České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství
Obor: Aplikace softwarového inženýrství

Plánování trasy projektu Revolution Train pomocí úlohy obchodního cestujícího

Route Planning for the Revolution Train Project Using Travelling Salesman Problem

BAKALÁŘSKÁ PRÁCE

Vypracoval: Kristian Matuštík
Vedoucí práce: Ing. Adam Borovička, Ph.D.
Rok: 2022

České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Katedra softwarového inženýrství

Akademický rok 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	**Kristian Matuštík**
Studijní program:	**Aplikace přírodních věd**
Obor:	**Aplikace softwarového inženýrství**
Název práce česky:	**Plánování trasy projektu Revolution Train pomocí úlohy obchodního cestujícího**
Název práce anglicky:	**Route Planning for the Revolution Train Project Using Travelling Salesman Problem**

Pokyny pro vypracování:

1. Specifikujte reálný ekonomický problém - optimalizace trasy projektu Revolution Train, a stanovte metodologii jeho řešení.
2. Představte teorii úlohy obchodního cestujícího, vysvětlete způsoby jejího řešení a popište některé modifikace úlohy pro reálné použití.
3. Sestavte graf a matematický model daného problému pro jeho řešení.
4. Naplánujte trasu projektu Revolution Train prostřednictvím hledaného řešení naformulovaného matematického modelu.
5. Analyzujte získané výsledky a porovnejte se stávající situací.

Doporučená literatura:

[1] Applegate, D. L. *The Traveling Salesman Problem: A Computational Study.* Princeton: Princeton University Press, 2006. ISBN 9780691129938.

[2] Gutin, G., Punnen, A. P. *The Traveling Salesman Problem and Its Variations.* Dordrecht: Kluwer Academic Publishers, 2002. ISBN 9781402006647.

[3] Toth, P., Vigo, D. *The Vehicle Routing Problem.* Philadelphia: Society for Industrial and Applied Mathematics, 2002. ISBN 9780898715798.

Jméno a pracoviště vedoucího práce:

Ing. Adam Borovička, Ph.D.
Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, ČVUT v Praze

Jméno a pracoviště konzultanta:

–

.....
vedoucí práce

Datum zadání bakalářské práce: 15. 10. 2021

Termín odevzdání bakalářské práce: 7. 7. 2022

Doba platnosti zadání je dva roky od data zadání.

.....
garant oboru

.....
vedoucí katedry

.....
děkan

V Praze dne 15. 10. 2021

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne ....

Kristian Matuštík

Poděkování

Chtěl bych poděkovat Ing. Adamu Borovičkovi, Ph.D. za trpělivost při vedení mé práce a podnětné návrhy k jejímu zpracování. Dále děkuji rodičům za pomoc s korekturou textu a projektu Revolution Train za poskytnutá data.

Kristian Matuštík

Název práce:

Plánování trasy projektu Revolution Train pomocí úlohy obchodního cestujícího

Autor: Kristian Matuštík

Studijní program: Aplikace přírodních věd
Obor: Aplikace softwarového inženýrství
Druh práce: Bakalářská práce

Vedoucí práce: Ing. Adam Borovička, Ph.D.
Katedra softwarového inženýrství, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze
Konzultant: –

Abstrakt: Tato práce se zabývá optimalizací tras projektu Revolution Train metodami pro řešení problému obchodního cestujícího. Za tímto účelem byl vytvořen program v jazyce Java, který pro zadané body na železniční síti nalezne potřebnou matici vzdáleností a následně pro ně trasu optimalizuje. Pro hledání této trasy byla implementována metoda optimalizující náhodná výchozí řešení 2-opt algoritmem, která dokázala pro testovací soubory s velikostí úloh do padesáti dvou bodů spolehlivě nalézt optimální řešení za méně než jednu sekundu. V práci byly programem nejdříve optimalizovány dvě již uskutečněné trasy projektu, pro které byly nalezeny cesty kratší o 13,3 %, resp. 3,7 % původní délky. Na závěr byla navržena trasa pro budoucí tour projektu.

Klíčová slova: 2-opt algoritmus, Java, optimalizace, problém obchodního cestujícího

Title:

Route Planning for the Revolution Train Project Using Travelling Salesman Problem

Author: Kristian Matuštík

Abstract: This paper focuses on the optimization of the Revolution Train project routes by methods for solving The Traveling Salesman Problem. For this purpose, a Java program has been developed to find the required distance matrix for given points on the rail network and subsequent optimization of the route. A method optimizing random default routes with a 2-opt algorithm was implemented for this purpose, which was able to reliably find the optimal solution in less than one second for test sets with problem sizes up to fifty-two points. The created program was first used to optimize two previous routes of the project, for which routes that were shorter by 13.3 % and 3.7 % respectively were found. Finally, a route was proposed for a future tour of the project.

Key words: 2-opt algorithm, Java, optimization, Travelling Salesman Problem

Obsah

Úvod

Minimalizace nákladů je jednou z priorit každé rozumně vedené firmy. Jedním ze způsobů, jakými lze snížení nákladů dosáhnout, je optimalizace pracovních postupů a tak snížení nákladů s nimi spojených. Ačkoliv se každá firma pohybuje v jiném oboru a zabývá se jinými problémy, některé jejich činnosti mohou nést společné základní rysy. Ty tak lze často shrnout do stejné teoretické úlohy, kterou je poté možné zkoumat a získané poznatky zpětně v daných odvětvích aplikovat. Tato práce se zabývá využitím jedné takové úlohy pro řešení reálného ekonomického problému, konkrétně optimalizací tras projektu Revolution Train za použití metod pro řešení problému obchodního cestujícího.

Revolution Train je projektem nadačního fondu NOVÉ ČESKO, zabývající se primární protidrogovou prevencí. Program projektu je koncipován jako interaktivní a zážitkové vzdělávání, založené na skutečném příběhu. Ten je prezentován v multimediální vlakové soupravě, díky které se může rychle a efektivně přesouvat mezi různými městy, bez nutnosti celou expozici sestavovat pokaždé znovu. Při jedné z tzv. „tour“ projektu vlak během krátké doby navštíví několik desítek měst a ujede tisíce kilometrů, od kterých se odvíjí celkové náklady přepravy. Při jejich plánování se trasy dosud vybíraly na základě intuice a nebyla snaha o nalezení efektivnější metody. Cílem této práce je tak využít existující teorie k problému obchodního cestujícího pro plánování tras tohoto projektu.

První kapitola práce se věnuje shrnutí některých poznatků k problému obchodního cestujícího. Nejdříve je zmíněna jeho základní definice spolu s matematickým modelem a grafickou reprezentací. Dále jsou představeny vybrané modifikace úlohy, které se často v reálných zadáních vyskytují. V neposlední řadě jsou pak uvedeny některé způsoby řešení této úlohy. Cílem kapitoly je poskytnout čtenáři základní znalosti problému, především ty důležité k pochopení dalších částí práce, kde se budou získané teoretické poznatky aplikovat při tvorbě počítačového programu a následné optimalizaci reálného ekonomického problému.

Druhá kapitola popisuje problematiku projektu Revolution Train. Po úvodním představení je popsán princip jeho fungování a způsob, jakým lze využít optimalizačních metod k řešení problému obchodního cestujícího pro minimalizaci jeho přepravních nákladů. Na závěr jsou zde prezentována vybraná data, která budou v dalších kapitolách optimalizována.

Třetí kapitola se věnuje struktuře a fungování počítačového programu, který jsem za účelem řešení této úlohy vytvořil. Nejdříve jsou vysvětleny důvody, proč jsem se rozhodl pro tento přístup k řešení. Je zde popsána problematika převedení ekono-

mického zadání do matematické formulace úlohy obchodního cestujícího, se kterou je možné v počítači dále pracovat. Poté je představena a otestována samotná metoda, kterou jsem pro hledání řešení této úlohy zvolil. Na závěr je pak vysvětlena práce se samotným grafickým rozhraním tohoto programu. Popis jeho funkčnosti i rozhraní je uzpůsoben čtenáři bez větších znalostí programování, detailněji lze však do fungování programu nahlédnout ve zdrojových kódech v přílohách práce.

Poslední kapitola obsahuje analýzu výsledků získaných optimalizací tras projektu za pomoci vytvořeného programu. Je zde odvozena účinnost této optimalizace srovnáním získaných tras s trasami původními, plánovanými předchozím způsobem. Data jsou prezentována jako výstupy vytvořeného programu a zobrazení těchto tras na mapě. Na závěr je pak navržena trasa pro budoucí tour projektu.

Kapitola 1

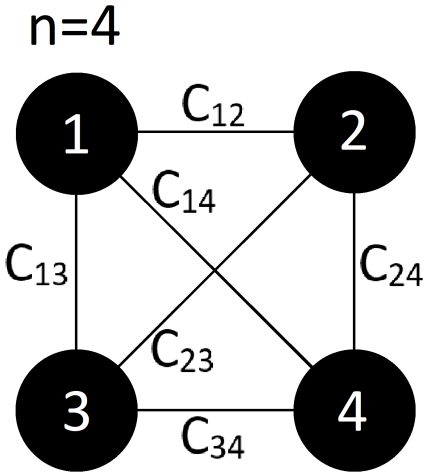
Problém obchodního cestujícího

Problém obchodního cestujícího je jednou z nejstarších optimalizačních úloh. Nachází uplatnění v široké škále činností, a proto se jedná o jedno z nejdůsledněji zkoumaných témat ve výpočetní matematice [14]. V této kapitole jsou shrnuty některé základní poznatky k této úloze, jako je její definice, některé modifikace a metody jejího řešení. Cílem kapitoly je poskytnout čtenáři teoretický základ, nutný pro porozumění následujících kapitol, kde se budou části této teorie aplikovat při tvorbě programu pro optimlizaci tras projektu Revolution Train.

1.1 Definice

Problém obchodního cestujícího lze stručně popsat jako úlohu, jejímž cílem je nalézt nejkratší okruh, tedy trasu vycházející z a vracející se do stejného bodu, který prochází právě jednou všemi zadanými místy. Obecněji a formálněji lze problém popsat jako hledání nejkratšího Hamiltonovského cyklu v ohodnoceném grafu. Nemusí se tak jednat o skutečná místa na mapě, ale o libovolné vrcholy propojené ohodnocenými hranami mezi nimi. Od tohoto grafu se pak pro zaručení existence řešení a funkčnosti některých algoritmů často vyžaduje úplnost, neboli existence hran mezi každými dvěma vrcholy. Pokud zadaný graf úplný není, je možné jej podle ekonomické podstaty reálného problému na úplný doplnit. To se provede například přidáním chybějících hran s prohibitivně vysokým ohodnocením, pokud je nežádoucí, aby se ve výsledném řešení využily. Další možností je přidání hran s ohodnocením odpovídajícím nejkratší cestě mezi těmito vrcholy v původním grafu, které lze dohledat například Dijkstrovým algoritmem. Hrany pak mohou být orientované, čili mít různé ohodnocení v obou směrech, nebo neorientované, s ohodnocením stejným.

Jak je z definice patrné, s tímto problémem se v jeho základní podobě setkávají například poštovní služby rozvážející zboží, nebo účastnící orientačního běhu. Využití však nachází i mimo plánování skutečné cesty dopravními prostředky. Nejkratší okruh mezi zadanými body dokáže zvýšit efektivitu také robotizovaných vrtaček [9] nebo 3D tisku [7], ale úloha a její modifikace nalézají použití i v oblastech astronomie [2] nebo sekvencování DNA [1].



Obrázek 1.1: Příklad úplného neorientovaného grafu

1.2 Matematický model

Požadavky slovní definice lze vyjádřit formálním matematickým modelem. Výslednou soustavu rovnic pak lze využít například pro samotné hledání řešení problému. V této formulaci n značí přirozené číslo odpovídající počtu vrcholů v grafu, tedy celkový počet bodů k navštívení. Čísla c_{ij} jsou ohodnocením jednotlivých hran mezi vrcholy i a j pro všechny dvojice indexů ij nabývajících hodnot od 1 do n. Těchto čísel je tak pro daný graf celkem n^2 a často se zapisují do takzvané matice vzdáleností. Jedná se o čtvercovou matici velikosti $n \times n$, kde jsou jednotlivé prvky umístěny do řádků a sloupců v pořadí podle jejich dvou indexů. Pokud se jedná o neorientovaný graf, tedy všechny hrany mají stejné ohodnocení v obou směrech, tato matice je navíc symetrická. Diagonální prvky c_{ii}, značící hranu vystupující z a vstupující do stejného bodu, se pokládají rovny nule, nebo nekonečnu. Matice vzdáleností je pak jediným údajem potřebným pro specifikaci tohoto modelu a následné řešení konkrétního problému, proto je její nalezení jedním ze dvou hlavních úkolů při řešení praktické úlohy. Ohodnocením hran může být libovolná veličina, kterou je potřeba pro celou cestu minimalizovat. Musí být však pro danou úlohu pouze jedna a ve stejných jednotkách u všech hran příslušného grafu. Při hledání skutečné trasy se tak může jednat například o vzdálenost mezi dvěma body, dobu trvání cesty mezi nimi, nebo náklady spojené s touto cestou.

$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij} : \tag{1.1}$$

$$\begin{aligned} \sum_{i=1, i \neq j}^{n} x_{ij} &= 1 \qquad j = 1, ...n; \\ \sum_{j=1, j \neq i}^{n} x_{ij} &= 1 \qquad i = 1, ...n; \end{aligned} \tag{1.2}$$

$$u_i - u_j + nx_{ij} \leq n - 1 \qquad 1 \leq i \neq j \leq n; \tag{1.3}$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 1, ..., n.$$

Proměnné x_{ij} nabývají hodnot pouze 0 nebo 1 a vyjadřují, jestli se ve výsledné cestě hrana z bodu i do bodu j využila, čili jestli skrze ni výsledný okruh vede. Rovnice 1.1, známá jako účelová funkce, udává hlavní cíl úlohy v podobě minimalizace celkové délky uskutečněné cesty. Podmínky 1.2 zajišťují, že se do každého vrcholu vstupuje a z každého se vystupuje právě jednou. Nakonec jsou nutné takzvané anticyklické podmínky (1.3). Ty zabraňují existenci více menších cyklů v řešení, neboli zajišťují, že výsledná trasa bude tvořit pouze jeden okruh procházející všemi body. Této podobě podmínek se říká po jejích představitelích Miller–Tucker–Zemlinova (MTZ) formualce [8]. Proměnné u_i (u_j) jsou pomocné a udávají pořadí navštívených míst.

1.3 Modifikace

Reálné ekonomické problémy se často od základní definice mírně liší. Vyplývají ze stejné podstaty v podobě minimalizace celkové délky trasy, ale dále úlohu určitým způsobem upravují. Může se jednat například o potřebu vstoupit do některých bodů vícekrát, nebo naopak možnost navštívit pouze určitý počet, nikoliv všechny zadané body. Z tohoto důvodu existují různé modifikace problému, které jednotlivé změny v zadání zohledňují. Ty je poté možné řešit buďto způsobem specifickým pro dané modifikace, nebo jejich transformací na základní model problému.

Jednou z často využívaných modifikací je vícenásobná úloha obchodního cestujícího. Jedná se o úlohu, kdy existuje pomyslných obchodních cestujících více. Může se tedy jednat například o více vozidel poštovní služby nebo o školní autobusy vozící studenty. V řešení se tato modifikace projeví existencí více menších okruhů, které se protínají ve výchozím bodě a které pouze po sjednocení prochází všemi zadanými místy. Úlohu je možné řešit například rozdělením všech bodů do několika oblastí s průnikem ve výchozím bodě a hledáním řešení základního problému každé z nich. Další možností je transformace úlohy na základní problém obchodního cestujícího vytvořením několika přídatných bodů. Těm se přiřadí nulové vzdálenosti do všech bodů zadané úlohy, zatímco do všech přídatných vzdálenosti prohibitivně velké [3]. Následně je možné řešit úlohu jako klasický problém obchodního cestujícího.

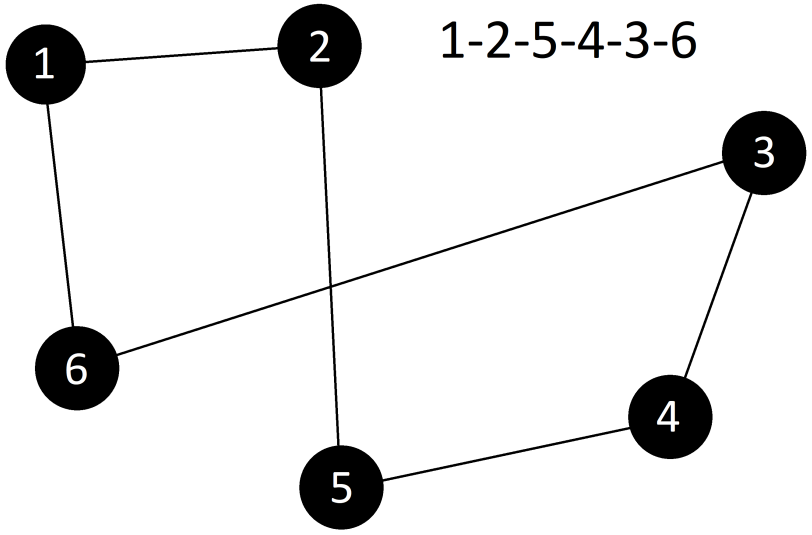
Další modifikací, která bude využita v praktické části této práce, je otevřený problém obchodního cestujícího. Od základní úlohy se liší pouze tím, že trasa není uzavřená, ale začíná a končí v jiných, předem daných místech. Ačkoliv jí lze také vyjádřit vlastním matematickým modelem, pro praktické řešení je výhodnější úlohu převést na základní problém obchodního cestujícího. To se provede vytvořením jednoho přídatného bodu, kterému se v grafu definují nulová ohodnocení hran jdoucích do počátečního a do koncového vrcholu, zatímco do všech ostatních se přiřadí nějaká prohibitivně vysoká hodnota. Tím se zařídí, že při následné optimalizaci úlohy jako základního problému obchodního cestujícího budou využity obě hrany s nulovým

ohodnocením a přídatný bod se bude nacházet mezi počátečním a koncovým vrcholem. Po jeho odstranění se tak získá cesta procházející všemi požadovanými body, která začíná v počátečním a končí v cílovém místě. Samozřejmě je důležité u získaného řešení ověřit, že tomu tak skutečně je a nebyly využity některé z prohibitivně ohodnocených cest, které by tento model „rozbily“.

1.4 Metody řešení

Problém obchodního cestujícího se řadí mezi NP-těžké úlohy. Počet přípustných řešení roste s faktoriálem počtu bodů a není znám žádný algoritmus, který by byl schopen nalézt optimální řešení úlohy v polynomiálním čase. Z toho důvodu je už pro malé problémy s desítkami bodů neefektivní, nebo i časově nemožné exaktní řešení hrubou silou nalézt. Při optimalizaci problému se tak využívají častěji heuristické algoritmy, spočívající v hledání nejlepšího možného řešení v dostupném čase.

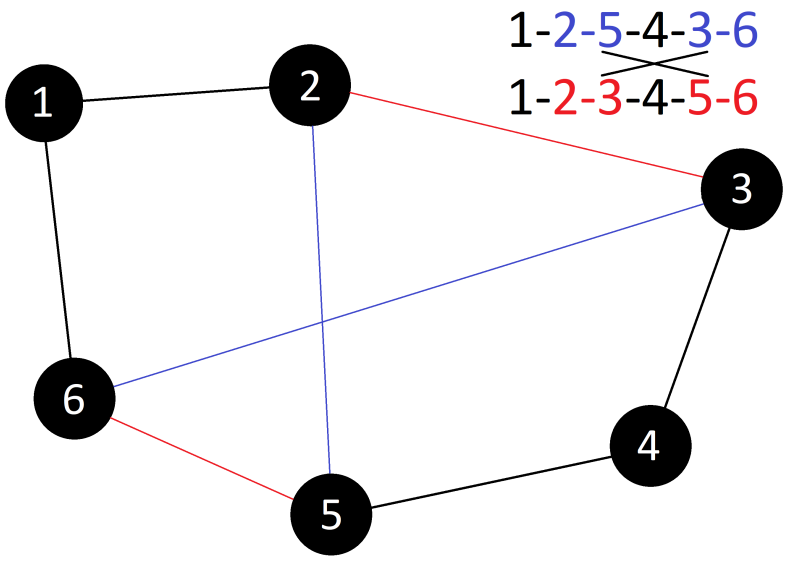
Jednou z možností řešení problému je využití programů pro optimalizaci lineárních soustav s matematickým modelem této úlohy, například softwaru LINGO [6]. Zatímco pro malé úlohy je možné nalézt i optimální řešení poměrně rychle, pro velké je podmínek velké množství a tento způsob řešení také není příliš efektivní. Častěji se proto nevychází z celého matematického modelu, ale využívají se metody pracující s již známými přípustnými řešeními. Ta lze totiž zapsat jako permutaci n různých čísel, odpovídajících jednotlivým bodům v úloze a značících jejich pořadí. Například trasu okruhu z obrázku 1.2 lze zapsat jako posloupnost 1-2-5-4-3-6. Stejně tak i libovolná permutace popisuje jedno z přípustných řešení problému a jejími úpravami s jinou výslednou permutací stejných čísel je možné dojít opět k přípustnému řešení problému. Algoritmy tak mohou pracovat pouze s touto posloupností a při zachování vlastností permutace postupně konvergovat k lepším přípustným řešením.



Obrázek 1.2: Příklad řešení úlohy obchodního cestujícího

Už počáteční trasu lze nalézt nějakým algoritmem, lepším než jen náhodným výběrem. Asi nejznámějším a nejjednodušším způsobem je metoda nejbližšího souseda. Ta spočívá ve výstavbě cesty postupným přidáváním nejbližšího, doposud nenavštíveného bodu. Nejdříve se tedy nalezne bod nejblíže k výchozímu místu. Poté se hledá nejbližší, doposud nenavštívený bod k tomuto novému konci trasy, tedy jakýkoliv kromě výchozího místa. Tímto způsobem se pokračuje dále, dokud existují v grafu nenavštívené vrcholy. Poté se cesta uzavře vrácením cesty zpět do výchozího místa. Ačkoliv je tento „chamtivý“ algoritmus velmi rychlý (složitost $O(n^2)$), většinou nenalezne příliš dobré řešení. Může však posloužit pro získání výchozího řešení před optimalizací jiným, složitějším algoritmem.

Jednou z heuristik operujících úpravami přípustných řešení, která bude využívána v praktické části této práce, je 2-opt algoritmus [4]. Jedná se o jednoduchou metodu, spočívající v postupných výměnách dvou hran z aktuální trasy za jiné. Algoritmus postupně prochází všechny jejich možné dvojice z aktuálního řešení a srovnává jejich kombinovanou délku s hranami, za které je lze zaměnit. Ty jsou pro každou dvojici původních hran jednoznačně určeny, jelikož se musí nacházet mezi stejnými čtyřmi body a zároveň zachovávat v řešení jediný cyklus. Názorně je výběr odpovídajících hran touto metodou zobrazen na ilustraci 1.3, kde se dvě modré z původního řešení mohou zaměnit za nové červené. Pro zápis řešení ve formě posloupnosti bodů lze proces popsat jako otočení pořadí čísel v části této posloupnosti mezi krajními body dané záměny, jak je také patrné z obrázku. Tímto způsobem se záměna například implementuje v počítačovém programu. Pokud se naleznou nové hrany, které jsou v součtu kratší než dvě původní, tedy i výsledná trasa by měla kratší celkovou délku, tato záměna se provede a postup se s novou trasou opakuje od začátku. Alternativně lze nalezené řešení pouze dočasně uložit, pokračovat přes zbylé dvojice hran aktuální trasy a srovnávat ostatní záměny s touto uloženou trasou. Nakonec se tak zvolí nejlepší nalezená záměna, se kterou se začíná od začátku. Pokud se prošly všechny dvojice hran a k žádnému zlepšení trasy nedošlo, algoritmus končí.



Obrázek 1.3: Příklad záměny tras 2-opt algoritmem

Ačkoliv je tato metoda schopna nalézt od pohledu dobrá řešení, stejně jako u jiných heuristických algoritmů zde není zaručeno, že se jedná o řešení optimální. Většinou je nalezeno pouze určité lokální minimum, do kterého algoritmus dokonvergoval a ze kterého už se stejným postupem k žádnému lepšímu řešení nedostane. Pro možnost nalezení kratší trasy se proto často využívá v kombinaci s dalšími metodami, jako je například simulované žíhání, kdy se výsledná trasa určitým způsobem zhorší a předpokládá se, že opětovná aplikace algoritmu umožní nalezení lepšího řešení.

Další jednoduchou metodou, využitou při řešení praktické úlohy v této práci, je optimalizace mnoha různých výchozích tras. Tímto způsobem se tak nalezne větší množství lokálních minim, ze kterých je poté možné zvolit to nejlepší. Ač tento způsob nejspíš není nejefektivnější, metoda je jednoduchá na implementaci a pro testované úlohy s rozsahem okolo padesáti měst byla schopná nalézt i optimální řešení velmi rychle (str. 25), proto je pro účely dané problematiky postačující.

Kapitola 2

Revolution Train

V této kapitole je představen projekt Revolution Train, princip jeho operace a způsob, jakým lze ke snížení jeho provozních nákladů využít optimalizačních metod problému obchodního cestujícího. Poté jsou prezentována samotná data, která budou v další kapitole využita pro optimalizaci.

2.1 O projektu Revolution Train

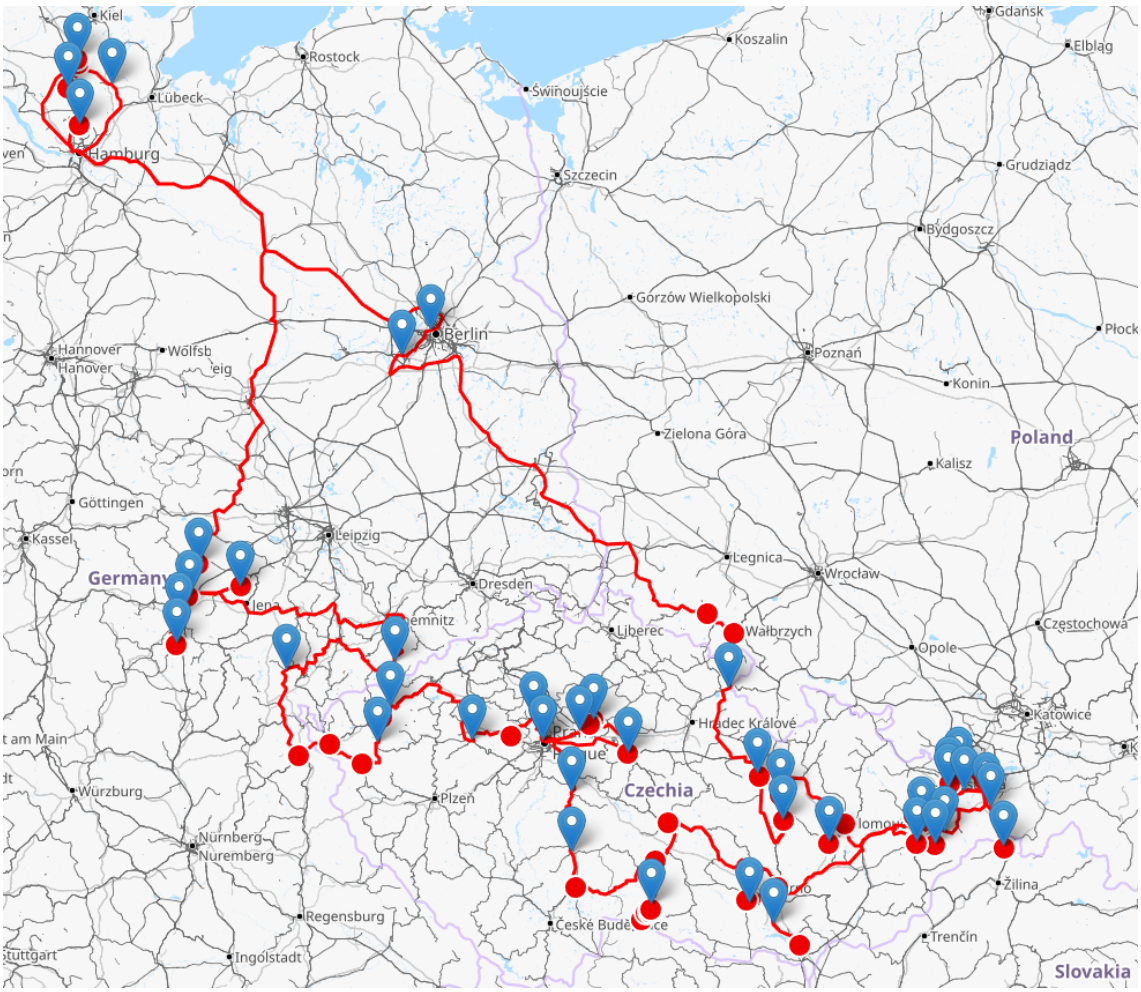
Revolution Train je jedním z projektů neziskové organizace NOVÉ ČESKO, zaměřený na primární protidrogovou prevenci, především u dospívajících dětí. Projekt je koncipován jako průchozí, interaktivní expozice vyprávějící skutečný příběh, který upozorňuje na nebezpečí a problémy spojené s užíváním drog. Celá expozice je navíc umístěna ve vlakové soupravě. Významným prvkem projektu tak je jeho mobilita a schopnost se přesouvat ze dne na den, díky které je schopný navštívit desítky měst během několika měsíců při každé z tour. Jediným limitujícím faktorem pro návštěvu města projektem je napojení na železniční síť a dostupnost volné koleje, kde může být vlak po danou dobu odstaven a lidé k němu mohou bezpečně přistupovat.

Vlaková souprava má svou domovskou stanici na nádraží Praha – Dejvice, ze které na jaře a na podzim vyjíždí na turné po České republice a okolních zemích. Před každou cestou jsou uzavřeny smlouvy s jednotlivými městy, která mají o přistavení vlaku zájem. Tyto smlouvy však nespecifikují přesné datum návštěvy. Projekt tak má při plánování trasy volnost v určování termínů návštěv a ve výsledku pořadí jednotlivých měst. Vyvstává tedy otázka, jak trasu co nejlépe naplánovat. Hlavní snahou projektu je minimalizace ujeté vzdálenosti vlaku, jelikož má přímý dopad na celkové náklady na přepravu. Vlak za tímto účelem využívá služeb Českých drah, které mu poskytují tažnou lokomotivu spolu se zprostředkováním veškeré potřebné infrastruktury a další logistiky spojené s přesunem vlakové soupravy. Ačkoliv České dráhy ve vystavených fakturách nespecifikují jednotlivé účtované položky, je rozumné předpokládat, že nejvýznamnější variabilní částí bude právě ujetá vzdálenost. Náklady pro uskutečnění daných cest nebo odstavení vlaku ve stanicích totiž zůstanou stejné, bez ohledu na zvolenou cestu a pořadí navštívených měst. Minimalizací ujeté vzdálenosti se také snižují náklady na údržbu samotné soupravy a čas strávený na trati.

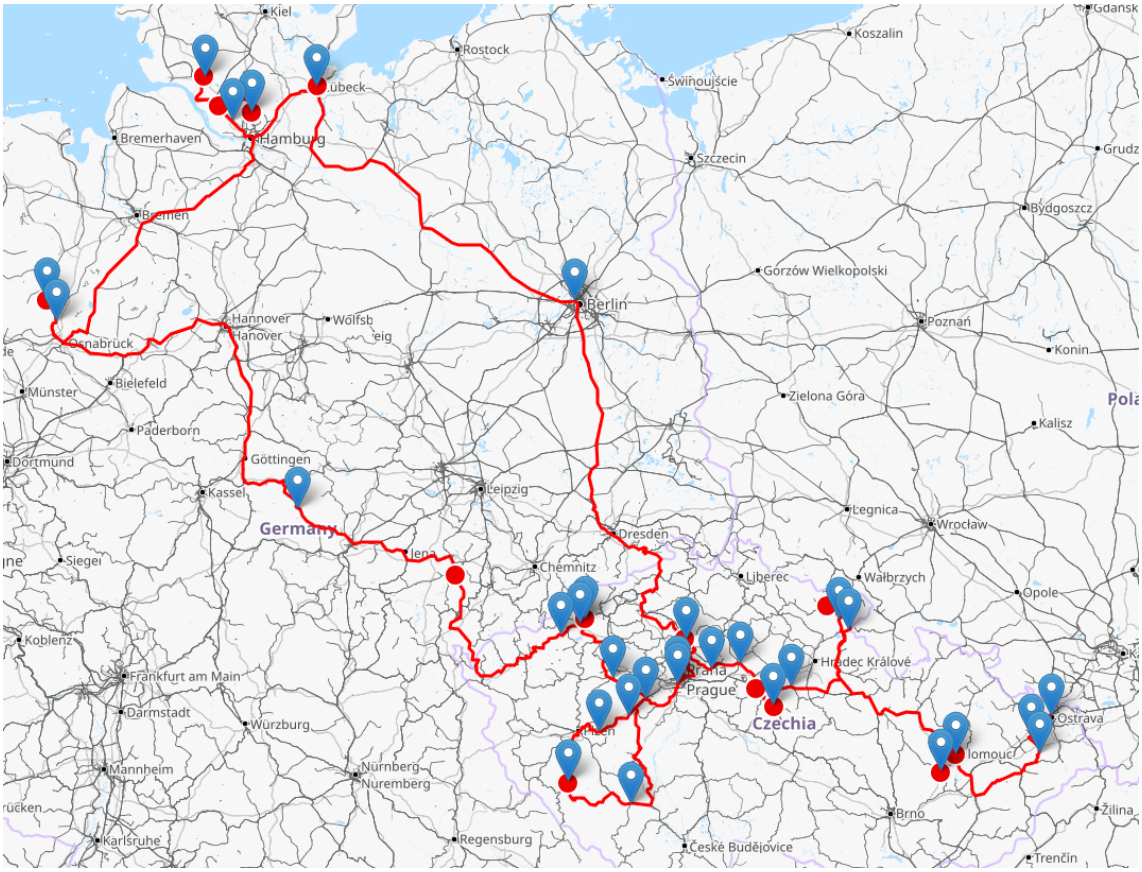
Ačkoliv je hlavním cílem při plánování tour minimalizace ujeté vzdálenosti, nikdy se důslednější optimalizaci trasy při jejím plánování nevěnovala příliš velká pozornost. Doposud se cesta vybírala na základě lidské intuice a prostého spojení požadovaných měst na mapě železniční sítě, jak to lidskému mozku přijde nejefektivnější. V této práci se proto pokusím cestu poprvé naplánovat s využitím optimalizačních algoritmů pro řešení problému obchodního cestujícího a srovnat délku vybraných, již uskutečněných tras s trasami získanými zvolenou metodou řešení.

2.2 Data pro optimalizaci

V této práci jsou optimalizovány celkem 3 různé trasy projektu. První dvě budou nalezeny optimalizací cest z již uskutečněných tour. Díky tomu bude možné srovnat jejich délky s cestami původními, plánovanými jednodušším způsobem. Konkrétně se jedná o trasy z podzimní tour 2021 a jarní tour 2022, navštěvující 42, respektive 31 měst. Mapy uskutečněných tras lze vidět na přiložených obrázcích, získaných pro daná města z raildar.fr/osrm/ [12].



Obrázek 2.1: Mapa tour podzim 2021



Obrázek 2.2: Mapa tour jaro 2022

Poslední trasa pak bude vytvořena pro města, která vlak plánuje navštívit při nadcházející tour na podzim 2022. Všechny tři seznamy těchto měst spolu s jejich souřadnicemi, které sloužily také jako vstupní soubory programu pro optimalizaci, lze nalézt v příloze A této práce. V případě jarní tour 2022 vlak navíc musel vyjíždět z jiného města, než ve kterém končil. Proto bylo nutné úlohu modifikovat pro otevřenou cestu způsobem popsaným v teoretické části práce. Výsledná matice vzdáleností, použitá pro hledání této cesty, je na přiloženém CD (příloha B).

Kapitola 3

Optimalizační program

Řešení ekonomické úlohy jako problému obchodního cestujícího lze rozdělit na dva dílčí úkoly. Prvním z nich je matematická formulace daného ekonomického problému, neboli nalezení matice vzdáleností pro požadovaná místa. Druhým úkolem je pak samotné řešení problému obchodního cestujícího s touto maticí. Pro řešení konkrétního problému, tedy plánování trasy vlakové soupravy, jsem se rozhodl vytvořit vlastní program, který je schopen řešit oba dílčí úkoly. Hlavním důvodem bylo nenalezení vhodného nástroje, který by zvládl matici vzdáleností pro železniční stanice nalézt. Jako nejlepší možnost se tak jevilo napsání vlastního kódu využívajícího routing engine OSRM Raildar [11] pro získání všech potřebných vzdáleností. Dále, když už byl program schopný tuto matici vytvořit, bylo rozumné přidat i nějaké funkce pro samotné řešení úlohy. Plánování trasy je tak velmi efektivní a pohodlné, jelikož není potřeba kopírovat data z jednoho programu do jiného, nebo získávat softwarové licence k jiným nástrojům. Popis funkčnosti programu a práce s jeho rozhraním je určen i čtenářům bez větších znalostí programování. Detailněji je však možné funkčnosti tohoto softwaru porozumět z jeho zdrojových kódů, umístěných na přiloženém CD (příloha B).

3.1 Funkčnost programu

Pro tvorbu programu jsem použil programovací jazyk Java a vývojové prostředí IntelliJ IDEA [5]. Jsou v něm využity knihovny java.nio, java.io, java.util a java.net pro samotné řešení úlohy. Pomocí knihoven java.awt, java.swing a java.text je pak vytvořeno grafické uživatelské rozhraní. Pro tvorbu matice vzdáleností na železnici je k nalezení potřebných vzdáleností využito API raildar.fr OSRM [11]. Program je dle funkčnosti rozdělen do dvou hlavních tříd. První třída s názvem „Stations“ obsluhuje samotnou logiku programu. Ukládají se zde potřebná data, tedy matice vzdáleností daného problému a trasa aktuálního řešení. Jsou zde také funkce pro nahrání těchto dat, nalezení matice vzdáleností se zadanými městy nebo pro samotné řešení úlohy. Druhá třída s názvem „Window“ pak vytváří grafické uživatelské rozhraní, které umožňuje snadnou obsluhu hlavních funkcí třídy „Stations“, jako je načítání vstupních dat ze souborů a řešení úlohy, ale také slouží k zobrazení výstupů programu.

3.1.1 Matice vzdáleností

Prvním krokem při řešení reálné úlohy odpovídající problému obchodního cestujícího je nalezení příslušné matice vzdáleností. To lze pro zadané body v závislosti na typu dopravy udělat mnoha způsoby. Asi nejjednodušší z nich, kterým je spočítání vzdáleností vzdušnou čarou pro dané souřadnice, není pro většinu problémů vhodný. Pokud se nejedná o trasu leteckou, lodní na otevřeném moři nebo jiný způsob dopravy umožňující přibližně přímou trasu, přípustné cesty budou konkrétním způsobem dopravy předem omezeny. Z toho důvodu často nelze počítat s tímto takzvaným euklidovským problémem, ale pro smysluplné řešení úlohy je nutné využít nějaký routing engine pro danou dopravní síť, který přesnější vzdálenosti daným způsobem dopravy nalezne. Zatímco pro silniční sítě existují různé aplikace k získání celé matice vzdáleností, pro železnici jsem žádný takový software nenašel. Narazil jsem však na nástroj OSRM Raildar [11]. Jedná se o routing engine pro železniční sítě, operující na podkladových datech z Open Railway Map. Volně dostupné API tohoto enginu umožňuje získání JSON souboru popisujícího trasu mezi zadanými body na železnici, který obsahuje mimo jiné celkovou délku této trasy. Z toho důvodu jsem se pro hledání matice vzdáleností rozhodl vytvořit jednoduchý program, který opakovaným voláním tohoto API pro všechny možné dvojice stanic potřebné vzdálenosti dohledá a matici vytvoří. Jako vstup programu jsou tak potřeba souřadnice míst k navštívení. Ty lze získat například ze stránky raildar.fr/osrm/ [12], která slouží k vizualizaci a ladění výstupu tohoto enginu. Proto je zde vhodné zároveň ověřit schopnost OSRM trasu do daného místa nalézt, jelikož jsem při testování tras na několik chyb ve výsledcích enginu narazil. Například se pro některé konkrétní souřadnice žádná trasa nenalezla, nebo chyběly v podkladových datech některé ze slepých průmyslových kolejí, které však vlak využíval. V prvním případě stačilo souřadnice lehce posunout, ve druhém pak nastavit na bod, kde se slepá kolej od zbytku železniční sítě odděluje.

Jako další variantu pro specifikaci matice program umožňuje i její přímé vložení. To je potřeba, pokud je matici nutné nejdříve upravit pro některou z modifikací problému obchodního cestujícího, nebo je již známá a lze tak programem řešit jakoukoliv úlohu obchodního cestujícího, ne jen na železnici. Mimo to lze matici získat i euklidovským způsobem pro 2D souřadnice, čehož bylo využito například při testování metody řešení, kdy byly některé z datasetů tímto způsobem zadány.

3.1.2 Řešení úlohy

Po definici konkrétní matice přichází na řadu samotná optimalizace. Pro tu jsem se rozhodl použít implementaci 2-opt algoritmu, popsaného v teoretické části práce. Jak bylo zmíněno, tato heuristika při spuštění nalezne pouze určité lokální minimum, podle zadaného výchozího řešení. Pro nalezení co nejlepšího výsledku jsem se proto rozhodl vyzkoušet metodu optimalizující velké množství náhodných výchozích tras, ze kterých se poté vybere ta nejlepší. Účinnost metody jsem se rozhodl otestovat na třech datasetech z TSPLIB95 [13], které velikostí odpovídají nejdelším trasám projektu Revolution Train.

Na třech vybraných souborech (příloha B) byla spuštěna testovací funkce, hledající nejlepší řešení daných úloh. Ta optimalizuje celkem milion náhodných výchozích tras 2-opt algoritmem a vypisuje každé nově nalezené řešení, které je lepší než všechna předchozí. Ve výstupu je v řádcích nejdříve pořadí dané optimalizace, následované délkou získané trasy a časem, ve kterém se k ní dospělo. Na závěr je pak vypsáno nejlepší nalezené řešení spolu s jeho podílem ve všech optimalizacích a celkovou dobou běhu testu. Zatímco jednotlivá průběžná řešení a časy jejich nalezení se mohou při každém spuštění testu lišit, výsledné „nejlepší“ řešení, jeho podíl ze všech optimalizací a doba běhu testu zůstávají pro daný soubor přibližně stejné. Výpisy jednoho z testů pro každý soubor jsou na přiložených obrázcích.

```
1: 1330.0       (10 ms)
7: 1297.0       (21 ms)
19: 1291.0      (26 ms)
34: 1273.0      (34 ms)
1000000+: 1273.0        (154919 ms)
best/all: 3.5789%
```

Obrázek 3.1: Výpis testu souboru **swiss42.txt**

```
1: 34849.0      (9 ms)
2: 34800.0      (13 ms)
3: 34272.0      (14 ms)
9: 33872.0      (17 ms)
35: 33858.0     (33 ms)
45: 33715.0     (38 ms)
299: 33587.0        (116 ms)
1206: 33522.0       (368 ms)
1000000+: 33522.0       (243146 ms)
best/all: 0.1782%
```

Obrázek 3.2: Výpis testu souboru **att48.txt**

```
1: 8179.0       (12 ms)
4: 7919.0       (15 ms)
14: 7796.0      (25 ms)
37: 7734.0      (41 ms)
48: 7685.0      (51 ms)
75: 7542.0      (68 ms)
1000000+: 7542.0        (310593 ms)
best/all: 1.124%
```

Obrázek 3.3: Výpis testu souboru **berlin52.txt**

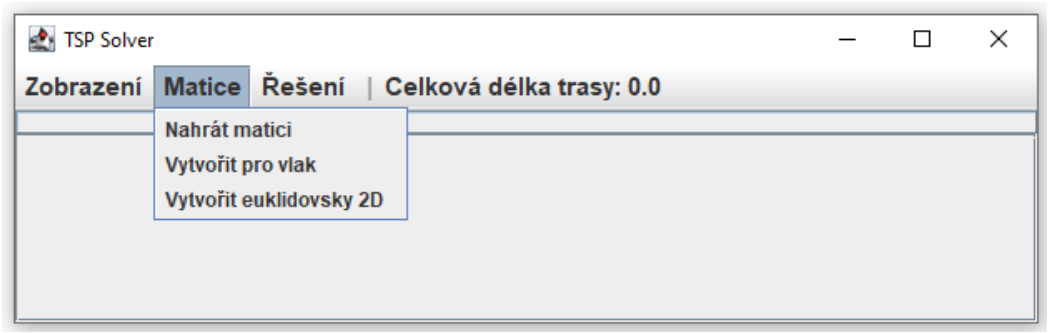
Z provedených testů vyplynulo několik důležitých poznatků. Pro vybrané úlohy bylo vždy po relativně krátké době nalezeno jedno lokální minimum, které už žádné jiné řešení při dalších optimalizacích nepřekonalo. U konkrétních testovacích souborů se navíc vždy jednalo o řešení optimální. Žádný důkaz pro potvrzení této pozorované vlastnosti i u jiných úloh nemám, avšak to není pro účely optimalizace trasy projektu důležité. Postačí znalost, že je metoda schopna nalézt jedno z nejlepších možných řešení v krátkém čase. Podíl zastoupení „nejlepšího“ řešení a tak pravděpodobnost jeho nalezení při optimalizaci náhodné trasy byla nejnižší pro dataset **att48.txt** s hodnotou 0,1782 %, kdy bylo z celkového milionu optimalizací nalezeno „nejlepší“ řešení 1782 krát. Průměrný čas optimalizace jedné z výchozích tras byl pozorován nejvýše 0,310 ms pro dataset **berlin52.txt**.

Z těchto důvodů jsem se rozhodl v programu hledat výsledné řešení právě optimalizací náhodných výchozích tras. Nejedná se sice o nejefektivnější způsob, avšak je jednoduchý na implementaci a jak vyplynulo z pozorování, pro velikosti daných problémů i velmi rychlý, účinný a tak více než postačující. Čas pro běh této metody byl nastaven na jednu sekundu. Při této době se i ve špatném scénáři, s průměrným časem 0,31 ms na optimalizaci, vyzkouší optimalizovat asi 3225 výchozích tras. Při nejnižším vypozorovaném zastoupení zmiňovaného „nejlepšího“ řešení 0,17 % pak bude tato trasa nalezena s pravděpodobností 99,58 % ($1 - 0,9983^{3225}$). I v tomto případě by tak se zvolenou metodou bylo téměř jisté, že se „nejlepší“ řešení nalezne. Zadání s menším počtem měst pak mají většinou průměrnou dobu běhu jedné optimalizace 2-opt algoritmem kratší a zastoupení „nejlepšího“ řešení ve výsledcích vyšší. Pro úlohy s podobným a nižším počtem měst, tedy velikosti tras tour projektu, tak byla jedna sekunda pro optimalizaci s výkonem použitého počítače postačující. V případě potřeby optimalizovat větší problémy by však bylo možné spustit hledání řešení touto metodou vícekrát, nebo ve zdrojovém kódu programu dobu běhu této metody změnit.

3.2 Použití programu

Samotná logika programu je pro jednoduchost doplněna o grafické uživatelské rozhraní. Tlačítka menu slouží k obsluze jednotlivých funkcí třídy „Stations“, ale je zde navíc i zobrazení aktuálního řešení ve formě tabulky s městy v daném pořadí, délky jednotlivých úseků a celkové délky této trasy. Také lze zobrazit používanou matici vzdáleností, pokud byla nalezena programem ze zadaných souřadnic a je potřeba ji upravit pro modifikace problému obchodního cestujícího, nebo uložit pro pozdější použití. Postup při optimalizaci trasy s pomocí tohoto programu je následující.

1. Nejdříve je potřeba problém specifikovat zadáním konkrétní matice vzdáleností. První možností, pokud je matice vzdáleností již známá, je její přímé vložení. Vstupem je textový soubor, ve kterém je nejdříve uveden počet bodů v úloze, následovaný názvy jednotlivých měst a řádky matice vzdáleností, jako ukazuje příklad z obrázku 3.5. Dialog pro nahrání souboru se otevře po kliknutí na menu: „Matice – Nahrát matici“. Tímto způsobem je tak možné optimalizovat libovolnou trasu, bez ohledu na způsob dopravy.



Obrázek 3.4: Možnosti zadání matice vzdáleností

```
4
město A   0 2 3 4
město B   2 0 5 7
město C   3 5 0 9
město D   4 7 9 0
```

Obrázek 3.5: Příklad vstupního souboru s maticí vzdáleností

Druhým způsobem, pro optimalizaci trasy vlaku nejdůležitějším, je specifikace úlohy textovým souborem se souřadnicemi daných měst, a to ve formátu příkladu z obrázku 3.6. Tento soubor se do programu nahrává v rámci menu: „Matice – Vytvořit pro vlak“. Program si poté vzdálenosti pro zadané souřadnice dohledá a matici vytvoří. Jelikož to může trvat pro větší úlohy delší čas, průběh tvorby matice lze vidět na progressbaru.

```
město A    50.00  15.00
město B    50.10  16.00
město C    49.80  15.20
město D    50.20  14.50
```

Obrázek 3.6: Příklad vstupního souboru se souřadnicemi

Posledním způsobem je spočítání euklidovské vzdálenosti pro zadané souřadnice. Vstupní soubor se vkládá v rámci menu „Matice - Vytvořit euklidovsky 2D“ ve stejném formátu, jako pro tvorbu matice na železnici, viz. obrázek 3.6. Je však nutné zvolit jiné souřadnice než geologické, jelikož jednotlivé stupně zeměpisné šířky a délky nejsou obecně stejně dlouhé. Vypočtená vzdálenost se zde také zaokrouhluje na celá čísla, jak to vyžadovaly testovací datasety, pro které byla metoda primárně určena. Z toho důvodu je dobré souřadnice specifikovat v celých jednotkách a řádech stovek nebo tisíců, aby docházelo k menším chybám. Vzdálenosti na zeměkouli jsou také 2D výpočtem zkresleny, funkce se proto může použít spíše pro úlohu vrtání děr do desky a dalších problémů, odpovídajících dvourozměrné ploše bez obstrukcí možných cest.

2. Po specifikaci úlohy nahráním vstupního souboru program nabízí 2 možnosti zobrazení výstupu. Na výpis aktuálního řešení, viz. obrázek 3.7, lze přepnout v rámci menu „Zobrazení - Trasa“. Tento výstup zobrazuje posloupnost měst v aktuální trase, spolu s délkami jednotlivých úseků. Celková délka aktuální trasy je pak zobrazena v horní liště vedle menu.

Obrázek 3.7: Zobrazení aktuální trasy

Druhou možností je zobrazení samotné matice vzdáleností, se kterou program pracuje, a to v rámci menu: „Zobrazení - Matice“. Výstup programu v tomto režimu lze vidět na obrázku 3.8. Toto zobrazení je užitečné, pokud je potřeba matici nejdříve upravit pro některou z modifikací problému obchodního cestujícího, nebo jí uložit pro pozdější použití.

Obrázek 3.8: Zobrazení používané matice

3. Optimalizaci zvolené trasy lze provést dvěma způsoby. Prvním z nich je metoda nejbližšího souseda, dostupná v programu v rámci menu: “Řešení - Metoda nejbližšího souseda”. Hlavní metodu pro optimalizaci, popsanou v teoretické části této práce, lze spustit pod menu: “Řešení - Náhodné 2-opt”. Tato metoda je nejdůležitější funkcí programu, kterou budou všechny trasy v této práci optimalizovány. Výchozí trasu, tedy pořadí měst ze vstupního souboru, je možné vždy opět zobrazit po kliknutí na menu: “Řešení - Výchozí”.

Kapitola 4

Analýza výsledků

Zvolená metoda řešení byla nejdříve pozorována na testovacích datech, kde se pro dané velikosti úloh ukázala jako velmi rychlá a účinná. U všech testovacích souborů nalezla optimální řešení za méně než jednu sekundu, a proto byla následně implementována jako hlavní metoda v programu pro řešení problému obchodního cestujícího. Celkem byly vytvořeným programem optimalizovány tři trasy projektu Revolution Train. Výsledky jsou prezentovány jako výstup programu a zobrazení získané trasy na mapě raildar.fr/osrm/ [12]. Zobrazená cesta nemusí přesně odpovídat nejlepšímu řešení, jelikož se OSRM při hledání trasy snaží o co nejvíce přímou jízdu, tedy bez „nepovolených“ otoček ve stanicích nebo na výhybkách. Při zadání více míst tak občas nevyužívá nejkratší nalezené cesty pro jednotlivé dvojice bodů použité v matici vzdáleností, ale některé delší cesty, umožňující přímé projetí stanic. Vlak je v nich přitom schopný přes noc lokomotivu přepojit na druhou stranu a vrátit se stejnou trasou. Z tohoto důvodu nemusí být celková vizualizace ve všech místech přesná a využívat nejkratší cesty nalezené pro matici vzdáleností. Při přesném plánování trasy vlaku tak budou dohledány jednotlivé úseky zvlášť. To by však zabralo v této práci příliš místa, kde je představa o výsledné podobě trasy důležitější než popis konkrétních tratí a výhybek, kterými vlak pojede.

4.1 Tour podzim 2021

První trasa byla vytvořena optimalizací podzimní tour 2021. Ta dosahovala podle získané matice vzdáleností pro daná města a pořadí jejich navštívení původní délky 4026,66 km. Pořadí měst a délky jednotlivých úseků lze vidět na obrázku 4.1. Po optimalizaci byla nalezena trasa o celkové délce 3492,06 km. Toto řešení tedy dosáhlo snížení ujeté vzdálenosti o 534,6 km, neboli 13,3 % původní délky. Optimalizovaná trasa je zobrazena ve výstupu programu na obrázku 4.2. Poslední obrázek (4.3) pak zobrazuje optimalizovanou trasu na mapě. Vstupní soubor, obsahující města se souřadnicemi pro toto zadání lze nalézt v příloze A. Maticí vzdáleností, specifikující matematický model (str. 14) pro tuto úlohu, pak na přiloženém CD (příloha B).

TSP Solver

Zobrazení Matice Řešení | Celková délka trasy: 4026.659

pořadí	město	vzdálenost z předchozího
1.	1 DEJVICE	
2.	2 ČELÁKOVICE	30,70
3.	3 MILOVICE	13,81
4.	4 KOLÍN	43,48
5.	5 KRALUPY n. VLTAVOU	86,26
6.	6 RAKOVNÍK	66,62
7.	7 OSTROV n. OHŘÍ	127,08
8.	8 BEČOV n. TEPLOU	37,84
9.	9 SCHLEIZ	158,02
10.	10 ANNABERG - BUCHHOLZ	129,99
11.	11 ERFURT	218,83
12.	12 SOMMERDA	25,02
13.	13 APOLDA	57,80
14.	14 ARNSTADT	59,07
15.	15 ILMENAU	27,36
16.	16 NORDERSTEDT	525,00
17.	17 BAD SEGEBERG	94,84
18.	18 BAD BRAMSTEDT	45,13
19.	19 NEUMNUNSTER	22,28
20.	20 BERLIN	361,01
21.	21 POSTDAM	27,30
22.	22 NÁCHOD	408,12
23.	23 ČESKÁ TŘEBOVÁ	85,37
24.	24 MORAVSKÁ TŘEBOVÁ	25,78
25.	25 VELKÉ OPATOVICE	93,54
26.	26 PROSTĚJOV	173,24
27.	27 NOVÝ JIČÍN	96,90
28.	28 TŘINEC	78,92
29.	29 ČADCA	32,14
30.	30 ČESKÝ TĚŠÍN	39,50
31.	31 BOHUMÍN	29,65
32.	32 OSTRAVA - JIH	21,33
33.	33 OSTRAVA	13,64
34.	34 HAVÍŘOV	27,31
35.	35 FRENŠTÁT p. RADHOŠTEM	52,03
36.	36 ROŽNOV p. RADHOŠTEM	37,71
37.	37 VALAŠSKÉ MEZIŘÍČÍ	13,40
38.	38 IVANČICE	163,86
39.	39 POHOŘELICE	50,89
40.	40 DAČICE	175,40
41.	41 TÁBOR	139,69
42.	42 BENEŠOV	52,29
43.	1 DEJVICE	58,49

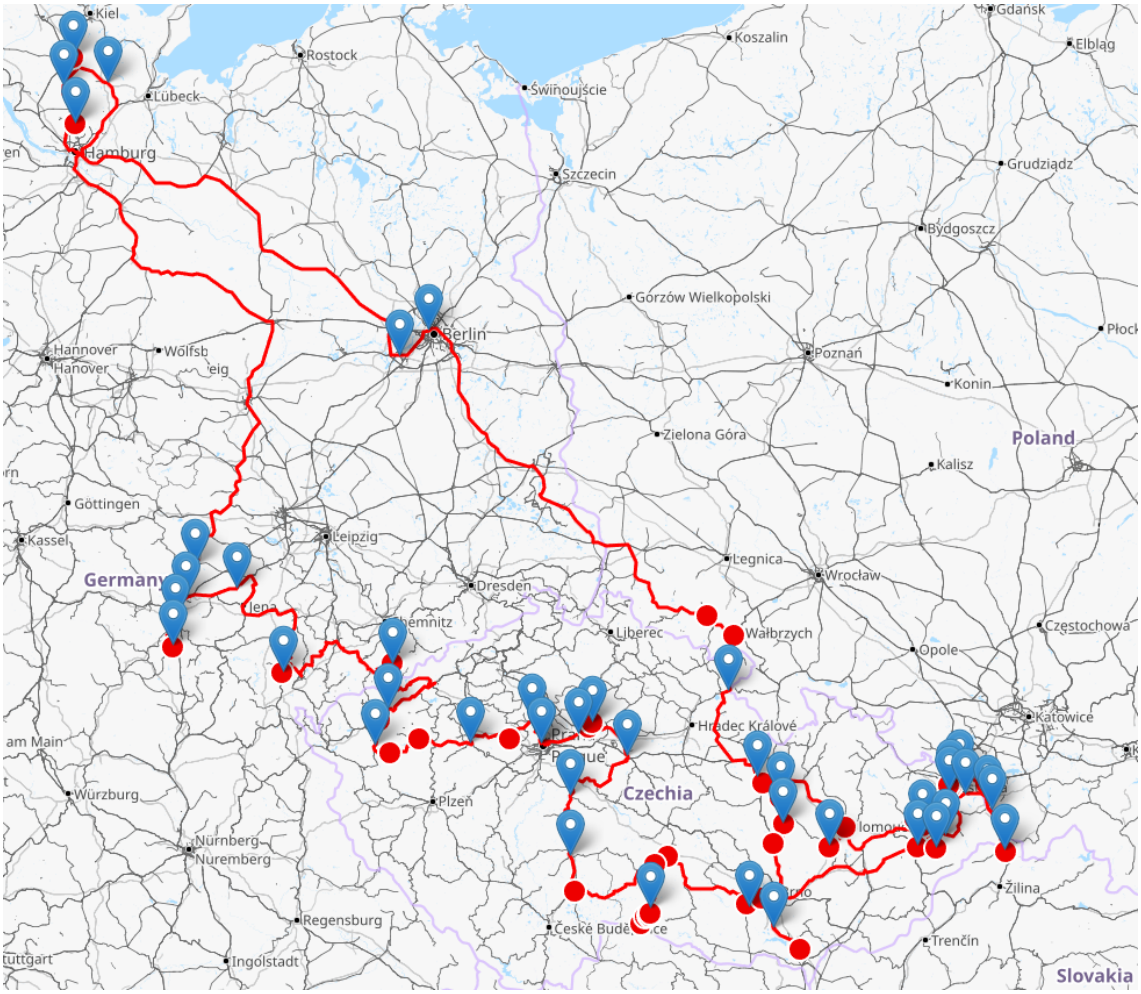
Obrázek 4.1: Původní trasa podzim 2021

TSP Solver

Zobrazení Matice Řešení | Celková délka trasy: 3492.0645

pořadí	město	vzdálenost z předchozího
1.	1 DEJVICE	
2.	5 KRALUPY n. VLTAVOU	36,21
3.	6 RAKOVNÍK	66,62
4.	8 BEČOV n. TEPLOU	87,70
5.	7 OSTROV n. OHŘÍ	37,84
6.	10 ANNABERG - BUCHHOLZ	119,94
7.	9 SCHLEIZ	129,99
8.	13 APOLDA	138,85
9.	14 ARNSTADT	59,07
10.	15 ILMENAU	27,36
11.	11 ERFURT	49,68
12.	12 SOMMERDA	25,02
13.	17 BAD SEGEBERG	444,33
14.	19 NEUMNUNSTER	28,43
15.	18 BAD BRAMSTEDT	22,28
16.	16 NORDERSTEDT	27,15
17.	21 POSTDAM	326,48
18.	20 BERLIN	27,30
19.	22 NÁCHOD	385,06
20.	23 ČESKÁ TŘEBOVÁ	85,37
21.	24 MORAVSKÁ TŘEBOVÁ	25,78
22.	26 PROSTĚJOV	118,90
23.	27 NOVÝ JIČÍN	96,90
24.	33 OSTRAVA	43,54
25.	31 BOHUMÍN	10,91
26.	30 ČESKÝ TĚŠÍN	29,65
27.	28 TŘINEC	7,36
28.	29 ČADCA	32,14
29.	34 HAVÍŘOV	58,48
30.	32 OSTRAVA - JIH	13,67
31.	35 FRENŠTÁT p. RADHOŠTEM	43,16
32.	37 VALAŠSKÉ MEZIŘÍČÍ	25,29
33.	36 ROŽNOV p. RADHOŠTEM	13,40
34.	25 VELKÉ OPATOVICE	197,27
35.	39 POHOŘELICE	87,41
36.	38 IVANČICE	50,89
37.	40 DAČICE	158,01
38.	41 TÁBOR	139,69
39.	42 BENEŠOV	52,29
40.	4 KOLÍN	74,62
41.	3 MILOVICE	43,48
42.	2 ČELÁKOVICE	13,81
43.	1 DEJVICE	30,70

Obrázek 4.2: Optimalizovaná trasa podzim 2021



Obrázek 4.3: Mapa optimalizované trasy podzim 2021

4.2 Tour jaro 2022

Druhá trasa byla získána optimalizací jarní tour 2022. Jelikož se jednalo o otevřenou cestu, kdy vlak kvůli servisu vyjížděl z jiného města, než ve kterém končil, bylo nutné matici vzdáleností po jejím vygenerování pro otevřenou úlohu modifikovat. To bylo provedeno způsobem popsaným v teoretické části práce (str. 15), tedy vytvořením přídatného města, ve výpisu programu označeného jako „*1-31*“. Původní trasa dosahovala dle získané matice délky 3086,7 km (obrázek 4.4). Po optimalizaci bylo nalezeno řešení o délce 2971,3 km (obrázek 4.5), tedy kratší o 115,4 km, nebo 3,7 % původní trasy. Vstupní soubor měst se souřadnicemi se nachází v příloze práce A, modifikovaná matice vzdáleností pak na přiloženém CD (příloha B).

TSP Solver

Zobrazení Matice Řešení | Celková délka trasy: 3086.7002

pořadí	město	vzdálenost z předchozího
1.	1 Ostrava	
2.	2 Studénka	23,50
3.	3 Kopřivnice	17,70
4.	4 Prostějov	115,50
5.	5 Olomouc	20,80
6.	6 Náchod	170,20
7.	7 Červený Kostelec	18,80
8.	8 Přelouč	95,60
9.	9 Čáslav	48,70
10.	10 Nymburk	44,30
11.	11 Čelákovice	23,10
12.	12 Mělník	39,00
13.	13 Praha-Smíchov	54,70
14.	14 Beroun	38,20
15.	15 Hořovice	19,30
16.	16 Rokycany	28,40
17.	17 Klatovy	70,90
18.	18 Strakonice	75,60
19.	19 Rakovník	162,80
20.	20 Chomutov	87,20
21.	21 Jirkov	6,00
22.	22 Klášterec n. Ohří	23,80
23.	23 Muhlhausen	352,90
24.	24 Ankum	349,70
25.	25 Bramsche	21,00
26.	26 Pinneberg	260,70
27.	27 Nordestedt	49,20
28.	28 Itzehoe	70,30
29.	29 Lubeck	130,90
30.	30 Berlin	285,10
31.	31 Dejvice	382,80
32.	*1-31*	0,00
33.	1 Ostrava	0,00

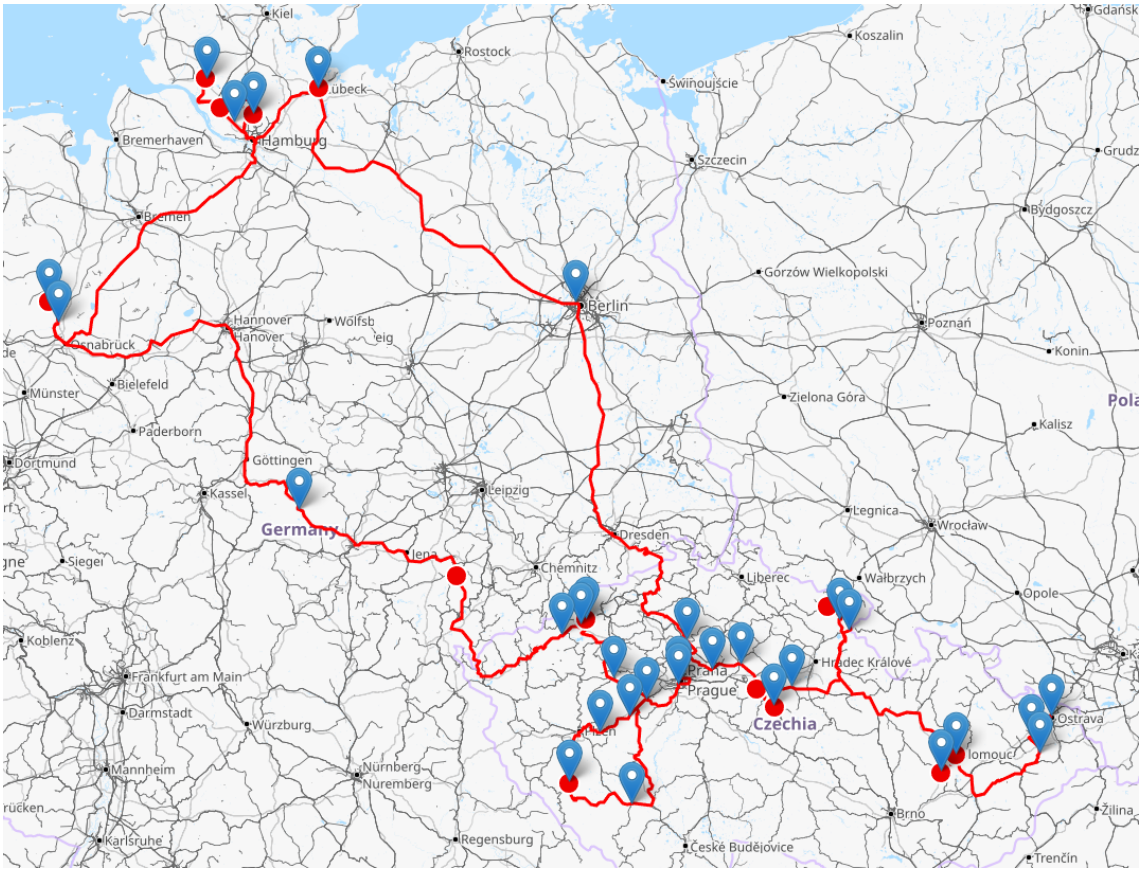
Obrázek 4.4: Původní trasa jaro 2022

TSP Solver

Zobrazení Matice Řešení | Celková délka trasy: 2971.2998

pořadí	město	vzdálenost z předchozího
1.	1 Ostrava	
2.	2 Studénka	23,50
3.	3 Kopřivnice	17,70
4.	4 Prostějov	115,50
5.	5 Olomouc	20,80
6.	7 Červený Kostelec	177,10
7.	6 Náchod	18,80
8.	8 Přelouč	88,60
9.	9 Čáslav	48,70
10.	10 Nymburk	44,30
11.	11 Čelákovice	23,10
12.	12 Mělník	39,00
13.	30 Berlin	328,20
14.	29 Lubeck	285,10
15.	27 Nordestedt	101,00
16.	28 Itzehoe	70,30
17.	26 Pinneberg	49,00
18.	24 Ankum	281,60
19.	25 Bramsche	21,00
20.	23 Muhlhausen	328,70
21.	22 Klášterec n. Ohří	352,90
22.	21 Jirkov	23,80
23.	20 Chomutov	6,00
24.	19 Rakovník	87,20
25.	15 Hořovice	62,00
26.	16 Rokycany	28,40
27.	17 Klatovy	70,90
28.	18 Strakonice	75,60
29.	14 Beroun	120,10
30.	13 Praha-Smíchov	38,20
31.	31 Dejvice	24,20
32.	*1-31*	0,00
33.	1 Ostrava	0,00

Obrázek 4.5: Optimalizovaná trasa jaro 2022



Obrázek 4.6: Mapa optimalizované trasy jaro 2022

4.3 Tour podzim 2022

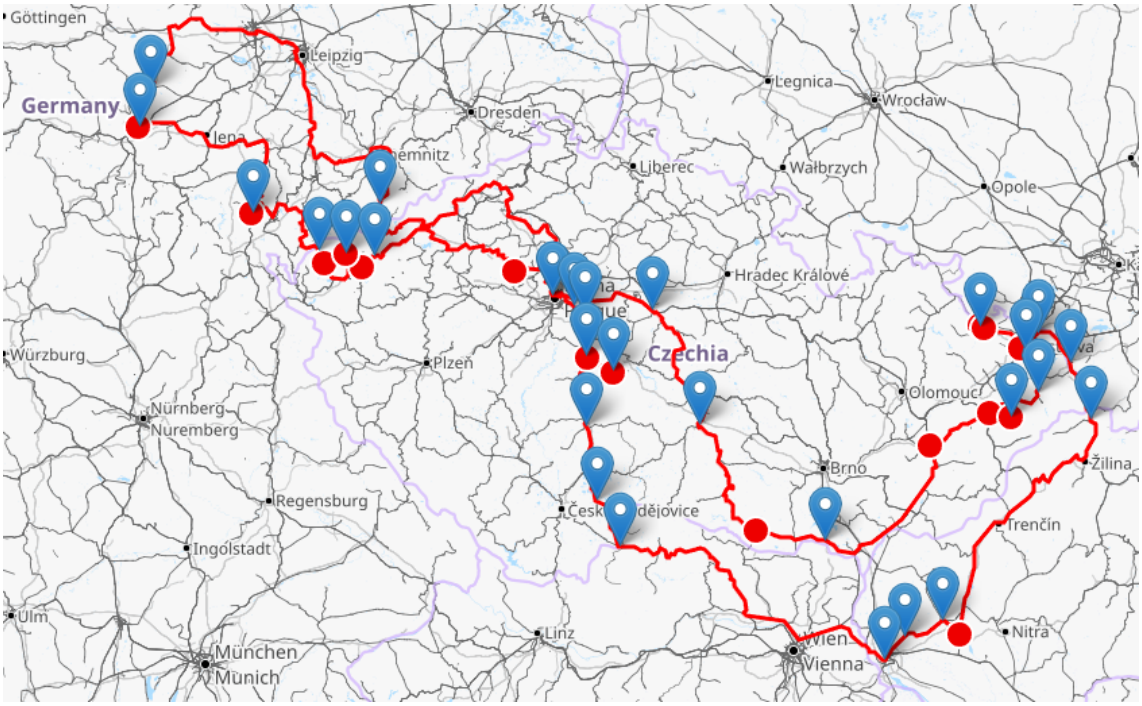
Na závěr byl program využit pro plánování trasy projektu na podzimní tour 2022. Jelikož se seznam měst může stále rozšířit, tato trasa nemusí být finální. Dále bude po zkompletování seznamu měst a předběžném výběru trasy potřeba cestu zkonzultovat se zástupcem Českých drah pro případ možných výluk a dalších okolností, které mohou výslednou trasu dále ovlivnit. V aktuální podobě (k 25.6.2022) navržená trasa navštěvuje celkem 28 měst a dosahuje celkové délky 2372,94 km. Podobu tohoto řešení lze vidět v přiloženém výstupu programu na obrázku 4.7. Vizualizace této trasy pak na mapě na obrázku 4.8.

TSP Solver

Zobrazení Matice Řešení | Celková délka trasy: 2372.94

pořadí	město	vzdálenost z předchozího
1.	Dejvice	
2.	Uhříněves	20,32
3.	Říčany	8,06
4.	Benešov	30,12
5.	Vlašim	22,84
6.	Tábor	73,63
7.	Třeboň	47,76
8.	České Velenice	33,72
9.	Bratislava	220,10
10.	Pezinok	19,00
11.	Trnava	26,80
12.	Čadca	187,94
13.	Český Těšín	39,50
14.	Bohumín	29,65
15.	Opava	44,57
16.	Vítkovice	38,51
17.	Frýdlant n. Ostravicí	28,57
18.	Rožnov p. Radhoštěm	52,30
19.	Mikulov	165,85
20.	Jihlava	143,10
21.	Kolín	99,16
22.	Annaberg-Buchholz	271,01
23.	Sommerda	239,87
24.	Erfurt	25,02
25.	Schleiz	142,42
26.	Kraslice	89,11
27.	Nejdek	57,95
28.	Ostrov n. Ohří	31,98
29.	Dejvice	184,09

Obrázek 4.7: Aktuální podoba plánované trasy



Obrázek 4.8: Mapa navržené trasy

Závěr

V teoretické části práce byly shrnuty základní poznatky k problému obchodního cestujícího. Byl zde představen matematický model úlohy a její grafické zobrazení. Dále byla popsána modifikace problému pro otevřenou cestu a způsob jejího převedení na základní úlohu, kterého bylo využito v praktické části práce. Na závěr kapitoly byl popsán 2-opt algoritmus, jehož implementace byla použita v programu pro řešení praktické úlohy.

Za pomoci programovacího jazyka Java, jeho knihoven a API OSRM Raildar byl vytvořen program pro optimalizaci tras projektu Revolution Train. Ten umožnuje nalézt vstupní matici vzdáleností na železniční síti pro potřeby vlakové soupravy. Jsou zde však i další možnosti jejího zadání, které umožňují například řešení modifikací problému.

Pro samotnou optimalizaci byla otestována metoda optimalizující náhodné výchozí trasy 2-opt algoritmem. Ta byla schopná nalézt optimální řešení vybraných úloh, velikostí podobných trasám projektu revolution train, s velkou jistotou už po jedné vteřině. Proto byla tato metoda zvolena pro hledání řešení problému a implementována do programu pro následné optimalizace.

Napsaný program byl využit pro optimalizaci celkem tří tras projektu. První dvě byly vytvořeny pro města navštívená při již uskutečněných tour. Byla zde vypozorována úspora zvolené metody vůči původnímu plánování 13,3 % resp 3,7 % délky původní trasy. Na závěr byla navržena trasa projektu pro plánovanou tour projektu na podzim roku 2022, v aktuální podobě procházející dvaceti osmi městy a dosahující délky 2372,94 km.

Ačkoliv byla použitá metoda schopna nalézt optimální řešení ve všech provedených testech s vysokou jistotou, nebylo v této práci nijak prokázáno, že je to ve stejném čase možné pro všechny úlohy podobné velikosti. Zastoupení nejlepšího řešení se pro testovací soubory podstatně lišilo a je tak možné, že existují problémy podobné velikosti, u kterých by bylo potřeba optimalizovat mnohem více tras pro jeho nalezení. Žádné studie zabývající se touto metodou jsem nenalezl a mohly by tak být do budoucna vítaným přínosem k tomuto tématu.

Literatura

[1] AGARWALA, R., D. L. APPLEGATE, D. MAGLOTT, G. D. SCHULER a A. A. SCHÄFFER. A Fast and Scalable Radiation Hybrid Map Construction and Integration Strategy. *Genome Research*. 2000, **10**(3), 350-364. ISSN 1088-9051. Dostupné z: doi:10.1101/gr.10.3.350

[2] BAILEY, C., T. MCLAIN a R. BEARD. Fuel saving strategies for separated spacecraft interferometry. *AIAA Guidance, Navigation, and Control Conference and Exhibit*. Reston, Virigina: American Institute of Aeronautics and Astronautics, 2000, 2000-08-14, -. ISBN 978-1-62410-301-8. Dostupné z: doi:10.2514/6.2000-4441

[3] BEKTAS, T. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*. 2006, **34**(3), 209-219. ISSN 03050483. Dostupné z: doi:10.1016/j.omega.2004.10.004

[4] CROES, G. A. A Method for Solving Traveling-Salesman Problems. *Operations Research*. 1958, **6**(6), 791-812. ISSN 0030-364X. Dostupné z: doi:10.1287/opre.6.6.791

[5] IntelliJ Idea Community Edition. JetBrains [online]. [cit. 2022-06-30]. Dostupné z: https://www.jetbrains.com/idea/

[6] LINGO. Lindo Systems Inc. [online]. [cit. 2022-06-25]. Dostupné z: https://www.lindo.com/index.php/products/lingo-and-optimization-modeling

[7] LIU, H., R. LIU, Z. LIU a S. XU. Minimizing the Number of Transitions of 3D Printing Nozzles Using a Traveling-Salesman-Problem Optimization Model. *International Journal of Precision Engineering and Manufacturing*. 2021, **22**(9), 1617-1637. ISSN 2234-7593. Dostupné z: doi:10.1007/s12541-021-00512-2

[8] MILLER, C. E., A. W. TUCKER a R. A. ZEMLIN. Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM*. 1960, **7**(4), 326-329. ISSN 0004-5411. Dostupné z: doi:10.1145/321043.321046

[9] ONWUBOLU, G. C. a M. CLERC. Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization. *International Journal of Production Research*. 2007, **42**(3), 473-491. ISSN 0020-7543. Dostupné z: doi:10.1080/00207540310001614150

[10] *Paint* [online]. Microsoft [cit. 2022-06-25]. Dostupné z: https://apps.microsoft.com/store/detail/paint/9PCFS5B6T72H?hl=en-us&gl=US

[11] Raildar OSRM. *Raildar.fr* [online]. [cit. 2022-06-25]. Dostupné z: http://wiki.raildar.fr/index.php/Instances_OSRM

[12] Raildar OSRM visualization. *Raildar.fr* [online]. [cit. 2022-06-25]. Dostupné z: http://raildar.fr/osrm/osrm.html

[13] REINELT, G. TSPLIB-A Traveling Salesman Problem Library. *ORSA Journal on Computing*. 1991, **3**(4), 376-384. ISSN 0899-1499. Dostupné z: doi:10.1287/ijoc.3.4.376

[14] Traveling Salesman Problem [online]. Kanada: University of Waterloo [cit. 2022-06-25]. Dostupné z: http://www.math.uwaterloo.ca/tsp/index.html

Příloha A

Vstupní soubory optimalizací

A.1 tour_podzim2022.txt

```
Dejvice                 50.096895   14.399557
Uhříněves               50.041238   14.577098
Říčany                  49.997481   14.663798
Annaberg-Buchholz       50.579503   12.998022
Schleiz                 50.528461   11.960785
Sommerda                51.164961   11.127509
Erfurt                  50.972008   11.036689
Kraslice                50.330594   12.502736
Nejdek                  50.319066   12.727586
Ostrov n. Ohří          50.301676   12.957859
Kolín                   50.025332   15.214863
Rožnov p. Radhoštěm     49.460510   18.134608
Frýdlant n. Ostravicí   49.588890   18.354635
Vítkovice               49.801378   18.261681
Opava                   49.939124   17.887990
Bohumín                 49.900999   18.359533
Český Těšín             49.743591   18.622727
Čadca                   49.444887   18.787372
Trnava                  48.367783   17.580721
Pezinok                 48.282536   17.270524
Bratislava              48.158961   17.106630
Mikulov                 48.801988   16.625662
Jihlava                 49.416578   15.598687
Třeboň                  49.015279   14.760700
České Velenice          48.769428   14.957500
Tábor                   49.414324   14.676876
Vlašim                  49.700555   14.897343
Benešov                 49.779823   14.682734
```

A.2 tour_podzim2021.txt

1	DEJVICE	50.096895	14.399557
2	ČELÁKOVICE	50.158061	14.754311
3	MILOVICE	50.229226	14.881861
4	KOLÍN	50.025332	15.214863
5	KRALUPY n. VLTAVOU	50.237544	14.317353
6	RAKOVNÍK	50.099097	13.736182
7	OSTROV n. OHŘÍ	50.301676	12.957859
8	BEČOV n. TEPLOU	50.083045	12.831334
9	SCHLEIZ	50.528461	11.960785
10	ANNABERG - BUCHHOLZ	50.579503	12.998022
11	ERFURT	50.972008	11.036689
12	SOMMERDA	51.164961	11.127509
13	APOLDA	51.030923	11.525946
14	ARNSTADT	50.843671	10.946825
15	ILMENAU	50.685712	10.924455
16	NORDERSTEDT	53.708551	9.992291
17	BAD SEGEBERG	53.933953	10.307901
18	BAD BRAMSTEDT	53.920472	9.890093
19	NEUMNUNSTER	54.076305	9.980189
20	BERLIN	52.535412	13.338561
21	POSTDAM	52.391499	13.067250
22	NÁCHOD	50.417741	16.173409
23	ČESKÁ TŘEBOVÁ	49.896846	16.446619
24	MORAVSKÁ TŘEBOVÁ	49.767462	16.666260
25	VELKÉ OPATOVICE	49.611266	16.691011
26	PROSTĚJOV	49.471865	17.129536
27	NOVÝ JIČÍN	49.598848	18.008732
28	TŘINEC	49.687414	18.661598
29	ČADCA	49.444887	18.787372
30	ČESKÝ TĚŠÍN	49.743591	18.622727
31	BOHUMÍN	49.900999	18.359533
32	OSTRAVA - JIH	49.801378	18.261681
33	OSTRAVA	49.842881	18.274641
34	HAVÍŘOV	49.791682	18.412056
35	FRENŠTÁT p. RADHOŠTEM	49.541954	18.223969
36	ROŽNOV p. RADHOŠTEM	49.460510	18.134608
37	VALAŠSKÉ MEZIŘÍČÍ	49.474894	17.960994
38	IVANČICE	49.098248	16.372719
39	POHOŘELICE	48.965075	16.603968
40	DAČICE	49.085890	15.439160
41	TÁBOR	49.414324	14.676876
42	BENEŠOV	49.779823	14.682734

A.3 tour_jaro2022.txt

```
1       Ostrava             49.842881   18.274641
2       Studénka            49.707501   18.066416
3       Kopřivnice          49.595914   18.147697
4       Prostějov           49.471865   17.129536
5       Olomouc             49.592214   17.278587
6       Náchod              50.417741   16.173409
7       Červený Kostelec    50.483569   16.075369
8       Přelouč             50.039598   15.574558
9       Čáslav              49.915233   15.394807
10      Nymburk             50.193337   15.045900
11      Čelákovice          50.158061   14.754311
12      Mělník              50.353412   14.491723
13      Praha-Smíchov       50.060562   14.408087
14      Beroun              49.957288   14.076941
15      Hořovice            49.843345   13.899196
16      Rokycany            49.740096   13.592041
17      Klatovy             49.401200   13.272879
18      Strakonice          49.255468   13.917832
19      Rakovník            50.099097   13.736182
20      Chomutov            50.456111   13.397398
21      Jirkov              50.492668   13.448811
22      Klášterec n. Ohří   50.385264   13.195632
23      Muhlhausen          51.209290   10.473833
24      Ankum               52.542048   7.879305
25      Bramsche            52.411634   7.974615
26      Pinneberg           53.655321   9.797240
27      Nordestedt          53.708551   9.992291
28      Itzehoe             53.924863   9.509461
29      Lubeck              53.867232   10.668712
30      Berlin              52.535412   13.338561
31      Dejvice             50.096895   14.399557
```

Příloha B

Přílohy na CD

Zdrojové kódy programu

- Stations.java
- Window.java
- Main.java

Matice vzdáleností použitá jako vstup optimalizace trasy 2

- matrix2.txt

Matice vzdáleností specifikující matematický model tras 1 a 3

- matrix1.txt
- matrix3.txt

Testovací soubory

- att48.txt
- berlin52.txt
- swiss42.txt