Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Optimalizace odečtové trasy

Bakalářská práce

František Svačina

Vedoucí bakalářské práce: Ing. Václav Novák, CSc.

České Budějovice 2014

Bibliografické údaje

Svačina, F., 2014: Optimalizace odečtové trasy. [Optimization of meter reading route. Bc. Thesis, in Czech.] – 55 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Abstrakt

Tato práce se zabývá vytvořením softwaru, který bude vypočítávat a zobrazovat optimalizované trasy. Cílem práce bylo vytvořit software pracující na operačním systému Windows. Základním smyslem aplikace je nalézt nejvhodnější trasu pro pohyb pracovníka tak, aby navštívil všechna zadaná místa.

Abstract

This thesis deals with the creation of software that will calculate and display the optimized route. The aim was to create software running on the Windows operating system. The basic purpose of the application is to find the most suitable route for the movement of workers, to visit all entered places.

Klíčová slova

Optimalizace tras, wpf, problém obchodního cestujícího

Key-words

Route optimization, wpf, travelling salesman problem

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 25. 04. 2014 František Svačina

Poděkování

Touto cestou bych rád poděkoval vedoucímu mé bakalářské práce panu Ing. Václavu Novákovi, CSc. za ochotu a vstřícnost při poskytování informací a rad.

Obsah

1 Úvod

Informační technologie patří v současné době k jednomu z nejrozvíjenějších průmyslů. Výpočetní technologie nepatří v dnešní době nejen k nezbytnému vybavení všech podniků, ale také se jedná o standardní vybavení téměř každé domácnosti.

Zároveň také dochází k většímu přemisťování lidí, kteří pro svou orientaci v neznámých oblastech často využívají právě informačních technologií, kde jsou přístupné mapy a plánovače tras.

Problém obchodního cestujícího byl popsán o mnoho dříve, než byl vynalezen první počítač. Tato problematika byla řešena před více než 250 lety, avšak dodnes nebyla nalezena metoda, která by v reálném čase dokázala najít optimální řešení a byla by použitelná pro všechny typy případů.

V současné době ve většině firem je tendence snižovat náklady a jedním z prostředků ke snižování nákladů u firem, které vykonávají velký objem služebních cest je optimalizace trasy. Nejen z tohoto důvodu je téma optimalizace tras aktuální, proto se touto problematikou zabývá i tato bakalářská práce, jejíž hlavní náplní je vývoj aplikace umožňující zaměstnancům firmy E.ON optimalizovat trasy cest vykovávané v rámci jejich činnosti.

2 Cíle práce

Při odečítání spotřeby energie u spotřebitelů musí zaměstnanci poskytovatele určeni pro shromažďování dat, zkráceně nazývaní „odečítači“, obejít jednotlivá měřící odběrná místa. V měřícím místě může být odběr více druhů energie. Vzniká tak klasický problém optimalizace cesty.

Úkolem je z dat od informačního systému nalézt nejvhodnější trajektorii pro pohyb pracovníka tak, aby pokryl příslušná odběrová místa a zároveň, aby trasa byla optimalizovaná z hlediska délky a významu dat.

Cílem je:

1. Nalézt optimalizační metodu a uplatnit ji.
2. Vytvořit software, který bude používat navrženou metodu.
3. Provést jednoduchý test řešení.

Cílem je na základě zadaných dat o zákaznících firmy E.ON vytvořit aplikaci, která bude zobrazovat optimalizované trasy. Tato aplikace je vhodná pro terénní pracovníky firmy E.ON, jejichž náplní práce je kontrola elektroměrů a plynoměrů. V současné době je tato aplikace vyvíjena jako testovací pro desktopové počítače s operačními systémy Windows a do budoucna je možné tuto aplikaci modifikovat i pro mobilní zařízení.

3 Přehled současného stavu řešené problematiky

3. 1 Charakteristika firmy

Tato bakalářská práce se zabývá vývojem aplikace na základě požadavků a zadaných dat firmy E.ON.

E.ON byl založen v červnu roku 2000 spojením VEBA a VIAG, dvou významných německých průmyslových skupin, z nichž každá měla impozantní historii. [10]

VEBA a VIAG byly založeny ve 20. letech minulého století jako holdingové společnosti státních průmyslových podniků. V 60. a 80. letech byly tyto společnosti zprivatizovány a jejich úspěch pokračoval. Po spojení E.ON provedl rozsáhlé zaměření strategií a dnes je jedním z celosvětově největších investorem vlastněných energetických společností. V současné době je jednatelem Michael Fehn a Lorenz Pronnet. [10]

E.ON Česká republika, s. r. o. je vedoucí společností pro následující čtyři právnické subjekty [11] :

- E.ON Energie, a. s.
- E.ON Distribuce, a. s.
- E.ON Trend s. r. o.
- E.ON Servisní, s. r. o.

3. 2 Okružní dopravní problém

Okružní dopravní problém neboli problém „obchodního cestujícího“ je jeden z nejznámějších kombinatorických problémů. Jednou z příčin je jeho historie, která sahá až do roku 1759, kdy se švýcarský matematik Leonhard Euler začal zabývat touto problematikou. Největší zájem o tuto problematiku však nastal s postupem rozvoje lineárního programování. Populárnost problematiky obchodního cestujícího však neustává ani v dnešní době díky množstvím praktických aplikací například v logistice, krystalografii, plánování a mnohých dalších. [1]

Definice Problému

Problém je definován takto: máme několik bodů B na mapě (měst), které jsou definovány prvky x, y a potřebujeme zajistit, aby obchodní cestující navštívil tyto body pouze jednou, vrátil se do místa, odkud vyrazil a urazil přitom mezi nimi co možná nejkratší vzdálenost (viz [2]).

Problém je možno definovat také pomocí grafů. Základem je úplný graf s nejméně třemi body, kterými je nutno projít (pro méně bodů by nemělo smysl se problémem zabývat). Graf musí mít kladně váhově ohodnoceny hrany a cílem je získat hamiltonovský cyklus takový, aby součet ohodnocených hran byl co nejmenší (viz [3]).

Dále je nutno zabývat se orientováním hran (zda je ulice jednosměrná či nikoli), pokud jsou některé hrany grafu orientované a jiné ne, řešení úlohy je NP-těžké. [9]

3. 3 Přehled metod

Problematika obchodního dopravního cestujícího nemá snadné řešení. V podstatě neexistuje žádná efektivní metoda, která by řešila daný problém. Pro nalezení řešení se používají takzvané aproximační neboli heuristické metody. Tyto metody nemusí řešit danou úlohu s optimálním výsledkem, avšak pro praktické použití jsou tyto výsledky uspokojivé. Jejich výhodou je oproti exaktním metodám jejich rychlost výpočtu a hlavně jejich obecnost a možnost flexibilních úprav, které zajišťují, že můžeme danou metodu použít nejen pro řešení okružního dopravního problému, ale i pro jiné typy úloh. [9]

Ve většině případů (například [9], [19]) jsou aproximační metody rozdělovány tímto způsobem:

- Metody, které vytvářejí řešení (vytváří řešení od úplného počátku) :
    - Sekvenční – pracovat začínají v jednom počátečním bodě a odtud dále řešení rozšiřují.
    - Paralelní – pracovat začínají v několika počátečních bodech najednou a postupně spojují daná řešení v jedno, oproti sekvenčním řešením jsou pomalejší a složitější, ale jejich výsledky jsou přesnější.
- Metody, které zlepšují řešení – na počátku dostanou již nějaké zhotovené řešení a toto řešení postupně upravují k lepšímu výsledku.

3. 3. 1 Hladové algoritmy

Také nazývané „greedy algorithms“ jsou algoritmy, které pracují na velmi jednoduchých principech. „Ukousnou si to nejlepší sousto“, například nejvýhodnější hranu a tu „zkonzumují“ to může znamenat, že přidají hranu do okruhu nebo ne a dále se k této hraně nevracejí. Tyto algoritmy jsou hlavně výhodné pro svou rychlost. [16]

3. 3. 2 Grafové algoritmy

Tyto algoritmy se dají velice těžko používat pro orientované grafy, spíše se používají pro neorientované grafy a především použitím hladových algoritmů. [16]

3. 3. 3 Clark-Wrightova metoda

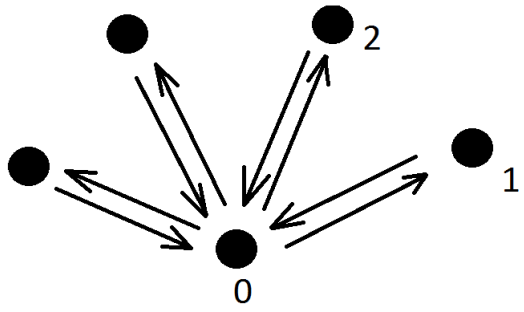
Metoda také nazývaná „Savings method“ nebo „Metoda výhodnostních čísel“ je jedna z nejpoužívanějších a zároveň nejstarších metod pro okružní úlohy. Tuto metodu navrhl Clark a Write [14] tak aby byla použitelná i pro trasovací problémy. Nejprve se označí počáteční uzel indexem 0 a dále se vypočítává výhodnostní číslo pro každou dvojici uzlů, například:

$$v_{12} = t_{10} + t_{20} - t_{12}$$

Kde t_{10} představuje vzdálenost mezi počátečním bodem a prvním následujícím bodem. t_{20} představuje vzdálenost mezi počátečním bodem a druhým následujícím bodem a t_{12} představuje vzdálenost od prvního následujícího bodu k druhému následujícímu bodu. v_{12} udává výhodnost dvojice uzlů.

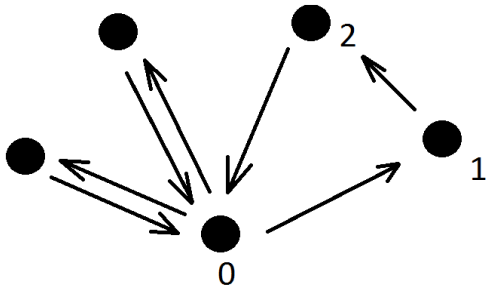
Podle výhodnostních čísel se trasy seřadí od největší po nejmenší a dále se v tomto pořadí zpracovávají a přidávají do okruhu, pokud je možné, aby jimi byl tvořen okruh. Nakonec vznikne cesta, která obsahuje všechny uzly, ale neprochází bodem 0, ten pouze stačí k dané trase přidat (viz [14])

Obrázek 1: Výhodnostní čísla 1. krok



Zdroj: [14]

Obrázek 2: Výhodnostní čísla 2. krok



Zdroj: [14]

3. 3. 4 Metoda nejbližšího souseda

Je to nejspíše nejjednodušší metoda pro okružní dopravní problém. Na počátku je určen počáteční bod, ze kterého se začíná a odtud se nalezne nejkratší hrana, která z tohoto bodu jde. Pokud je nalezena, postoupí se na konec této hrany do dalšího bodu a z něho se hledá znovu nejkratší hrana vedoucí k dosud nenavštívenému uzlu. Na konec je zavedena hrana z posledního bodu do počátečního. Pro zpřesnění této funkce a nalezení co nejoptimálnější trasy je výhodné tuto metodu aplikovat na všechny obsažené uzly a porovnávat výsledné hodnoty funkcí. [18]

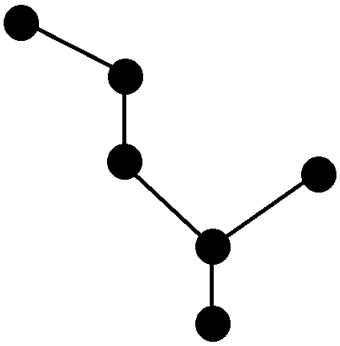
Tuto metodu podrobili bližšímu zkoumání Rosenkrantz, Stearns a Lewis [20] a zjistili, že na rozdíl od jiných, jimi studovaných metod, nelze určit jakýkoli odhad chyby jejich řešení.

3. 3. 5 Metoda minimální kostry

Pro nalezení minimální kostry v grafu existuje řada algoritmů. Na počátku je vybrán jeden z algoritmů nalezení nejmenší kostry. Po nalezení minimální kostry, viz Obrázek 3, se každá hrana kostry nahradí dvěma hranami, viz Obrázek 4, které jsou opačně orientované. V grafu

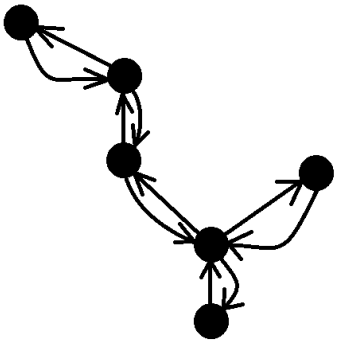
se nalezne orientovaný tah, který se transformuje tím, že se zachová pouze první výskyt uzlů a ostatní se vynechá, viz Obrázek 5. [15]

Obrázek 3: Minimální kostra 1. krok



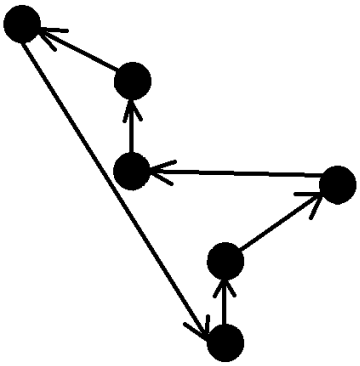
Zdroj: [9]

Obrázek 4: Minimální kostra 2. krok



Zdroj: [9]

Obrázek 5: Minimální kostra 3. krok



Zdroj: [9]

3. 3. 6 Vogelova metoda

Metoda také nazývána „loss method“, neboli „metoda ztrát“, nese pojmenování podle jejího tvůrce. Princip metody je takový, že nejprve vypočteme diference pro každý řádek a sloupec, to je rozdíl dvou nejmenších hodnot v řádku nebo sloupci. Z řady s největší diferencí vybereme nejmenší hodnotu, viz Tabulka 1, danou trasu zapíšeme a odstraníme řádek i sloupec. Trasy, které jsme již zařadili a trasy, které předčasně uzavírají okruh, také odstraníme. Znovu vypočítáme Vogelovy diference a znovu vybereme z řady s největší diferencí nejmenší hodnotu a takto opakujeme, viz Tabulka 2, až zůstanou poslední dvě trasy, které aplikujeme do okruhu, viz tabulka 3. [9]

Tabulka 1: Vogelova aproximační metoda 1. krok

	Praha	Brno	Ostrava	Plzeň	Řádkové diference
Praha	-	207	371	102	105
Brno	207	-	168	297	39
Ostrava	371	**168**	-	461	203
Plzeň	102	297	461	-	195
Sloupcové diference	105	39	203	195	

Zdroj: [9]

Ostrava -> Brno

Tabulka 2: Vogelova aproximační metoda 2. krok

	Praha	Brno	Ostrava	Plzeň	Řádkové diference
Praha	-	-	371	102	269
Brno	207	-	-	297	90
Ostrava	-	**168**	-	-	-
Plzeň	**102**	-	461	-	359
Sloupcové diference	105	-	90	195	

Zdroj: [9]

Plzeň -> Praha

Tabulka 3: Vogelova aproximační metoda 3. krok

	Praha	Brno	Ostrava	Plzeň	Řádkové diference
Praha	-	-	371	-	-
Brno	-	-	-	297	-
Ostrava	-	**168**	-	-	-
Plzeň	**102**	-	-	-	-
Sloupcové diference	-	-	-	-	

Zdroj: [9]

Ostrava -> Brno -> Plzeň -> Praha -> Ostrava

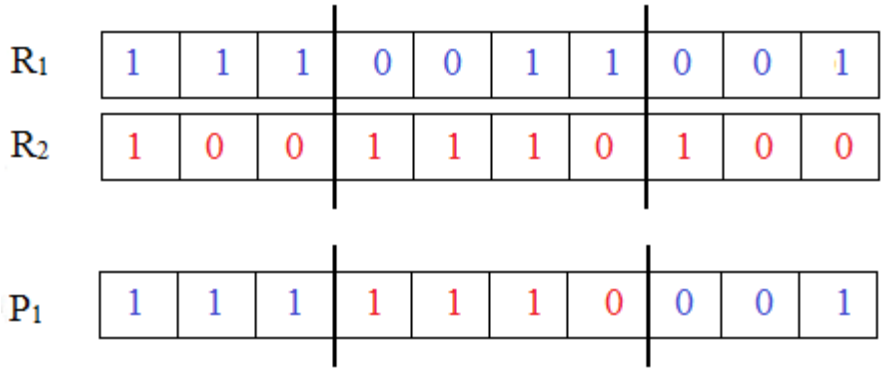
3. 3. 7 Genetické algoritmy

Genetický algoritmus popsaný v [17] patří do skupiny evolučních algoritmů. Je založen na principu přirozeného výběru. Vzorem pro genetický algoritmus jsou metody vývoje, které se vyskytují v přírodě. Základem je počáteční populace jedinců, každý jedinec má různé vlastnosti zakódované v jeho genech.

Algoritmus pracuje na principu křížení a mutace. Při křížení zdědí jedinec část genů od jednoho rodiče a část genů od druhého, viz Obrázek 6. Mutací u jedince dojde k náhodné změně některého genu nebo několika genů v genomu. Vyhodnocením jeho vlastností je zjištěno, jakou šanci má jedinec obstát v přirozeném výběru a zda vytvoří další generaci. Tento proces se neustále opakuje a jeho průběhem se pokaždé zlepšují genetické vlastnosti jedinců (viz [17]).

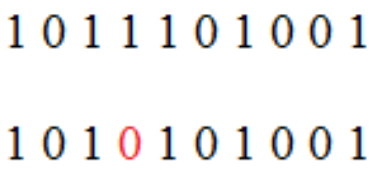
Pro selekci neboli výběr jedinců do následující populace může být použito různých metod. Například do nové populace vybrat jen určité procento s největší zdatností, tedy s jejich nejlepšími vlastnostmi z populace. K mutaci může dojít taktéž různými metodami, například pro genom zakódovaný do binárního čísla, vygenerujeme náhodné číslo, které bude udávat pozici bitu, jenž máme změnit. Jak je vidět na Obrázku 7, bylo nejdříve vygenerováno náhodné číslo, v tomto případě číslo 4 a změníme hodnotu čtvrtého bitu. [17]

Obrázek 6: Dvoubodové křížení



Zdroj: [17]

Obrázek 7: Mutace



Zdroj: [17]

4 Použité technologie

4. 1 OpenStreetMap

Projekt, který vytváří a distribuuje volně k dispozici geografická data pro celý svět. Tento projekt vznikl na základě toho, že většina map, o kterých se předpokládá, že jsou volně k dispozici, mají ve skutečnosti právní nebo technické omezení jejich použití. Omezují tedy uživatele v jejich používání kreativními, produktivními nebo neočekávanými způsoby. OpenStreetMap je svobodná, citovatelná mapa celého světa. Na rozdíl od proprietárních datových sad jako je Google Map Maker, licence OpenStreetMap umožňuje volný přístup k úplné datové sadě mapy. Toto masivní množství dat lze volně stáhnout v plné výši. [4]

Hlavní cestou je, že sami uživatelé se účastní na úpravě map. S volným uživatelským účtem lze provádět zlepšení mapy tak, že problémy lze opravit a přidat data, která jsou viditelná pro každého. [4]

OpenStreetMap financování a infrastruktura je podporována neziskovou organizací OpenStreetMap Foundation. I když nemá práva na vlastnictví datové sady nebo kontrolu projektu, nadace podporuje růst, vývoj a šíření svobodných geografických dat k užívání a sdílení pro každého. [4]

OpenStreetMaps byla založena v roce 2004 Steavem Coastem. Zpočátku se zaměřovala na zmapování Velké Británie. V dubnu 2006 byla založena nadace OpenStreetMap Foundation na podporu růstu, rozvoje a šíření svobodných geografických dat a poskytnutí geoprostorových údajů komukoli k užívání a sdílení. Od prosince 2006 OpenStreetMap využívá leteckého snímkování jako podklady pro produkci map. [4]

Kód, který běží na openstreetmap.org se skládá z nezávislých komponent, které pracují společně, aby poskytly API, Slippy Map a další funkcionality. Data OpenStreetMap jsou uložena v POstgreSQL a POSTGIS a překládána pomocí Mapnik do mapových dlaždic. Slippy Map interface pro tyto dlaždice, který umožňuje posouvat a přibližovat mapu, je poháněn Leaflet. [4]

4. 2 Microsoft Visual C#

Microsoft Visual C# je výkonný, ale jednoduchý jazyk zaměřený především na vývojáře vytvářející aplikaci pomocí Mictosoft .NET Framework. Zdědil mnoho z nejlepších vlastností C++ a Microsoft Visual Basic, ale jen málo z nesrovnalostí a anachronismů, což

zajišťuje, že se jedná o čistší a logičtější jazyk. C# 1.0 byl představen veřejnosti v roce 2001. Nástup C# 2.0 s Visual Studiem 2005 přinesl několik nových důležitých funkcí, například Generics, iterátory a anonymní metody. C# 3.0, který byl vydán společně s Visual Studiem 2008 přidal rozšířené metody, lambda výrazy a ze všech nejvíce známý Language Integrated Query, neboli LINQ. Poslední inkarnace jazyka C 4.0 poskytuje další vylepšení, která zlepšují jeho interopterabilitu s jinými jazyky a technologiemi. Tyto funkce podporují pojmenování, volitelné argumenty, dynamický typ, který naznačuje, že jazyk runtime by měl realizovat dynamickou vazbu pro objekt a rozptyl, který řeší některé problémy ve způsobu, jakým jsou generická rozhraní definována. C# 4.0 využívá verzi .NET Framework 4.0. Existuje mnoho doplňků k této verzi .NET Framework, ale pravděpodobně nejvíce významné jsou třídy typy, které tvoří Task Parallel Library (TPL). Použitím TPL lze vytvářet vysoce škálovatelné aplikace, které mohou plně využít více jádrové procesory velmi snadno a rychle. Podpora webových služeb a Windows Conunication Foundation (WCF) byla rovněž rozšířena, lze vytvářet služby, které se řídí REST modelem, stejně jako více tradičním SOAP schématem. Vývojové prostředí Microsoft Visual Studio 2010 umožňuje používat všechny tyto výkonné funkce velmi snadno a rychle.[5]

4. 3 Microsoft Visual Studio 2010

Visual Studio 2010 je programovací prostředí velmi bohaté na nástroje, obsahující funkce, kterými lze vytvářet velké či malé C# projekty. Lze v něm dokonce vytvářet projekty, které bez problémů budou kombinovat moduly napsané pomocí různých programovacích jazyků, jako je C++, Visual Basic a F#. [5]

4. 4 Proč používat Visual Studio 2010?

Existuje mnoho důvodů, proč přejít k Visual Studiu 2010 Professional [8] :

- Vestavěné nástroje pro Windows 7, včetně multitouch a UI komponent.
- Nový bohatý editor, postavený na Windows Presentation Foundation (WPF), který lze snadno přizpůsobit, tak aby vyhovoval práci.
- Multimonitor podpora.
- Nové Quick Search, které pomáhá najít relevantní výsledky pouze rychlým zadáním několika prvních písmen všech metod, tříd nebo nastavení.

- Velká podpora pro vývoj a nasazení systému Microsoft Office 2010, SharePoint 2010 a Windows Azure aplikací.

- Více jádrová podpora, která umožňuje paralelizovat aplikace a specializovaný debugger, který umožňuje sledovat úkoly jednotlivých vláken.

- Zlepšení ASP.NET AJAX rámce a podpory jádra JavaScript IntelliSense.

- Podpora multitargeting / multiframework.

- Podpora rozvoje WPF a Silverlight aplikací se zvýšenou drag-and-drop podporou bindováním dat.

- Velká podpora pro Team Foundation Server (TFS) 2010 (a předchozí verze) pomocí Team Exploreru. To umožňuje používat údaje a zprávy, které jsou automaticky shromažďovány Visual Studiem 2010, což umožňuje sledovat a analyzovat stav projektů.

4. 5 OsmSharp

OsmSharp je open-source projekt se zaměřením na směrování a logistiku optimalizace pomocí OpenStreetMap (OSM). Mnoho funkcí a vlastností existuje pro zpracování OSM dat, například přístup k OSM API, převod OSM dat a mnoho dalších. Cílem OsmSharp je, aby aplikace založené na OSM byly jednodušší a efektivnější. OsmSharp by nemohl existovat bez OSM a má za cíl přispět k OSM, aby pomohl zlepšit a podpořit tento velký projekt. Směrování na OSM je možné a existuje mnoho řešení, avšak OsmSharp se snaží poskytnout snadné řešení, které je použitelné v celosvětovém měřítku nejen pro desktopové počítače, ale i pro mobilní zařízení. [6]

V současné době existují dvě možnosti jak směrovat pomocí OsmSharp [6] :

- Uchovávat všechny směrování dat v paměti. Tento proces vyžaduje nejméně 12 GB RAM pro země jako je Anglie nebo Německo. Tento druh směrování je velmi rychlý, protože všechny údaje jsou uchovávány v paměti, to je především ideální pro webové služby, kde je potřeba mnoho výpočtů v krátkém časovém horizontu.

- Uchovávat všechny směrování dat v databázi. To vyžaduje pouze databázi a může to být použito téměř na všech zařízeních. Data jsou uložena v mezipaměti a je s nimi

naloženo podle potřeby, ale tento druh směrování je mnohem pomalejší v porovnání s předchozí možností.

V této chvíli na mobilních zařízeních, které pracují offline je podpora špatná a pouze první možnost je k dispozici. Je-li prostředí omezeno na jednu určitou část, tak vše funguje bez problémů, ale pro větší plochy je potřeba nějaký nový vývoj. [6]

4. 6 Windows Presentation Foundation

Windows Presentation Foundation (WPF) je další generace prezentačního systému pro vytváření klientských aplikací pro systém Windows.

Jádrem Windows Presentation Foundatin je vektorový engine nezávislý na rozlišení, který je postaven na moderním grafickém hardwaru. Windows Presentation Foundation rozšiřuje jádro s komplexní aplikací pro rozvoj funkcí, které zahrnují Extensible Application Markup Language (XAML), ovládací prvky, bindování dat, uspořádání, 2-D a 3-D grafiku, animace, styly, šablony, dokumenty, média, texty a typografie. Windows Presentation Foundation je součástí Microsoft .NET Framework, takže lze vytvářet aplikace, které obsahují další prvky .NET Framework. [12]

4. 7 Bing Maps Windows Presentation Foundation (WPF) Control

WPF Control má vše, co se dá čekat od Bing Map ovládání včetně možnosti prezentovat informace prostřednictvím WPF, jako je [7] :

- Styly map:
    - Silniční
    - Letecká
    - Hybridní
- Možnosti umístit prvky na mapě skrz Latitude / Longitude:
    - Piny
    - Křivky
    - Mnohoúhelníky
- Navigace mapy pomocí pan a zoom kláves

Jeden z nejzajímavějších aspektů WPF Control je podpora Microsoft Surface, což znamená, že použití WPF v Surface aplikacích má nativní podporu pro dotykové funkce. [7]

4. 8 Nuget

Nuget je rozšíření Microsoft Visual Studia, které umožňuje snadno přidávat, odstraňovat a aktualizovat knihovny a nástroje třetích stran v projektech Microsoft Visual Studio, které využívají rozhraní .NET Framework. Při instalování balíčků pomoci Nuget, se soubory s knihovnami nakopírují do požadovaného řešení a automaticky aktualizuje vybraný projekt (přidá reference, mění konfigurační soubory, atd.). Při odstranění balíčku Nuget zruší jakékoli změny, to znamená, že v projektu nevznikne žádný nepořádek.[13]

5 Návrh řešení

5. 1 Cíle aplikace

V následujících bodech jsou popsány dílčí cíle, jejichž splnění je nutné k dosažení hlavního cíle aplikace:

- získat GPS souřadnice ze zadaného textového souboru;
- zjistit aktuální pozici a počet míst, které chce uživatel obejít;
- nalézt nejbližší body od zadaných souřadnic;
- vypočítat trasy mezi nalezenými body tak, aby došlo ke zvýšení efektivity plánování cest a tedy k úspoře pohonných hmot v případě jízdy automobilem nebo k úspoře ujitých kroků v případě chůze;
- konečná data se následně budou zpracovávat do finální podoby, aby se v nich uživatel dokázal orientovat, a aby uživatel mohl aplikaci snadno obsluhovat.

5. 2 Vstupy aplikace

5. 2. 1 Vstupní data

Na Obrázku 8 je zobrazena část z interních dat firmy E.ON. Tato data tvoří hlavní vstupy aplikace, jedná se o GPS souřadnice všech míst, které musí pracovník firmy E.ON navštívit.

Obrázek 8 :Vstupní data

```
DEMELOVA              SAP;4928.3267;1706.7197   OPTIKA
DUKELSKA BRANA        SAP;4928.3178;1706.7983
NAM. T. G. MASARYKA   SAP;4928.2690;1706.6503   U DOLAKA NA DVORE ZA OP
KRAMARSKA             SAP;4928.2794;1706.8450   NOVOSTAVBA ZA VELODROMEM
PERNSTYNSKE NAM.      SAP;4928.2474;1706.6576
KRAVAROVA             SAP;4928.3027;1706.6820
FUGNEROVA             SAP;4928.2933;1706.8152   VYTAPENI BETONU-VLASTNI STAVBA
```

Zdroj: Interní data firmy E.ON

Vstupní GPS souřadnice jsou ve formátu DD MM.MMMM, například (48°28.2474'), bylo tedy nutné jejich převodu na formát DD.dddddd, například (48.47079°).

5. 2. 2 Formulář

Na Obrázku 9 je zobrazen malý formulář kde má uživatel možnost zadat počet míst, které chce navštívit. Dále je zde možnost zadat GPS souřadnice výchozího bodu (aktuální pozice) a výběr dopravního prostředku. Dále je na výběr metoda, která bude použita pro nalezení

trasy mezi jednotlivými body a zadání doby která je potřeba na zpracování jednoho místa odečítání.

Obrázek 9: Vstupní formulář

Zdroj: Vlastní práce

Pro testovací účely uživatel musí zadat GPS souřadnice jeho polohy ručně, až bude aplikace vyvíjena pro mobilní zařízení, nebude zde muset uživatel zadávat aktuální pozici ručně, protože GPS souřadnice aktuální polohy budou získány z mobilního zařízení.

5. 2. 3 File Dialog

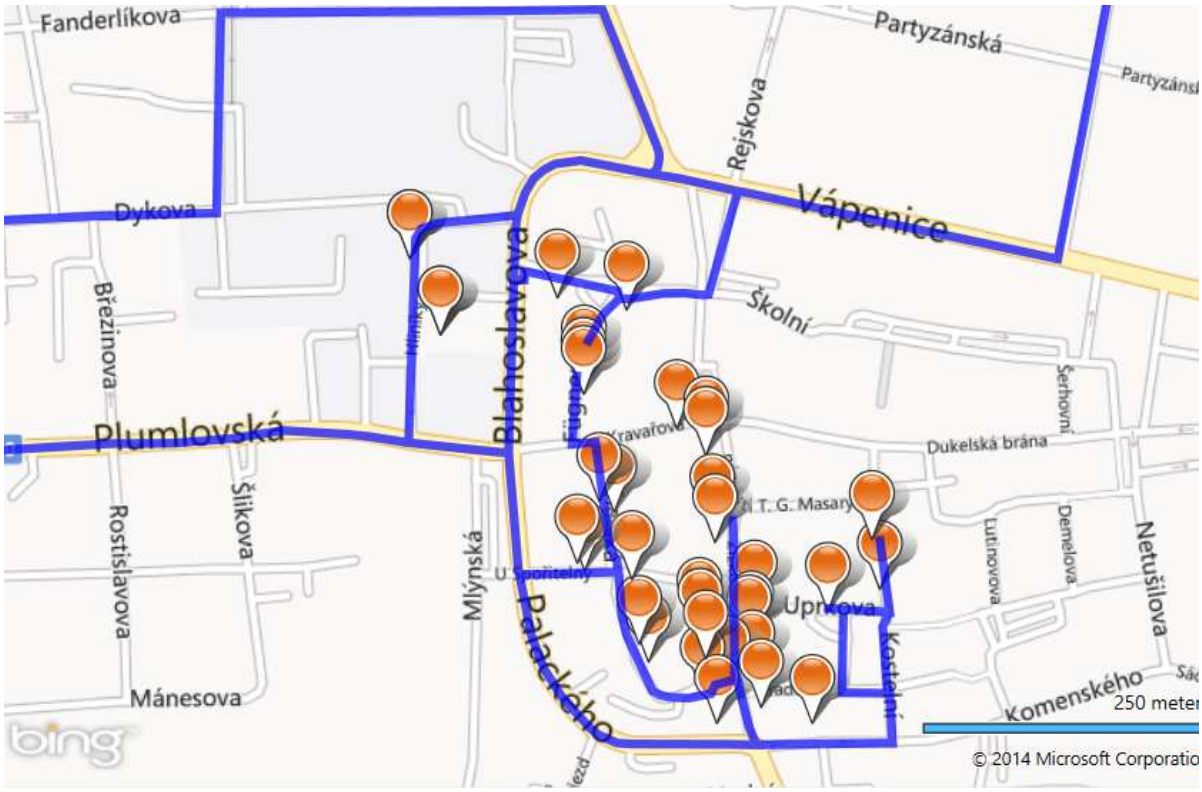
Pomocí Open File Dialogu má uživatel možnost si vybrat data ve formátu .pre, která byla získána od firmy E.ON nebo data ve formátu .txt, které generuje samotná aplikace. Pokud již uživatel navštívil několik míst, aplikace uloží zbytek nenavštívených míst ve formátu .txt.

5. 3 Výstupy

5. 3. 1 Optimalizovaná trasa

Hlavním výstupem je optimální zobrazení trasy na mapě, jak je vidět na Obrázku 10, trasa je vykreslena modrou barvou a na trase jsou znázorněny jednotlivé body, které musí odečítač navštívit.

Obrázek 10: Optimalizovaná trasa



Zdroj: Vlastní práce

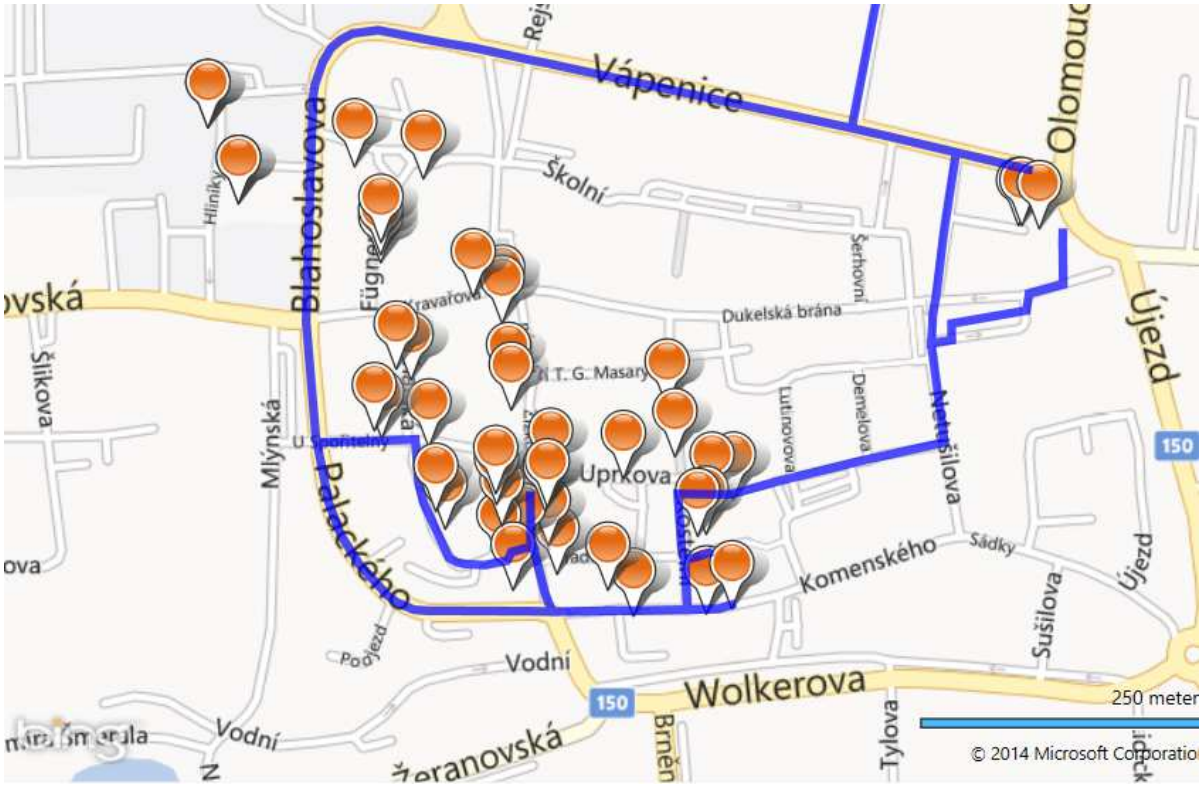
Pro konečné zobrazení trasy byly použity Bing Mapy. Dalším dílčím výstupem je zobrazení detailnějších informací o jednotlivých místech, které má uživatel možnost zobrazit při kliknutí na konkrétní bod. Zobrazení informací o bodech je umístěno pod mapou z důvodů nepřekrývání mapy a tedy k její přehlednosti.

Dalším výstupem je celková délka trasy. Pro porovnání je zde zobrazena délka trasy získaná z metod knihoven OSMSharp, metodou Haversine, metodou Spherical Law of Cosines a metodou Equirectangular approximation, které jsou detailněji popsány v kapitole 5. 6. 2.

5. 3. 2 Nenavštívené body

Dalším dílčím výstupem je zobrazení nenavštívených bodu z toho důvodů, aby měl uživatel aplikace přehled, kolik mu ještě zbývá obejít míst, toto je znázorněno na Obrázku 11.

Obrázek 11: Nenavštívené body



Zdroj: Vlastní práce

Stejně jako je tomu u bodů, které jsou zobrazeny na trase, tak i u nenavštívených bodů je možnost po kliknutí na konkrétní bod zobrazit podrobnější informace o daném místě.

5. 3. 3 Ukládání stavu

Dalším dílčím výstupem je ukládání nenavštívených bodů. Pokud si uživatel aplikace již nechal vykreslit nějakou trasu na mapě a obešel všechna místa na zobrazené trase, má možnost si tento stav uložit. Uložení stavu ukládá aplikace do textového souboru. V textovém souboru nejsou zapsána místa, která již byla navštívena, ale pouze místa která dosud navštívena nebyla. V tomto textovém souboru se o těchto bodech ukládají pouze GPS souřadnice ve formátu DD.dddddd° a podrobnější informace, které slouží k zobrazení informací při kliknutí na konkrétní bod.

5. 4 Použité technologie

5. 4. 1 Microsoft Visual Studio 2010

Tento nástroj slouží jako vývojové prostředí pro tvorbu aplikací pro programovací jazyk C# a prostředí .NET. Microsoft Visual Studio 2010 bylo vybráno především z toho důvodu, že

licence pro toto vývojové prostředí byla získána prostřednictvím Microsoft Developer Network Academic Aliance, která poskytuje studentům a členům Ústavu aplikované informatiky licence pro různé programy. Dalším důvodem je velké rozšíření platformy .NET a možnost modifikace aplikace pro mobilní zařízení s operačními systémy Windows Phone 7 nebo Windows Phone 8 při minimálních nákladech. Ačkoli je na trhu k dispozici Microsoft Visual Studio 2012 a rovněž je zde možnost získání licence prostřednictvím Microsoft Developer Network Academic Aliance, bylo zvoleno vývojové prostředí Microsoft Visual Studio 2010 z důvodů volby knihoven projektu OsmSharp, které mají prozatím stabilní verzi vydanou pouze pro Microsoft Visual Studio 2010 a pro 64 bitový operační systém.

5. 4. 2 OpenStreetMap

Hlavní kritérium, které využívané mapy pro výpočet optimalizované trasy musely splňovat je, aby byly volně šiřitelné a aktuální. OpenStreetMap byly zvoleny pro jejich rozšíření, jak v rámci uživatelů a jejich možnosti svobodné editace, tak především pro jejich rozšíření mezi vývojáři. To zajišťuje aktuálnost dat a významnou podporu open source projektů a knihoven využívajících OpenStreetMap.

5. 4. 3 OsmSharp

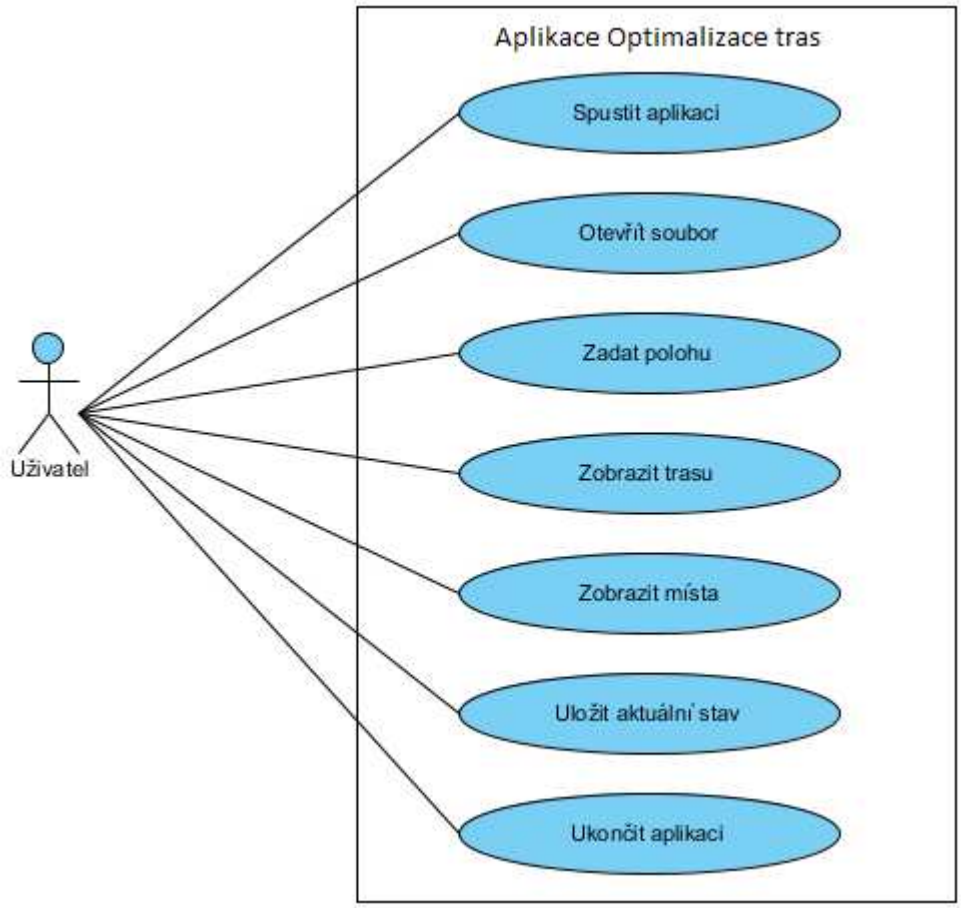
Jeho výběr plyne hlavně z výběru Microsoft Visual Studia 2010 a výběru OpenStreetMap, takže bylo nutné hledat mezi projekty pracující v prostředí .NET a využívající OpenStreetMap. Dále jeho výběr plyne z toho, že je to pravděpodobně jediný open source projekt splňující předchozí dvě kritéria a zabývající se problematikou obchodního cestujícího. Knihovny tohoto open source projektu dokážou nejen vypočítat optimalizovanou trasu mezi dvěma body, ale také najít nejkratší cestu mezi více body.

5. 4. 4 Bing Maps

Původně byly hledány offline vektorové mapy pro vykreslování výstupní optimalizované trasy nebo knihovny, které by dokázaly vykreslovat OpenStreetMap z xml souboru. Po četných potížích, jak při nalezení knihoven vykreslujících OpenStreetMap, tak při vykreslování vektorových map byly zvoleny Bing Maps. Bing Maps byly zvoleny hlavně pro jejich snadnou implementaci do prostředí Windows Presentation Foundation.

5. 5 Diagram užití

Obrázek 12: Use case diagram



Zdroj: Vlastní práce

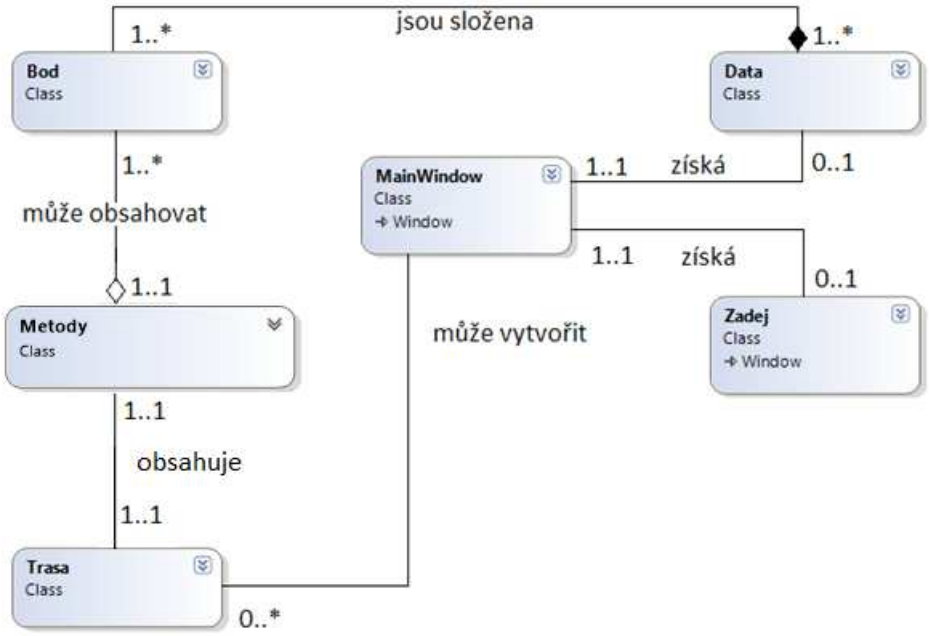
Uživatel má možnost spustit a ukončit aplikaci, dále má možnost otevřít soubor, ve kterém jsou uloženy záznamy o zákaznících firmy E.ON ve formátu .pre nebo pokud má uložené nějaké rozpracované soubory ve formátu .txt. Je zde možnost zadat aktuální pozici a počet míst, které chce obejít. Pokud byly tyto kroky učiněny správně, je zde možnost vykreslení trasy na mapě. Pokud byla načtena data, má uživatel možnost vykreslit zbývající místa, která ještě nenavštívil. Po dokončení trasy má uživatel k dispozici možnost uložit aktuální stav a dále pracovat s místy, která dosud nenavštívil.

5. 6 Diagram tříd

Základní navržený diagram tříd s relacemi je znázorněn na Obrázku 13 a kompletní diagram tříd se všemi obsaženými metodami je zobrazen na Obrázku 14. K jeho zpracování se

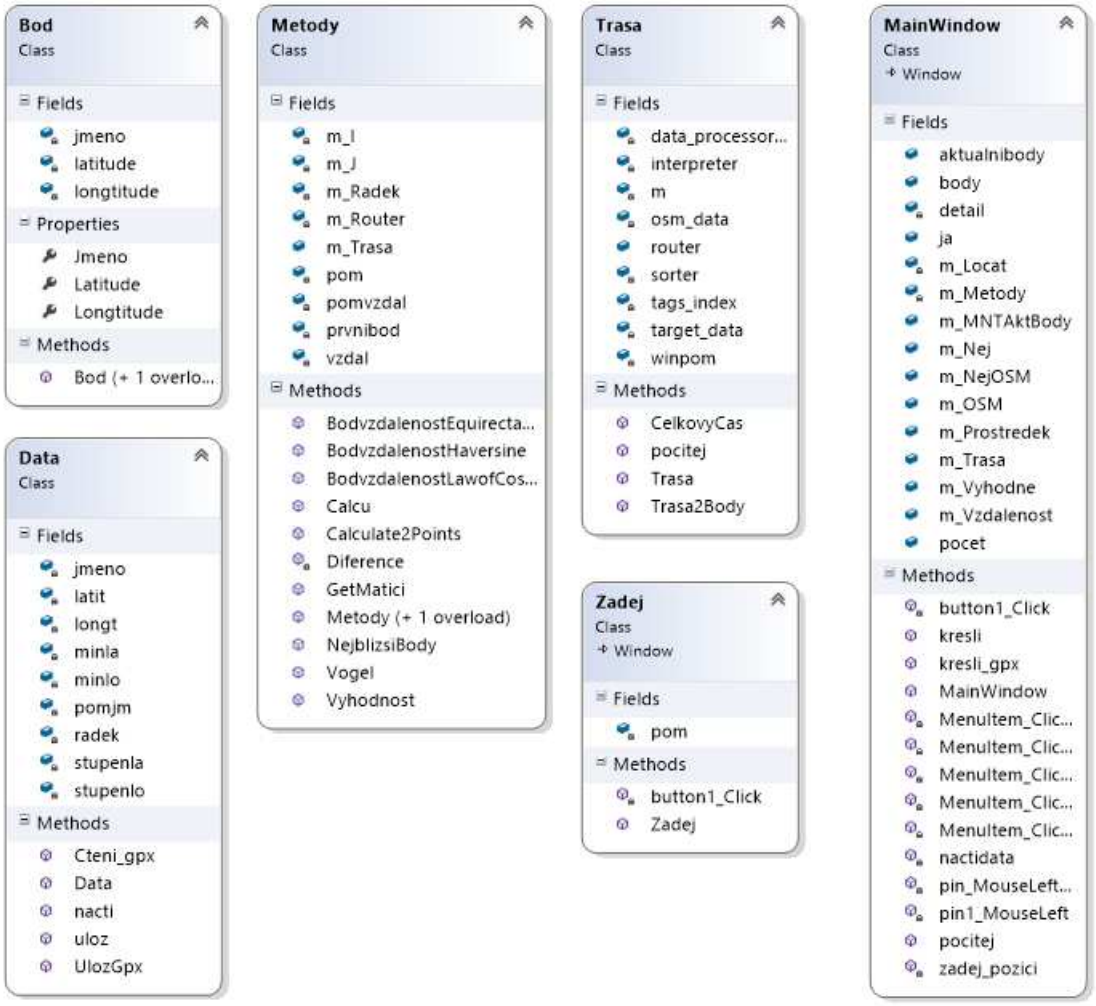
vycházelo z objektů a jejich metod, které se v projektu budou vyskytovat. Je zde vidět objekt Bod, Trasa, Data, Metody a dále dva formuláře, se kterými bude pracovat uživatel.

Obrázek 13: Class diagram s relacemi



Zdroj: Vlastní práce

Obrázek 14: Class diagram kompletní



Zdroj: Vlastní práce

5. 6. 1 Třída Bod

Tato třída obsahuje tři členské proměnné, které slouží jako definice konkrétního bodu. Obsahuje zeměpisnou šířku a výšku. Dále obsahuje jméno, které se zobrazuje při kliknutí na Pushpin.

5. 6. 2 Třída Metody

Tato třída obsahuje porovnávané metody k nalezení nejkratší trasy mezi zadaným počtem bodů. Dále pak metody pro nalezení nejkratší trasy mezi dvěma body vzduchem, jejichž výpočet je popsán níže. Metodu z projektu OSMSharp, která také získá vzdálenost mezi dvěma body, ale u níž je možnost zadat dopravní prostředek. Získaná délka trasy touto metodou není vzduchem jako v předchozích případech, ale za použití vybraného dopravního prostředku, případně pěšky. Z této metody lze rovněž určit dobu trvání cesty mezi dvěma body.

Vzdálenost mezi dvěma body vzduchem

K výpočtu vzdálenosti mezi dvěma body na Zemi je zapotřebí vypočítat vzdálenost mezi dvěma body na kružnici. A=| φ_1; φ_2|, B=|λ_1; λ_2|. Je zapotřebí znát poloměr země R a pomocí vzorce se vypočítá vzdálenost d mezi dvěma body. Kde φ je zeměpisná šířka, λ je zeměpisná délka a R je poloměr země, tedy 6371 km.

Haversine

$a=\sin^2(\Delta\varphi/2)+\cos(\varphi_1).\cos(\varphi_2).\sin^2(\Delta\lambda/2)$

$c=2.\text{atan2}(\sqrt{a},\sqrt{(1-a)})$

$d = R.c$

Spherical Law of Cosines

$d = \text{acos}(\sin(\varphi_1).\sin(\varphi_2) + \cos(\varphi_1).\cos(\varphi_2).\cos(\Delta\lambda)).R$

Equirectangular approximation

$x=\Delta\lambda.\cos(\varphi)$

$y=\Delta\varphi$

$d = R.\sqrt{x^2 + y^2}$

5. 6. 3 Třída Data

Tato třída slouží pro práci s daty. Obsahuje metodu „nacti“, která zajišťuje načtení dat ze souboru .pre nebo .txt. Dále obsahuje metodu „uloz“, která zajišťuje uložení rozpracované práce do souboru .txt. Obsahuje metodu „Cteni_gpx“, která umožňuje rozparsování xml souborů, ve kterých je uložena trasa a seřazené body, které má pracovník ještě obejít. Také je zde metoda „UlozGpx“ pro uložení těchto souborů.

5. 6. 4 Třída MainWindow

Třída MainWindow představuje hlavní okno pro zobrazení Bing Map, pro vykreslení trasy a zobrazení jednotlivých bodů v této mapě. Dále obsahuje veřejné statické proměnné určující aktuální pozici a počet bodů, které chce uživatel navštívit. Dále obsahuje list bodů, které jsou obsaženy v zadaném souboru dat a druhý list, ve kterém jsou uloženy body, jenž jsou zrovna vykresleny na mapě.

5. 6. 5 Třída Zadej

Třída Zadej představuje formulář, ve kterém uživatel zadá aktuální pozici a počet míst, která chce navštívit. Pracuje se statickými proměnnými, které jsou uloženy v třídě MainWindow. Dále obsahuje typ dopravní ho prostředku a vybranou metodu. Vybraná metoda bude použita pro výpočet trasy mezi zadanými body.

5. 6. 6 Třída Trasa

Tato třída obsahuje metody projektu OSMSharp, pro výpočet vzdálenosti mezi jednotlivými body s vybraným dopravním prostředkem a dobu, která je potřeba na obejití vypočítané trasy. Dále obsahuje výpočet celé trasy pomocí metody CalculateTSP, která využívá pro výpočet genetický algoritmus.

6 Implementace

6. 1 Příprava technologií

6. 1. 1 Příprava Microsoft Visual Studia 2010

Na počátku bylo zapotřebí nainstalovat vývojové prostředí Microsoft Visual Studio 2010, které bylo získáno prostřednictvím Microsoft Developer Network Academic Aliance. Dále už jen doinstalovat do tohoto vývojového prostředí správce balíčků Nuget. n.

6. 1. 2 Příprava OsmSharp

Nejprve bylo nutné opatřit knihovny, které lze použít k vývoji aplikace ve vývojovém prostředí Microsoft Visual Studio 2010 na platformě Windows Presentation Foundatin. Tato poslední stabilní verze knihoven byla stažena prostřednictvím správce balíčků Nuget. Na počátku byla aplikace vyvíjena na 32 bitovém operačním systému Microsoft Windows 7 Professional x86, na kterém bylo zjištěno, že některé metody z použitých knihoven OsmSharp nefungují tak, jako byla prvotní představa nebo nefungují vůbec. Prostřednictvím různých diskusních fór bylo zjištěno, že je potřeba aplikaci vyvíjet na 64bitovém operačním systému. Z tohoto důvodu bylo pro kontrolu, zda jsou vůbec tyto knihovny použitelné pro vývoj aplikace, nainstalováno virtuální prostředí s operačním systémem Microsoft Windows 8 Professional x64, na kterém byly testovány metody knihoven OsmSharp. Po zjištění, že metody těchto knihoven fungují a jsou použitelné pro další vývoj aplikace, následovalo přeinstalování stávajícího operačního systému Microsoft Windows 7 Professional x86 systémem Microsoft Windows 8 Professional x64. Tento krok byl proveden z důvodu rychlejšího vývoje a postupu práce, protože vývoj ve virtuálním prostředí byl pomalý a neefektivní.

6. 1. 3 Příprava OpenStreetMap

Prvním krokem bylo stažení mapy České republiky ve formátu xml. Tyto mapy byly staženy z Projekty z České OpenStreetMap (http://osm.kyblsoft.cz/), kde se každý den aktualizuje mapa České republiky. Vzhledem k velikosti souboru xml, který charakterizuje mapu České republiky, bylo nutné pro využití směrování dat s využitím operační paměti nikoli databáze pro budoucí modifikaci projektu pro operační systémy Windows Phone, vyfiltrovat jen důležitá data, která se budou nadále používat prostřednictvím knihoven OsmSharp. Pro představu, velikost xml souboru České republiky je přibližně 7, 2 GB. Pro vyfiltrování byl použit program Osmosis, který dokáže parsovat OpenStreetMap ve formátu xml. Pro

filtrování bylo nutné zadat příkaz v příkazovém řádku pod právy správce, který vypadá například takto:

osmosis –read-xml mapcz.osm –bounding-box top=“49.49“ left=“16.55“ bottom=“49.12“ right=“17.29“ –way-key keyList=“highway“ –used-node –write-xml map.osm

Tímto příkazem byl vytvořen ze zdroje „mapcz.osm“ výřez zadaných souřadnic. Klíčem „highway“ bylo zaručeno získání pouze pozemních komunikací, tímto krokem se z původní velikosti 7, 2 GB vytvořila mapa map.osm o velikosti 983 KB. Tato velikost je více než dostačující jak pro načtení do operační paměti mobilního zařízení, tak pro osobní počítače.

6. 1. 4 Příprava Bing Map

Dále bylo nutné zajistit podporu Bing Maps v prostředí Windows Presentation Foundation. Toto bylo zajištěno stažením a vývojového kitu (SDK) Bing Maps Windows Presentation Foundation (WPF) Control, Version 1.0 z oficiálních stránek (www.microsoft.com). Po nainstalování bylo nutné přidat do projektu referenci na knihovny využívající Bing Mapy, konkrétně Microsoft.Maps.MapControl.WPF.dll. Dále bylo nutné do hlavního okna projektu vložit ovladač mapy. To bylo zajištěno přidáním příkazu:

```
xmlns:m="clr-namespace:Microsoft.Maps.MapControl.WPF;
assembly=Microsoft.Maps.MapControl.WPF"
```

Tento příkaz se vložil do xaml popisující hlavní okno. Dále bylo nutné založit Bing Maps účet na stránkách (www.bingmapsportal.com), kde bylo nutné vygenerování Bing Maps unikátního klíče, který je nutný vložit do aplikace. Tento klíč se vložil do aplikace pomocí následujícího příkazu:

```
<Grid>
    <m:Map CredentialsProvider="ZDE VLOŽTE MAPS KLÍČ "/>
</Grid>
```

Toto je jako v předchozím případě rovněž implementováno do xaml popisující hlavní okno.

6. 2 Vývoj aplikace

6. 2. 1 Parsování dat

Prvním krokem ve vývoji aplikace bylo získání dat v takovém formátu, aby byla ulehčena práce s nimi. Nejprve bylo nutné přečíst zadaná interní data od firmy E.ON ve formátu .pre.

```
                latit = Convert.ToDouble(new string(minla));
                latit = latit / 60;
                latit = stla + latit;
                longt = Convert.ToDouble(new string(minlo));
                longt = longt / 60;
                longt = stlo + longt;
                latit = 0;
                longt = 0;

            body.Add(new Bod(jmeno, latit, longt));
```

V první fázi se otevře FileDialog pro zadání buď souboru .pre se zadanými daty, nebo souboru .txt s rozpracovanými daty. Pokud FileDialog skončí s úspěšně vybraným souborem, pak dále pokračuje deklarace pomocných proměnných, které se budou dále využívat. Pokud není nic vybráno, tak metoda skončí.

Za předpokladu, že byl vybrán soubor, se pomocí StreamReader otevře soubor, který uživatel vybral ve FileDialogu a jeho cesta je uložena v dlg.FileName. Následně se čte soubor po řádcích.

Ze zadaného souboru bylo zjištěno, že jméno konkrétního bodu má počátek jako 279. znak a jeho rozsah je 30 míst. Z tohoto důvodu je zde cyklus, který prochází řádek od místa 279 až do místa 308 a každý znak jednotlivě ukládá do pole charů „pomjm“ a po dokončení cyklu se převedou znaky pole „pomjm“ do stringu „jmeno“.

Dále se pomocí cyklu získávají latitude minuty tak, že se prochází řádek od pozice 457 do pozice 463, pokud jsou na těchto místech čísla, zapíšou se do pole „minla“ a pokud je na těchto pozicích mezera, zapíše se do pole „minla“ nula. Pokud se v cyklu narazí na desetinnou tečku, tak je nahrazena desetinou čárkou. Toto se provádí pro pozdější převod ze tring do double. Tento postup se opakuje i pro nalezení longitude minut, ale řádek se prochází na jiných místech.

Následně se získávají z daného řádku stupně zeměpisné šířky a délky, pokud jsou na místech, kde by se měli nacházet stupně pouze mezery, zapíše se do zeměpisné šířky

a délky nula. Pokud zde mezery nejsou, po znacích se přepíší stupně do pomocného pole, a poté se převede pole do pomocné proměnné buď „stla“ nebo „stlo“.

Dále je nutné převést minuty na stupně, to je provedeno tím způsobem, že se minuty vydělí šedesáti a přičtou se k získaným stupňům. To je výsledná zeměpisná šířka nebo výška, se kterou se dále bude pracovat.

Nyní už zbývá poslední krok a to uložit jméno, zeměpisnou šířku a délku do veřejného statického listu, který má jako datový typ třídu Bod, která obsahuje pouze jméno ve formátu string, zeměpisnou šířku a délku ve formátech double.

6. 2. 2 Metoda nejbližšího souseda

Toto je metoda, kterou bylo potřeba do práce implementovat, protože před samotným nalezením postupu pro optimalizování jednotlivých tras bylo nutné nejdříve nalézt body, mezi kterými se bude optimalizovaná trasa zpracovávat. Cílem této metody je získat zadaný počet bodů od aktuální pozice tak, aby se mezi nimi utvořila nejkratší cesta.

```
for (int i = 0; i < MainWindow.pocet; i++)
 {
   vzdal = double.MaxValue;
   for (int j = 0; j < MainWindow.body.Count; j++)
   {
     bool shoda = false;
     foreach (Bod akt in MainWindow.aktualnibody)
     {
       if (akt == MainWindow.body[j]) shoda = true;
     }
     if ((MainWindow.body[j].Latitude != 0) && (shoda == false))
     {
       pomvzdal = BodvzdalenostHaversine(MainWindow.body[j], prvnibod);
       if (pomvzdal < vzdal)
       {
         vzdal = pomvzdal;
         pom = MainWindow.body[j];
       }
     }
   }
   prvnibod = pom;
   MainWindow.aktualnibody.Add(prvnibod);
 }
```

Princip metody je následující: nejprve je potřeba naleznout nejbližší bod od aktuální pozice, která je ukryta v instanci třídy „prvnibod“. Pomocí cyklu for jsou procházeny všechny dosud nenavštívené body, které jsou ukryty v listu „body“. Mezi každým bodem a aktuální pozicí je měřena vzdálenost, která je vypočítána v metodě „bodvzdalenost“, jenž používá metodu Haversine pro výpočet vzdálenosti. Tato metoda je popsána v kapitole 5. 6. 2 a vrací pouze

vzdálenost mezi dvěma body. Tato vzdálenost je porovnávána proměnnou „vzdal“, která má prvotní hodnotu záměrně co největší.

Po nalezení nejbližšího bodu k aktuální pozici se tento bod ukrytý v pomocné instanci třídy „pom“ uloží do instance třídy „prvnibod“ a dále se přidá do listu „aktualnibody“.

Tento proces se pomocí cyklu opakuje tolikrát, kolik uživatel zadal míst, které chce navštívit. V cyklu je ještě obsažena kontrola, zda bod, ke kterému chceme počítat vzdálenost, již není obsažen v listu „aktualnibody“. Toho je dosaženo pomocí foreach, který prochází list „aktualnibody“ a pokud nalezne shodu, do proměnné shoda se uloží hodnota false.

6. 2. 3 Metoda nejbližšího souseda s pomocí OSMSharp

Tato metoda vychází z předchozí metody popsané v kapitole 6. 2. 2. Základ je naprosto totožný, pouze se liší výpočet vzdálenosti mezi dvěma body. V předchozím případě se počítá vzdálenost pomocí Haversine vzorce. V této metodě je využita metoda knihoven OSMSharp:

Calculate(OsmSharp.Routing.VehicleEnum vehicle, TResolvedType source, TResolvedType target)

Tato metoda obsahuje tři parametry. Parametr vehicle udává typ dopravního prostředku nebo chůze, parametr source udává počáteční bod a parametr target udává cílový bod.

6. 2. 4 Metoda nejbližšího souseda s pomocí OSMSharp pro zadaný počet bodů

Tato metoda pracuje obdobně jako předchozí metoda „Metoda nejbližšího souseda s pomocí OSMSharp“. Jediný rozdíl je že předchozí metoda prochází body ze souboru, který byl vybrán při startu aplikace, tedy všechny možné body. Zatímco na začátku této metody se použije „Metoda nejbližšího souseda“ popsaná v kapitole 6. 2. 2, která zredukuje všechny body na počet, který uživatel zadal a následně je na tyto body použita tato metoda. Tím je zajištěna větší rychlost, než u metody nejbližšího souseda s pomocí OSMSharp a zároveň větší přesnost, než u metody nejbližšího souseda.

6. 2. 5 Clark-Wrightova metoda

Tak jako u předchozí metody byla i zde rovněž nejdříve použita metoda nejbližšího souseda. Tím byl získán zadaný počet bodu a následně na tyto body byla použita Clark-Wrightova metoda, jejíž princip je popsán v kapitole 3.3.3.

```
int i = 0;
int j = 0;
double vyhodnost = double.MinValue;
for (int x = 1; x < pocet; x++)
{
  for (int y = 1; y < x; y++)
  {
    if (matice[x, y] != null)
      if ((matice[x, 0] + matice[0, y] - matice[x, y]) > vyhodnost)
      {
        vyhodnost = (double)(matice[x, 0] + matice[0, y] - matice[x, y]);
        i = x;
        j = y;
      }
  }
}
```

Nejprve bylo zapotřebí vypočítat matici vzdálenosti mezi jednotlivými body. Vzdálenosti mezi dvojicemi bodů byla počítána metodou „Calculate“, která je obsažena v projektu OSMSharp. Následně se tato matice prochází a hledá se největší výhodnost, která se zapisuje do proměnné „vyhodnost“. Při každém nálezu se rovněž se zápisem výhodnosti provede také zápis polohy bodu v matici. Po projití všech vzdáleností se zkontroluje, zda lze body připojit do okruhu a zda neuzavírají okruh. Tento cyklus se opakuje až do vybrání všech bodu, které nalezla metoda nejbližšího souseda. Na konci je pouze zajištěno vykreslení trasy mezi seřazenými body.

6. 2. 6 Metoda CalculateTSP

K výpočtu trasy byly využity metody knihoven open-source projektu OsmSharp, které výslednou trasu mohou uložit v podobě souboru .gpx, což je xml soubor obsahující pouze GPS souřadnice. Nebo lze vygenerovat souřadnice do Listu.

```
OsmSharpRoute route = tsp_router.CalculateTSP(vehicle, points, 0, true);
m_Trasa = route.GetPoints();
m_Vzdalenost += route.TotalDistance;
```

Metoda CalculateTSP obsahuje čtyři parametry. První parametr udává typ dopravního prostředku, druhý parametr obsahuje pole míst, které chce pracovník navštívit. Třetí parametr říká, které místo v poli je počáteční a poslední parametr je nastavován, aby se zjistilo, zda je první bod zároveň i poslední, tedy zda trasa tvoří kolo.

Nejprve se vybere soubor „map3.osm“, což je soubor ve formátu xml, který popisuje mapu cest v dané lokalitě. Dále se využívají metody knihoven OsmSharp pro předzpracování dat a uložení do paměti.

Následně je vytvořen list s datovým typem GeoCoordinate, kam se budou ukládat souřadnice jednotlivých bodů, které chce uživatel navštívit. Dále je cyklem procházen list aktuálních bodů, které zpracovala metoda nejbližšího souseda. Tento list je procházen po jednom záznamu a každý bod je ukládán do listu GeoCoordinate.

Poté jsou znovu použity metody knihoven OsmSharp k uložení listu souřadnic do pole „points“ a vypočítání optimalizované trasy mezi těmito body.

Nakonec je vygenerován List<GeoCoordinate>, který obsahuje body potřebné pro identifikaci trasy na mapě.

6. 2. 7 Vykreslení trasy

Pokud je zapotřebí vykreslit trasu po výpočtu některou z výše popsaných metod, je nutné použít List<GeoCoordinate>, který jednotlivými souřadnicemi udává trasu na mapě a List<Bod>. Ten obsahuje místa, které chce pracovník navštívit.

Jediná metoda, která zajistí oba potřebné Listy je metoda „CalculateTSP“ z projektu OSMSharp. V ostatních případech je ještě potřeba dopočítat trasu, kterou udává List<GeoCoordinate>. Trasu lze dopočítat snadno za použití metody „Calculate“ z knihoven OSMSharp, která vypočítá zároveň dálku mezi dvěma body a zároveň vrací List<GeoCoordinate>, který tvoří trasu mezi těmito body. Stačí tedy cyklem procházet pole seřazených bodů a přidávat vygenerovanou trasu do celkového List<GeoCoordinate> „locat“, který tvoří trasu mezi všemi body.

Nejdříve ještě před samotným vykreslením se všechny prvky, které mohou být zobrazeny na mapě, vymažou, to proto, aby se zobrazovala stále aktuální data.

Dále se nadefinuje mnohoúhelník, který bude tvořit danou trasu. Byla mu nastavena modrá barva, šířka 5 a průhlednost 0, 7.

Dále je vytvořen Pushpin pro aktuální polohu a zobrazen na mapě. Následuje přiřazení kolekce lokací „locat“ jako zdroj pro vytvoření mnohoúhelníku, který se zobrazí na mapě. Dále je střed mapy nastaven na poslední uložený bod do kolekce lokací a mapa přiblížena na úroveň 10.

Nakonec se pomocí foreach procházejí aktuální body, mezi kterými se vypočítávala optimalizovaná trasa a těmto bodům je přiřazen Pushpin a vytvořen Event pro kliknutí na konkrétní Pushpin.

6. 2. 8 Uložení trasy

Aplikace umožňuje také uložení vykreslené trasy do xml souborů. K definování trasy je zapotřebí uložit List<GeoCoordinate> „locat“ a List<Bod> „aktualnibody“, který obsahuje místa, jenž chce pracovník navštívit. K uložení byl použit XmlSerializer, který serializuje List<GeoCoordinate> „locat“ a List<Bod> „aktualnibody“ do textu a ten je uložen do souboru s příponou .gpx nebo .bod

6. 2. 9 Načtení trasy

K načtení trasy byl taktéž použit XmlSerializer, kterým byl deserializován text z načteného souboru a vytvořen List<GeoCoordinate> „locat“ a List<Bod> „aktualnibody“. Následně stačí vykreslit trasu na mapě postupem popsaným v kapitole 6. 2. 4.

6. 2. 10 Uložení nenavštívených bodů

Pokud uživatel obešel jednu nebo více tras, má možnost si aktuální stav uložit pomocí tlačítka „Uložit“ a pro příště už hledat trasu jen mezi body, které ještě nenavštívil.

Nejprve se otevře nebo vytvoří soubor pro zápis „rozpracováno.txt“. Následně se pomocí foreach procházejí všechny body uloženy v listu „body “ a pomocí druhého vnořeného foreach je kontrolována shoda v listu „aktualnibody“. Pokud je nalezena shoda, bod se neuloží a začne kontrola u dalšího bodu z listu „body“.

Pokud však nebyla nalezena shoda, bod se zapíše do souboru na ta samá místa jako v souboru .pre. Pomocí cyklu while se zapisují jednotlivé znaky, jestliže nenastane žádná z podmínek, tedy pokud se poloha pro zápis znaku nenachází na místě 279, kde začíná jméno nebo na místě 455, kde začíná zeměpisná šířka nebo na místě 465, kde začíná zeměpisná délka, zapisuje se vždy mezera. Pokud některá z podmínek nastane, zapisuje se pomocí cyklu for daná hodnota po znacích a pokud je počet znaků této hodnoty menší než počet míst pro něj vymezených, je zbytek těchto míst doplněn mezerami.

7 Testování aplikace

Byla testována rychlost běhu různých částí aplikace pomocí knihovny tříd Systém.Diagnostic. Využitím třídy Stopwatch byl měřen uplynulý čas při běhu některých metod v aplikaci. Toto měření proběhlo na zařízení s čtyř jádrovým procesorem Intel Core i7-3630QM 2.40GHz a operační pamětí 8 GB paměti DDR3 1 600 MHz.

Dále byla testována vzdálenost trasy po výpočtu použitých metod ve vyvíjené aplikaci. Čas potřebný pro obejití celé trasy byl také zaznamenáván. Toto testování probíhalo pro všechny použité metody shodně. Počáteční bod byl pro testování metod pokaždé stejný a pro měření byla použita shodná data se zadanými body. Měření proběhlo pro deset různých hodnot v rozsahu od 5 do 50 zadaných bodů a pro každou zadanou hodnotu byla zapisována délka a čas pro obejití trasy.

7. 1 Metoda nejbližšího souseda

Rychlost metody byla měřena pro rozsah pozic od 10 do 80 bodů, ze kterých se hledal uživatelem zadaných počet pozic, které chce navštívit. Bylo zadáváno pět různých hodnot v rozpětí od 5 do 25 a pro každou hodnotu byl měřen čas v milisekundách. Výsledky měření jsou znázorněny v Tabulce číslo 4 a na Obrázku číslo 15 (viz Příloha 1).

Dále byla měřena délka tras po výpočtu touto metodou a čas potřebný pro obejití trasy. Výsledky měření jsou znázorněny v Tabulce číslo 9 (viz Příloha 1). Délky tras jsou znázorněny na Obrázku číslo 21 a časy pro obejití tras jsou znázorněny na Obrázku číslo 20 (viz Příloha 1).

7. 2 Metoda nejbližšího souseda s pomocí OSMSharp

Tato metoda byla měřena stejným postupem, jako metoda nejbližšího souseda. Výsledky měření rychlosti metody jsou znázorněny v Tabulce číslo 5 a na Obrázku číslo 16 (viz Příloha 1).

Výsledky měření délky trasy a času pro obejití trasy jsou znázorněny v Tabulce číslo 10 (viz Příloha 1). Délky tras jsou znázorněny v Obrázku číslo 23 a časy pro obejití tras jsou znázorněny v Obrázku číslo 22 (viz Příloha 1).

7. 3 Metoda nejbližšího souseda s pomocí OSMSharp pro zadaný počet bodů

U této metody počet bodů, který je uveden v načítaných datech nemá vliv na rychlost běhu metody. Toto je eliminováno použitím metody nejbližšího souseda, před použitím této metody, čímž dostaneme pouze počet bodů, které chce pracovník obejít. Proto se zde zkoumá vliv pouze zadaného počtu bodů uživatelem na rychlost metody. Pro celkový čas výpočtu tras je nutno započítat ještě čas pro výpočet metodou nejbližšího souseda, který není v tomto měření zahrnut. Výsledky tohoto měření jsou znázorněny v Tabulce číslo 6 a Obrázku číslo 17 (viz Příloha 1).

Výsledky měření délky trasy a času pro obejití trasy jsou znázorněny v Tabulce číslo 11 (viz Příloha 1). Délky tras jsou znázorněny v Obrázku číslo 25 a časy pro obejití tras jsou znázorněny v Obrázku číslo 24 (viz Příloha 1).

7. 4 Clark-Wrightova metoda

Testování rychlosti této metody probíhalo shodně jako u metody nejbližšího souseda s pomocí OSMSharp pro zadaný počet bodů. Výsledky tohoto měření jsou znázorněny v Tabulce číslo 7 a na Obrázku číslo 18 (viz Příloha1).

Výsledky měření délky trasy a času pro obejití trasy jsou znázorněny v Tabulce číslo 12 (viz Příloha 1). Délky tras jsou znázorněny na Obrázku číslo 27 a časy pro obejití tras jsou znázorněny na Obrázku číslo 26 (viz Příloha 1).

7. 5 Metoda CalculateTSP

Testování rychlosti této metody probíhalo shodně jako u metody nejbližšího souseda s pomocí OSMSharp pro zadaný počet bodů. Výsledky tohoto měření jsou znázorněny v Tabulce číslo 8 a na Obrázku číslo 19 (viz Příloha1).

Výsledky měření délky trasy a času pro obejití trasy jsou znázorněny v Tabulce číslo 13 (viz Příloha 1). Délky tras jsou znázorněny na Obrázku číslo 29 a časy pro obejití tras jsou znázorněny na Obrázku číslo 28 (viz Příloha 1).

8 Závěr

Na začátku této práce jsem si určil za cíl vytvořit testovací aplikaci, která bude vypočítávat optimalizované trasy. Tato aplikace by měla hlavně pomoci terénním pracovníkům firmy E.ON. K dosažení tohoto cíle jsem si zvolil vyvíjet samotnou aplikaci v prostředí .NET Framework a jako podklady pro výpočet jsem využil open-source mapy projektu OpenStreetMap. Samotný výpočet byl proveden pomocí open-source knihoven OsmSharp.

Hlavním přínosem této práce pro praxi je zejména úspora času. Pracovníci šetří čas, již při samotném plánování tras, které za ně provede aplikace. Nemusí se tedy zabývat přemýšlením nad tím, jaké místo navštívit jako první, a které z následujících míst je tomuto místu nejblíže. Dochází tedy k efektivnímu plánování práce. Vzhledem k nalezení optimalizované trasy má pracovník jistotu, že tuto trasu zvládne obejít v nejkratším možném čase.

Dalším neméně důležitým přínosem je úspora nákladů na pohonné hmoty. Vzhledem k tomu, že pracovníci používají k navštívení míst na trase automobil, dochází ke spotřebovávání pohonných hmot. Optimalizovaná, tedy nejkratší trasa, šetří zaměstnancům ujeté kilometry, čímž samozřejmě dochází k úspoře nákladů na provoz vozidla.

Jako další přínos lze uvést efektivní vytížení pracovníků. Zaměstnanec zvládne díky optimalizované trase navštívit více míst za stejný časový úsek nebo se může díky ušetřenému času věnovat jiné práci. Tento přínos může vést k úspoře mzdových nákladů firmy.

Používání aplikace není výhodou jen pro firmu, ale i pro samotné pracovníky, kteří se díky ní mohou snáze orientovat v méně známých oblastech. Při navštívení neznámých míst je pro pracovníka velmi obtížné plánovat zde optimální trasu ručně, proto je pro něj používání této aplikace značnou výhodou.

Do budoucna by bylo nejvýhodnější vyvinout tuto aplikaci pro mobilní zařízení. Během vypracovávání této práce bylo zjištěno, že offline výpočet optimální trasy pomocí použitých technologií je momentálně vhodný pouze pro malé oblasti jako jsou města. Ve velkých oblastech nastává enormní nárok na operační paměť zařízení. Proto je v současné době možné pouze řešení výpočtu optimalizované trasy na velkých oblastech pouze na externích

výpočetních serverech. Do mobilního zařízení by se tedy posílala pouze data, která by charakterizovala trasu, a uživatel by si pomocí svého mobilního zařízení tuto trasu vykreslil.

Jako nejlépe použitelná metody výpočtu trasy se jeví metoda CalculateTSP použita z projektu OSMSharp. Výsledky měření metod ukazují, že tato metoda je nejpřesnější a zároveň rychlost výpočtu trasy je dostatečně nízká, aby uživatel nečekal na výpočet příliš dlouho.

9 Seznam Obrázků

10 Seznam tabulek

11 Seznam literatury

[1] HYNEK, Josef. *Genetické algoritmy a genetické programování*. 1. vyd. Praha: Grada, 2008, 182 s. ISBN 978-80-247-2695-3.

[2] KUČERA, Luděk. *Kombinatorické algoritmy*. 1. vyd. Praha: SNTL, 1983, 283 s.

[3] KORTE, B a Jens VYGEN. *Combinatorial optimization: theory and algorithms*. 5th ed. New York: Springer, c2012, xix, 659 p. ISBN 9783642244889-.

[4] *OpenStreetMap Wiki* [online]. 2013 [cit. 2013-04-14]. Dostupné z: http://wiki.openstreetmap.org/wiki/Main_Page

[5] SHARP, John. *Microsoft Visual C# 2010 step by step*. Redmond, Wash.: Microsoft Press, c2010, xxx, 748 s. ISBN 978-0-7356-2670-6.

[6] *OsmSharp | OpenStreetMap (OSM) Library and Tools!* [online]. 2013 [cit. 2013-04-15]. Dostupné z: http://www.osmsharp.com/

[7] *Bing* [online]. 2013 [cit. 2013-04-15]. Dostupné z: http://www.bing.com/

[8] PELLAND, Patrice, Pascal PARE a Ken HAINES. *Moving to Microsoft Visual Studio 2010* [online]. 2012 [cit. 2013-04-15]. ISBN 978-0735647886. Dostupné z: http://blogs.msdn.com/b/microsoft_press/archive/2010/09/13/free-ebook-moving-to-microsoft-visual-studio-2010.aspx

[9] KUČERA, Petr. *Metodologie řešení okružního dopravního problému*. Praha, 2009. Disertační práce. Česká zemědělská univerzita v Praze, Provozně ekonomická fakulta, Katedra systémového inženýrství.

[10] *E.ON SE - Energy Provider, Renewables, Power, Gas* [online]. 2013 [cit. 2013-04-16]. Dostupné z: http://www.eon.com/en.html

[11] *E.ON - Úvodní stránka* [online]. 2013 [cit. 2013-04-16]. Dostupné z: http://www.eon.cz/

[12] *MSDN – the Microsoft Developer Network* [online]. 2013 [cit. 2013-04-16]. Dostupné z: http://msdn.microsoft.com/en-us/

[13] *NuGet Gallery | Home* [online]. 2013 [cit. 2013-04-17]. Dostupné z: http://www.nuget.org/

[14] G. Clarke and J. Wright “Scheduling of vehicles from a central depot to a number of delivery points”, Operations Research, 12 #4, 568-581, 1964.

[15] Minimální kostra v grafu | Algoritmy. *Minimální kostra v grafu | Algoritmy* [online]. 2013 [cit. 2014-04-03]. Dostupné z: http://algoritmy.eu/zga/minimalni-kostra/

[16] CORMEN, Thomas H. *Introduction to algorithms*. 3rd ed. Cambridge: MIT Press, c2009, xix, 1292 s. ISBN 978-0-262-03384-8.

[17] Genetické algoritmy a jejich aplikace v praxi. *Programujte.com* [online]. 2005 [cit. 2014-04-06]. Dostupné z: http://programujte.com/clanek/2005072601-geneticke-algoritmy-a-jejich-aplikace-v-praxi/

[18] Nearest Neighbour. *Cardiff University* [online]. 2014 [cit. 2014-04-09]. Dostupné z: https://users.cs.cf.ac.uk/C.L.Mumford/howard/NearestNeighbour.html

[19] *The traveling salesman problem: a computational study*. Princeton: Princeton University Press, 2006, ix, 593 s. ISBN 978-0-691-12993-8.

[20] ROSENKRANTZ, Daniel J, S RAVI a Sandeep K SHUKLA. *Fundamental problems in computing: essays in honor of Professor Daniel J. Rosenkrantz*. Dordrecht?: Springer, c2009, xxi, 515 p. ISBN 978-140-2096-877.

12 Seznam příloh

Příloha 1: Doprovodné tabulky a grafy

Příloha 2: Obsah přiloženého CD

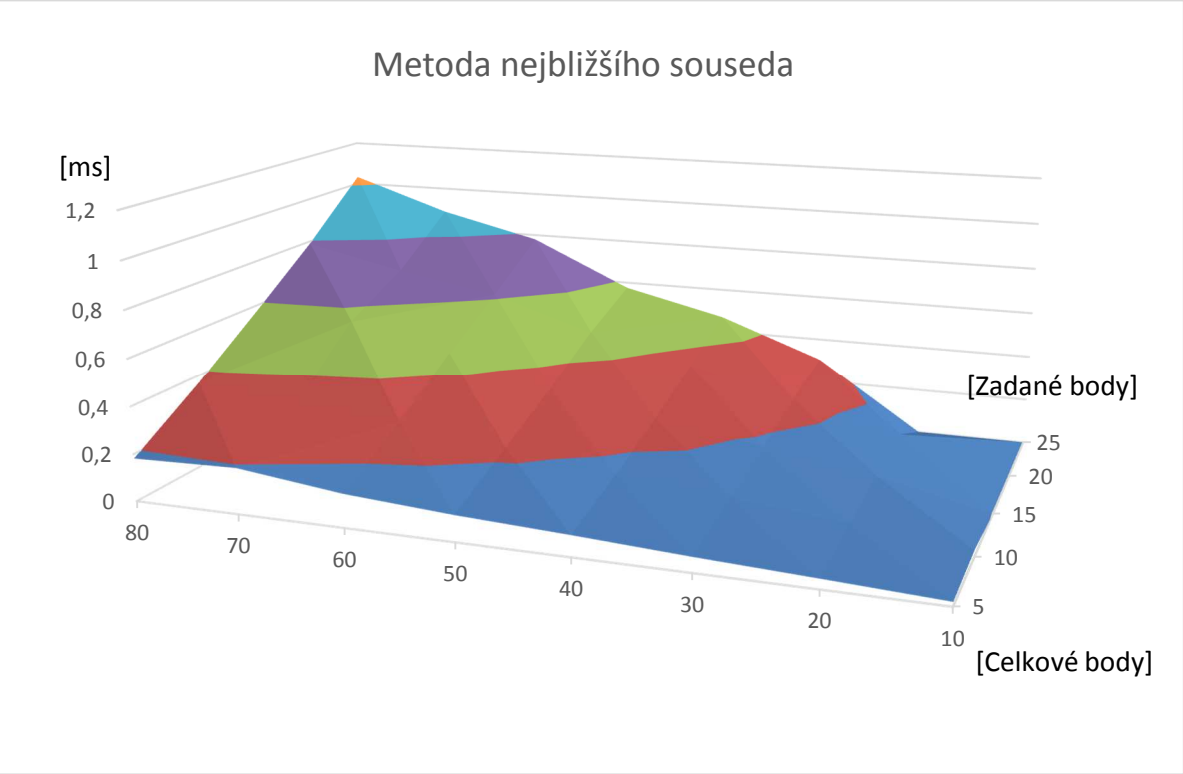
Příloha 1: Doprovodné tabulky a grafy

Tabulka 4: Naměřené hodnoty metody nejbližšího souseda v [ms]

Celkové body						
80	0,1813	0,3694	0,5764	0,7902	1,0323	
70	0,1932	0,3241	0,5622	0,6931	0,886	
60	0,1406	0,2865	0,4567	0,6149	0,7795	
50	0,1099	0,2266	0,3429	0,4738	0,5747	
40	0,088	0,1646	0,2557	0,3557	0,4686	
30	0,0624	0,1257	0,1808	0,2463	0,3023	
20	0,041	0,0778	0,1069	0,1351		
10	0,0179	0,0277				
0	5	10	15	20	25	Zadané body

Zdroj: Vlastní práce

Obrázek 15: Rychlost metody nejbližšího souseda



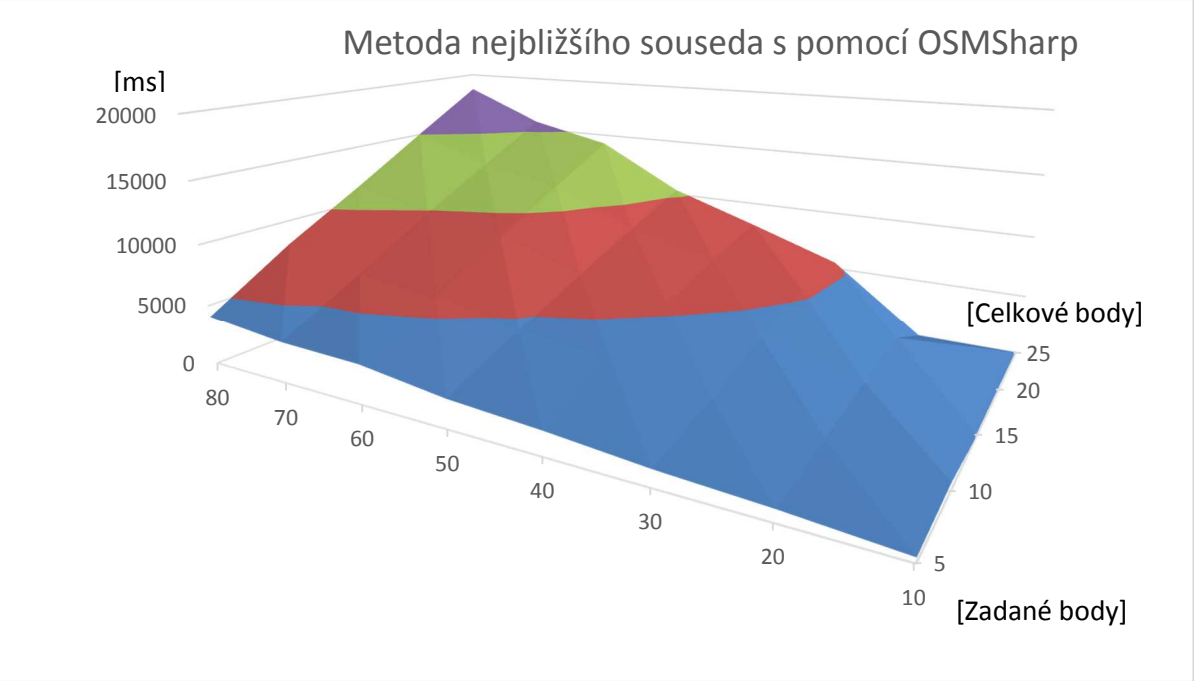
Zdroj: Vlastní práce

Tabulka 5: Naměřené hodnoty metody nejbližšího souseda s pomocí OSMSharp v [ms]

Celkové body						
80	4032	8020	11444	15039	18554	
70	3383	6573	9831	12799	15729	
60	3212	5802	8822	11495	14184	
50	2337	4516	6579	8526	10401	
40	1912	3733	5401	6853	8015	
30	1326	2536	4116	5307	5525	
20	940	1542	2178	2696		
10	378	627				
0	5	10	15	20	25	Zadané body

Zdroj: Vlastní práce

Obrázek 16: Metoda nejbližšího souseda s pomocí OSMSharp



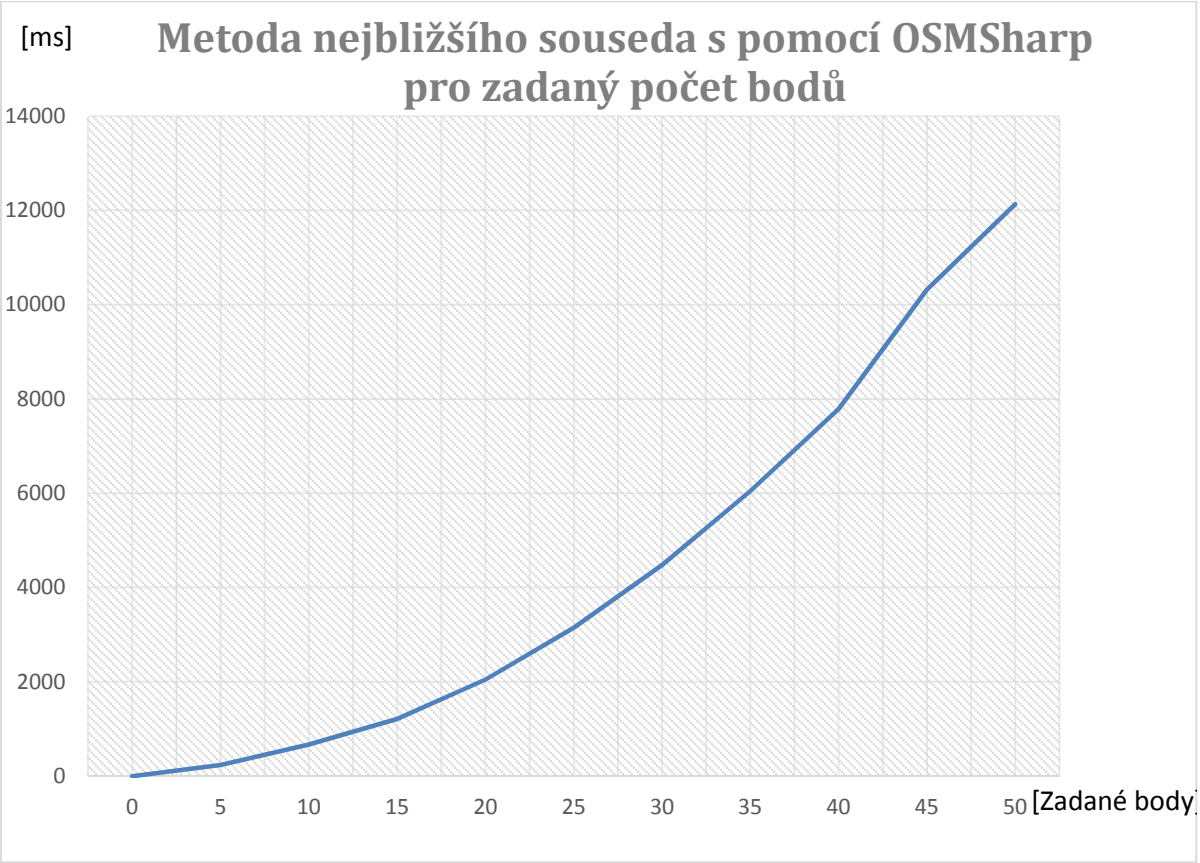
Zdroj: Vlastní práce

Tabulka 6: Naměřené hodnoty metody nejbližšího souseda s pomocí OSMSharp pro zadaný počet bodů v [ms]

Čas	0	231	669	1212	2046	3145	4477	6040	7782	10315	12132
Zadané body	0	5	10	15	20	25	30	35	40	45	50

Zdroj: Vlastní práce

Obrázek 17: Metoda nejbližšího souseda s pomocí OSMSharp pro zadaný počet bodů



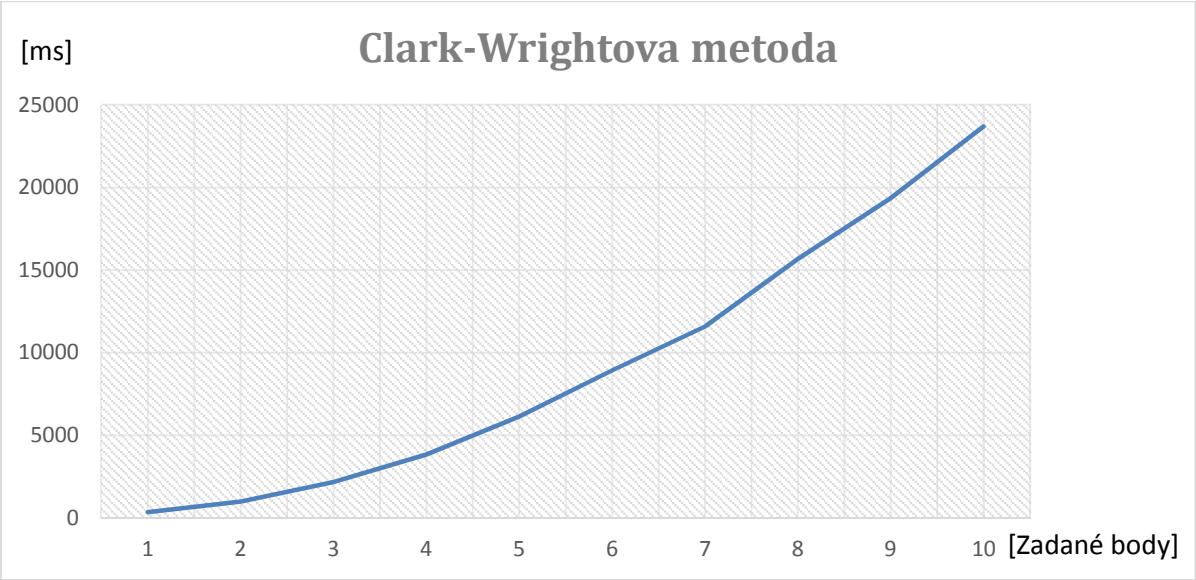
Zdroj: Vlastní práce

Tabulka 7: Naměřené hodnoty Clark-Wrightovou metodou pro zadaný počet bodů v [ms]

Čas	0	251	800	1868	3433	5463	8093	11257	14751	18898	23597
Zadané body	0	5	10	15	20	25	30	35	40	45	50

Zdroj: Vlastní práce

Obrázek 18: Clark-Wrightova metoda



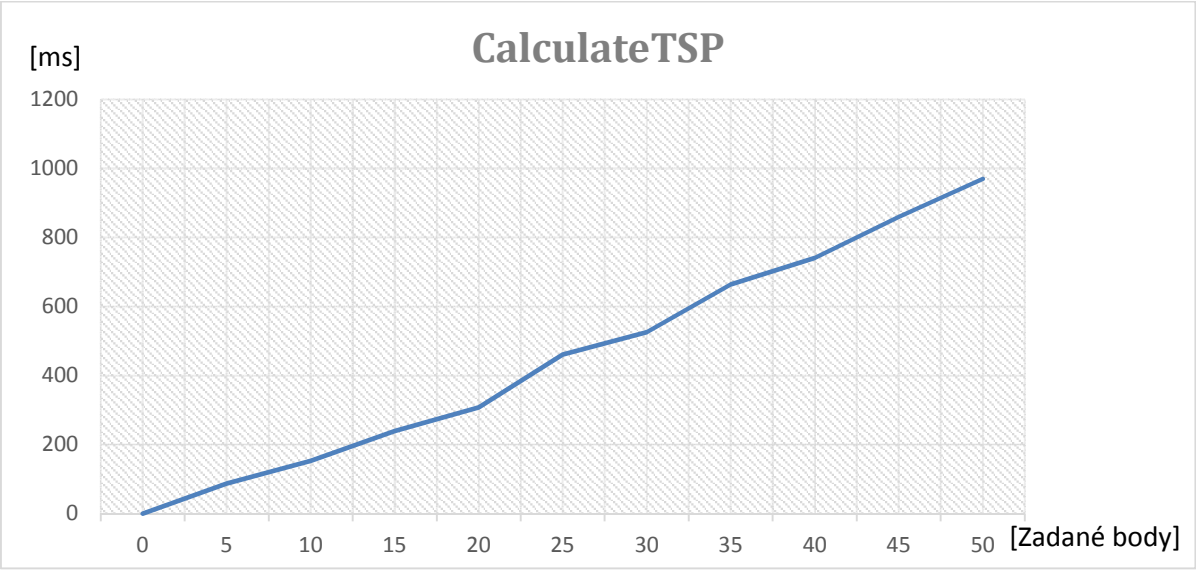
Zdroj: Vlastní práce

Tabulka 8: Naměřené hodnoty metodou CalculateTSP pro zadaný počet bodů v [ms]

Čas	0	87	153	239	308	461	526	664	741	860	970
Zadané body	0	5	10	15	20	25	30	35	40	45	50

Zdroj: Vlastní práce

Obrázek 19: Metoda CalculateTSP



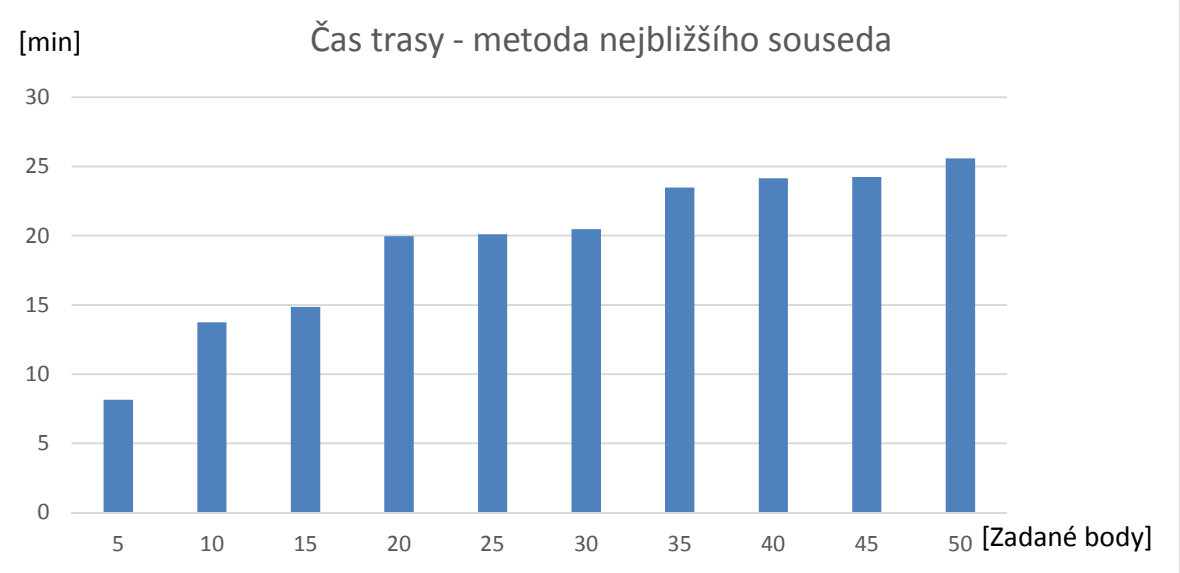
Zdroj: Vlastní práce

Tabulka 9: Vzdálenost a čas metodou nejbližšího souseda

Zadané body	Vzdálenost [m]	Rychlost [min]
5	2343	8,15
10	3100	13,74
15	3702	14,85
20	5023	19,96
25	5144	20,1
30	5142	20,47
35	7099	23,48
40	7645	24,14
45	7741	24,25
50	8465	25,58

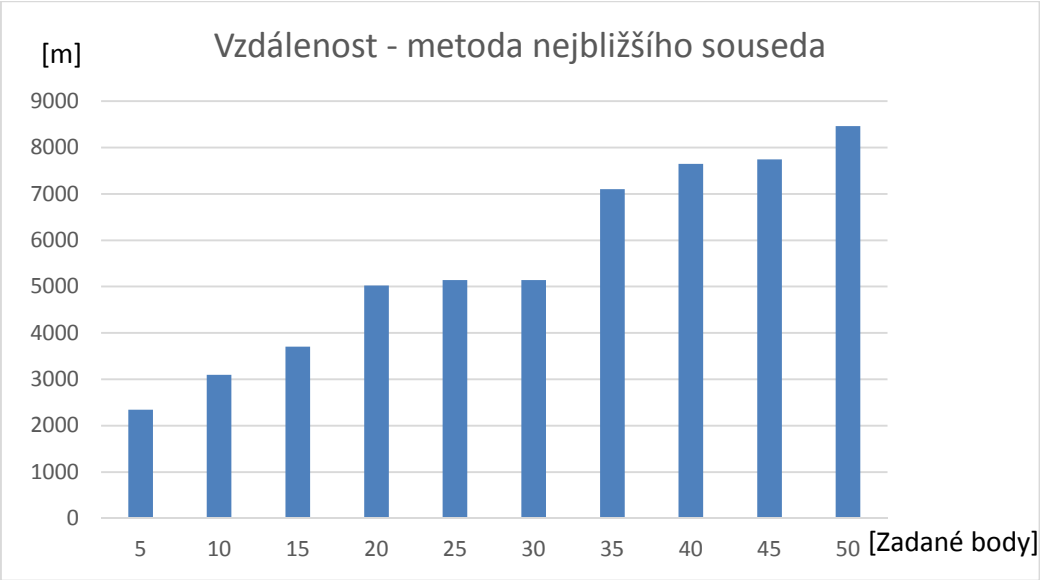
Zdroj: Vlastní práce

Obrázek 20: Čas trasy - metoda nejbližšího souseda



Zdroj: Vlastní práce

Obrázek 21: Vzdálenost - metoda nejbližšího souseda



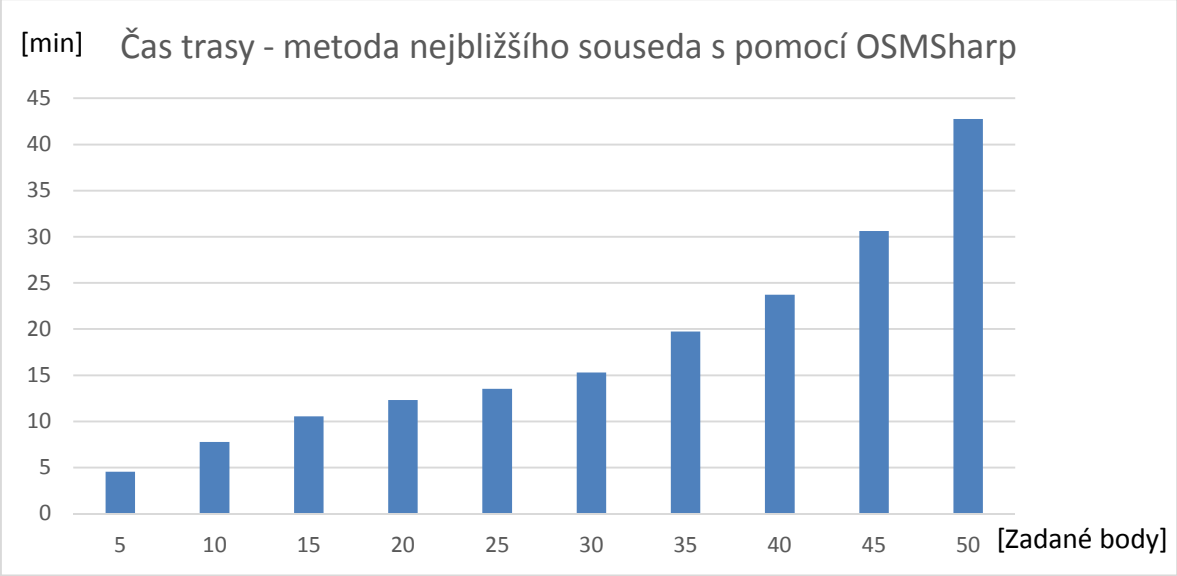
Zdroj: Vlastní práce

Tabulka 10: Vzdálenost a čas metodou nejbližšího souseda s pomocí OSMSharp

Zadané body	Vzdálenost [m]	Rychlost [min]
5	2268	4,54
10	3534	7,77
15	5440	10,56
20	5970	12,31
25	6943	13,54
30	8410	15,31
35	10314	19,74
40	11022	23,71
45	14333	30,61
50	18861	42,75

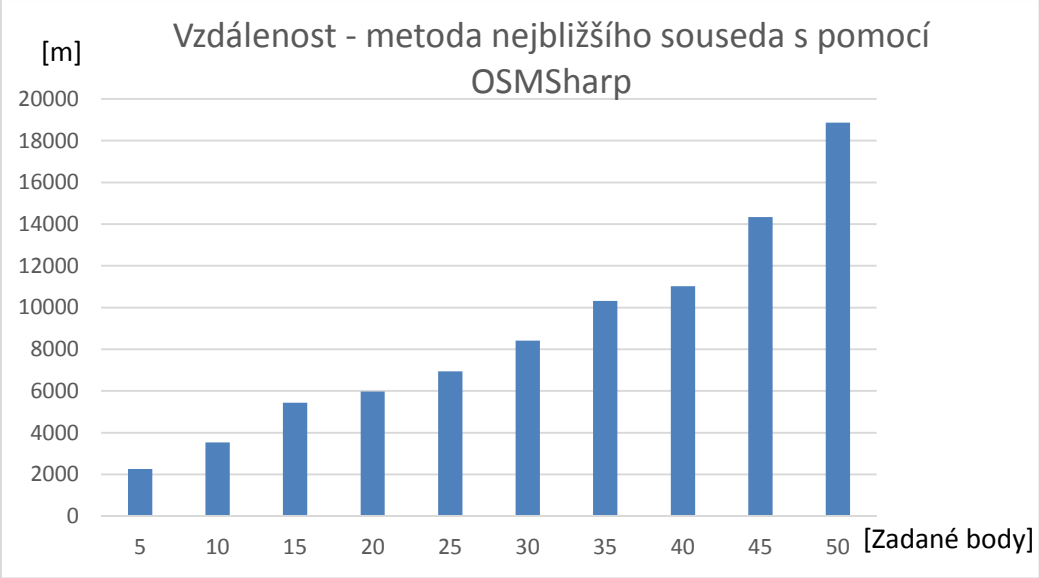
Zdroj: Vlastní práce

Obrázek 22: Čas trasy - metoda nejbližšího souseda s pomocí OSMSharp



Zdroj: Vlastní práce

Obrázek 23: Vzdálenost - metoda nejbližšího souseda s pomocí OSMSharp



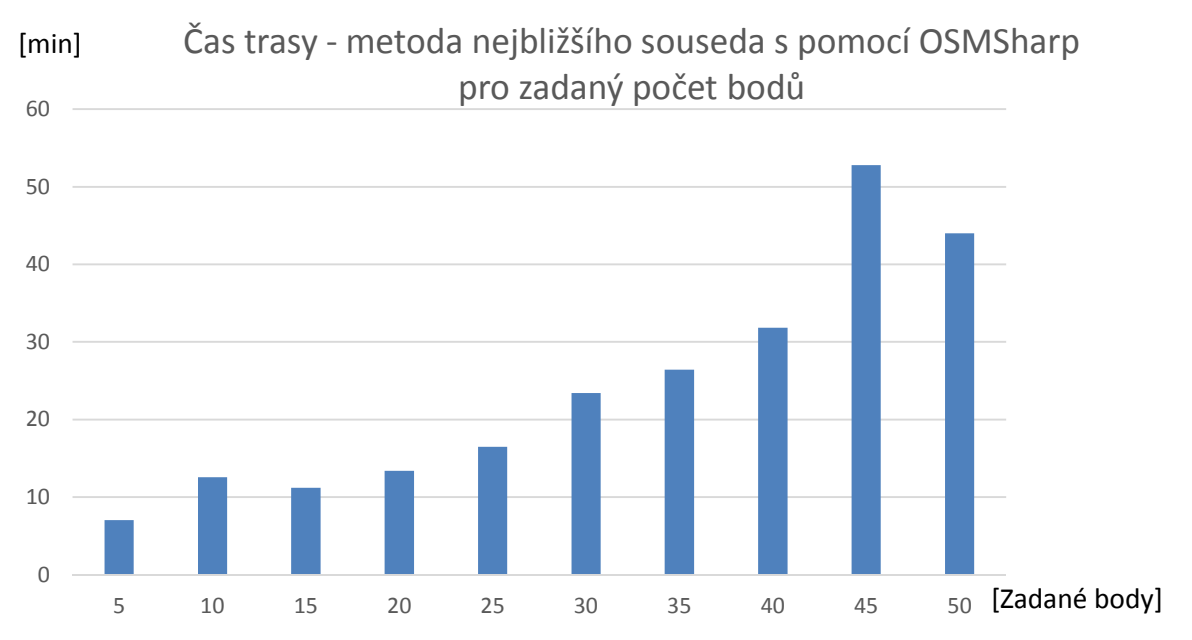
Zdroj: Vlastní práce

Tabulka 11: Vzdálenost a čas metodou nejbližšího souseda s pomocí OSMSharp pro zadaný počet bodů

Zadané body	Vzdálenost [m]	Rychlost [min]
5	1257	7,06
10	3788	12,57
15	5727	11,19
20	6522	13,39
25	7598	16,49
30	10139	23,4
35	11740	26,43
40	14935	31,85
45	16707	52,81
50	17745	44

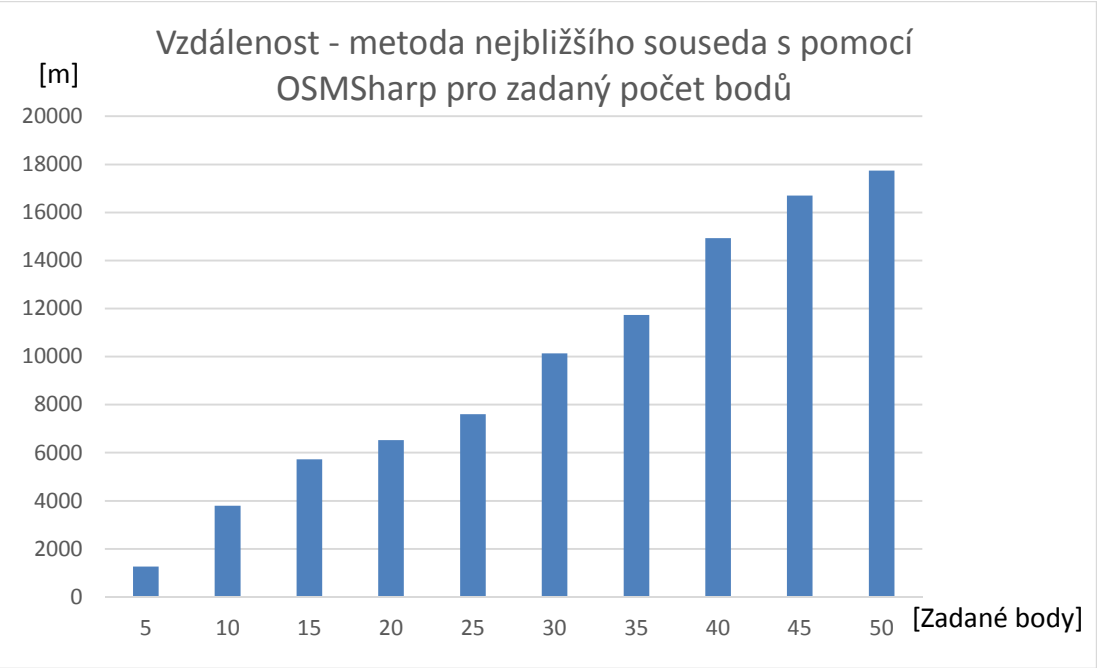
Zdroj: Vlastní práce

Obrázek 24: Čas trasy - metoda nejbližšího souseda s pomocí OSMSharp pro zadaný počet bodů



Zdroj: Vlastní práce

Obrázek 25: Vzdálenost - metoda nejbližšího souseda s pomocí OSMSharp pro zadaný počet bodů



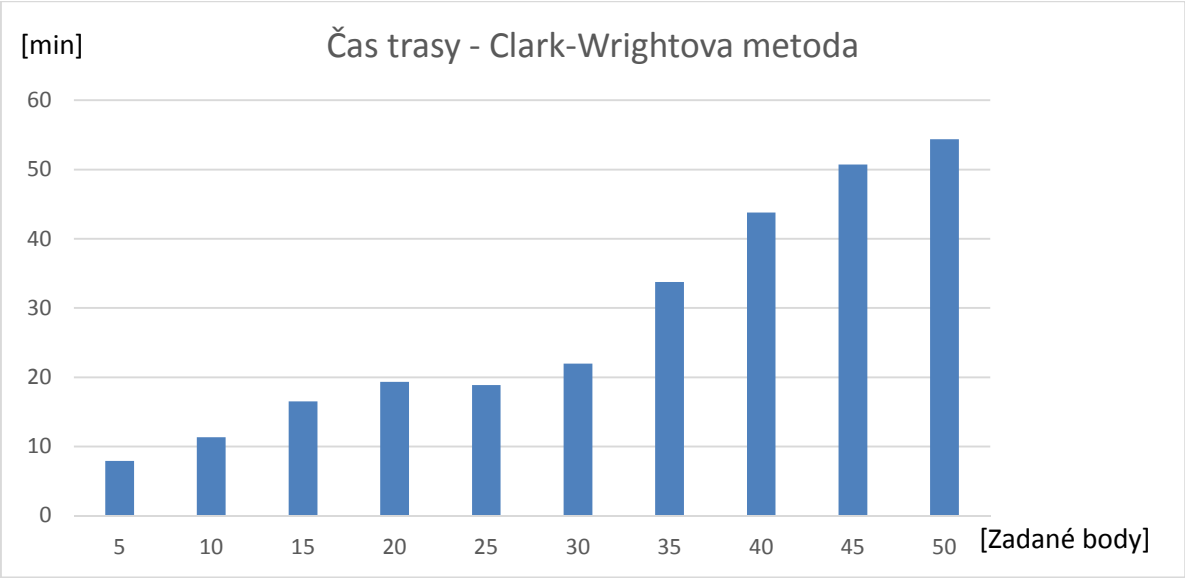
Zdroj: Vlastní práce

Tabulka 12: Vzdálenost a čas Clark-Wrightovou metodou

Zadané Body	Vzdálenost [m]	Rychlost [min]
5	3637	7,9
10	4427	11,35
15	5330	16,5
20	6759	19,35
25	7674	18,87
30	9074	21,96
35	12315	33,77
40	16448	43,79
45	19394	50,75
50	21063	54,4

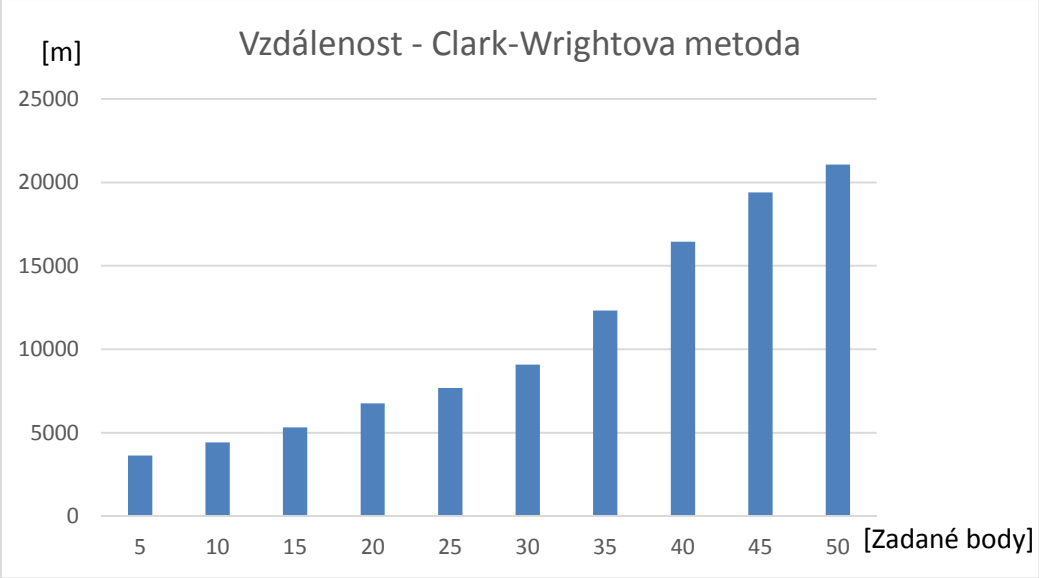
Zdroj: Vlastní práce

Obrázek 26: Čas trasy - Clark-Wrightova metoda



Zdroj: Vlastní práce

Obrázek 27: Vzdálenost - Clark-Wrightova metoda



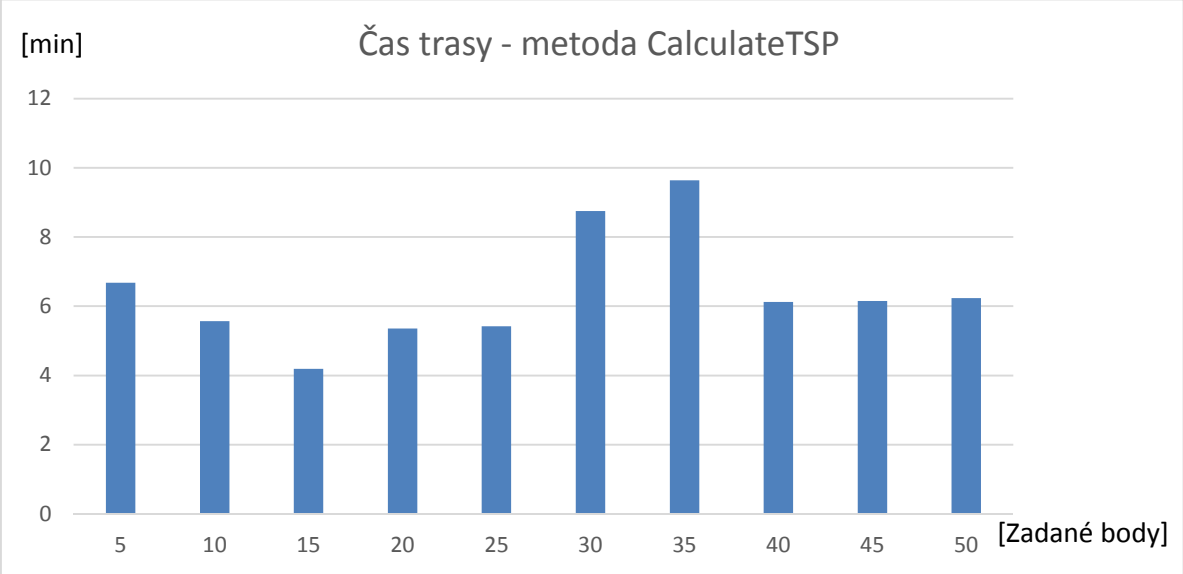
Zdroj: Vlastní práce

Tabulka 13: Vzdálenost a čas metodou CalculateTSP

Zadané body	Vzdálenost [m]	Rychlost [min]
5	2300	6,68
10	1904	5,57
15	2254	4,19
20	2867	5,36
25	2913	5,42
30	4021	8,75
35	4800	9,64
40	4658	6,12
45	4665	6,15
50	4737	6,23

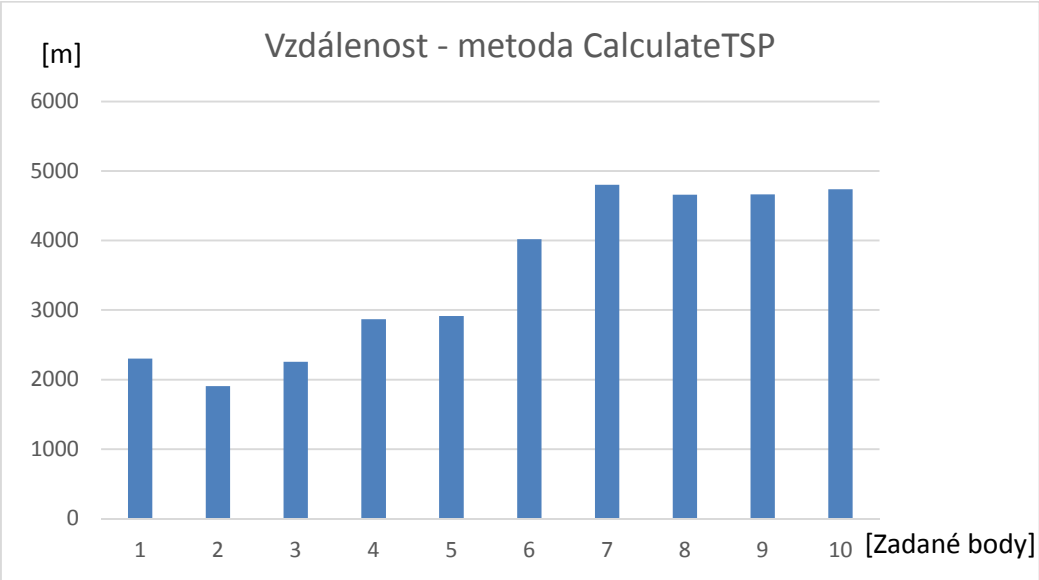
Zdroj: Vlastní práce

Obrázek 28: Čas trasy - metoda CalculateTSP



Zdroj: Vlastní práce

Obrázek 29: Vzdálenost - metoda CalculateTSP



Zdroj: Vlastní práce

Příloha 2: Obsah přiloženého CD

- Adresář Optimalizace_tras obsahující aplikaci a dokumentace.
- Adresář Optimalizace obsahující zdrojové kódy a použité knihovny.
- Soubor Bakalářská práce - Svačina (2013).pdf obsahující text bakalářské práce.
- README.txt