ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA EKONOMICKÁ

Bakalářská práce

Implementace jednodušších heuristik pro řešení specifických variant rozvozních problémů

Implementation of simpler heuristics to solve some specific variants of delivery problems

Marek Křevký

Plzeň 2022

Čestné prohlášení

Prohlašuji, že jsem bakalářskou práci na téma

„Implementace jednodušších heuristik pro řešení specifických variant rozvozních problémů“

vypracoval samostatně pod odborným dohledem vedoucího bakalářské práce za použití pramenů uvedených v přiložené bibliografii.

Plzeň dne ……..

Podpis autora

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce prof. Dr. Ing. Miroslavu Plevnému za odborné vedení práce.

Obsah

Úvod

Doručování zboží zákazníkům je velkým tématem u každé výrobní či obchodní firmy. Je to naprosto běžná část podnikové dopravy. Pokud se trasy nastavují špatně, může to být pro firmu naprosto zbytečný náklad navíc.

Cílem této bakalářské práce je vytvořit aplikaci za použití online map, kde bude moci uživatel zadat adresy a webová aplikace mu zobrazí trasu za použití heuristik.

Dílčí cíle pro splnění práce budou:

- formulování dopravního problému,
- specifikování variant okružních jízd,
- vytvoření funkční aplikace,
- testování aplikace.

Teoretická část bude zaměřena na základy logistiky a samotných rozvozních problému. Budeme definovat známější metody pro řešení okružních jízd a také heuristiky, které budeme později aplikovat do aplikace.

V praktické části bude ukázáno, jak aplikace funguje, jak ji obsluhovat. Bude také ukázáno porovnání aplikace s ručním výpočtem heuristiky.

V závěru budou shrnuty poznatky z praktické části, porovnání a poznámky k dalšímu vylepšení aplikace.

1 Vymezení základních pojmů

1.1 Logistika

Pro logistiku existuje hned několik definicí. Jednou z nich je například: „Logistika se zabývá pohybem zboží a materiálů z místa vzniku do místa potřeby a s tím souvisejícím informačním tokem.“ (Drahotský & Řezníček, 2003, s. 1). Jiné definice se liší v detailech, nicméně podstata zůstává vždy stejná. Víceméně je logistika souhrn takových nevýrobních činností, aby správné zboží v požadované kvalitě a kvantitě, bylo k dispozici na požadovaném místě, ve správném čase a to vše za správnou cenu, tedy všechny tyto činnosti s minimálními náklady. (Daněk & Plevný, 2005)

Slovo logistika použil poprvé v roce 1838 švýcarský generál Antoine-Henri Jomini v jeho díle Précis de l'Art de la Guerre což v češtině ve volném překladu znamená „Přehled umění války (Jomini, 1838). Principy logistiky se totiž využívají ve vojenském prostředí, kde se jedná o přesun vojska, munice atd.

1.2 Doprava

Doprava je nedílnou a důležitou součástí logistiky. Definovat ji lze podle Sixta a Mačáta takto: „Doprava je záměrná pohybová činnost, která spočívá v přemístění věcí nebo osob prostřednictvím pohybu dopravních prostředků po dopravních cestách.“ (Sixta a Mačát, 2005, s. 161). Hlavním úkolem dopravy je přesun zboží v prostoru, tzv. přesun z místa vzniku až do místa spotřeby - zákazníkům.

Dopravu můžeme rozdělit například na vnitropodnikovou a mimopodnikovou. Vnitropodniková doprava je přeprava materiálu či zboží uvnitř podniku. Mimopodniková doprava je naproti tomu realizována při přepravě zboží od dodavatele do podniku a poté z podniku k zákazníkovi.

Nejrozšířenější doprava je doprava silniční která je flexibilní díky hustotě silniční sítě. Dalo by se říci, že nejvíce vyhovuje požadavkům zákazníků a proto její využití neustále roste.

Druhá nejvíce používaná doprava v České republice je doprava železniční. Její síť je pevně dána železničními tratěmi a její výhodu je, že je levnější než doprava silniční nebo letecká.

Jako třetí typ dopravy zde uvedu dopravu leteckou. Jedná se o nejdražší druh dopravy a využívá se hlavně u zboží s vysokou hodnotou nebo samozřejmě u zboží, které je potřeba vzhledem k velké vzdálenosti přepravit velmi rychle.

Pro přepravu objemného zboží nebo zboží o velkém množství se používá doprava lodní. Tu musí dovolit zeměpisné podmínky.

V praxi se hojně využívá kombinovaná doprava využívající právě více kombinací doprav. (Sixta & Mačát, 2005)

1.3 Teorie grafů

Pro potřebu této práce zde uvedu základní pojmy z teorie grafů.

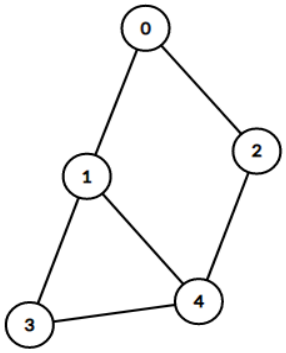
Graf je definován množinou vrcholů (uzlů), které představují jednotlivé místa (například města) a množinou hran (cest). Vrcholy jsou mezi sebou propojeny hranami kdy jedna hrana vždy propojuje právě 2 vrcholy.

Matematicky můžeme psát graf G = (V, H), kde:

- $V = \{u_1, u_2, \ldots, u_n\}$ je konečná, neprázdná množina vrcholů
- $H = \{h_{ij}\}$ je množina hran, přičemž $h_{ij} = (u_i, u_j)$, kde $u_i, u_j \in V$

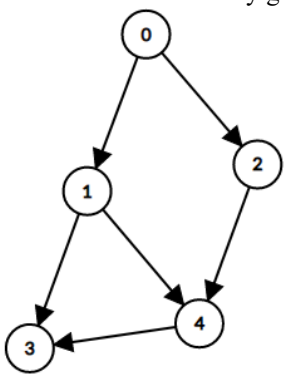
Neorientovaný graf je graf jehož hrany jsou definovány jako neuspořádané dvojice vrcholů, tj. $(u_i, u_j) = (u_j, u_i)$, nezáleží tedy na pořadí v jakém jsou vrcholy uvedeny. Naopak graf jehož hrany jsou definovány jako uspořádané dvojice vrcholů, tj. $(u_i, u_j) \neq (u_j, u_i)$ se nazývá **orientovaný graf**.

Obrázek 1: Neorientovaný graf



Zdroj: vlastní zpracování, 2022

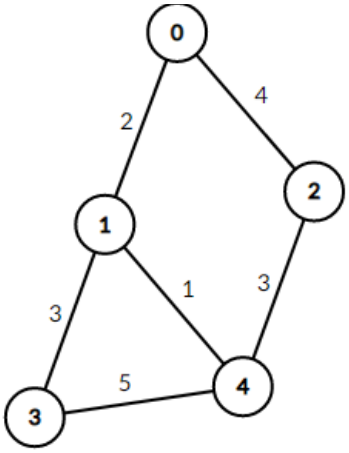
Obrázek 2: Orientovaný graf



Zdroj: vlastní zpracování, 2022

Pokud je hranám přidělená nějaká číselná hodnota mluvíme o tzv. **hranově ohodnoceném** grafu. Přiřazená číselná hodnota může vyjadřovat například vzdálenost v kilometrech.

Obrázek 3: Hranově ohodnocený graf



Zdroj: vlastní zpracování, 2022

Sled v grafu G je posloupností vrcholů, které jsou vzájemně spojeny hranami. V případě ohodnoceného grafu lze určit délku sledu jako součet ohodnocení jednotlivých hran.

Cesta mezi vrcholem u_{i0} a u_{ik} představuje sled začínající ve vrcholu u_{i0} a končící v u_{ik}, v němž se každý vrchol vyskytuje pouze jednou.

Kružnice je souvislý neorientovaný graf, v němž z každého vrcholu vychází právě dvě hrany.

Hamiltonovská cesta je definována jako cesta, která obsahuje všechny vrcholy v grafu, ale nemusí obsahovat všechny hrany grafu. Hamiltonovská kružnice je uzavřená Hamiltonovská cesta. (Plevný & Žižka, 2010)

2 Rozvozní problém

S problematikou rozvozních problémů se setkal alespoň nepřímo každý z nás. Ať už nakupujete v místním maloobchodě, přišel vám dopis od České pošty, popeláři vyvezli vaše popelnice nebo jste si objednali jídlo domů z restaurace, která má vlastní rozvoz. Někdo musel přivézt čerstvý chléb do obchodu, pošťák musel doručit váš dopis do schránky a určitě při tom využili předem naplánovanou trasu. Ve velmi zjednodušeném případě si můžeme rozvozní problém představit i jako dopravu dětí do škol. Můžete každé ráno rozvážet Vaše děti do školy resp. školky, a každé dítě přitom navštěvuje jiné zařízení. Musíte si potom naplánovat trasu a vlastně vyřešit velmi zjednodušený rozvozní problém. Toto je jen pár příkladů, které alespoň nějakým minimálním způsobem tuto problematiku řeší.

Pojem se poprvé objevil už v roce 1959 v článku od autorů George Dantzig a John Ramser. V tomto článku byl sepsán první algoritmus pro řešení problému a ten byl později aplikován na dodávky benzínu. (Dantzig & Ramser, 1959)

V roce 1964 Clarke a Wright zlepšili přístup Dantziga a Ramseryho použitím efektivního hladového algoritmu, který v každém kroku hledá lokální maximum resp. minimum přičemž existuje šance, že takto nalezne i globální maximum respektive minimum a tím pádem i optimální řešení. V angličtině se nazývá „savings algorithm“, volně přeloženo do češtiny jako algoritmus úspor. Tato metoda bude více vysvětlena v další kapitole. (Clarke & Wright, 1964)

Problém je většinou spjat s depem a zákazníky, kteří si objednávají určitý počet jednotek nějakého zboží a my je musíme z depa (například skladu) obsloužit.

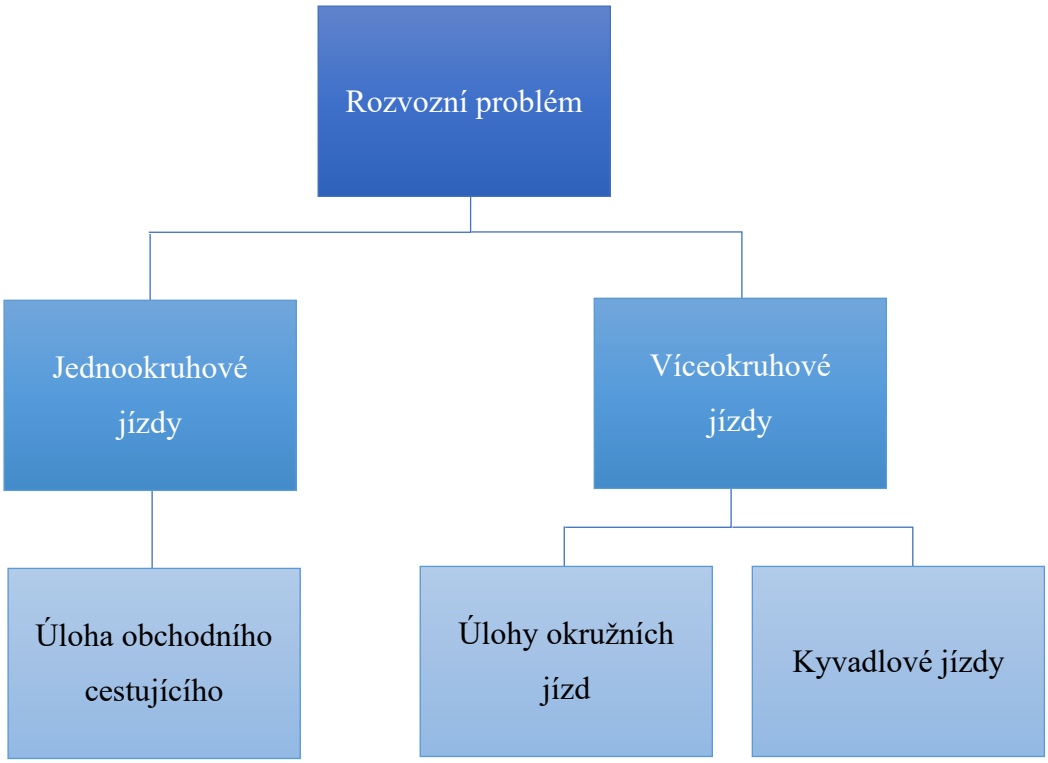
Úlohy klasifikujeme jako NP-těžké. Ve stručnosti to znamená, že s přibývajícími zákazníky výpočetní složitost úlohy exponenciálně roste a proto nelze objemné úlohy optimálně vyřešit v rozumném čase. (Toth & Vigo, 2002)

Rozvozní úlohy můžeme dále dělit na statické či dynamické. Při řešení statických úloh máme od zákazníka všechny informace ještě před řešením úlohy. Naopak v případě dynamických úloh nejsou všechny informace známé a zákazník nám je poskytuje v průběhu. (Fiala, 2010)

3 Klasifikace rozvozních problémů

Rozvozní problém můžeme klasifikovat podle následujícího obrázku.

Obrázek 4: Klasifikace rozvozního problému



Zdroj: vlastní zpracování, 2022

3.1 Jednookruhové jízdy

Úloha, která se zabývá sestavením jedné trasy, se označuje jako **úloha obchodního cestujícího**.

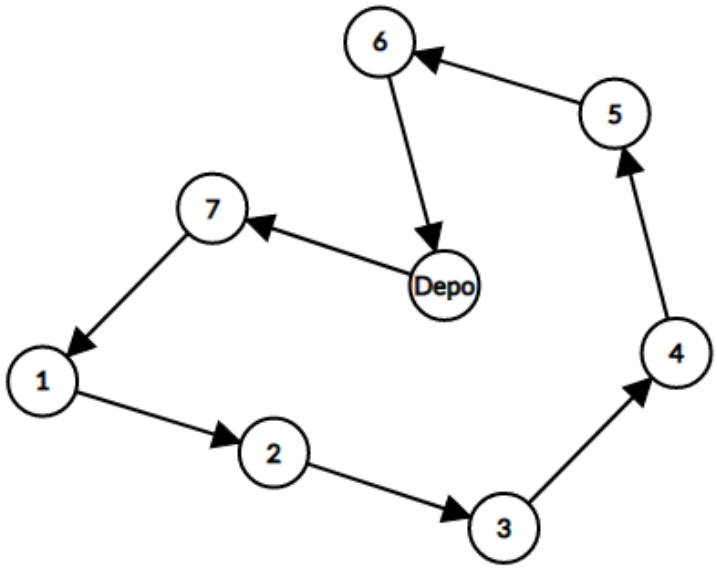
TSP (Travelling Salesman Problem) je známý kombinatorický problém, kdy obchodní cestující musí navštívit každé z množiny daných měst přesně jednou a vrátit se do místa, odkud vyrazil. (Jablonský, 2007)

Podstatou úlohy je nalézt nejkratší sled v grafu, který obsahuje všechny vrcholy grafu (všechna odběrná místa). Z hlediska teorie grafů jde o určení uzavřené cesty neboli Hamiltonovské kružnice. (Plevný & Žižka, 2010)

Začátek trasy je dán umístěním depa, z kterého se postupuje do dalších vrcholů do doby, než budou navštíveny všechny. Celková vzdálenost je dána součtem ohodnocení

hran. Pro optimální řešení musí být minimální a každý vrchol musí být navštíven právě jednou.

Obrázek 5: Problém obchodního cestujícího



Zdroj: vlastní zpracování, 2022

Matematický model TSP

Řídící proměnná: $x_{ij} = \begin{cases} 1 & vozidlo\ pojede\ z\ místa\ i\ do\ místa\ j \\ 0 & jinak \end{cases}$ (1)

Účelová funkce: $min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij}$ (2)

Za podmínek:

$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \qquad j = 1, \dots, n;$ (3)

$\sum_{j=1, i \neq j}^{n} x_{ij} = 1 \qquad i = 1, \dots, n;$ (4)

$u_i - u_j + (n-1)x_{ij} \leq n - 2 \qquad 2 \leq i \neq j \leq n;$ (5)

$1 \leq u_i \leq n - 1 \qquad 2 \leq i \leq n;$ (6)

$x_{ij} \in \{0, 1\} \qquad i, j = 1, \dots, n;$ (7)

$u_i \in \mathbb{Z} \qquad i = 2, \dots, n.$ (8)

Kde n je počet míst, které musíme navštívit, c_{ij} vzdálenost mezi místem i a j, x_{ij} binární proměnná (1 pokud vozidlo pojede z místa i do místa j a 0 pokud nikoliv) a proměnná u_i, která udává pořadí, v kterém jsou místa navštívena.

Podmínky (3) a (4) zajišťují, že každé město bude navštíveno právě jednou a podmínky (5) a (6) vynucují, aby existovala pouze jedna trasa zahrnující všechna města, a nikoli dvě nebo více oddělených tras, které společně zahrnují všechna města. Podmínky (7) a (8) jsou obligatorní. (Fiala, 2010)

3.2 Víceokruhové jízdy

Z reálného hlediska obvykle není možné uspokojit požadavky všech zákazníků jedním dopravním prostředkem. Víceokruhové jízdy jsou rozšířením klasické jednookruhové jízdy o podmínky, které znemožňují obsloužit všechny zákazníky pouze jedním okruhem. Důsledkem těchto podmínek je vznik více okruhů. Nejčastěji se jedná o kapacitní nebo časové podmínky. Víceokruhové jízdy se dále dělí na okružní a kyvadlové jízdy. (Brožová & Houška, 2008)

3.2.1 Kyvadlové jízdy

Kyvadlové jízdy se uskutečňují jednorázově nebo opakovaně mezi místem nakládky a místem vykládky, protože požadavky jednoho zákazníka jsou stejné nebo přesahují kapacitu vozu. Dopravní prostředek tak podniká krátké cesty sem a tam. (Fiala, 2010)

3.2.2 Úlohy okružních jízdy

V angličtině se úlohy okružních jízd označují jako Vehicle Routing Problems (VRP). Jedná se o úlohy, v nichž se řeší rozvoz zásilek z centra k jednotlivým spotřebitelům, přičemž požadavky jednotlivého zákazníka nezaplňují celou kapacitu nákladního vozidla, což umožňuje obsloužit více zákazníků během jedné jízdy.

Cílem je najít trasu, která je co nejkratší a uspokojí každý požadavek jedinou obsluhou. Ve skutečnosti se množství obsloužených míst může denně měnit, proto je velmi důležité mít na tento problém rychlou odpověď.

V praxi nás omezuje velké množství faktorů. Můžeme být omezeni kapacitou našich vozidel nebo potřebujeme být u jednotlivých zákazníků v určitý časový interval, atd. (Toth & Vigo, 2002)

Matematický model VRP

Mějme graf G = (V; H), kde V = (u_1, … , u_n) je množina zákazníků a H je množina všech hran. Depo je umístěno do uzlu u_0. Matice D je maticí vzdálenosti a udává vzdálenost mezi jednotlivými uzly. K dispozici máme množinu vozidel K o kapacitě C. (Toth & Vigo, 2002)

Řídící proměnná: $x_{ijk} = \begin{cases} 1 & vozidlo\ k\ pojede\ z\ místa\ i\ do\ místa\ j \\ 0 & jinak \end{cases}$ (9)

Účelová funkce: $min \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} \cdot x_{ijk}$ (10)

Za podmínek:

$\sum_{i \in V} x_{ijk} = \sum_{i \in V} x_{jik} \qquad \forall j \in V \cup \{u_0\};\ k \in K, kde\ i \neq j;$ (11)

$\sum_{k \in K} \sum_{i \in V} x_{ijk} = 1 \qquad \forall j \in V, kde\ i \neq j;$ (12)

$\sum_{j \in V} d_j \sum_{i \in V} x_{ijk} \leq C_k \qquad \forall k \in K, kde\ i \neq j;$ (13)

$\sum_{j \in S} \sum_{i \in S} x_{ijk} \leq |S| - 1 \quad \forall k \in K, S \subseteq V, |S| \geq 2, kde\ i \neq j;$ (14)

$x_{ijk} \in \{0,1\} \qquad \forall k \in K, i \in V \cup \{u_0\}, j \in V \cup \{u_0\}, kde\ i \neq j;$ (15)

Řídící proměnná je stejně jako v případě TSP binární. Nabývá hodnoty 1, pokud vozidlo k pojede z místa i do místa j. Účelová funkce je také opět minimalizační, kde d_{ij} popisuje vzdálenost mezi místem i a j. Počet vozidel označujeme písmenem p a počet míst písmenem n.

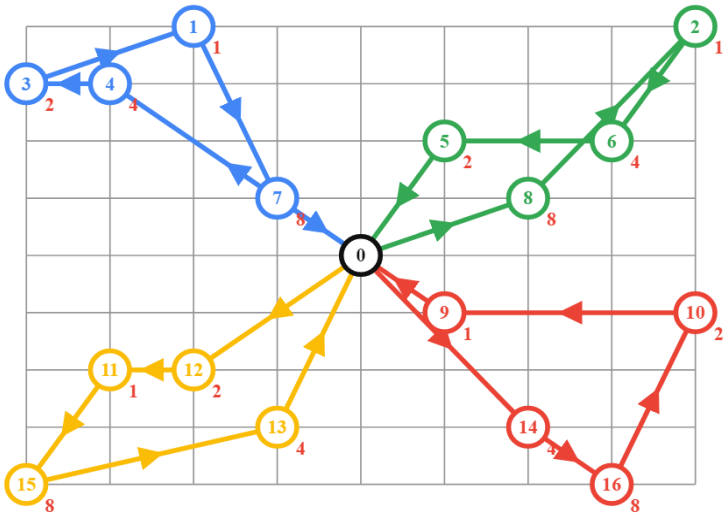
Podmínka (11) nám zajišťuje skutečnost, že vozidlo které navštíví místo tak toto místo i opustí. Každé místo musí být navštíveno právě jednou. To nám zajišťuje výraz (12). Podmínka (13) říká, že nesmíme překročit kapacity vozidel. Podmínka (14) udává povinnost tras tvořit uzavřený sled uzlů, kde se žádný nich neopakuje. Dále nám zajišťuje, že každá trasa bude procházet depem. Podmínka (15) je obligatorní. (Toth & Vigo, 2002)

3.2.2.1 Varianty úloh okružních jízd

Kapacitně omezená úloha okružních jízd (Capacited Vehicle Routing Problem - CVRP)

Tento typ úlohy právě respektuje kapacitní omezení dopravního prostředku. Dále platí všechny podmínky a cíle klasického okružního problému. Jedná se o jednu z nejvíce řešených variant VRP. (Iory, Gonzalez & Vigo, 2005)

Obrázek 6: CVRP

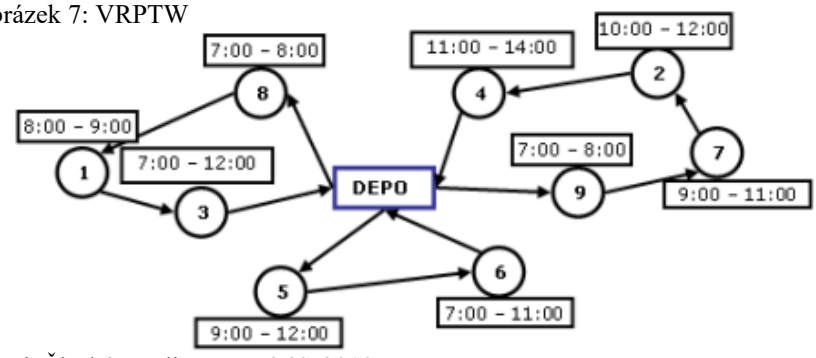


Zdroj: google.com, 2022

Úloha okružních jízd s časovými okny (Vehicle Routing Problem with Time Windows - VRPTW)

Tato modifikace VRP přidává časová okna kdy je potřeba zákazníka obsloužit. Jedná se o základní modifikaci a v reálném světě se víceméně nesetkáme se zákazníkem kterého můžeme obsloužit v libovolném čase. Kombinací CVRP a VRPTW se už blížíme reálnému světu. (Iory, Gonzalez & Vigo, 2005)

Obrázek 7: VRPTW

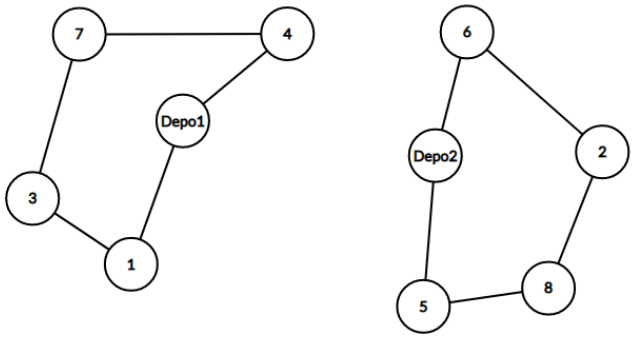


Zdroj: Široký & Slivoně, s. 264, 2010

Úloha okružních jízd s více středisky (Multiple Depot Vehicle Routing Problem - MDVRP)

I touto modifikací se blíže přibližujeme k realitě. V té totiž můžeme mít více dep (skladů) s vlastní flotilou vozidel. Zákazníci jsou obsluhováni několika depy, a vozy se vrací do stejného depa odkud vyrazily. Tato varianta souvisí s distribuční technologií Hub&Spoke, která právě uvažuje více skladů. Technologii využívá například Česká pošta, která převáží velké množství zásilek do svých středisek a odtud až k jednotlivým zákazníkům. (Iory, Gonzalez & Vigo, 2005)

Obrázek 8: MDVRP



Zdroj: vlastní zpracování

Úloha okružních jízd pro delší období (Periodic Vehicle Routing Problem - PVRP)

V klasických úlohách se typicky plánují jízdy na jeden den. V PVRP tomu tak není a počet dní může být obecně *M*. To tedy znamená, že vozidlo, které odjede ze střediska jeden den se ten samý den nemůže vrátit. (Iory, Gonzalez & Vigo, 2005)

Úloha okružních jízd s rozdělenou dodávkou (Split Delivery Vehicle Routing Problem – SDVRP)

U této varianty uvažujeme, že jeden vrchol může být navštíven vícekrát například z důvodu omezení počtu vozidel, které můžeme použít. Lehce se může v praxi stát, že máme k dispozici dvě vozidla a do každého se vejdou 3 palety. Úkolem je obsloužit 3 zákazníky A, B, C, kteří chtějí po 2 paletách. Obě auta naplníme maximálním počtem palet tj. 3. První auto vyšleme k zákazníkovi A předáme mu 2 palety. Poté první auto pošleme k zákazníkovi B a předáme mu jen jednu paletu. Druhé auto plně obslouží zákazníka C a zákazníkovi B doveze zbylou paletu. Zákazník B byl tedy navštíven celkem dvakrát. (Iory, Gonzalez & Vigo, 2005)

Stochastická úloha okružních jízd (Stochastic Vehicle Routing Problem - SVRP)

Stochastická úloha okružních jízd upravuje zadání tak, že alespoň jedna složka se řídí náhodným rozdělením. Může jít například o čas obsluhy, dobu cesty nebo poptávka zákazníka a další. (Shen, Desouky & Ordonez, 2007)

Úloha okružních jízd s nakládkou a vykládkou (Vehicle Routing Problem with Pick-up and Delivering - VRPPD)

V této úloze se uvažuje o možnosti, že je možné na straně zákazníka zboží vyložit a naložit. Je přitom třeba vzít v úvahu, že zboží naložené na straně zákazníka nesmí překročit kapacitu vozidla. Naložené zboží pak může být dodáno jinému zákazníkovi nebo převezeno zpět do skladu. (Toth & Vigo, 2002)

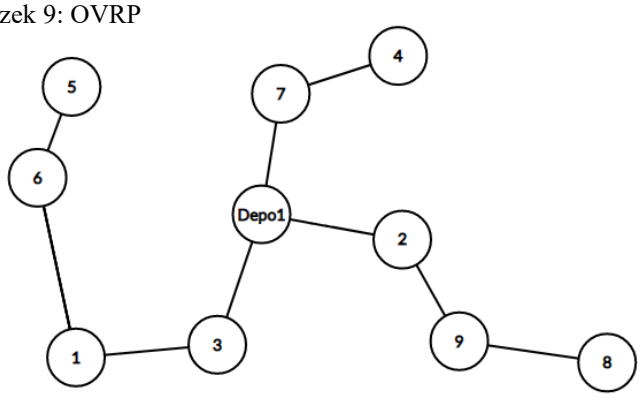
Úloha okružních jízd se zpětným sběrem (Vehicle Routing Problem with Backhauls - VRPB)

Jedná se o podobný případ jako VRPPD, ale rozdíl je v tom, že vozidlo musí být nejdříve prázdné a až poté můžeme začít s nakládáním zboží u zákazníků. (Toth & Vigo, 2002)

Otevřená úloha okružních jízd (Open Vehicle Routing Problem – OVRP)

V této úloze se vozidlo nevrací zpět do depa. Jinak platí všechny podmínky jako u klasické úlohy. Dále můžeme přidávat různá omezení jako například kapacitní, časová okna atd.

Obrázek 9: OVRP



Zdroj: vlastní zpracování

4 Metody pro řešení úloh okružních jízd

Metod a algoritmů, pro řešení okružního problému existuje celá řada. Při jejich výběru se potřebujeme zamyslet a vybrat tu která nejvíce vyhovuje naší situaci. V implementaci můžeme metody i například kombinovat abychom došli k nám požadovaným výstupům.

4.1 Exaktní metody

Protože je VRP NP-těžký problém, čas potřebný k nalezení optimálního řešení roste s rostoucím počtem bodů potřebných navštívit exponenciálně. Proto u mnoha skutečných a testovacích úloh, nejsou optimální řešení vůbec známa. K jejich nalezení je nutné použít některé z exaktních algoritmů, který je však pro větší instance VRP velmi problematický z hlediska výpočetního čas. (Laporte, 1991)

4.1.1 Zpětné prohledávání (Backtracking)

Postupně se prohledává celý prostor možných řešení. Každé vyhovující řešení si zapamatuje a po prohledání celého prostoru je nejlepší řešení označeno jako optimální.

Zpětné vyhledávání funguje na principu prohledávání do hloubky. Nejprve se vybere základní řešení a pak se od konce vrací zpět na začátek. Při návratu se všechny ostatní možnosti přezkoumávají. Tato metoda je velmi neefektivní a sama o sobě se k řešení VRP nepoužívá. Je však základ pro další algoritmy, které mohou optimalizovat jeho výpočetní nároky. (Laporte, 1991)

4.1.2 Metoda větví a hranic (Branch and bound)

Tento přístup rozděluje strom všech možných řešení do podstromů. Vyhledávání se provádí jednou z dalších metod, jako je backtracking. Rozdíl mezi touto metodou a zpětným prohledáváním je vynechání podstromů, pro které neexistuje optimální řešení.

Metoda se skládá ze dvou částí, dělení stromu na podstromy a zpracování konkrétního podstromu. Důležitým parametrem pro efektivitu výpočtu je počáteční nastavení hodnoty pro nejlepší řešení. To může být nastaveno jako extrém hodnotící funkce. Mnohem efektivnější přístup je použít hodnotu již známého suboptimálního řešení. To lze získat buď náhodně nebo výpočtem pomocí aproximačních metod.

Algoritmus na začátku rozděluje prostor možných řešení na podstromy. V dalším kroku určí pro daný podstrom hraniční hodnotu, která je lepší než hodnota všech řešení nacházejících se v podstromu. Pokud je taková hraniční hodnota horší než hodnota dosud nejlepšího známého řešení, daný podstrom je vyloučen z prohledávání.

Pokud se během určování hraniční hodnoty narazí na funkční řešení, které je lepší než dosud nejlepší řešení, zapamatuje se místo předchozího. Po určení hraničních hodnot pro všechny podstromy je vybrán podstrom s nejlepší hraniční hodnotou a dochází k jeho rozdělení na ještě menší podstromy.

Tento postup se opakuje, dokud nezůstanou žádné podstromy k prohledávání. Řešení, které je v té chvíli nejlepší a je prohlášeno za optimální. (Laporte, 1991)

4.1.3 Lineární programování

VRP lze jak již bylo naznačeno přepsat do matematického modelu v podobě několika soustav rovnic a nerovnic s několika proměnnými. Model je potřeba sestavit co nejmenší, aby se zkrátila doba výpočtu. Pro řešení úlohy lineárního programování existuje mnoho algoritmů. Jedním z nich je například simplexová metoda. (Laporte, 1991)

4.2 Zlepšovací algoritmy

Tyto algoritmy pracují s již existujícími řešeními VRP a vylepšují jej. Tyto algoritmy jsou často používány jako součást meta-heuristik. Pomocí určených postupů hledají lepší řešení v prostoru v okolí aktuálního řešení.

4.2.1 Znič a vytvoř (Ruin and Recreate)

Tato metoda je založena na zničení části řešení a vybudování nového. Metoda začíná s kompletním řešením VRP, ze kterého postupně odstraní jedno nebo více míst a přidá je do nezařazených míst. Tehdy nastupuje některá z heuristik a ta nezařazená místa zpracuje svým algoritmem. (Salaj, 2007)

4.3 Heuristiky

Heuristika prohledává pouze malou část všech řešení. Díky tomu jsou výrazně méně náročné na výpočetní složitost. Narozdíl od exaktních algoritmů, heuristiky často neprohledávají oblasti, kde se nacházejí kvalitní řešení. Z tohoto důvodu se na výsledná řešení často aplikují ještě další zpřesňující algoritmy.

Heuristiky pro VRP lze rozdělit do tří kategorií: konstruktivní, dvoufázové a vkládací heuristiky. (Salaj, 2007)

4.3.1 Konstruktivní heuristiky

Nejprve vytvoří cesty pro každého zákazníka nebo skupinu zákazníků a poté tyto cesty propojí. Zohledňují celkové náklady na řešení a dodržování všech podmínek a omezení. (Salaj, 2007)

4.3.1.1 Algoritmus sekvenčního vkládání (Sequential Insertion algorithm)

Algoritmus sekvenčního vkládání vytvoří přípustné řešení, které je dostupnou množinou tras při opakovaného zahrnování zákazníků, kteří se neobjevili v žádné dočasné aktuální trase. (Salaj, 2007)

4.3.1.2 Nearest Neighbor heuristic

Algoritmus nejbližšího souseda funguje tak, že hledáme právě nejbližšího souseda od místa, v kterém se momentálně nacházíme. Nejbližší místo zařadíme do trasy a opět hledáme nejbližšího souseda od tohoto místa. Při umístění zákazníka do trasy, aktualizujeme seznam míst, které jsou potřeba obsloužit. Při překročení kapacitní podmínky vozidla se trasa ukončí návratem zpět do depa, odkud se bude hledat jeho další nejbližší soused ze zbývajících míst. (Salaj, 2007)

4.3.1.3 Clark & Wright Savings algorithm

Slučování cest v tomto algoritmu je realizováno na základě úspor, které sloučení přinese. Výpočet probíhá ve 2 krocích, které se opakují. Konec nastane, když není možné spojit žádné 2 cesty tak, aby nebyly porušeny zadané podmínky. Těmi může být například překročení kapacity vozidel.

V prvním kroku se vypočítají úspory pro všechny dvojice zákazníků. Druhý krok se liší pro sekvenční a paralelní verzi algoritmu. U sekvenčního se nejprve vybuduje jedna celá trasa, a teprve když ji nelze dále prodloužit, přejde se na další. Při paralelním přístupu se vybere největší úspora a dané cesty se propojí. Rozšiřuje se tedy více cest najednou, což dává této verzi lepší výsledky než sekvenční verzi. (Salaj, 2007)

4.3.2 Dvoufázové heuristiky

VRP můžeme rozdělit na 2 podproblémy. Prvním podproblémem je přiřazení míst, která jsou třeba navštívit jednotlivým vozidlům a druhým je plánování samotné trasy. Existují dva typy algoritmů.

4.3.2.1 Cluster first – route second

Už název napovídá jak algoritmy fungují. Nejdříve se vytvářejí shluky míst tzv. clustery a ty se přiřazují vozidlům. Pro jednotlivé clustery se pak řeší úloha obchodního cestujícího. (Salaj, 2007)

Sweep algoritmus

Ze skladu je postupně dokola opisována polopřímka. Lze si ji představit jako hodinovou ručičku. Místa, přes které přejíždí jsou připisována k témuž vozidlu až do chvíle než dojde k porušení podmínek kapacity vozidla. Poté jsou místa připisována novému vozidlu. Pro každé vozidlo je řešena úloha obchodního cestujícího.

4.3.2.2 Route first – cluster second

Víceméně opak pro algoritmy cluster first – route second. Nejdříve je vytvořena jedna velká trasa a ta je poté rozdělena tak, aby byly splněné zadané podmínky. (Salaj, 2007)

4.3.3 Vkládací heuristiky

Vkládací heuristiky pracují na principu vybrání ještě nezařazeného místa a jeho následného vložení do trasy. Existují celkem dvě verze. Sekvenční nejprve vytvoří jednu kompletní trasu a teprve potom se přechází na další. Paralelní verze vytváří najednou několik tras.

V první fázi algoritmu dochází k výběru nezařazeného místa podle zvolených kritérií. Vybírá se buď nejbližší nebo nejvzdálenější místo. Samozřejmě u různých variant VRP se mohou kritéria lišit. Ve druhé fázi je vybrané místo vloženo do některé z tras, případně je založena nová. Vloženo je tak, aby se celková délka tras zvětšila co nejméně.

4.4 Meta-heuristiky

Rozdíl mezi meta-heuristikami a heuristikami je ve velikosti prostoru, který je prohledáván. Meta-heuristiky přinášejí často lepší výsledky díky tomu, že prohledávají větší prostor a zaměřují se na nejslibnější oblasti. Na druhé straně se tím zvedají i výpočetní nároky, které jsou stále podstatně menší než nároky exaktních metod.

Mezi meta-heuristiky patří například genetické algoritmy. Tyto algoritmy napodobují vývoj živých organismů. Jednotlivá řešení jsou zakódována do chromozomů, které jsou obvykle ukládány jako posloupnost bitů nebo čísel.

Tabu-procházení patří mezi metody lokálního prohledávání. Využívá však paměť, aby se dokázala vyhnout cyklování v lokálním minimu. Výpočet začíná v nějakém náhodně zvoleném řešení. Pomocí lokálních změn vybírá nejlepší řešení z množiny sousedů. Tam patří všechna řešení, kterých lze dosáhnout provedením dovolených změn a posunů z aktuálního řešení. Algoritmus končí když aktuální řešení splňuje zadané parametry nebo bylo dosaženo povoleného počtu kroků. (Salaj, 2007)

5 Implementace aplikace

5.1 API pro práci s mapou

Google API

Velmi propracované a obsáhlé API. Pro jeho použití si je potřeba nastavit platební údaje, protože využití tohoto API může být zpoplatněno. Zaznamenávají se jednotlivé požadavky, které pošleme na server. Využití je zpoplatněno podle toho, kolik požadavků za měsíc pošleme. V současné době dostane každý uživatel $200 volný rozpočet pro tvoření mapových aplikací.

Služba nabízí spoustu funkcí. Pro cíle této práce by byly potřeba například funkce pro zobrazení mapy, získání matice vzdálenosti mezi zadanými místy a geokódování (proces převedení adresy do souřadnic nebo naopak – reverzní geokódování) . Pro všechny tři tyto funkce existují různé ceníky.

Obrázek 10: Ceník pro zobrazení mapy

MONTHLY VOLUME RANGE (Price per MAP LOAD)		
0–100,000	100,001–500,000	500,000+
0.007 USD per each (7.00 USD per 1000)	0.0056 USD per each (5.60 USD per 1000)	Contact Sales for volume pricing

Zdroj: developers.google.com

Obrázek 11: Ceník pro získání matice vzdálenosti

MONTHLY VOLUME RANGE (Price per ELEMENT)		
0–100,000	100,001–500,000	500,000+
0.005 USD per each (5.00 USD per 1000)	0.004 USD per each (4.00 USD per 1000)	Contact Sales for volume pricing

Obrázek 12: Ceník pro geokódování

MONTHLY VOLUME RANGE (Price per REQUEST)		
0–100,000	100,001–500,000	500,000+
0.005 USD per each (5.00 USD per 1000)	0.004 USD per each (4.00 USD per 1000)	Contact Sales for volume pricing

Zdroj: developers.google.com

Mapy.cz API

České mapy od společnosti Seznam.cz také nabízejí svojí knihovnu pro tvorbu webových aplikací s mapami. Knihovna je napsaná v jazyce JavaScript a má přehlednou dokumentaci i s ukázkami jejího použití. Pro osobní účely je využití zcela zdarma.

Knihovna má spoustu funkcí:

- Zobrazení mapy,
- tvorba a úprava značek,
- plánování trasy,
- našeptávač,
- geokódování,
- reverzní geokódování,
- další.

Všechny tyto funkce lze použít při vývoji aplikace, která řeší úlohy okružních jízd.

Výběr API

Knihoven pro práci s mapami je ještě daleko více. Uvedu například Mapbox, OpenStreetMap, HERE a další. Pro implementaci aplikace byla vybraná API Mapy.cz. Má strukturovanou a přehlednou dokumentaci a spoustu funkcí. Možnou nevýhodou tohoto výběru je, že se omezujeme na implementaci aplikace v jazyce JavaScript. Program poběží ve webových prohlížečích, ale všechny výpočty proběhnou na straně klienta. Se serverem aplikace komunikovat nebude (vyjma komunikace ze strany API).

5.2 Vývoj aplikace

Jelikož je zvolená knihovna funkcí napsaná v jazyce JavaScript, je v ní vyvíjená i vlastní aplikace.

Před samostatným řešením úloh okružních jízd je potřeba připravit uživatelsky vhodné prostřední a ovládací prvky aplikace.

Zobrazení mapy

Instancí třídy *SMap* dostaneme objekt mapy kterou zobrazujeme v aplikaci. Mapě přidáváme ovládací prvky pro umožnění pohybu po mapě a zoomu pomocí kolečka myši.

Značky

Knihovní třída *SMap.Marker*

Zobrazování značek na mapě je jedna z funkcí API. Značky jsou přizpůsobitelné a můžeme jim přidat dynamické vlastnosti, jako je například „přesouvatelnost“.

Značka se vytvoří po odchycení události kliknutí na mapu. Vytvoření značky znamená tyto 3 operace:

1. Samotné vytvoření značky na mapě
    - Z kliknutí na mapu získáme souřadnice kliknutí použijeme společně s knihovní funkci pro reverzní geokódování.
2. Vytvoření záznamu v tabulce
3. Přidání uzlu do pole uzlů

Značkám můžeme taky přidávat vlastní popis. To pro nás bude výhodné při určování tras.

Aplikace nabízí mazání jednotlivých značek. Zde nastává problém s číslováním. Máme 10 značek očíslovaných 1 – 10 a smažeme značku číslo 6. Jenom při samostatném odstranění značky by nám v tabulce i na mapě zůstaly značky očíslované jako 1, 2, 3, 4, 5, 7, 8, 9, 10. Skript tedy řeší přes jednoduchou funkci opravu číslování při mazání značek.

Našeptávač

Knihovní třída SMap.Suggest

Instanci třídy získáme zavoláním příslušného konstruktoru, kterému předáme textové pole, na který má našeptávač reagovat. Po potvrzení vyhledávání přesuneme pozici v mapě na konkrétní lokalitu. Potvrzení lze provést kliknutím na tlačítko nebo výběrem možnosti z nabízeného seznamu lokalit.

Import uzlů

Je řešen přes textové pole. Rozdělíme vstup na řádky a ty postupně procházíme a tvoříme uzly, značky a záznamy v tabulce. Pokud jsou zadány adresy, používáme funkci pro geokódování a pokud souřadnice tak reverzní geokódování.

Tady nás webové rozhraní pochopitelně omezuje v tom, že nemůžeme vytvořit velké množství uzlů najednou. Kdybychom chtěli vytvořit například 20 míst znamenalo by to poslání 20ti požadavků na server během pár milisekund. Ten by takové požadavky zamítl a nám by se uzly nevytvořily. V programu je to vyřešeno tak, že se po každém požadavku počká 250 milisekund.

Export uzlů

Probíhá naplněním textového pole pomocí informací uložených o jednotlivých uzlech.

Řešení úloh okružních jízd

Program řeší kapacitně omezené úlohy okružních jízd (CVRP) a to dvěma způsoby. První způsob je přes algoritmus úspor a druhý přes metodu nejbližšího souseda. Před samotným řešením konkrétních algoritmů je společným krokem je získání všech tras a vytvoření matice vzdálenosti.

Získání matice vzdálenosti

Funkce, která matici vrací je z jednoho důvodu problematická. Jde o podobný důvod jako u tvorby několika značek najednou. Pro získání vzdálenosti mezi jednotlivými uzly používáme knihovní třídu *SMap.Route*, konkrétně její statickou metodu *SMap.Route.route,* které předáme souřadnice a volitelné nastavení pro trasu, například požadavek pro nejkratší trasu. Když bychom měli 21 uzlů a budeme řešit symetrickou

úlohu, celkový počet požadavků by se vypočítal jako $\frac{21*20}{2} = 210$. Pokud bychom poslali požadavky na server najednou, určitě jich pár zamítne a my nedostaneme odpověď.

Statická metoda *SMap.Route.route* je asynchronní. Ve stručnosti to znamená, že provedení této metody nemusí projít v pořadí, v jakém máme kód napsaný, ale třeba až na konci programu. Ve skutečnosti toho můžeme využít, jelikož při práci s asynchronní metodou máme k dispozici klíčové slovo *await*, které zajistí, že nepošleme další požadavek do doby, než získáme odpověď na požadavek současný.

Tím zajistíme, že dostaneme výsledky na každý požadavek. Značně se ale může prodloužit čekání na odpovědi.

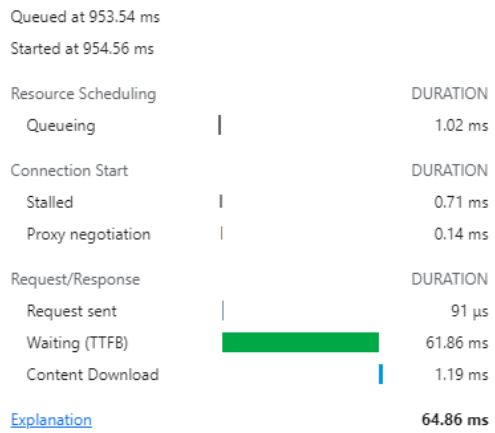
Obrázek 13: Posílání požadavků

route	200	xhr	smap-jak.js?v=0.5292633259067727:...	7.8 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	14.1 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	12.4 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	8.8 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	4.7 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	12.4 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	14.3 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	12.0 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	5.9 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	12.1 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	15.5 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	20.7 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	19.1 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	12.8 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	8.4 kB
route	200	xhr	smap-jak.js?v=0.5292633259067727:...	11.0 kB

Zdroj: Google DevTools

Na obrázku č. 14 si můžeme prohlédnout část poslaných požadavků na server. Pro nás jsou důležitější informace na obrázku č. 15. Zde vidíme informace, o tom jak dlouho jsme čekali na odpověď serveru na konkrétní požadavek. V průměru získáme odpověď

Obrázek 14: Doba čekání na odpověď od serveru



Zdroj: Google DevTools

za cca 60 ms, ale občas se může stát, že nás server „zmrazí“ a my získáme výsledky třeba až za 2-3 sekundy. Takové navýšení je docela problematické, a když to nastane 20 krát, celková doba na získání matice vzdálenosti bude přibližně 1 minuta.

Implementace heuristik

1. Metoda nejbližšího souseda
    - V poli si uchováváme zákazníky které jsme dosud nenavštívili.
    - Následuje cyklus č. 1, který iteruje tak dlouho, dokud nebude výše zmíněné pole prázdné
    - V proměnné si držíme aktuální uzel v kterém se nacházíme
    - Ve vnořeném cyklu č. 2 nalezneme nejbližšího souseda a pokusíme se ho přidat do trasy
    - V případě úspěchu se přesuneme do přidaného uzlu, který odstraníme z pole nenavštívených uzlů
    - Když neuspějeme, uzel přidat nelze, protože bychom překročili kapacitu vozidla. Ukončíme tedy současnou trasu a nastavíme aktuální uzel na depo

2. Clark Wrightova metoda
    - Před tvorbou tras je potřeba získat matici úspor. Vzorec (17) pro výpočet úspor je uveden na s. 23.
    - Získané úspory uložíme do pole a seřadíme je sestupně.
    - Do řešení vložíme elementární trasy, které vedou z depa k zákazníkovi a zpět
    - Následuje cyklus, který iteruje přes pole úspor
    - V cyklu hledáme indexy tras, v kterých se objeví uzly

6 Testování programu a porovnání heuristik

Předmětem testování aplikace bude zkouška, jestli naimplementované metody pro řešení VRP opravdu počítají jak mají. Zjistíme to tak, že provedeme kontrolní výpočet pro konkrétní zadání a porovnáme výsledky s výstupem programu.

Porovnání heuristik bude provedeno na několika sadách zadání a porovnání jejich výsledků.

6.1 Testování programu

Zadání je dáno tabulkou č. 1.

Tabulka 1: Zadání první úlohy

Adresa	Označení	Omezení a požadavky
Mikoláše Alše 253, Kolín, 280 02, Kolín	DEPO	4
Jiřího z Poděbrad 288/13, Kutná Hora, 284 01, Kutná Hora	Z1	1
Jiráskova 727, Jičín, 506 01, Jičín	Z2	1
Palackého 150, Trutnov, 541 01, Trutnov	Z3	1
Suvorovova 394, Český Brod, 282 01, Kolín	Z4	1
Pavlíkova 1520, Benešov, 256 01, Benešov	Z5	1
Milheimova 2856, Pardubice, 530 02, Pardubice	Z6	1
Lužická 2114/6, Mělník, 276 01, Mělník	Z7	1
Zámecká 500, Litomyšl, 570 01, Svitavy	Z8	1
Dvořákova 603, Vamberk, 517 54, Rychnov nad Kněžnou	Z9	1
Špreňarova 1322, Náchod, 547 01, Náchod	Z10	1

Zdroj: vlastní zpracování, 2022

Společným krokem je získání matice vzdálenosti. Tu získám zadáním lokalit do map a naplánováním trasy.

Tabulka 2: Matice vzdálenosti první úlohy, vlastní zpracování

Matice vzdálenosti											
km	DEPO	1	2	3	4	5	6	7	8	9	10
DEPO	0	12,6	54,9	101,9	25,5	59,4	48,7	74,2	98,9	93,9	95
1	12,6	0	64,8	102,7	38,2	59,9	41,6	86,7	90,8	86,8	95,9
2	54,9	64,8	0	50,4	63,6	105,8	67,8	71,7	101,9	83,7	66,3
3	101,9	102,7	50,4	0	111,6	153,8	69,9	121,6	97,1	65,4	32
4	25,5	38,2	63,6	111,6	0	41,9	75,3	46,8	124,5	113	114,1
5	59,4	59,9	105,8	153,8	41,9	0	101,1	74,6	145,2	146,3	155,9
6	48,7	41,6	67,8	69,9	75,3	101,1	0	115,7	51,5	45,7	60,2
7	74,2	86,7	71,7	121,6	46,8	74,6	115,7	0	163,9	152,4	138,2
8	98,9	90,8	101,9	97,1	124,5	145,2	51,5	163,9	0	37,1	73,9
9	93,9	86,8	83,7	65,9	113	146,3	45,7	152,4	37,1	0	37,8
10	95	95,9	66,3	32	114,1	155,9	60,2	138,2	73,9	37,8	0

Zdroj: vlastní zpracování, 2022

6.1.1 Výpočet pomocí Clarke & Wrightovy metody za pomocí Excelu

Výpočet matice úspor

Úspory se vypočítají podle vzorce:

$$v_{ij} = d_{i0} + d_{0j} - d_{ij} \tag{17}$$

kde:

v_{ij} … *úspora kterou získáme přidáním cesty z i do j*

d_{i0} … *vzdálenost mezi depem a i – tým zákazníkem*

d_{0j} … *vzdálenost mezi depem a j – tým zákazníkem*

d_{ij} … *vzdálenost mezi depem a i – tým a j – tým zákazníkem*

Tabulka 3: Matice úspor první úlohy, vlastní zpracování

Matice úspor										
	1	2	3	4	5	6	7	8	9	10
1	0	2,7	11,8	-0,1	12,1	19,7	0,1	20,7	19,7	11,7
2		0	106,4	16,8	8,5	35,8	57,4	51,9	65,1	83,6
3			0	15,8	7,5	80,7	54,5	103,7	130,4	164,9
4				0	43	-1,1	52,9	-0,1	6,4	6,4
5					0	7	59	13,1	7	-1,5
6						0	7,2	96,1	96,9	83,5
7							0	9,2	15,7	31
8								0	155,7	120
9									0	151,1
10										0

Zdroj: vlastní zpracování, 2022

Jelikož je matice vzdálenosti symetrická podle diagonály, bude matice úspor také symetrická. Z tohoto důvodu nám stačí pracovat pouze s půlkou této matice.

Nadefinování elementárního řešení

Elementární, neboli počáteční řešení získáme tak, že ke každému zákazníkovi uděláme zvlášť trasu, která povede z depa k zákazníkovi a odtud hned zpět do depa.

Tabulka 4: Elementární řešení první úlohy, vlastní zpracování

Průběžně řešení	Vzdálenost [km]	Náklad
DEPO -> 1 -> DEPO	25,2	1
DEPO -> 2 -> DEPO	109,8	1
DEPO -> 3 -> DEPO	203,8	1
DEPO -> 4 -> DEPO	51	1
DEPO -> 5 -> DEPO	118,8	1
DEPO -> 6 -> DEPO	97,4	1
DEPO -> 7 -> DEPO	148,4	1
DEPO -> 8 -> DEPO	197,8	1
DEPO -> 9 -> DEPO	187,8	1
DEPO -> 10 -> DEPO	190	1

Zdroj: vlastní zpracování, 2022

Zbytek řešení

Nalezneme největší hodnotu z matice úspor a pokusíme se sloučit trasy, které obsahují dané zákazníky.

Největší úsporu získáme přidáním trasy [3 – 10]. Můžeme provést sloučení a získat tak řešení po prvním kroku.

Tabulka 5: První úloha, první průběžné řešení

Průběžně řešení	Vzdálenost [km]	Náklad
DEPO -> 1 -> DEPO	25,2	1
DEPO -> 2 -> DEPO	109,8	1
DEPO -> 3 -> 10 -> DEPO	228,9	2
DEPO -> 4 -> DEPO	51	1
DEPO -> 5 -> DEPO	118,8	1
DEPO -> 6 -> DEPO	97,4	1
DEPO -> 7 -> DEPO	148,4	1
DEPO -> 8 -> DEPO	197,8	1
DEPO -> 9 -> DEPO	187,8	1

Zdroj: vlastní zpracování, 2022

Druhou největší úsporu získáme přidáním trasy [8 – 9]. K sloučení nám opět nic nebrání.

Tabulka 6: První úloha, druhé průběžné řešení

Průběžně řešení	Vzdálenost [km]	Náklad
DEPO -> 1 -> DEPO	25,2	1
DEPO -> 2 -> DEPO	109,8	1
DEPO -> 3 -> 10 -> DEPO	228,9	2
DEPO -> 4 -> DEPO	51	1
DEPO -> 5 -> DEPO	118,8	1
DEPO -> 6 -> DEPO	97,4	1
DEPO -> 7 -> DEPO	148,4	1
DEPO -> 8 -> 9 -> DEPO	229,9	2

Zdroj: vlastní zpracování, 2022

Další přidaná trasa by měla být [9 – 10]. To by znamenalo sloučit trasu [DEPO – 3 – 10 – DEPO] s trasou [DEPO – 8 – 9 – DEPO]. To dosáhneme tím, že poslední trasu obrátíme, získáme tím tedy trasu [DEPO – 9 – 8 – DEPO] a tu už můžeme jednoduše sloučit s první zmiňovanou trasou. Získáme tedy následující průběžné řešení.

K třetí trase v tabulce č. 7 už nebudeme moct přidávat další zastávky, protože bychom určitě překročili kapacitní omezení vozidla.

Tabulka 7: První úloha, třetí průběžné řešení

Průběžně řešení	Vzdálenost [km]	Náklad
DEPO -> 1 -> DEPO	25,2	1
DEPO -> 2 -> DEPO	109,8	1
DEPO -> 3 -> 10 -> 9 -> 8 -> DEPO	307,7	4
DEPO -> 4 -> DEPO	51	1
DEPO -> 5 -> DEPO	118,8	1
DEPO -> 6 -> DEPO	97,4	1
DEPO -> 7 -> DEPO	148,4	1

Zdroj: vlastní zpracování, 2022

Další úspora má hodnotu 130,4 a jedná se o cestu [3 – 9]. Do řešení ji ale nejsme schopni přidat, protože oba dva uzly už existují ve stejné trase. Stejné platí pro následující úsporu a její cestu [8 – 10], pokračujeme tedy dále s vybráním další úspory v pořadí. Jedná se o úsporu z cesty [2 – 3]. Přidáním této trasy bychom museli sloučit druhou a třetí trasu. Když bychom tak učinili, překročili bychom kapacitu vozidla. Trasu nepřidáme.

Následující trasy nemůžeme přidat ze stejných důvodů jako v předcházejícím odstavci: [3 – 8], [6 – 8], [3 – 6].

Existuje ještě jedna situace kdy nemůžeme trasy sloučit a to ta kdy je alespoň jeden uzel vnitřním. Představme si, že kapacita vozidla není 4 ale 5. Úspory z tras [6 – 9], [2 – 10], [6 – 10] a [2 – 9] nemůžeme přidat, protože alespoň jeden zákazník není na kraji trasy. Kdybychom chtěli přidat cestu [6 – 9] můžeme zkusit sloučit trasy a dostali bychom například trasu [DEPO – 3 – 10 – 6 – 9 – 8 – DEPO]. V trase máme cestu [6 – 9] ale přidala se nám i cesta [10 – 6] u které nemůžeme zaručit, že úspora z této cesty bude výhodná, dokonce by se mohlo stát, že bude záporná.

V dalším kroku je největší úspora z cesty [5 - 7]. Zákazníci s čísly 5 a 7 jsou stále v elementárních trasách a nic nám tedy nebrání tyto 2 trasy spojit do jedné.

Tabulka 8: První úloha, čtvrté průběžné řešení

Průběžně řešení	Vzdálenost [km]	Náklad
DEPO -> 1 -> DEPO	25,2	1
DEPO -> 2 -> DEPO	109,8	1
DEPO -> 3 -> 10 -> 9 -> 8 -> DEPO	307,7	4
DEPO -> 4 -> DEPO	51	1
DEPO -> 5 -> 7 -> DEPO	208,2	2
DEPO -> 6 -> DEPO	97,4	1

Zdroj: vlastní zpracování, 2022

Další úspora v pořadí je z přidání trasy [2 – 7]. Sloučení 2. s 5. trasou je možné a neporušíme tím ani kapacitní podmínky vozidla.

Tabulka 9: První úloha, páté průběžné řešení

Průběžně řešení	Vzdálenost [km]	Náklad
DEPO -> 1 -> DEPO	25,2	1
DEPO -> 3 -> 10 -> 9 -> 8 -> DEPO	307,7	4
DEPO -> 4 -> DEPO	51	1
DEPO -> 5 -> 7 -> 2-> DEPO	260,6	3
DEPO -> 6 -> DEPO	97,4	1

Zdroj: vlastní zpracování, 2022

Další úspora je z cesty [3 – 7]. Třetí a čtvrtou trasu ale nemůžeme sloučit, protože náklad který by vznikl součtem současných nákladů na trasách, tedy 4 + 3 = 7 je větší než kapacitní omezení vozidla, které je nastavené na 4. Druhým důvodem by byl i fakt, že je zákazník č. 3 je vnitřním uzlem ve 3. trase.

Následující cesty nemůžeme přidat: [4 – 7] a [2 – 8]. Do řešení připojíme až cestu [4 – 5] a vznikne nám následující průběžné řešení.

Tabulka 10: První úloha, šesté průběžné řešení

Průběžně řešení	Vzdálenost [km]	Náklad
DEPO -> 1 -> DEPO	25,2	1
DEPO -> 3 -> 10 -> 9 -> 8 -> DEPO	307,7	4
DEPO -> 4 -> 5 -> 7 -> 2 -> DEPO	268,6	4
DEPO -> 6 -> DEPO	97,4	1

Zdroj: vlastní zpracování, 2022

Druhou a třetí trasu už nemůžeme s žádnou jinou sloučit, z důvodu překročení kapacity vozidla. Jediné možné sloučení je tedy tras 1 a 4. Úspora z přidání cesty [1 – 6] je 19,7 > 0. Můžeme tedy trasy sloučit a získat konečné řešení úlohy.

Tabulka 11: První úloha, konečné řešení

Konečné řešení	Vzdálenost [km]	Náklad
DEPO -> 1 -> 6 -> DEPO	102,9	2
DEPO -> 3 -> 10 -> 9 -> 8 -> DEPO	307,7	4
DEPO -> 4 -> 5 -> 7 -> 2 -> DEPO	268,6	4
Celkem	679,2	10

Zdroj: vlastní zpracování, 2022

Porovnání s aplikací

Stejné zadání úlohy jsem nastavil i do aplikace a získal tyto výsledky:

Tabulka 12: První úloha, výsledky z aplikace

Výsledky z aplikace	Vzdálenost [km]	Náklad
DEPO - 1 - 6 - DEPO	102,932	2
DEPO - 3 - 10 - 9 - 8 - DEPO	303,713	4
DEPO - 2 - 7 - 5 - 4 - DEPO	270,478	4
Celkem	677,123	10

Zdroj: vlastní zpracování, 2022

Můžeme si všimnout jsou všechny trasy stejné, maximálně je 3. trasa v obráceném pořadí. Jelikož máme symetrickou matici vzdálenosti, tak nám tato skutečnost nevadí.
Vzdálenosti tras se liší, to může být způsobeno tím, že nám API vrátilo trochu odlišné vzdálenosti mezi uzly. Tato odlišnost je minimální a můžeme ji zanedbat.

6.1.2 Ruční výpočet pomocí metody Nearest Neighbour

1. Trasa

   Vybereme z distanční matice nejkratší vzdálenost z depa k jakémukoliv zákazníkovi. Tím je zák. č. 1. Začneme tedy novou trasu a přidáme do ní našeho zákazníka. První trasa vypadá následovně: [DEPO – 1].

   Nyní nalezneme nejbližšího souseda k zák. č. 1, kterého jsme ještě nenavštívili. Je jím zák. č. 4 a připojíme ho na konec první trasy. Ta se tedy změnila do následující podoby: [DEPO – 1 – 4].

Takto pokračujeme dále až do té doby kdy bychom přidáním dalšího nejbližšího souseda porušili kapacitní omezení vozidla. To nastane ve fázi kdy máme trasu [DEPO – 1 – 4 – 5 – 7]. Nyní bychom chtěli přidat zák. č. 2, ale náklad vozidla by byl 5. Kapacita vozidla je nastavená na 4 a proto tuto změnu neschválíme, ukončíme trasu návratem do depa a začneme trasu novou. Konečná první trasa vypadá následovně: [DEPO – 1 – 4 – 5 – 7 – DEPO]

2. Trasa

   Nejbližší soused, který ještě nebyl navštíven je zák. č. 6. Uděláme k němu tedy trasu a pokračujeme stejně jako dosud. Konečná druhá trasa vypadá následovně: [DEPO – 6 – 9 – 8 – 10 – DEPO]

3. Trasa

   Konečná třetí trasa: [DEPO – 2 – 3 – DEPO]

Celkové výsledky udává tab. č. 13.

Tabulka 13: První úloha, metoda NN

Konečné řešení	Vzdálenost [km]	Náklad
DEPO – 1 – 4 – 5 – 7 – DEPO	241,5	4
DEPO – 6 – 9 – 8 – 10 – DEPO	300,4	4
DEPO – 2 – 3 – DEPO	207,2	2
Celkem	749,1	10

Zdroj: vlastní zpracování, 2022

Porovnání výsledků s aplikací

Zadání bylo opět přeneseno do aplikace která vrátila výsledky uvedené v tabulce č. 14.

Tabulka 14: První úloha, metoda NN, výsledky z aplikace

Konečné řešení	Vzdálenost [km]	Náklad
DEPO – 1 – 4 – 5 – 7 – DEPO	242,504	4
DEPO – 6 – 9 – 8 – 10 – DEPO	299,7	4
DEPO – 2 – 3 – DEPO	203,234	2
Celkem	745,438	10

Zdroj: vlastní zpracování, 2022

Výsledky mají opět zanedbatelné odchylky. Všechny trasy vyšly stejně jako při ručním propočítání.

6.2 Porovnání výsledků heuristik

Pro porovnání heuristik jsme zvolili jednotné nastavení. Jako začáteční trasa bylo zvoleno 5 adres v České republice. U každého kroku byla zvolena kapacita vozidla na hodnotu 5.

Pro každé porovnání byla nastavena tyto pravidla:

- Kapacitní omezení vozidla: 5
- Nastavení mapy.cz: krátká
- Depo: 1, Praha

1. Porovnání

Počet adres: 5

Metoda Clarke-Wright: 549,246 km

- Příloha 1

Metoda Nearest Neighbour: 549,246 km

- Příloha 2

Výsledek: Příloha 3

V prvním porovnání nám vyšlo, že se trasa vytvořila naprosto stejně. Je to tím, že u tak malého vzorku nemusí být takový rozdíl mezi heuristikami, protože se jedná o jednodušší problém.

2. Porovnání

Počet adres: 10

Metoda Clarke-Wright: 599,663 km

- Příloha 4

Metoda Nearest Neighbour: 607,84 km

- Příloha 5

Výsledek: Příloha 6

Ve druhém porovnání již můžeme vidět, že metoda Clarke -Wright byla úspěšnější než metoda nejbližšího souseda. Prozatím nemůžeme říci, že je metoda přesnější, ale z principu metody výpočtu se může stát, že u metody nejbližšího souseda můžeme čekat horší výsledky.

3. Porovnání

- Počet adres: 15

Metoda Clarke-Wright: 1467,786 km

- Příloha 7

Metoda Nearest Neighbour: 1345,407 km

- Příloha 8

Výsledek: Příloha 9

Ve třetím porovnání byla překvapivě lepší heuristika nejbližšího souseda, která poskytla o 112 kilometrů kratší trasu.

4. Porovnání

Počet adres: 20

Metoda Clarke-Wright: 1772,689 km

- Příloha 10

Metoda Nearest Neighbour: 2104,868 km

- Příloha 11

Výsledek: Příloha 12

Pro 20 adres byl výsledek jednoznačný. Metoda Clarke-Wright nalezla trasu, která byla kratší o více než 330 kilometrů.

5. Porovnání

Počet adres: 25

Metoda Clarke-Wright: 2092,569 km

- Příloha 13

Metoda Nearest Neighbour: 2142,626 km

- Příloha 14

Výsledek: Příloha 15

V posledním porovnání, kde bylo 25 adres byla opět lepší heuristika Clarke-Wright, která nalezla o 50 kilometrů lepší výsledek.

6.2.1 Souhrn výsledků porovnávání

Tato kapitola byla věnována porovnáním mezi heuristikami, které byly využity ve webové aplikaci. Ve 3 případech vyšla heuristika Clarke – Wright lépe než metoda nejbližšího souseda. V jednom případě metody vyšly stejně a v jednom případě byla metoda nejbližšího souseda lepší.

Ze závěru lze interpretovat, že pokud máme více adres, tak je bezpečnější použít metodu Clarke – Wright pro její komplexnější přístup k vypočtu. U metody nejbližšího souseda se může stát, že první vybrání trasy může být výhodné, ale pozdější výběry mohou být již velmi nevýhodné kvůli „hloupému“ výpočtu následujících zastávek.

7 Uživatelská příručka

Jelikož program běží ve webovém prohlížeči není potřeba ho nějak instalovat, ale stačí ho pouze zapnout, případně nám stačí přejít na stránky https://home.zcu.cz/~krevkym/bp/.

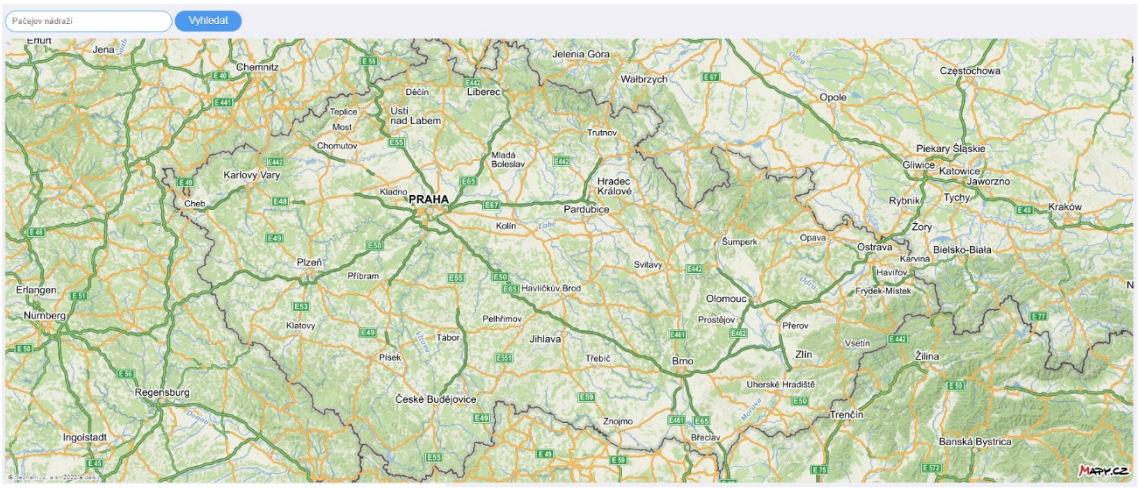
7.1 Komponenty aplikace:

Když program spustíme uvidíme mapu a pod ní tři sloupečky. První sloupeček slouží pro výpočet tras, druhý pro vytvoření míst z textové podoby a do třetího se zaznamenávají výsledky algoritmů.

Mapa

Mapa zabírá velkou plochu aby se s ní dalo příjemně pracovat. Nad mapou se nachází vyhledávací pole k nalezení žádané lokality.

Obrázek 15: Mapa v aplikaci



Zdroj: vlastní zpracování, 2022

Mapa je interaktivním rozhraním. Kliknutím na jakékoliv místo, se vytvoří značka a místo se přidá do tabulky pod mapou. Značka je takzvaně „draggable“. V češtině to znamená „přetahovatelné“ a dovoluje přesouvat značku pomocí podržení levého tlačítka myši, přetáhnutím a uvolněním tlačítka. Značky se na mapě číslují a podle tohoto značení je můžeme vyhledávat i v tabulce.

Ovládací část

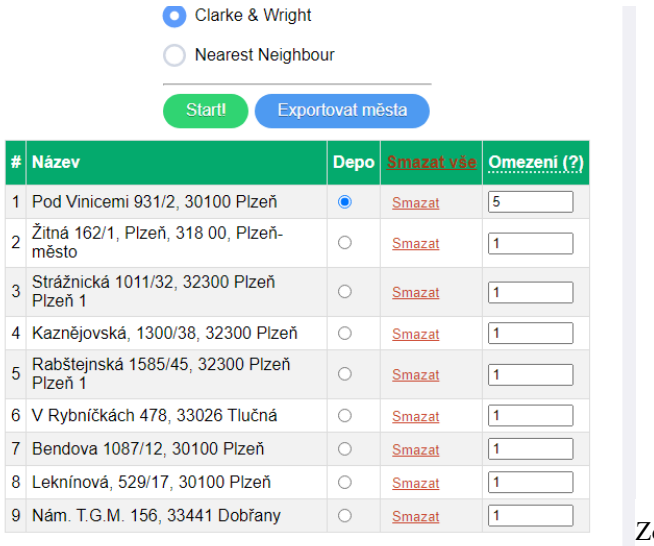
Tato komponenta je jednou ze tří komponent pod mapou a slouží nám pro zobrazení zadání úlohy. V horní části se nacházejí dvě přepínací tlačítka pro výběr algoritmu, který bude použit pro výpočet tras. Na výběr je heuristika Clarke & Wright savings algorithm a metoda nejbližšího souseda.

Pod výběrem algoritmu jsou další dvě klasická tlačítka. Tlačítko Start! slouží k zahájení výpočtu tras pomocí zvoleného algoritmu. Tlačítko Exportovat města nám dovoluje exportovat města do textového pole, které se nachází v prostředním sloupečku. Formát exportovaných dat je v souřadnicovém tvaru, jak bylo vysvětleno v minulé kapitole.

V další části se nachází tabulka s vybranými místy. Tabulka má tyto následující sloupce:

- První sloupeček udává číslo lokality, které koresponduje s číslem na mapě.
- Druhý sloupec je adresa našeho místa.
- V třetím sloupci se nachází přepínací tlačítko pro výběr depa, které můžeme vybrat pouze jedno. Výběr depa je vyžadován před startem výpočtu, tj. před kliknutím na tlačítko Start!.
- Předposlední sloupec obsahuje tlačítko na vymazání místa z tabulky i mapy. Tlačítko v hlavičce tabulky smaže veškeré záznamy.
- Poslední sloupec slouží k definování omezení. K řádku s depem se zadá kapacitní omezení pro vozidla a k zákazníkům zadáme jejich požadavky.

Obrázek 16: Ovládací prvky aplikace



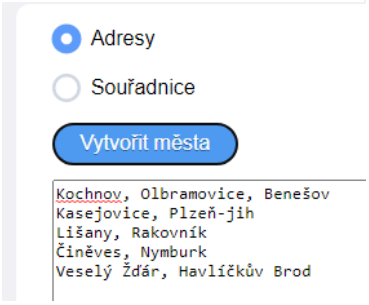
Zdroj: vlastní zpracování, 2022

Část pro import lokalit

V této části programu můžeme vytvořit města pomocí vícenásobného importu. Slouží k tomu textové pole. Můžeme si vybrat jakým způsobem budeme zadávat data. Na výběr přicházejí v úvahu dvě možnosti:

- Adresy – data zadáváme ve tvaru *adresa;omezení* (např. Mikoláše Alše 253, Kolín, 280 02, Kolín;14). Omezení je volitelné a nemusí být zadáno. V tomto případě se automaticky nastaví na hodnotu 1. Po kliknutí na tlačítko „Vytvořit města“ se naplní tabulka a na mapě se zobrazí příslušné značky.
- Souřadnice – zadáváme ve tvaru: *zeměpisná šířka␣zeměpisná délka␣omezení*. Omezení opět není povinné zadávat a automaticky se nastaví na hodnotu 1.

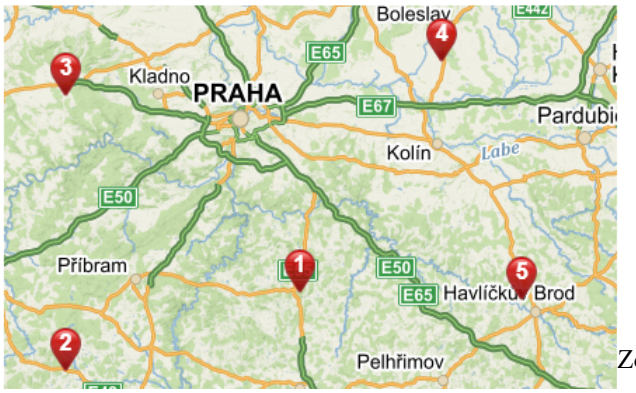
Obrázek 18: Import pomocí adres



Zdroj: vlastní zpracování, 2022

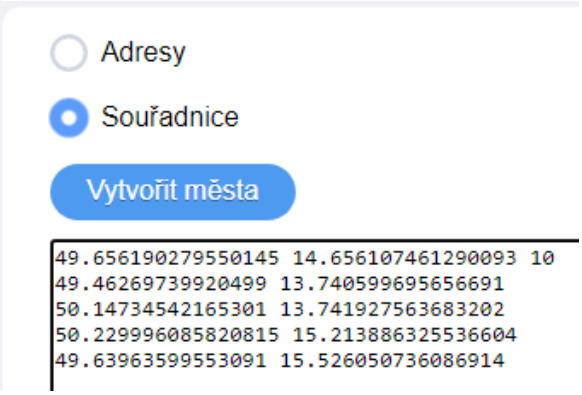
Obrázek 17: Výsledky importu pomocí adres

#	Název	Depo	Smazat vše	Omezení (?)
1	Kochnov, Olbramovice, Benešov	○	Smazat	1
2	Kasejovice, Plzeň-jih	○	Smazat	1
3	Lišany, Rakovník	○	Smazat	1
4	Činěves, Nymburk	○	Smazat	1
5	Veselý Žďár, Havlíčkův Brod	○	Smazat	1



Zdroj: vlastní zpracování, 2022

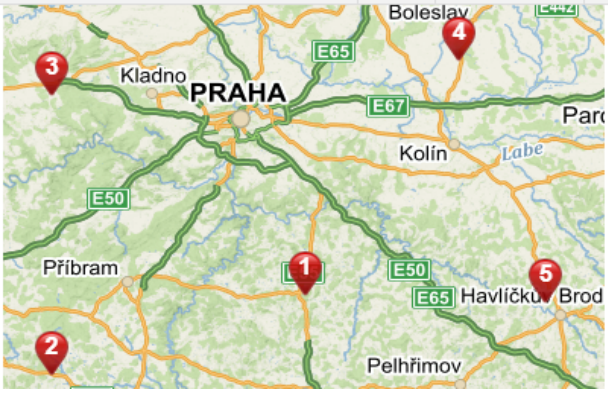
Obrázek 20: Import pomocí souřadnic



Zdroj: vlastní zpracování, 2022

Obrázek 19: Výsledky importu pomocí souřadnic

#	Název	Depo	Smazat vše	Omezení (?)
1	Kochnov 1, Olbramovice, 259 01, Benešov	○	Smazat	10
2	Kasejovice 24, Kasejovice, 335 44, Plzeň-jih	○	Smazat	1
3	Pražská, Lišany, Rakovník	○	Smazat	
4	Činěves 34, Činěves, 289 01, Nymburk	○	Smazat	1
5	Veselý Žďár 35, Veselý Žďár, 580 01, Havlíčkův Brod	○	Smazat	1



Zdroj: vlastní zpracování, 2022

Výsledky

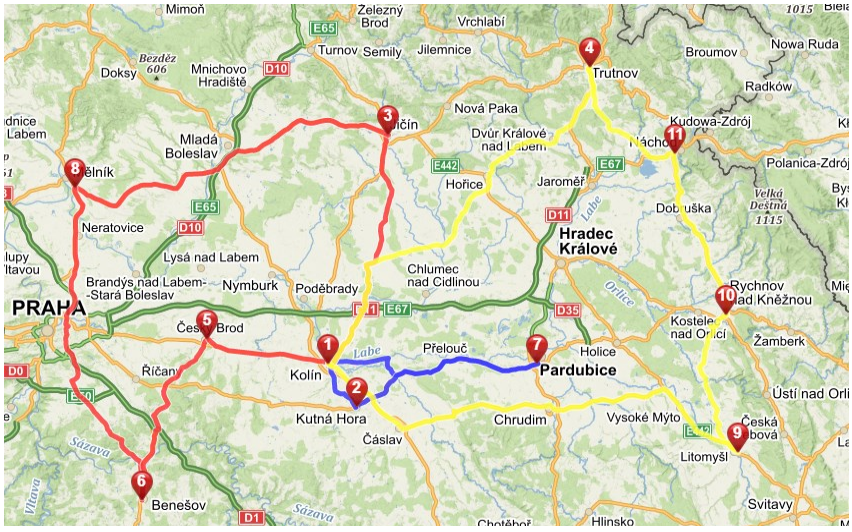
Výsledky výpočtu v textové podobě se zaznamenávají do třetího sloupce. Je k dispozici tlačítko „Smazat vše“, které smaže veškeré zapsané výsledky nebo lze vymazat jen konkrétní výsledek.

Obrázek 21: Ukázka textového výpisu

Zdroj: vlastní zpracování, 2022

Ve výsledku je uveden algoritmus kterým se úloha řešila. Na další řádce začínají jednotlivé trasy. Trasa je popsána posloupností čísel. Ty jsou shodné s číslováním v tabulce resp. v mapě. Pod každou trasou jsou informace o její délce a o tom jak velký bude náklad vozidla. Na úplně poslední řádce je vypočítaná celková délka všech okruhů.

Obrázek 22: Výsledek na mapě



Zdroj: vlastní zpracování, 2022

7.2 Postup práce s aplikací

Příprava dat

Začneme s přípravou dat pro program. Nejdříve je potřeba sestavit množinu uzlů, které budeme zpracovávat. Můžeme pracovat s mapou a přidávat místa jedno po druhém, nebo pokud máme adresy resp. souřadnice, tak je můžeme importovat pomocí textového pole.

Po vytvoření množiny uzlů doplníme omezení do příslušného textového pole.

Spuštění aplikace

Po stisknutí tlačítka start se začne zpracovávat zadaný vstup. Při větším množství uzlů se můžeme potýkat s větší prodlevou. Důvod byl vysvětlen v předcházející kapitole.

Čtení výsledků

Po úspěšném výpočtu tras nám schází jen výsledky přečíst v pravé dolní části a prohlédnout si vytvořené trasy na mapě.

8 Závěr

Cílem bakalářské práce bylo vytvořit webovou aplikaci pro plánování tras za pomocí heuristik.

V praktické části byl nejdříve popsán vývoj webové aplikace, kde bylo také popsáno propojení s rozhraním pro využití mapování tras v bakalářské práci. Ve webové aplikaci bylo vytvořeno několik sekcí, které slouží pro správnou funkčnost. Je možné naklikat daná města na mapě, vyhledat přes našeptávač, anebo zadat adresy či souřadnice hromadně.

Stěžejní částí aplikace je možnost využití dvou heuristik pro barevné označení tras na mapě s výsledkovou tabulkou návaznosti tras.

V praktické části jsme porovnali také algoritmus ve webové aplikaci s ručním výpočtem metod, kde se prokázalo, že metody funguje stejně, akorát ruční metoda zabere mnohem více času. Heuristické metody jsme také porovnali mezi sebou na pěti okruzích, kde jsme si ukázali, že dané heuristiky počítají dané trasy jinak při větším počtu adres k doručení.

V závěru praktické části můžeme naleznout manuál, pro práci s aplikací.

Seznam použitých zdrojů

BROŽOVÁ, H., & HOUŠKA, M. (2008). *Základní metody operační analýzy*. Praha: Česká zemědělská univerzita v Praze

CLARKE, G., & WRIGHT J. W. (1964). *Scheduling of Vehicles from a Central Depot to a Number of Devilery Points*. USA: INFORMS

DANTZIG G. B., & RAMSER J. H. (1959). *The Truck dispatching problem*. USA: INFORMS

FIALA, P. (2010). *Operační výzkum - nové trendy*. Praha: Professional Publishing

IORI, M., GONZALEZ, J., & VIGO, D. (2005). *An exact approach for the vehicle routing problem with twodimensional loading constraints*. Bologna: Universita degli studi di Bologna, 2005.

JABLONSKÝ, J. (2007). *Operační výzkum. Kvantitativní modely pro ekonomické rozhodování* (3. vyd.). Praha: Professional Publishing

JOMINI A. H. (1838). *Summary of the Art of War*. Neznámý vydavatel.

LAPORTE, G. (1991). *The Vehicle Routing Problem: An overview of exact and aproximate algorithms*, Montreal: Universite de Montreal

PLEVNÝ, M., & ŽIŽKA M. (2010). *Modelování a optimalizace v manažerském rozhodování* (2. vyd.). Plzeň: Západočeská univerzita v Plzni

SALAJ, D. (2007). *Vehicle Routing Problem Metódy Riešenia*, Bratislava: Univerzita Komenského

SHEN, Z., DESOUKY M., & ORDONEZ F. (2007). *The Stochastic Vehicle Routing Problem for Large-scale Emergencies*, Los Angeles: University of Southern California,

SIXTA, J., & MAČÁT, V. (2005). *Logistika: teorie a praxe* (1. vyd.). Brno: CP Books.

ŠIROKÝ, V., & SLIVONĚ , M. (2010). *Optimalizace svozu a rozvozu kusových zásilek. Perner´s contacts.* Dostupné z: https://pernerscontacts.upce.cz/index.php/perner/article/view/959/793.

TOTH, P., & VIGO, D. (2002). *The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications* (9. vyd.). Philadelphia: Society for Industrial and Applied Mathematics.

Google.com (2022). *Vehicle Routing Problem with Time Windows.* Dostupné z: https://developers.google.com/optimization/routing/vrptw

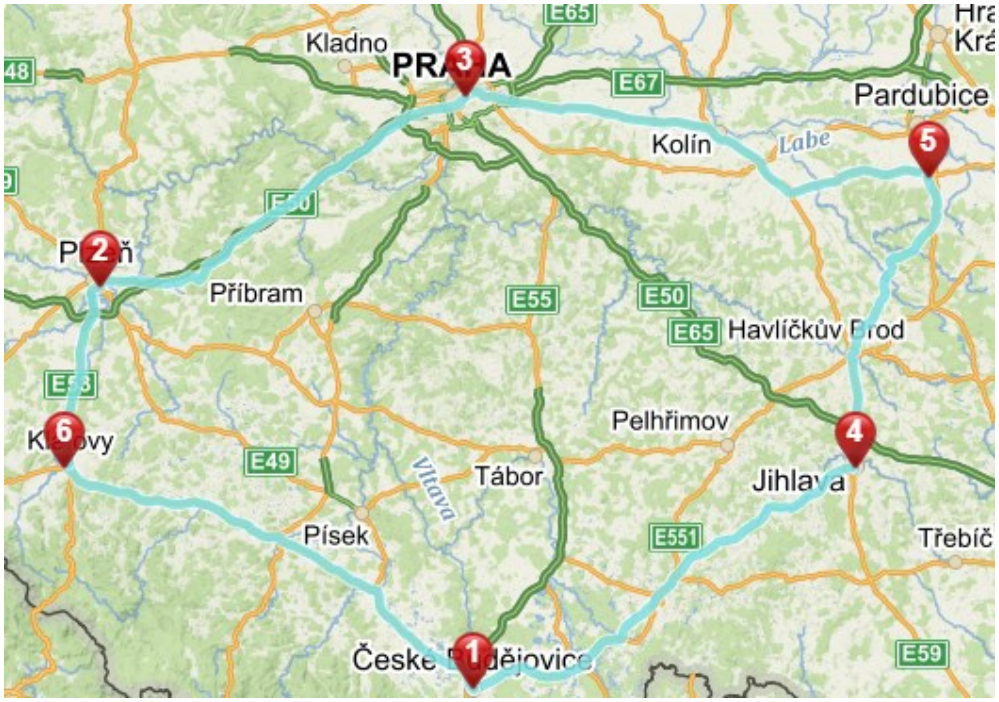
Seznam tabulek

Seznam obrázků

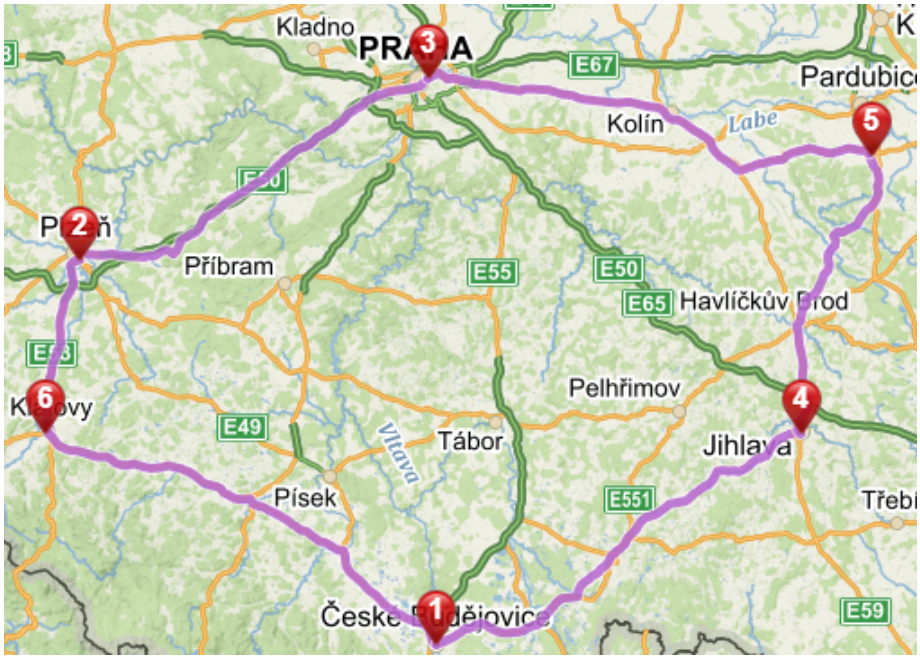
Seznam příloh

Přílohy

Příloha 1: Clarke-Wright (5 adres)



Příloha 2: Nearest Neighbour (5 adres)



Příloha 3: Výsledek metod (5 adres)

Clarke & Wright Smazat

1. trasa: 3 5 4 1 6 2 3
Délka trasy: 549.246 km; Náklad: 5

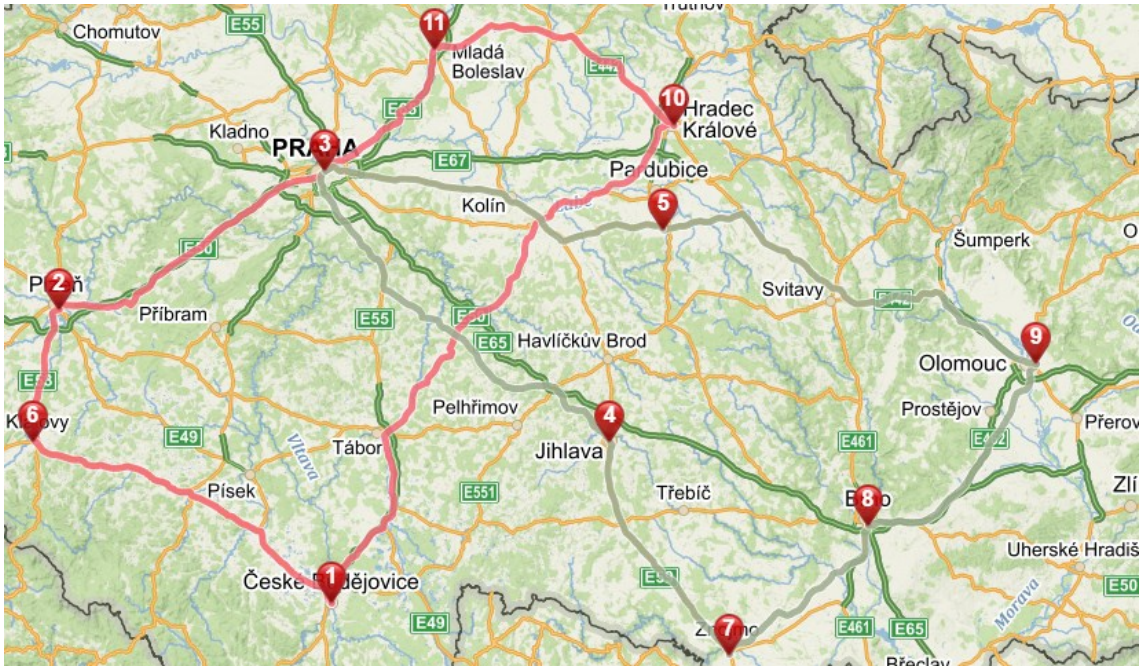
Celková délka: 549.246 km

Nearest Neighbour Smazat

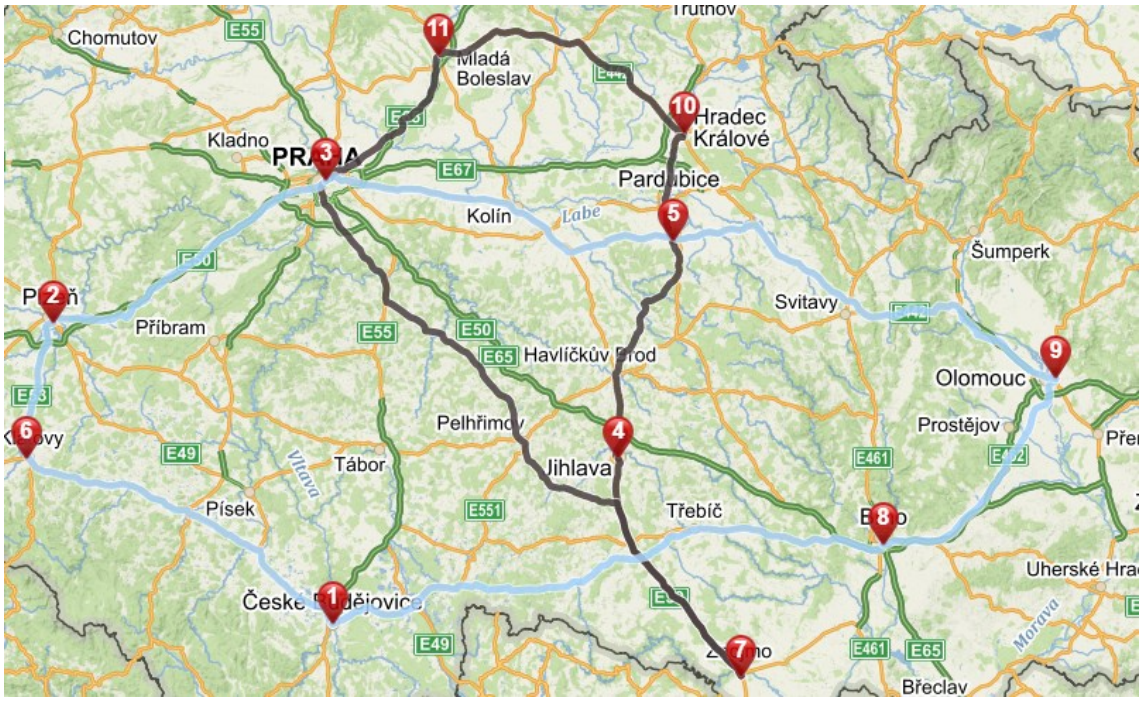
1. trasa: 3 2 6 1 4 5 3
Délka trasy: 549.246 km; Náklad: 5

Celková délka: 549.246 km

Příloha 4: Clarke-Wright (10 adres)



Příloha 5: Nearest Neighbour (10 adres)



Příloha 6: Výsledek metod (10 adres)

Clarke & Wright Smazat

1. trasa: 3 2 6 1 10 11 3
Délka trasy: 599.663 km; Náklad: 5

2. trasa: 3 4 7 8 9 5 3
Délka trasy: 607.84 km; Náklad: 5

Celková délka: 1207.503 km

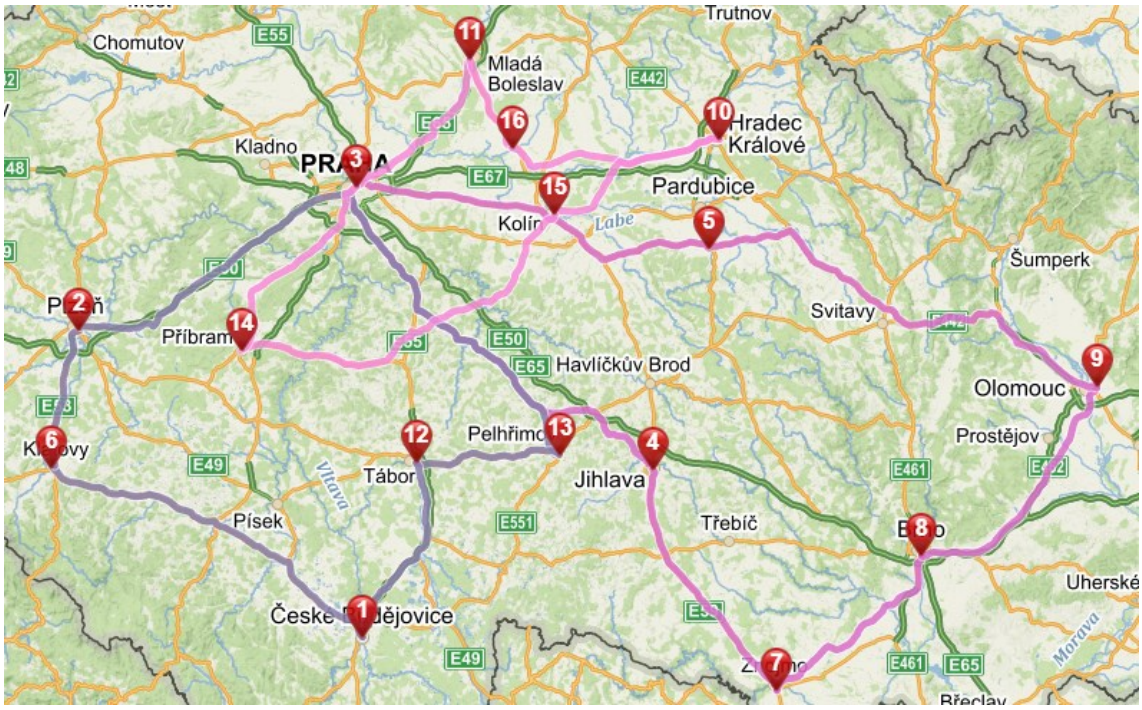
Nearest Neighbour Smazat

1. trasa: 3 11 10 5 4 7 3
Délka trasy: 538.859 km; Náklad: 5

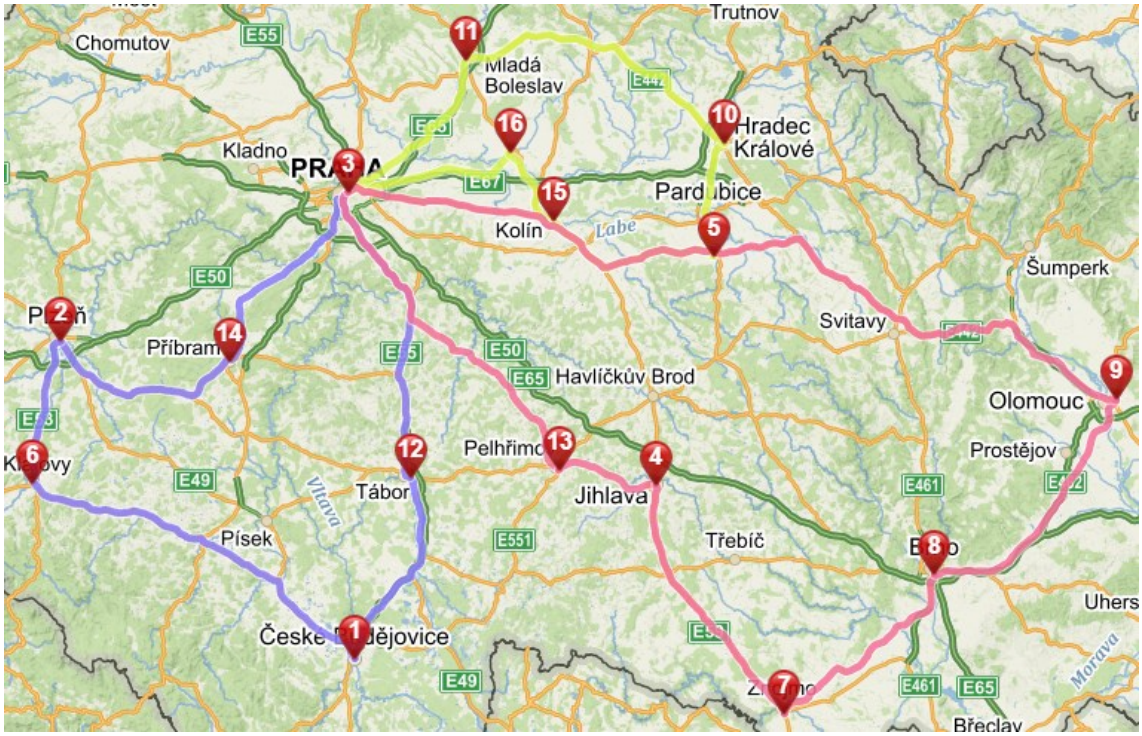
2. trasa: 3 2 6 1 8 9 3
Délka trasy: 765.15 km; Náklad: 5

Celková délka: 1304.009 km

Příloha 7: Clarke-Wright (15 adres)



Příloha 8: Nearest Neighbour (15 adres)



Příloha 9: Výsledek metod (15 adres)

Clarke & Wright Smazat

1. trasa: 3 4 7 8 9 5 3
Délka trasy: 607.84 km; Náklad: 5

2. trasa: 3 13 12 1 6 2 3
Délka trasy: 467.468 km; Náklad: 5

3. trasa: 3 11 16 10 15 14 3
Délka trasy: 392.478 km; Náklad: 5

Celková délka: 1467.786 km

Nearest Neighbour Smazat

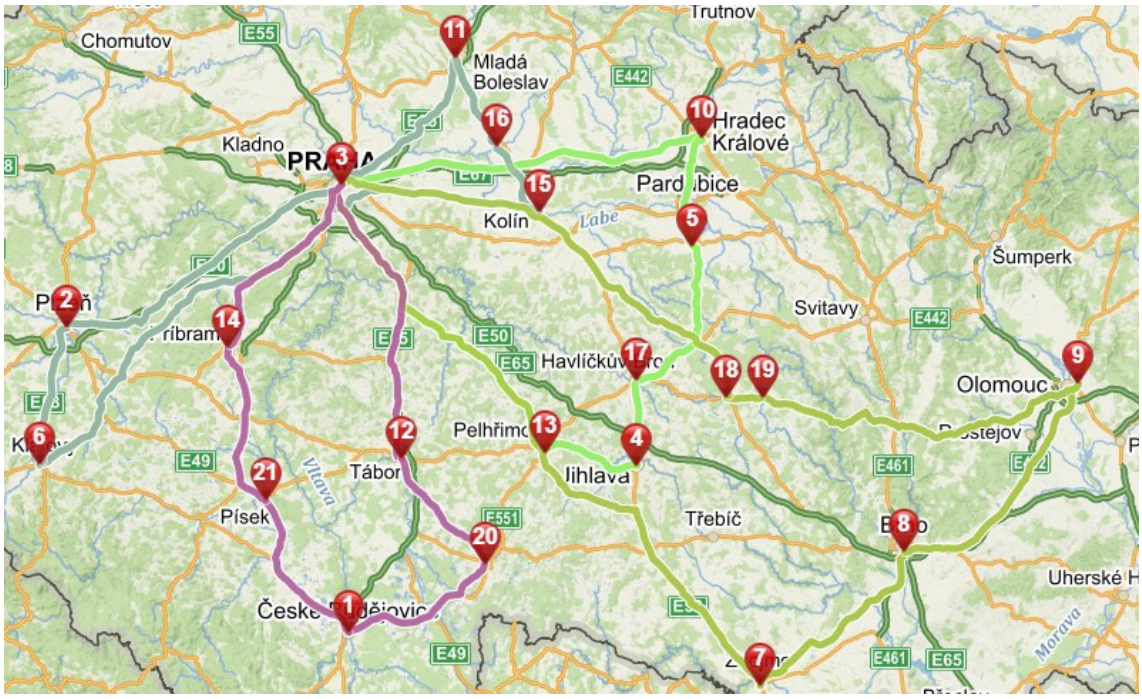
1. trasa: 3 16 15 5 10 11 3
Délka trasy: 301.817 km; Náklad: 5

2. trasa: 3 14 2 6 1 12 3
Délka trasy: 433.002 km; Náklad: 5

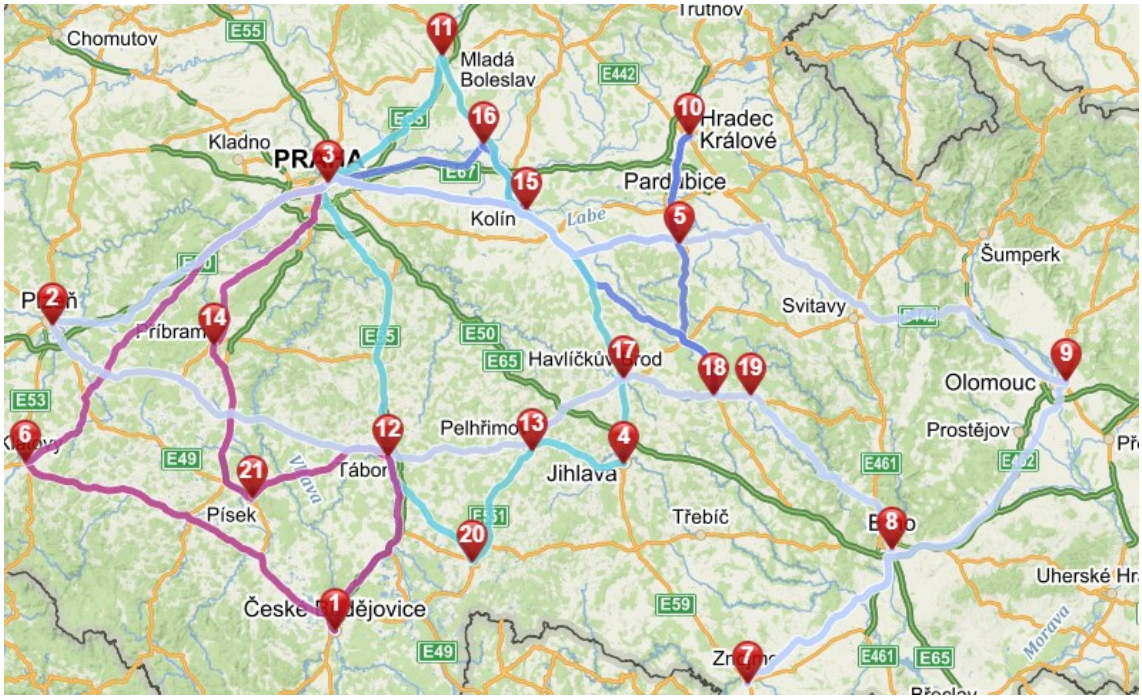
3. trasa: 3 13 4 7 8 9 3
Délka trasy: 610.588 km; Náklad: 5

Celková délka: 1345.407 km

Příloha 10: Clarke-Wright (20 adres)



Příloha 11: Nearest Neighbour (20 adres)



Příloha 12: Výsledek metod (20 adres)

Clarke & Wright Smazat

1. trasa: 3 10 5 17 4 13 3
Délka trasy: 359.197 km; Náklad: 5

2. trasa: 3 11 16 15 6 2 3
Délka trasy: 442.131 km; Náklad: 5

3. trasa: 3 7 8 9 19 18 3
Délka trasy: 618.578 km; Náklad: 5

4. trasa: 3 12 20 1 21 14 3
Délka trasy: 352.783 km; Náklad: 5

Celková délka: 1772.689 km

Nearest Neighbour Smazat

1. trasa: 3 16 15 5 10 18 3
Délka trasy: 385.942 km; Náklad: 5

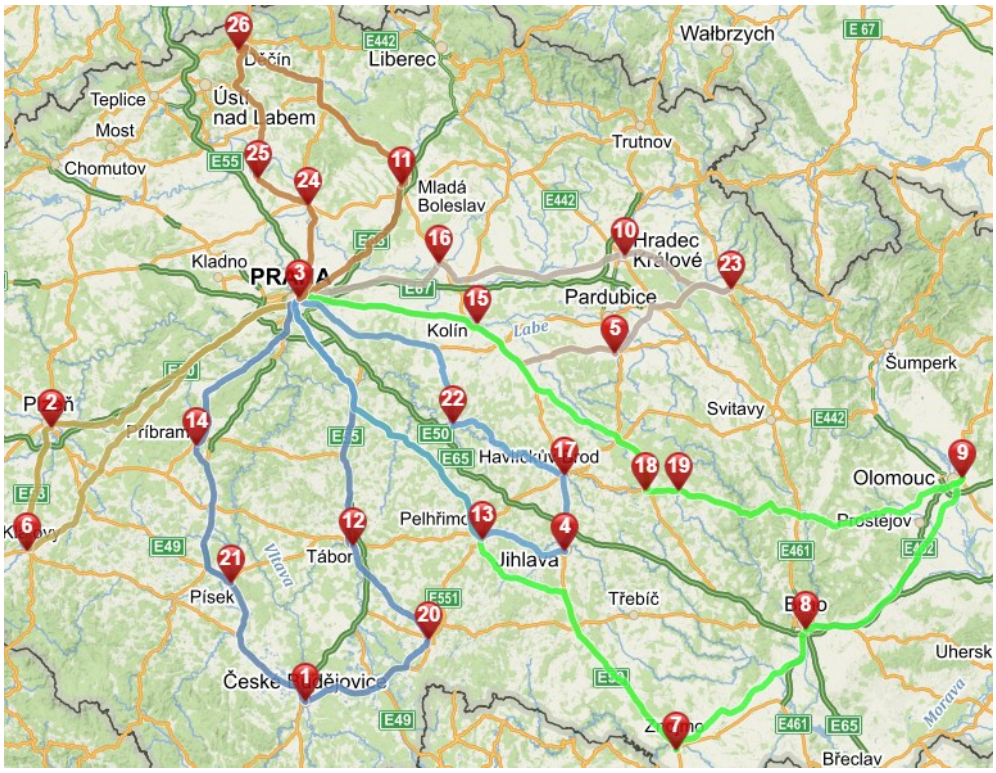
2. trasa: 3 11 17 4 13 20 3
Délka trasy: 396.895 km; Náklad: 5

3. trasa: 3 14 21 12 1 6 3
Délka trasy: 466.836 km; Náklad: 5

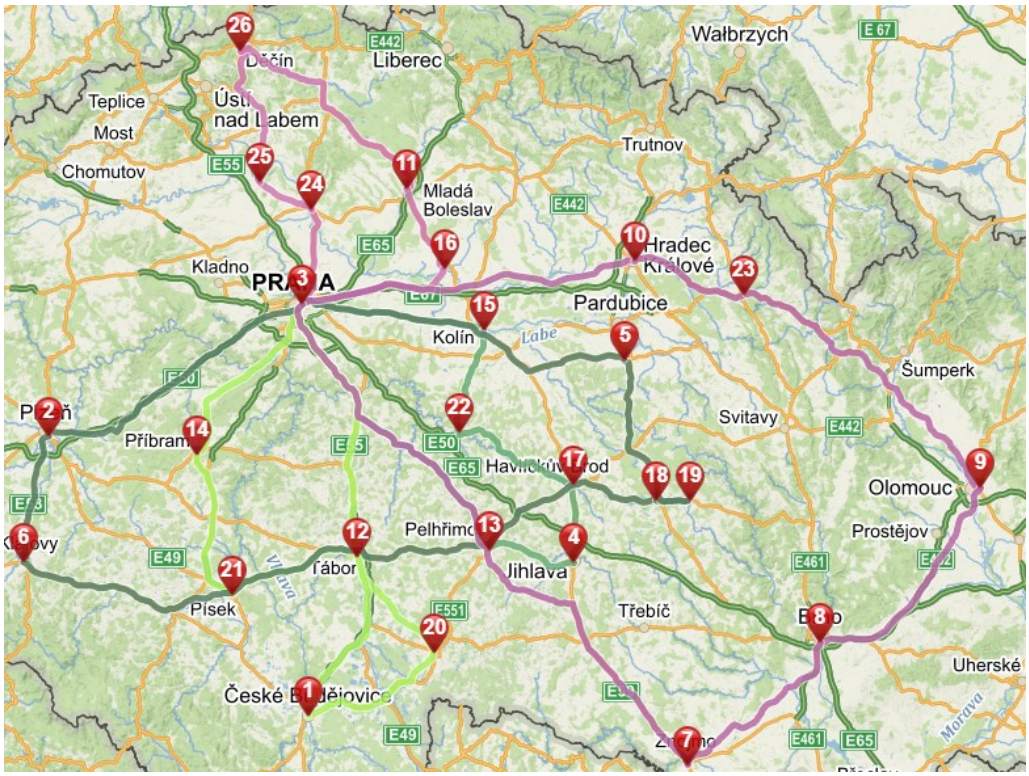
4. trasa: 3 2 19 8 7 9 3
Délka trasy: 855.195 km; Náklad: 5

Celková délka: 2104.868 km

Příloha 13: Clarke-Wright (25 adres)



Příloha 14: Nearest Neighbour (25 adres)



Příloha 15: Výsledek metod (25 adres)

Clarke & Wright Smazat

1. trasa: 3 6 2 3
Délka trasy: 275.45 km; Náklad: 2

2. trasa: 3 16 10 23 5 15 3
Délka trasy: 312.045 km; Náklad: 5

3. trasa: 3 7 8 9 19 18 3
Délka trasy: 618.578 km; Náklad: 5

4. trasa: 3 12 20 1 21 14 3
Délka trasy: 352.783 km; Náklad: 5

5. trasa: 3 22 17 4 13 3
Délka trasy: 293.742 km; Náklad: 4

6. trasa: 3 11 26 25 24 3
Délka trasy: 239.971 km; Náklad: 4

Celková délka: 2092.569 km

Nearest Neighbour Smazat

1. trasa: 3 24 25 26 11 16 3
Délka trasy: 264.009 km; Náklad: 5

2. trasa: 3 15 22 17 4 13 3
Délka trasy: 310.276 km; Náklad: 5

3. trasa: 3 14 21 12 20 1 3
Délka trasy: 406.316 km; Náklad: 5

4. trasa: 3 2 6 18 19 5 3
Délka trasy: 550.878 km; Náklad: 5

5. trasa: 3 10 23 9 8 7 3
Délka trasy: 611.147 km; Náklad: 5

Celková délka: 2142.626 km

Abstrakt

Křevký, M. (2022). *Implementace jednodušších heuristik pro řešení specifických variant rozvozních problémů (*Bakalářská práce), Západočeská univerzita v Plzni, Fakulta ekonomická, Česko.

Klíčová slov

heuristiky, meta-heuristiky, rozvozní problémy, dopravní problémy, metoda nejbližšího souseda, webová aplikace, Clarke-Wright

Tato bakalářská práce probírá porovnání dvou různých heuristik za použití webové aplikace. Teoretická část je věnována základním pojmům, rozvozním úlohám a základům metod pro výpočet tras.

V praktické části je řešena webové aplikace, na které funguje výpočet plánování tras za použití dvou heuristik a to Clarke – Wrightovy metody a metody nejbližšího souseda. Na konci je přiložen také manuál, který ukazuje, jak s danou aplikací zacházet.

Abstract

Krevký, M. (2022). *Implementation of simpler heuristics to solve some specific variants of delivery problems (*Bachelor thesis), University of West Bohemia, Faculty of Economics, Czech Republic.

Keywords

heuristics, meta-heuristics, distribution problems, vehicle routing problems, nearest neighbour method, web application, Clarke-Wright

This bachelor thesis discusses the comparison of two different heuristics using a web application. The theoretical part is devoted to basic concepts, derivation problems and the basics of heuristics for route computation.

The practical part deals with a web application that works to compute route planning using two heuristics namely Clarke - Wright method and nearest neighbour method. At the end there is also a manual that shows how to use the application.