Vysoká škola ekonomická v Praze
Fakulta informatiky a statistiky

Teorie a praxe vícestupňové úlohy obchodního cestujícího v ČR

BAKALÁŘSKÁ PRÁCE

Studijní program: Kvantitativní metody v ekonomice
Studijní obor: Matematické metody v ekonomii

Autor: Vojtěch Vávra
Vedoucí práce: prof. Ing. Josef Jablonský, CSc.

Praha, květen 2020

Prohlášení

Prohlašuji, že jsem bakalářskou práci *Teorie a praxe vícestupňové úlohy obchodního cestujícího v ČR* vypracoval samostatně za použití v práci uvedených pramenů a literatury.

V Praze dne 10. května 2020 .....

Podpis studenta

Poděkování

Rád bych poděkoval prof. Ing. Josefovi Jablonskému, CSc. za všechny rady a podněty, které mi pomohly při vzniku práce. Dále bych rád poděkoval rodičům za trpělivost a podporu během studia a svému bratrovi Mgr. Janu Vávrovi za pomoc s programovacím jazykem R.

Abstrakt

Název práce: Teorie a praxe vícestupňové úlohy obchodního cestujícího v ČR
Autor: Vojtěch Vávra
Katedra: Katedra ekonometrie
Vedoucí práce: prof. Ing. Josef Jablonský, CSc.

Tato práce se zabývá především vícestupňovým problémem obchodního cestujícího. Obsahuje krátce sepsanou historii. V teoretické části jsou formulovány ekonomické i matematické modely těchto problémů s rozšířením o minimální a maximální meze pro počet měst, které má každý obchodní cestující navštívit. V části praktické jsou tyto modely implementovány na vybraných místech České republiky. Řešení těchto problémů jsou získána pomocí programu MPL for Windows a řešitele GUROBI. V závěru práce jsou všechny použité modely porovnány a vyhodnoceny nejen číselně, ale i na mapě.

Klíčová slova

vícestupňový problém obchodního cestujícího, smíšené celočíselné programování, metoda větví a mezí, MPL for Windows, GUROBI

Abstract

Title: Theory and practice of the multiple travelling salesman problem in Czech Republic
Author: Vojtěch Vávra
Department: Department of Econometrics
Supervisor: prof. Ing. Josef Jablonský, CSc.

This thesis deals primarily with multiple travelling salesman problem. It contains a brief history. In the theoretical part we deliver formulations of economic and mathematic models for these problems with extension of minimal and maximal bounds for number of cities which are imposed for every salesman. In the practical part, these models are implemented on selected places of the Czech Republic. Solutions are obtained by the program MPL for Windows and the solver GUROBI. In conclusion of the thesis all used models are compared not only numerically but also graphically.

Keywords

multiple travelling salesman problem, mixed integer programming, Branch and bound method, MPL for Windows, GUROBI

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

ACO Ant Colony Optimization (optimalizace mravenčí kolonií)
B&B Branch and Bound (metoda větví a mezí)
BB Best Bound (nejlepší vazba)
BCO Bee Colony Optimization (optimalizace včelí kolonií)
CP Cutting Planes (metoda sečných nadrovin)
GA Genetic algorithm (genetický algoritmus)
GAP maximální chyba v procentech pro hodnotu účelové funkce od nejlepšího možného optimálního řešení
GPS Global Positioning System (globální poziční systém)
IPM Interior-Point Method
(bariérová metoda, metoda vnitřní čárkou, metoda vnitřním bodem)
KaBe Kara-Bektas
LK, LKH Lin-Kernighan heuristika (metoda výměn)
LP Linear Programming (lineární programování)
MIP Mixed Integer Programming (smíšené celočíselné programování)
MIQCP Mixed-Integer SOCP (smíšené celočíselné programování kuželu druhého řádu)
MIQP Mixed Integer Quadratic Programming
(smíšené celočíselné kvadratické programování)
MPL Mathematical Programming Language (matematický programovací jazyk)
MTSP Multiple Travelling Salesman Problem
(vícestupňový problém obchodního cestujícího)
MTZ podmínka Miller-Tucker-Zemlin
n počet měst v souboru, počet měst na grafu G
NP nedeterministicky polynomiální
P polynomiální
QP Quadratic Programing (kvadratické programování)
SEC Subtour Elimination Constraints (eliminační omezení podcyklů)
SOCP Second Order Cone Programing (programování kuželu druhého řádu)
TSP Travelling Salesman Problem
(problém obchodního cestujícího, okružní dopravní problém)
VRP Vehicle Routing Problem (rozvozní problém)
Z celková ujetá vzdálenost všech obchodních cestujících

Úvod

V létě roku 2019 se na prahu naší ostroměřské sokolovny (tělocvičny) objevil Rome Milan, člen předsednictva Amerického Sokola a člen výboru Amerického sokolského muzea a knihovny. Jezdil totiž po celé České republice, objížděl jednotlivé sokolovny a hledal sokolské artefakty, např. historické prapory, sokolské kroje, cvičební nářadí, budovy sokoloven a mnohé další. Ty následně fotografoval a pokud byla některá cvičební pomůcka již vyřazena, mohl se s majiteli domluvit, zda by ji mohli věnovat muzeu.

Otázka tedy zní, jak by mohl své výpravy po celé České republice naplánovat co nejefektivněji. Vezměme v potaz, že všechny sokolovny za jednu cestu nezvládne objet, protože jednot dnes existujících a z dob minulých je přes 1000.

Jelikož je problém příliš složitý, rozdělíme ho na dílčí problémy po župách (správa několika sokolských jednot dle oblastí), tedy zhruba po 8–68 městech. Takto se již stává úloha řešitelnou v rozumném čase.

V rámci této bakalářské práce se především podíváme na rozšíření problému obchodního cestujícího (dále TSP z anglického Travelling Salesman Problem) na vícestupňový problém obchodního cestujícího (dále MTSP z anglického Multiple Travelling Salesman Problem). Rozdíl mezi těmito úlohami je, že pro MTSP můžeme použít více než jednoho obchodního cestujícího, kdežto u TSP můžeme použít jednoho jediného obchodního cestujícího.

Klíčové bude nalézt vhodný matematický model, který bude vzhledem ke složitosti MTSP hledat časově nejlepší řešení. Porovnáme i jednotlivé modely pro verzi TSP. Pro vyřešení modelů použijeme programu MPL for Windows a řešitele GUROBI, do kterého je implementována řada metaheuristik.

Celkem budeme brát v úvahu pět matematických modelů, z čehož jsou čtyři určeny pro řešení MTSP a všech pět lze použít k řešení TSP. Jednou z variant bude i převedení úlohy rozvozního problému (dále VRP z anglického Vehicle Routing Problem) na úlohu MTSP. Druhou variantou, kterou budeme brát v úvahu, je rozšíření modelu TSP na MTSP. Nakonec na tyto dvě varianty ještě uvalíme dolní a horní meze pro všechny obchodní cestující. Tyto meze budou omezovat minimální a maximální množství měst, které mohou jednotliví obchodní cestující navštívit.

Práce je rozdělena do čtyř částí. V kapitole 1 se seznámíme s historii řešení TSP a jak vlastně tato úloha vznikla. V kapitole 2 se podíváme na ekonomický i matematický model TSP a MTSP. Navrhneme také metody řešení a nabídneme řadu dalších přístupů. V kapitole 3 zrekapitulujeme zisk dat a vzdáleností mezi jednotlivými sokolskými jednotami. Dále vysvětlíme programy, které použijeme k řešení problémů a zpracování dat. V poslední kapitole 4 provedeme řadu výpočtů, které okomentujeme, vykreslíme na mapu a pokusíme se vybrat nejlepší z navrhovaných modelů.

1. Historie TSP

Nejprve se podívejme, jak TSP vznikl. Neopomeneme ani pestrou historii a vývoj řešení TSP. Porovnáme několik přístupů a zmíníme špičky v tomto oboru, od kterých se můžeme inspirovat a případně implementovat jejich řešení TSP.

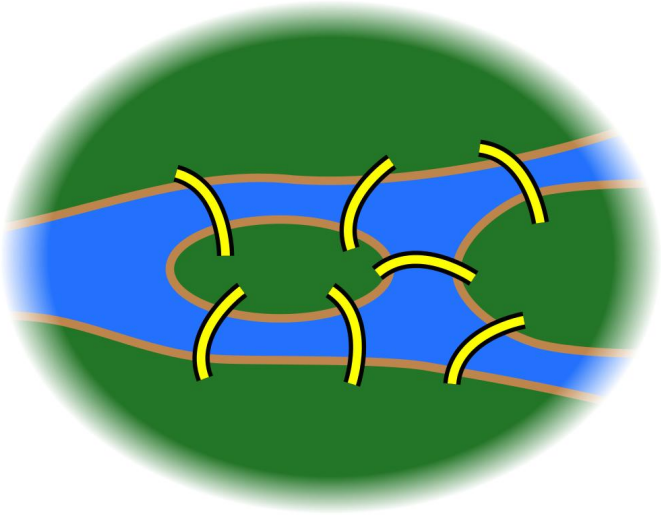
1.1 První krůčky k TSP

První a pravděpodobně nejstarší úlohou, která předchází úloze TSP podle (Applegate et al., 2006, kap. 1.2), je jezdcova procházka. Pro tuto úlohu máme k dispozici klasickou šachovnici rozměru 8×8 a jednoho jezdce. Hráč musí vstoupit jezdcem na všechna políčka na šachovnici tak, aby na žádné políčko nevstoupil dvakrát. V původní verzi se jezdec nemusí vracet na startovní políčko, ale lze ji zadat i tak, aby se jezdec musel vracet na startovní políčko, což odpovídá TSP.

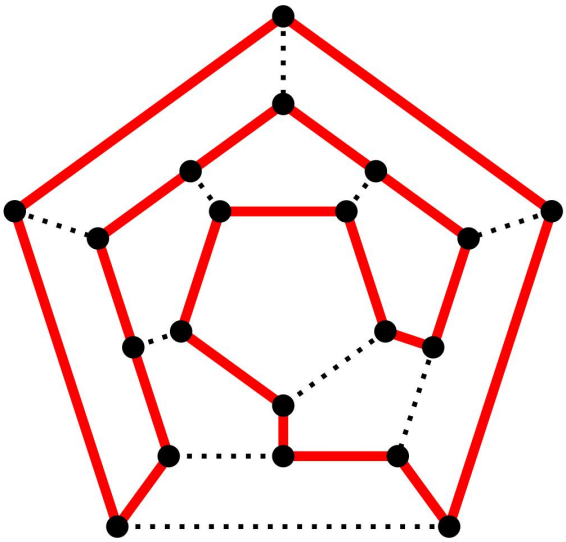
Jezdec nemůže skákat na libovolná políčka, pohybuje se jen do tvaru písmene „L“. Pohyb jezdce můžeme přesněji definovat jako posun jezdce o dvě pole vertikálním nebo horizontálním směrem a následně ještě posun o jedno pole kolmo libovolným směrem na zvolený směr v prvním kroku. Tuto úlohu lze také rozšířit na šachovnice jiných rozměrů než 8×8, tedy $i \times j$, kde i a j jsou kladné a celočíselné. Ovšem pro některé typy nemusí existovat řešení. Například na šachovnici o rozměru 3×3 nemůžeme z prostředního políčka na žádné jiné políčko vstoupit.

S matematickým řešením jezdcovy procházky přichází v roce 1759 Euler. Vydal studii, kde podrobně rozebírá několik řešení této úlohy a dodává matematický pohled na danou úlohu. Také je považován za zakladatele teorie grafů a podařilo se mu vyřešit problém sedmi mostů města Královce. V tomto problému řešíme přejití všech žlutých mostů dle obrázku 1.1, aniž bychom jediný z nich použili dvakrát. Hledáme tedy Eulerovu kružnici, viz oddíl 2. Euler přišel na to, že tato úloha nemá žádné řešení a taková cesta neexistuje, jak předkládá (Alexanderson, 2006).

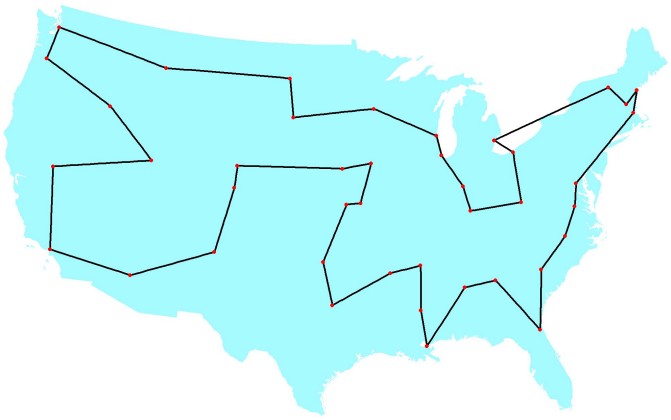
Hamilton přichází v roce 1856 s hrou, jejíž název zní „The Icosian Game“, jak udává (Dalgety, 2020). V této hře hledáme cestu mezi všemi dvaceti vrcholy dodekaedru (dvanáctistěnu) po jeho třiceti hranách tak, aby se žádná hrana ani vrchol neopakovaly. Hledáme tedy Hamiltonovu kružnici, viz oddíl 2. Tento problém lze překreslit do 2D podoby, viz obrázek 1.2. Červenou barvou je označeno řešení.



Obrázek 1.1: Problém sedmi mostů města Královce, autor (Martin, 2020).



Obrázek 1.2: Hra „The Icosian Game“, autor (Sommer, 2020).



Obrázek 1.3: Optimální cesta skrz 42 měst USA, autor (Cook, 2020).

1.2 První přístupy k TSP až po současnost

Poprvé byl v matematické literatuře zmíněn TSP Mengerem v roce 1930. Inspiroval se problémy pošťáků, kteří se s tímto problémem denně potkávali. Definoval „Botenproblem“ jako hledání nejkratší cesty mezi konečným počtem bodů a danými délkami mezi nimi tak, aby každý bod prošel cestovatel pouze jednou, jak předkládá (Applegate et al., 2006, kap. 1.2). Problém však nezahrnuje zpáteční cestu na začátek. Dnes by se spíše tento problém definoval jako „Messenger problem“, případně pošťákův problém. Menger se zajímal především o komplexnost tohoto problému.

V následujících letech přicházejí matematici s podobnými problémy, např. problém čínského listonoše, problém cestujícího farmáře, problém školního autobusu, problém vozu čistírny a byl definován i TSP v geometrické podobě.

Důležitým milníkem pro řešení TSP je studie (Dantzig et al., 1954). V této studií byl řešen TSP pro zadání o počtu 49 hlavních měst jednotlivých států USA. Avšak 7 měst bylo vyřazeno, protože mezi Washingtonem, D.C. a Bostonem vedla nejkratší cesta skrz těchto 7 měst (např. New York a Filadelfie), jak uvádí (Cook, 2020). Na optimální cestu se podívejme na obrázku 1.3. Zmiňovaná studie pokládá základní kámen pro řešení TSP.

Po objevu klíčového přístupu (Dantzig et al., 1954) se začaly řešit problémy s větším množstvím měst. Pokroky v objemu dat si prohlédněme v tabulce 1.1 podle předlohy (Applegate et al., 2006, kap. 1.7). Pokud úloha nebyla pojmenována, pak byly použity náhodně vygenerované body.

S modernizací hardwaru postupně přicházela i modernizace softwaru, což vybízelo k novým přístupům řešení TSP. V roce 1990 vznikla knihovna TSPLIB s 24 příklady a stala se novou velkou výzvou. Když ani to nestačilo, protože dva příklady vyřešily již napsané kódy, byla knihovna rozšířena v roce 1995 na 34 příkladů. Dnes knihovna čítá daleko více problémů, viz on-line databáze univerzity (Reinhelt, 2014).

Tabulka 1.1: Přehled důležitých milníků výpočtů TSP a počtu měst.

Rok	Autoři	Počet měst	Jméno úlohy
1954	G. Dantzig, R. Fulkerson, S. Johnson	49	dantzig42
1971	M. Held, R. M. Karp	57	[304]
1971	M. Held, R. M. Karp	64	náhodné body
1975	P. M. Camerini, L. Fratta, F.Maffioli	67	náhodné body
1975	P. Miliotis	80	náhodné body
1977	M. Grötschel	120	gr120
1980	H. Crowder, M. W. Padberg	318	lin318
1987	M. Padberg, G. Rinaldi	532	att532
1987	N. Grötschel, O. Holland	666	gr666
1987	M. Padberg, G. Rinaldi	1 002	pr1002
1987	M. Padberg, G. Rinaldi	2 392	pr2392

Začal tedy závod ve zbrojení kódů. Pro příklad si uvedeme jeden z kódů jménem „Concorde“ (Combinatorial Optimization and Networked Combinatorial Optimization Research and Development Environment), jehož autory jsou Applegate, Bixby, Chvátal a Cook. Autoři se inspirovali prací (Dantzig et al., 1954). Kód je naprogramován v jazyce C, přesahuje přes 130 000 řádků a používá výpočetní schéma pro pokračování procesu řešení TSP, když řezné roviny již dále nedělají významný pokrok. Tento skript dokázal vyřešit zbylých 32 příkladů. Významné milníky tohoto kódu, viz tabulka 1.2. Nejobsáhlejší úloha původní verze knihovny TSPLIB obsahovala na 85 900 měst, jak píší (Applegate et al., 2006).

Tabulka 1.2: Pokroky kódu Concorde.

Rok	kód	Počet měst	Jméno úlohy
1992	Concorde	3 038	pcb3038
1993	Concorde	4 461	fnl4461
1994	Concorde	7 397	pla7397
1998	Concorde	13 509	usa13509
2001	Concorde	15 112	d15112
2004	Concorde	24 978	sw24978
2004	Concorde s domino-paritou	33 810	pla33810
2006	Concorde s domino-paritou	85 900	pla85900

O rekord pro jeden z největších problémů s názvem „World TSP“ se přetahoval již zmíněný kód Concord a práce Helsgauna. Tento problém obsahuje 1 904 711 měst a byl stvořen v roce 2001. Helsgaun vylepšil své nejlepší řešení na délku 7 515 772 107 m v březnu roku 2018. Pro svůj kód využil LKH (Lin-Kernighan heuristic) algoritmu. Kód Concord vypočítal své nejlepší řešení této úlohy o délce 7 512 218 268 m v červnu roku 2007, jak udává (Cook, 2020). Toho bylo dosaženo rok po vydání knihy (Applegate et al., 2006), která v dnešní době slouží jako odrazový můstek pro výzkum v oblasti TSP.

2. Formulace (M)TSP

V této části práce se podíváme na teoretickou stránku problému ohledně dat, modelů a výpočtů spojených s hledáním optimálního řešení jednotlivých typů úloh.

Zavedeme si nejprve pojmy z teorie grafů, která se zabývá strukturami bodů a hran mezi nimi. Takovou strukturu nazýváme grafem. Mějme ohodnocený graf $G = (V,E)$, kde V značí množinu všech vrcholů (měst) a E označuje množinu všech hran (spojů) mezi vrcholy, tedy hrana je určena vždy dvojicí vrcholů, jak uvádí (Pelikán, 2001, kap. 3.8.4).

Úplný neorientovaný graf definujeme jako graf G, pro který platí, že mezi každou dvojicí vrcholů V existuje právě jedna neorientovaná hrana s daným ohodnocením, kde neorientovanou hranou rozumíme obousměrnou spojnici.

Cestou rozumíme posloupnost vrcholů a hran, která spojuje dva vybrané vrcholy. Cestu končící ve stejném vrcholu, v jakém začíná, nazýváme cyklem.

Hamiltonovská kružnice je cyklus v grafu G, který spojuje všechny vrcholy V a každý vrchol je navštíven právě jednou.

Eulerovu kružnici definujeme jako cyklus přes všechny hrany E grafu G, aniž bychom jedinou hranu využili víckrát než jednou.

2.1 Formulace TSP

TSP zná téměř každý, kdo se alespoň trochu zajímá o operační výzkum. Někteří mu také říkají okružní dopravní problém. Jedná se o jednoduchý problém, který však není snadné vyřešit. Úkolem obchodního cestujícího je, aby prošel všemi městy (vrcholy) tak, aby urazil co nejkratší vzdálenost, každé město navštívil jednou a vrátil se zpět do své výchozí pozice. Tuto úlohu tedy definujeme jako nalezení nejkratší hamiltonovské kružnice v úplném neorientovaném grafu G.

TSP lze definovat jako symetrický a nebo také jako asymetrický. Pokud zvolíme variantu asymetrickou, pak vzdálenost mezi dvěma městy může být odlišná, když se změní směr cesty mezi těmito městy. V této práci však budeme uvažovat variantu symetrickou, abychom usnadnili softwaru výpočet, protože vzdálenosti mezi takovými dvěma městy jsou obousměrně stejné, tedy $c_{ij} = c_{ji}$, kde c_{ij} značí vzdálenost mezi městy i a j.

Existuje několik modifikací této úlohy, např. TSP s časovými okny nebo vícestupňový TSP, kterým se dále zabývá tato práce. Také lze TSP upravit na problém čínského listonoše, kde místo nejkratší cesty po Hamiltonově kružnici hledáme nejkratší cestu po Eulerově kružnici.

2.1.1 Matematický model TSP

TSP lze definovat jako úlohu smíšeného celočíselného programování (MIP). Pro model TSP si nejprve definujme následující symboliku pro

- ▷ n - počet měst na grafu G,
- ▷ $i, j = 1, \ldots, n$ - indexy měst,
- ▷ c_{ij} - kladná vzdálenost mezi městy i a j,
- ▷ $x_{ij} \in \{0,1\}$ - indikátor použití přímé cesty z města i do města j,
- ▷ Z - celková ujetá vzdálenost obchodního cestujícího,
- ▷ u_i - pořadové číslo navštívení města i.

Základní model TSP pro celočíselné lineární programování podle (Miller et al., 1960) lze zapsat následovně

$$Z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \longrightarrow MIN, \quad i \neq j, \tag{2.1}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i, \tag{2.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j, \tag{2.3}$$

$$u_i - u_j + nx_{ij} \leq n-1, \quad i \neq j, i = 1, \ldots, n,\ j = 2, \ldots, n, \tag{2.4}$$

$$x_{ij} \in \{0;1\}, \quad \forall i, \forall j, \tag{2.5}$$

$$u_i \in \mathbb{Z}^+, \quad \forall i, \tag{2.6}$$

kde účelová funkce (2.1) minimalizuje celkovou ujetou vzdálenost obchodním cestujícím za podmínek:

- ▷ (2.2) - z každého města i se vyjede právě jednou,
- ▷ (2.3) - každé město j může být navštíveno pouze jednou,
- ▷ (2.4) - zamezuje tvoření nežádoucím podcyklům navíc mezi městy,
- ▷ (2.5) - binarita proměnných x_{ij},
- ▷ (2.6) - usměrňuje u_i do množiny kladných celých čísel.

Podmínku (2.4) nazýváme smyčkovou (anticyklickou) podmínkou, označovanou jako SEC (subtour elimination constraints). V této podobě jí říkáme MTZ podle autorů Millera, Tuckera a Zemlina. Poslední podmínka (2.6) může být vynechána a u_i určeno jako volná proměnná, nicméně pak tato proměnná ztrácí na svém významu, ale optimální řešení bude stále nalezeno.

2.1.2 Složitost TSP a P-NP problém

Úloha typu TSP má však obrovský problém v podobě množství navštívených měst. S nízkým počtem měst není až tak složité vyzkoušet všechny varianty a porovnat celkové vzdálenosti. Pokud však počet měst roste, počet všech přípustných řešení κ, která je potřeba prozkoumat, se navyšuje faktoriálním růstem, který lze vyjádřit jako

$$\kappa = \frac{(n-1)!}{2}, \quad n \geq 3, \tag{2.7}$$

kde n je počet měst. Jelikož předpokládáme symetrický TSP, můžeme předpokládat $2\times$ menší počet kombinací, protože lze každou cestu použít dvěma směry tam a zase zpátky, což vysvětluje dělení dvěma ve vzorci (2.7).

Přehled, jak rychle se problém stává těžším i při poměrně malém množství měst, si prohlédněme v tabulce 2.1, kde od $n = 25$ odhadujeme κ pouze řádově.

Tabulka 2.1: Přehled počtu měst a možných řešení TSP.

Počet měst n	Počet kombinací κ
3	1
4	3
5	12
6	60
7	360
8	2 520
9	20 160
10	181 440
11	1 814 400
13	239 500 800
15	43 589 145 600
18	177 843 714 048 000
20	60 822 550 204 416 000
25	$3{,}102 \cdot 10^{23}$
30	$4{,}421 \cdot 10^{30}$
32	$4{,}111 \cdot 10^{33}$
34	$4{,}342 \cdot 10^{36}$
38	$6{,}882 \cdot 10^{42}$
40	$1{,}020 \cdot 10^{46}$
46	$5{,}981 \cdot 10^{55}$
49	$6{,}207 \cdot 10^{60}$
53	$4{,}033 \cdot 10^{67}$
68	$1{,}824 \cdot 10^{94}$
156	$2{,}395 \cdot 10^{273}$

TSP spadá mezi nedeterministicky polynomiální (NP) problémy. Znamená to, že řešení může software hledat velmi dlouhou dobu, ovšem ověřit správnost výsledku zvládne v polymoniálním (P) čase. Laicky řečeno, optimální řešení lze ověřit během několika vteřin.

Jedním z největších nevyřešených otázek matematiky pro třetí tisíciletí je problém toho, zda se P rovná či nerovná NP. Tento a další problémy podrobně rozebírá (Devlin, 2005). Za vyřešení těchto problémů nabízí Clayův matematický ústav 1 000 000 amerických dolarů. Pokud by platilo, že P se rovná NP, pak by existovalo elegantní řešení TSP problému, které by úlohu vyřešilo ve znatelně kratším čase. Pokud by platilo, že P se nerovná NP, pak se nic nezmění a přikloníme se k současným řešením problému. Existují dva typy:

1. Najdeme přibližný optimální výsledek, který bude daný s určitou přesností α např. 5 %.
2. Hledáme přesný výsledek, ovšem softwaru pomůžeme s ořezáním neefektivních cest (např. mezi nejsevernějšími a nejjižnějšími městy). Tato úprava však potřebuje pro každou úlohu svoji modifikaci a úloha od úlohy se liší, jak uvádí (Devlin, 2005).

2.2 Formulace MTSP

MTSP vychází z TSP a platí pro něj stejné definice již použitého značení v kapitole 2.1.1. MTSP je ovšem složitější o počet obchodních cestujících, který budeme označovat jako m. Speciální případ $m = 1$ odpovídá obecnému TSP. Pro $m \geq 2$ již úlohu specifikujeme pouze jako MTSP. Lze tedy předpokládat, že složitost úlohy se bude zvyšovat s počtem obchodních cestujících, ale nemusí to být pravidlem, jak uvádí (Bektas, 2006).

Tuto úlohu lze rozdělit do dvou kategorií:

- ▷ single depot - každý obchodní cestující vychází ze stejného města,
- ▷ multiple depot - každý obchodní cestující vychází z jiného města.

V této práci se budeme držet první varianty.

Dále definujeme L jako maximální počet navštívených měst jedním obchodním cestujícím. Tento problém lze matematicky zapsat mnoha způsoby, jak blíže specifikuje (Bektas, 2006). Následovně uvedeme modely MIP, které jsou pro tuto práci stěžejní.

2.2.1 Matematický model MTSP

Jeden z modelů, který uvádí (Bektas, 2006, kap. 4.1) pro jediné výchozí místo, vychází z TSP. Podmínky jsou velmi podobné, jen jsou upraveny pro počet obchodních cestujících m a maximální počet měst, která mohou jednotliví obchodní cestující navštívit L.

Model MTSP lze zapsat následovně:

$$Z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \longrightarrow MIN, \quad i \neq j, \tag{2.8}$$

$$\sum_{j=2}^{n} x_{1j} = m, \tag{2.9}$$

$$\sum_{i=2}^{n} x_{i1} = m, \tag{2.10}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 2,\ldots,n, \tag{2.11}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 2,\ldots,n, \tag{2.12}$$

$$u_i - u_j + Lx_{ij} \leq L - 1, \quad i \neq j, i = 1,\ldots,n,\ j = 2,\ldots,n, \tag{2.13}$$

$$x_{ij} \in \{0;1\}, \quad \forall i, \forall j, \tag{2.14}$$

$$u_i \in \mathbb{Z}^+, \quad \forall i, \tag{2.15}$$

kde účelová funkce (2.8) minimalizuje celkovou ujetou vzdálenost všech obchodních cestujících za podmínek:

- ▷ (2.9) - ze startovního města se vyjede právě m-krát,
- ▷ (2.10) - do startovního města se vjede právě m-krát,
- ▷ (2.11) - z každého města i, kromě startovního, se vyjede právě jednou,
- ▷ (2.12) - každé město j, kromě startovního, může být navštíveno pouze jednou,
- ▷ (2.13) - zamezí nežádoucím podcyklům navíc mezi městy (vycházející z SEC-MTZ),
- ▷ (2.14) - binarita proměnných x_{ij},
- ▷ (2.15) - usměrňuje u_i do množiny kladných celých čísel.

Pokud budeme chtít, aby obchodní cestující nebyli omezeni maximálním množstvím měst, která mohou navštívit, pak platí, že $L = n$.

2.2.2 Podmínky Kara a Bektas

Pro SEC-MTZ (2.13) v modelu MTSP lze využít i jiných variant. Např. (Kara et al., 2006, kap. 2) rozšířili základní model o proměnnou K, která umožňuje nastavit minimální množství měst, která musí jednotliví obchodní cestující navštívit a upravili SEC-MTZ do více podmínek.

Tyto podmínky budeme dále označovat zkratkou KaBe podle autorů Kara a Bektas[1]. KaBe-SEC platí pouze pro smysluplné zadání minimálního a maximálního množství navštívených měst všemi obchodními cestujícími, tedy $2 \leq K \leq \lfloor (n-1)/m \rfloor$ a $K \leq L \leq n$. Pokud budeme chtít pomocí těchto podmínek řešit pouze TSP, pak $L = n$ a $K \leq L - 1$.

[1] Kulkarni a Bhave také definovali SEC, jejichž verze používá zkratky KB, proto se této alternativě vyhneme.

KaBe-SEC můžeme formulovat následovně:

$$u_i + (L-2)x_{1i} - x_{i1} \leq L-1, \qquad i = 2,\ldots,n, \tag{2.16}$$

$$u_i + x_{1i} + (2-K)x_{i1} \geq 2, \qquad i = 2,\ldots,n, \tag{2.17}$$

$$x_{1i} + x_{i1} \leq 1, \qquad i = 2,\ldots,n, \tag{2.18}$$

$$u_i - u_j + Lx_{ij} + (L-2)x_{ji} \leq L-1, \qquad i \neq j, i = 1,\ldots,n,\ j = 2,\ldots,n, \tag{2.19}$$

kde (2.19) má stejný význam jako (2.13), tedy zabránit tvorbě dodatečných nežádoucích podcyklů. Podmínka (2.16) vytvoří horní mez pro počet navštívených měst jedním obchodním cestujícím a podmínka (2.17) vytvoří dolní mez pro totéž. Nakonec podmínka (2.18) je potřeba pouze pro $K = 2$ a $K = 3$. Doplňuje totiž podmínky (2.16) a (2.17), pro které v tomto případě neplatí rovnost $x_{1i} = x_{i1} = 1$.

2.2.3 Model MTSP podle VRP a tří-indexové proměnné

MTSP lze také definovat jako VRP, od kterého odebereme podmínky pro omezení kapacity a ceny vozidla, jak uvádí (Bektas, 2006, kap. 4.4). Indikátorovou proměnnou x_{ij} rozšíříme o index $k = 1,\ldots,m$ představující obchodní cestující na x_{ijk}. Pokud je rovna 1, pak proběhne cesta z města i ihned do města j a provede ji obchodní cestující k. Pokud je rovna 0, pak nikoliv. Maximální počet měst, která musí jednotliví obchodní cestující navštívit, odpovídá proměnné L. V této úloze platí, že pomocný index p nabývá hodnot $p = 1,\ldots,n$.

Bohužel se však (Bektas, 2006, kap. 4.4) dopustil v modelu dvou chyb v indexech. Nejprve si však ukažme opravený model založený na modelu VRP podle (Christofides et al., 1981, kap. 2), který vypadá následovně:

$$Z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \sum_{k=1}^{m} x_{ijk} \longrightarrow MIN, \tag{2.20}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ijk} = 1, \quad j = 2,\ldots,n, \tag{2.21}$$

$$\sum_{i=1}^{n} x_{ipk} - \sum_{j=1}^{n} x_{pjk} = 0, \quad \forall k, \forall p, \tag{2.22}$$

$$\sum_{j=2}^{n} x_{1jk} = 1, \quad \forall k, \tag{2.23}$$

$$u_i - u_j + L\sum_{k=1}^{m} x_{ijk} \leq L-1, \quad i \neq j, i = 1,\ldots,n,\ j = 2,\ldots,n, \tag{2.24}$$

$$x_{ijk} \in \{0;1\}, \quad \forall i, \forall j, \forall k, \tag{2.25}$$

$$u_i \in \mathbb{Z}^+, \quad \forall i, \tag{2.26}$$

kde účelová funkce (2.20) minimalizuje ujetou vzdálenost všech obchodních cestujících za podmínek:

- ▷ (2.21) - každé město kromě startovního bude navštíveno právě jednou,
- ▷ (2.22) - každý vjíždějící obchodní cestující do města ho musí také opustit,
- ▷ (2.23) - každý obchodní cestující vytvoří jeden okruh,
- ▷ (2.24) - zamezí nežádoucím podcyklům navíc mezi městy,
- ▷ (2.25) - binarita proměnných x_{ijk},
- ▷ (2.26) - usměrňuje u_i do množiny kladných celých čísel.

Anticyklická podmínka (2.24) je pouhým rozšířením MTZ-SEC pro tří-indexovou proměnnou x_{ijk} vycházející z SEC-MTZ. V původní formulaci byla místo proměnné pro maximální počet navštívených měst L uvedena proměnná pro počet měst n. Tuto obměnu provedeme z důvodu porovnání modelů v praktické části.

Jak již bylo řečeno, (Bektas, 2006, kap. 4.4) se bohužel dopustil v tomto modelu dvou chyb v indexech:

- ▷ První chyba se vyskytla u podmínky (2.21), kde (Bektas, 2006, kap. 4.4) říkal, že tato podmínka proběhne pro $j = 1, \ldots, n$. Kdyby tomu tak bylo, pak by nemohli všichni obchodní cestující cestovat z libovolného města i do startovního města, tedy $j = 1$. Do startovního města by se mohl vrátit pouze jeden obchodní cestující a bylo by porušeno zadání úlohy. Tuto chybu jsme tedy vyřešili upravením indexu j na $j = 2, \ldots, n$, abychom cestu zpět do startovního města umožnili všem obchodním cestujícím.
- ▷ Druhé chyby se dopustil při psaní podmínky (2.23), kde (Bektas, 2006, kap. 4.4) říkal, že levá strana podmínky má podobu následující $\sum_{j=1}^{n} x_{1jk}$. Kdyby levá strana takto platila, pak by mohl obchodní cestující také zůstat ve startovním městě a nikam necestovat. Tím by bylo opět porušeno zadání MSTP problému, tentokrát ve snížení počtu okruhů m, které je potřeba vytvořit.

Pravdou je, že druhá chyba by se dala ještě tolerovat, protože maximálně $m - 1$ obchodních cestujících by mohlo zůstat ve startovním městě a nikam necestovat. Tomuto problému by se dalo také vyhnout, kdyby cena startovního města c_{ij}, kde $i = j = 1$, byla rovna kladnému nekonečnu nebo dostatečně velkému číslu M, např. $M = 10^5$. Teoreticky bychom také mohli cenu ponechat 0 a pro výpočet optimální cesty by byl vždy použit nejlepší možný počet obchodních cestujících pro minimalizaci Z. Pro některé případy by mohlo být optimální použít například o jednoho obchodního cestujícího méně, ten by tedy nikam necestoval.

2.2.4 Podmínky Kara a Bektas pro tří-indexovou proměnnou

Podmínky KaBe-SEC lze upravit i pro tří-indexové proměnné x_{ijk}. Platí pro ně stejná pravidla jako v kapitolách 2.2.2 a 2.2.3. Pro úpravu SEC dle KaBe na tří-indexové proměnné x_{ijk}, které nahrazují podmínku (2.24), je potřeba drobných úprav pomocí sumarizací pro indikátory x_{ijk}.

Po úpravách vypadají podmínky následovně:

$$u_i + (L-2)\sum_{k=1}^{m} x_{1ik} - \sum_{k=1}^{m} x_{i1k} \leq L-1, \qquad i = 2,\ldots,n, \tag{2.27}$$

$$u_i + \sum_{k=1}^{m} x_{1ik} + (2-K)\sum_{k=1}^{m} x_{i1k} \geq 2, \qquad i = 2,\ldots,n, \tag{2.28}$$

$$\sum_{k=1}^{m} x_{1ik} + \sum_{k=1}^{m} x_{i1k} \leq 1, \qquad i = 2,\ldots,n, \tag{2.29}$$

$$u_i - u_j + L\sum_{k=1}^{m} x_{ijk} + (L-2)\sum_{k=1}^{m} x_{jik} \leq L-1, \qquad i \neq j, \forall i,j = 2,\ldots,n, \tag{2.30}$$

kde (2.30) má stejný význam jako (2.24), tedy zabránit tvorbě dodatečných nežádoucích podcyklů. Podmínka (2.27) vytvoří horní mez pro počet navštívených měst jedním obchodním cestujícím a podmínka (2.28) vytvoří dolní mez pro totéž. Nakonec podmínka (2.29) je potřeba pouze pro $K = 2$ a $K = 3$. Doplňuje totiž podmínky (2.27) a (2.28), pro které v tomto případě neplatí rovnost $\sum_{k=1}^{m} x_{1ik} = \sum_{k=1}^{m} x_{i1k} = 1$.

2.3 Metoda větví a mezí

Jelikož budeme řešit úlohy i o velké složitosti $n = 156$, pro které exaktní metody nemohou nalézt optimální řešení, využijeme toho, že uvedené matematické modely jsou typu MIP. Pro tento typ úloh můžeme použít k nalezení řešení metody větví a mezí (B&B), také známé jako metoda větvení a hranic.

Při řešení pomocí této metody se TSP dělí do menších podúloh, které jsou vyřešeny jako dílčí problémy. Složením těchto dílčích podúloh vznikne řešení úlohy jako celku. Důležitou vlastností algoritmu je, že v dílčích úlohách dochází k oříznutí variant, které nemohou být optimální, a znatelně tím zkracují dobu výpočtu, jak uvádí (Balas et al., 1983).

Tento algoritmus je také implementován do kódu CONCORD nebo také do řešitele GUROBI, který podrobněji rozebereme v kapitole 3.5.

2.4 Heuristiky pro řešení TSP

Jelikož je velice náročné hledání optimálního řešení, existují heuristiky, které mohou najít přípustné řešení. Takovéto řešení pouze splňuje všechny dané podmínky, avšak nemusí být nutně optimální. Heuristiky pro hledání přípustných řešení, které si zde uvedeme, bude však možné vztahovat pouze k řešením jednotlivých TSP.

Heuristické metody v tabulce 2.2 nejsou navržené tak, aby hledaly přímo skutečné optimální řešení TSP, ale tak, aby zjednodušenou formou nalezly nějaké rozumné přípustné řešení. Jedná se tedy o nástřel, který nám alespoň přibližně určí, kde se bude hodnota optimální hodnoty účelové funkce pohybovat.

Tabulka 2.2: Přehled heuristik pro řešení TSP.

Zkratka	Původní název	Český překlad
NN	Nearest Neighbour	Metoda nejbližšího souseda
C-W	Savings algorithm, Clark-Wright	Metoda výhodnostních čísel
-	Insertion method	Metoda vkládací
CH	Convex hull	Metoda konvexního obalu
MST	Minimum Spanning Tree algorithm	Metoda minimální kostry
-	Christofides algorithm	Christofidova metoda
NM	Nearest Merger method	Metoda zatřiďovací
LK, LKH	Lin-Kernighan heuristic	Metoda výměn či Lin-Kernighanova heuristika

2.4.1 Metoda nejbližšího souseda

Tato metoda se také označuje jako „greedy metoda.“ Nezískáme příliš dobré řešení, ale pro výpočet potřebujeme pouze řádově n^2 operací.

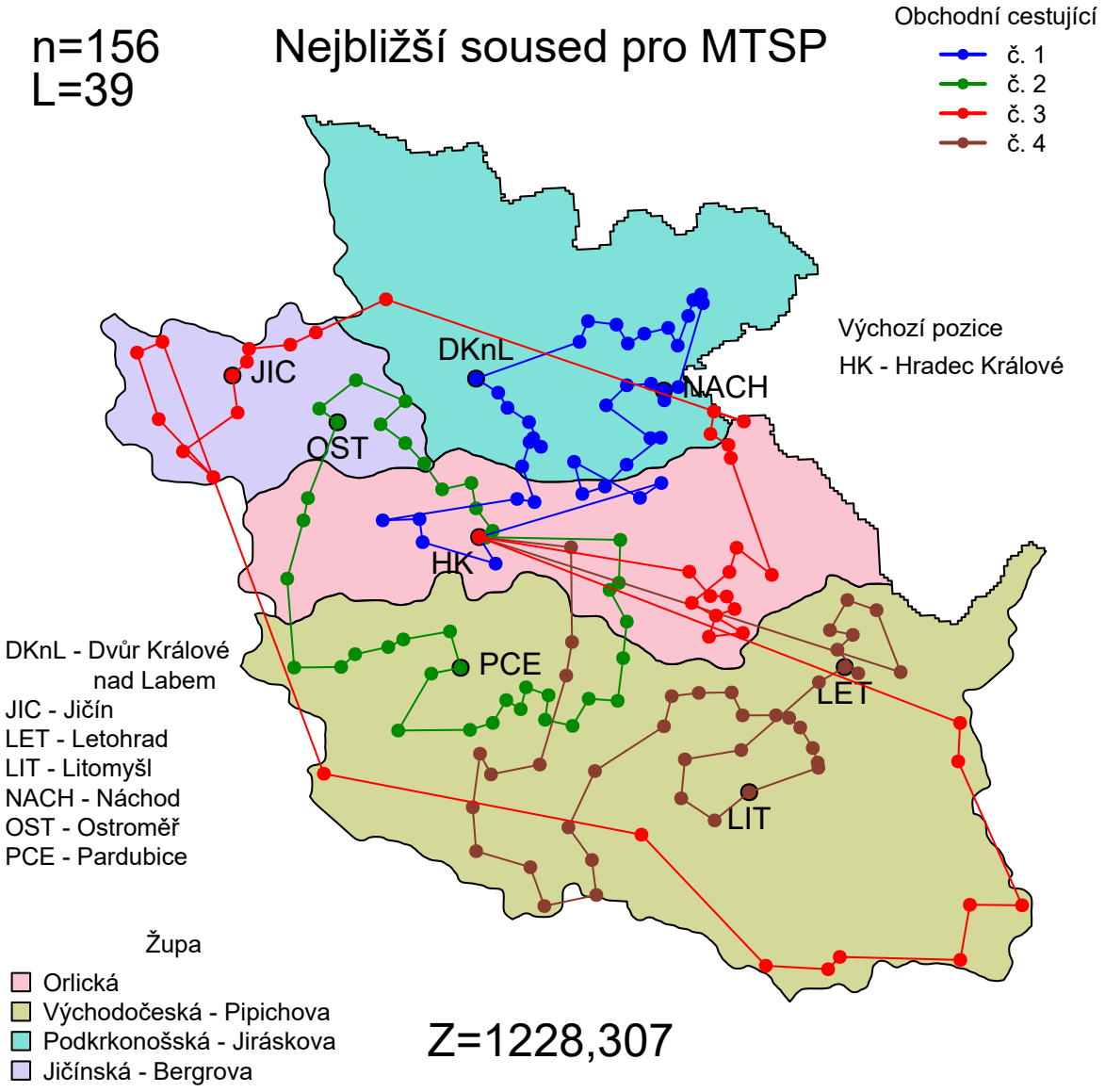
Začínáme ve startovním městě. Hledáme cestu z tohoto města, která je ze všech dostupných cest nejkratší. Tu zvolíme a dorazíme po ní do druhého města. Opět hledáme nejkratší hranu do města, které ještě nebylo navštíveno, a provedeme cestu do města třetího. Nyní metodu opakujeme do té doby, než navštívíme všechna města. Jakmile navštívíme poslední město, vrátíme se do startovního města a ukončíme výpočet.

Tuto metodu lze snadno rozšířit na úlohu více obchodních cestujících. Nastavíme $L = \lceil \frac{n}{m} \rceil$, čímž docílíme, že každý obchodní cestující projede přibližně stejný počet měst. Ukázku takového výstupu můžeme vidět na obrázku 2.1. Při porovnání s výsledky z kapitoly 4.3 vidíme, že tento prostý algoritmus trpí značnými nedostatky. Obchodní cestující č. 3, který vyrazil jako poslední, je zde donucen projet zbylá města rozprostřená po celé oblasti.

2.4.2 Metoda výměn

Metodu výměn (LK, LKH) vymysleli Lin a Kernighan, jak uvádí (Pelikán, 2001, kap. 9). Podstata metody vychází ze zlepšování již existujícího přípustného řešení. Vždy vybereme dvě hrany, po kterých obchodní cestující v aktuálním řešení prochází. Ty vyměníme s jinými dvěma hranami tak, aby opět vznikl Hamiltonův cyklus. Pokud došlo ke zlepšení, pak změnu ponecháme, pokud ne, změnu neprovedeme. Takto opakujeme, dokud je možné řešení zlepšovat. Postup popsaný výše se nazývá 2-opt (optimální). Lze také zvolit obecný počet k vyměněných hran, což nazýváme k-opt.

Tuto metodu rozšířil Helsgaun. Jeho přístup se dnes považuje za jeden z nejúčinnějších. Více informací lze dohledat na osobních internetových stránkách (Helsgaun, 2020).



Obrázek 2.1: Aplikace metody nejbližšího souseda na jednoty z Královéhradeckého a Pardubického kraje pro čtyři obchodní cestující.

2.5 Metaheuristiky pro řešení TSP

Metahauristika je heuristika, jejíž algoritmus lze použít obecně na více typů úloh a ne pouze na jednu jedinou. Můžeme je tedy použít i pro problémy, které nemají s TSP mnoho společného. Značné množství těchto heuristik zmiňují (Pelikán, 2001, kap. 9) a (Bonyadi et al., 2008). Přehled těchto metaheuristik, viz tabulka 2.3. Tyto metaheuristiky čerpají z mnoha exaktních, populačních, přírodních a evolučních metod.

Tabulka 2.3: Přehled metaheuristik pro řešení TSP.

Zkratka	Původní název	Český překlad
CP	Cutting planes	Metoda sečných nadrovin
LS	Local search method	Metoda lokálního hledání
TS	Tabu search	Tabu search metoda
TA	Threshold accepting method	Metoda prahové akceptace
SIAM, SA	Simulated annealing	Metoda simulovaného žíhání
EA	Evolutionary algorithms	Evoluční algoritmus
GA	Genetic algorithms	Genetický algoritmus
ACO	Ant colony optimization	Optimalizace mravenčí kolonií
PSO	Particle swarm optimization	Optimalizace rojů částic
IWD	Intelligent water drops	Inteligentní kapky vody
AIS	Artificial immune systems	Umělé imunitní systémy
IGA	Immune genetic algorithm	Imunitní genetický algoritmus
BCO	Bee colony optimization	Optimalizace včelí kolonie
EM	Electromagnetism-like Mechanisms	Elektromagnetické mechanismy

2.5.1 Optimalizace mravenčí kolonií

Optimalizace mravenčí kolonií (ACO) vychází z přírody podobně jako optimalizace včelí kolonií (BCO). Mravenci si při cestě z mraveniště za potravou předávají informace pomocí feromonu. Ten mravenci řekne, kde lze ucítit potravu a které cesty již využili ostatní mravenci. Mravenec se pak sám rozhodne, jestli použije cestu, kterou už použil jiný mravenec, nebo třeba použije cesty od dvou předchůdců a zkombinuje je. Také se může rozhodnout najít úplně novou cestu. Naneštěstí ani feromon netrvá věčně, takže pokud feromon vyprchá, mravenec bude hledat novou cestu. Tímto způsobem může nalézt lepší nebo i kratší cestu od potravy k mraveništi. Tento myšlenkový proces proběhne u každého mravence, který se k potravě vydá, takže pro programátora tohoto kódu se zde nabízí vytvořit výpočetní smyčku.

Ve skutečnosti však ACO využívá několik mravenců. Ti spolupracují a předávají si informace skrz feromon na každé hraně grafu. Algoritmus se tedy snaží simulovat chování mravenců, jak prezentuje (Junjie et al., 2006).

2.5.2 Genetický algoritmus

Genetický algoritmus (GA) je jeden z nejstarších metaheuristických algoritmů na světě. Tento algoritmus adaptivně hledá postup řešení pro kombinatorickou optimalizaci založený na přirozené genetice a přírodním výběru, jak sděluje (Sedighpour et al., 2012).

GA začíná s počáteční populací (přípustné řešení) a genomem určeným pomocí chromozomu. Následně „fitness“ funkce určí, jak moc je toto řešení optimální. Pokud již máme optimální řešení, můžeme proces ukončit. Pokud nemáme optimální řešení, pak vybereme vysoce zdatné jedince (nízké vzdálenosti) a pomocí křížení (prohození pořadí cest), mutace (náhodná nová cesta) a reprodukce (stejná cesta) vygenerujeme nové řešení. Nyní přepočteme „fitness“ funkci a pokud řešení není optimální, postup opakujeme od výběru nových zdatných jedinců. Pokud máme optimální řešení, proces ukončíme. Pokud dosáhneme určeného množství reprodukčních cyklů, pak vybereme dosud nejlepší řešení.

3. Zpracování dat

Ještě než se pustíme do výpočtů, definujme a formulujme si podobu dat, na kterých budeme provádět výpočetní experimenty. Pro problém typu TSP potřebujeme sadu bodů a vzdáleností mezi nimi. V této kapitole tedy podrobně rozebereme, jak se k těmto datům dostaneme, jak vypočteme jednotlivé vzdálenosti, jaký program použijeme pro hledání optimálního řešení a způsob, jak budeme vykreslovat výsledky na mapě.

3.1 Sokol

Sokol je tělocvičná organizace založená roku 1862 Miroslavem Tyršem a Jindřichem Fügnerem. Má tedy za sebou více než 150letou bohatou historii plnou rozkvětu, ale i zkázy způsobenou minulými režimy. Po dobu existence bylo založeno po celé České republice značné množství jednot. Toto číslo dnes přesahuje 1000 jednot. Každá jednotlivá jednota v historii vlastnila budovu sokolovny (tělocvičny), s jejichž lokalitami budeme pracovat.

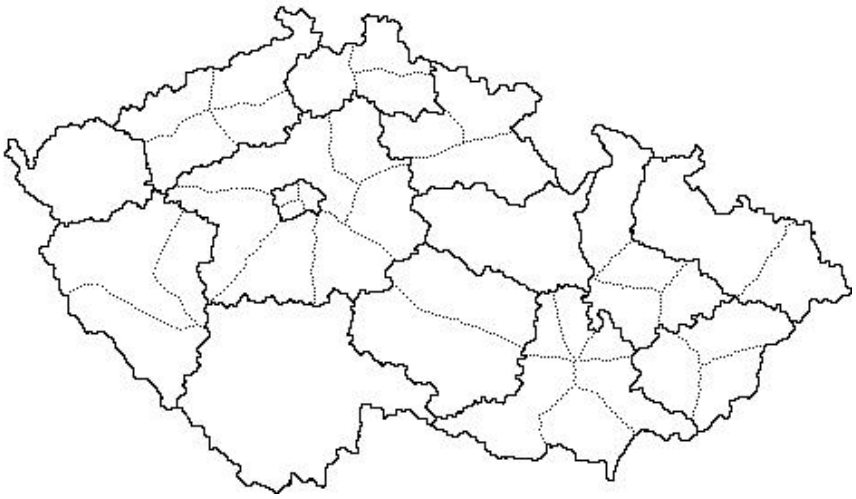
Sokolové dělí Českou republiku do 42 žup (správních celků), jejichž dělení se podobá okresům, ovšem není to pravidlem. Výpočetní metody budeme používat na jednotlivé župy, které se liší počtem sokoloven. Znázornění žup je na obrázku 3.1.

3.2 Datový soubor

Vytvořili jsme si soubor Microsoft Excel (dále Excel) viz příloha B. Pro vybrané župy, viz tabulka 3.1, kde n značí počet jednot, byl založen list. Do každého listu byly následně vloženy všechny jednoty, které spadají do dané župy. Pro každou jednotu jsme zapsali GPS souřadnice, které byly dohledány na internetové mapě (Seznam.cz, 2020) pomocí adres, které jsou uvedené na internetových stránkách (Česká obec sokolská, 2020). Z této stránky jsme také převzali seznam všech jednot i těch, které byly již zrušeny. Pro ověření bylo ještě použito nezávislých internetových stránek (Pávek, 2020).

Pro některé jednoty jsou však na obou stránkách uvedeny pouze korespondenční adresy. Pokud se nám podařilo dohledat přesnou adresu sokolovny pomocí webových stránek jednotlivých sokoloven, pak GPS souřadnice upravíme tak, aby odpovídaly co nejlépe. Pokud ne, ponechali jsme alespoň souřadnice korespondenční adresy, protože na tomto místě nalezneme osoby spojené s místním sokolem, a sokolovnu pak můžeme dohledat přímo na místě. Ve většině případů se jednalo o obecní úřady nebo rodinné domy. Jedná se o kompromis, který bylo vzhledem k objemu dat nutné přijmout, zvláště pak u malých obcí. Neměl by však závažně pohnout s přesností.

Nakonec jsme ještě přidali výchozí bod, kterým bude ústředí jednotlivých žup. Pro některé



Obrázek 3.1: Mapa rozdělení žup v České republice, autor (Pávek, 2020).

Tabulka 3.1: Přehled žup použitých v této práci.

n	Župa	Jméno v souboru
45	Barákova	barakova
40	Dr. Jindry Vaníčka	dr_jindry
53	Jana Máchala	jana_ma
20	Ještědská	jestedska
18	Jičínská - Bergrova	jicinska
25	Komenského	komen
15	Krušnohorská - Kukaňova	krusnohorska
34	Olomoucká - Smrčkova	olomoucka
38	Orlická	orlicka
30	Plzeňská	plzenska
32	Podkrkonošská - Jiráskova	podkrkonoska
50	Tyršova	tyrsova
8	Valašská - Františka Palackého	valasska
68	Východočeská - Pippichova	vychodoceska

župy je toto ústředí totožné s jednotou (např. župa Jičínská - Bergrova, župa Orlická), pro některé tomu tak není (např. župa Barákova). Nicméně tento výchozí bod je pro každou župu umístěn v souboru na první pozici ($i = 1$, $j = 1$).

Ukázku takto získaných dat si prohlédněme v tabulce 3.2 na župě Valašské s nejmenším množstvím jednot v České republice.

Pro otestování náročnější úlohy jsme využili spojení žup nacházejících se v Královéhradeckém a Pardubickém kraji. Tyto župy spolu úzce spolupracují, tudíž se jednalo o jasnou volbu pro testování většího souboru. Spojili jsme tedy data žup Orlické, Jičínské - Bergrovy, Podkrkonošské - Jiráskovy a Východočeské - Pipichovy. List byl pojmenován „KH_PCE“ a obsahuje 156 jednot. Startovním bodem byla zvolena jednota Pražské Předměstí z župy Orlické nacházející se v Hradci Králové.

Tabulka 3.2: Župa Valašská - Františka Palackého.

i	Jednota	severní šířka [°]	východní délka[°]
1	Valašské Meziříčí	49,470 446 7	17,963 012 2
2	Hošťálkova	49,357 096 9	17,876 137 5
3	Jablůnka - Pržno	49,387 901 1	17,949 953 9
4	Jasenná	49,250 366 1	17,895 071 1
5	Karolinka	49,351 355 0	18,239 908 1
6	Rožnov pod Radhoštěm	49,465 157 8	18,149 368 3
7	Valašské Klobouky	49,139 503 3	18,007 590 3
8	Vsetín	49,338 354 4	17,991 216 1

3.3 Výpočet vzdálenosti

Nyní potřebujeme získat vzdálenosti mezi jednotlivými sokolovnami. Existuje několik možností, jak vypočítávat vzdálenosti mezi dvěma body. My však použijeme variantu, ve které lze přesně zaznamenat bod na mapě, a tím jsou GPS souřadnice, což vzhledem k povaze sběru dat bude ideální přístup. Proto v této práci pro výpočet vzdálenosti použijeme Haversinova vzorce (3.1), jak jej uvádí (Chopde et al., 2013, kap. III.), kde vzdálenost mezi městy (jednotami) i a j definovanou jako c_{ij} vypočítáme následovně:

$$c_{ij} = 2R \arcsin \sqrt{\sin^2\left(\frac{\varphi_j - \varphi_i}{2}\right) + \cos\varphi_i \cos\varphi_j \sin^2\left(\frac{\psi_j - \psi_i}{2}\right)}, \tag{3.1}$$

kde

- R je poloměr země v km, pro naše data je vhodný kvocient $R = 6371$ km,
- φ_i a φ_j značí GPS hodnotu zeměpisné šířky (latitude) měst i a j v radiánech,
- ψ_i a ψ_j značí GPS hodnotu zeměpisné délky (longitude) měst i a j v radiánech.

Tento vzorec přeformulujeme do prostředí Excel.

```
=ZAOKROUHLIT(2*6371*ARCSIN(ODMOCNINA(
            (SIN((RADIANS(D$2)-RADIANS($B4))/2))^2
            +COS(RADIANS($B4))*COS(RADIANS(D$2))*
            (SIN((RADIANS(D$3)-RADIANS($C4))/2)^2)
       ));3)
```

Buňky `$B4` a `$C4` označují GPS lokátor zeměpisné šířky a délky pro sokolovnu i, buňky `D$2` a `D$3` označují GPS lokátor zeměpisné šířky a délky pro sokolovnu j. Všechny souřadnice se nacházejí na polokouli severní šířky a východní délky.

Pomocí funkce `RADIANS` převádíme stupně na radiány vyžadované vzorcem (3.1). Funkce `ZAOKROUHLIT` zaokrouhlí tyto hodnoty na 3 desetinná čísla, budeme tedy počítat s přesností na metry. Ostatní funkce vychází z Haversinova vzorce (3.1).

Tabulku 3.2 obsahující souřadnice jednot župy Valašské jsme pro ukázku výše uvedeným postupem převedli na tabulku 3.3 vzdáleností v kilometrech.

Tabulka 3.3: Matice vzdáleností [km] jednot župy Valašské - Františka Palackého.

i/j	1	2	3	4	5	6	7	8
1	0	14,084	9,227	24,962	24,014	13,479	36,941	14,829
2	14,084	0	6,348	11,947	26,356	23,133	26,009	8,593
3	9,227	6,348	0	15,802	21,385	16,787	27,935	6,267
4	24,962	11,947	15,802	0	27,410	30,160	14,792	12,014
5	24,014	26,356	21,385	27,410	0	14,249	28,971	18,074
6	13,479	23,133	16,787	30,160	14,249	0	37,642	18,160
7	36,941	26,009	27,935	14,792	28,971	37,642	0	22,143
8	14,829	8,593	6,267	12,014	18,074	18,160	22,143	0

3.4 MPL for Windows

Z široké nabídky softwarů, které máme k dispozici, si vybereme MPL for Windows (Mathematical Programming Language), který vytvořila společnost Maximal Software, Inc. Tento software je otevřený, tzn. že do něj můžeme implementovat další řešitele, jejichž licence můžeme volně stáhnout, dokoupit nebo získat zdarma za studijním účelem. Ve výpočetní části použijeme řešitele Gurobi, blíže popsaného v kapitole 3.5. Dále se nabízí například řešitelé CPLEX 300 nebo Lindo podle (Maximal Software, 2020). V této práci používáme verzi MPL for Windows 5.0.8.116 (64-bit).

MPL slouží k formulování komplikovaných optimalizačních modelů čistou, stručnou a efektivní cestou. Uživatel může zadávat algebraické rovnice a nemusí se starat tolik o programování, jak uvádí (Maximal Software, 2020). MPL může také načítat data z databází, např. z Excelu a po výpočtu i exportovat získané výsledky. Může také komunikovat s dalšími aplikacemi ve Windows.

Pro zápis modelu do skriptu v MPL se užívá zavedeného pořádku. Každý řádek musí být ukončen středníkem „;“. Jedinou výjimkou jsou příkazy, které oddělují skript do několika částí. Platí pro ně přesné pořadí, ve kterém mohou, ale nemusí, být zapsány ve skriptu. Jsou to následující příkazy.

TITLE
Zde můžeme náš skript pojmenovat. Jméno skriptu však nesmí obsahovat mezery.

OPTIONS
Tato oblast slouží pro nastavení skriptu. Můžeme určit, z kterého souboru má MPL načítat data. To můžeme udělat pomocí následujícího příkazu.
`ExcelWorkBook="jméno souboru.xlsx";`
Pokud máme více listů v souboru, upřesníme list k načítání pomocí následujícího příkazu.
`ExcelSheetname="jméno listu";`

INDEX
Pokud máme větší úlohu, hlavně ve dvou a více dimenzích, indexování nám usnadní orientaci například v maticích. Hodnoty mohou být jak číselné, tak textové. Můžeme je také načítat z Excelu pomocí následující funkce.
`ExcelRange("pojmenování indexu")`

DATA
Data, která použijeme při výpočtu, vypíšeme právě v této části. Lze také načítat z jiných aplikací, např. z Excelu. Pro načítání opět použijeme následující funkci.
`ExcelRange("pojmenování oblasti dat")`

VARIABLES
V odstavci pro proměnné definujeme, jakého typu mohou proměnné nabývat. Ať už se jedná o binární (BINARY), celočíselné (INTEGER) nebo třeba neomezené (FREE) proměnné. Pokud si budeme přát exportovat data do Excelu, použijeme následující funkci.
`Export to ExcelRange("pojmenování cílové oblasti")`

MACROS
Pod tímto příkazem můžeme definovat vlastní makra, např. pro často používané vzorce. Makra mohou také v některých případech sloužit k eliminaci nepotřebných či nesmyslných rovností omezení nebo např. snížit velikost problému, a tím usnadnit výpočet pro řešitele.

MODEL
Do této oblasti zapíšeme výpočetní vzorec pro účelovou funkci. Výpočet začínáme frází MIN nebo MAX podle toho, jestli chceme minimalizovat nebo maximalizovat účelovou funkci.

SUBJECT TO
Část, do které se vkládají omezující podmínky. Každou podmínku musíme pojmenovat, vzhledem k počtu některých omezení je toto více než vhodné.

BOUNDS
V této sekci lze omezit proměnné horními a dolními mezemi. V rámci práce nebudeme tuto sekci potřebovat.

END
Ukončuje celý model.

Pokud chceme vytvořit ve skriptu komentář, abychom si poznamenali, co se v dané části skriptu počítá nebo jaká je myšlenka výpočtu, použijeme před textem komentáře vykřičník „!“.

Pro detailnější popis a výpis všech funkcí můžeme využít manuál (Maximal Software, 2018b, kap. 5).

3.5 GUROBI

Řešitel GUROBI byl vyvinut společností Gurobi Optimization, LLC. Verze GUROBI (7.5.2.), která je použita v této práci, je určena k řešení úloh typů viz tabulka 3.4. Dnes nejnovější verze řešitele tohoto typu na trhu je GUROBI (9.0).

GUROBI využívá metod CP a B&B k řezům, které mohou pomoci nalézt lepší řešení, avšak zpomalují výpočet. Agresivitu těchto řezů lze řešiteli přednastavit. Pokud se tedy metoda B&B nepohybuje perspektivní větví, může agresivnější řez řešení úlohy urychlit za přijatelnější čas. Pro získání lepšího řešení můžeme také využít více heuristik, ovšem bude zvýšena časová náročnost. Například předběžná řešení (Presolve) jsou důležitá pro úlohy třídy MIP. Jsou však velmi náročná, proto jsou ve výchozím nastavení snížena. Má tedy smysl zvolit agresivnější nastavení pro těžké úlohy kvůli urychlení výpočtu, jak uvádí (Maximal Software, 2018a, str. 6).

Tabulka 3.4: Přehled úloh řešitelných pomocí řešitele GUROBI.

zkratka	úloha
LP	Lineární programování
MIP	Smíšené celočíselné programování
Simplex based QP	Kvadratické programování založené na simplexové metodě
IPM LP	Lineární programování založené na bariérové metodě
IPM QP	Kvadratické programování založené na bariérové metodě
IPM SOCP	Programování kuželu druhého řádu založené na bariérové metodě
MIQP	Smíšené celočíselné kvadratické programování
MIQCP	Smíšené celočíselné programování kuželu druhého řádu

3.6 Jazyk R a mapa

Z několika možností, jak vykreslovat mapy a cesty obchodních cestujících mezi nimi, jsme si vybrali jazyk R a využijeme balíčku `maptools` a souboru hranic mezi kraji a okresy. Hranice můžeme získat ze souborů `CZE_adm1` a `CZE_adm2`, ze kterých vybereme potřebné hranice pro Královéhradecký a Pardubický kraj a příslušných okresů. Tyto soubory můžeme volně stáhnout z internetových stránek (Hijmans, 2020). Hranice je potřeba ořezat jen na potřebné okresy a kraje.

Souřadnice měst si načteme ze souboru `vystupy_x.xlsx` a listu `proR_KH_PCE`. K zobrazení sokolských žup použijeme GPS lokátorů. Během vykreslování sokolských jednot do mapy jsme se dozvěděli, že na Svitavsku a Trutnovsku nejsou žádné sokolské jednoty, proto jsou tato místa prázdná. Některé jednoty také spadají pod župy, které neodpovídají okresům. Takovým příkladem může být Dolní Kalná, která sice spadá pod župu Jičínskou - Bergrovu, ale územně by spíše měla patřit k župě Podkrkonošské - Jiráskově.

Pro zobrazení cest budeme potřebovat proměnné x_{ij} a x_{ijk}, které vypočítáme pomocí MPL for Windows a uložíme si je do souboru `vystupy_x.xlsx`.

Aktuální verze skriptu `mapka.R` je částečně automatizovaná. Hodnoty účelových funkcí se vkládají ručně a k načítání listů je potřeba dodávat čísla (počet hodin výpočtu) nových listů. Je určena pro $m = 4$, ale toto lze snadno pozměnit. Výsledné obrázky jsou rovnou ukládány do složky s R skriptem.

Pro zajímavost jsme také vytvořili pomocí algoritmu nejbližšího souseda řešení pro stejnou úlohu. I pro skript `greedy_MTSP.R` jsme částečně využili výše popsané vykreslení do mapy. Řešení algoritmu jsme aplikovali v prostředí jazyka R a výsledek vidíme na obrázku 2.1.

Skripty uvedené v tomto oddíle jsou součástí přílohy B.

4. Výpočetní experimenty

Ve výpočetní části se podíváme na implementaci jednotlivých modelů na soubory dat a porovnáme účinnost těchto skriptů pro různě velká množství měst, různá množství obchodních cestujících a omezení v podobě minimálního a maximálního množství měst, která mohou jednotliví obchodní cestující navštívit. Tyto varianty obměn budou provedeny pro modely, které jsou k tomu odpovídající.

Pro všechna řešení využijeme programu MPL for Windows, se kterým jsme se obeznámili v kapitole 3.4. V rámci MPL použijeme řešitele GUROBI verze (7.5.2.), o kterém jsme se zmínili v kapitole 3.5. Využijeme výchozí automatické natavení řešitele s úpravou pro časové omezení výpočtu, které budeme upravovat podle potřeby.

4.1 Implementace modelů

Všechny modely z kapitoly 2 jsme implementovali v jazyce programu MPL. Vysvětlivky ke skriptům jsou obsaženy v příloze A. Jednotlivé skripty jsou uvedeny v přílohách:

A.1 - model TSP podmínka MTZ (dále TSP-MTZ),
A.2 - model MTSP (dále MTSP),
A.3 - model MTSP dle VRP (dále VRP),
A.4 - model MTSP podmínky KaBe (dále MTSP-KaBe),
A.5 - model MTSP dle VRP podmínky KaBe (dále VRP-KaBe).

4.2 Porovnání výsledků

V tomto oddíle používáme následující symboliku:

- n - počet měst v souboru,
- m - počet obchodních cestujících,
- L - maximální množství měst, které mohou jednotliví obchodní cestující navštívit,
- K - minimální množství měst, které mohou jednotliví obchodní cestující navštívit,
- t - čas potřebný k výpočtu pro řešitele GUROBI (h = hodiny, m = minuty, s = sekundy),
- Z - minimalizovaná vzdálenost, kterou urazí všichni obchodní cestující dohromady,
- $MIPBB$ - nejlepší vazba celočíselného smíšeného programování,
- $GAP = \frac{MIPBB}{Z} 100\ \%$ - relativní mezera mezi aktuálním a nejlepším možným řešením.

Prázdná políčka v tabulkách indikují hodnotu 0.
Hodnota „x“ označuje, že nebylo nalezeno žádné řešení.

Podívejme se na porovnání všech skriptů z kapitoly 4.1 pro $m = 1$, $L = n$ a $K = n - 1$ v tabulce 4.1. Poměrně špatně si vedly modely TSP-MTZ a VRP, protože výpočetní čas úloh byl nejdelší. Zdaleka nejlepšími se ukázaly modely MTSP-KaBe a VRP-KaBe, které prokázaly časovou efektivitu i pro úlohu $n = 68$.

Tabulka 4.1: Porovnání modelů pro 1 obchodního cestujícího.

n	TSP-MTZ		MTSP		VRP		MTSP-KaBe		VRP-KaBe	
	t	Z (km)	t	Z (km)	t	Z (km)	t	Z (km)	t	Z (km)
8	0,28s	107,525	0,01s	107,525	0,01s	107,525	0,01s	107,525	0,06s	107,525
15	0,25s	97,448	0,18s	97,448	0,21s	97,448	0,11s	97,448	0,26s	97,448
18	0,66s	121,770	0,41s	121,770	0,31s	121,770	0,33s	121,770	0,37s	121,770
20	2,12s	145,166	1,21s	145,166	0,80s	145,166	0,28s	145,166	0,33s	145,166
25	0,37s	123,449	0,38s	123,449	0,19s	123,449	0,02s	123,449	0,09s	123,449
30	7,78s	187,358	18,52s	187,358	11,17s	187,358	3,69s	187,358	4,46s	187,358
32	14,43s	133,868	12,53s	133,868	12,85s	133,868	6,78s	133,868	5,03s	133,868
34	5,25s	138,444	3,78s	138,444	6,68s	138,444	1,06s	138,444	0,95s	138,444
38	11,39s	221,321	7,38s	221,321	7,21s	221,321	3,85s	221,321	4,28s	221,321
40	9,61s	146,502	7,48s	146,502	6,12s	146,502	1,21s	146,502	1,55s	146,502
46	13m 9s	250,872	28,39s	250,872	26,32s	250,872	18,09s	250,872	6,01s	250,872
49	50m 30s	279,679	14,32s	279,679	19,41s	279,679	6,38s	279,679	5,34s	279,679
53	16m 38s	230,984	3m 39s	230,984	54,70s	230,984	13,99s	230,984	15,35s	230,984
68	3h*	431,871	55m 27s	431,871	1h 43m 41s	431,871	1m 38s	431,871	44,04s	431,871

Symbol * indikuje nenalezení optimálního řešení s jistotou. Hodnota GAP pro tento výpočet skončila na 0,62%.

Pro následující výpočty využíváme omezení řešitele GUROBI na 1 hodinu výpočtu. V tabulce 4.2 vidíme porovnání modelů MTSP a VRP pro různé hodnoty n, m i L. V těchto problémech dominuje časově i hodnotou účelové funkce model MTSP oproti VRP. Model VRP proto již nebudeme nadále využívat.

Tabulka 4.2: Porovnání modelů MTSP a VRP.

n	m	L	MTSP			VRP		
			t	Z (km)	GAP (%)	t	Z (km)	GAP (%)
15	2	10	0,40s	103,656		1,11s	103,656	
	3	8	0,47s	107,825		0,42s	107,825	
	3	5	0,97s	111,932		2,51s	111,932	
	4	8	0,21s	116,099		0,32s	116,099	
	4	4	1,84s	121,758		8,12s	121,758	
	5	5	0,19s	122,377		0,50s	122,377	
	5	3	2,00s	131,178		5m 8s	131,178	
18	2	9	1,60s	133,543		4,36s	133,543	
	3	6	7,65s	155,448		47,63s	155,448	
	4	10	0,51s	142,716		3,94s	142,716	
	4	5	37,50s	174,790		7m 28s	174,790	
	5	4	2m 41s	195,995		43m 42s	195,995	
	6	3	19,87s	220,925		41m 55s	220,925	
20	2	13	1m 23s	160,780		17m 13s	160,780	0,25
	2	10	4m 45s	164,814		24m 10s	164,814	0,01
	3	10	9m 58s	168,417		43m 34s	168,417	0,22
	3	7	1h	187,351	4,72	1h	187,351	11,50
	4	8	6m 16s	179,723		40m 32s	179,723	0,08
	4	5	1h	207,157	9,70	1h	213,558	18,90
	5	6	6m 26s	195,727	0,20	59m 21s	195,727	0,06
	5	4	1h	223,480	6,41	1h	228,576	22,30
25	2	15	15m 8s	145,315	0,01	28m 3s	145,315	0,01
	3	12	20m 37s	162,693	0,01	1h	162,693	9,71
	3	9	1h	176,401	9,08	1h	187,432	23,06
	4	7	1h	194,259	11,40	1h	213,519	31,55
	5	10	8m 12s	172,966	0,01	1h	173,396	9,53
	5	5	1h	235,154	21,80	1h	x	x
30	3	10	29m 42s	201,968	0,01	1h	202,585	3,20
	4	10	3m 15s	203,411	0,01	47m 46s	203,411	0,01
	5	7	2m 34s	212,797	0,01	1h	213,981	2,74
	5	6	32m 50s	220,785	0,01	1h	x	x

Opět omezujeme řešitele GUROBI na 1 hodinu. V tabulce 4.3 vidíme přehled modelů MTSP-KaBe a VRP-KaBe. Zcela jasně dominuje model MTSP-KaBe. Dokázal vyřešit téměř všechny problémy do optimality v rámci časového omezení, kdežto model VRP-KaBe nikoliv.

Tabulka 4.3: Porovnání modelů MTSP-KaBe a VRP-KaBe.

n	m	L	K	MTSP-KaBe			VRP-KaBe		
				t	Z (km)	GAP (%)	t	Z (km)	GAP (%)
15	3	12	3	0,30s	111,659		0,70s	111,659	
	3	12	4	0,31s	111,932		1,87s	111,932	
	4	8	3	0,23s	121,487		2,39s	121,487	
	4	4	3	0,39s	121,758		1,68s	121,758	
18	3	10	5	1,30s	155,448		14,42s	155,448	
	3	8	4	1,66s	154,524		10,79s	154,524	
	3	8	5	2,00s	155,448		5,38s	155,448	
	4	12	4	2,74s	179,935		24,95	179,935	
	4	5	4	1,48s	179,935		3m 30s	179,935	
	5	4	2	1,00s	195,995		56,13s	195,995	
20	2	12	8	0,74s	161,387		6,90s	161,387	0,01
	3	12	6	15,60s	187,894		5m 35s	187,894	
	3	9	4	15,57s	181,932	0,01	5m 52s	181,932	0,01
	3	9	6	57,43s	187,894		25m 36s	187,894	0,01
	3	7	6	16,38s	187,894		44m 41s	187,894	0,01
	4	5	4	1m	207,102		3m 56s	207,102	
	5	4	2	16,64s	223,480		11m 5s	223,480	
25	2	20	8	10,80s	139,663		1m 20s	139,663	
	2	20	12	51m 15s	160,254	0,01	1h	160,254	5,93
	3	16	6	36m 37s	169,987		1h	169,987	8,14
	3	12	2	19m 23s	164,750	0,01	1h	164,835	11,23
	4	14	2	23,27s	155,699		2m 37s	155,699	
	4	14	4	4m 20s	179,790		1h	179,790	11,12
	5	15	2	0,58s	158,610		4,18	158,610	
	5	5	4	24m 41s	223,261		1h	234,303	25,56
30	3	10	9	36,9s	201,968		17m 48s	201,968	0,01
	4	20	6	4m 10s	204,860	0,01	59m 20s	204,860	0,01
	5	20	5	1h	216,062	1,99	1h	216,062	3,56
	6	20	2	2,74s	212,018		16m 13s	212,018	0,01

Nyní porovnáme modely MTSP a MTSP-KaBe, protože se prokázaly býti lepšími. Používáme $K = 2$ pro co nejvěrohodnější srovnání modelů a čas pro výpočet 1 hodinu. Při porovnání těchto modelů v tabulce 4.4 vidíme, že MTSP-KaBe je rychlejší a nachází lepší hodnoty účelové funkce než MTSP. Označujeme tedy model MTSP-KaBe jako efektivnější.

Musíme však podotknout, že model MTSP-KaBe neuvažuje navštívení pouze jednoho města obchodním cestujícím, tedy $K = 1$, kdežto model MTSP tuto variantu implicitně obsahuje. Proto v některých případech dosahuje model MTSP nižší účelové funkce. Pro hledání těchto variant tedy upřednostníme model MTSP, protože MTSP-KaBe je neumožňuje.

4.3 Zobrazení cest

Podívejme se, jak si řešitel GUROBI poradí se souborem o velikosti $n = 156$. Nařídíme MPL, aby zaznamenal do Excelu hodnoty x_{ij}, případně x_{ijk}, a vykreslíme pomocí statistického prostředí R do mapy. V modelech z kapitoly 4.1 pro úlohu typu MTSP provedeme výpočty pro

$$m = 4, \quad L = 50, \quad K = 2,$$

kde K se pochopitelně používá u verze podmínek KaBe.

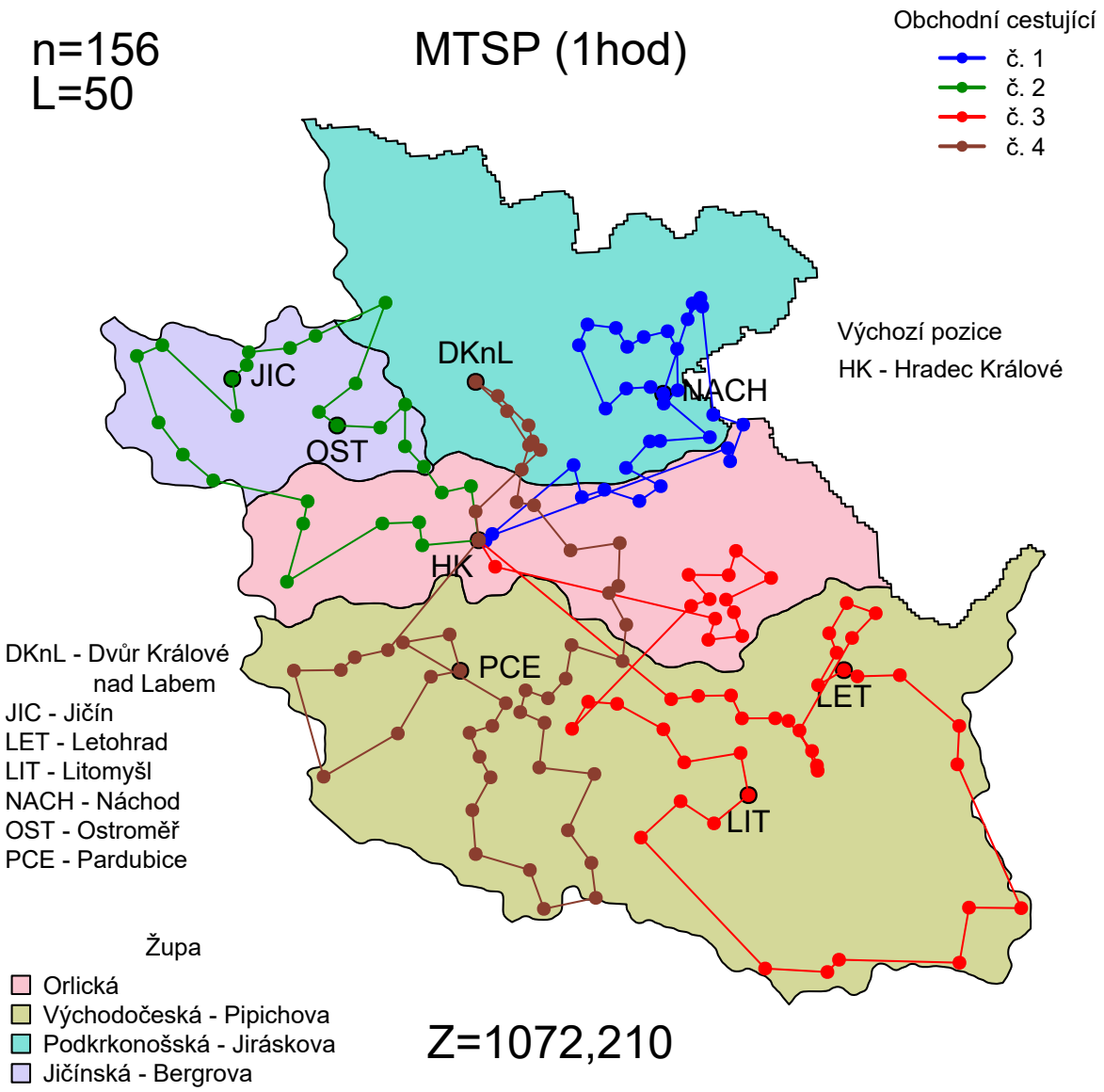
Pro největší soubor z listu KH_PCE nastaly však problémy s načítáním dat. Bylo tedy potřeba změnit načítání indexů `i`, `j` a `p`. Místo načítání textu pro pojmenování měst z Excelu jsme použili číselné ohodnocení `i:=1..156;` a stejně i pro `j` a `p`. Načítání dat do MPL je zřejmě omezeno, a jelikož některé názvy jednot byly příliš dlouhé, zaplňovaly paměť a neumožňovaly načtení všech dat.

Výpočty limitujeme časovým omezením 1, 8, 16 a 24 hodin. Obrázky 4.1, 4.2, 4.3 a 4.4 zachycují mapy přesunů obchodních cestujících pro model MTSP. Pohyby obchodních cestujících navržených metodou MTSP-KaBe jsou znázorněny na obrázcích 4.5, 4.6, 4.7 a 4.8.

Modely MTSP-VRP a MTSP-VRP-KaBe jsou pomalejší a pro tento soubor nedokázaly najít žádné přípustné řešení do časového limitu jedné hodiny. Jelikož se jedná o modely, pro které řešitel GUROBI špatně hledá řešení, vynecháme je z této zkoušky a doporučujeme se jim pro takto rozsáhlé problémy vyhýbat.

Tabulka 4.4: Porovnání modelů MTSP a MTSP-KaBe.

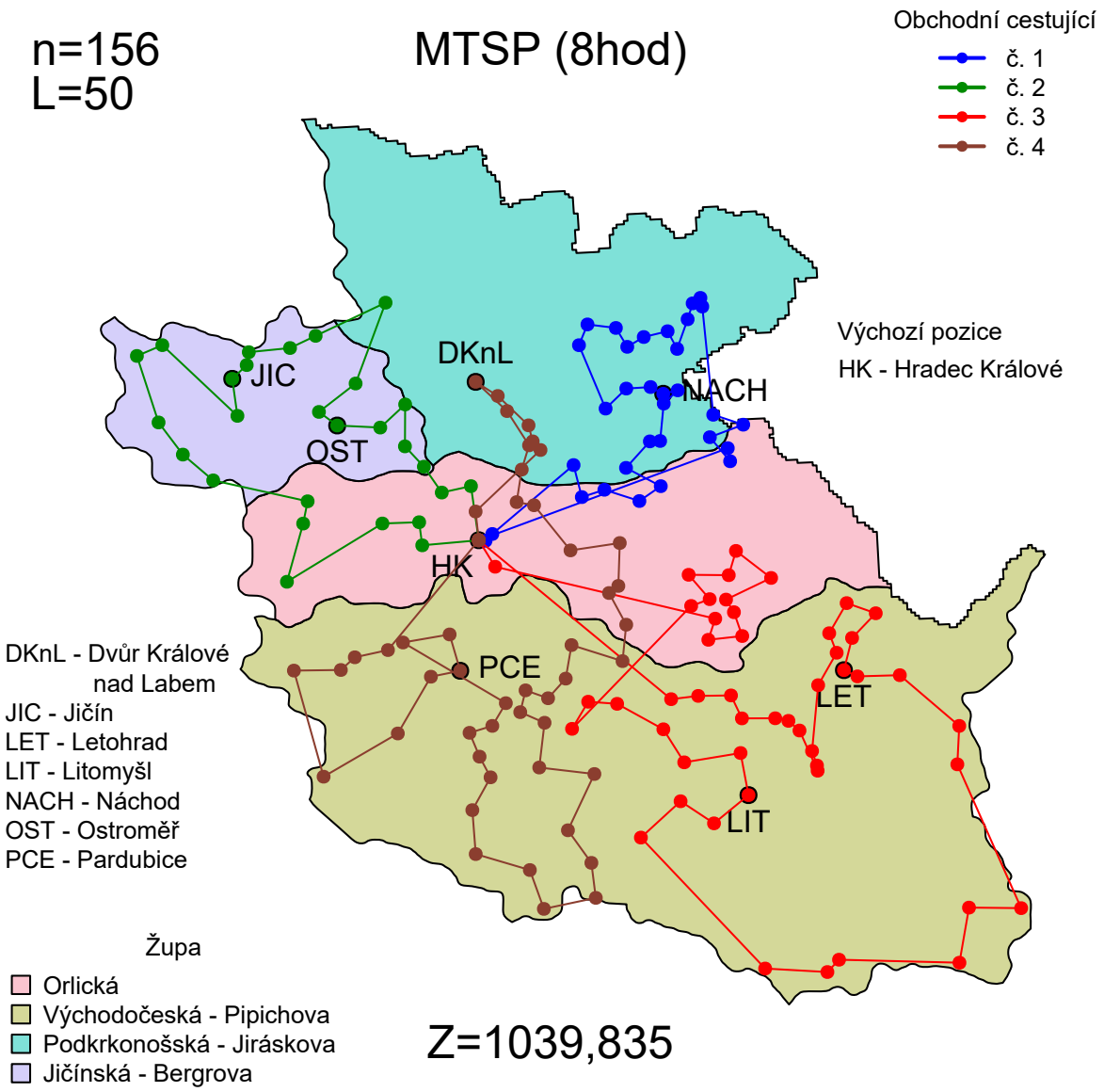
n	m	L	MTSP			MTSP-KaBe		
			t	Z (km)	GAP (%)	t	Z (km)	GAP (%)
30	3	10	35m 4s	201,968	0,01	43,27s	201,968	
	4	8	13m 8s	207,911	0,01	32,01s	207,911	
	5	10	38,05s	207,019		7,96s	209,745	
	6	5	1h	241,220	1,84	45,36s	241,220	
32	2	20	1h	144,013	0,87	12m 40s	144,013	0,01
	3	11	1h	165,286	12,37	1h	163,495	4,87
	4	16	10m 53s	149,061	0,01	1m 10s	157,149	
	4	8	1h	190,006	23,10	1h	186,158	11,01
34	2	20	4m 40s	148,925	0,01	1m 16s	148,925	0,01
	2	17	16m 31s	150,966	0,01	2m 5s	150,966	
	3	20	3m 53s	150,926		35,74s	150,926	
	3	12	1h	168,092	8,84	1h	168,021	5,29
38	2	25	28,99s	227,030		29,58s	227,030	
	2	19	1h	258,263	12,29	1h	256,490	11,45
	3	15	1h	274,168	16,79	1h	268,110	12,22
	4	10	1h	317,589	25,03	1h	313,954	21,55
40	2	30	10,02s	147,752		1,48s	147,752	
	2	20	3m 21s	149,400		5,91s	149,400	
	3	15	1h	159,568	4,24	1h	159,568	2,46
	4	10	1h	183,860	15,80	1h	183,466	14,34
46	2	40	1m 25s	267,936	0,01	6,51s	267,936	
	2	23	1h	278,528	3,13	37m 17s	278,528	0,01
	3	20	1h	307,620	7,01	1h	307,050	2,64
	4	12	1h	367,712	15,47	1h	359,528	8,98
49	2	25	2m 14s	291,176		11,48s	291,176	
	3	17	1h	310,959	1,72	4m 21s	310,578	
	4	13	1h	344,525	7,66	4m 44s	330,077	
	5	10	1h	362,696	7,83	1h	359,038	3,53
53	2	40	1h	242,537	2,85	1h	242,170	1,41
	3	20	1h	301,671	21,09	1h	270,691	9,28
	4	15	1h	350,588	30,13	1h	304,013	16,61
	5	15	1h	330,635	23,64	1h	303,612	13,94
68	2	35	1h	492,031	12,30	1h	463,943	5,44
	3	25	1h	562,944	21,87	1h	525,149	14,16
	4	20	1h	584,667	22,47	1h	539,450	13,39
	5	15	1h	658,964	28,85	1h	619,311	21,41



Obrázek 4.1: Nalezené řešení pro jednoty z Královéhradeckého a Pardubického kraje metodou MTSP pro čtyři obchodní cestující po 1 hodině.

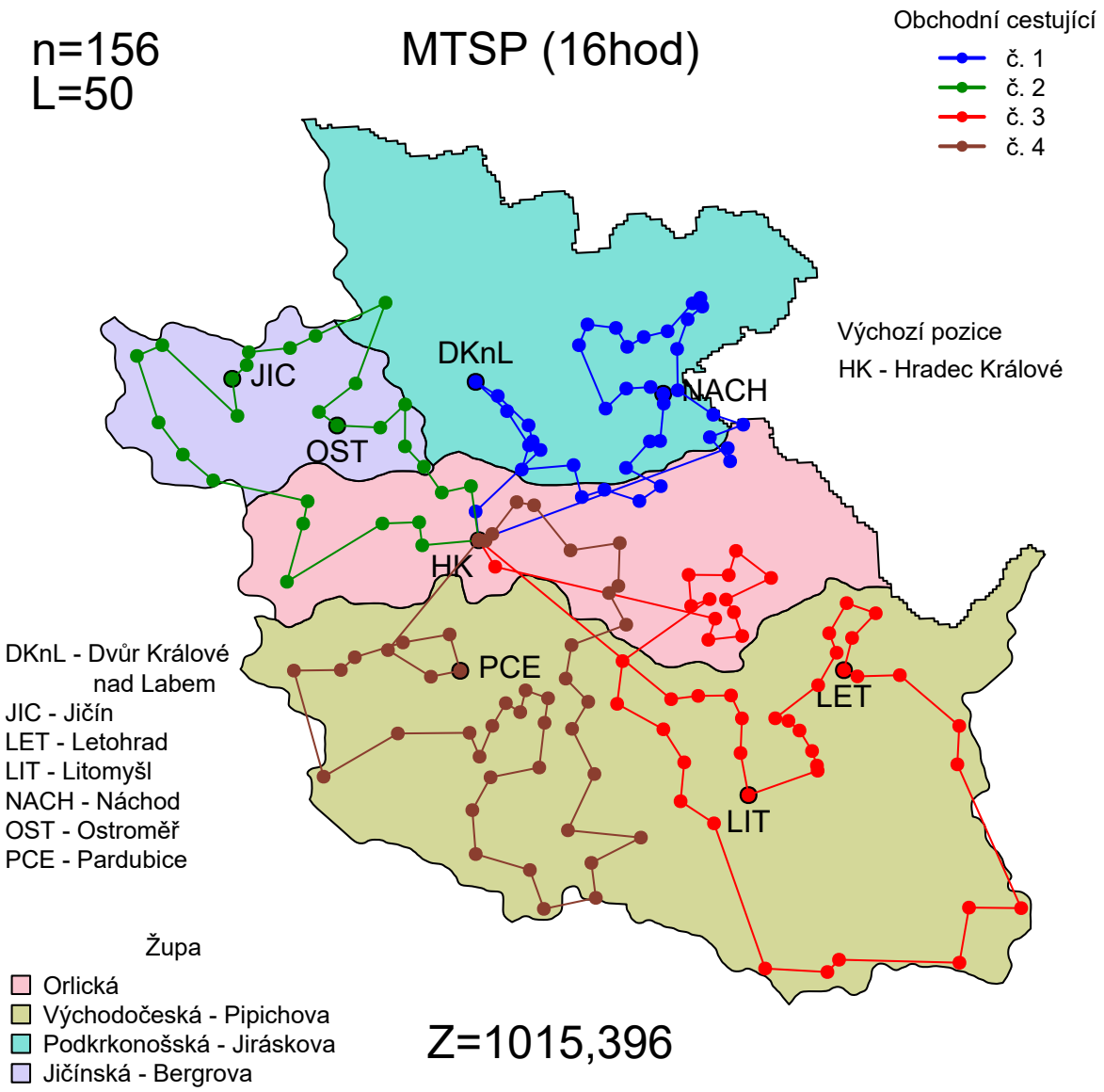
Při porovnání s metodou nejbližšího souseda, kterou jsme prezentovali v části 2.4.1 obrázkem 2.1 vidíme zlepšení o více než 150 kilometrů.

Vidíme, že u Náchoda (modrá) dochází k spletitým cestám. Dále si můžeme povšimnout, že obchodní cestující č. 4 (hnědá) cestuje nejen na jih od Hradce, ale udělá si i zajížďku na sever do Dvora Králové nad Labem. Na několika místech také dochází k překřížení cest, což může indikovat neoptimalitu.



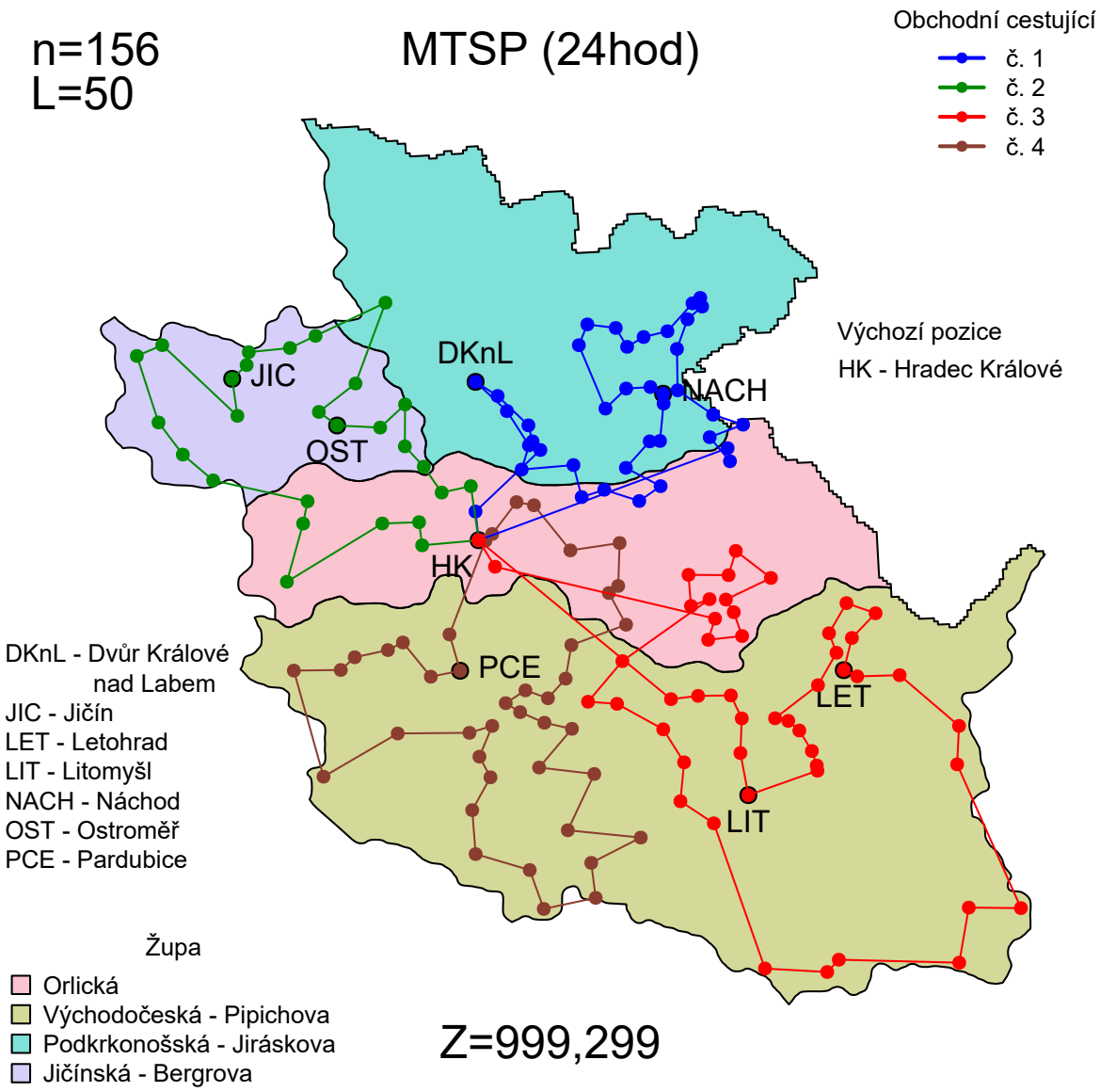
Obrázek 4.2: Nalezené řešení pro jednoty z Královéhradeckého a Pardubického kraje metodou MTSP pro čtyři obchodní cestující po 8 hodinách.

Oproti předchozímu obrázku pozorujeme zlepšení v oblasti okolo Náchoda, bylo dosaženo přibližně 30kilometrového zkrácení celkové délky tras. Nicméně stále vidíme potenciál pro zlepšení.



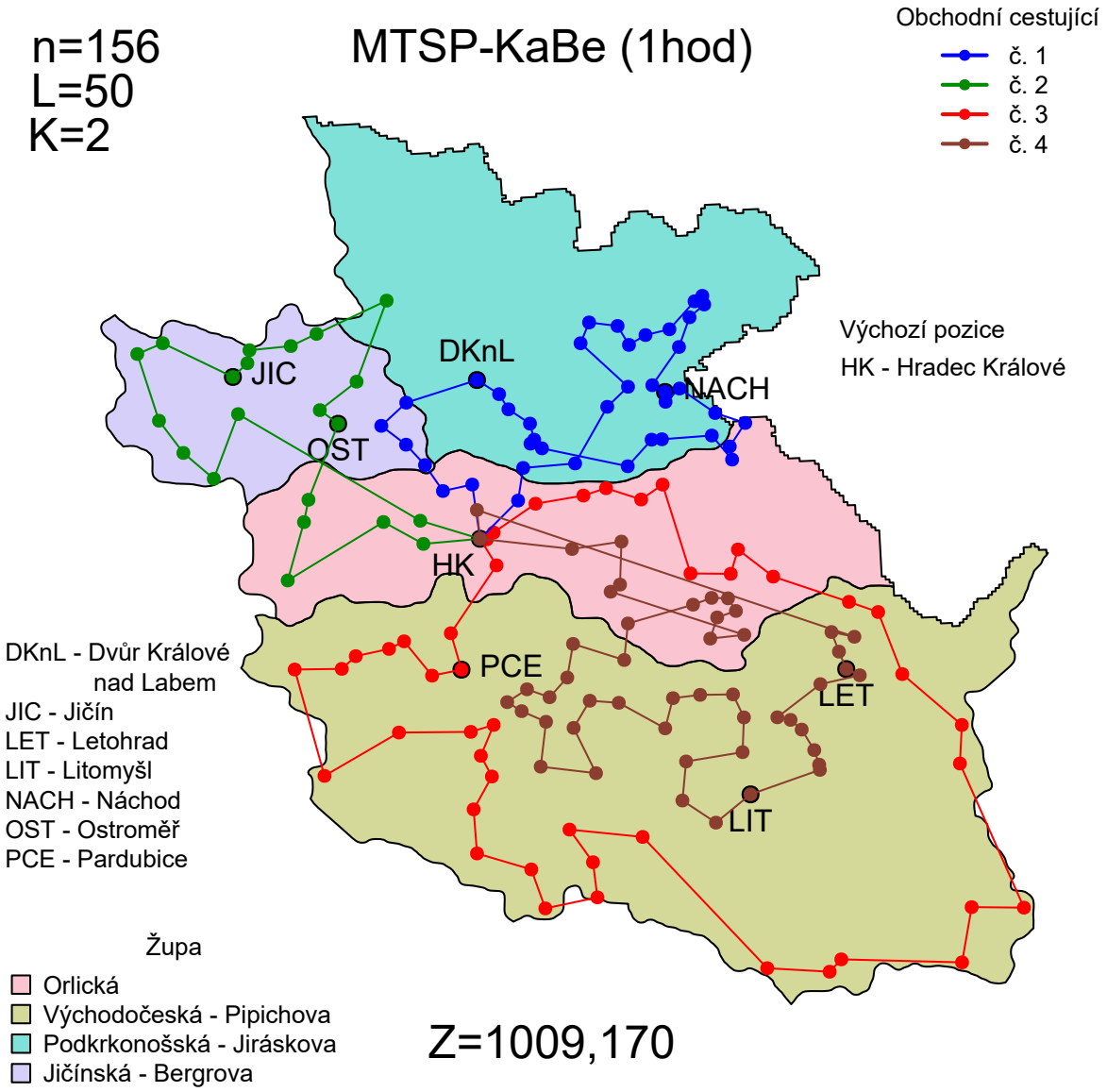
Obrázek 4.3: Nalezené řešení pro jednoty z Královéhradeckého a Pardubického kraje metodou MTSP pro čtyři obchodní cestující po 16 hodinách.

Po dalších osmi hodinách se podařilo dosavadní řešení vylepšit přibližně o 25 kilometrů. Vidíme, že jednoty po cestě z Hradce Králové do Dvora Králové nad Labem nyní převzal obchodní cestující č. 1 (modrá) od obchodního cestujícího č. 4 (hnědá).



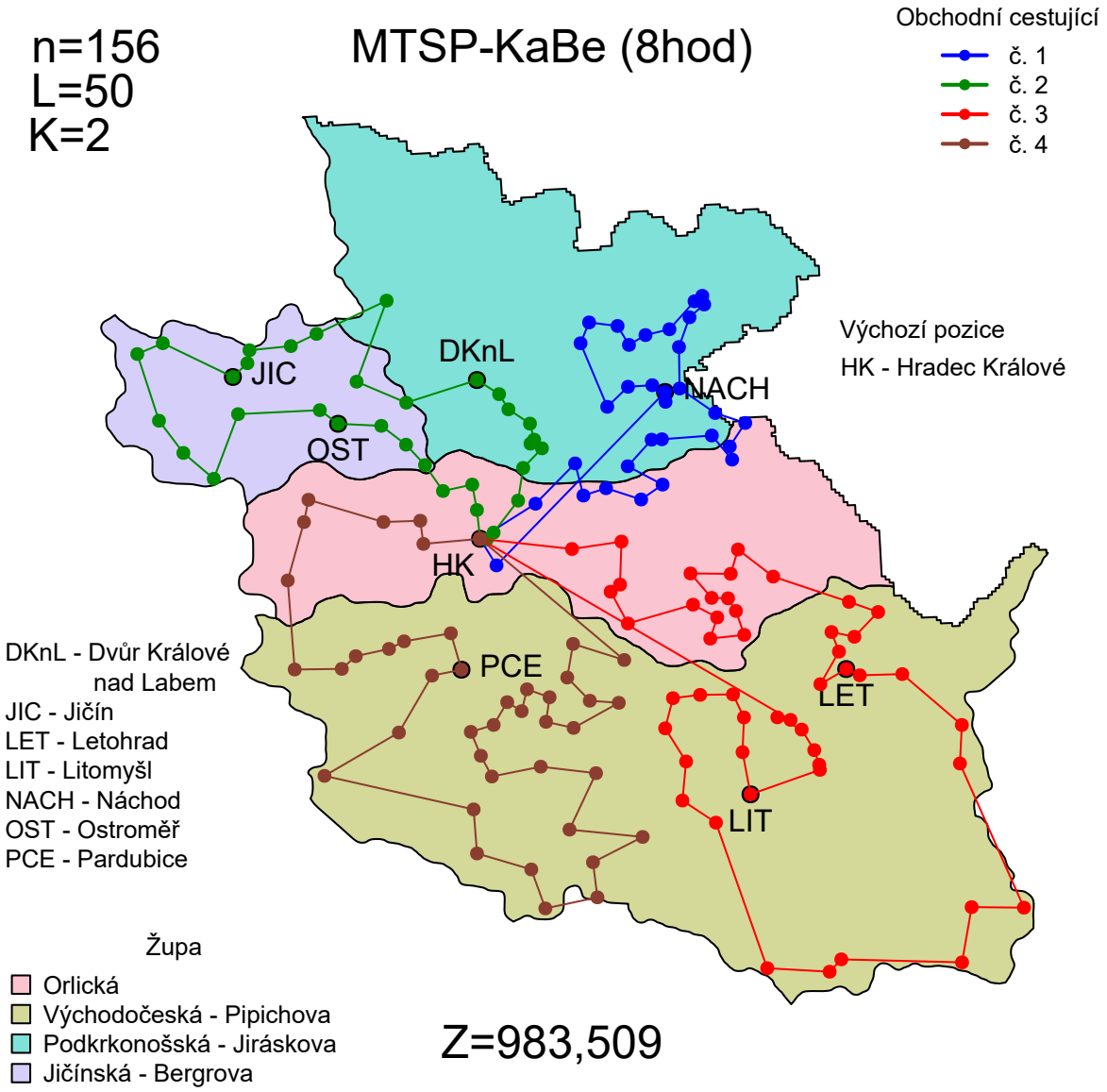
Obrázek 4.4: Nalezené řešení pro jednoty z Královéhradeckého a Pardubického kraje metodou MTSP pro čtyři obchodní cestující po 24 hodinách.

Po 24 hodinách výpočtu, se podařilo dostat pod hranici 1000 ujetých kilometrů. Zlepšení spočívá v západní části župy Východočeské - Pipichovy a na východ od Pardubic si obchodní cestující č. 3 a č.4 vyměnily pár jednot, zatímco ostatní trasy se zdají být oproti předchozí 16hodinové variantě (obrázek 4.3) nedotčené. Nevidíme zde již tolik prostoru pro zlepšení jako v předchozích případech s výjimkou obchodního cestujícího č. 3 na území župy Orlické.



Obrázek 4.5: Nalezené řešení pro jednoty z Královéhradeckého a Pardubického kraje metodou MTSP-KaBe pro čtyři obchodní cestující po 1 hodině.

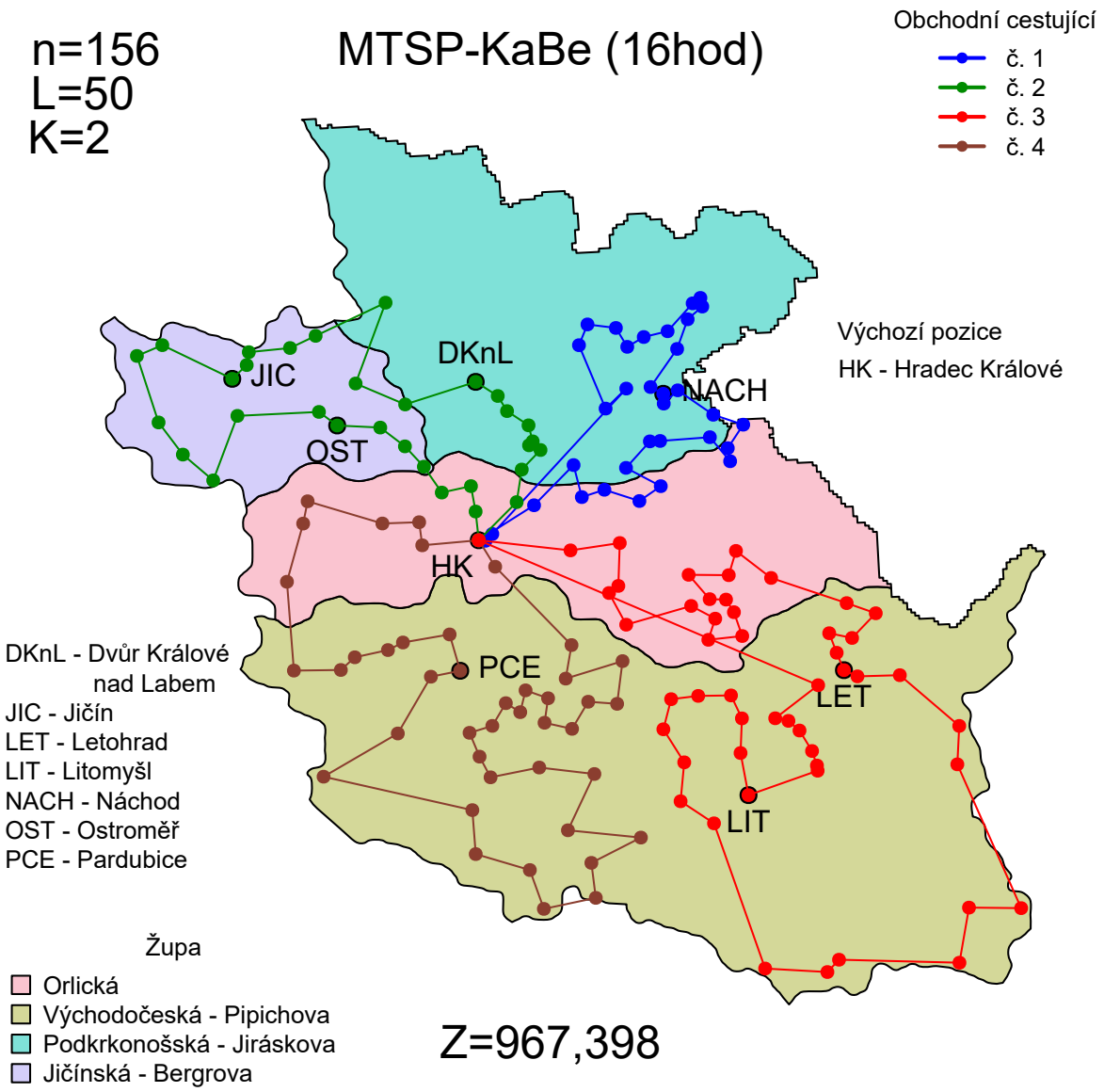
Metoda MTSP-KaBe po 1 hodině výpočtů začíná s poněkud odlišným řešením než předchozí metoda MTSP. Zásadní rozdíl spočívá v obsluze oblasti župy Východočeské - Pipichovy, kde obchodní cestující č. 3 (červená) projíždí jednoty na její hranici, zatímco obchodní cestující č. 4 (hnědá) obstarává její centrum poměrně efektivně, avšak na území župy Orlické si několikrát kříží trasu. Prostor pro zlepšení dále vidíme u obchodních cestujících č. 1 (modrá) a č. 2 (zelená), kteří si také kříží vlastní trasu. Toto řešení je však z hlediska celkové ujeté vzdálenosti mnohem lepší než to, s kterým po hodině přišla metoda MTSP (obrázek 4.1).



Obrázek 4.6: Nalezené řešení pro jednoty z Královéhradeckého a Pardubického kraje metodou MTSP-KaBe pro čtyři obchodní cestující po 8 hodinách.

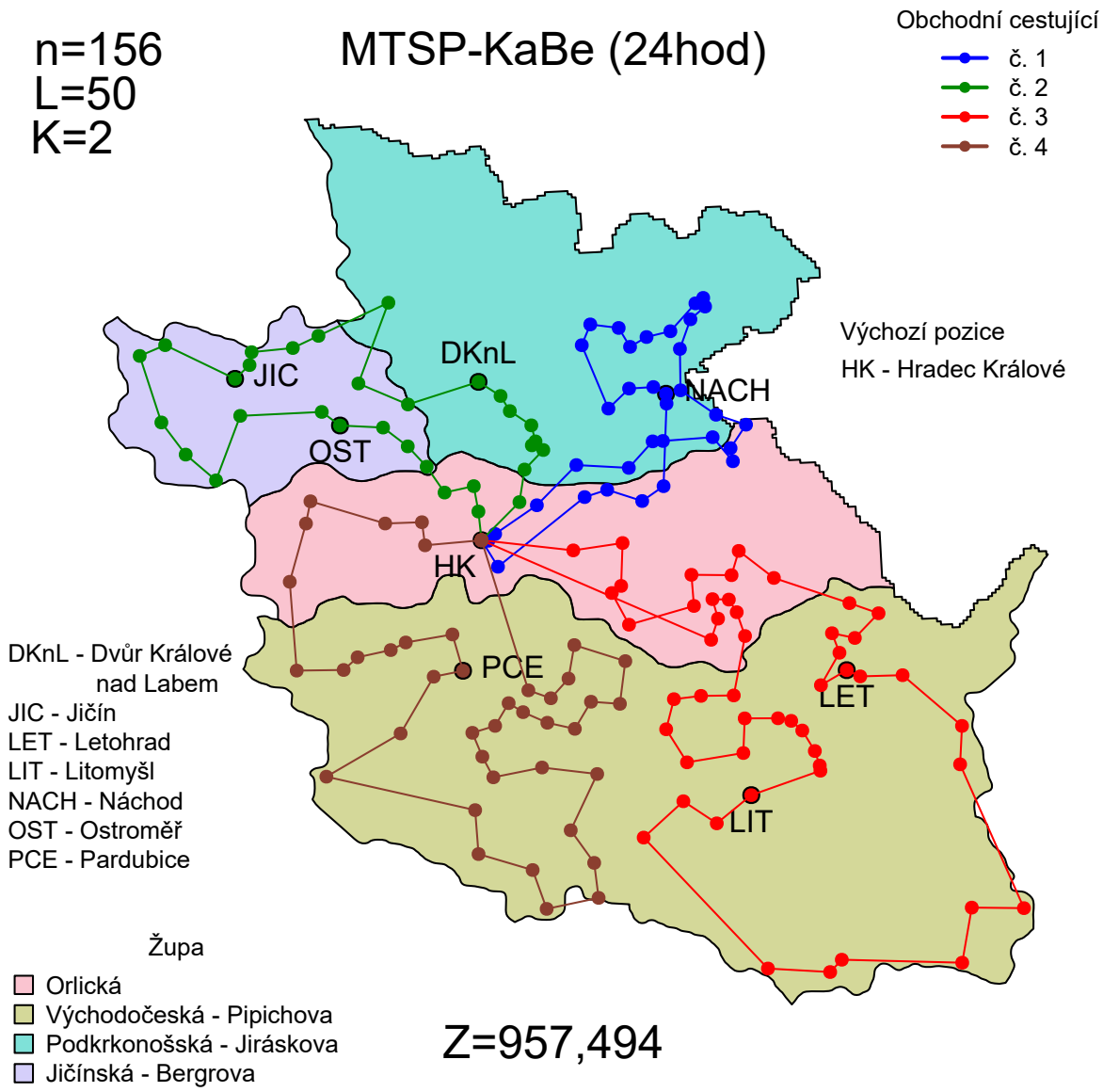
Je zřejmé, že i MTSP-KaBe dospěl k závěru, že přeci jen bude efektivnější neprovádět okružní cestu po hranici župy Východočeské - Pipichovy. Obchodní cestující č. 3 (červená) a č. 4 (hnědá) si rozdělili východní a západní část této župy. Nejen, že jsme se oproti předchozímu řešení zlepši přibližně o 25 kilometrů, ale dokonce jsme pokořili 24hodinový výpočet metodou MTSP ($Z = 999{,}299$ km).

Inovací oproti předchozím řešením je, že obchodní cestující č. 4 (hnědá) převzal jednoty na západě župy Orlické, které do té doby obvykle obsluhoval obchodní cestující č. 2 (zelená). Ten místo toho vyráží z Hradce Králové směr Ostroměř a Jičín a vrací se zpět přes Dvůr Králové nad Labem. Obchodní cestující č. 1 (modrá) využívá jednot, které předtím obsluhoval obchodní cestující č. 3 (červená). Možnost pro zlepšení vidíme v křížení vlastních cest u obchodních cestujících č. 1 (modrá) a č. 3 (červená).



Obrázek 4.7: Nalezené řešení pro jednoty z Královéhradeckého a Pardubického kraje metodou MTSP-KaBe pro čtyři obchodní cestující po 16 hodinách.

Po dalších osmi hodinách výpočtů se podařilo předchozí už tak celkem obstojné řešení ještě vylepšit, a to zhruba o 15 kilometrů. Největší změnu pozorujeme u obchodního cestujícího č. 1 (modrá), který značně optimalizoval svoji cestu v okolí Náchoda. Obchodní cestující č. 2 (zelená) si ponechává svoji dosavadní trasu nezměněnou. Obchodní cestující č. 3 (červená) a č. 4 (hnědá) provedli téměř nepatrné úpravy.



Obrázek 4.8: Nalezené řešení pro jednoty z Královéhradeckého a Pardubického kraje metodou MTSP-KaBe pro čtyři obchodní cestující po 24 hodinách.

Tento obrázek poskytuje nejlepší nalezené řešení pro tuto sadu jednot v Královéhradeckém a Pardubickém kraji. Čtyři obchodní cestující dokáží objet všech 156 jednot v celkové trase 957,494 kilometrů. Oproti předchozímu řešení jsme dosáhli drobného zlepšení, už je však těžké určit, ve kterých místech skutečně došlo k snížení účelové funkce.

Vidíme, že došlo k nepatrným prohozením jednot mezi obchodními cestujícími. Pořadí průchodů jednot prošlo také několika změnami. Pro obchodního cestujícího č. 2 (zelená) nedošlo k žádným změnám. V této oblasti se zřejmě jedná o optimální trasu.

4.4 Zhodnocení

Model TSP s MTZ-SEC zaostává za ostatními modely v řešení úloh typu TSP. Významný časový skok byl zaznamenán při řešení úloh o velikosti n mezi 40–50 městy. Ovšem při dostatečném množství času dokáže také nalézt optimální řešení jako ostatní metody. V řešení této úlohy se prokázaly jako nejlepší modely MTSP-KaBe a VRP-KaBe.

Modely MTSP-VRP a MTSP-VRP-KaBe pro tří-indexovou proměnou nejsou časově efektivní. V těchto modelech pro MTSP obsluhujeme m-krát více proměnných, než je nutné, což způsobuje zpomalení výpočtů.

Modely MTSP a MTSP-KaBe se prokázaly lepší než jejich protějšky modelů VRP. Zbývá rozhodnout, který z těchto dvou modelů je efektivnější. V testech mezních hodnot ($n \approx mL$) se MTSP-KaBe osvědčil lépe, protože dokázal tyto problémy řešit rychleji. Zřejmě za to může ořezání možných řešení v podobě podmínek dolních mezí. Tento rozdíl lze pozorovat pro všechna porovnávaná měření.

Pokud však vezmeme v potaz úlohy, pro které uvažujeme $K = 2$ a libovolné hodnoty L, pak po hlubším prostudování zjistíme, že MTSP-KaBe a MTSP vykazují znatelné rozdíly v hodnotě účelové funkce Z. To je způsobeno agresivním ořízntutím dolních mezí podmínkou KaBe, která neumožňuje návštěvu jednoho města a ihned návrat do startovního města. Je tedy potřeba dávat pozor na takovéto hodnoty. Model MTSP-KaBe tedy nedokáže řešit úlohy, kde může být optimální použít cesty o nejvýše jedné zastávce.

Obecně však začíná model MTSP-KaBe celkově dominovat (jak ve výpočetním čase, tak v optimalitě nalezeného řešení) nad modelem MTSP od $n = 30$ a pro libovolný přiměřený (vzhledem k n) počet obchodních cestujících. MTSP-Kabe dominuje nad MTSP ve všech složitějších problémech, kromě zmíněného problému s $K < 2$, pro který nedokáže najít nižší hodnotu účelové funkce.

Jak vidíme z předchozích obrázků, metoda MTSP-KaBe opět dosahuje lepších výsledků než MTSP. MTSP-KaBe překonal 24hodinový výpočet metody MTSP již po necelých osmi hodinách. Nicméně ani jeden z těchto modelů nebyl schopen najít optimum po 24 hodinách výpočtu, jak vidíme v tabulce 4.5.

Tabulka 4.5: Porovnání modelů MTSP a MTSP-KaBe pro $n = 156$.

m	L	MTSP			MTSP-KaBe		
		t	Z (km)	GAP (%)	t	Z (km)	GAP (%)
4	50	1h	1072,210	20,63	1h	1009,170	13,52
4	50	8h	1039,835	17,89	8h	983,509	11,37
4	50	16h	1015,396	15,75	16h	967,398	9,79
4	50	24h	999,299	14,40	24h	957,494	8,77

Závěr

Práci jsme zahájili představením problému obchodního cestujícího a vytvořili historický přehled řešení úloh s touto tematikou. Na základech teorie grafů jsme postavili formulaci modelu TSP, který jsme posléze rozšířili na vícestupňový problém obchodního cestujícího, a představili si čtyři modely pro tento typ úloh. Představili jsme také několik variant řešení pomocí heuristik a metahauristik.

Zkompletovali jsme soubor souřadnic GPS 476 sokolských jednot ze 14 žup. Pomocí Haversiánova vzorce jsme dopočítali vzájemné vzdálenosti těchto jednot. Implementovali jsme všechny předložené modely do MPL for Windows, které jsme aplikovali na jednotlivých župách a řešili pomocí nástroje GUROBI.

Model TSP s MTZ-SEC zaostává za ostatními modely v řešení úloh typu TSP. Není ani divu, když se jedná o jeden z nejstarších modelů. Významný časový skok byl zaznamenán při řešení úloh o velikosti n mezi 40–50 městy. V porovnání s ostatními už přestává být časově efektivní tento model používat. V případě většího množství měst tedy doporučujeme použít některý z dalších navrhovaných modelů.

Modely MTSP-VRP a MTSP-VRP-KaBe pro tří-indexovou proměnnou x_{ijk} se časově příliš neosvědčily. V těchto úlohách totiž uvažujeme nejméně m-krát větší množství proměnných, což vyžaduje větší množství času. Ačkoliv se může tento zápis zdát vhodnější pro přiřazení cest jednotlivým obchodním cestujícím, dvou-indexový zápis pro naše potřeby poskytuje stejné množství informací. Nicméně v této práci uvádíme podmínky KaBe i pro tří-indexovou proměnnou, ty v dnešní literatuře nebyly dosud k dispozici nebo k nalezení.

Slibné modely MTSP a MTSP-KaBe jsme dále podrobili zevrubnějším testům. MTSP-KaBe se ve většině případů ukázal rychlejší pro testy mezních hodnot, tedy v případech, kdy každý obchodní cestující musí použít stejné množství cest, případně o jedinou méně, proto jej v tomto případě upřednostňujeme. Co se však týče řešení úloh, které jsou dány pouze horní mezí, docházíme k zajímavému závěru. Varianta MTSP-KaBe nedokáže hledat řešení, kde by obchodní cestující mohl cestovat pouze jedním městem a hned se vrátil do výchozího města. Pro hledání takovýchto variant tedy MTSP-KaBe nedoručujeme, protože k tomu nebyl navržen. Obecně metoda MTSP-KaBe začíná nad metodou MTSP dominovat přibližně od $n = 30$ a pro libovolný přiměřený počet obchodních cestujících.

Na vzorku s nejvyšším počtem sokolských jednot $n = 156$ jsme porovnávali výsledné trasy čtyř obchodních cestujících na mapě. Oba dva modely MTSP a MTSP-KaBe jsme řešili po dobu 24 hodin a porovnávali průběžná optimální řešení. Ilustrovali jsme tak, jakými způsoby dochází k zlepšování již nalezeného řešení. Zároveň jsme ukázali, že MTSP-KaBe dokázal v třetinovém čase předčít nejlepší řešení nalezené klasickým MTSP. Optimálního řešení jsme bohužel ani po 24 hodinovém výpočtu nedosáhli. Modely VRP a VRP-KaBe nebyly schopny nalézt přípustné řešení v limitu jedné hodiny výpočtu.

Pro případné pokračování této práce by bylo vhodné vyzkoušet další metaheuristiky a porovnat jejich efektivitu jak už z hlediska výpočetního času, tak i schopnosti nalézt optimální řešení, zvláště pak v případě vysokého počtu měst n. Na základě této práce doporučujeme srovnávat výsledná řešení s preferovaným MTSP-KaBe.

Seznam použité literatury

ALEXANDERSON, Gerald, 2006.
About the cover: Euler and Königsberg's Bridges: A historical view.
Bulletin of the american mathematical society. Roč. 43, č. 4, s. 567–573.
Dostupné také z: https://www.ams.org/journals/bull/2006-43-04/S0273-0979-06-01130-X/S0273-0979-06-01130-X.pdf.

APPLEGATE, David L.; BIXBY, Robert E.; CHVÁTAL, Vašek; COOK, William J., 2006.
The Traveling Salesman Problem: A Computational Study.
New Jersey: Princeton university press.
Dostupné také z: https://www.ceas3.uc.edu/ret/archive/2017/ret/docs/readings/Project%203/Project%203_Introduction%20to%20TSP.pdf.

BALAS, Egon; TOTH, Paolo, 1983.
Branch and bound methods for the traveling salesman problem.
Dostupné také z: https://apps.dtic.mil/dtic/tr/fulltext/u2/a126957.pdf.
Technická zpráva.
Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group.

BEKTAS, Tolga, 2006. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*. Roč. 34, č. 3, s. 209–219.
Dostupné také z: https://www.academia.edu/23697129/The_multiple_traveling_salesman_problem_an_overview_of_formulations_and_solution_procedures.

BONYADI, Mohammad R.; AZGHADI, Mostafa R.; SHAH-HOSSEINI, Hamed, 2008.
Population-Based Optimization Algorithms for Solving the Travelling Salesman Problem, Traveling Salesman Problem. Federico Greco (Ed.), InTech. ISBN 978-953-7619-10-7.
Dostupné také z:
https://researchonline.jcu.edu.au/45702/1/45702_Bonyadi_2008_chapter.pdf.

COOK, William J., 2020. *The Travelling Salesman Problem.*
Dostupné také z: http://www.math.uwaterloo.ca/tsp/.

ČESKÁ OBEC SOKOLSKÁ, ČOS, 2020. *Česká obec sokolská.*
Dostupné také z: https://www.sokol.eu/.

DALGETY, James, 2020. *The Icosian Game*. Dostupné také z:
https://www.puzzlemuseum.com/month/picm02/200207icosian.htm.

DANTZIG, George; FULKERSON, Ray; JOHNSON, Selmer, 1954.
Solution of a large-scale traveling-salesman problem.
Journal of the operations research society of America. Roč. 2, č. 4, s. 393–410.
Dostupné také z:
https://www.jstor.org/stable/166695?seq=18#metadata_info_tab_contents.

DEVLIN, Keith, 2005.
Problémy pro třetí tisíciletí Sedm největších nevyřešených otázek matematiky.
Praha: Dokořán. ISBN 978-953-7619-10-7.

HELSGAUN, Keld, 2020. *Keld Helsgaun*. Dostupné také z: http://akira.ruc.dk/~keld/.

HIJMANS, Robert J., 2020. *DIVA-GIS*.
Dostupné také z: http://www.diva-gis.org/gdata.

CHOPDE, Nitin R.; NICHAT, Mangesh K., 2013.
Landmark based shortest path detection by using A* and Haversine formula.
International Journal of Innovative Research in Computer and Communication Engineering. Roč. 1, č. 2, s. 298–302. Dostupné také z:
https://www.researchgate.net/publication/282314348_Landmark_based_shortest_path_detection_by_using_A_Algorithm_and_Haversine_Formula.

CHRISTOFIDES, Nicos; MINGOZZI, Aristide; TOTH, Paolo, 1981. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations.
Mathematical programming. Roč. 20, č. 1, s. 255–282.
Dostupné také z: https://link.springer.com/article/10.1007/BF01589353.

JUNJIE, Pan; DINGWEI, Wang, 2006.
An ant colony optimization algorithm for multiple travelling salesman problem. In:
First International Conference on Innovative Computing, Information and Control-Volume I (ICICIC'06). Sv. 1, s. 210–213.
Dostupné také z: https://www.researchgate.net/profile/Dingwei_Wang2/publication/220792786_An_Ant_Colony_Optimization_Algorithm_for_Multiple_Travelling_Salesman_Problem/links/55b9e4d408ae9289a0902008/An-Ant-Colony-Optimization-Algorithm-for-Multiple-Travelling-Salesman-Problem.pdf.

KARA, Imdat; BEKTAS, Tolga, 2006. Integer linear programming formulations of multiple salesman problems and its variations. *European Journal of Operational Research*.
Roč. 174, č. 3, s. 1449–1458. Dostupné také z: http://yeh.nutn.edu.tw/ncku/Math/integer%20programming/Integer%20linear%20programming%20formulations%20of%20multiple%20salesman%20problems%20and%20its%20variations.pdf.

MARTIN, Chris, 2020. *File:7 bridges.svg*.
Dostupné také z: https://commons.wikimedia.org/wiki/File:7_bridges.svg.

MAXIMAL SOFTWARE, Inc., 2018a. *GUROBI Solver Guide for MPL*.

MAXIMAL SOFTWARE, Inc., 2018b. *MPL for Windows User's Guide*.

MAXIMAL SOFTWARE, Inc., 2020. *Maximal Software*.
Dostupné také z: http://www.maximalsoftware.com/.

MILLER, Clair E.; TUCKER, Albert W.; ZEMLIN, Richard A., 1960.
Integer programming formulation of traveling salesman problems.
Journal of the ACM (JACM). Roč. 7, č. 4, s. 326–329.
Dostupné také z: https://dl.acm.org/doi/pdf/10.1145/321043.321046.

PÁVEK, Michal, 2020. *sokol.cz*.
Dostupné také z: https://www.sokol.cz/sokol/index.php?action=orgzupy.

PELIKÁN, Jan, 2001. *Diskrétní modely v operačním výzkumu*.
Praha: Professional Publishing. ISBN 80-86419-17-7.

REINHELT, Gerhard, 2014. *TSPLIB: a library of sample instances for the TSP (and related problems) from various sources and of various types*. Dostupné také z: http://comopt.%20ifi.%20uniheidelberg.%20de/software/TSPLIB95.

SEDIGHPOUR, Mohammad; YOUSEFIKHOSHBAKHT, Majid; MAHMOODI DARANI, Narges, 2012. An effective genetic algorithm for solving the multiple traveling salesman problem. *Journal of Optimization in Industrial Engineering*. Č. 8, s. 73–79. Dostupné také z: http://www.qjie.ir/article_89_fd3d6037a76015c814dd3d82969392c0.pdf.

SEZNAM.CZ, a.s., 2020. *Mapy.cz*. Dostupné také z: https://mapy.cz.

SOMMER, Christoph, 2020. *File:Hamiltonian path.svg*. Dostupné také z: https://commons.wikimedia.org/w/index.php?curid=1724516.

Přílohy

A. Ukázka skriptů

Přikládáme použité skripty v prostředí MPL pro výpočetní úkony. Před započetím výpočtů je potřeba upravit skripty následujícím způsobem.

`JMENO LISTU`
: Zadejme jméno listu, ze kterého chceme data čerpat, například `jicinska`.

`POCET OBCHODNICH CESTUJICICH M`
: Zadejme požadované množství obchodních cestujících, například `4`.

`HORNI MEZ L`
: Zadejme požadované maximální množství měst, která mohou jednotliví obchodní cestující navštívit, například `15`.

`DOLNI MEZ K`
: Zadejme požadované minimální množství měst, která mohou jednotliví obchodní cestující navštívit, například `4`.

Pokud si budeme přát, aby hodnoty proměnných `x[i,j]`, případně `x[i,j,k]` byly po optimalizaci přesunuty do Excelu, použijeme v příslušném modelu následujícího příkazu.

```
Export to ExcelRange("KAM EXPORTOVAT");
```

A.1 MPL model TSP

Soubor `TSP.mpl` obsahuje následující kód.

```
TITLE TSP;

OPTIONS
ExcelWorkBook="data.xlsx";
ExcelSheetname="JMENO LISTU";

INDEX
i:=ExcelRange("i");
j:=ExcelRange("j");

DATA
c[i,j]:=ExcelRange("data");
n:=ExcelRange("n");

BINARY VARIABLES
!x[i,j] Export to ExcelRange("xx");
x[i,j];

INTEGER VARIABLES
!u[i] Export to ExcelRange("u");
u[i];

MODEL
MIN Z = sum(i,j:c[i,j]*x[i,j]);

SUBJECT TO
vyjezdy[i]:   sum(j:x[i,j])=1;
vjezdy[j]:    sum(i:x[i,j])=1;
hrana[i,j>=2]: u[i]+1-n*(1-x[i,j])<=u[i:=j];

END
```

A.2 MPL model MTSP

Soubor `MTSP.mpl` obsahuje následující kód.

```
TITLE MTSP;

OPTIONS
ExcelWorkBook="data.xlsx";
ExcelSheetname="JMENO LISTU";

INDEX
i:=ExcelRange("i");
j:=ExcelRange("j");

DATA
c[i,j]:=ExcelRange("data");
m = POCET OBCHODNICH CESTUJICICH M;
L = HORNI MEZ L;

BINARY VARIABLES
!x[i,j] Export to ExcelRange("xx");
x[i,j];

INTEGER VARIABLES
!u[i] Export to ExcelRange("u");
u[i];

MODEL
MIN Z = sum(i,j:c[i,j]*x[i,j]);

SUBJECT TO
startvyjezdy:   sum(j>=2:x[1,j]) = m;
startvjezdy:    sum(i>=2:x[i,1]) = m;
vyjezdy[i>=2]:  sum(j:x[i,j]) = 1;
vjezdy[j>=2]:   sum(i:x[i,j]) = 1;
anticykl[i,j>=2]: u[i]-u[i:=j] + L*x[i,j] <= L-1;

END
```

A.3 MPL model MTSP-VRP

Soubor `VRP.mpl` obsahuje následující kód.

```
TITLE x_ijk_VRP;

OPTIONS
ExcelWorkBook="data.xlsx";
ExcelSheetname="JMENO LISTU";

INDEX
i:=ExcelRange("i");
j:=ExcelRange("j");
p:=ExcelRange("i");
k = 1..POCET OBCHODNICH CESTUJICICH M;


DATA
c[i,j]:=ExcelRange("data");
L = HORNI MEZ;

BINARY VARIABLES
!x[i,j,k] Export to ExcelRange("xxx");
x[i,j,k];

INTEGER VARIABLES
!u[i] Export to ExcelRange("u");
u[i];

MODEL
MIN Z = sum(i,j: c[i,j] * sum(k: x[i,j,k]) );

SUBJECT TO
navstivit[j>=2]:  sum(i,k: x[i,j,k]) = 1;
ustalenost[k,p]:  sum(i: x[i,j:=p,k]) - sum(j: x[i:=p,j,k]) = 0;
salesman[k]:      sum(j>=2: x[1,j,k]) = 1;
anticykl[i,j>=2]: u[i] - u[i:=j] + L * sum(k: x[i,j,k]) <= L - 1;

END
```

A.4 MPL model MTSP-KaBe

Soubor `MTSP_KaBe.mpl` obsahuje následující kód.

```
TITLE MTSP_KaBe;

OPTIONS
ExcelWorkBook="data.xlsx";
ExcelSheetname="JMENO LISTU";

INDEX
i:=ExcelRange("i");
j:=ExcelRange("j");

DATA
c[i,j]:=ExcelRange("data");
m = POCET OBCHODNICH CESTUJICICH M;
L = HORNI MEZ L;
K = DOLNI MEZ K;

BINARY VARIABLES
!x[i,j] Export to ExcelRange("xx");
x[i,j];

INTEGER VARIABLES
!u[i] Export to ExcelRange("u");
u[i];

MODEL
MIN Z = sum(i,j:c[i,j]*x[i,j]);

SUBJECT TO
startvyjezdy:   sum(j>=2:x[1,j]) = m;
startvjezdy:    sum(j>=2:x[i:=j,1]) = m;
vjezdy[j>=2]:   sum(i:x[i,j]) = 1;
vyjezdy[i>=2]:  sum(j:x[i,j]) = 1;

hornimez[i>=2]: u[i] + x[1,j:=i]*(L-2) - x[i,1] <= L-1;
dolnimez[i>=2]: u[i] + x[1,j:=i] + x[i,1]*(2-K) >= 2;
doplnek[i>=2]:  x[1,j:=i] + x[i,1] <= 1;
anticykl[i,j>=2]:   u[i]-u[i:=j] + L*x[i,j] + x[i:=j,j:=i]*(L-2) <= L-1;

END
```

A.5 MPL model MTSP-VRP-KaBe

Soubor `VRP_KaBe.mpl` obsahuje následující kód.

```
TITLE x_ijk_VRP_KaBe;

OPTIONS
ExcelWorkBook="data.xlsx";
ExcelSheetname="JMENO LISTU";

INDEX
i:=ExcelRange("i");
j:=ExcelRange("j");
p:=ExcelRange("i");
k = 1..POCET OBCHODNICH CESTUJICICH M;

DATA
c[i,j]:=ExcelRange("data");
L = HORNI MEZ L;
K = DOLNI MEZ K;

BINARY VARIABLES
!x[i,j,k] Export to ExcelRange("xxx");
x[i,j,k];

INTEGER VARIABLES
!u[i] Export to ExcelRange("u");
u[i];

MODEL
MIN Z = sum(i,j: c[i,j] * sum(k: x[i,j,k]) );

SUBJECT TO

navstivit[j>=2]:    sum(i,k: x[i,j,k]) = 1;
ustalenost[k,p]:    sum(i: x[i,j:=p,k]) - sum(j: x[i:=p,j,k]) = 0;
salesman[k]:        sum(j>=2: x[1,j,k]) = 1;

hornimez[i>=2]:  u[i] + sum(k: x[1,j:=i,k]*(L-2)) - sum(k: x[i,1,k]) <= L-1;
dolnimez[i>=2]:  u[i] + sum(k: x[1,j:=i,k]) + sum(k: x[i,1,k]*(2-K)) >= 2;
doplnek[i>=2]:   sum(k: x[1,j:=i,k]) + sum(k: x[i,1,k]) <= 1;
anticykl[i,j>=2] : u[i] - u[i:=j] + sum(k: L * x[i,j,k]) +
                                    sum(k: x[i:=j,j:=i,k]*(L-2)) <= L - 1;

END
```

B. Přiložené soubory

K práci přikládáme následující soubory.

1. Soubor `data.xlsx` obsahující získané GPS lokátory a vzdálenosti mezi jednotlivými sokolskými jednotami pro určité sokolské župy. Obsahuje také kombinaci několika žup v jednom listu.
2. Soubor `porovnani_vysledku.xlsx` obsahující získané výsledky programu MPL for Windows a řešitele GUROBI pro všechny skripty z přílohy A a různě nastavené počty obchodních cestujících a mezí, které musí splňovat.
3. Soubor `vystupy_x.xlsx` obsahující data pro skript v jazyku R, tedy matice přechodů, GPS lokátory a další potřebné údaje pro vykreslování obrázků.
4. Soubor `mapka.R`, který slouží k vykreslení cest obchodních cestujících do mapy žup Jičínské - Bergrovy, Orlické, Podkrkonošské - Jiráskovy a Východočeské - Pipichovy.
5. Soubor `greedy_MTSP.R`, který řeší úlohu MTSP o velikosti $n = 156$ metodou nejbližšího souseda. Výsledek je následně zakreslen do mapy.