, současně i nezbytnost globálního pojetí. Je důležité zdůraznit náhled, jenž je podáván v poslední z těchto dvou definic, a sice to, že jde o tvorbu, řízení a regulaci celého procesu materiálového toku.

---

[6] CAMPBELL, J. *Druhá světová válka*. 1. české vyd. Praha: Mladá fronta, 1995, ISBN 80-204-0474-0

[7] *Operation Overlord - Wikipedia, the free encyclopedia*.[on-line].(17.1.2007) URL: <http://en.wikipedia.org/wiki/File:NormandySupply_edit.jpg>

[8] SIXTA, J. a MAČÁT, V. *Logistika - teorie a praxe*.1. vyd. Brno: CP Books, 2005, ISBN 80-251-0573-3

Zajímavé a cenné jsou i definice logistiky, které uvedli čeští autoři logistické literatury:

Logistiku si lze představit jako posloupnost činností zahrnujících řízení a vlastní realizaci pohybu a skladování materiálů, polotovarů a finálních výrobků. Jde v podstatě o sled obchodních a fyzických operací končících dopravou výrobku k odběrateli. [9]

Logistika je disciplína, která se zabývá řízením toku materiálu v čase a prostoru, a to v komplexu se souvisejícími toky informací a v pojetí, které zahrnuje fyzickou i hodnotovou stránku pohybu materiálu (zboží). [10]

Nelze nezmínit se zde o velmi poučné definici logistiky, kterou vydala Evropská logistická asociace.

Organizace, plánování, řízení a výkon toků zboží vývojem a nákupem počínaje, výrobou a distribucí podle objednávky finálního zákazníka konče tak, aby byly splněny všechny požadavky trhu při minimálních nákladech a minimálních kapitálových výdajích.[11]

Z této definice od Evropské logistické asociace je dobře patrné, že je zde upřednostňována i ekonomická stránka.

Na základě výše uvedených definicí můžeme shrnout definice logistiky jako řízení materiálového, finančního a informačního toku se zohledněním včasného splnění požadavků našeho zákazníka, avšak s ohledem na tvorbu zisku v celém materiálovém toku.

Logistický sytém pak představuje účelně uspořádanou množinu všech technických prostředků, zařízení, budov, cest a pracovníků podílejících se na uskutečňování logistických řetězců. Logistický systém lze považovat za zvláštní druh multisystému, který vymezujeme jako technicko-technologický, informační komunikační systém a systém řízení. Cílem logistického systému podniku je upevnění a posílení pozice podniku jako ekonomického subjektu na trhu.[12]

Logistický řetězec je složen s dílčích hmotných, informačních, peněžních a jiných toků, které probíhají mezi různými subsystémy ve výrobě, dopravě, zasilatelství a v obchodě.[13]

Pro představu si můžeme vybrat logistický řetězec na obr. 2 - 3.1.2, který nám zobrazuje posloupnost jednotlivých prvků v řetězci od dodavatele až k zákazníkovi.

---

[9] GROS, I. *Logistika.* 1. vyd. Praha: VŠCHT v Praze, 1993, ISBN 80-7080-216-2

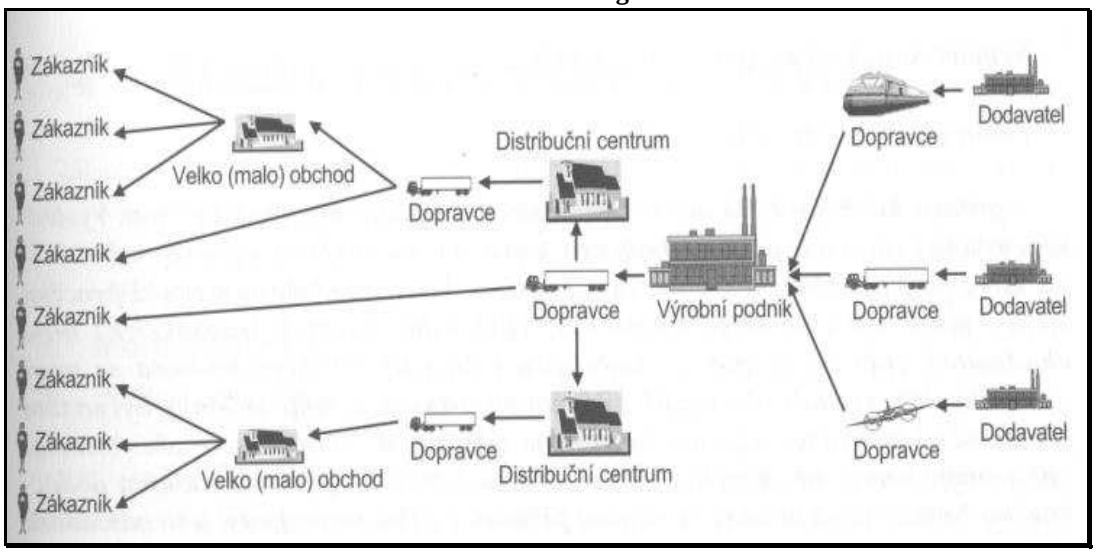
[10] PERNICA, P. *Logistika (základy).* 1. vyd. Praha: VŠE v Praze, 1991, ISBN 80-7079-158-6

[11] *Logistika - zdroj přidané hodnoty!*. [on-line].(17.1.2007) URL: <http://www.logistika.cz/index. php?menu=31>

[12] ZÍSKAL, J. a HAVLÍČEK, J. *Ekonomicko matematické metody II Studijní texty pro distanční studium.* 2. vyd. Praha: ČZU, 2003, ISBN 80-213-0664-5

[13] ZÍSKAL, J. a HAVLÍČEK, J. *Ekonomicko matematické metody II Studijní texty pro distanční studium.* 2. vyd. Praha: ČZU, 2003, ISBN 80-213-0664-5

Obrázek 2 - 3.1.2 Stručné schéma logistického řetězce[14]



Logistiku dále můžeme členit podle oblastí jejího působení na:

- Makrologistiku zabývající se řešením ucelených souborů logistických řetězců v rámci regionů, např. státu.
- Mikrologistiku, která se zabývá logistickými řetězci pouze v rámci jednotlivých podniků.
- Obchodní logistiku, která je zaměřena na logistické řetězce důležité pro podnik v rámci obchodní činnosti.
- Dopravní a zasilatelskou logistiku, která je jednou z nejrozšířenějších neboť koordinuje, synchronizuje a optimalizuje pohyby zásilek po dopravní síti od místa vstupu až k příjemci. [15]

3.2 Funkce dopravy v logistice

Doprava je jedna z lidských činností, která slouží k uspokojování lidských potřeb přemisťování hmotných statků a lidí. Přepravní a dopravní systémy mají v logistice, která představuje řízení materiálových toků od dodavatele přes distribuční kanály až po konečného spotřebitele, důležitou úlohu. Doprava nejenže propojuje jednotlivé části logistického procesu, ale pomáhá i lepšímu napojení a celkové synchronizaci jednotlivých míst styku logistických subsystémů. Tato pomoc je mnohem účinnější, mohou-li přepravní prostředky plnit i určité funkce manipulační, skladovací či obalové jednotky. Z hlediska přemisťování hmotných statků se jedná o tři fáze reprodukčního procesu:

- Doprava ve sféře výroby – uspokojuje potřeby vyvolané technologií výroby, dělbou činností a kooperací a specializací výroby.

[14] SIXTA, J. a MAČÁT, V. *Logistika - teorie a praxe*.1. vyd.Brno: CP Books, 2005, ISBN 80-251-0573-3
[15] SIXTA, J. a MAČÁT, V. *Logistika - teorie a praxe*.1. vyd.Brno: CP Books, 2005, ISBN 80-251-0573-3

- Doprava ve sféře oběhu – uspokojuje potřeby přemisťování nutné k realizaci ekonomického oběhu.
- Doprava ve sféře spotřeby – uspokojuje potřeby přemísťování výrobků, které již vstoupily do spotřeby.[16]

Přemisťování lidí uskutečňuje doprava ve dvou formách. Doprava do pracovního procesu, doprava ve volném čase.

3.3 Metodologie dopravní logistiky

Logistika jako taková by se dala nazvat „kvantitativním řízení“, jehož cílem je upozorňovat na úzká místa v toku materiálu. Logistika má schopnost utvářet procesy od začátku tvorby strategie až po okamžik konečného dodání výrobku, produktu vycházejícího z konečné fáze transformačního procesu, spotřebiteli. Tímto zdůrazněním kvantitativního charakteru logistiky, se dá poukázat na to, že se v logistice neobejdeme bez užívání celé řady metod sloužících pro podporu rozhodování a pro snahu optimalizovat logistické řetězce. Zejména se jedná o metody heuristické a exaktní.

3.3.1 Heuristické metody

Heuristické metody jsou zpravidla užívány pro rozhodovací procesy s vysokou mírou neurčitosti a jejichž postupy nejsou s ohledem na informační zabezpečení a stabilitu algoritmizovány. Řešení těchto úloh se s nemožností užít exaktních metod vyrovnává užitím intuitivních, avšak vysoce kvalifikovaných odhadů expertních pracovníků. Tyto metody je možné členit do dvou skupin. Na metody expertních systémů a na metody tvořivého myšlení.

3.3.1.1 Expertní systémy

Expertní systém je systém, který využívá možností paměťové kapacity výpočetní techniky, její relativní rychlosti při zpracovávání dat a informací a možností určité algoritmizace procesů, které jsou za určitých podmínek schopny již vytvářet první předstupeň umělé inteligence. Expertní systémy lze realizovat buď v dialogovém režimu nebo v režimu dávkovém. Na to, v jakém režimu bude systém pracovat, mají vliv především různé typy problému, kterými se systém zabývá, na požadované době odezvy, na možnostech aplikované výpočetní techniky a to, kde je systém provozován.[17]

3.3.1.2 Metody tvořivého myšlení

Systémy tvořivého myšlení využívají tvůrčí schopnosti expertů. Základním principem tvořivého procesu je výběr, přetváření a spojování prvků předcházejících skutečností. Analýzy ukázaly, že tvořivý potenciál organizací je často důležitější, než ostatní zdroje (finanční, materiálové, lidské apod.). Současné poznatky výzkumu i praxe

[16] DRAHOTSKÝ, I. a ŘEZNÍČEK, B. *Logistiky – procesy a jejich řízení*. 1. vyd. Brno: Computer Press, 2003, ISBN 80-7226-521-0
[17] SIXTA, J. a MAČÁT, V. *Logistika - teorie a praxe*.1. vyd.Brno: CP Books, 2005, ISBN 80-251-0573-3

však ukazují, že tvořivost není výsledkem samovolně probíhajících procesů, ale že je možné a nutné ji posilovat vhodným řízením a usměrňováním.[18]

Při zvyšování tvůrčí kapacity organizace je třeba upřít pozornost především na personální, organizační, technické, informační, pracovní a metodologické aspekty tohoto procesu.

Jednou z významných metod tvůrčího myšlení je brainstorming. Právě brainstorming je založen na tvořivosti a na tvůrčím myšlení. Je to metoda, jak přimět skupinu lidí, aby vytvořily množství nápadů během relativně krátkého času. Je to skupinová práce, která má oproti individuálnímu přístupu několik výhod. Tou hlavní je to, že počet asociací, které účastníky napadnou vzhledem ke stanovenému tématu, je vyšší, než procuje-li jedinec individuálně. Za druhou výhodou zdravého kolektivu lze považovat přirozenou soutěživost, která může zvýšit výkonnost jednotlivých účastníků.

3.3.2 Exaktní metody

Metody exaktní využívají poznatků z vědních oborů, nejčastěji z matematických disciplín, ale částečně i z věd přírodních a využívají se především pro optimalizační a prognostické úlohy rozhodovacích procesů.

3.3.2.1 Metody sloužící k analýze logistických procesů

Tyto metody analyzují a snaží se objasnit celkové logistické procesy v daném podniku. Jedná se zejména o systémovou analýzu, o analýzu ABC, která se hodí k určování prvků, jenž mají největší ekonomický význam v systému popřípadě jeho podsystému. Hodnotová analýza mé za úkol analyzování celkové hospodárnosti podniku při pohybu toku materiálu a analýza nákladů je metoda zabývající se celkovým zjišťováním nákladů v pohybových segmentech logistiky podniku.

3.3.2.2 Statistické metody

Zde se jedná o využití matematické statistiky, a to ve formě statistické analýzy pro diagnózu řídících systémů.[19]

Nejčastěji se zde užívá analýza časových řad s užitím regresní a korelační analýzy, nebo analýza příčinných vazeb mezi ekonomickými, provozními a kvalitativními ukazateli a to v jednoduché či v násobné formě. Výsledkem užití těchto statistických metod je zjištění změny trendu a jaké následné změny vyvolá změna tohoto trendu. Popřípadě se matematická statistika dá využít pro různé formy matematického modelování. Takto získaný statisticko-matematický model se zabývá získáním, popisem a následným zpracováním údajů s cílem nalézt příslušné zákonitosti náhodných hromadných jevů.

3.3.2.3 Simulační metody

Tyto exaktní metody jsou založeny na tom, že za pomocí algoritmu, jenž s dostatečnou přesností popisuje určité děje, zobrazují určitý systém, popřípadě chování

[18] SIXTA, J. a MAČÁT, V. *Logistika - teorie a praxe.*1.vyd. Brno: CP Books, 2005, ISBN 80-251-0573-3
[19] SIXTA, J. a MAČÁT, V. *Logistika - teorie a praxe.*1.vyd. Brno: CP Books, 2005, ISBN 80-251-0573-3

tohoto systému v dostatečně dlouhém časovém období. Zpravidla jsou tyto metody značně složité a neobejdou se bez kvalitní výpočetní techniky.

Základem simulačních metod je to, že umožňují poměrně jednoduchým způsobem využít zobrazení v počítači pro popis poměrně složitých vazeb mezi prvky v systému. Avšak určitou nevýhodou simulace je: specifický postup, který používá simulaci jako umělý statistický experiment, tj. výsledky simulace nejsou stejně přesné, jako výsledky odpovídajícího analytického modelu. [20]

A proto se simulace užívá hlavně v těch případech, kdy je analytické zobrazení daného systému příliš komplikované, nebo dokonce zcela nemožné. V praktickém užití těchto metod lze oba přístupy navzájem kombinovat.

3.3.2.4 Metody teorie grafů

Metody teorie grafů se využívají velmi často pro řešení dopravních systémů ve formě orientovaného nebo neorientovaného grafu. Metod teorie grafů se užívá jednak jako metod optimalizačních, ale i jako metod diagnostických. Jedná se zejména o řešení těchto úloh:

- Úlohy síťové analýzy pro optimalizaci postupů technologických procesů. Zde se užívají například metody CPM, PERT, MPM.
- Problém obchodního cestujícího, čili problém průchodu Hamiltnovskou kružnicí, nebo ještě jinak okružní dopravní systém, a to pro určení optimálního pořadí dopravní obsluhy určených míst.
- Stanovení optimálních toků v sítích pro optimalizaci dopravní obsluhy na základě zvoleného optimalizačního kritéria na předem definované dopravní síti.

Další možností využití grafů pro rozhodování spočívá v používání rozhodovacích stromů jako nástroje pro zobrazení víceetapových rozhodovacích procesů za rizika a nejistoty, zejména v oblasti vrcholového řízení, kde rozhodovací procesy mají koncepční charakter. [21]

3.3.2.5 Prognostické metody

Cílem prognostické činnosti je tvorba prognos, čili odhadů budoucího vývoje nebo budoucích událostí.

3.3.2.6 Metody Operační analýzy

Pod pojmem metody operační analýzy, resp. Operačního výzkumu se rozumí souhrn metod, které pomocí řady matematických disciplin modelují určité stavy technologických procesů nebo procesů rozhodovacích. Z operační analýzy se nejvíce uplatňují metody:

---

[20] ZÍSKAL, J. a HAVLÍČEK, J. *Ekonomicko matematické metody II Studijní texty pro distanční studium.* 2. vyd. Praha: ČZU, 2003, ISBN 80-213-0664-5

[21] ZÍSKAL, J. a HAVLÍČEK, J. *Ekonomicko matematické metody II Studijní texty pro distanční studium.* 2. vyd. Praha: ČZU, 2003, ISBN 80-213-0664-5

- Teorie zásob
- Teorie obnovy
- Teorie front
- Částečně i lineární programování a další

3.4 Využití operační analýzy v dopravní logistice

Metody operační analýzy je možné užít v nejrůznějších článcích logistického řetězce a to jak v hmotných, tak v nehmotných segmentech.

Logistický řetězec je složen s dílčích hmotných, informačních, peněžních a jiných toků, které probíhají mezi různými subsystémy ve výrobě, dopravě, zasilatelství a v obchodě. [22]

Tento řetězec vede od míst prvotních zdrojů surovin přes distribuční a výrobní subjekty až do míst finální spotřeby produktů.

Při prosazování metod operační analýzy v logistice je třeba mít na zřeteli vágnost pojmu „optimální“. Cílem většiny metod operační analýzy aplikovaných v logistice je snížení nákladů ve všech článcích logistického řetězce, které předcházejí konečné spotřebě výrobků u zákazníka. V současném tržním hospodářství se trh prodávajícího stává trhem kupujícího a požadavky na kvalitu zboží, distribučními a prodávajícími organizacemi („náš zákazník – náš pán“). Z toho důvodu je žádoucí dosahovat nikoli minimálních, ale přiměřeně nízkých nákladů, které zajišťují dostatečně pohotové uspokojování potřeb zákazníků. [23]

Dopravní a zasilatelská logistika se řadí mezi časté aplikace hospodářské logistiky. Snaží se synchronizovat, koordinovat a v nemenší míře i optimalizovat pohyby jednotlivých zásilek v dopravní síti a to od místa vstupu do dopravního systému až po místo výstupu ke konečnému spotřebiteli.

V dopravní oblasti bylo vyvinuto množství nejrůznějších modelů, které je dají v dopravní logistice využít a tyto modely zobrazují a zkoumají různé situace, které mohou v dopravním systému nastat. Např. doprava s tranzitem, či model přímé dopravy a nebo v současnosti velice využívaný okružní dopravní model. Také v oblasti manipulace s materiály a výrobky byly vyvinuty různé modely, jejíchž cílem je nalezení optimálního případných časových a kapacitních rezerv, např. model dimenzování skladů na optimální úroveň.

[22] ZÍSKAL, J. a HAVLÍČEK, J. *Ekonomicko matematické metody II Studijní texty pro distanční studium.* 2. vyd. Praha: ČZU, 2003, ISBN 80-213-0664-5

[23] ZÍSKAL, J. a HAVLÍČEK, J. *Ekonomicko matematické metody II Studijní texty pro distanční studium.* 2. vyd. Praha: ČZU, 2003, ISBN 80-213-0664-5

3.5 Okružní dopravní problém

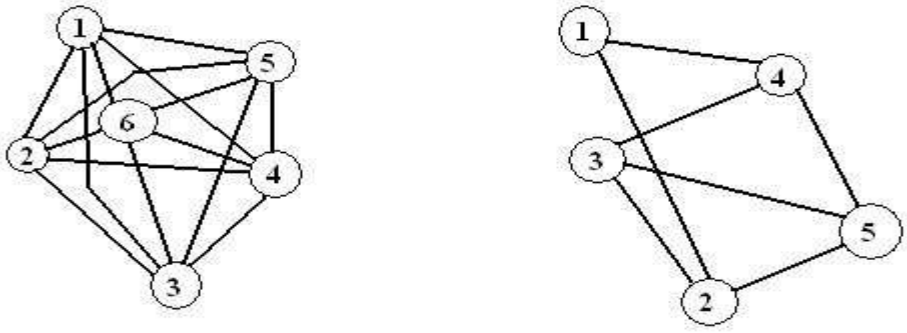
Okružní dopravní problém neboli problém listonoše či ještě jinak problém obchodního cestujícího (anglicky: Traveling Salesman Problem - TSP) je dosti obtížný optimalizační problém dopravní logistiky, matematicky vyjadřující a zobecňující úlohu nalezení nejkratší možné cesty procházející všemi zadanými body na mapě.

Laicky řečeno: Nechť existuje *n* měst, mezi nimiž jsou silnice o známých délkách. Úkolem je najít nejkratší možnou přijatelnou trasu, procházející právě všemi městy a vracející se nazpět do výchozího města.

Matematická formulace používající pojmosloví teorie grafů: Jak v daném ohodnoceném úplném grafu efektivně najít nejkratší Hamiltonovskou kružnici? Hamiltonovská kružnice grafu je taková cesta, která projde právě jednou všechny uzly grafu a mezi jejíž výchozím a konečným uzlem existuje platná cesta.Každý graf nemusí mít nutně Hamiltonovskou kružnici. Nutnými (avšak nikoli postačujícími) podmínkami je, že graf musí být souvislý a každý uzel musí mít stupeň nejméně rovný dvěma (ke každému uzlu musí vést alespoň 2 hrany). [24]

Základní dva typy okružních dopravních problémů se liší charakterem cestní sítě. Na obrázku 3 – 3.5 je znázorněn problém s úplnou cestní sítí (vlevo), ve které existují mezi libovolnými dvěma místy přímé spojení a problém s neúplnou cestní sítí (vpravo), ve kterém nelze realizovat v libovolném směru přímé spojení každé dvojice míst, bez nutného projetí jiným místem.

Obrázek 3 - 3.5 Okružní problém s úplnou a neúplnou cestní sítí



Daný problém netkví ani tak ve stanovení libovolného postupu nalezení nejkratší možné cesty, jeden takový postup je totiž nasnadě: stačí jednoduše prohledat všechny možné uzavřené cesty mezi danými městy a vybrat nejkratší z nich. Obtíž však je, že s rostoucím počtem měst (či uzlů souvislého grafu) počet možných cest velice rychle narůstá, a tím se doba potřebná k propočtu hrubou silou na soudobých počítačích stává zcela neúnosnou už při několika málo desítkách uzlů. Klíčová obtíž je tedy v nalezení časově efektivního algoritmu hledání nejkratších cest.

[24] *Problém obchodního cestujícího - Wikipedie, otevřená encyklopedie*.[on-line]. (6.6.2007) URL: <http://cs.wikipedia.org/wiki/Probl%C3%A9m_obchodn%C3%ADho_cestuj%C3%ADc%C3%Adho>

3.5.1 TSP jako NP-úplný problém

Tato úloha patří mezi tzv. NP-úplné úlohy, tzn. v obecném případě není známo ani jak nalézt přesné řešení v rozumném čase a dokonce ani zda vůbec může existovat algoritmus, který takové řešení najde v čase úměrném nějaké mocnině počtu uzlů. NP-úplné (NP-complete, NPC) problémy jsou takové nedeterministicky polynomiální problémy, na které jsou polynomiálně redukovatelné všechny ostatní problémy z NP. To znamená, že třídu NP-úplných úloh tvoří v jistém smyslu ty nejtěžší úlohy z NP.

Že jde o nedeterministicky polynomiální problém je patrné z toho, že nedeterministický počítač, umožňující v každém kroku rozvětvit výpočet na libovolný počet větví, by mohl začít v některém „městě“, rozdělit propočet délky trasy na tolik větví, kolik z města vede silnic, a v každém z cílových měst postupovat stejně s výjimkou tras vedoucích do již navštívených měst. Tak by prohledal všechny možné trasy v *n* výpočetních krocích, pokud počet měst činí n, a rozvětvil by se maximálně do (*n* - 1)! větví. [25]

V praktickém řešení se podobná úloha obvykle řeší pouze přibližně (heuristickými algoritmy, např. genetickými algoritmy, tabu prohledáváním atd.). Tím se, za cenu vzdání se nalezení přesného řešení dosahuje prakticky použitelných časů.

Pro NP-úplné problémy tedy neexistuje žádný efektivní algoritmus, který by nalezl přesné matematické optimum. Je to způsobeno tím, že počet omezujících podmínek v matematickém modelu této úlohy roste velmi rychle (exponenciálně) s rostoucím počtem míst, a tak doba výpočtu jakoukoli metodou roste při nejlepším stejně rychle a již pro středně velké úlohy by byla nesrovnatelně větší než např. délka lidského života a možná i než doba existence vesmíru. Naštěstí existuje řada aproximačních metod, jejichž řešení lze považovat za ekonomické optimum. [26]

Z výše zmíněného vyplývá, že NP-úplnost úlohy značí velice obtížné nalezení přesného řešení v obecném případě v rozumném čase.

Pokud bychom měli k dispozici hypotetický nedeterministický počítač, který v každém kroku rozvětví úlohu do tolika větví, kolik je cest z daného města, získáme řešení v čase *n* [27] (pro případ okružní úlohy značí *n* počet měst).

Známé algoritmy pro přesné řešení této obecné úlohy na existujících (deterministických) počítačích mají složitost odpovídající faktoriálu počtu měst, což už pro relativně malé úlohy (stovky měst) znamená nepoužitelně obrovské časové nároky. V praxi se podobná úloha obvykle řeší pouze přibližně (heuristickými algoritmy, např. genetickými algoritmy, tabu prohledávání atd.). Tím se (za cenu vzdání se nároku na nalezení přesného řešení) dosahuje prakticky použitelných časů.[28]

Mezi typické NP-úplné úlohy patří kromě okružní dopravní úlohy také problém SAT (satisfiability problem – problém splnitelnosti logických formulí), hledání hamiltonovské kružnice, hledání nezávislé množiny, problém kliky (hledání úplného

[25] *Problém obchodního cestujícího - Wikipedie, otevřená encyklopedie*.[on-line]. (6.6.2007)URL: <http://cs.wikipedia.org/wiki/Probl%C3%A9m_obchodn%C3%ADho_cestuj%C3%ADc%C3%Adho>
[26] ŠUBRT, T., BROŽOVÁ, H., DÖMEOVÁ, L., KUČERA, P. *Ekonomicko matematické metody II*, aplikace a cvičení, skripta PEF ČZU Praha, 2000
[27] *Wikipedie, otevřená encyklopedie.* [on-line].(10.6.2008)URL:<*http*://cs.wikipedia.org>
[28] *Wikipedie, otevřená encyklopedie.* [on-line].(10.6.2008)URL:<*http*://cs.wikipedia.org>

podgrafu), faktorizace na prvočísla, hledání isomorfního podgrafu, 3-barevnost grafu, vrcholové pokrytí, zavazadlový problém (tzv. problém batohu), problém dvou loupežníků (zda danou množinu přirozených čísel lze rozdělit na dvě části tak, aby se jejich součty rovnaly) atd.[29]

3.5.2 Zadání problému TSP

Jak bylo již tedy řečeno Okružní dopravní úloha se zabývá rozvozem popřípadě svozem zboží či materiálů po okružní trase, kterou se snaží určit a její co možná nejkratší trasu neboli:

Okružní cesty mohou zajišťovat buď rozvoz zboží, nebo svoz zboží, případně současně rozvoz i svoz. Rozvozy se týkají zásobování prodejen výrobci nebo velkoobchodem, svozy se týkají svozu zemědělských produktů (např. mléka do mlékáren) nebo odpadků na skládky, svozy a rozvozy se týkají zásobování se souběžným sběrem obalů (rozvoz nápojů se sběrem prázdných lahví, rozvoz zakázek se současným sběrem nových zakázek – opravy, fotoslužby apod.).[30]

Z toho můžeme usuzovat, že představují požadavek na racionální dopravu (při minimalizaci přepravních nákladů) stejnorodého produktu od dodavatele odběratelům se známými požadavky při daných nákladech (nebo vzdálenostech) mezi dodavatelským a odběratelskými místy.[31]

Okružní úlohy jsou tedy úlohy hledání optimální trasy pro vozidla realizující přepravu okružním způsobem, kdy uzly distribuční sítě jsou obsluhovány (zásobovány apod.) během jedné jízdy, nebo několika jízd, které většinou začínají a končí ve výchozím uzlu sítě. Účelem je minimalizovat celkovou délku tras vozidel s respektováním požadavků odběratelů i technologických omezení dopravních prostředků.[32]

Nejdůležitějším kritériem posuzující výsledek okružní dopravní úlohy nemusí být vždy pouze celková vzdálenost trasy (kilometrová vzdálenost), ale mohou to být i jiná kritéria např.: spotřeba pohonných hmot, zatížení životního prostředí, či i přepravní náklady celkové, popř. jen dopravní náklady. Dnes je významnou složkou dopravních nákladů i položka nákladů na mýtné, které může hrát též významnou roli jak si později ukážeme v případové studii (kapitola 4.2).

Hledání optimálních okružních tras většinou vychází z určité komunikační sítě (například silniční sítě se zadanými kilometrovými vzdálenostmi), u sítě uvnitř městské zástavby z plánu ulic a dopravních omezení (zákazy vjezdu, jednosměrný provoz…). Pokud nejsou k dispozici vzdálenosti mezi komunikačními uzly, je možné tuto vzdálenost přibližně určit na základě euklidovské vzdálenosti bodů na mapě (daných souřadnicemi) nebo na základě „obdélníkové“ vzdálenosti v případě městské sítě, kde vzdálenost musí respektovat většinou pravoúhle se křížící městské komunikace.[33]

---

[29] *Wikipedie, otevřená encyklopedie*[on-line].(10.6.2008)URL:<*http*://cs.wikipedia.org>
[30] PELIKÁN, J. *Praktikum z operačního výzkumu*, Praha: VŠE, 1992
[31] ZÍSKAL, J. a HAVLÍČEK, J. *Ekonomicko matematické metody II Studijní texty pro distanční studium.* 2. vyd. Praha: ČZU, 2003, ISBN 80-213-0664-5
[32] PELIKÁN, J. *Praktikum z operačního výzkumu*, Praha: VŠE, 1992
[33] PELIKÁN, J. *Praktikum z operačního výzkumu*, Praha: VŠE, 1992

Pro zadání a matematický model TSP užijeme jednoduché definice okružní dopravní úlohy:

Existuje n měst a mezi nimi silnice o známých délkách. Úkolem je najít pro cestujícího cestu, procházející všemi městy, která je kratší než nějaká hodnota, existuje-li (popř. konstatovat, že taková neexistuje).[34]

Výše zmíněná velmi obecná definice může být rozšířena a tím se dostaneme k tomu, že je dáno n míst (či měst) a sazba c_{ij} pro každou dvojici těchto měst (i, j) představující např. vzdálenost, spotřebu času nebo náklady pro přímé (či nejvýhodnější) spojení z místa i do místo j . Cílem úlohy je propojit všechna místa okružním spojením, tj. najít takovou posloupnost těchto míst, ve které se každé z nich vyskytuje právě jednou s výjimkou počátečního, které se objeví opět na jejím konci, aby součet sazeb pro jednotlivá spojení v této posloupnosti byl minimální.[35]

3.5.2.1 Matematický model TSP

Matematický model těchto úloh je graf $G= \{V,E\}$, ve kterém uzly představují místa, kam se zboží rozváží a hrany představující komunikační síť (silnice, železnice). V této úloze se hledá uzavřená cesta (cyklus), která obsahuje všechny uzly grafu. V grafu G jsou všechny hrany ohodnoceny, hrana (i, j) je ohodnocena číslem c_{ij} , v praxi představuje náklady (příp. dobu přejezdu, kilometrovou vzdálenost, …) spojené s přejezdem z uzlu i do uzlu j . Celkové ocenění uzavřené cesty v grafu G je součet ocenění hran ležících na této cestě.

Úkolem je hledání takové uzavřené cesty v grafu, obsahující všechny uzly právě jednou a jejíž celkové ohodnocení je nejnižší. Hledaný cyklus (Hamiltonův cyklus) je složen z hran z množiny E . Zavedeme tedy bivalentní proměnné x_{ij} tak, že $x_{ij} = 1$, jestliže hrana (i, j) leží na hledaném cyklu, v opačném případě je $x_{ij} = 0$. Mějme $V=\{1, 2,...,n\}$, pak podmínku, že uzel i leží na tomto cyklu, lze formulovat tak, že právě jedna hrana z cyklu končí v uzlu i a právě jedna hrana z cyklu z uzlu i vychází. Tyto dvě podmínky lze zapsat ve tvaru:

$$\sum_{j=1}^{n} x_{ij} = 1$$

(Vzorec 3.1)

$$\sum_{j=1}^{n} x_{ji} = 1$$

(Vzorec 3.2)

[34] *Wikipedie, otevřená encyklopedie*[on-line].(10.1.2009)URL:<*http*://cs.wikipedia.org>
[35] ŠUBRT, T., BROŽOVÁ, H., DÖMEOVÁ, L., KUČERA, P.*Ekonomicko matematické metody II*, aplikace a cvičení, skripta PEF ČZU Praha, 2000

Ocenění tohoto cyklu je součet ocenění hran, tedy hran, pro které je $x_{ij} = 1$. Toto ocenění lze zapsat ve tvaru:

$$Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

(Vzorec 3.3)

Tuto účelovou funkci budeme minimalizovat.

Lze ukázat, že popsaná soustava rovnic nezaručuje, že její řešení je skutečně cyklus (Hamiltonův). Mějme například:

$$n = 6$$

(Vzorec 3.4)

$$x_{12} = x_{23} = x_{31} = 1$$

(Vzorec 3.5)

$$x_{45} = x_{56} = x_{64} = 1$$

(Vzorec 3.6)

Toto řešení obsahuje hrany (1, 2), (2,3) a (3,1) tvořící cyklus obsahující uzly 1, 2 a 3 a hrany (4,5), (5,6) a (6, 4), které tvoří cyklus přes uzly 4 , 5 a 6 . Představuje tedy dva částečné (parciální) cykly, což není řešení, jež hledáme.

Je tedy třeba k uvedeným omezením připojit další omezení, která tyto parciální cykly vyloučí, tzv. smyčkové podmínky. Vezměme si libovolnou podmnožinu uzlů:

$$U \subset V$$

(Vzorec 3.7)

$$2 \leq |U| < n - 2$$

(Vzorec 3.8)

Smyčkové podmínky musí vyloučit taková řešení, která by představovala cyklus v množině uzlů U , tedy takzvaný parciální cyklus. Protože cyklus v množině U by musel obsahovat U hran, můžeme toto omezení formulovat jako:

$$\sum_{(i,j) \in U} x_{ij} \leq |U| - 1$$

(Vzorec 3.9)

platící pro vzorec 3.7. a 3.8. Počet těchto omezení je značný, řádově $2^{|V|}$.

Zbývá ještě ukázat, že smyčkové podmínky splňuje každé řešení, které představuje cyklus přes všech n uzlů (Hamiltonův cyklus). Mějme tedy řešení $X = \{x_{ij}\}$, které představuje Hamiltonův cyklus. Položme $u_i = t$, kde t je pořadové číslo uzlu i na cyklu, začneme-li uzlem u_1 jako prvním.

Jestliže hrana (i, j) leží na cyklu, tj. $x_{ij} = 1$, pak uzel j přímo následuje po uzlu i podél cyklu, proto je-li $u_i = t$, pak $u_j = t + 1$. Potom:

$$u_i - u_j + nx_{ij} = t - (t+1) + n = n - 1$$

(Vzorec 3.10)

což znamená, že pro libovolnou hranu (i, j) z Hamiltonova cyklu smyčková podmínka platí (a je splněna jako rovnice).

Pokud hrana (i, j) neleží na cyklu ($x_{ij} = 0$), pak platí:

$$u_1 - u_j - nx_{ij} = u_i - u_j$$

(Vzorec 3.11)

Protože platí:

$$u_i \in \{1, 2, \ldots, n\}$$

(Vzorec 3.12)

$$u_j \in \{1, 2, \ldots, n\}$$

(Vzorec 3.13)

proto je rozdíl

$$u_i - u_j \leq n - 1$$

(Vzorec 3.14)

a smyčková podmínka je tedy splněna.[36]

Při bližším pohledu na matematický model úlohy obchodního cestujícího můžeme říci, že v matematické formulaci základní okružní úlohy je dána konečná množina míst a vzdálenosti, spotřeba času nebo náklady, sazby pro spojení každé dvojice těchto míst. Hledáme tedy takovou posloupnost míst, ve které se každé místo objeví právě jednou a součet ohodnocení jednotlivých spojení v této posloupnosti je minimální.

Označíme-li vybranou posloupnost m míst indexy $i_1, i_2, \ldots i_m$, můžeme hodnotu tohoto spojení vypočítat jako součet sazeb (vzdálenosti):

$$\sum_{k=1}^{m=1} c_{ik,ik+1} + c_{im,i_1}$$

(Vzorec 3.15)

Požadavek, aby se každé místo objevilo ve vybrané trase jen jednou, nelze chápat tak, že se každým místem projíždí pouze jednou, neboť nemusí vždy existovat unikátní spojeni mezi každou dvojicí míst a je třeba do trasy zařadit odbočky a koncová místa.

[36] PELIKÁN, J. *Diskrétní modely v operačním výzkumu*, Professional Publishing, 2001

Pro úplnost uvedeme ještě Tuckerovu formulaci problému obchodního cestujícího. Obchodní cestující má navštívit *n* míst. Vzdálenost mezi *i*-tým a *j*-tým místem označíme symbolem d_{ij} . Celkovou délku okružní cesty, která se má minimalizovat lze vyjádřit vztahem:

$$Z = \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} \cdot x_{ij}$$

(Vzorec 3.16)

kde x_{ij} je počet jízd z místa *i* do místa *j* .

Protože na okružní cestě navštíví obchodní cestující každé místo jednou, musí platit podmínky omezení:

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,2,\ldots,n$$

(Vzorec 3.17)

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,2,\ldots,n$$

(Vzorec 3.18)

které nestačí k přesné formulaci problému, neboť je možné je splnit také tak, že se jednotlivá místa objedou po několika samostatných okruzích. Aby se vyloučila možnost dalších okruhů, formuloval Tucker další omezení:

$$u_i - u_j + n x_{ij} \le n - 1, \quad (i \ne j, \quad i,j = 1,2,\ldots,n)$$

(Vzorec 3.19)

kde u_i a u_j je neznámé reálné číslo přiřazené místu *i* , resp. místu *j*.[37]

[37] ZÍSKAL, J. a HAVLÍČEK, J. *Ekonomicko matematické metody II Studijní texty pro distanční studium.* 2. vyd. Praha: ČZU, 2003, ISBN 80-213-0664-5

Po shrnutí můžeme matematický model úlohy obchodního cestujícího tedy formulovat takto:

$$Z_{(\min)} = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{pro } i = 1,2,\ldots,n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{pro } j = 1,2,\ldots,n$$

$$u_i - u_j + nx_{ij} \le n-1, \quad i = 1,2,\ldots,n, \quad j = 2,3,\ldots,n,$$

$$i \ne j, \quad x_{ij} \in \{0,1\}, \quad i,j = 1,2,\ldots,n$$

(Vzorec 3.20)

Smyčkové podmínky 3.19 lze zapsat též ve tvaru:

$$u_1 + 1 + n\left(x_{ij} - 1\right) \le u_j$$

(Vzorec 3.21)

Pro hodnotu $x_{ij} = 1$ tato nerovnost představuje nerovnost:

$$u_1 + 1 \le u_j$$

(Vzorec 3.22)

pro $x_{ij} = 0$ pak nerovnost:

$$u_i - (n-1) \le u_j$$

(Vzorec 3.23)

neboli:

$$u_i - u_j \le n-1$$

(Vzorec 3.24)

První nerovnost (vzorec 3.22) znamená, že hodnoty u_i podél okruhu budou růst alespoň o 1. Druhá nerovnost (vzorec 3.23 nebo vzorec 3.24) určuje, že rozdíl nejvyšší a nejnižší hodnoty čísel u_i bude nejvýše $n - 1$, tedy obě zajišťují to, že čísla u_i rostou podél okruhu právě o hodnotu 1. Pokud ještě připojíme další podmínku ve tvaru:

$$1 \le u_1 \le n-1, \quad i = 2,3,\ldots,n$$

(Vzorec 3.25)

pak u_i skutečně budou pořadová čísla uzlů na Hamiltonově cyklu.[38]

[38] PELIKÁN, J. *Diskrétní modely v operačním výzkumu*, Professional Publishing, 2001

3.5.3 Možnosti řešení okružního dopravního problému

Řešení okružního dopravního problému můžeme provádět za pomoci více metod, jejichž principem je vytvoření a zpracování posloupností sledovaných míst, u nichž se musí každé místo objevovat právě jednou. Pro zamezení předčasného uzavření kruhu, je zapotřebí vyloučit všechny trasy, která by předčasně tento kruh uzavřely. Velmi důležité je také vyloučit současné zařazení jednoho úseku oběma směry a zpětnou vazbu každého uzlu.

3.5.4 Aproximační metody

3.5.4.1 Mayerova metoda

Mayerova metoda se používá u víceokruhových dopravních modelů a to pro roztřídění jednotlivých míst do jednotlivých okruhů přepravy resp. jednotlivých tras. Algoritmus Mayerovy metody spočívá v tom, že:

1. V tabulce sazeb si seřadíme místa v řádcích i sloupcích podle vzdálenosti místa centrálního svozu (rozvozu), které samotné můžeme v tabulce vynechat a přidáme sloupec obsahující požadavky jednotlivých míst.
2. Označíme první sloupec této tabulky (tj. první místo vybereme do první okružní trasy) a požadavek v prvním řádku a vyškrtneme první řádek.
3. Pro každé z ostatních míst sečteme jeho přepravní požadavek s označeným a u všech míst, kde tento součet bude větší než kapacita vozidla, vyškrtneme v prvním sloupci buňku v příslušném řádku.
4. Z nevyškrtnutých prvků v prvním sloupci vybereme minimální, není-li výběr jednoznačný, pak zvolíme první takový prvek v pořadí (nejhořejší). Ten označuje místo, které jako další přiřazujeme do právě konstruované okružní trasy.
5. Odpovídající sloupec a požadavek v odpovídajícím řádku označíme nebo nějak zvýrazníme a řádek vyškrtneme.
6. Sečteme vyznačené požadavky a pro ta místa, kde přičtením jejich požadavku k uvedenému součtu je překročena kapacita vozidla, opět vyškrtneme v označených sloupcích buňky v odpovídajících řádcích.
7. Z nevyškrtnutých prvků v označených sloupcích stejným způsobem vybereme minimální prvek a tím i další místo okružní trasy.
8. Celý postup opakujeme, dokud při porovnávání kapacit nevyškrtáme všechny sazby v označených sloupcích. Tím jsme vybrali místa pro první okružní trasu.
9. Tato místa si poznamenáme, vyškrtáme příslušné sloupce a požadavky a ve zbylé části tabulky hledáme stejným způsobem místa do dalších okružních tras.

10. V jednotlivých okruzích místa ještě seřadíme pomocí metod pro jednookruhové modely. [39]

3.5.4.2 Vogelova metoda

Upravená Vogelova metoda je vhodná pro jednookruhové dopravní modely. Při použití Vogelovy metody pro řešení okružního dopravního problému postupujeme tako:

1. Před zahájením výpočtu se zapíší do tabulky sazební ohodnocení jednotlivých cest.
2. V každé řadě (řádku i sloupci) vypočítáme diferenci mezi dvěma nejmenšími sazbami (jelikož se jedná o minimalizační úlohu). Vyskytnou-li se dvě nebo i více stejně velkých nejvýhodnějších sazeb, vybereme jen jednu a odečteme ji od další nejvýhodnější sazby (tj. u minimalizace nejbližší vyšší sazby).
3. V řadě s nejvyšší diferencí označíme buňku s nejmenší sazbou, což znamená, že spojení odpovídající této buňce je zařazováno do konstruované trasy obchodního cestujícího.
4. Dále se vyškrtává řádek i sloupec, ve kterých se obsazovaná buňka nachází (obchodní cestující jede z i do každého místa jen jednou), a kromě toho je třeba vyškrtnout ještě jednu další buňku, která s právě obsazenou buňkou a případně ještě několika již dříve obsazenými uzavírá kruh, který neprochází všemi místy.
5. Po tomto vyškrtání je třeba opět přepočítat řádkové i sloupcové diference.
6. Postup 3. až 5. opakujeme a průběžně zaznamenáváme spojení trasy obchodního cestujícího, dokud nedosáhneme konečného řešení. [40]

3.5.4.3 Metoda nejbližšího souseda

Princip této metody spočívá v tom, že si zvolíme výchozí místo, z něj se vydáme do místa, do něhož je nejvýhodnější spojení z výchozího místa, odtud pak do dalšího z těch míst, kde jsme ještě nebyli, které má nejvýhodnější spojení z místa, kde se právě nacházíme atd. Po projetí všech míst se vracíme zpět do výchozího.

Postup této metody se skládá z několika kroků:

1. Ve výchozí tabulce vyškrtneme sloupec odpovídající výchozímu místu (do tohoto místa totiž prozatím nepojedeme, vrátíme se tam až nakonec).
2. V řádku odpovídajícím výchozímu místu najdeme buňku s minimální a tudíž pro nás nejvýhodnější sazbou a označíme ji, tj. příslušné spojení bude součástí výsledné okružní trasy.

[39] ŠUBRT, T. a kol. *Ekonomicko matematické metody II Aplikace a cvičení*. 2. vyd. Praha: ČZU, 2005, ISBN 80-213-0721-8
[40] ŠUBRT, T. a kol. *Ekonomicko matematické metody II Aplikace a cvičení*. 2. vyd. Praha: ČZU, 2005, ISBN 80-213-0721-8

3. Tímto spojením jsme se přesunuli do místa, jemuž odpovídá sloupec, v němž se tato buňka nachází. Tento sloupec vyškrtneme (do tohoto místa se již nebudeme více vracet).
4. V řádku odpovídajícím tomuto místu vybereme z buněk v dosud nevyškrtnutých sloupcích opět tu s nejvýhodnější sazbou.
5. Celý postup opakujeme, dokud nejsou všechny sloupce vyškrtány (tj. dokud jsme nenavštívili všechna místa).
6. V řádku, v němž jsme se ocitli nakonec, obsadíme buňku ve sloupci odpovídajícím výchozímu místu.
7. Postupně zvolíme všechna místa jako výchozí a pro každé najdeme tímto postupem okružní trasu. Má-li úloha nesymetrickou matici sazeb, provedeme pro každé místo také hledání trasy „pozpátku“, tj. buď vyškrtáme řádky a hledáme minimální sazby ve sloupcích nebo původní postup aplikujeme na transformovanou matici. Ze všech takto nalezených tras vybereme nejvýhodnější (s nejmenším součtem sazeb). [41]

3.5.5 Heuristické metody

Heuristické metody lze rozdělit na metody vytvářející řešení a metody řešení zlepšující. Jde o metody polynomiální, které obecně nedávají optimální řešení. Metody vytvářející řešení postupují buď sekvenčním, nebo paralelním postupem, to znamená, že u sekvenčního postupu trasu vytvářejí tak, že z výchozího uzlu hledají postupně další uzly na trase, paralelní postup propojuje postupně uzly do částečných cest a ty se nakonec spojí ve výsledný cyklus. Sekvenční postup je rychlejší, jednodušší, výsledné řešení ale nemusí být tak dobré, zejména v poslední části trasy. Paralelní postup je výpočetně obtížnější, avšak získané řešení může být lepší než řešení získané sekvenčním postupem.[42]

3.5.6 Přehled dalších metod

Existuje celá řada dalších metod pro řešení okružního dopravního problému, v této práci budou užity jen Mayerova metoda, Vogelova aproximační metoda a metoda nejbližšího souseda, u nichž byl již i uveden podrobný algoritmus výpočtu.

Pro přehled si ale můžeme ještě uvést nepříklad:

Dantzigova, Fulkersonova a Johnsonova metoda převádí řešení okružního problému na úlohu celočíselného programování řešenou simplexovou metodou. Postup je značně komplikovaný. V podstatě využívá přiřazovací problém s maximální degenerací. Croesova metoda řeší problém postupným zlepšováním počátečního řešení určitými změnami v pořadí vrcholů tak dlouho, dokud je to možné. Nalezené řešení ovšem obecně není optimální. K nalezení optima se používá dalšího značně složitého postupu. Barták s kolektivem pracovníků vyvinuli pro různé typy okružních úloh

[41] ŠUBRT, T. a kol. *Ekonomicko matematické metody II Aplikace a cvičení*. 2. vyd. Praha: ČZU, 2005, ISBN 80-213-0721-8
[42] PELIKÁN, J. *Praktikum z operačního výzkumu*, skripta VŠE Praha, 1992

kombinatorickou metodu, přičemž pro testování nalezeného řešení používají maďarskou metodu.[43]

Metoda větví a hranic (označována též jako Littlova metoda) je v současné době jednou z nejefektivnějších optimalizačních metod. Její podstatou je rozdělení oblasti přípustných řešení (to jsou taková řešení, která vyhovují omezujícím podmínkám úlohy – v našem případě jsou přípustnými řešeními všechny Hamiltonovy cykly) na několik částí, obvykle na dvě. Pokud se jedná o úlohu minimalizační, pak nalezneme na každé z těchto částí minorantu, což je taková funkce, která je na celé části množiny přípustných řešení menší než původní cílová funkce úlohy. U nesouměrně zadané matice hodnot dále větvíme tu část množiny přípustných řešení, která má menší minorantu (předpokládáme, že je více pravděpodobné, že hledané optimální řešení je v této části). Odtud tedy část názvu metody – metoda větví.[44]

3.6 Microsoft Excel

Microsoft Excel je součástí skupiny programů zvané tabulkové procesory.

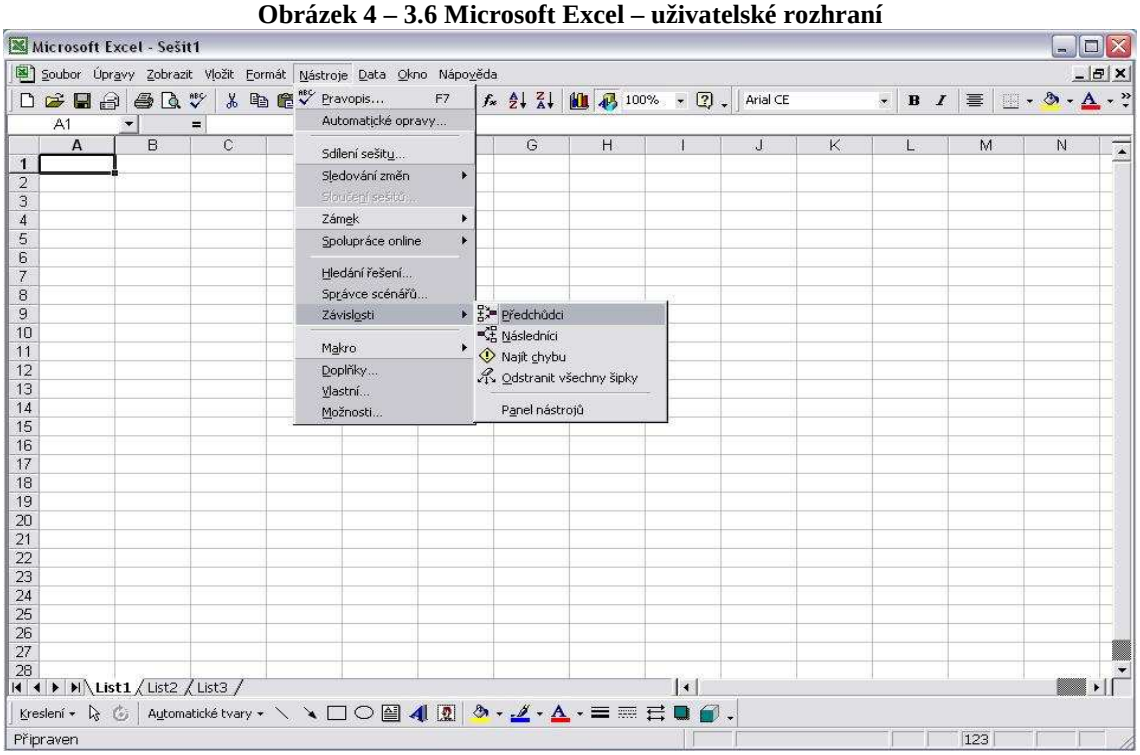
Tabulkový procesor je program zpracovávající tabulku informací (je to vlastně matice). V jednotlivých buňkách mohou být uložena data či vzorce počítající s těmi daty. V tom případě se v tabulce zobrazují data vypočtená ze vzorců. Dnes jsou hojně integrovány do kancelářských balíků. Zprvu byl tabulkový procesor využíván zejména ve finančnictví, proto byly první verze vybaveny zejména funkcemi vhodnými na finanční výpočty, dnes je však využíván k širokému množství výpočtů a jiných zpracování dat.[45]

Výhodou MS Excelu byla oproti ostatním tabulkovým procesorům orientace na grafické uživatelské rozhraní, zejména častá práce s myší. Uživatelské rozhraní je prostředek, kterým uživatel komunikuje s počítačovým programem. Nabídky v MS Excelu jsou svou podstatou různé panely nástrojů, a to od základního panelu nástrojů, jenž můžeme vidět na obrázku 4 – 3.6 Microsoft Excel – uživatelské rozhraní zcela nahoře až po panely nejrůznějších nabídek následované za základním panelem. Některé z položek nabídek jsou tzv. kaskádové nabídky, výsledkem je otevření další nabídky, která obsahuje některé další příkazy, kupříkladu nabídka Nástroje jak lze vidět na obrázku níže. Celý systém nabídek může být přizpůsoben dle nejrůznějších potřeb konečného uživatele popřípadě programátora.

Postupem času se MS Excel stal dominantním hráčem na trhu tabulkových procesorů pro běžné uživatele. Své postavení leadera v oblasti upevňoval zejména pravidelným uvolňováním nových verzí.

[43] ZÍSKAL, J. a HAVLÍČEK, J. *Ekonomicko matematické metody II Studijní texty pro distanční studium.* 2. vyd. Praha: ČZU, 2003, ISBN 80-213-0664-5
[44] BERKA, M. *Operační výzkum*, Brno: VUT, 1991
[45] *Wikipedie, otevřená encyklopedie*[on-line].(10.1.2009)URL:<*http*://cs.wikipedia.org>

Obrázek 4 – 3.6 Microsoft Excel – uživatelské rozhraní



První verze Excelu, která se „vyvinula“ z MultiPlanu (první nepříliš podařený tabulkový procesor Microsoftu z roku 1982) spatřila světlo světa v roce 1985 na počítačích Macintosh. V listopadu 1987 Microsoft oznámil distribuci první verze Excelu pro Windows.[46]

Současná verze Excel 2003 je celkem dobře programovatelný produkt, a proto je považována pro vývojáře tabulkových aplikací zcela vhodnou volbou. Používá jazyk Visual Basic for Applications (VBA), který je v dnešní době často používán v nejrůznějších aplikacích.

Z pohledu vývojáře mezi klíčové rysy Excelu patří následující:

- Visual Basic for Applications. Tento makro-jazyk umožňuje vytvářet strukturované programy přímo v Excelu.
- Vlastní dialogy. Snadné vytvoření vlastních profesionálně vypadajících dialogů. Technologie Excelu s názvem UserForms (byla představena v Excelu 97) představuje obrovský pokrok při tvorbě vlastních dialogů v porovnání se staršími dialogovými listy.
- Široké možnosti zabezpečení. Aplikace je možno chránit proti zvídavým očím a lze je zabezpečit i proti nežádoucím změnám.

[46] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4

- Možnost vytváření „komplikovaných“ doplňků. Jediným příkazem lze vytvořit soubory XLA obsahující doplňky, které se pak připojují do uživatelského rozhraní Excelu. [47]

Jedním z cílů této diplomové práce je naprogramovat modul pro řešení okružních dopravních úloh. Nejedná se však o programování v pravém slova smyslu jako v klasických programovacích jazycích, jako např.: C nebo Pascal, ale jde o proces vytváření aplikací, které pak následně pracují s tabulkami. Společné je však to, že aplikace budou používat další koncoví uživatelé, nikoli jen programátor samotný, proto musí být koncovému uživateli příjemná a snadno srozumitelná.

Dobrá tabulková aplikace se vyznačuje následujícími charakteristikami:

- Umožňuje koncovému uživateli provádět ty činnosti, které by jinak nebyl schopen udělat.
- Nabízí vhodné řešení daného problému (prostředí tabulkového procesoru nenabízí vždy vhodné podmínky.)
- Dělá opravdu to, co se od ní očekává. Vypadá to sice jako samozřejmost, ale divili byste se, kolik aplikací tímto testem neprošlo.
- Jejím výstupem jsou přesné výsledky a neobsahuje žádné chyby.
- Používá pro svoji činnost vhodné a efektivní metody a algoritmy.
- Zachytává chyby ještě předtím, než se uživatel má s nimi vůbec možnost setkat „tváří v tvář
- Nedovolí uživateli nechtěně (nebo záměrně) smazat důležité součásti aplikace.
- Uživatelské rozhraní je přehledné a konzistentní, takže uživatel vždy ví, jak má pokračovat dál.
- Vzorce, makra a prvky uživatelského rozhraní jsou dobře dokumentovány, takže v případě potřeby je možné je dále upravit.
- Je navržena tak, aby mohla být jednoduše upravena bez nutnosti velkých změn. Změna potřeb uživatele po určitém čase je přirozenou součástí života.
- Má snadno dostupný systém nápovědy, který poskytuje užitečné informace a popisuje alespoň hlavní postupy při práci.
- Je navržena tak, aby byla přenositelná a mohla pracovat na libovolném systému, který má nainstalovaný odpovídající software (v tomto případě je to Excel). [48]

[47] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4

[48] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4

3.7 Visual Basic for Applications

Programovací jazyk Visual Basic for Applications (VBA) je vytvořený společností Microsoft. Pracuje v operačním prostředí Windows.

VBA se dá nejlépe popsat jako obecný skriptovací jazyk Microsoftu, který je dnes již součástí všech aplikací Office od Microsoftu (balíček aplikací Excel, Word, Access, PowerPoint, apod.), a dokonce i aplikací od jiných dodavatelů.[49]

VBA patří do rodiny programovacích jazyků BASIC. Jsou to „programovací jazyky vysoké úrovně, které byly zavedeny jako jednoduchý nástroj pro výuku programování. K jednoduchosti přispívalo i to, že klíčová slova jazyka vychází z běžné angličtiny. Jazyk navrhli v roce 1963 John G. Kemeny a Thomas E. Kurtz z Dartmouthské univerzity (Hanover, New Hampshire). Název BASIC je zkratkou anglických slov Beginner's All-purpose Symbolic Instruction Code.[50]

Jako první aplikace, která používala jazyk Visual Basic for Applications, byl Excel 5. Před nástupem verze 5 Excel používal výkonný (ale velmi záhadný) makrojazyk XLM. Novější verze Excelu umí stále makra XLM spouštět, ovšem schopnost záznamu maker v jazyce XLM byla počínajíc Excelem 97 odstraněna. Vývojáři by měli makra XLM vést v patrnosti (pro případ, že se s nimi setkají), ale pro nová řešení by měli používat výhradně VBA.[51]

VBA je programovacím jazykem objektově orientovaným. V takto orientovaném jazyce je každý aspekt počítačového kódu založen na prvcích uživatelského rozhraní. Prvky jsou tu prezentovány objekty a v nich jsou pak obsaženy data a všechny činnosti. VBA vlastně tedy jen manipuluje s objekty a všechny produkty (jako např.: Excel, Word, Access, Power Point a další) mají svůj jedinečný objektový model. V aplikaci se pak tedy programuje pomocí jejich objektů.

Objektový model Excelu má například několik velmi výkonných objektů pro zpracování a analýzu dat. Jde o objekty tabulek, grafů, kontingenčních tabulek a mnoho numerických, matematických, finančních a inženýrských i zcela obecných ekonomických funkcí. Pomocí VBA lze pak s těmito objekty libovolně pracovat a popřípadě i vytvářet automatizované procedury.

Samotná procedura Sub je ve své podstatě kus počítačového kódu, která provádí určité předem definované činnosti. Je umístěna v modulu VBA, který ale může obsahovat libovolný počet procedur.

Avšak část programátorů se odmítá zabývat psaním značně dlouhých procedur, které mají provádět množství nejrůznějších operací. Tvoří kratší procedury, které je pak možné postupně volat z určité nadřazené procedury. Tento jednodušší a přehlednější postup byl využit i v této práci. Hlavním důvodem je jasná přehlednost a pro jednoduchost kódu i snazší úprava kódu i jeho následné udržování.

---

[49] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4

[50] *Wikipedie, otevřená encyklopedie*[on-line].(10.6.2008)URL:<*http*://cs.wikipedia.org>

[51] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-44

Některé procedury jsou napsané tak, že očekávají určité parametry. Parametr je informace, kterou procedura používá. Parametry se do procedury „předávají“ při jejím volání. Příkazy uvnitř procedur zpravidla provádějí různé logické operace a výsledky procedury jsou zpravidla závislé na hodnotách parametrů. Kromě procedur Sub může modul VBA rovněž obsahovat funkce (deklarované klíčovým slovem Function). Funkce na rozdíl od procedury Sub vrací jednu hodnotu (případně pole proměnných). Funkce může být volána z jiné procedury VBA nebo použita ve vzorci na pracovním listu. VBA manipuluje s objekty, které obsahuje hostitelská aplikace (v tomto případě Excel). Excel nabízí více než 100 tříd objektů, se kterými je možné manipulovat. Za všechny objekty jako příklad uveďme sešit, pracovní list, oblast v listu, graf, vykreslený obdélník. K dispozici je spousta dalších objektů, se kterými můžete prostřednictvím VBA manipulovat. Objekty mohou vstupovat jako kontejnery pro jiné objekty. Třídy objektů jsou uspořádány v určité hierarchii.[52]

Struktura objektů tedy zohledňuje určitou nadřazenost a podřazenost objektů. Množina objektů s určitými shodnými charakteristikami tvoří třídu. Třídu lze nadefinovat a poté lze vytvářet i jednotlivé instance této třídy, jedná se vlastně o skutečné objekty. Odkazy na tyto objekty lze pak v programu ukládat do proměnných. Vynechá-li se odkaz na určitý objekt, použije Excel objekt aktivní. Obecně existují čtyři aspekty objektů, za jejichž pomoci lze vytvořit program. Jedná se o vlastnosti, metody, události a kolekce.

Na vlastnost objektu lze pohlížet jako na určité nastavení daného objektu.

Vlastnosti jsou proměnné popisující některé aspekty objektu, ve kterém jsou obsaženy. Běžnou vlastností objektů v aplikaci Excel je Name (Název), který nese hodnotu identifikace přiřazenou uživatelem nebo aplikací Excel sešitu, listu, rozsahu buněk nebo jinému objektu. Pokud se například změní název listu (pomocí kódu VBA nebo klepnutím pravým tlačítkem myši na záložku listu), změní se hodnota uložená ve vlastnosti Name. V jazyce VBA jsou vlastnosti v programu uvedeny zápisem s tečkou, kde název objektu je uveden jako první, název vlastnosti jako druhý a tyto dva prvky jsou odděleny tečkou. Například pokud je třeba změnit název sešitu, je nutno použít vlastnost Worksheet.Name.[53]

Metoda představuje nějakou akci, která je na objektu provedena. To znamená, že metoda je akce, jejíž postup provedení objekt „zná“ a umí ji provést. Metoda je tedy vlastním objektem vyvolána při příjmu odpovídající zprávy. Metody jsou podobně jako vlastnosti volány pomocí zápisu s tečkou. Metoda je v kódu zapsána jako kombinace názvu objektu a názvu metody, které jsou navzájem oddělené tečkou.[54]

Událost je tedy akce, jejíž spuštění a průběh daný objekt rozpozná a kód tak může proběhnout na základě určité akce (např. klepnutí myší, opuštění objektu, aj.).

[52] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4
[53] FRYE, C., FREEZE, W., BUCKINGHAM, F. K. *Microsoft Office Excel 2003 Programming Inside Out*, Microsoft Press, 2004
[54] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4

Aplikace Excel obsahuje několik ovladačů událostí neboli rutin kódu, které sledují určité akce. Nastane-li některá z těchto akcí a uživatel „informoval“ aplikaci Excel o tom, co má v takovém případě provést, spustí aplikace kód v ovladači událostí.[55]

Objekty v Excelu umí reagovat na velké množství událostí. Můžeme si

představit některé z nich:

- Události sešitu: probíhají v rámci sešitu (např. vytvoření, otevření sešitu, uložení nebo vytisknutí či přidání nového listu).
- Události listu: jsou volány pro určitý list (např. aktivování listu, změna výběru na listu nebo přepočítání listu).
- Události grafu: jsou volány pro určitý sešit typu graf (např. změna velikosti grafu nebo změna datového bodu v nějaké datové řadě).
- Události celé aplikace: probíhají na úrovni celého Excelu (např. vytvoření nového sešitu).
- Události formulářů a jejich ovládacích prvků: se vyskytují v rámci určitého formuláře nebo ovladače.
- Události, které neprobíhají v rámci žádného objektu: se funkčně odlišují od jiných událostí. [56]

Některé akce mohou spouštět i více událostí za sebou. Bližší průzkum posloupností, v jakých jednotlivé události postupně probíhají, se ale v praxi vyplatí pouze v případech, kdy je použito větší množství událostí najednou. Je-li třeba například spustit jednu proceduru při otevírání sešitu a druhou pak při přepočtu určitého listu, není třeba se ničeho zvlášť obávat.

Posledním prvkem objektově orientovaného programování je kolekce. Jak napovídá název, kolekce je skupina objektů stejného typu obsažených v jiném objektu. Sešit například obsahuje kolekci jednoho nebo více listů.[57] Ovšem platí, že i kolekce samé jsou též objekty.

Objektové proměnné tedy podstatně zjednodušují a zrychlují průběh kódu. Jakmile je objekt přiřazen do určité proměnné, VBA k němu může přistupovat rychleji než k normálnímu dlouhému odkazu, který musí neustále průběžně vyhodnocovat. Pakliže je rychlost provádění kódu důležitým faktorem, je nutno používat objektové proměnné.

VBA nabízí dvě základní možnosti, které zrychlí průběh kódu a značně zjednoduší práce s objekty a kolekcemi. První z nich je konstrukce With – End With, která umožní provést více operací s jedním objektem. Podoba bez této konstrukce je sice možná čitelnější a srozumitelnější, avšak procedura, která pro změnu několika vlastností objektu používá konstrukci With – End With, může být výrazně rychlejší než procedura

[55] FRYE, C., FREEZE, W., BUCKINGHAM, F. K. *Microsoft Office Excel 2003 Programming Inside Out*, Microsoft Press, 2004
[56] ČERNÝ, J. *Programování v Excelu 2000, 2002, 2003*, Grada, Praha, 2005
[57] FRYE, C., FREEZE, W., BUCKINGHAM, F. K. *Microsoft Office Excel 2003 Programming Inside Out*, Microsoft Press, 2004

první, jež se explicitně na objekt odkazuje v každém příkazu. Druhou možností je konstrukce For Each – Next, která se hodí zejména v případě, kdy je třeba „něco provést se všemi objekty určité kolekce. Nebo je nezbytné všechny objekty vyhodnotit a podle nějakého kritéria pak provést určitou činnost.[58]

Důležité je, že u tohoto typu konstrukce není potřeba znát počet prvků, pro něž sled příkazů opakujeme.

Při tvorbě kódu často potřebujeme nějakým způsobem řídit tok příkazů, větvit program, vykonávat určité části opakovaně apod. VBA nabízí různé způsoby řízení provádění procedur nebo proces opakování určitého bloku příkazu. Větvení nebo podmíněný příkaz popisují schopnost provést jednu z několika možných posloupností příkazů (jednu z větví) a to v závislosti na nějaké podmínce.[59]

3.7.1 Způsoby řízení provádění procedur ve VBA

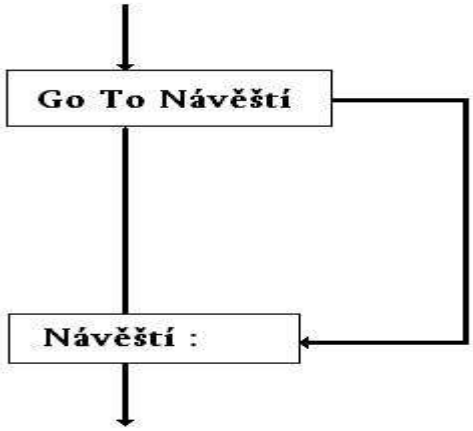
Způsoby řízení provádění procedur ve VBA dle Wolkenbacha:[60]

- Příkazy GoTo

Nejjednodušším způsobem změny toku programu je použití příkazu GoTo. Tento příkaz jednoduše přesune provádění programu na jiný příkaz, před kterým musí být návěští (textový řetězec následovaný dvojtečkou nebo číslo bez dvojtečky). Strukturu příkazu Go To můžeme vidět na obrázku 5 – 3.7.1 Příkaz Go To.

Procedury VBA mohou obsahovat libovolný počet návěští a příkaz GoTo nedokáže „odskočit“ mimo proceduru. Obecně platí, že příkaz GoTo by se měl používat jen v nezbytných případech, kdy neexistuje žádné jiné řešení požadovaného úkolu. V praxi se tento příkaz ve VBA používá jedině při zachytávání a zpracování chyb.

Obrázek 5 – 3.7.1 Příkaz Go To

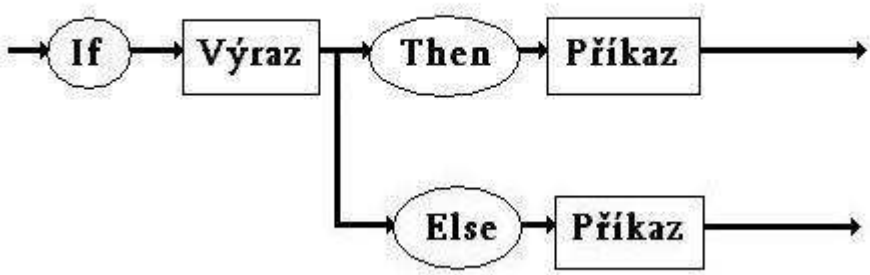


[58] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4
[59] GAULD, A. *Jak se naučit programovat*.[on-line].(18.3.2008) URL: <http://www.freenetpages.co.uk/hp/alan.gauld/czech3/index.html>
[60] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4

- Konstrukce If-Then

Snad nejčastěji používaným blokovým příkazem ve VBA je konstrukce If - Then, která představuje jeden ze způsobů, jak do aplikace zabudovat rozhodování o následujícím postupu. Kvalitní rozhodovací procesy jsou klíčovým prvkem úspěšných programů. Dobrá aplikace v Excelu se v podstatě dá „smrsknout“ na rozhodování a následné reakce. Vložené struktury typu If – Then bývají často těžkopádné, proto je lepší je používat pouze pro jedno rozhodování typu ano-ne.

Obrázek 6 – 3.7.1 Příkaz If-Then-Else



Tato konstrukce může mít několik nepovinných částí. Podmínka If – Then odpovídá „jestliže platí podmínka, udělej následující blok příkazů“. Pokud připojíme část Else (If – Then – Else), podmínka odpovídá: „jestliže platí podmínka, udělej následující blok příkazů, jinak udělej jiný blok příkazů“, strukturu tohoto příkazu můžeme viděk na obrázku 6 – 3.7.1 Příkaz If-Then-Else. Pokud chceme rozvětvit na tři různé cesty, použijeme další nepovinnou část ElseIf (If – Then – ElseIf – Else). Pak by to odpovídalo: „jestliže platí podmínka, udělej následující blok příkazů; pokud neplatí, ale platí druhá podmínka, udělej jiný blok příkazů; pokud neplatí ani jedna podmínka, udělej ještě jiný blok příkazů.

- Funkce VBA Iif

VBA ještě nabízí jednu alternativu ke konstrukci If – Then - funkci IIf. Tato funkce očekává tři parametry (vstupní rozhodování a výstup v případě pravdy či nepravdy) a pracuje téměř stejně jako funkce IF (KDYŽ) pracovního sešitu v Excelu.

- Konstrukce Select Case

Konstrukce Select Case se hodí pro rozhodování mezi třemi a více možnostmi. Tato konstrukce je dobrou alternativou k If - Then - Else. V každé větvi Case může být zapsán libo- volný počet příkazů, které jsou provedeny v případě, kdy je podmínka vyhodnocena pravdivě. Cykly umožňují opakování nějakých příkazů.

- Cykly For – Next

Nejjednodušší typ dobrého cyklu je cyklus For – Next. Když se použije cyklus For - Next, je důležité si uvědomit, že počítadlo cyklu je vlastně normální proměnná - na tom není nic zvláštního. Z toho však vyplývá, že je možné hodnotu proměnné změnit

kdekoli uvnitř bloku kódu prováděného mezi příkazy For a Next. To je ale velmi špatný postup a může způsobit nepředvídatelné výsledky. Nevýhodou může být také to, že musíme vědět, nebo musíme být schopni předem vypočítat, počet prováděných iterací. Nepovinný parametr Step slouží pro přeskakování některých hodnot počítadla cyklu. Nepovinný příkaz Exit For umožňuje okamžité ukončení cyklu a program pokračuje příkazem následujícím za příkazem Next aktuálního cyklu For - Next.

- Cykly Do While

Cyklus Do While je dalším typem cyklu ve VBA. Na rozdíl od cyklů For - Next se cyklus Do While provádí, dokud je zadaná podmínka splněna. To znamená, že tento druh cyklu můžeme použít v případech, kdy chceme pokračovat v provádění určitého úkolu až do doby, kdy nastane určitá situace, ale když přitom nevíme, kdy k dané situaci dojde. Podmínka While může být umístěna buď na začátku, nebo na konci cyklu. Rozdíl mezi těmito dvěma zápisy je v tom, kdy bude podmínka vyhodnocována. V prvním případě se obsah cyklu vůbec nemusí provést. Ve druhém případě se obsah cyklu vždy provede alespoň jednou. Cykly Do While může také obsahovat jeden nebo více příkazů Exit Do. Když program dospěje k příkazu Exit Do, cyklus se ihned ukončí.

- Cykly Do Until

Struktura cyklu Do Until je velmi podobná struktuře cyklu Do While. Rozdíl je zřejmý pouze v okamžiku testu podmínky. V cyklu Do Whi1e se cyklus provede, pokud je podmínka splněna. U cyklu Do Until se naopak cyklus provádí tak dlouho, dokud podmínka není splněna.

Jak je již uvedeno výše, příkaz GoTo se používá pouze v nezbytných případech, kdy není možné žádné jiné řešení. Na úrovni programovacích jazyků tento příkaz odporuje myšlence strukturovaného programování. Strukturované programy jsou považovány za lepší než programy nestrukturované. Základním předpokladem je to, že rutina nebo úsek kódu bude mít pouze jeden vstupní bod a jeden bod výstupní. Jinými slovy, tělo kódu by mělo být samostatnou jednotkou a program by neměl skákat někam dovnitř rutiny ani nevyskakovat z jejího kódu. Takže je jasné, že strukturované programování vylučuje používání příkazu GoTo. Během psaní strukturovaného kódu bude program postupovat systematicky řádek po řádku a bude ho možné snadno sledovat na rozdíl od tzv. „špagetového“ kódu (jeho tok se jeví jako náhodný), kdy program přeskakuje, jak se mu zlíbí. Strukturovaný program se snadněji čte a chápe než program nestrukturovaný. A co je důležité, také se snadněji upravuje.[61]

Hlavním cílem VBA je práce s daty, ty jsou uloženy v proměnných. Proměnná je de facto jednoduše pojmenovaná pozice v počítačové paměti, určená pro uložení zpravidla hodnoty. Do proměnných lze ukládat zpravidla široký výběr různých datových typů. Hodnota, kterou v sobě daná proměnná uchovává, je proměnné přiřazena pomocí operátoru rovnosti.

[61] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4

Názvy proměnných musí dodržovat následující pravidla:

- V názvech proměnných lze použít písmena, číslice a některá interpunkční znaménka. První znak však musí být písmeno.
- VBA nerozlišuje velikost písmen. Aby byly názvy proměnných dobře čitelné, programátoři v nich často používají velká i malá písmenka.
- Není možno používat mezery či tečky. Pro zlepšení čitelnosti názvu proměnných někdy programátoři používají znak podtržítka.
- Speciální znaky pro deklaraci datového typu proměnné (#, $, %, & nebo !) není možno použít uvnitř názvu proměnné.
- Názvy proměnných mohou mít až 254 znaků – ale nikdo se zdravým rozumem tak dlouhý název nepoužívá! [62]

Ve VBA lze používat širokou řadu vyhrazených slov. Jsou to ta, která nelze již použít jako názvy proměnných či procedur. Pokus se tak ale stane, editor ohlásí chybu. VBA má oproti jiným programovacím jazykům jednu nespornou výhodu, a to, že automaticky spravuje všechny interní detaily vyplývající z aktuální práce s daty.

Datový typ proměnné nám udává, jakým způsobem budou data dále uložena v počítači, jestli jako celá čísla, reálná čísla či řetězce atp. Ačkoliv se VBA automaticky stará o typy dat, je vhodné typy dat pečlivě nadefinovat, jelikož automatické přiřazení typu dat VBA zabírá větší množství spotřebované paměti. To je zejména u rozsáhlých a složitých aplikací na překážku. Protože pokud není deklarován datový typ proměnné, VBA použije výchozí datový typ Variant, data uložená v tomto datovém typu mění svůj typ podle toho, co je právě s daty prováděno, a proto je pro datový typ Variant vyčleněno značné množství paměti počítače, a ačkoli není vždy zcela využita, nelze ji pro nic jiného použít.

Obecné pravidlo je, že se používá takový datový typ, který pro danou proměnnou využívá co nejmenší počet bajtů, při kterém ještě zcela pojme všechna data, která mu mají být přidělena.

Při práci VBA s daty lze rychlost práce vyjádřit jako funkci počtu bajtů, které má VBA v daný okamžik k dispozici. Máme-li tedy menší počet bajtů obsazených daty, VBA k nim má rychlejší přístup a i s nimi rychleji pracuje.

Rozsah platnosti proměnné určuje, ve kterých modulech a procedurách může být proměnná použita:

- Lokální proměnná je deklarovaná uvnitř nějaké procedury. Může být používána pouze v té proceduře, v jejímž těle je deklarována. Když procedura skončí, tato proměnná nebude nadále existovat a Excel uvolní paměť, jež jí byla přidělena. Deklaruje se pomocí příkazu Dim nebo Private uvnitř dané procedury.
- Statická proměnná je deklarována na úrovni procedur a uchovává si svoji hodnotu dokonce i po ukončení procedury. Je deklarována pomocí klíčového slova Static.

[62] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4

- Modulová proměnná je dostupná pro všechny procedury v modulu. Hodnota této proměnné se při ukončení procedury nezmění. Výjimkou je jen případ, kdy je procedura ukončena příkazem End. Při provedení tohoto příkazu jsou „zahozeny“ hodnoty všech modulových proměnných. Deklaruje se pomocí příkazu Dim nebo Private na začátku modulu (před první procedurou).
- Veřejná proměnná je dostupná pro jakoukoliv proceduru v projektu, dokonce i pro procedury v jiných modulech. Deklarujeme ji pomocí příkazu Public na začátku modulu (před první procedurou). Tento typ deklarace se musí nacházet ve standardním modulu VBA – ne v modulu kódu pracovního listu nebo ve formuláři UserForms. [63]

Pole proměnných je skupina prvků stejného datového typu, které mají společný název. Na určitý prvek pole se odkazuje pomocí názvu pole a pořadového čísla prvku (indexu). Pole proměnných se deklaruje příkazy Dim nebo Public stejně jako obyčejná proměnná. V deklaraci lze zadat také počet prvků pole a to tak, že zadáme hodnotu prvního a i posledního indexu pole oddělené klíčovým slovem To. „Pole ve VBA mohou mít až 60 rozměrů, ale více jak 3 rozměry se využije jen zřídka.

Dvojrozměrné pole se nejsnáze představí jako klasická matice, třírozměrné pole pak jako krychle. Představa čtyřrozměrného pole je již složitější, ale šlo by si ho představit jako krychli ubíhající v čase, pole o více jak čtyř rozměrech se pak již vymyká běžným lidským představám.

Při tvorbě modulu pro výpočet TSP v této práci bylo často užito dynamického pole, které nemá předem daný počet prvků. Než se v kódu takové dynamické pole užije, je proto třeba použít příkaz pro stanovení počtu prvků v poli ReDim, popř příkaz ReDim Preserve, je-li třeba zachovat stávající hodnoty v poli. Oba příkazy lze v kódu použít libovolněkrát, a tedy měnit tak velikost pole tak často, jak je nezbytné.

Jako obdobné programovací jazyky i VBA má širokou paletu již vestavěných funkcí, které zjednodušují prováděné výpočty a operace. Mnohé tyto funkce jsou obdobné s funkcemi pracovních listů Excelu a používají se téměř stejným způsobem jako funkce v pracovních listech. Funkce lze též do sebe vnořovat.

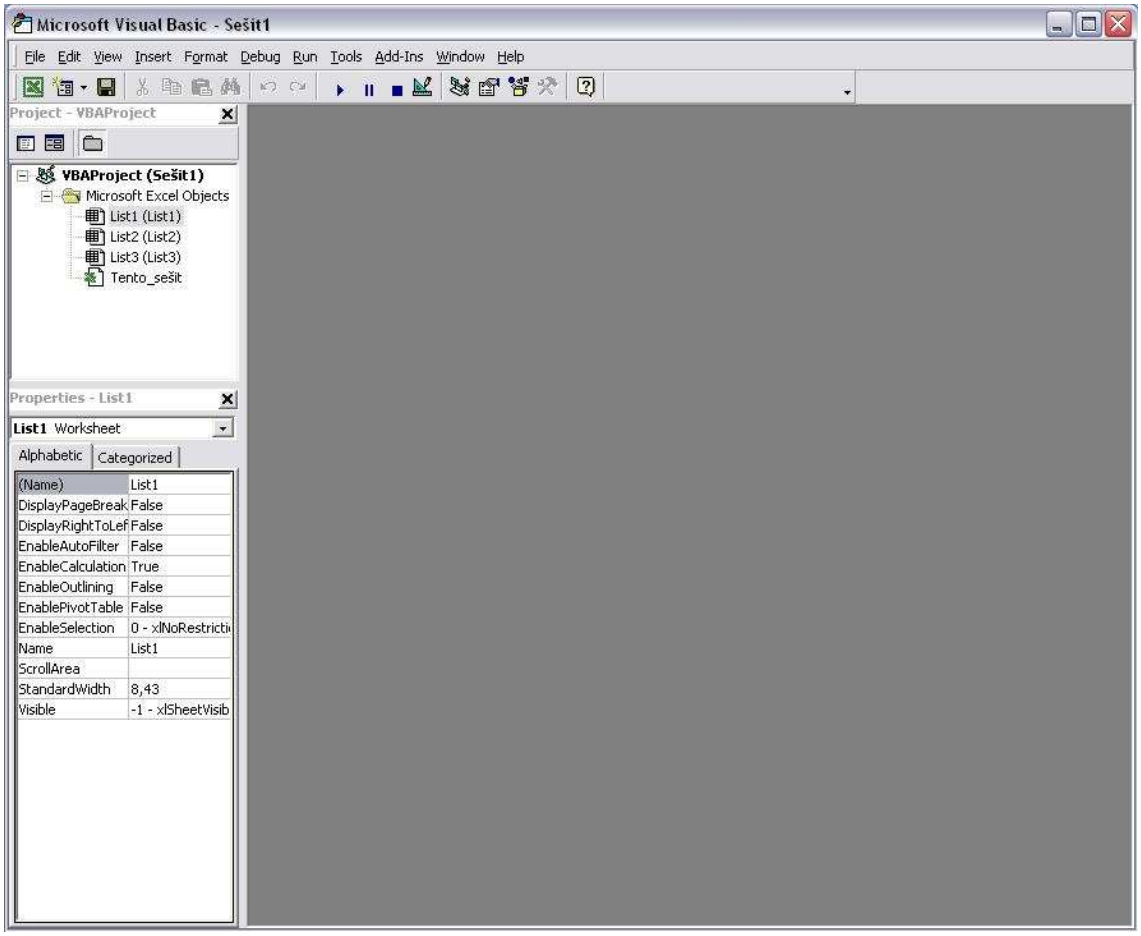
Z výše zmíněného výčtu může shrnout, že: jazyk VBA obsahuje všechny konstrukce moderních programovacích jazyků, včetně polí proměnných, cyklů a podobně. VBA se prvně objevilo v Excelu 5. V této verzi (a také v Excelu 95) se modul VBA zobrazoval přímo v sešitu jako samostatný list. Modul VBA obsahuje kód jazyka VBA. Počínajíce Excelem 97 se moduly již nezobrazují jako samostatné listy, veškerá práce s nimi probíhá v editoru jazyka Visual Basic (VBE).[64]

[63] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4

[64] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4

Aktivace VBE se provede stiskem kombinace kláves Alt+F11, po kterém se nám zobrazí základní obrazovka Microsoft Visual Basic, jak lze vidět na obrázku 7 – 4.2.1 Microsoft Visual Basic.

Obrázek 7 – 4.2.1 Microsoft Visual Basic



4 Moduly pro řešení okružního dopravního problému v MS-Excel v jazyce VBA

4.1 Tvorba modulu

V předešlé kapitole 3 Okružní dopravní problém, význam a metody bylo objasněno co je to okružní dopravní problém resp. okružní dopravní úloha a že je nedílnou součástí dopravní logistiky. Též byl vysvětlen pojem logistika a jakou roli hraje v dnešní dynamicky se rozvíjející společnost. Nahlédli jsme do metodologie užívané při řešení dopravních úloh, s tím, že jsme se podrobně seznámili s algoritmy metod v této práci používaných. Seznámili jsme se i se základními vlastnostmi programovacího jazyka Visual Basic for Applications (VBA), včetně vysvětlení pojmů, které jsou při tvorbě modulů v této práci využity. Nyní přikročíme k samotné tvorbě těchto modulů.

Jak již bylo řečeno, jedním z hlavních cílů této diplomové práce je vytvoření modulů pro řešení okružního dopravního problému neboli Travelling Salesman Problem (TSP). Moduly jsou psány v kódu VBA v editoru jazyka Visual Basic, který spustíme přímo v MS Excelu volnou v menu Nástroje -> Makro –> Editor jazyka Visual Basic a nebo pomocí klávesové zkratky Alt + F11.

Editor jazyka obsahuje řadu částí. Okno Project Explorer na obrázku 8 – 4.1 vlevo zobrazuje stromový diagram, který obsahuje každý sešit, jenž je zrovna v Excelu otevřen (včetně doplňků a skrytých sešitů). Každý sešit se označuje jako projekt (project). Pokud okno Project Explorer není vidět, stiskneme Ctrl + R. Okno kódu (někdy se mu také říká okno modulu, viz obrázek 8 – 4.1 vpravo) obsahuje kód VBA. Každá položka v projektu má přiřazené okno kódu. Chceme-li okno kódu vidět musíme poklepat na objekt v okně Projesct Explorer. Okno Immediate se hodí především pro přímé provádění příkazů VBA, testování příkazů a odlaďování kódu. Toto okno může nebo nemusí být viditelné. Pokus okno Immediate viditelné není, stiskneme kombinaci kláves CTRL + G.[65]

Okno kódu zpravidla může obsahovat čtyři typy kódu. Procedury Sub (skupina příkazů, které provádějí různou činnost), procedury Function (skupina příkazů, které vracejí buď jednu nebo celé pole hodnot), procedury vlastností (velmi speciální procedury používané v modulech tříd) a deklarace (informace o proměnných).

Jeden modul může obsahovat jakékoliv množství procedur všech procedur a deklarací. Způsob uspořádání v modulu je pak zcela v kompetenci programátora. V tomto případě je celá aplikace pro výpočet okružní dopravní úlohy rozdělena do dvou formulářů a čtyř modulů:

- Formuláře
    - Napoveda
    - Vstupni_Formular

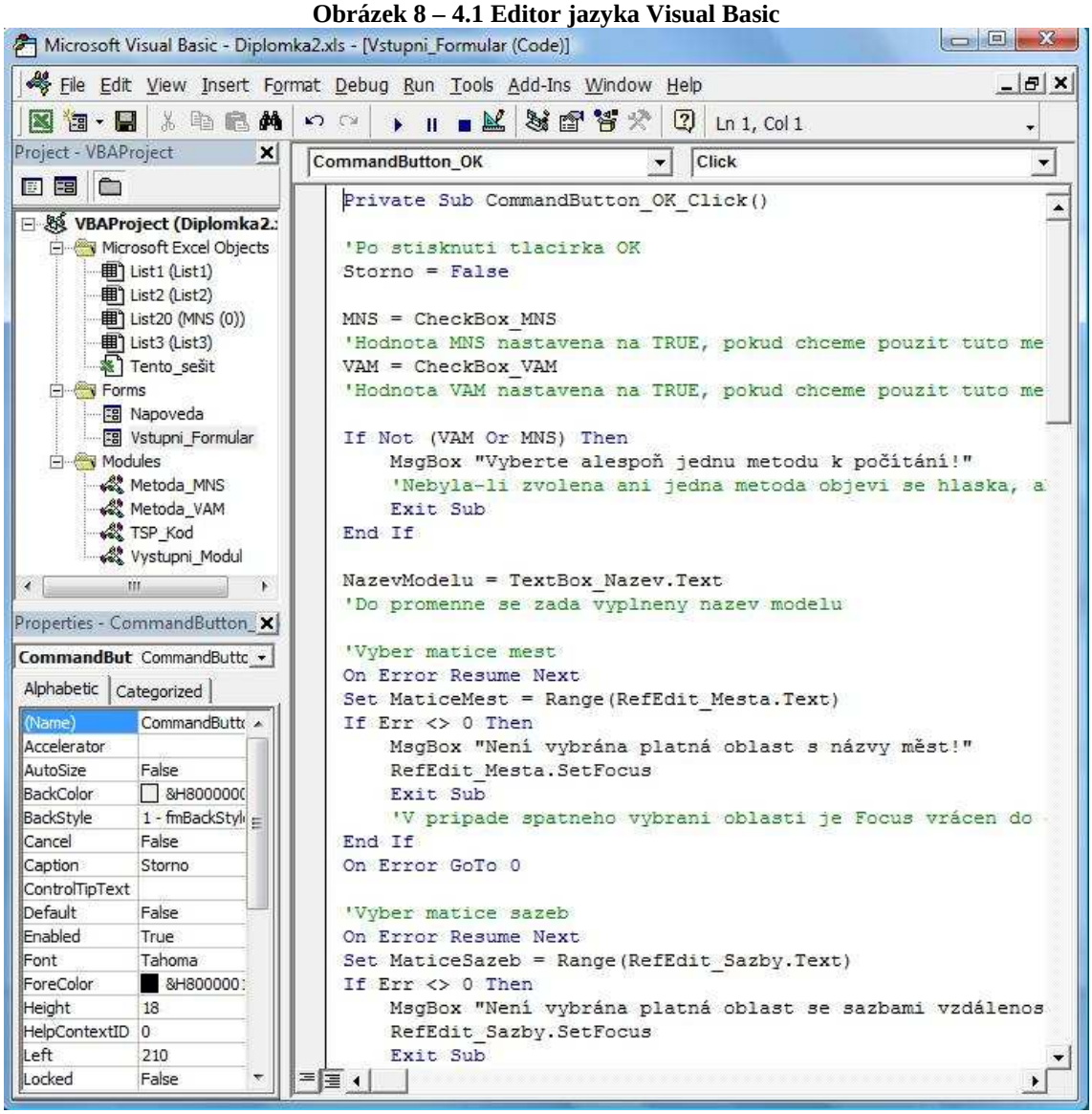
[65] WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*, Brno: Computer Press, 2006, ISBN 80-7226-547-4

- Moduly
    - Metoda_MNS
    - Metoda_VAM
    - TSP_Kod
    - Vstupni_Modul

Samotný kód je možné opatřit poznámkami komentujícími dané části kódu, ty musí vždy začínat apostrofem a je možné je umísti buď na začátek řádku, nebo na konec řádku za příslušný kód. Tento text je pak při samotném provádění činnosti procedur zcela ignorován, nemají tudíž žádný vliv na průběh procedury.

Celý kód této aplikace je přiložen jako příloha Kód modulů pro řešení okružní dopravní úlohy.

Obrázek 8 – 4.1 Editor jazyka Visual Basic



4.1.1 Modul Tsp_Kod

Tento modul obsahuje v horní části deklaraci veřejných proměnných a pak celkem tři procedury: Tsp, PridejDoMenu a VymazZMenu.

Samotný modul pro výpočet okružního dopravního problému se spouští pomocí procedury Tsp. Ta nejdříve provede zobrazení dialogového okna formuláře Vstupni_Formular pomocí příkazu Vstupni_Formular.Show. Po jeho zobrazení zůstane formulář na obrazovce tak dlouho, dokud ho sám uživatel opět nezavře. Jednotlivé oběkty a procedury, jakož i jejich funkce, vstupního formuláře jsou vysvětleny v kapitole 4.1.2 Formulář Vstupni_Formular.

Po zadání hodnot do formuláře a stisknutí tlačítka OK probíhá větvení dle booleovské proměnné Storno. Obsahuje-li proměnná Storno hodnotu True celá aplikace se ukončí a uzavře. V opačném případě, obsahuje-li hodnotu False, proběhne zbytek celé procedury.

Pomocí příkazu Application.ScreeenUpdating = False se vypne aktualizace obrazovky, tím se zrychlí průběh celého kódu, což je znatelné v případě má-li se počítat s mnoha městy. Do proměnné Mesta se načte počet zadaných měst a nastane opět dělení zda uživatel vybral matici formátu *n* x 1 či 1 x *n*.

V poslední řadě příkazů se pomocí příkazu If Then vybere jaká z metod se bude počítat, to pomocí načtených hodnot do proměnných MNS a VAM z formuláře.

Procedura PridejDoMenu je volána při otvírání sešitu MS Excelu a při uzavírání tohoto sešitu je volána procedura VymazZMenu. Procedura PridejDoMenu nejpve volá další procedura VymazZMenu pro případné odstranění již existující položky v menu. Následně je zde pomocí metody FindControl třídy CommandBars dohledána položka v menu Nástroje a přidána do objektu MenuNastroje. A k přidání položky NovaPolozka je užito metody Add kolekce Controls objektu MenuNastroje. Její vlastnost Caption značí její název, tedy Výpočet TSP. Vlastnost OnAction značí, že se má při spuštění spustit procedura Tsp.

Pcocedura VymazZmenu volá metodu Delete kolekce Controls Metody FindControl třídy CommandBars a odstraní se položka Výpočet TSP z menu.

Přidání či vymazání nové položky do či z menu nastává v okamžiku otevření či zavření sešitu MS Excelu pomocí Private Sub Workbook_Open, kde je volána procedura PridejDoMenu, a pomocí Private Sub Workbook_BeforeClose, kde je volána procedura VymazZMenu.

Tímto způsobem se zjednoduší uživateli přístup ke zpuštění aplikace pro výpočet okružní dopravní úlohy a zefektivní se práce uživatele.

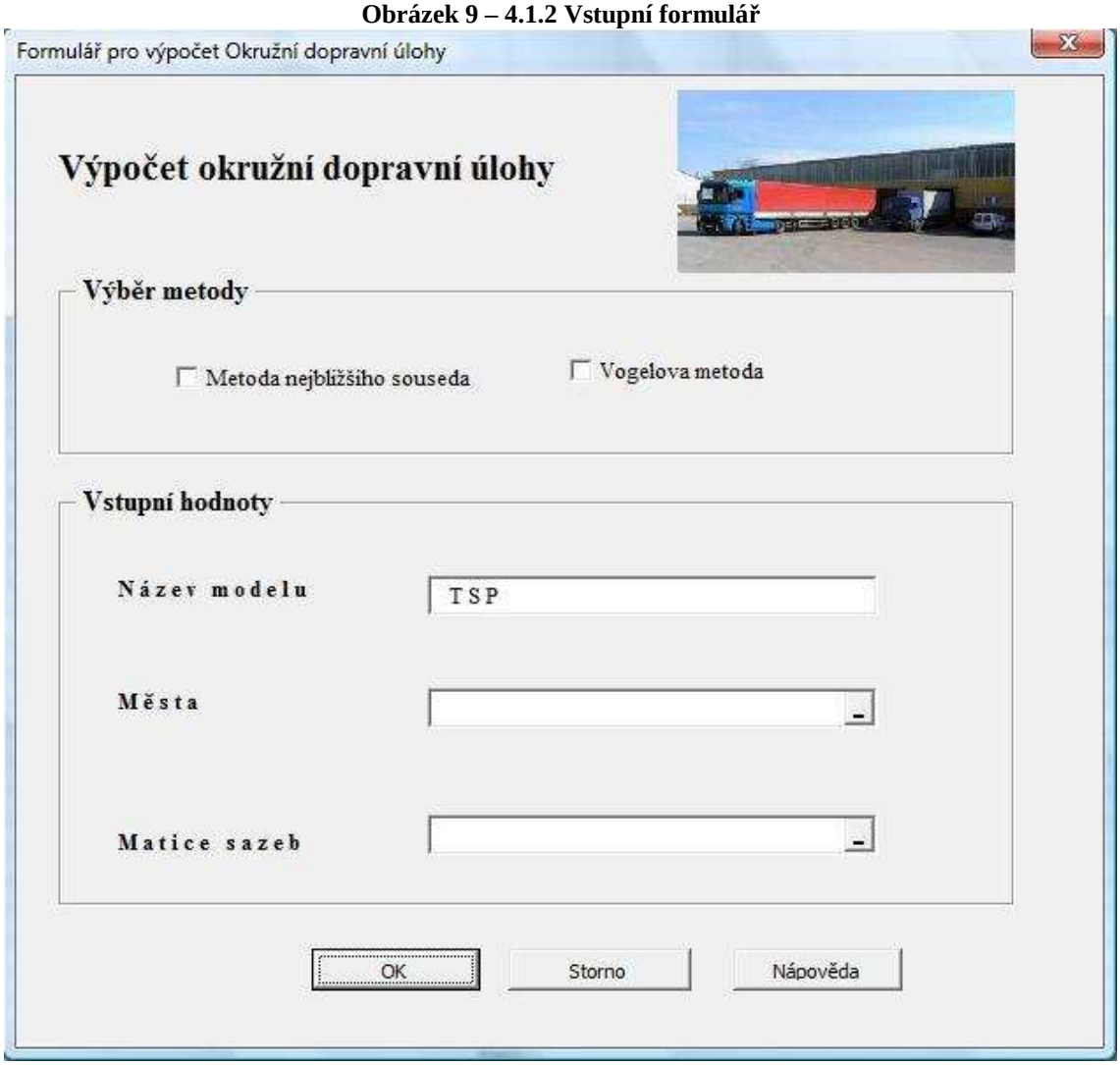
4.1.2 Formulář Vstupni_Formular

Formulář je tvořen ze dvnou částí, z objektu, tím je dialogové okno, které se spustí při výběru v menu, a ze samotného kódu, ten se zapisuje do okna kódu resp. modulu.

Po spuštění modulu výběrem v menu se zobrazí vstupní formulář pro výpočet okružní dopravní úlohy viz obr. 9 – 4.1.2, do kterého uživatel zadává potřebné vstupní hodnoty, volí zde metodu výpočtu, v horní části formuláře na obrázku a v prostřední

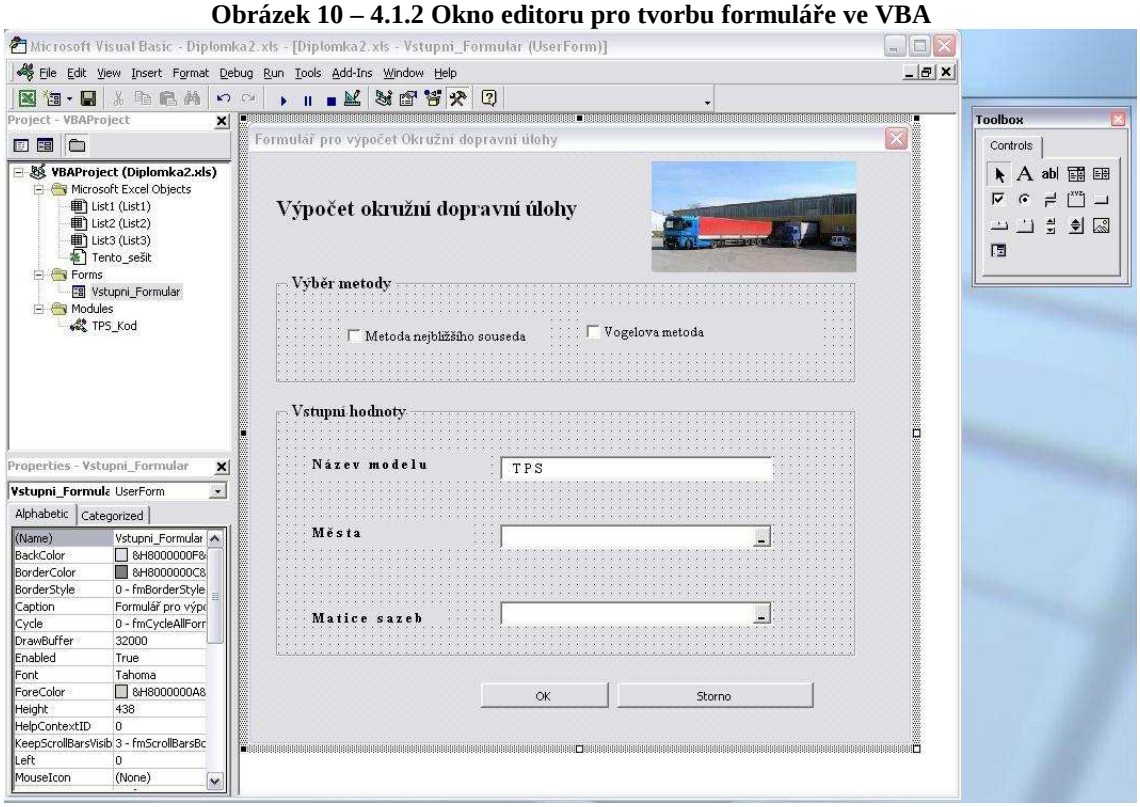
části pak zadává název modelu, matici měst a matici sazeb vzdálenosti mezi městy. Nejdůležitější vlastností celého vstupního formuláře je uživatelská jednoduchost a vstřícnost pro koncového uživatele. Proto je vstupní formulář tvořen základními ovládacími prvky, které se snadno ovládají a jsou přehledně uspořádány.

Obrázek 9 – 4.1.2 Vstupní formulář



Každý formulář se skládá z různých ovládacích prvků. Pro přidávání těchto prvků do formuláře slouží okno Toolbox, na obrázku 10 – 4.1.2 Okno editoru pro tvorbu formuláře ve VBA vpravo, které obsahuje dle původního nastavení nejpoužívanější ovládací prvky. Další prvky, které nejsou v okně Toolbox zobrazeny, ale jsou nainstalovány lze do Toolboxu přidat. Ovládací prvky jsou z Toolboxu do formuláře přidávány pomocí jednoduché techniky drag&drop (uchycení zvoleného objektu, přetažení na potřebnou pozici a následného puštění, vše se provádí pomocí kurzoru myši). Druhou možností jak zvolený objekt přidat do formuláře je pomocí výběru ikony zastupující daný objekt a následným vykreslením do formuláře v námi požadované velikosti.

Obrázek 10 – 4.1.2 Okno editoru pro tvorbu formuláře ve VBA



Na obrázku 10 – 4.2.1 Okno editoru pro tvorbu formuláře ve VBA je vidět, že formulář vytvořený pro tuto aplikaci k výpočtu okružní dopravní úlohy se skládá z několika rozličných ovládacích prvků. Každý z nich je definován svým vlastním názvem, na který se lze odvolávat v programovacím kódu formuláře.

K seskupení několika ovládacích prvků ve formuláři se často využívá ovládací prvek Frame (rámeček). Docílí se tím přehlednosti a snazšího užívání formuláře, obzvláště jsou-li prvky podobného charakteru. V tomto případě jsou použity dva rámečky rozdělující formulář na dvě základní části, a to na část pro volbu metody výpočtu okružní dopravní úlohy a na část pro zadávání vstupních dat.

Ve vrchní části formuláře je ještě použit objekt Label (popisek) v dialogovém okně zobrazuje text, který nemůže být uživatelem změněn, v tomto případě slouží jako nadpis celého formuláře a ve spodní části jsou pak použity dvě příkazová tlačítka (CommandButton) viz obrázek 10 – 4.1.2, k jejich popisu se však ještě dostaneme.

Součástí prvního rámečku jsou dvě zaškrtávací tlačítka (CheckBox), která jsou využita pro volbu z více možností. Rozhoduje-li se uživatel kterou volbu si vybere nebo zda s něčím souhlasí či nikoliv. Z toho tedy vyplívá, že toto tlačítko má pouze dvě hodnoty, a to buď True (zaškrtnuto) nebo False (nezaškrtnuto).

V našem konkrétním případě se první zaškrtávací políčko v pořadí jmenuje CheckBox_MNS. Vlastnost Caption umožňuje přidat k tomuto políčku určitý text, v našem případě, protože políčko slouží k volbě metody nejbližšího souseda pro výpočet TSP se jedná o text metoda nejbližšího souseda. Vlastnost Value je standardně

nastavena na False (nezaškrtnuto), při zaškrtnutí tohoto políčka se mění na hodnotu True (zaškrtnuto).

Druhé zaškrtávací políčko se jmenuje CheckBox_VAM, slouží pro volbu Vogelovy metody, a tedy vlastnost Caption obsahuje analogický text jako první políčko, tedy Vogelova metoda, vlastnost Value je stejná jako u prvního políčka.

Druhá oblast obsahuje tři druhy ovládacích prvků. Label (popisek), TexBox (textové pole) a RefEdit (zadávací pole).

Label nám, jak již bylo výše zmíněno, slouží jako určitý popis, v našem případě pro identifikaci navazujícího textového pole a dvou zadávacích polí (RefEdit). Jejich vlastnosti Caption jsou postupně: Název modelu, Města a Matice sazeb.

Textové pole je pojmenováno TextBox_Nazev a umožňuje koncovému uživateli zadat název modelu. V tomto případě je vložena předdefinovaná hodnota Value TSP jako Traveling Salesman Problem.

Pro vybrání určité oblasti buněk z pracovního listu se používá ovládací prvek RefEdit. První z nich je v našem případě pojmenován RefEdit_Města a slouží pro výběr matice názvů měst a druhý v pořadí je pojmenován RefEdit_Sazby a slouží pro výběr matice sazeb vzdáleností mezi městy.

Poslední část formuláře obsahuje dvě příkazová tlačítka (CommandButton). První v pořadí je CommadButton_OK, kterým se potvrzují všechny zadané a vybrané hodnoty formuláře a druhým příkazovým tlačítkem je CommadButton_Storno, které zavře celý formulář a tím i ukončí celou aplikaci, třetím příkazovým tlačítkem je CommandButton_Napoveda, po jehož spuštění se zobrazí malý formulář s nápovědou.

Kromě okna vlastního návrhu pracovních objektů ve formuláři (možno zobrazit pomocí View Object) obsahuje formulář i editor programovaného kódu (možno zobrazit pomocí View Code). Ten obsahuje událostní procedury ovládacích prvků. Při vyvolání nějaké události ovládacího prvku, jsou tyto procedury volány. Kromě vlastních událostních procedur existují i procedury kontrolující zadaný obsah ve formuláři, a ty jsou v našem případě užity také.

Dále si popíšeme jednotlivé procedury událostní i kontrolní, které jsou v modulu pro výpočet okružního dopravního problému využity.

Procedura CommandButton_OK_Click obsahuje příkazy, jež budou provedeny po stisknutí tlačítka OK, kterým zároveň potvrdíme celý formulář se všemi jeho výběry a zadáními. Na začátku se nejprve do daných proměnných uloží hodnoty ovládacích prvků formuláře. Zároveň se do proměnné Storno přiřadí hodnota False, která říká, že nedošlo ke zrušení celého formuláře.

Po tomto načtení hodnot do proměnných následují kontroly načtených hodnot. Nejprve se kontroluje zda došlo k vybrání jedné z metod výpočtu okružního dopravního problému, pokud ne, zobrazí se hláška pomocí MsgBox, která signalizuje, že jsme nevybrali žádnou z metod a říká „Vyberte alespoň jednu metodu k počítání !“ Dále jsou kontrolovány hodnoty načtené pomocí ovládacích prvků RefEdit a to do proměnných MaticeMest a MaticeSazeb. Dojde-li v přiřazování hodnot k chybě, opět se zobrazí hláška pomocí MsgBox, která nás na tuto skutečnost upozorňuje s textem „Není vybrána platná oblast s názvy měst“ resp. „Není vybrána platná oblast se sazbami

vzdáleností“, fokus je vrácen do příslušného prvku RefEdit a celá procedura je ukončena, přičemž zůstává formulář stále aktivní.

Poté je volána procedura Kontrola, která dále volá procedury jednotlivých kontrol, které kontrolují hodnoty přiřazené proměnným zadané uživatelem, jejich výstupem je proměnná Rozsah, do nějž je v případě, že je vše v pořádku zadána hodnota True. Shledá-li kontrola nějakou chybu zadaných hodnot je do proměnné Rozsah přiřazena hodnota False, zobrazí se hláška upozorňující na tuto chybu, fokus je vrácen na příslušný RefEdit a procedura je ukončena.

Celkem je tedy kontrolováno:

Kontrola_Rozsahu kontroluje zda uživatel nezadal matici názvů měst větší než rozsahu 255 x 255. Protože v hlavní proceduře Tsp je tento rozměr přidělen do proměnné Mesta datového typu Byte o rozsahu hodnot 0 až 255.

Kontrola_Ctvercove_Matice kontroluje zda je matice čtvercového rozměru, a tedy zda-li s ní můžeme nadále počítat.

Kontrola_Nad_Hlavni_Diagonalou kontroluje hodnoty matice sazeb vzdáleností mezi města, zda jsou hodnoty nezáporné, nenulové a zda jsou všechny hodnoty číselného typu.

Then Kontrola_Mest_A_Sazeb kontroluje zda odpovídá počet názvů měst rozměru matice sazeb vzdáleností mezi městy.

Procedura CommandButton_Storno_Click nastaví po vyvolání události Click hodnotu proměnné Storno na True, což způsobí ukončení dialogového okna vstupního formuláře a tím ho celý ukončí a uvolní z paměti počítače.

Procedura CommandButton_Napoveda_Click je velmi jednoduchá, má za úkol pouze spustit formulář s nápovědou, jež vysvětluje jaký druh údajů se má do různých částí vstupního formuláře vkládat.

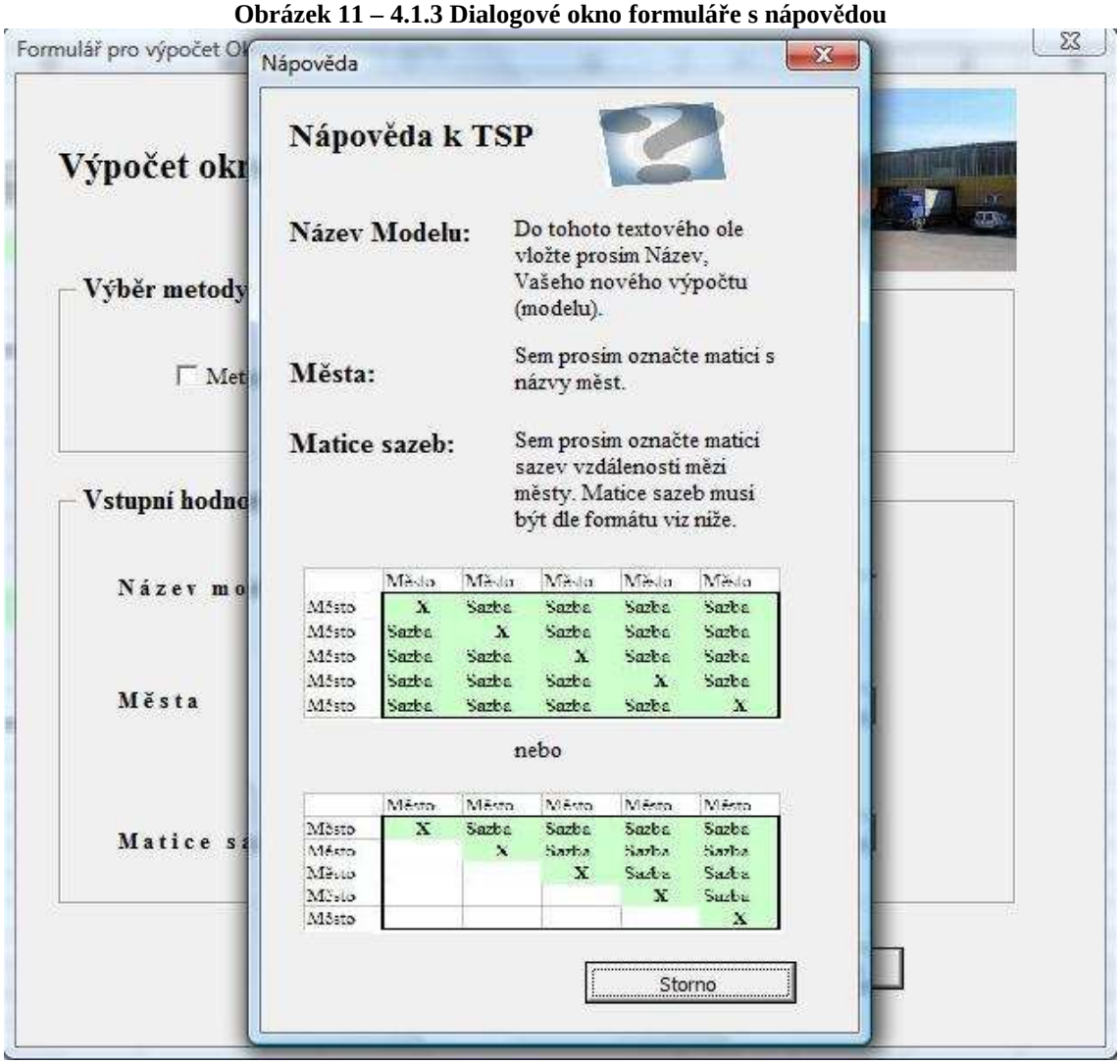
4.1.3 Formulář Napoveda

Formulář Napoveda je vcelku jednoduchý formulář, obsahuje pouze dva druhy ovládacích prvků. Objekt Label (popisek) ten zde byl použit pro vložení textu a obrázků a jedno příkazové tlačítko (CommandButton), které nese název CommandButton_Storno.

Pomocí textu je zde popsáno, jaká data má uživatel vkládat do vstupního formuláře a k tomu slouží i dva obrázky znázorňující formát matice sazeb vzdáleností mezi jednotlivými městy viz obr. 11 – 4.1.3 Dialogové okno formuláře s nápovědou, aby nedocházelo k tomu, že by ji následná procedura, která bude s maticí paracovat, nedokázala rozpoznat.

Procedura CommandButton_Storno_Click funguje na obdobném principu jako ve vstupním formulář. Čili změní hodnotu vlastnosti Cencel z False na True a zavře celý formulář s nápovědou.

Obrázek 11 – 4.1.3 Dialogové okno formuláře s nápovědou



4.1.4 Modul Metoda_VAM

Tento modu obsahuje dvě procedury, první je procedura Upava_Matice, která slouží pro úpravu matice sazeb vzdáleností a procedura Vogelova_Metoda, ta počítá okružní dopravní úlohu pomocí Vogelovy metody.

Před vlastním prováděním příkazů procedury Vogelova_Metoda k výpočtu proběkne volaná procedura Uprava_Matice, proběhnou dva cykly For, přižemž jeden je vnořen do druhého a hodnoty buněk na hlavní diagonále jsou změněny na hodnotu znaku X, zajistí se tak, že s nimi nebude počítáno jako se vzdálenostními sazbami. Vzdálenost z jednoho města to téhož by musela být jinak nula, výpočtu nesmí uvažovat, a proto je nahrazena znakem X. Do hodnot buněk pod hlavní diagonálou jsou překopírovány hodnoty osově souměrných buněk nad hlavní diagonálou, to pro případ, že je matice sazeb vzdáleností zadána jako trojúhelníková matice, tj. druhý způsob zadání matice dle nápovědy.

V horní části kódu dochází k deklaraci proměnných, po níž následuje přiřazení hodnoty do proměnné NazevListu_VAM, která je pak předána jako parametr při volání procedury Vypis na konci kódu pro vypsání dosažených výsledků.

Jelikož jsou s maticí sazeb během tohoto algoritmu prováděny nejrůznější změny, je hodnota MaticeSazeb vložena do proměnné MaticeVAM, s níž se bude dále pracovat, a tím zůstane puvodní matice sazeb pro další účely nezměněná. Počáteční celková vzdálenost je nastavena jako nulová.

Algoritmus Vogelovy metody byl podrobně popsán výše v kapitole 3.5.4.2 Vogelova aproximační metoda a podobně jako u metody nejbližšího souseda se jedná o algoritmu neustále se opakující. Proto byl je tvořen jedním velkým cyklem For, který se opakuje pro všechna města až na poslední dvě, které zbudou. Tedy se zařazováním měst do se postupně v matici sazeb vyškrtávají hodnoty řad (slupce a řádky), a sazba, jež by předčasně ukončila okruh až zbudou poslední dvě nevyškrtnuté hodnoty sazeb určující poslední dvě města okružní trasy. Hodnota proměnné Step pak určuje město zařazované do výsledné trasy.

Algoritmus nejprve vyhledá nejmenší řádkovou s sloupcovou hodnotu v matici sazeb vzdáleností a z nich pak utvoří rozdíl. Minimum v řádku je uloženo do proměnné NejmensiRadek1 a druhá nejmenší sazba do proměnné NejmensiRadek2, obdobně i ve sloupci. Na samém začátku je to těchto proměnných uložena hodnota znaku X, která je vždy větší než číslo a poté je nahrazena. Poté dojde k porovnání těchto proměnných s ostatními hodnotami a je-li nějaká hodnota menší, nahradí dosud uloženou, tím docílíme nalezení minima a druhého minima v řadě.

Poté se stanoví rozdíl těchto dvou hodnot, případ, kdy by některá z hodnot nebyla číselného charakteru je ochráněn funkcí IsNumeric, která vrací hodnotu True, je-li hodnota číselná. Proměnná Rozdil, do které se vypočtený rozdíl ukládá, je dvourozměrné pole, jehož první index určuje pořadí v řadě a druhý určuje, jedná-li se o řádek (pak hodnota 1) či sloupec (pak hodnota 2).

Z pole rozdílů se následně vyhledá největší hodnota. Nejdříve je tato hodnota nastavena na –1, každá hodnota v poli je větší než –1, poté je každá hodnota z pole Rozdil pomocí proměnné m porovnávána s doposud největší vyhledanou hodnotou a při tomto porovnávání mohou nastat celkem tři situace: když je m menší než doposud nalezená hodnota, nic se nestane, když je m rovno doposud nalezené hodnotě, velikost pole rozdílů se zvětší o jedničku a je přidána hodnota m, a když je m větší doposud nalezené hodnotě, je hodnota resp. jeho hodnoty nahrazeny hodnotou m a s ní se uloží i její pozice do proměnné NR_Pozice (pořadí v řadě) a hodnota 1 či 2 v proměnné NR_RS značí, jedná-li se o řádek či sloupec.

Dále dle algoritmu Vogelovy metody určíme v řadě s největším rozdílem nejmenší hodnotu vzdálenosti mezi městy. Prohledá se tolik řad, kolik je rozměr pole největších rozdílů (NR) pomocí cyklu For, kde funkce Ubound (NR) určuje největší index pole. Nejmenší takto nalezená hodnota do proměnné MinSazba a její souřadnice do MinSazbaR pro řádek a MinSazbaS pro sloupec. Poté se tyto hodnoty přiřadí do pole výsledného okruhu a hodnota délky se přičte k celkové délce výsledné trasy.

Následuje vyškrtání hodnot v řadě určené minimální sazby a hodnoty v opačném směru trasy i hodnoty trasu předčasně uzavírající, a to pomocí pomocného pole

PTVAM, které svými indexy odpovídá proměnná VAM_od a VAM_do, tedy např. druhá hodnota pole odkazuje na druhou spojnici výsledné trasy, to vše z toho důvodu, že takto nalezená hodnota vzdáleností nemusí navazovat na některou z již předtím vyhledaných.

Po vyškrtání se cyklus For opakuje až po jeho skončení je do výsledné trasy přiřazeno $n - 2$ měst, kde n značí počet měst, pro přidání posledních dvou měst do výsledné trasy slouží další cyklus For, který prohledá všechny hodnoty pole matice MaticeVAM, ta však již obsahuje pouze dvě číselné hodnoty, ostatní byly v průběhu vyškrtávání nahrazeny znakem X. A tak pomocí funkce IsNumeric rozpozná proměnná pm poslední a předposlední hodnotu vzdáleností patřících do výsledné trasy. Zároveň se vzdálenosti celkové trasy obě hodnoty i přičtou.

Další příkazy kódu vypíší výsledek dosažený touto metodou. O výpisu výsledků výpočtu pojednává kapitola 4.1.6 Modul Vystupni_modul.

4.1.5 Modul Metoda_MNS

Před vlastním prováděním příkazů procedury Metoda_Nejblizsiho_Souseda k výpočtu proběkne volaná procedura Uprava_Matice, která slouží k úpravě matice MaticeSazeb. Funkce této procedury je popsána již v předchozí kapitole 4.1.4 Modul Metoda_VAM.

Procedura Metoda_Nejblizsiho_Souseda, která se nachází v modulu Metoda_MNS, vyřeší zadanou okružní dopravní úlohu za pomoci metody nejbližšího souseda.

Na začátku procedury jsou deklarovány proměnné, kterých bude použito. Název listu je proměnná, která nese údaj o názvu listu a která je předávána při volání pozdější procedury Vypis jako parametr.

Algoritmus celé metody, který byl popsán výše v kapitole 3.5.4.3 Metoda nejbližšího souseda, je naprogramován jako cyklus For, neboť algoritmus výpočtu je nutno opakovat pro každé město, jako město výchozí. Celková trasa z jednotlivých výchozích měst je pak mezi sebou porovnána a jako optimální je pak vybrána ta, která má nejmenší hodnotu celkové vzdálenosti. Celý algoritmus nepracuje s názvy měst jako s textovým údajem, ale jako s číselným, protože se uvažuje pořadí měst, resp. výchozího města. Hodnota proměnné Step je tedy pořadím výchozího města.

Do proměnné MaticeMNS se na začátku celého cyklu vloží obsah matice MaticeSazeb, protože s první maticí jsou prováděny během výpočtu různé úpravy a změny, přičemž druhá matice musí být zachována v nezměněné podobě pro další výpočty při změnách výchozích měst.

Hlavním principem algoritmu je, že ke každému městu v řádku hledáme město ve sloupci s nejmenší vzdáleností. Čili, že město v řádku je počátkem dílčího úseku trasy a město ve sloupci koncem tohoto úseku trasy. Celé toto hledání proběhne i-krát, kde i je počet měst. Jelikož je algoritmus pro hledání minima vždy stejná je v kódu uveden jako cyklus For, přičemž proměnná x značí pořadí měst ve sloupci.

Proměnná MinVzdálenost udává nejmenší hodnotu v řadě, která je právě prohledávána, přičemž MinVzdálenostX její sloupcovou souřadnici a MinVzdálenostY

její řádkovou souřadnici. Hledáme-li nějakou minimální hodnotu je zapotřebí do porovnávané proměnné na začátku zadat nějakou velmi vysokou hodnotu, protože však nemůžeme předpokládat jaké hodnoty se v matici sazeb vzdáleností, které uživatel zadá, budou nacházet, je výhodné jako prvotní hodnotu zadat znak, jelikož ten je vždy větší než číslo. Proto je do proměnné MinVzdálenost vložena hodnota MaticeMNS o souřadncích 1,1, což dle úprav matice sazeb popsaných v kapitole 4.1.2 Formulář Vstupni_Formular představuje znak X. Obdobně i do proměnné MinVzdálenostX je vloženo pořadí výchozího města 1. Do proměnné MinVzdalenostY je pořadí města vkládáno dle vzorce v cyklu For.

Při průběhu cyklu For je celý y-tý řádek prohledán a je-li nalezena vzdálenost kratší než je aktuální je minimální vzdálenost, je minimální vzdálenost nalezenou kratší nahrazena. Přičemž jsou i přepsány souřadnice v proměnných MinVzdálenostX a MinVzdálenostY.

Po nalezení minimální vzdálenosti v prvním řádku stačí již jen prohledat i ostatní řádky, ale před tím musíme ještě vyškrtnout možné vzdálenosti s městy, které by nám předčasně uzavřely okruh. Toho docílíme obdobně jako v proceduře Uprava_Matice, a to tak, že namísto takovéto hodnoty dáme znak X. Je třeba vyškrtnout hodnoty v daném řádku i sloupci nalezeného minima i cestu v opačném směru, protože však ale příslušný řádek již nebude v cyklu znovu použit, stačí vyškrtnout hodnoty jen v daném sloupci a hodnotu vzdálenosti cesty v opačném směru. Cestu v opačném směru vyškrtneme pomocí vyškrtnutí vzdálenosti v matici osově souměrnou a sloupec pomocí cyklu For.

Velikost polí UsekMestStep a UsekVzdalenostStep je postupně zvětšována podle zatím nalezených úseků při výpočtu z příslušného výchozího města (místa). Nejprve má hodnotu 1 dle počáteční hodnoty Step, tzn. že do výchozí trasy je nejprve přiděleno výchozí město. Postupně se vždy velikost tohoto pole zvětší o jedničku a je přidáno další město. Obdobně se postupuje i s proměnnou UsekVzdalenostStep. Kdy jsou postupně ukládány jednotlivé vzdálenosti výsledných měst.

Proměnná CelkovaVzdalenostStep je dána jako součet jednotlivých úseků nalezené trasy.

Poté co byly nalezeny celkové vzdálenosti tras pro všechna výchozí města, je musíme navzájem porovnat a vybrat tu nejkratší, kterou budeme uvažovat jako optimální, a tudíž výslednou.

Další příkazy kódu vypíší výsledek dosažený touto metodou. Volá se pomocí procedury Vypis. O výpisu výsledků výpočtu pojednává kapitola 4.1.6 Modul Vystupni_modul.

4.1.6 Modul Vystupni_ modul

Tento modul sloužící pro výstup resp. vypsání vypočtených výsledků okružní dopravní úlohy, a to voláním procedury Vypis na konci kódu jednotlivých metod. To z toho důvodu, že postup pro vypsání dosažených výsledků je téměř shodný, proto se celý algoritmus soustředil to tohoto modulu. Jsou tu zapsány celkem dvě procedury: samotná procedura Vypis, v níž je ještě volána procedura NovyList.

Procedura NovyList provede vložení nového pracovního listu, jeho následný přesun na konec řady listů a přejmenování dle předaného parametru. Pojmenování listu

obsahuje i číslovku pro odlišení více listů se stejným názvem. Pro stanovení číslovky se nejprve zjistí aktuální číslo daného listu v sešitě a poté pomocí cyklu For porovnání listů s názvem předaným pomocí parametru a proměnnou NoveCislo a následně přidána číslovka o jednu vyšší než dosavadní nejvyšší číslo. Po vytvoření nového listu se již mohou vkládat dosažené výsledky pomocí procedury Vypis.

Na začátku procedury Vypis se upraví šířka jednotlivých sloupců v nově vytvořeném pracovním listě. Jednotlivé buňky jsou upraveny pomocí Range popř. celé oblasti buněk pomocí konstrukce With a EndWith, v horní části obrázku 12 – 4.1.6 Výstupní tabulka vypočteného výsledku je vidět provedené sloučení buněk. Samotná procedura Vypis však nevypíše výsledek celý. Po upravení buněk vypíše pouze do hlavičky výpisu, na obr. 12 – 4.1.6 řádek 1 až 6, název modelu (jak se náš výpočet bude jmenovat), název zvolené metody, počet měst cekem, celkovou kilometrovou vzdálenost výsledné okružní trasy a záhlaví pro jednotlivé úseky trasy.

Výslednou trasu resp. její jednotlivé úseky vloží do nově vytvořeného listu procedura volající proceduru Vypis, tedy buď Medota_Nejblizsiho_Souseda nebo Vogelova_Metoda ke konci svého kódu pomocí jednoho cyklu For s prvkem ActiveCell.Offset, na obrázku 12 – 4.1.6 začíná výpis jednotlivých úseků trasy názvem města Liberec.

Obrázek 12 – 4.1.6 Výstupní tabulka vypočteného výsledku

T S P			
Metoda nejbližšího souseda			
Celkem měst:			5
Celková vzdálenost okruhu v km:			513
Jednotlivé úseky trasy:			km
Liberec	--->	Ústí nad Labem	92
Ústí nad Labem	--->	Teplice	19
Teplice	--->	Praha	94
Praha	--->	Jihlava	123
Jihlava	--->	Liberec	185

4.1.7 Otestování aplikace

U každé aplikace je nanejvýš důležitá spolehlivost této aplikace. Je zcela nežádoucí, aby nám v kritickém okamžiku práce či výpočtu daná aplikace zkolabovala a přestala fungovat. Pro vyvarování se této události a pro odladění případných chyb aplikace, zejména jedná-li s o složité výpočty, je nutno provést testování aplikace. Testy patří mezi rozhodující fáze vývoje, zpravidla zabírá testování a odlaďování aplikací skoro stejně velké množství času jako vytvoření aplikace samé. Většina testů se však již provádí při samotném programování, ale i přes to je nutné se vždy přesvědčit, že daná

aplikace funguje podle očekávaných představ a koncový uživatel ji může snadno ovládat.

V našem případě proběhne otestování aplikace pro výpočet okružní dopravní úlohy přímo v prostředí nejnáročnějším na fungování aplikace, protože zatěžkáno zkouškou bude nejen samotné správné fungování aplikace, ale i uživatelská srozumitelnost a jednoduchost ovládání. Protože jedním z rozhodujících faktorů je i zda se koncovému uživateli s danou aplikací jednoduše a příjemně pracuje. Proto využijeme aplikaci v případové studii, kde je řešen okružní dopravní problém rozvozu brzdových součástek firmou Alfasped Logistik do jednotlivých provozoven po ČR. Výsledky této zatěžkávací zkoušky budou popsány níže v kapitole 4.2 Případová studie.

4.2 Případová studie

4.2.1 Firma Alfasped Logistik

4.2.1.1 Obecná charakteristika zvolené firmy

Společnost Alfasped Logistik s.r.o. je společnost s ručením omezeným, která se zabývá vnitrostátním a mezinárodním zasilatelstvím a provozováním skladového centra. Sídlí v Jablonci nad Nisou, Podhorská 93a, č.p. 1124, PSČ 466 01. IČO: 254 10 792. Zápis v obchodním rejstříku vedeným Krajským soudem v Ústí nad Labem oddíl C, vložka 16262 dne 9. 12. 1999.

Organizační struktura firmy se vyvíjela na základě požadavků a potřeb zákazníků. Firma má jednu organizační jednotku (provozovnu) uvnitř areálu firmy TRW Lucas Varity s.r.o., Na Roli 26, Jablonec n.N. (Spedice). Druhou provozovnou je nově vybudované logistické centrum rovněž v Jablonci n.N., ul. Belgická č.p. 4883, Rýnovice (skladové centrum), vedení firmy a účtárna sídlí v pronajatých kancelářích v Podhorské ulici v Jablonci nad Nisou.

4.2.1.2 Historie podniku

Společnost Alfasped Logistik s.r.o. byla oficiálně založena 9. prosince 1999 jako následník společností Alfasped s.r.o. a Dopravní kanceláře Škoda a Schulz. Obě původní firmy působily v Jablonci nad Nisou již od roku 1990 a zabývaly se poskytováním zasilatelských služeb, bez vlastních dopravních prostředků. Vlastní fyzická přeprava zboží probíhala a v současnosti nadále probíhá pomocí smluvních přepravců. Sloučení obou firem vyplynulo ze snahy většího uspokojování požadavků rozhodujícího zákazníka a to firmy TRW Lucas Varity Jablonec, jež zprvu představovala 95% celkových zakázek obou firem. Sloučením společností se docílilo větší stability a konkurenceschopnosti v podnikání v uvedené oblasti. Následné vypsání výběrového řízení na komplexního poskytovatele log. služeb firmou TRW a vítězstvím firmy Alfasped Logistik s.r.o., jednoznačně potvrdilo správnost nastolené cesty. Podmínkou pro poskytování těchto služeb bylo i získání certifikátu kvality ISO, které v roce 2001 firma získala. Komplex poskytovaných služeb: vnitrostátní a mezinárodní zasilatelství, zajišťování celních služeb, péče o provoz referentských vozidel TRW, celní, konsignační skladování, mytí plastových obalů, evidence a řízení toků obalů a prodej náhradních dílů z celosvětové produkce TRW. Tato silná vazba na společnost

TRW Jablonec i nadále přetrvává, ale firma Alfasped Logistik s.r.o. se orientuje i na ostatní zákazníky a to zejména v jabloneckém a libereckém regionu.

Prvním podnikatelským rokem byl rok 2000, ve kterém se takto nově vzniklá firma stabilizovala na regionálním severočeském trhu v oblasti dopravy a logistiky. V prvních letech byla hlavní podnikatelskou činností spedice s převažujícím mezinárodní dopravou. Od roku 2002 se rozšířila působnost podnikání i na poskytování skladovacích služeb, čímž společnost dosáhla vyšší úrovně plnění potřeb zákazníků a tím i jejich spokojenosti. V započatém trendu firma pokračovala i v dalších letech, důsledkem čehož bylo vybudování nového skladovacího centra s vyšší kapacitou skladovacích prostor, jelikož dosavadní pronajaté skladové haly přestaly již svou kapacitou vyhovovat. Nové skladové centrum poskytuje mimo vlastního skladování i službu povrchového čištění obalových materiálů. Na obr. 13 - 4.2.1.2 je zachyceno skladové centrum z průčelí, v jehož pravé horní části jsou administrativní kanceláře, ve spodní části je umístěna průmyslová automatická mycí linka a v levé části budovy jsou vlastní skladové prostory.

Obrázek 13 – 4.2.1.2 Logistické a skladové centrum firmy Alfasped Logistik s.r.o.

Vstupem do Evropské unie došlo k mnohým změnám legislativy České republiky, což se sice promítlo ve firmě Alfasped Logistik například změnou DPH apod., ale zásadnější důsledky tato událost na běh společnosti neměla, jelikož společnost byla již v této době plně připravena na integraci do evropských struktur.

Do budoucna se společnost chce i nadále zabývat racionalizací práce, synchronizací a většímu zefektivňování logistických procesů, ale především prohlubování styku se zákazníky a co nejlepším uspokojováním jejich potřeb a přání.

4.2.1.3 Předmět podnikání firmy

Hlavním předmětem podnikání společnosti Alfasped Logistik s.r.o. podle zákona o živnostenském podnikání a na základě vydaných živnostenských listů, koncesních listin a společenské smlouvy je:

- Vnitrostátní a mezinárodní zasilatelství vč. zprostředkovatelské činnosti v této oblasti.
- Provozování skladů, skladování zboží vč. zprostředkovatelské činnosti v této oblasti.
- Silniční motorová doprava nákladní.
- Opravy silničních vozidel. [66]

4.2.1.4 Hospodaření firmy

Firma Alfasped Logistik s.r.o. se řadí mezi společnosti střední velikosti. Od počátku své podnikatelské činnosti hospodaří každoročně se ziskem, který není použit pouze pro osobní spotřebu formou výplaty podílů společníků na zisku, ale rozhodnutím valné hromady společnosti se tento zisk vždy transformuje do investičních aktivit podniku.

Dosavadní hospodářský vývoj poukazuje na trvale se zvyšující hodnotu podniku a další rozšíření poskytovaných služeb pro uspokojování potřeb zákazníků.

4.2.1.5 Informační systém firmy

Vlastním základem logistiky je práce s daty a informacemi. Proto každá firma, která chce v tomto oboru podnikání uspět, musí mít propracovaný informační systém tak říkajíc „ušitý na míru“ dané společnosti. Pomocí tohoto systému by se měli jednotlivé zakázky a přepravy tak zkombinovat, aby se dosáhlo co nejvyššího využití užitné hmotnosti přepravního vozidla s maximálním uspokojením přání zákazníka.

Firma Alfasped Logistik s.r.o. má vlastní informační systém sloužící logistice, jehož topologie se odvíjí od organizačního schématu společnosti, tento informační systém je složen ze dvou základních modulů a je koncipován na dvě sféry působnosti. Na sféru vnitřní, která slouží k potřebám jednotlivým zaměstnancům, kteří se systémem pracují a mají do něj diferencovaný přístup podle jejich příslušné kompetence. A sféra vnější je zaměřena na poskytování služeb zákazníkům, kde mají možnost podávat objednávky na přepravu zboží a materiálů. Podat objednávku lze několika způsoby. Tím nejjednodušším je osobní či telefonní kontakt se speditérem firmy. Tento způsob však již není preferován, protože zde může dojít k nedorozumění či nepřesnému zadání objednávky a především chybí jakýkoliv doklad, který by v případě nejasností věc vysvětlil. Druhým způsobem je pomocí faxu, kde odpadá problém nedostatku objednávkového dokumentu. Efektivním způsobem je dnes elektronické podání objednávky pomocí e-mailu, ale jednoznačně preferovaným způsobem a ve firmě Alfasped Logistik s.r.o. převážně užívaným je zadávání přepravních zakázek on-line způsobem pomocí informačního systému TOCDB.

[66] *Alfasped s.r.o.* [on-line].(6.6.2008)URL:<http://www.alfalog.cz/profil.html>

Jak již bylo zmíněno firma Alfasped Logistik s.r.o. má informační systém skládající se ze dvou modulů. Prvním modulem je TOCDB (Transport Order Client Data Base), což je systém pro zakládání objednávek spolu s databázovým systémem pro on-line sledování každé přepravní zakázky. Tento internetový systém umožňuje klientům i firemním zaměstnancům sledovat aktuální stav a vyřizování konkrétní přepravní zakázky. Přístup do tohoto systému je ošetřen přístupovým kódem, jenž má každý zákazník a příslušný zaměstnanec v souladu s jeho organizační kompetencí.

Druhým modulem je STOCKS, což je informační skladovací systém v logistickém centru firmy. Zde je možno sledovat aktuální stav a průběh skladování a expedice zboží jednotlivých zákazníků. Přístup do tohoto systému je taktéž ošetřen přístupovými kódy jednotlivých zákazníků a zaměstnanců.

Vnitřní firemní počítačová síť podniku je organizována pomocí internetového spojení do jednotlivých středisek odpovídajících organizačním jednotkám a je spravována síťovým správcem a pomocí centrálního serveru. Úplná implementace informačního systému ve společnosti zajišťuje přístup k potřebným informacím pro práci a plně podchycuje rozhodující parametry v oblasti přepravy a skladování.

Z informační databáze tohoto systému se čerpají informace pro zabezpečení přepravy, jedná se zejména o charakter zboží, jeho hmotnost, objem, nebezpečnost materiálu, čas dodávky aj., díky nimž se sestavuje model přepravního řetězce, tedy bude-li se jednat o model hvězdicové přepravy či o model sběrné okružní přepravy. Čím více informací je v databázi, tím efektivněji a přesněji se speditér rozhoduje, jaký model přepravy zvolí a tím lépe se vyhoví jak ekonomickým požadavkům, tak i přáním zákazníka.

4.2.2 Zadání

Firma Alfasped Logistik se zabývá mezinárodním a vnitrostátním zasilatelstvím (jedná se o tzv. „čistou spedici“[67]). Firma vyhrála konkurz na velkou zakázku distribuce brzdových součástek ze skladu firmy TRW v Liberci do 15 provozoven k následnému opracování.

Firma má k dispozici požadavky jednotlivých provozoven na brzdové součástky a může si k zajištění přepravy najmout od soukromých dopravců vozidla o různé užitné hmotnosti s rozdílnými sazbami přepravních nákladů na 1 km a různými sazbami mýtného. Manipulační poplatek[68] je stanoven pevnou sazbou 70,- Kč na paletu, přičemž se na jednu paletu dá naložit cca. 1 t brzdových součástek.

1) Stanovit jednotlivé trasy rozvozu brzdových součástek a zvolit nejvýhodnější druh vozidla s přihlédnutím na dopravní náklady a sazby mýtného.
2) Spočítat celkové přepravní náklady firmy Alfasped Logistik a celkovou fakturační sumu firmě TRW (spediční firma si účtuje 12%-ní marži z celkových přepravních nákladů).

[67] „Čistou spedicí“ se rozumí spedice bez vlastních dopravních prostředků a k zajištění přepravy si najímá služeb dopravců.
[68] Mezi manipulační náklady dopravců se uvažuje nakládka a vykládka přepravovaného materiálu.

4.2.3 Požadavky jednotlivých provozoven

Jednotlivé provozovny na opracování brzdových součástek jsou celkem v 15 městech uvedených v následující tabulce 1 – 4.2.3 Přehled provozoven s požadavky, ve které je současně uveden i požadavek na brzdové součástky v tunách pro zajištění plynulého opracovávání součástek.

Tabulka 1 – 4.2.3 Přehled provozoven s požadavky

Město	Požadavek
Hradec Králové	4,0
Cheb	4,5
Jihlava	5,2
Kladno	4,5
Klatovy	4,0
Olomouc	5,2
Opava	4,0
Ostrava	5,2
Pardubice	4,5
Písek	5,2
Plzeň	5,2
Praha	4,0
Teplice	4,5
Ústí nad Labem	4,0
Celkem	64,0

4.2.4 Význam přepravních nákladů

Přepravní náklady jsou součástí velké složky logistických nákladů. Cílem dobře fungujícího podnikového managementu je minimalizovat veškeré náklady tedy i náklady na přepravu. Použitím metod operační a systémové analýzy, v našem případě Mayerovy metody, metody nejbližšího souseda a Vogelovy metody při řešení okružního dopravního problému, lze nacházet výhodná řešení, jejímž výběrem můžeme účinně snižovat dopravní náklady a tím tedy i celkové náklady na přepravu zboží a materiálů.

Přepravní náklady můžeme členit na:

1) Dopravní náklady – v dopravním prostředku
2) Manipulační náklady – mimo dopravní prostředek

4.2.5 Seznam měst a jejich vzdáleností

Tabulka 2 – 4.2.5 Seznam měst a jejich vzájemných vzdáleností

	Hradec Králové	Cheb	Jihlava	Kladno	Klatovy	Liberec	Olomouc	Opava	Ostrava	Pardubice	Písek	Plzeň	Praha	Teplice	Ústí nad Labem
Hradec Králové	x	287	110	143	248	97	149	205	240	21	217	206	112	206	166
Cheb	287	x	298	156	130	247	450	524	537	279	180	101	175	145	164
Jihlava	110	298	x	154	193	185	166	240	253	89	118	186	123	217	215
Kladno	143	156	154	x	129	133	306	380	393	135	114	87	31	76	81
Klatovy	248	130	193	129	x	238	360	434	447	240	75	42	136	169	188
Liberec	97	247	185	133	238	x	246	300	335	118	207	196	102	102	92
Olomouc	149	450	166	306	360	246	x	74	93	147	285	369	275	369	367
Opava	205	524	240	380	434	300	74	x	35	210	359	443	317	388	369
Ostrava	240	537	253	393	447	335	93	35	x	240	372	456	362	456	454
Pardubice	21	279	89	135	240	118	147	210	240	x	188	198	104	198	196
Písek	217	180	118	114	75	207	285	359	372	188	x	81	105	199	197
Plzeň	206	101	186	87	42	196	369	443	456	198	81	x	94	127	146
Praha	112	175	123	31	136	102	275	317	362	104	105	94	x	94	92
Teplice	206	145	217	76	169	102	369	388	456	198	199	127	94	x	19
Ústí nad Labem	166	164	215	81	188	92	367	369	454	196	197	146	92	19	x

4.2.6 Mýtné

Na začátku roku 2007 byl v ČR zprovozněn na dálnicích a rychlostních komunikacích mýtný systém založený na mikrovlnné technologii. Jeho provozovatelem je Státní fond dopravní infrastruktury (SFDI).

Mýtný systém u nás vybudovala společnost Kapsch na 970 km dálnic a rychlostních komunikacích, na kterých stojí celkem 178 mýtných bran, prostřednictvím nichž jsou vybírány poplatky za jízdu vozidla po těchto zpoplatněných pozemních komunikacích. Mýtné je stanoveno v závislosti na počtu ujetých km mezi 2 branami a na typu vozidla.

Povinnost platit mýtné je předepsáno novelou zákona 13/1997 Sb., o pozemních komunikacích. Týká se silničních motorových vozidel, které mají nejméně 4 kola a maximální povolenou hmotnost[69] 12 a více tun. Jednotlivé sazby mýtného jsou uvedeny v tabulce 3 – 4.2.6 Sazba mýtného v ČR.

Tabulka 3 – 4.2.6 Sazby mýtného v ČR[70]

Sazby mýtného pro dálnice a rychlostní silnice					
Tabulka mýtných sazeb (Kč/km)					
Emisní třída do Euro II			Emisní třída Euro III nebo vyšší		
Počet náprav					
2	3	4<	2	3	4<
2,30 Kč/km	3,70 Kč/km	5,40 Kč/km	1,70 Kč/km	2,90 Kč/km	4,20 Kč/km

Sazby mýtného pro silnice I. třídy					
Tabulka mýtných sazeb (Kč/km)					
Emisní třída do Euro II			Emisní třída Euro III nebo vyšší		
Počet náprav					
2	3	4<	2	3	4<
1,10 Kč/km	1,80 Kč/km	2,60 Kč/km	0,80 Kč/km	1,40 Kč/km	2,00 Kč/km

4.2.7 Druhy vozidel

K zajištění přepravy brzdových součástek do jednotlivých provozoven musí firma Alfasped Logistik najmout vozidla od dopravců, přičemž na výběr mají čtyři nejčastěji používané druky vozidel v ČR, která jsou uvedena v následující tabulce 4 – 4.2.7 Druhy vozidel.

Z požadavků jednotlivých provozoven je zřejmé, že vozidlo o užitné hmotnosti do 3,5 t se zcela vylučuje, protože každý požadavek tuto hmotnost převyšuje. Vozidlo o užitné hmotnosti do 5 t můžeme taktéž vyloučit, protože část požadavků tuto hmotnost také převyšuje. Zbývají tedy pouze poslední dva druhy vozidel, které oba spadají do kategorie vozidel podléhajících elektronickému mýtnému na zpoplatněných úsecích pozemních komunikací ČR. Z výše uvedených důvodů bylo proto zvoleno vozidlo čtvrtého druhu o užitné hmotnosti do 24 t.

---

[69] Maximální povolená hmotnost vozidla je součtem užitné hmotnosti vozidla a hmotnosti vozidla samotného, proto i vozidlo o užitné hmotnosti do 10 t spadá do kategorie vozidel podléhajících mýtnému systému v ČR.

[70] *Dálniční známky a sazby mýtného pro dálnice, rychlostí silnice a silnice I.Třídy*.[on-line] (29.12.2008).URL:<http://business.center.cz/business/finance/dane/dalnice.aspx>

Tabulka 4 – 4.2.7 Druhy vozidel

Tabulka sazeb dopravních nákladů na 1 km		
Druh vozidla	**Kč bez DPH**	**Kilogramová sazba v Kč**
Vozidlo o užitné hmotnosti do 3,5 t	13,50	0,0039
Vozidlo o užitné hmotnosti do 5 t	14,00	0,0028
Vozidlo o užitné hmotnosti do 10 t	19,00	0,0019
Vozidlo o užitné hmotnosti do 24 t	22,00	0,0009

4.2.8 Výpočet tras

Z tabulky 1 Přehled provozoven s požadavky jsme zjistili, že celková suma požadavků je 64,0 t brzdových součástek. Vozidlo o užitné hmotnosti do 24 t jich uveze nejvýše 24 t, tedy budeme potřebovat 3 vozidla.

4.2.8.1 Přepravní okruhy

Pomocí **Mayerovy metody** stanovíme jednotlivé okruhy přeprav. Principem této metody, jak již bylo v kapitole 3.5.4.1 Mayerova metoda popsáno, je, že v matici vzdáleností začínáme od nejvzdálenějšího místa k místu výchozímu. K již vybranému místu pak přiřazujeme další místo, které má k němu nejmenší vzdálenost. Chceme-li přidat další místo do okružní trasy musíme nejdříve provést součet přepravních požadavků vybraných míst a porovnat ji s možnou přepravní kapacitou námi zvoleného vozidla. Pokud ještě není kapacita vozu naplněna můžeme přiřadit podle nejmenší vzdálenosti další místo do okruhu a opět provedeme porovnání součtu požadavků s maximální možnou kapacitou dopravního vozidla. Takto postupujeme až do naplnění kapacity vozu.

Další okruh začínáme opět nejvzdálenějším ze zbylých míst a pokračujeme obdobným způsobem dokud nezařadíme všechna města do přepravních okruhů.

Tabulka 5 – 4.2.8.1 Přepravní okruhy

1	Liberec	Ostrava	Opava	Olomouc	Pardubice	Hradec Králové		Celkem
		5,2	4,0	5,2	4,5	4,0		22,9
2	Liberec	Cheb	Plzeň	Klatovy	Písek	Kladno		
		4,5	5,2	4,0	5,2	4,5		23,4
3	Liberec	Jihlava	Praha	Ústí nad Lab	Teplice			
		5,2	4,0	4,0	4,5			17,7
							Celkem	64,0

	První okruh					
	Liberec	Ostrava	Opava	Olomouc	Pardubice	Hradec Králové
Liberec	x	335	300	246	118	97
Ostrava	335	x	35	93	240	240
Opava	300	35	x	74	210	205
Olomouc	246	93	74	x	147	149
Pardubice	118	240	210	147	x	21
Hradec Krá	97	240	205	149	21	x
	Druhý okruh					
	Liberec	Cheb	Plzeň	Klatovy	Písek	Kladno
Liberec	x	247	196	238	207	133
Cheb	247	x	101	130	180	156
Plzeň	196	101	x	42	81	87
Klatovy	238	130	42	x	75	129
Písek	207	180	81	75	x	114
Kladno	133	156	87	129	114	x
	Třetí okruh					
	Liberec	Jihlava	Praha	Ústí nad La	Teplice	
Liberec	x	185	102	92	102	
Jihlava	185	x	123	215	217	
Praha	102	123	x	92	94	
Ústí nad La	92	215	92	x	19	
Teplice	102	217	94	19	x	

Jak je na tabulce 5 – 4.2.8.1 Přepravní okruhy patrno, byly vybrány celkem 3 okruhy vždy s počátečním městem Liberec:

1) Liberec, Ostrava, Opava, Olomouc, Pardubice, Hradec Králové
2) Liberec, Cheb, Plzeň, Klatovy, Písek, Kladno
3) Liberec, Jihlava, Praha, Ústí nad Labem, Teplice

4.2.8.2 Přepravní trasy

Z přepravních okruhů však ještě stále není patrno, jak by měla dopravní trasa vypadat a kudy má vozidlo jet, aby pokud možno najelo co nejmenší vzdálenost, k tomu můžeme užít dalších metod systémové a operační analýzy.

Při výpočtu jednotlivých tras použijeme dvojí postup. Nejprve se pokusíme vypočítat optimální okruh tradičně ručně a poté pomocí naprogramované aplikace v MS Excel pro výpočet okružní dopravní úlohy. Tento postup volíme proto, abychom si prakticky ukázali nejen ověření, že aplikace dochází ke stejnému optimálnímu řešení jako tradiční ruční postup, ale hlavně proto, abychom poukázali na složitost a časovou nákladnost ručního postupu oproti jednoduchost a rychlosti počítání pomocí aplikace.

Přepravní trasy vypočítané pomocí **metody nejbližšího souseda**. Principem této metody je, jak již bylo v kapitole 3.5.4.3 Metoda nejbližšího souseda popsáno, že si v matici vzdáleností zvolíme výchozí místo a z něj pokračujeme do dalšího nejbližšího místa, kde jsme ještě nebyli, dokud neprojedeme všechna místa. Nakonec se vracíme do výchozího místa. Výchozím místem postupně volíme všechna místa v matici. Ze všech takto nalezených tras volíme nejvýhodnější, má nejmenší součet vzdáleností.

Jak je v následující tabulce 6 – 4.2.8.2 Jednotlivé okruhy první trasy patrné, je způsob ručního výpočtu značně zdlouhavý, jelikož si musíme pro každé výchozí město vypočítat průběh okruhu a jeho vzdálenost, což se učiní celkem v šesti dílčích tabulkách. Posléze se porovnají jednotlivé celkové vzdálenosti a za optimální se vybere

ta nejkratší. Je tedy vybrán třetí okruh první trasy o délce 693 km. Takto se pak postupuje i pro zbylé dvě trasy.

Tabulka 6 – 4.2.8.2 Jednotlivé okruhy první trasy

První okruh							
	Liberec	Ostrava	Opava	Olomouc	Pardubice	Hradec Krá	
Liberec	x	335	300	246	118	97	
Ostrava	335	x	35	93	240	240	
Opava	300	35	x	74	210	205	
Olomouc	246	93	74	x	147	149	
Pardubice	118	240	210	147	x	21	
Hradec Krá	97	240	205	149	21	x	709 km
	Liberec	Ostrava	Opava	Olomouc	Pardubice	Hradec Krá	
Liberec	x	335	300	246	118	97	
Ostrava	335	x	35	93	240	240	
Opava	300	35	x	74	210	205	
Olomouc	246	93	74	x	147	149	
Pardubice	118	240	210	147	x	21	
Hradec Krá	97	240	205	149	21	x	709 km
	Liberec	Ostrava	Opava	Olomouc	Pardubice	Hradec Krá	
Liberec	x	335	300	246	118	97	
Ostrava	335	x	35	93	240	240	
Opava	300	35	x	74	210	205	
Olomouc	246	93	74	x	147	149	
Pardubice	118	240	210	147	x	21	
Hradec Krá	97	240	205	149	21	x	693 km
	Liberec	Ostrava	Opava	Olomouc	Pardubice	Hradec Krá	
Liberec	x	335	300	246	118	97	
Ostrava	335	x	35	93	240	240	
Opava	300	35	x	74	210	205	
Olomouc	246	93	74	x	147	149	
Pardubice	118	240	210	147	x	21	
Hradec Krá	97	240	205	149	21	x	713 km
	Liberec	Ostrava	Opava	Olomouc	Pardubice	Hradec Krá	
Liberec	x	335	300	246	118	97	
Ostrava	335	x	35	93	240	240	
Opava	300	35	x	74	210	205	
Olomouc	246	93	74	x	147	149	
Pardubice	118	240	210	147	x	21	
Hradec Krá	97	240	205	149	21	x	713 km
	Liberec	Ostrava	Opava	Olomouc	Pardubice	Hradec Krá	
Liberec	x	335	300	246	118	97	
Ostrava	335	x	35	93	240	240	
Opava	300	35	x	74	210	205	
Olomouc	246	93	74	x	147	149	
Pardubice	118	240	210	147	x	21	
Hradec Krá	97	240	205	149	21	x	734 km

Tabulka 7 – 4.2.8.2 Přepravní trasy pomocí metody nejbližšího souseda – ruční postup

Trasa ->							Délka
Liberec	Hradec Králo	Pardubice	Olomouc	Opava	Ostrava	Liberec	709
Ostrava	Opava	Olomouc	Pardubice	Hradec Králové	Liberec	Ostrava	709
Opava	Ostrava	Olomouc	Pardubice	Hradec Králové	Liberec	Opava	693
Olomouc	Opava	Ostrava	Pardubice	Hradec Králové	Liberec	Olomouc	713
Pardubice	Hradec Králo	Liberec	Olomouc	Opava	Ostrava	Pardubice	713
Hradec Králo	Pardubice	Liberec	Olomouc	Opava	Ostrava	Hradec Kra	734

Trasa ->							Délka
Liberec	Kladno	Plzeň	Klatovy	Písek	Cheb	Liberec	764
Cheb	Plzeň	Klatovy	Písek	Kladno	Liberec	Cheb	712
Plzeň	Klatovy	Písek	Kladno	Liberec	Cheb	Plzeň	712
Klatovy	Plzeň	Písek	Kladno	Liberec	Cheb	Klatovy	747
Písek	Klatovy	Plzeň	Kladno	Liberec	Cheb	Písek	764
Kladno	Plzeň	Klatovy	Písek	Cheb	Liberec	Kladno	764

Trasa ->							Délka
Liberec	Ústí nad Lab	Teplice	Praha	Jihlava	Liberec		513
Jihlava	Praha	Ústí nad Lab	Teplice	Liberec	Jihlava		521
Praha	Ústí nad Lab	Teplice	Liberec	Jihlava	Praha		521
Ústí nad Lab	Teplice	Praha	Liberec	Jihlava	Ústí nad Labem		615
Teplice	Ústí nad Lab	Liberec	Praha	Jihlava	Teplice		553
Kladno	Plzeň	Klatovy	Písek	Cheb	Liberec	Kladno	764

Pomocí tohoto postupu jsme dostali tyto nejvýhodnější trasy, s tím, že počínáme v Liberci (sklady firmy TRW), kam se na konec i vracíme:

1) Liberec, Opava, Ostrava, Olomouc, Pardubice, Hradec Králové, Liberec – celkem 693 km
2) Liberec, Cheb, Plzeň, Klatovy, Písek, Kladno, Liberec – celkem 712 km
3) Liberec, Ústí nad Labem, Teplice, Praha, Jihlava, Liberec – celkem 513 km

Totožných tras, při jejich srovnání od výchozího města Liberec, do nějž se pak na závěr trasy vracíme, jsme dosáhli i pomocí naprogramovaného modulu pro řešení okružního dopravního problému v MS Excel nepoměrně rychlejším a jednodušším způsobem.

Jak je v tabulce 8 – 4.2.8.2 Přepravní trasy pomocí metody nejbližšího souseda – pomocí modulu patrné, jedná se o totožné výsledné okruhy jako při ručním postupu, přičemž jsou přehledně uspořádány a navíc jsou zobrazeny i kilometrové délky jednotlivých úseků vypočtených okruhů.

Tabulka 8 – 4.2.8.2 Přepravní trasy pomocí metody nejbližšího souseda – pomocí modulů

První trasa			
Metoda nejbližšího souseda			
Celkem měst:			6
Celková vzdálenost okruhu v km:			693
Jednotlivé úseky trasy:			km
Opava	--->	Ostrava	35
Ostrava	--->	Olomouc	93
Olomouc	--->	Pardubice	147
Pardubice	--->	Hradec Králové	21
Hradec Králové	--->	Liberec	97
Liberec	--->	Opava	300

Druhá trasa			
Metoda nejbližšího souseda			
Celkem měst:			6
Celková vzdálenost okruhu v km:			712
Jednotlivé úseky trasy:			km
Cheb	--->	Plzeň	101
Plzeň	--->	Klatovy	42
Klatovy	--->	Písek	75
Písek	--->	Kladno	114
Kladno	--->	Liberec	133
Liberec	--->	Cheb	247

Třetí trasa			
Metoda nejbližšího souseda			
Celkem měst:			5
Celková vzdálenost okruhu v km:			513
Jednotlivé úseky trasy:			km
Liberec	--->	Ústí nad Labem	92
Ústí nad Labem	--->	Teplice	19
Teplice	--->	Praha	94
Praha	--->	Jihlava	123
Jihlava	--->	Liberec	185

Dále se pokusíme přepravní trasy vypočítat pomocí **Vogelovy metody.**Metoda Vogelova (metoda VAM), jak již bylo v kapitole 3.5.4.2 Vogelova metoda popsáno, využívá rozdílů mezi velikostí sazeb v řadách dopravní tabulky, a tím zajišťuje obsazení velmi výhodných spojů rovnoměrně v průběhu celého algoritmu.

Principem této metody je, že v každém řádku a sloupci matice vzdáleností určíme rozdíl mezi dvěma nejmenšími vzdálenostmi, diference. V řádku nebo sloupci s nejvyšší diferencí obsadíme políčko s nejmenší vzdáleností a z tabulky vyškrtneme daný řádek a sloupec. Řádkové a sloupcové diference přepočítáme a pokračujeme obdobným způsobem. Z matice vzdáleností vyškrtáváme i pozice, které by vedly k předčasnému uzavření přepravního okruhu. Algoritmus končí projetím všech míst.

Tabulka 9 – 4.2.8.2 Přepravní trasy pomocí Vogelovy metody – ruční postup

První okruh										
	Liberec	Ostrava	Opava	Olomouc	Pardubice	Hradec Krá				
Liberec	x	335	300	246	118	97	21	128	54	54
Ostrava	335	x	35	93	240	240	58	58	58	58
Opava	300	35	x	74	218	265	39	39	39	39
Olomouc	246	93	74	x	147	149	19	19	19	
Pardubice	118	240	218	147	x	21	97			
Hradec Krá	97	240	265	149	21	x	76			
	21	58	39	19	97	76				
	149	58	39	19	29					
		58	39	19	63					
		300	265	19						
Opava	Ostrava	Olomouc	Pardubice	Hradec Krá	Liberec	Opava	693			

Druhý okruh										
	Liberec	Cheb	Plzeň	Klatovy	Písek	Kladno				
Liberec	x	247	196	230	287	133	63			
Cheb	247	x	101	130	188	196	29	29	50	67
Plzeň	196	101	x	42	81	87	39	39	20	95
Klatovy	230	130	42	x	75	129	33	33		
Písek	287	188	81	75	x	114	6	6	105	
Kladno	133	196	87	129	114	x	27	27	15	42
	63	29	39	33	6	27				
	11	29	39	33	6					
	11	55		54	33					
	51	55			66					
Kladno	Písek	Klatovy	Plzeň	Cheb	Liberec	Kladno	712			

Třetí okruh								
	Liberec	Jihlava	Praha	Ústí nad La	Teplice			
Liberec	x	185	102	92	182	10	0	83
Jihlava	185	x	123	215	217	62	62	
Praha	102	123	x	92	94	2	8	8
Ústí nad La	92	215	92	x	19	73	0	123
Teplice	182	217	94	19	x	75		
	10	62	2	73	75			
	10	62	10		8			
	10	30			8			
Teplice	Ústí nad Lab	Liberec	Jihlava	Praha	Teplice	513		

Z tabulek 9 – 4.2.8.2 a 10 – 4.2.8.2 je opět patrný rozdíl ve složitosti jednotlivých postupů (ručního a pomocí modulů) a v jejich přehlednosti i předpokládané časové náročnosti. Je naprosto zřejmé, že pro manažery operativní úrovně popřípadě špeditéry, kteří rozhodují o trase pronajatého vozidla od dopravce, je daleko snazší využít aplikace v MS Excel než se mořit s ručním výpočtem.

Tabulka 10 – 4.2.8.2 Přepravní trasy pomocí Vogelovy metody – pomocí modulů

První trasa

Vogelova metoda			
Celkem měst:			6
Celková vzdálenost okruhu v km:			693
Jednotlivé úseky trasy:			km
Pardubice	--->	Hradec Králové	21
Hradec Králové	--->	Liberec	97
Olomouc	--->	Pardubice	147
Opava	--->	Ostrava	35
Liberec	--->	Opava	300
Ostrava	--->	Olomouc	93

Druhá trasa

Vogelova metoda			
Celkem měst:			6
Celková vzdálenost okruhu v km:			747
Jednotlivé úseky trasy:			km
Liberec	--->	Kladno	133
Plzeň	--->	Klatovy	42
Písek	--->	Plzeň	81
Klatovy	--->	Cheb	130
Cheb	--->	Liberec	247
Kladno	--->	Písek	114

Třetí trasa

Vogelova metoda			
Celkem měst:			5
Celková vzdálenost okruhu v km:			513
Jednotlivé úseky trasy:			km
Teplice	--->	Ústí nad Labem	19
Jihlava	--->	Praha	123
Ústí nad Labem	--->	Liberec	92
Liberec	--->	Jihlava	185
Praha	--->	Teplice	94

Pomocí této metody jsme tedy vypočetli tyto trasy, s tím, že počínáme ve městě skladů firmy TRW v Liberci, kam se na konec i vracíme:

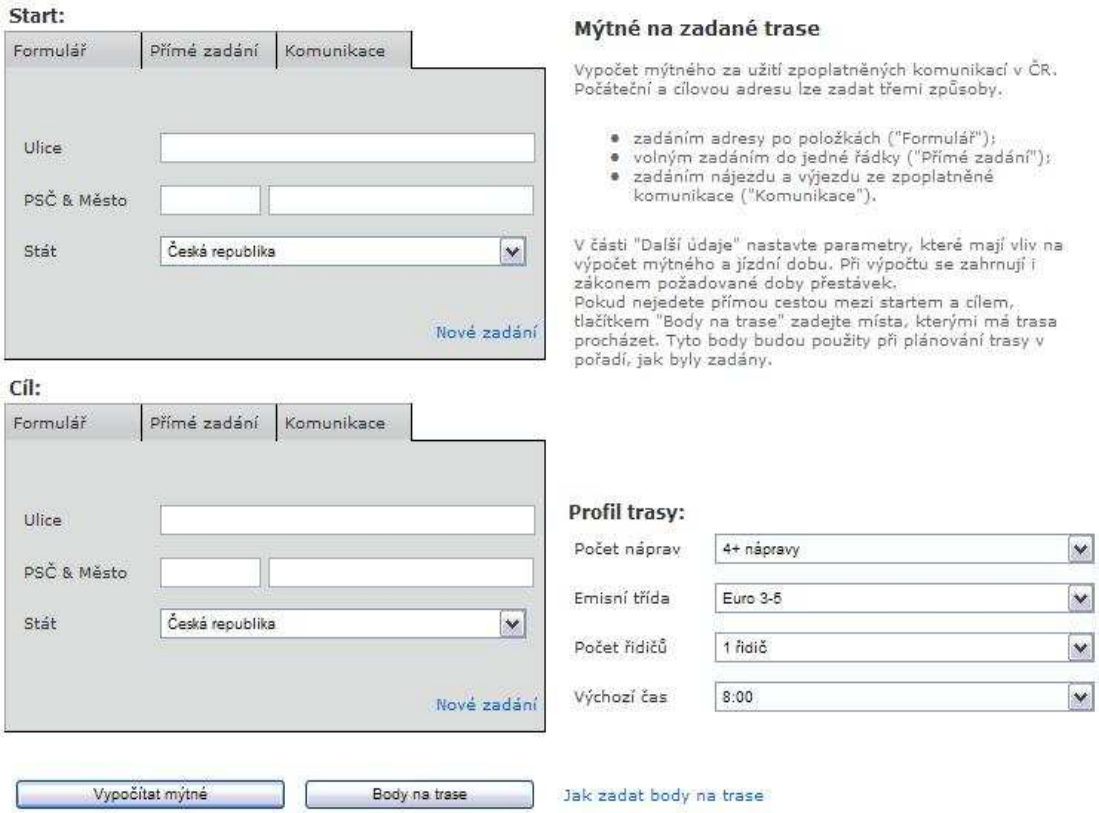
1) Liberec, Opava, Ostrava, Olomouc, Pardubice, Hradec Králové, Liberec – celkem 693 km
2) Liberec, Kladno, Písek, Klatovy, Plzeň, Cheb, Liberec, – celkem 712 km
3) Liberec, Jihlava, Praha, Teplice, Ústí nad Labem, Liberec – celkem 513 km

Pomocí Vogelovy metody jsme vypočítali stejné trasy jako pomocí Metody nejbližšího souseda, přičemž druhá a třetí trasy je shodná, ale v opačné posloupnosti měst.

4.2.9 Výpočet mýtného

Na začátku roku 2007 byl v ČR zprovozněn na dálnicích a rychlostních komunikacích mýtný systém, který se týká silničních motorových vozidel, které mají nejméně 4 kola a maximální povolenou hmotnost 12 a více tun. Vybrané vozidlo pro přepravu brzdových součástek o užitné hmotnosti 24 t do této kategorie spadá, proto se ho placení mýtného na zpoplatněných úsecích pozemních komunikací ČR týká.Výpočet mýtného pro jednotlivé trasy přepravy byl proveden pomocí webové aplikace MYTO CZ[71], jejíž dialogové oknu vstupního formuláře je zobrazeno na obr. 14 – 4.2.9.

Obrázek 14 4.2.9 Aplikace MYTO CZ pro výpočet mýtného



[71] *Výpočet mýtného a informace*.[on-line].(29.12.2008).URL: <http://tollcz.ptv.de/premid/ti/index.jsf?Locale =cs_CZ>

Obrázek 15 – 4.2.9 Mýtné pro 1. trasu

Itinerář:

Odjezd:	1.1.09 8:00	Příjezd:	2.1.09 13:39
Start:	46* *** Liberec	Cíl:	46* *** Liberec
přes:	74* *** Opava		
	7** *** Ostrava		
	7** *** Olomouc		
	53* *** Pardubice		
	50* *** Hradec Kralove		
Doba:	29:39.21	Jízdní doba zahrnuje přestávky požadované zákonem o délce: 12.5 h	
Zpoplatněná délka:	137,7 km		
Mýtné:	461,52 Kč		

Schéma trasy

Mýtný poplatek pro první trasu (Liberec, Opava, Ostrava, Olomouc, Pardubice, Hradec Králové, Liberec), jejíž průběh je znázorněn na obrázku 15 – 4.2.9 Mýtné pro 1. trasu, o vzdálenosti 693 km při zpoplatnění 137,7 km trasy tedy činí 461,52 Kč.

Obrázek 16 – 4.2.9 Mýtné pro 2. trasu

Itinerář:

Odjezd:	1.1.09 6:00	Příjezd:	2.1.09 10:36
Start:	46* *** Liberec	Cíl:	Liberec
přes:	35* *** Cheb		
	3** *** Plzen		
	339 01 Klatovy		
	397 01 Pisek		
	27* *** Kladno		
Doba:	28:36.59	Jízdní doba zahrnuje přestávky požadované zákonem o délce: 12.5 h	
Zpoplatněná délka:	418,6 km		
Mýtné:	1 758,12 Kč		

Schéma trasy

Mýtný poplatek druhou trasu (Liberec, Cheb, Plzeň, Klatovy, Písek, Kladno, Liberec)), jejíž průběh je znázorněn na obrázku 16 – 4.2.9 Mýtné pro 2. trasu, o vzdálenosti 712 km při zpoplatnění 418,6 km trasy tedy činí 1 758,12 Kč.

Obrázek 17 – 4.2.9 Mýtné pro 3. trasu

Mýtný poplatek pro třetí trasu (Liberec, Ústí nad Labem, Teplice, Praha, Jihlava, Liberec), jejíž průběh je znázorněn na obrázku 15 – 4.2.9 Mýtné pro 3. trasu, o vzdálenosti 513 km při zpoplatnění 383,7 km trasy tedy činí 1 611,54 Kč.

Celkové náklady na mýtné tedy činí:

1. Trasa - 461,52 Kč
2. Trasa - 1 758,12 Kč
3. Trasa - 1 611,54 Kč

Celkem = **3 831,18** Kč

4.2.10 Optimalizace přepravních tras

Po předložení výsledných okruhů příslušným špeditérům firmy Alfasped Logistik, však bylo rozhodnuto, že ačkoliv početně ano, vzhledem k mýtnému systému v ČR nejsou zcela optimální.

Jak je patrno na obrázku 17 – 4.2.9 Mýtné pro 3. trasu, je třetí trasa značně nesourodá, Jihlava se do tohoto okruhu příliš nehodí protože celý úsek mezi Jihlavou a Prahou je zpoplatněn a projíždí se de facto dvakrát, prohodíme Jihlavu s Pardubicemi z první trasy. Tím se vytvoří ucelenější okruh třetí trasy a zamezí se zbytečnému projetí zpoplatněného úseku trasy. Nové trasy jsou znázorněny v následující tabulce 11- 4.2.10.

Tabulka 11 – 4.2.10 Optimalizované přepravní okruhy

Liberec	Ostrava		Opava	Olomouc	Jihlava	Hradec Králové	Celkem
	5,2		4,0	5,2	5,2	4,0	23,6
Liberec	Cheb		Plzeň	Klatovy	Písek	Kladno	
	4,5		5,2	4,0	5,2	4,5	23,4
Liberec	Pardubice		Praha	Ústí nad L	Teplice		
	4,5		4,0	4,0	4,5		17,0
						Celkem	64,0

Vzdálenosti a pořadí měst projetí prvního a třetího optimalizovaného okruhu vypočteme opět pomocí Vogelovy metody, jejíž výsledek je znázorněn v tabulkách 12 – 4.2.10 a 13 – 4.2.10.

Tabulka 12 – 4.2.10 Optimalizované přepravní trasy pomocí Vogelovy metody – ruční postup

První okruh										
	Liberec	**Ostrava**	**Opava**	**Olomouc**	**Jihlava**	**Hradec Krá**				
Liberec	x	335	300	246	185	97	**88**			
Ostrava	335	x	35	93	253	240	58	58	58	58
Opava	300	35	x	74	240	205	39	39	39	39
Olomouc	246	93	74	x	166	149	19	19	19	
Jihlava	185	253	240	166	x	110	56	19		
Hradec Krá	97	240	205	149	110	x	13	39	56	35
	88	58	39	19	56	13				
	61	58	39	19	56					
		58	39	19	**74**					
		205	170	19						
Opava	Ostrava	Olomouc	Jihlava	Liberec	Hradec Krá	Opava	781			

Třetí okruh								
	Liberec	**Pardubice**	**Praha**	**Ústí nad L**	**Teplice**			
Liberec	x	118	102	92	102	10	0	0
Pardubice	118	x	104	196	198	14	14	80
Praha	102	104	x	92	94	2	8	
Ústí nad La	92	196	92	x	19	73	0	0
Teplice	102	190	94	19	x	**75**		
	10	14	2	73	75			
	10	**14**	10		8			
	26		10		**96**			
	Praha	Pardubice	Liberec	Teplice	Ústí nad L	Praha	435	

Tabulka 13 – 4.2.10 Optimalizované přepravní trasy pomocí Vogelovy metody – pomocí modulů

První okruh - Optimalizovaný			
Vogelova metoda			
Celkem měst:			6
Celková vzdálenost okruhu v km:			**781**
Jednotlivé úseky trasy:			km
Liberec	--->	Hradec Krá	97
Jihlava	--->	Liberec	185
Olomouc	--->	Jihlava	166
Opava	--->	Ostrava	35
Hradec Krá	--->	Opava	205
Ostrava	--->	Olomouc	93

Třetí okruh - Optimalizovaný			
Vogelova metoda			
Celkem měst:			5
Celková vzdálenost okruhu v km:			**427**
Jednotlivé úseky trasy:			km
Teplice	--->	Ústí nad Labem	19
Pardubice	--->	Praha	104
Ústí nad Labem	--->	Liberec	92
Liberec	--->	Pardubice	118
Praha	--->	Teplice	94

Optimalizované okruhy jsou tedy následující:

1) Liberec, Hradec Králové, Opava, Ostrava, Olomouc, Jihlava, Liberec – celkem 781 km
2) Liberec, Teplice, Ústí nad Labem, Praha, Pardubice, Liberec – celkem 435 km

Pro nově vypočtené trasy spočítáme výši mýtného opět pomocí webové aplikace MYTO CZ.

Obrázek 18 Mýtné pro 1. optimalizovanou trasu

Mýtný poplatek pro první trasu po optimalizaci (Liberec, Hradec Králové, Opava, Ostrava, Olomouc, Jihlava, Liberec) o vzdálenosti 781 km při zpoplatnění 175,3 km trasy tedy činí 619,44 Kč.

Obrázek 19 Mýtné pro 3. optimalizovanou trasu

Mýtný poplatek pro třetí trasu (Liberec, Teplice, Ústí nad Labem, Praha, Pardubice, Liberec) o vzdálenosti 435 km při zpoplatnění 174,3 km trasy tedy činí 700,38 Kč.

Nyní je již třetí okruh daleko vhodnější a sourodý, tzn. zbytečně se nezajíždí do měst, ze kterých pak není plynulá návaznost na město následující a tím se vyhneme i zbytečnému projetí úseku trasy zpoplatněného mýtem.

Celkové náklady na mýtné tedy činí:

1. Trasa – 619,44 Kč
2. Trasa – 1 758,12 Kč
3. Trasa – 700,38 Kč

Celkem = **3 077,94** Kč

4.2.11 Výpočet přepravních nákladů

4.2.11.1 Dopravní náklady

Dopravní náklady (transportation cost) na trasu ve vztahu k vzdálenosti: [72]

$TTC = c_d * d + (c'_{d1} * d_1' + c'_{d2} * d_2')$

c_d ..... sazba nákladů na dopravu na km

c'_{d1} ..... sazba nákladů na mýtné na km zpoplatněných dálnic a rychlostních silnic

c'_{d2} ..... sazba nákladů na mýtné na km zpoplatněných silnic I. třídy

d..... vzdálenost

d_1' ..... vzdálenost zpoplatněného úseku dálnic a rychlostních silnic

d_2' ..... vzdálenost zpoplatněného úseku silnic I. Třídy

1) Liberec – Opava – Ostrava – Olomouc – Pardubice – Hradec Králové – Liberec

Celkem 693 km, sazba 22 Kč/km => 693 * 22 = 15 246,- Kč
Mýtné 461,52 Kč
TC_1 = 15 246 + 461,52 = 15 707,52 Kč

2) Liberec – Cheb – Plzeň – Klatovy – Písek – Kladno – Liberec

Celkem 712 km, sazba 22 Kč/km => 712 * 22 = 15 664,- Kč
Mýtné 1 758,12 Kč
TC_2 = 15 664 + 1 758,12 = 17 422,12 Kč

3) Liberec – Ústí nad Labem – Teplice – Praha – Jihlava – Liberec

Celkem 513 km, sazba 22 Kč/km => 513 * 22 = 11 286,- Kč
Mýtné 1 611,54 Kč
TC_3 = 11 286 + 1 611,54 = 12 897, 54 Kč

Náklady na přepravu celkem: $TTC = TC_1 + TC_2 + TC_3$

TTC = 15 707,52 + 17 422,12 + 12 897, 54 = **46027,18**Kč

[72] V tomto zjednodušením případě neuvažujeme náklady při zastavení, dodatečné náklady na jednotku vzdálenosti znamenají náklady na mýtné, zjištěné pomocí aplikace MYTO CZ v kapitole 4.2.9.

4.2.11.2 Manipulační náklady

Manipulační náklady (handling cost) na trasu: TLC = c * v

c ..... sazba manipulačních nákladů na paletu

v ..... množství (počet palet)

1) Liberec – Opava – Ostrava – Olomouc – Pardubice – Hradec Králové – Liberec

Manipulační sazba 70,- Kč / paleta, přepravované množství zboží 22,9 t, 1 paleta cca. 1t
22,9 t => 23 palet LC_1 = 23* 70 = 1 610,- Kč

2) Liberec – Cheb – Plzeň – Klatovy – Písek – Kladno – Liberec

Manipulační sazba 70,- Kč / paleta, přepravované množství zboží 23,4 t, 1 paleta cca. 1 t
23,4 t => 24 palet LC_2 = 24 * 70 = 1 680,- Kč

3) Liberec – Ústí nad Labem – Teplice – Praha – Jihlava – Liberec

Manipulační sazba 70,- Kč / paleta, přepravované množství zboží 17,7 t, 1 paleta cca. 1 t
17,7 t => 18 palet LC_3 = 18 * 70 = 1 260,- Kč

Náklady na manipulaci celkem: $TCL = LC_1 + LC_2 + LC_3$
TCL = 1 610 + 1 680 + 1 260 = **4 550,-** Kč

4.2.11.3 Přepravní náklady celkem

Náklady přepravní celkem (motion cost): TMC = TTC * p + TLC
TMC = 46 027,18 + 4 550 = **50 577,18** Kč

4.2.12 Optimalizované přepravní náklady

1) Liberec – Hradec Králové – Opava – Ostrava – Olomouc – Jihlava – Liberec

Celkem 781 km, sazba 22 Kč/km => 781 * 22 = 17 182,- Kč
Mýtné 619,44 Kč
TC_1 = 15 246 + 619,44 = 15 865,44 Kč

2) Trasa je beze změny, čili: TC_2 = 15 664 + 1 758,12 = 17 422,12 Kč

3) Liberec – Teplice – Ústí nad Labem – Praha – Pardubice – Liberec

Celkem 435 km, sazba 22 Kč/km => 435 * 22 = 9 570,- Kč
Mýtné 700,38 Kč
TC_3 = 11 286 + 700,38 = 11 986, 38 Kč
TTC = 15 865,44 + 17 422,12 + 11 986, 38 = **45 273,94** Kč

Manipulační náklady zůstávají stejné jako před optimalizací, mění se pouze výše celkových přepravních nákladů:

TMC = 45 273,94 + 4 550 = **49 823,94** Kč

Rozdíl optimalizovaných a původních nákladů činí:

50 577,18 - 49 823,94 = 753,24 KČ

Ačkoliv se celkem za všechny tři trasy dohromady najede o 10 km více, celkem 1928 km oproti předchozím 1918 km, z důvodu zpoplatnění pouze některých úseků pozemních komunikací (dálnic, rychlostních silnic a silnic I. třídy) jsou celkové přepravní náklady o 753,24 Kč nižší, protože správným výběrem dopravních tras můžeme ovlivnit nejen rychlost dopravy, ale i délku zpoplatněných úseků, jimiž budeme muset projet.

4.2.13 Závěr případové studie

Celkové přepravní náklady firmy Alfasped Logistik pro zajištění rozvozu celkem 64,0 t brzdových součástek do 15 provozoven (viz tabulka 1 – 4.2.3 Přehled provozoven s požadavky) pomocí pronajatých tří nákladních vozidel o užitné hmotnosti 24 t od dopravců jsou 49 823,94 Kč. Z toho si firma účtuje provizi 12 %, vyfakturuje firmě TRW za distribuci brzdových součástek celkem **55 802,81** Kč.

Použitím metod operační a systémové analýzy, v našem případě Mayerovy metody, metody nejbližšího souseda a Vogelovy metody, při řešení problémů přepravy, lze nacházet výhodná řešení, jejichž výběrem můžeme účinně snižovat dopravní náklady a tím i celkové náklady na přepravu zboží a materiálů.

V tomto případě jsme díky použití těchto metod docílili výběru nejvhodnějších okružních tras (viz kapitola 4.2.8.2 Přepravní trasy) pro přepravu brzdových součástek, a tím snížili nejen celkovou výši kilometrových nákladů samotných, ale i nákladů na elektronické mýtné, což bylo zlepšeno i následnou optimalizací první a třetí trasy jak je vypočteno v kapitole 4.2.10 Optimalizace přepravních tras.

4.3 Moduly pro řešení okružního dopravního problému jako prvky systému pro podporu rozhodování

4.3.1 Systémy pro podporu rozhodování

Systémy na podporu rozhodování (z angl. *Decision Support Systems* – DSS) pomáhají svým uživatelům (manažerům) při realizaci řídících a rozhodovacích činností. Uživatel tu může srovnávat dílčí výsledky řešení se svými představami a podle toho ovlivňovat další průběh řešení.[73]

Systémy pro podporu rozhodování poskytují uživateli nabídky řešení a mohou i případným kladením otázek a dotazů usměrňovat jeho postup. Tyto systémy však nikdy nemohou zcela nahradit rozhodovatele, jejich výsledkem tedy není samotné rozhodnutí, ale vytvářejí pro rozhodovatele pouze přehled možností a variant, urychlují a zpřesňují propočty jejich důsledků a kvantifikují rizika.

V našem případě se tedy jedná o to, navrhnout pro koncového uživatele resp. speditéra určitý systém, který by mu pomáhal stanovit jednotlivé trasy při řešení okružního dopravního problému. A jako součást tohoto systému by se právě mohly využít naprogramované moduly v MS Excel v jazyce Visual Basic for Applications.

Tento systém by mohl být snadno rozšiřitelný i o jiné moduly sloužící k výpočtům v ostatních oblastech logistiky, např.: přidání modulu pro výpočet jednostupňové dopravní úlohy nebo přidáním modulu pro výpočet dvoustupňové dopravní úloze by bylo možné vypočítat vhodné umístění meziskladů či stanovit jejich optimální kapacity.

Nyní se tedy uveďme jednotlivé charakteristiky pomyslného systému z hlediska okružního dopravního problému.

4.3.1.1 Obecná charakteristika problému

Jak je tedy z předešlých kapitol patrno, celý systém pro podporu speditérova rozhodování by byl koncipován do oblasti dopravní logistiky. Základní charakteristiky oblasti tedy jsou:

- Oblast distribuce a dopravy
- Okružní dopravní problém
- Podpora při stanovování tras z pohledu „čisté“ spedice

4.3.1.2 Rizika

Pro tvorbu DSS je nutné mít na paměti i možná rizika dané oblasti, kterých by se mohl uživatel dopustit a kterých, jsou-li v systému dobře podchyceny, se systém musí vyvarovat. Správným definováním rizik se docílí toho, že můžeme systém pro podporu rozhodování navrhnout tak, že se jich nejen nedopustí, ale bude je předcházet a uživatele před nimi i varovat. Základní příčiny rizik v této oblasti tedy jsou:

- Velké množství možných tras

[73] *Systémy pro podporu rozhodování - Wikipedie, otevřená encyklopedie* [on-line].(5.4.2009).URL: <http://cs.wikipedia.org/wiki/Syst%C3%A9my_pro_podporu_rozhodov%C3%A1n%C3%AD>

- Velké množství navzájem kombinovatelných úseků trasy
- Rizika:
    - Vynechání úseku trasy
    - Možná duplicita úseku trasy
    - Špatná volba trasy – vysoké náklady na přepravu

4.3.1.3 Úkol systému

Úkolem tohoto systému je splňovat požadavky koncového uživatele, u pracovníků spedice se tedy jedná o:

- Možnost výběru měst
- Kilometrovník + Sazby mýtného
- Návrh tras
    - Každým místem (úsekem trasy) projet právě jednou
    - Minimální vzdálenost
    - Minimální přepravní náklady
    - Časově nejkratší trasa (s přihlédnutím k časovému faktoru)
- Možnost modifikace trasy – změna vstupních dat

4.3.1.4 Omezení systému

Žádný systém nemůže fungovat bez jakýchkoli omezení, všechny systému jsou navrhovány a tvořeny pro konkrétní prostředí a uživatele, stejně tak i tento systém pro podporu rozhodování , jeho omezení tedy jsou:

- Vkládání dat pouze v předem definovaném formátu (toto omezení by bylo v systému již ošetřeno tím, že by uživatel pouze vybíral města v okruhu a strukturovanou matici sazeb by si již tvořil systém sám)
- Omezení počtu vkládaných míst (toto omezení se týká modulů pro výpočet trasy, zde je proměnná Mesta definovaná jako datový typ Byte, tedy max 255 měst, pomocí změny datového typu lze toto omezení taktéž omezit)

4.3.1.5 Řešení

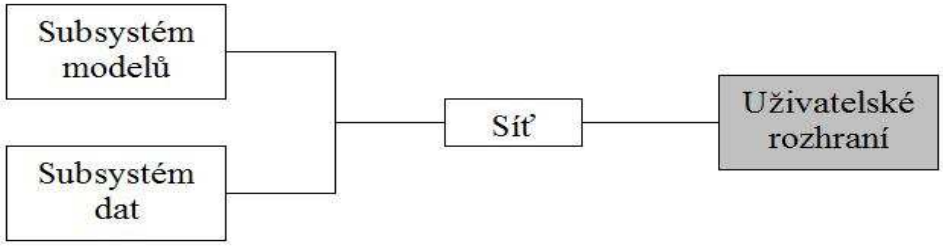
Výsledkem celého systému je pak návrh řešení, která slouží speditérovi jako podpora pro jeho konečné řešení jako hlavního rozhodovatele.

- Exaktní přesné vyhledání tras – 3 řešení – podpora v rozhodování
    - Minimální vzdálenost (trasa o nejkratší vzdálenosti mezi jednotlivými městy)
    - Minimální přepravní náklady (trasa s nejmenšími dopravními náklady, v nich jsou zahrnuty i náklady na mýtné)
    - Časově nejkratší trasa (tras, jejíž projetí zabere dopravnímu vozidlu nejkratší dobu)
- Alternativ možno dosáhnout změnou vstupních dat

4.3.2 Struktura

V oblasti dopravní logistiky, kde se k řešení úloh využívají převážně kvantitativní metody, je vhodné použít modelově orientovaný systém pro podporu rozhodování, který nám umožňuje porovnávat jednotlivé modelové simulace, a to tím, že díky jednotlivým algoritmům výpočtů nám poskytuje exaktní kvantifikované návrhy řešení. Jednotlivá řešení lze mezi sebou porovnávat, a tak může koncový uživatel vybrat pro něj to nejvhodnější. Obecná struktura modelově orientovaného systému pro podporu rozhodování ze je znázorněna na obrázku 20 – 4.3.2, kde jsou patrné tři základní části systému, jsou to: subsystém modelů, subsystém dat a uživatelské rozhraní, všechny tři části jsou pak spojeny sítí, skrze kterou dochází k interakcím jednotlivých částí.

Obrázek 20 – 4.3.2 Obecná struktura modelově orientovaného DSS



Struktura celého systému pro podporu rozhodování v oblasti dopravní logistiky, konkrétně v problematice řešení okružního dopravního problému, která je znázorněna na obrázku 20 – 4.3.1.6 Struktura systému je členěna do tří základních subsystémů.

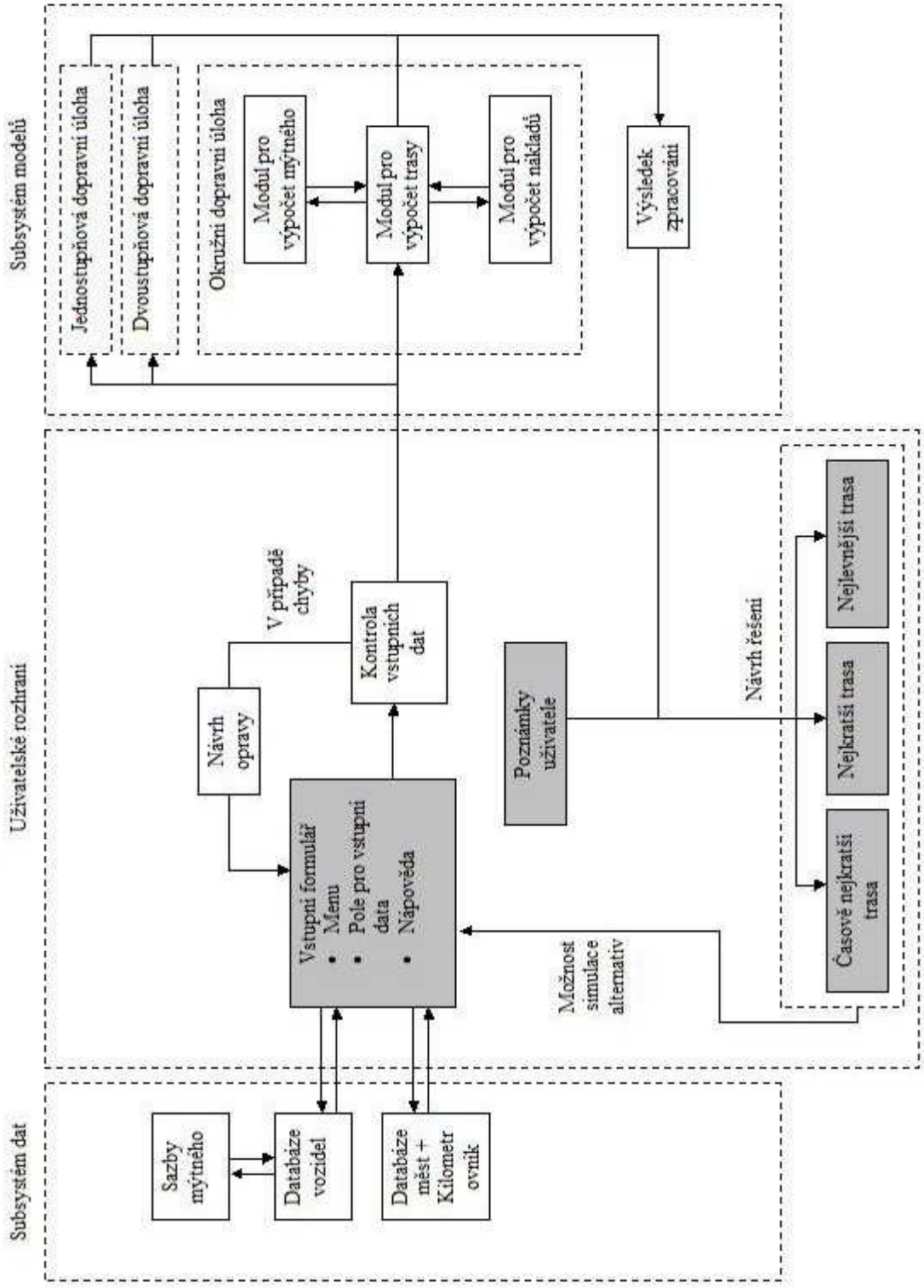
Pro uživatele je viditelný pouze subsystém uživatelského rozhraní, jehož prostřednictvím komunikuje uživatel se systémem. Pomocí vstupního formuláře vybírá uživatel jednotlivá města okružní trasy a volí druh vozidla, který by chtěl pro transport zboží či materiálů použít. Systém si pak sám dle voleb koncového uživatele (speditéra) načítá z datového subsystému jednotlivé údaje o druhu vozidel a jeho případné příslušnosti k zpoplatněným skupinám vozidel dle mýtného systému v ČR a z databáze měst a jejich vzájemných vzdáleností si utvoří strukturovanou matici sazeb, se kterou nadále počítá. Po načtení dat probíhá jejich kontrola, v případě chyby, např.: není-li matice sazeb úplná či není li některá sazba číselného charakteru, se uživatel pomocí hlášky o chybě dozví a je mu navrhnut scénář možné korekce.

Po zkontrolování jednotlivých údajů postupují do subsystému modelů, kde jsou na základě předešlé volby speditéra příslušným způsobem zpracovány. V tomto případě obsahuje subsystém modelů pouze tří hlavní moduly, pro výpočet trasy v jednostupňové dopravní úloze, dvoustupňové dopravní úloze a v okružní dopravní úloze, přičemž výstupem z tohoto subsystému je výsledek zpracování, který je v číselné podobě. K němu jsou následně dodány poznámky uživatele a celý výsledek pak vyúsťuje opět formulářem v uživatelském prostředí, kde jsou koncovému uživateli předkládány možná řešení, ze kterých pak vybírá.

Při změně vstupních údajů pak může uživatel simulovat alternativní řešení a navzájem je s prvotními porovnávat.

Výhodou tohoto systému by byla možnost implementace další výpočetních modulů do subsystému modelů. Tím by se zvýšila využitelnost systému i v ostatních oblastech, např.: úpravou sazeb v dvoustupňové dopravní úloze by bylo možné vypočítat vhodné umístění meziskladů či stanovit jejich optimální kapacity.

Obrázek 21 – 4.3.1.6 Struktura systému



5 Závěr

Na závěr této diplomové práce si můžeme shrnout poznatky z obou dvou hlavních částí této práce.

V první jsme se formou literární rešerše seznámili s problematikou okružního dopravního problému, jeho začleněním v dopravní logistice a s metodologií, která se pro jeho řešení dá použít, jakož i s algoritmy jednotlivých metod. Také jsme se obeznámili s prostředím MS Excel a jeho programovacího jazyka Visual Basic for Applications.

V druhé části jsme se věnovali samotnému programování modulů k řešení úloh a jejich praktického využití v reálné firmě.

Vytvořená aplikace pro řešení okružního dopravního problému se dá použít jako nástroj pro podporu rozhodování v oboru dopravní logistiky. Lze její pomocí nejen rychle a téměř optimálně určit trasu pro rozvoz popř. svoz různého zboží a materiálů, ale lze ji použít i pro řešení svozu komunálního odpadu či čištění vozovek a mnoha dalších situacích.

Přínosem této aplikace a tím i přínosem celé práce je právě, krom samotného vytvoření modulů pro výpočet okružních dopravních úloh, její využití v reálné situaci, a to ve skutečné firmě, zabývající se vnitrostátním a mezinárodním zasilatelstvím a každodenně řešící problémy dopravní logistiky. Snadné užívání a uživatelsky přívětivé prostředí napomáhá konečnému uživateli snadné ovládání celé aplikace. Slouží tedy reálně jako prostředek napomáhající každodennímu operativnímu rozhodování a usnadňuje práci speditérů. Důležitým přínosem je i díky optimalizaci přepravních tras úspora v oblasti dopravních nákladů a tím i zlepšení konkurenceschopnosti firmy na trhu dopravních služeb.

Dalším přínosem této aplikace je její použitelnost v případném komplexním systému pro podporu rozhodování v oblasti využití kvantitativních metod v logistickém rozhodování. Její pozice v tomto systému byla naznačena v kapitole 4.3 Moduly pro řešení okružního dopravního problému jako prvky systému pro podporu rozhodování.

Výsledkem této diplomové práce je tedy modul pro řešení okružního dopravního problému v prostředí MS Excel.

6 Seznam literatury

6.1 Seznam literatury

1) *Nový velký ilustrovaný slovník naučný*. Praha: Gutenberg, 1931, sv. XII.
2) *Příruční slovník naučný*. Praha: Nakladatelství Československé akademie věd, 1964, díl II.
3) *Všeobecná encyklopedie ve čtyřech svazcích*. Praha: Nakladatelský dům OP, 1997, díl 2
4) KORTSCHAK, B. H. *Úvod do logistiky - Co je logistika?*. 2. české vyd. Praha: Babtex, 1995, ISBN 80-85816-06-7
5) SIXTA, J. a MAČÁT, V. *Logistika - teorie a praxe*. 1. vyd. Brno: CP Books, 2005, ISBN80-251-0573-3
6) CAMPBELL, J. *Druhá světová válka*. 1. české vyd. Praha: Mladá fronta, 1995, ISBN 80-204-0474-0
7) GROS, I. *Logistika*. 1. vyd. Praha: VŠCHT v Praze, 1993, ISBN 80-7080-216-2
8) PERNICA, P. *Logistika (základy).* 1. vyd. Praha: VŠE v Praze, 1991, ISBN 80-7079-158-6
9) ZÍSKAL, J. a HAVLÍČEK, J. *Ekonomicko matematické metody II Studijní texty pro distanční studium*. 2. vyd. Praha: ČZU, 2003, ISBN 80-213-0664-5
10) DRAHOTSKÝ, I. a ŘEZNÍČEK, B. *Logistiky – procesy a jejich řízení*. 1. vyd. Brno: Computer Press, 2003, ISBN 80-7226-521-0
11) ŠUBRT, T., BROŽOVÁ, H., DÖMEOVÁ, L., KUČERA, P. *Ekonomicko matematické metody II, aplikace a cvičení*, Praha: ČZU, 2000
12) WALKENBACH, J. *Microsoft Office Excel 2000 & 2002 – Programování ve VBA*. Brno: Computer Press. 2006. ISBN 80-7226-547-4
13) BERKA, M. *Operační výzkum*, skripta VUT Brno, 1991
14) RYJÁČEK, Z. *Teorie grafů a diskrétní optimalizace 1, pracovní texty přednášek*, 2005
15) PELIKÁN, J. *Diskrétní modely v operačním výzkumu*, Professional Publishing, 2001
16) PELIKÁN, J. *Praktikum z operačního výzkumu*, Praha: VŠE, 1992
17) FRYE, C., FREEZE, W., BUCKINGHAM, F. K. *Microsoft Office Excel 2003 Programming Inside Out*, Microsoft Press, 2004
18) ČERNÝ, J. *Programování v Excelu 2000, 2002, 2003*, Grada, Praha, 2005

6.2 Elektronické zdroje

19) *Operation Overlord - Wikipedia, the free encyclopedia*. [on-line].(17.1.2007) URL:<http://en.wikipedia.org/wiki/File:NormandySupply_edit.jpg>

20) *Logistika - zdroj přidané hodnoty!*. [on-line].(17.1.2007) URL:<http://www.logistika.cz/index.php?menu=31>

21) GAULD, A. *Jak se naučit programovat*. [on-line].(18.3.2008) URL:<http://www.freenetpages.co.uk/hp/alan.gauld/czech3/index.html>

22) *Alfasped s.r.o.* [on-line].(6.3.2009)URL:<http://www.alfalog.cz/profil.html>

23) *Dálniční známky a sazby mýtného pro dálnice, rychlostí silnice a silnice I.Třídy*. [on-line]. (29.12.2008).URL:<http://business.center.cz/business/finance/dane/dalnice.aspx>

24) *Wikipedie, otevřená encyklopedie.* [on-line].URL:<*http*://cs.wikipedia.org>

25) *Problém obchodního cestujícího - Wikipedie, otevřená encyklopedie*. [on-line].(6.6.2007)URL:<http://cs.wikipedia.org/wiki/Probl%C3%A9m_obchodn%C3%ADho_cestuj%C3%ADc%C3%Adho>

26) *Systémy pro podporu rozhodování - Wikipedie, otevřená encyklopedie.* [on-line]. (5.4.2009).URL: <http://cs.wikipedia.org/wiki/Syst%C3%A9my_pro_podporu_rozhodov%C3%A1n%C3%AD>

27) *Výpočet mýtného a informace*. [on-line].(29.12.2008). URL:<http://tollcz.ptv.de/premid/ti/index.jsf?locale=cs_CZ>

7 Přílohy

7.1 Seznam obrázků

7.2 Seznam tabulek

7.3 Kód modulů pro řešení okružní dopravní úlohy

Formulář Napoveda

```
Private Sub CommandButton_Storno_Click()

'Po stisknuti tlacitka Storno
Cencel = True
Unload Me
'Hodnota Cencel bude nastavena na True a formular se uzavre

End Sub
```

Formulář Vstupni_Formular

```
Private Sub CommandButton_OK_Click()

'Po stisknuti tlacitka OK
Storno = False

MNS = CheckBox_MNS
'Hodnota MNS nastavena na TRUE, pokud chceme pouzit tuto metodu
VAM = CheckBox_VAM
'Hodnota VAM nastavena na TRUE, pokud chceme pouzit tuto metodu

If Not (VAM Or MNS) Then
    MsgBox "Vyberte alespoň jednu metodu k počítání!"
    'Nebyla-li zvolena ani jedna metoda objevi se hlaska, abychom alespon jednu zvolili
    Exit Sub
End If

NazevModelu = TextBox_Nazev.Text
'Do promenne se zada vyplneny nazev modelu

'Vyber matice mest
On Error Resume Next
Set MaticeMest = Range(RefEdit_Mesta.Text)
If Err <> 0 Then
    MsgBox "Není vybrána platná oblast s názvy měst!"
    RefEdit_Mesta.SetFocus
    Exit Sub
    'V pripade spatneho vybrani oblasti je Focus vrácen do daneho prvku RefEdit a procedura je ukoncena
End If
On Error GoTo 0

'Vyber matice sazeb
On Error Resume Next
Set MaticeSazeb = Range(RefEdit_Sazby.Text)
If Err <> 0 Then
    MsgBox "Není vybrána platná oblast se sazbami vzdáleností!"
    RefEdit_Sazby.SetFocus
    Exit Sub
    'V pripade spatneho vybrani oblasti je Focus vrácen do daneho prvku RefEdit a procedura je ukoncena
End If
On Error GoTo 0

Me.Hide
'Skryti formulare
Kontrola
'Vyvolani procedury pro kontrolu zadanych vstupnich dat

End Sub
```

```
Private Sub CommandButton_Storno_Click()

'Po stisknuti tlacitka Storno
Storno = True
```

```
Unload Me
'Hodnota Storno bude nastavena na True a formular se uzavre

End Sub
```

```
Private Sub CommandButton_Napoveda_Click()

Napoveda.Show

End Sub
```

```
Sub Kontrola()

'Tato procedura je urcena k provedeni kontroly vstupnich hodnot
Kontrola_Rozsahu
'Zavolani procedury Kontrola_Rozsahu

'Volani jednotlivych procedur
If Rozsah Then Kontrola_Ctvercove_Matice
If Rozsah Then Kontrola_Mest_A_Sazeb
If Rozsah Then Kontrola_Nad_Hlavni_Diagonalou

If Not Rozsah Then Vstupni_Formular.Show
'Neni-li rozsah zadane matice mest v poradku objevi se znovu Vstupni formular pro nove zadani

End Sub
```

```
Sub Kontrola_Rozsahu()

'Protoze datovy typ Byte ma rozsah pouze 0 - 255, kontroluje se zda neni zadano vice bunek
If (MaticeMest.Rows.Count > 255 Xor MaticeMest.Columns.Count > 255) Then
    MsgBox "Bylo zadano více než 255 měst!To je prilis mnoho pro tuto aplikaci, prosim zadejte znovu."
    RefEdit_Mesta.SetFocus
    Rozsah = False
    'Pri nesplneni podminky rozsahu do 255 je hodnota Rozsah nastavena na False
Else: Rozsah = True
'Pri splneni podminky rozsahu do 255 je hodnota Rozsah nastavena na True
End If

End Sub
```

```
Sub Kontrola_Ctvercove_Matice()

'Kontroluje se, jestli je matice sazeb ctvercoveho rozmeru
If MaticeSazeb.Columns.Count <> MaticeSazeb.Rows.Count Then
    MsgBox "Počet řádků v matici sazeb neodpovídá počtu sloupců! Matice sazeb musí být čtvercového rozměru. "
    'Nerovna-li se pocet sloupcu poctu radku matice neni ctvercova
    RefEdit_Sazby.SetFocus
    Rozsah = False
Else: Rozsah = True
End If

End Sub
```

```
Sub Kontrola_Nad_Hlavni_Diagonalou()

'Kontrola zadanych hodnot nad hlavni diagonalou - hodnoty maji byt kladna cisla
For a = 1 To MaticeSazeb.Rows.Count - 1
For b = a + 1 To MaticeSazeb.Rows.Count
If MaticeSazeb(a, b) < 0 Then
    MsgBox "Matice sazeb obsahuje nad hlavní diagonálou alespoň jednu zápornou hodnotu!"
    RefEdit_Sazby.SetFocus
    ZadaniJeVPoradku = False
    Exit Sub
ElseIf MaticeSazeb(a, b) = 0 Then
    MsgBox "Matice sazeb obsahuje nad hlavní diagonálou alespoň jednu nulovou hodnotu!"
    RefEdit_Sazby.SetFocus
    ZadaniJeVPoradku = False
    Exit Sub
ElseIf Not IsNumeric(MaticeSazeb(a, b)) Then
    MsgBox "Matice sazeb obsahuje nad hlavní diagonálou alespoň jednu nečíselnou hodnotu!"
    RefEdit_Sazby.SetFocus
    ZadaniJeVPoradku = False
    Exit Sub
Else: ZadaniJeVPoradku = True
End If

Next b
'If Not ZadaniJeVPoradku Then Exit For
Next a

End Sub
```

```
Sub Kontrola_Mest_A_Sazeb()

'Kontrola jesli pocet mest odpovida rozsahu matice sazeb
If Not (MaticeMest.Rows.Count = MaticeSazeb.Rows.Count Or MaticeMest.Columns.Count = MaticeSazeb.Columns.Count) Then
    MsgBox "Počet měst neodpovídá rozměrům matice sazeb!"
    RefEdit_Mesta.SetFocus
    Rozsah = False
Else: Rozsah = True
End If

End Sub
```

Modul Metoda_MNS

```
Sub Uprava_Matice()

'Nahrazeni hodnot na hlavni diagonale a pod hlavni diagonalou
For y = 1 To Mesta
    For x = 1 To Mesta

        'Prepisani hodnoty pod diagonalou hodnotami nad diagonalou
        If x > y Then MaticeSazeb(x, y) = MaticeSazeb(y, x)
        'Prepsani hodnot na hlavni diagonale na znaky "X"
        If x = y Then MaticeSazeb(x, x) = "X"


   Next x
Next y
End Sub
```

```
Sub Metoda_Nejblizsiho_Souseda()

Uprava_Matice

Dim NazevListu As String
Dim CelkovaVzdalenostStep As Double
'Celková kumulativni vzdalenost podle prave provadeneho kroku (step)
Dim Step As Byte
'Urcuje jednotlive kroky celeho cyklu
Dim MinVzdalenostX, MinVzdalenostY As Byte
'Cislo sloupce nebo radku, v nemz je MinVzdálenost
'MinVzdalenost je nejmensi hodnota v prave prohledavanem radku
Dim i, j As Byte
'Vyuzijeme pro cyklus For
Dim x, y As Byte
'Prave Prohledavane cislo sloupce nebo radku

NazevListu_MNS = "MNS"

For Step = 1 To Mesta

    MaticeMNS = MaticeSazeb
    'Priradi matici sazeb do nove promenne,nebude tak dochazet k zamene zadanych udaju pri behu procedury

    Dim UsekVzdalenostStep() As Double
    'Definuje hodnoty jako typ double
    'UsekVzdalenostStep je matici sazeb jednotlivych vzdalenosti useku mezi mesty dle provadeneho kroku step

    ReDim UsekMestStep(1 To 1)
    'UsekMestStep je matice s cisly mest useku trasy postupne prirazovanych do celkove trasy podle provadeneho koroku step
    'Zacina se s polem o velikosti 1
    UsekMestStep(1) = Step
    'Pocatecni mesto je prirazeno dle prave probihajiciho kroku step
    MinVzdalenostX = Step
    'Zacina se ve meste podle Step
    CelkovaVzdalenostStep = 0
    'Celkova vzdalenost na zacatku je znama, je to 0 km

    For i = 1 To Mesta
        MinVzdalenost = MaticeMNS(1, 1)
        'Hleda se nejmensi hodnotu v radku podle Step
        y = MinVzdalenostX

        For x = 1 To Mesta
            Vzdalenost = MaticeMNS(y, x)

            If MinVzdalenost > Vzdalenost Then
                MinVzdalenost = Vzdalenost
                MinVzdalenostX = x
                MinVzdalenostY = y
            End If

        Next x

        If i < (Mesta - 1) Then MaticeMNS(MinVzdalenostX, Step) = "X"
        'Krizem se vyskrtne prirazena hodnotu, krome posledniho kroku, kdy se s vypoctem konci a hodnota tak nemusi byt skrtana

        For j = 1 To Mesta
            MaticeMNS(j, MinVzdalenostX) = "X"
        Next j
        'Vyskrtne se sloupec, ve kterem se nachazi prirazena hodnota


        ReDim Preserve UsekMestStep(1 To i + 1)
        UsekMestStep(i + 1) = MinVzdalenostX
        ReDim Preserve UsekVzdalenostStep(1 To i)
        UsekVzdalenostStep(i) = MinVzdalenost
        CelkovaVzdalenostStep = CelkovaVzdalenostStep + MinVzdalenost
    Next i

    If Step = 1 Then
        CelkovaVzdalenost_MNS = CelkovaVzdalenostStep
        Trasa_MNS = UsekMestStep
        Vzdalenost_MNS = UsekVzdalenostStep
    Else
        If CelkovaVzdalenost_MNS > CelkovaVzdalenostStep Then
            CelkovaVzdalenost_MNS = CelkovaVzdalenostStep
            Trasa_MNS = UsekMestStep
            Vzdalenost_MNS = UsekVzdalenostStep
        End If
    End If

Next Step

Vypis (NazevListu_MNS)

Range("B4") = "Metoda nejbližšího souseda"
Range("E6") = CelkovaVzdalenost_MNS
Range("A8").Select
```

```
'Vysledny vypis hodnot
For i = 1 To Mesta
    ActiveCell.Offset(i - 1, 1) = MaticeMest(Trasa_MNS(i))
    ActiveCell.Offset(i - 1, 2) = "--->"
    ActiveCell.Offset(i - 1, 3) = MaticeMest(Trasa_MNS(i + 1))
    ActiveCell.Offset(i - 1, 4) = Vzdalenost_MNS(i)
Next i


Range("A1").Select

End Sub
```

Modul Metoda_VAM

```
Sub Uprava_Matice()

'Nahrazeni hodnot na hlavni diagonale a pod hlavni diagonalou
For y = 1 To Mesta
    For x = 1 To Mesta

        'Prepisani hodnoty pod diagonalou hodnotami nad diagonalou
        If x > y Then MaticeSazeb(x, y) = MaticeSazeb(y, x)
        'Prepsani hodnot na hlavni diagonale na znaky "X"
        If x = y Then MaticeSazeb(x, x) = "X"


   Next x
Next y
End Sub
```

```
Sub Vogelova_Metoda()

Uprava_Matice

ReDim Rozdil(1 To Mesta, 1 To 2)
'Dvourozmerne pole - poradi v rade a druh rady (1-radek, 2-sloupec)
ReDim VAM_od(1 To Mesta)
'Pole s pocatecnimi mesty v useku
ReDim VAM_do(1 To Mesta)
'Pole s koncovymi mesty v useku
ReDim PTVAM(1 To Mesta)
'Pomocné pole s castmy cykl - stejne cislo oznacuje souvisejici castecny cyklus
ReDim Vzdalenost_VAM(1 To Mesta)
'Pole vzdalenosti mezi mesty
Dim Step As Byte
Dim x, y As Byte
Dim p As Byte
'Pomocna promenna urcujici pozici v poli Rozdil
ReDim NR(1 To 2 * Mesta)
'Nejvetsi rozdíl ze všech hodnot pole Rozdil
ReDim NR_Pozice(1 To 2 * Mesta)
ReDim NR_RS(1 To 2 * Mesta)
'Urcuje, zda byl nejvetsi rozdil nalezen v radku ci sloupci
Dim MinSazbaR, MinSazbaS As Byte
'Radek ci sloupec s nejmensi hodnotou
Dim i, j As Byte
Dim pm As Integer

NazevListu_VAM = "VAM"

MaticeVAM = MaticeSazeb

CelkovaVzdalenost_VAM = 0

For Step = 1 To Mesta - 2
    'Cyklus se provede pouze (n-2) krat, protoze nakonec v tabulce zbydou dve hodnoty
    'Ostatni hodnoty budou proskrtane (prirazeno "X")
    'Posledni dve hodnoty tedy urci poésledni dva mesta
```

```
'Spocteni radkovych minim
For y = 1 To Mesta
    NejmensiRadek1 = MaticeVAM(1, 1)
    NejmensiRadek2 = MaticeVAM(1, 1)

    For x = 1 To Mesta
        If MaticeVAM(y, x) < NejmensiRadek2 Then NejmensiRadek2 = MaticeVAM(y, x)
        If NejmensiRadek2 < NejmensiRadek1 Then
            Temp = NejmensiRadek1
            'Temp promenna pro prohazovani dvou nejmensich hodnot v radku ci sloupci
            NejmensiRadek1 = NejmensiRadek2
            NejmensiRadek2 = Temp
        End If
    Next x

    If IsNumeric(NejmensiRadek1) Then
        Rozdil(y, 1) = NejmensiRadek2 - NejmensiRadek1
    Else: Rozdil(y, 1) = "X"
    End If
    'Rozdil - dvourozmerne pole hodnot z radku ci sloupce(prni cislo je poradi, druhe zda se jedna o radek - 1 či sloupec - 2)

Next y

'Spocteni sloupcovych minim
For x = 1 To Mesta
    NejmensiSloupec1 = MaticeVAM(1, 1)
    NejmensiSloupec2 = MaticeVAM(1, 1)

    For y = 1 To Mesta
        If MaticeVAM(y, x) < NejmensiSloupec2 Then NejmensiSloupec2 = MaticeVAM(y, x)
        If NejmensiSloupec2 < NejmensiSloupec1 Then
            Temp = NejmensiSloupec1
            NejmensiSloupec1 = NejmensiSloupec2
            NejmensiSloupec2 = Temp
        End If
    Next y

    If IsNumeric(NejmensiSloupec1) Then
        Rozdil(x, 2) = NejmensiSloupec2 - NejmensiSloupec1
    Else: Rozdil(x, 2) = "X"
    End If

Next x

'Urceni nejvetsiho rozdilu
NR(1) = -1
'Vsechny rozdily jsou nezaporne
p = 0
For Each m In Rozdil
    p = p + 1
    If m >= NR(1) And IsNumeric(m) Then

        If m > NR(1) Then
            k = 1
            ReDim NR(1 To k)
            'Pole se nahradi opet jednickovym
        Else
            k = k + 1
            ReDim Preserve NR(1 To k)
            'Pole se o jednicku zvetsi
        End If

        NR(k) = m
        If p <= Mesta Then
            'Z radkovych rozdilu
            NR_RS(k) = 1
            NR_Pozice(k) = p
        Else
            NR_RS(k) = 2
            'Ze sloupcovych rozdilu
            NR_Pozice(k) = p - Mesta
            'Je treba snizit NR_Pozice
        End If
    End If
Next m

'Nalezeni minima v radku ci sloupci s nejvetsim rozdilem
MinSazba = MaticeVAM(1, 1)

For j = 1 To UBound(NR)
    For i = 1 To Mesta
        If NR_RS(j) = 1 Then 'Rozdil nalezen v radku

            If MaticeVAM(NR_Pozice(j), i) < MinSazba Then
                MinSazba = MaticeVAM(NR_Pozice(j), i)
                MinSazbaR = NR_Pozice(j)
                MinSazbaS = i
            End If

        Else 'Rozdil nalezen ve sloupci

            If MaticeVAM(i, NR_Pozice(j)) < MinSazba Then
                MinSazba = MaticeVAM(i, NR_Pozice(j))
                MinSazbaR = i
                MinSazbaS = NR_Pozice(j)
            End If

        End If
    Next i
Next j

VAM_od(Step) = MinSazbaR
VAM_do(Step) = MinSazbaS
Vzdalenost_VAM(Step) = MaticeSazeb(VAM_od(Step), VAM_do(Step))
```

```
    If Step > 1 Then 'Neprovade se v prvnim kroku
        PTVAM(Step) = Step
        'Pomocne pole PTVAM urcuje jiz sestavene souvisle useky, skupinu souvislych useku urcuji podle stejne hodnoty v tomto poli
        For i = 1 To Step - 1
            If VAM_od(i) = MinSazbaS Then PTVAM(Step) = PTVAM(i)
            If VAM_do(i) = MinSazbaR Then PTVAM(Step) = PTVAM(i)
        Next i

        Else: PTVAM(1) = 1
        'Nastane u prvniho kroku
    End If

    For i = 1 To Step - 1
    'Zkontrolovani prvku poli VAM_od a VAM_do s poslednim prvkem, pri navatnosti prvku se nahradi pomocna promenna
        If VAM_od(i) = VAM_do(Step) Then
            For j = 1 To Step
                If PTVAM(j) = PTVAM(Step) Then PTVAM(j) = PTVAM(i)
            Next j
        End If

        If VAM_do(i) = VAM_od(Step) Then
            For j = 1 To Step
                If PTVAM(j) = PTVAM(Step) Then PTVAM(j) = PTVAM(i)
            Next j
        End If

    Next i

    CelkovaVzdalenost_VAM = CelkovaVzdalenost_VAM + Vzdalenost_VAM(Step)

    'Vyskrtneme sloupec, ve kterem se nachazi prirazena hodnota
    For x = 1 To Mesta
        MaticeVAM(x, MinSazbaS) = "X"
    Next x

    'Vyskrtneme radek, ve kterem se nachazi prirazena hodnota
    For y = 1 To Mesta
        MaticeVAM(MinSazbaR, y) = "X"
    Next y

    'Vyskrtnuti krizem prirazene hodnoty (krome posledniho kroku)
    'MaticeVAM(MinSazbaS, MinSazbaR) = "X"
    'Vyskrtnuti hodnot, ktere predcasne uzavrou cyklus
    For x = 1 To Step
        For y = 1 To Step
            If PTVAM(x) = PTVAM(y) Then MaticeVAM(VAM_do(x), VAM_od(y)) = "X"
        Next y
    Next x

Next Step

pm = 1
'Pomocna promenna pro pridani poslednich dvou useku

For y = 1 To Mesta
    For x = 1 To Mesta
        If IsNumeric(MaticeVAM(x, y)) Then
            VAM_od(Mesta - pm) = x
            VAM_do(Mesta - pm) = y
            Vzdalenost_VAM(Mesta - pm) = MaticeVAM(x, y)
            CelkovaVzdalenost_VAM = CelkovaVzdalenost_VAM + Vzdalenost_VAM(Mesta - pm)
            pm = pm - 1
            'Jako dalsi je zarazen posledni usek
        End If
    Next x
Next y

Vypis (NazevListu_VAM)

Range("B4") = "Vogelova metoda"
Range("E6") = CelkovaVzdalenost_VAM
Range("A8").Select

'Vysledny vypis hodnot
For i = 1 To Mesta
    ActiveCell.Offset(i - 1, 1) = MaticeMest(VAM_od(i))
    ActiveCell.Offset(i - 1, 2) = "--->"
    ActiveCell.Offset(i - 1, 3) = MaticeMest(VAM_do(i))
    ActiveCell.Offset(i - 1, 4) = Vzdalenost_VAM(i)
Next i

Range("B8").CurrentRegion.Select
Range("A1").Select

End Sub
```

Modul TSP_Kod

```
Public MaticeMest
Public MaticeSazeb
Public NazevModelu As String
Public MNS, VAM As Boolean
Public Rozsah As Boolean
Public Storno As Boolean
Public Mesta As Byte
Public Trasa_MNS, TrasaOd_VAM, TrasaDo_VAM
Public CelkovaVzdalenost_MNS, CelkovaVzdalenost_VAM As Double
```

```
Sub Tsp()

'Spusteni Vstupniho formulare pro zadani hodnot
Vstupni_Formular.Show
If Not Storno Then

    'Vypina probihajici aktualizace obrazovky, urychli se tim vypocet
    Application.ScreenUpdating = False

    'Nacteni poctu mest s ohledem na zadani matice ve formatu 1xM nebo Mx1
    If MaticeMest.Rows.Count > MaticeMest.Columns.Count Then
    Mesta = MaticeMest.Rows.Count
    Else: Mesta = MaticeMest.Columns.Count
    End If

    'Spousteni samotnych metod vypoctu TSP
    If MNS Then Metoda_Nejblizsiho_Souseda
    If VAM Then Vogelova_Metoda

End If
End Sub
```

```
Sub PridejDoMenu()

Dim MenuNastroje As CommandBarPopup
Dim NovaPolozka As CommandBarButton


VymazZMenu
'Vymaze polozku, pokud jiz existuje

Set MenuNastroje = CommandBars(1).FindControl(ID:=30007)
'Najde v menu Nastroje

If MenuNastroje Is Nothing Then
MsgBox "Položku nelze přidat do nabídky Nástroje"
Exit Sub
Else
Set NovaPolozka = MenuNastroje.Controls.Add(Type:=msoControlButton)
With NovaPolozka
.Caption = "Výpočet TSP"
.OnAction = "TSP_Kod.Tsp"
End With
End If

End Sub
```

```
Sub VymazZMenu()

On Error Resume Next
CommandBars(1).FindControl(ID:=30007).Controls("Výpočet TSP").Delete

End Sub
```

Modul Vstupni_Modul

```
Sub Vypis(NazevListu As String)

NovyList (NazevListu)

'Formatovani jednotlivyych bunek ci poli bunek
Range("B1").ColumnWidth = 15
Range("C1").ColumnWidth = 5
Range("D1").ColumnWidth = 15
Range("E1").ColumnWidth = 5
Range("A1").Select

With Range("B2:E2")
    .MergeCells = True
    .HorizontalAlignment = xlCenter
    .Font.Name = "Times New Roman"
    .Font.Size = 14
    .Font.Bold = True
    .Font.Underline = True
End With

ActiveCell.Cells(2, 2) = NazevModelu

With Range("B4:E4")
    .MergeCells = True
    .HorizontalAlignment = xlCenter
    .Font.Name = "Times New Roman"
    .Font.Size = 12
    .Font.Bold = True
End With

Range("B5:D5").MergeCells = True
Range("B5") = "Celkem měst:"
Range("E5") = Mesta
Range("B6:D6").MergeCells = True
Range("B6") = "Celková vzdálenost okruhu v km:"
Range("E6").Font.Bold = True
Range("B7:D7").MergeCells = True
Range("B7") = "Jednotlivé úseky trasy:"
Range("B7").Font.Underline = True
Range("E7") = "km"

End Sub
```

```
Sub NovyList(NazevListu As String)

Dim PocetListu As Byte
'Celkovy pocet listu prave aktivniho sesitu
Dim NoveCislo As Byte
'Nove cislo k pojmenovani dalsiho listu v sesitu
Dim i As Byte

PocetListu = ActiveWorkbook.Sheets.Count
'Pocet listu v sesitu

For i = 1 To PocetListu
    If ActiveWorkbook.Sheets(i).Name = NazevListu + " (" + CStr(NoveCislo) + ")" Then NoveCislo = NoveCislo + 1
Next i
'Nazev listu je bran z prislusneho modulu pro vypocet pomoci zvolene metody

ActiveWorkbook.Sheets.Add after:=ActiveWorkbook.Sheets(PocetListu)
'Novy list se vlozi nakonec rady listu

On Error Resume Next
'Existuje-li uz takovy list, nedojde ke prejmenovani listu

ActiveSheet.Name = NazevListu + " (" + CStr(NoveCislo) + ")"
'Nazev noveho listu

End Sub
```