ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Problém cestujícího lovce hry Geocaching I
Student:	Klára Schovánková
Vedoucí:	Ing. Jan Baier
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

Navrhněte a implementujte prototyp modulu do aplikace GeoGet [2], která podporuje tvorbu výletů spojených se hrou Geocaching [3].

Klíčové funkcionality, které musí modul obsahovat:
1) import/export strojově zpracovatelných dat (souřadnice schránek, atributy, ...),
2) nalezení nejvhodnější trasy výletu dle zvolených kritérií (např. délka trasy, typ, oblíbenost a časová náročnost schránky a další),
3) možnost výběru alternativních tras,
4) plná funkčnost v offline režimu,
5) zobrazení trasy na mapě včetně odhadu celkové časové náročnosti.

Stěžejní je návrh a implementace algoritmů pro nalezení nejvhodnější trasy. Zaměřte se na 2-aproximační a hladové algoritmy. Stanovte metriky pro určení kvality nalezené trasy a ohodnoťte použité algoritmy. Svoje metriky použijte pro ohodnocení tras v konkurenční implementaci modulu od Jana Staňka z roku 2016.

Seznam odborné literatury

[1] Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research, 59(2), 231-247.
[2] http://geoget.ararat.cz
[3] http://geocaching.com

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 24. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Bakalářská práce

Problém cestujícího lovce hry Geocaching I

Klára Schovánková

Vedoucí práce: Ing. Jan Baier

15. května 2016

Poděkování

Děkuji Ing. Janu Baierovi za cenné rady a trpělivost při vedení této práce. Dále bych ráda poděkovala svým rodičům za podporu během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2016 .....

Odkaz na tuto práci

Schovánková, Klára. *Problém cestujícího lovce hry Geocaching I.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce se zabývá hledáním tras pro hru Geocaching. Její součástí je porovnání různých algoritmů pro hledání vhodných tras se zaměřením na hladové algoritmy a 2-aproximační algoritmus. Těchto algoritmů je následně využito při návrhu a implementaci prototypu modulu do aplikace GeoGet, který navrhuje vhodné trasy. Cílem této práce je zefektivnění plánování výletů spojených se hrou Geocaching.

Klíčová slova Geocaching, GeoGet, hladové algoritmy, problém obchodního cestujícího, plánování výletů

Abstract

This thesis deals with finding routes for the Geocaching game. It includes a comparison of different algorithms for finding suitable routes, focusing on greedy algorithms and 2-approximation algorithm. These algorithms are subsequently used in design and implementation of a prototype module for the GeoGet application which proposes suitable routes. The goal of this thesis is to make planning of Geocaching trips more efficient.

Keywords Geocaching, GeoGet, greedy algorithms, traveling salesman problem, planning trips

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Mnoho lidí na celém světě si oblíbilo Geocaching, hru, která kombinuje turistiku s dobrodružným hledáním schránek (keší). Tyto schránky mohou mít různou podobu a jsou ukryty všude kolem nás. Na celém světě se nachází téměř tři miliony keší, v České republice jich je téměř padesát tisíc. Hráč obvykle hledá keše pomocí zadaných zeměpisných souřadnic.

Motivace

Často si hráč chce udělat výlet spojený s hledáním keší a hledá trasu, po které se vydat. Najít vhodné trasy dle požadovaných kritérií je pro člověka časově náročné a momentálně neexistuje nástroj, který by toto hledání usnadňoval. Právě tyto důvody autorku motivovaly k vypracování bakalářské práce na toto téma a k vytvoření modulu, který pomáhá s tvorbou tras pro hru Geocaching. Autorka se již šest let věnuje Geocachingu a během této doby nasbírala mnoho zkušeností, ze kterých čerpala při tvorbě této práce.

Cíl práce

Cílem praktické části práce je návrh a implementace prototypu modulu do aplikace Geoget, který usnadňuje hráčům hry Geocaching plánování tras. Na základě uživatelem zvolených kritérií (např. časová náročnost trasy, výchozí bod, obtížnost a oblíbenost keší . . .) modul navrhne uživateli několik vhodných tras. Modul musí umožňovat zobrazování tras na mapě, včetně krátkého popisu keší, které se na dané trase nachází. Cílem rešeršní části práce je analýza možností řešení problému obchodního cestujícího (včetně zobecněného problému obchodního cestujícího) a návrh a implementace vybraných algoritmů pro hledání nejkratších tras. Tato práce se zaměřuje na hladové algoritmy, exaktní algoritmy a 2-aproximační algoritmus.

Tématem této práce není implementace všech možných řešení problému obchodního cestujícího. Tato práce se nezabývá genetickými algoritmy a neuronovými sítěmi, důraz je kladen spíše na dosažení řešení v uživatelsky přijatelném čase, i za cenu suboptimálního řešení.

Struktura práce

V první kapitole se autorka zabývá popisem problému obchodního cestujícího a definicemi souvisejících pojmů.

Druhá kapitola je věnována vybraným způsobům řešení problému obchodního cestujícího. Jsou zde také rozebrány možnosti řešení zobecněného problému obchodního cestujícího.

Ve třetí kapitole jsou vysvětleny pojmy týkající se satelitní navigace a Geocachingu. Dále je popsána aplikace GeoGet a současné možnosti plánování tras pro Geocaching.

Čtvrtá kapitola popisuje funkcionality a implementaci modulu TripMaker sloužícího k plánování tras.

Pátá kapitola pojednává o metodách měření úspěšnosti hledání tras. Dále obsahuje výsledky experimentálních měření a porovnání modulu s konkurenční implementací modulu od Jana Staňka z roku 2016.

Kapitola 1

Definice a pojmy

Tato kapitola definuje pojmy z teorie grafů, které se budou vyskytovat v této práci. Dále představuje samotný problém obchodního cestujícího, jeho varianty a jeho obtížnost. Formální definice jsou převzaty z [1].

1.1 Graf

Definice 1.1. Nechť H, U jsou libovolné disjunktní mmnožiny a $\varrho : H \mapsto U \times U$ zobrazaní. *Neorientovaným grafem* nazýváme uspořádanou trojici $G = \langle H, U, \varrho \rangle$, prvky množiny H jsou *hranami* grafu G, prvky množiny U *uzly* grafu G a zobrazení ϱ *incidencí* grafu G.

Definice 1.2. Nechť $G = \langle H, U, \varrho \rangle$ je graf, $u \in U$ libovolný uzel a $A \subseteq U$ libovolná podmnožina uzlů. *Množinou sousedů* $\Gamma(u)$ *uzlu* u nazýváme podmnožinu uzlů definovanou vztahem

$$\Gamma(u) = \{v \in U \ : \exists h \in H(\varrho(h) = [u, v])\}.$$

Stupněm $\delta_G(u)$ uzlu u v grafu G nazýváme počet hran s ním incidujících.

Definice 1.3. Graf $G' = \langle H', U', \varrho' \rangle$ nazýváme *podgrafem* grafu $G = \langle H, U, \varrho \rangle$ (zapisujeme $G' \subseteq G$), jestliže platí

$$(H' \subseteq H) \wedge (U' \subseteq U) \wedge \forall h \in H'(\varrho'(h) = \varrho(h)).$$

Podgraf $G' = \langle H', U', \varrho' \rangle$, jehož množina uzlů je shodná s množinou uzlů grafu G, nazýváme *faktorem* grafu G.

Definice 1.4. Nechť pro danou dvojici uzlů u a v grafu $G = \langle H, U, \varrho \rangle$ existuje posloupnost uzlů a hran

$$S \ = \langle u_0, h_1, u_1, h_2, \ldots, u_{n-1}, h_n, u_n \rangle,$$

kde $h_i \in H$, $\varrho(h_i) = [u_{i-1}, u_i]$ pro $i = 1, 2, \ldots, n$, $u_i \in U$ pro $i = 0, 1, \ldots, n$, $u_0 = u$, $u_n = v$. Pak tuto posloupnost nazýváme *sledem* grafu G mezi uzly u a v. Sled s alespoň jednou hranou, v němž jsou uzly u a v shodné, nazýváma *uzavřeným*, ostatní sledy (včetně sledů nulové délky) nazýváme *otevřenými*.

Definice 1.5. *Tahem grafu G* nazýváme takový jeho sled, v němž jsou všechny hrany různé. *Cestou grafu G* nazýváme takový tah, v němž každý uzel inciduje nejvýše se dvěma hranami tohoto tahu. *Kružnicí* nazýváme uzavřenou cestu.

Definice 1.6. *Souvislým grafem* nazýváme takový neorientovaný graf, mezi jehož libovolnými dvěma uzly existuje sled. *Komponentou grafu* nazýváme každý jeho maximální souvislý podgraf.

Definice 1.7. *Stromem* nazýváme souvislý graf neobsahující kružnice.
Kostrou grafu rozumíme takový jeho faktor, který je stromem.

Definice 1.8. Nechť H, U jsou libovolné disjunktní mmnožiny a $\varrho : H \mapsto U \times U$ zobrazaní. *Orientovaným grafem* nazýváme uspořádanou trojici $G = \langle H, U, \varrho \rangle$, prvky množiny H nazýváme *orientovanými hranami* grafu G, prvky množiny U *uzly* grafu G a zobrazení ϱ *incidencí* grafu G.

Definice 1.9. *Hamiltonovskou kružnicí* grafu G nazýváme takovou kružnici, která obsahuje všechnu uzly grafu G.

1.2 Třídy složitosti

Definice 1.10. *Třídou složitosti P* nazýváme množinu všech rozhodovacích úloh řešitelných v polynomiálně omezeném čase.

Definice 1.11. *Třídou složitosti NP* nazýváme množinu všech rozhodovacích úloh, pro které existuje polynomiálně omezený nedeterministický algoritmus jejich řešení.

Definice 1.12. *NP-úplné* problémy jsou natolik obtížné NP problémy, že každá jiná úloha třídy NP je na ně redukovatelná v polynomiálním čase.

Definice 1.13. *NP-těžké* problémy jsou takové, že na ně lze v polynomiálním čase redukovat každý problém L z NP.

1.3 Problém obchodního cestujícího

Název úlohy je odvozen z praktického problému: je dán určitý počet míst, které je třeba navštívit, a cesty o známých délkách, které tato místa spojují. Je třeba nalézt nejkratší možnou trasu, která projde každým místem právě jednou a vrátí se zpět do výchozího místa.

Z pohledu teorie grafů jsou místa reprezentována uzly a cesty jsou reprezentovány hranami. Řešením problému obchodního cestujícího (TSP) je, pro zadaný graf $G = \langle H, U, \varrho \rangle$ s ohodnocením hran $w : H \mapsto \mathbb{R}^+$, nalézt Hamiltonovskou kružnici s minimální délkou [1]. Takto zadaný TSP se také nazývá *optimalizační* problém obchodního cestujícího, protože je hledána minimální možná (optimální) délka kružnice.

Existuje také *rozhodovací* verze TSP, přičemž cílem je rozhodnout, zda pro daný graf G a danou délku L existuje v grafu G Hamiltonovská kružnice délky maximáně L.

1.3.1 Symetrie

Z hlediska orientace hran jsou rozlišovány dvě varianty problému obchodního cestujícího, a to symetrický TSP a asymetrický TSP.

Symetrický předpokládá, že vzdálenost mezi dvěma uzly je stejná v obou směrech, z hlediska teorie grafů je tedy zadán neorientovaný graf.

Naopak u *asymetrického* TSP mohou být cesty v opačných směrech různě dlouhé. Tuto situaci reprezentuje orientovaný graf.

Reálnému životu více odpovídá varianta asymetrického TSP, která dokáže brát v potaz např. jednosměrné ulice nebo sklon trasy. V implementační části a v dalším textu je však uvažováno se symetrickou variantou, protože hráči Geocachingu se obvykle pohybují pěšky a chodcům obvykle trvá cesta oběma směry přibližně stejně dlouho.

1.3.2 Úplnost

Při řešení problému obchodního cestujícího je zadaný graf G obvykle úplný. Úplnost grafu odpovídá i problému řešenému v této práci (trasy spojující místa na zemském povrchu), protože je pravda, že z každého místa existuje cesta do libovolného jiného místa. Touto cestou může být např. ortodroma, nejkratší spojnice dvou bodů na kulové ploše. Navíc, pokud by byl dán graf G, ve kterém neexistuje hrana mezi dvěma uzly, je možné do grafu přidat hranu, která je delší než nejdelší hrana v G, čímž lze vytvořit úplný graf a zároveň se nezmění optimální trasa. Nadále je problém řešen nad úplnými grafy, pokud není řečeno jinak.

1.3.3 Metrika

V obecné variantě problému není vyžadováno, aby v grafu platila trojúhelníková nerovnost, tedy aby nejkratší cesta z místa A do místa B nebyla delší než libovolná cesta z A do B přes další místo C. Avšak v tomto případě, kdy je hledána cesty mezi místy na zemském povrchu, vždy trojúhelníková nerovnost platí. Proto lze uvažovat speciální variantu problému, tzv. *metrický problém obchodního cestujícího*, kde trojúhelníková nerovnost platí. To,

že řešený problém je metrický, je významné, protože poté lze pro řešení použít aproximační algoritmy [2]. Konkrétní algoritmy jsou popsány v kapitole 2.

1.3.4 Obtížnost

Je dán úplný graf $G = \langle H, U, \varrho \rangle$, přičemž $|U| = n$. Kolik různých Hamiltonovských kružnic v něm existuje?

Protože každým uzlem je třeba projít právě jednou, je třeba zjistit, kolik různých pořadí uzlů existuje. Jestliže je dáno n uzlů, tak počet způsobů, v jakém pořadí je lze projít, je $n!$. Při hledání kružnic nezáleží, v jakém místě cesta začne. Např. kružnice $A \to B \to C \to D \to A$ je totožná s kružnicí $C \to D \to A \to B \to C$. Každá kružnice byla započítána $n$krát, protože každá kružnice prochází právě n uzly a v každém z nich mohla uvažovaná kružnice začít. Proto je nutné počet získaných kružnic vydělit číslem n. Nakonec je třeba vzít v potaz, že při hledání Hamiltonovksých kružnic není důležité, v jakém směru jsou procházeny. Např. kružnice $A \to B \to C \to D \to A$ je totožná s kružnicí $A \to D \to C \to B \to A$. Protože každá kružnice byla započítána dvakrát, je nutné získaný počet vydělit dvěma. Výsledný počet P různých Hamiltonovských kružnic v závislosti na počtu uzlů n daného grafu G je tedy

$$P = \frac{(n-1)!}{2}.$$

Hamiltonovských kružnic je exponenciálně mnoho v závislosti na počtu uzlů. Např. již pro 17 uzlů počet různých Hamiltonovských kružnic přesáhne deset biliónů. Tento počet je příliš vysoký na to, aby byly postupně vyzkoušeny všechny kružnice a aby z nich následně byla vybrána ta nejkratší.

Věta 1.1. Problém obchodního cestujícího je NP-těžký.

Důkaz 1.1. Problém H je NP-těžký, pokud existuje problém G, který je NP-úplný a lze ho převést na H v polynomiálním čase. Poté je možné každý problém L z třídy složitosti NP převést v polynomiálním čase na G, tudíž lze L převést i na H v polynomiálním čase.

Rozhodnutí, zda graf obsahuje Hamiltonovskou kružnici, je NP-úplný problém [1]. Podle [3] lze problém hledání Hamiltonovské kružnice (HC) v grafu G vyřešit pomocí TSP, na který je problém HC převeden v polynomiálním čase. Pro řešení je použita rozhodovací verzi TSP popsaná výše.

Dle [3] je možné zredukovat problém hledání HC na TSP následovně: Je dán graf $G = \langle H, U, \varrho \rangle$, přičemž $|U| = n$, a cena hrany e je definována $c(e) = 1$ pro všechny $e \in H$. Poté jsou přidány hrany H' do G tak, aby byl G úplný graf a je definována $c(e) = 2$ pro všechny $e \in H'$. Toto lze provést v polynomiálním čase. Nyní je použita rozhodovací verze TSP. Pokud existuje v G Hamiltonovská kružnice délky maximálně n, potom existuje HC v původním grafu, protože použité hrany musí být z H, jinak by délka byla větší než n.

Protože problém Hamiltonovské kružnice je NP-úplný, rozhodovací verze TSP je také NP-úplná, tudíž optimalizační verze TSP je alespoň tak těžká jako problém HC, tudíž je NP-těžká.

Problém obchodního cestujícího je NP-těžký. Za předpokladu, že $P \neq NP$, neexistuje polynomiální algoritmus, který by našel libovolné řešení, které je nejhůře k-násobkem optimálního řešení. Problém zůstává NP-těžký, i když se jedná o metrický TSP.

Častým omylem je tvrzení, že problém obchodního cestujícího je NP-úplný problém. Aby byl problém NP-úplný, musí být zároveň NP a NP-těžký. Aby byl TSP NP, musí být možné v polynomiálním čase o předložené kružnici rozhodnout, zda je, či není řešením TSP, tzn. ověřit dvě tvrzení:

1. Každý uzel byl navštíven právě jednou.

2. Neexistuje kratší kružnice než ta, o které je rozhodováno.

Ověření první podmínky je triviální. Ale druhou podmínku nelze dosud ověřit v polynomiálním čase [4]. Proto TSP není NP, tudíž není ani NP-úplný.

1.4 Problém orientačního běžce

Zobecněním problému obchodního cestujícího je *problém orientačního běžce* (OP). Tento problém se od TSP liší tím, že hledaná kružnice (trasa) nemusí procházet všemi vrcholy grafu (místy), ale musí mít délku maximálně B. Formálněji řečeno dle [5], je dán graf $G = \langle H, U, \varrho \rangle$, dva uzly $s, t \in U$ a mez B. Úkolem je nalézt v G cestu z s do t, která má délku maximáně B a maximalizuje počet různých uzlů, kterými tato cesta prochází.

Z hlediska programování je OP kombinací TSP s *úlohou batohu*. V úloze batohu je dána množina předmětů, přičemž každý předmět je definován svojí váhou a cenou, a cílem je umístit podmnožinu předmětů do batohu s omezenou nosností tak, aby byla maximalizována cena nákladu.

Protože TSP je speciálním případem OP a zároveň je NP-těžký, je i problém orientačního běžce NP-těžký. Proto se při řešení tohoto problému opět využívá aproximačních algoritmů.

Zvláštním případem OP je *vážený problém orientačního běžce*, ve kterém jsou navíc uzlům přiděleny váhy. Úkolem je nalézt cestu, která maximalizuje součet vah uzlů, kterými tato cesta prochází.

Kapitola 2

Řešení problému obchodního cestujícího

Problém obchodního cestujícího je, jak bylo ukázáno v sekci 1.3.4, NP-těžký. A tak přestože existují algoritmy řešící problému obchodního cestujícího a nacházející optimum, je prakticky nemožné pro větší vstupy získat optimální řešení v rozumném čase. Exaktní algoritmy, ačkoliv jsou vylepšeny (např. metodou větví a hranic), jsou příliš výpočetně náročné, protože problém řeší v exponenciálním čase. Proto je TSP řešen převážně heuristickými algoritmy, které jsou schopny problém řešit v polynomiálním čase, i když nalezená cesta nemusí být optimální.

Mezi nejznámější heuristické algoritmy patří například hladový, 2-aproximační, Christofidesůvs algoritmus, genetické algoritmy, řešení pomocí neuronových sítí ...

Efektivita těchto algoritmů závisí nejen na velikosti zadaného problému, ale také na konkrétním zadání.

Algoritmů pro řešení TSP je velké množství, zmíněny jsou pouze ty nejdůležitější. Větší prostor dán algoritmům, na které se tato práce zaměřuje a které jsou využity a v implementační části této práce (2-aproximační a hladové algoritmy).

Pokud jsou v textu uvedeny ilustrační obrázky grafů, nejsou v nich pro přehlednost uvedeny všechny hrany. Pokud není řečeno jinak, vždy se jedná o úplné grafy, kde vzdálenosti odpovídají euklidovským vzdálenostem uzlů.

2.1 Exaktní algoritmy

Exaktní algoritmy vždy naleznou optimální řešení. Jejich nevýhodou je, že u nemalých problémů nedokážou nalézt řešení v rozumném čase, protože jejich časová složitost je exponenciální. Základní myšlenkou exaktních algoritmů je projít všechna možná řešení a vybrat z nich to nejlepší. Dobu výpočtu lze

zkrátit tím, že se neprochází všechna možná řešení, ale projdou se jen ta, která by mohla být optimální.

2.1.1 Řešení hrubou silou

Nejpřímočařejším řešením je řešení hrubou silou. To znamená vyzkoušet všechny možné permutace uzlů, tedy najít všechny možné trasy a zjistit, která z nich je nejkratší. Časová složitost tohoto řešení je $\mathcal{O}(n!)$.

2.1.2 Held-Karpův algoritmus

O něco lepší výsledky než řešení hrubou silou dává Held-Karpův algoritmus, který řeší problém v čase $\mathcal{O}(n^2 * 2^n)$ [6].

Základem tohoto algoritmu je tvrzení, že každá část nejkratší cesty je sama o sobě také nejkratší cestou na podmnožině uzlů. Tento algoritmus využívá principů dynamického programování. Předem vypočte řešení všech subproblémů (menších množin uzlů), přičemž začne od těch nejmenších a při řešení větších využívá již vypočtená řešení. Díky tomu algoritmus nepočítá stejné výpočty vícekrát a je rychlejší než řešení hrubou silou.

2.1.3 Metoda větví a hranic

Při užití metody větví a hranic (B&B) se systematicky prochází všechna možná řešení, přičemž velké podmnožiny možných řešení nejsou prozkoumávány, pokud nemohou obsahovat řešení, které je lepší než dosud nejlepší nalezené řešení.

Důležitou hodnotou pro B&B je *spodní mez délky hledané cesty*. Cesta by byla nejkratší, pokud by byl každý uzel v cestě napojen dvěma nejkratšími hranami, které z něj vychází (s ohledem na zadaná omezení, které hrany v cestě musí, nebo nesmí být). V prohledávané větvi stavového prostoru není možné nalézt řešení, které by bylo kratší než tato spodní mez. Pokud je množina uzlů grafu označena V, pak spodní mez S lze vyjádřit jako

$$S = \frac{1}{2} \sum_{v \in V} (\text{součet délek dvou nejkratších hran vycházejících z } v).$$

Způsobů, jak aplikovat metodu větví a hranic je více, zde je vycházeno z postupu uvedeného v [7]. Při prohledávání stavového prostoru je postupně určováno, které hrany v řešení musí, nebo nesmí být. Je zvolena hrana e, o které ještě nebylo rozhodnuto, zda v daném řešení musí, či nesmí být a aktuální instance problému se rozvětví na dvě nové, které mají stejná omezení jako aktuální instance a navíc jedna z nich hranu e obsahovat musí a druhá tuto hranu obsahovat nesmí. V první úrovni není u žádné hrany určeno, jestli v grafu musí či nesmí být. V druhé úrovni se vytvoří dvě instance problému, v první instanci musí být jedna hrana e_1 a ve druhé instanci tato hrana být

nesmí. Ve třetí úrovni se opět každá instance rozdělí na dvě další - jedna z nich musí obsahovat další hranu e_2, druhá ji obsahovat nesmí. Takto se rekurzivně pokračuje, dokud není pro každou hranu určeno, zda v řešení musí, nebo nesmí být.

Je očividné, že problém se velice rychle větví na mnoho instancí. Postupně je pro každou hranu určeno, jestli ji podproblém musí, či nesmí obsahovat. Teoreticky tak vzniká ve výsledku až $2^{\text{počet hran}}$ možných řešení. Ve skutečnosti je prohledaných řešení méně, protože vždy, když jsou určena nová omezení, které hrany v cestě musí, nebo nesmí být, je kontrolováno, zda jsou splněny následující podmínky:

1. V grafu nevznikl cyklus, který neprochází všemi uzly.
2. Z žádného uzlu nevychází tři hrany, které v cestě musí být.
3. Každý uzel musí mít alespoň dvě hrany, které nejsou označeny, že v cestě nesmí být.
4. Spodní mez nově vzniklé instance je nižší než nejlepší dosud nalezené řešení.

Pokud tyto podmínky nejsou splněny, v prohledávání této větve stavového prostoru dále nepokračujeme, tato větev je tzv. odříznuta.

Při rekurzivním prohledávání je řešení hledáno nejprve v perspektivnějším potomkovi, tedy v potomkovi s nižší spodní mezí. Pokud je v perspektivnějším potomkovi nalezeno řešení, které je kratší než spodní mez jeho sourozence, nemusí být tento sourozenec prohledáván a je zkrácena doba výpočtu.

Algoritmus využívající metodu větví a hranic je velmi datově citlivý. Vždy záleží, jak velká část stavového prostoru je odříznuta. Může se stát, že nedojde k žádnému odříznutí a poté je tato metoda stejně pomalá jako řešení hrubou silou. V průměrném případě však dochází ke značnému zrychlení. Srovnání doby výpočtu bez využití této metody a s ní je v tabulce 2.1.

2.2 Heuristické a aproximační algoritmy

Problém obchodního cestujícího lze řešit heuristickými a aproximačními algoritmy, které dodávají v rozumném čase, obvykle polynomiálním, vyhovující řešení i pro větší vstupy. Toto řešení nemusí být optimální. Oproti exaktním algoritmům jsou však tyto algoritmy schopné dodat řešení i pro extrémně velké problémy (až miliony uzlů), přičemž je velká pravděpodobnost, že řešení nebude o mnoho delší než optimální řešení a že pro praktické účely bude toto řešení postačující.

Heuristické algoritmy při prohledávání stavového prostoru neprohledávají všechny stavy, ale volí, na základě dostupných informací, kterou větev pro

Tabulka 2.1: Doba výpočtu v sekundách bez a s metodou větví a hranic (B&B)

Vstup	Bez metody B&B	S metodou B&B
input10 (10 uzlů)	0,062	0,012
input10b (10 uzlů)	0,036	0,004
input11 (11 uzlů)	0,378	0,013
input12 (12 uzlů)	11,93	0,020
input12b (12 uzlů)	10,82	0,002
input13 (13 uzlů)	72,40	0,063
input13b (13 uzlů)	64,51	0,058
input14b (14 uzlů)	172,80	0,057
input14c (14 uzlů)	658,61	0,152
input15 (15 uzlů)	6425,96	0,114
input15c (15 uzlů)	3516,47	0,115
input16 (16 uzlů)	více než 10 hodin	0,480
input16b (16 uzlů)	-	1,18
input20 (20 uzlů)	-	6,09
input21 (21 uzlů)	-	25,54
input22 (22 uzlů)	-	6,04
input23 (23 uzlů)	-	16,02
input24 (24 uzlů)	-	551,74

hledání řešení zvolit. Například u TSP zvolí, která hrana bude součástí řešení. Přestože heuristické algoritmy produkují často velmi dobré výsledky, není u nich obecně garantována žádná kvalita ani rychlost nalezeného řešení.

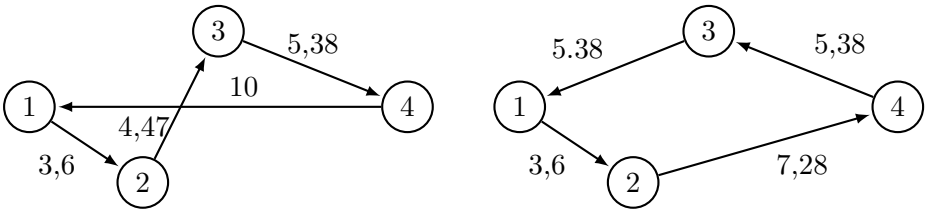
Doba potřebná pro běh heuristických algoritmů je závislá na velikosti úlohy i na požadované přesnosti řešení.

Heuristické algoritmy je možné rozdělit na *konstruktivní*, které vytváří řešení, a *iterativní*, které zlepšují řešení.

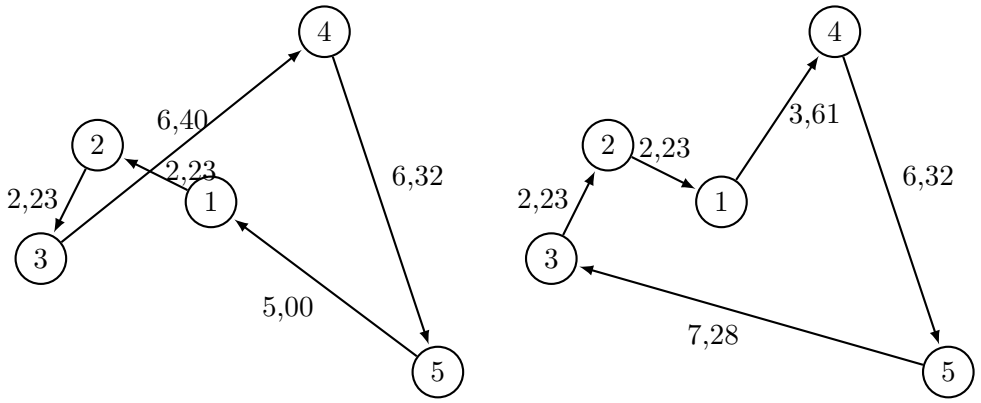
Aproximační algoritmy jsou podmnožinou heuristických algoritmů. Od těch se liší tím, že garantují určitou kvalitu řešení. Obvykle je dán aproximační faktor α, který udává, že řešení získané daným aproximačním algoritmem je v nejhorším případě αkrát horší než optimální řešení. Další důležitou vlastností těchto algoritmů je, že vždy skončí v polynomiálním čase. Tyto algoritmy existují pouze pro metrický TSP. Pro obecný TSP neexistuje α-aproximační algoritmus, kde by faktor α byl konstantou [8].

2.2.1 Hladový algoritmus

Hladový algoritmus, neboli také algoritmus nejbližšího souseda, je heuristický konstruktivní algoritmus pro řešení TSP. Je zvolen počáteční uzel trasy a vždy jako další uzel je vybrán nejbližší uzel, který dosud není součástí trasy.



Obrázek 2.1: Srovnání neoptimální cesty (vlevo), kterou nalezne hladový algoritmus, s optimální cestou (vpravo)



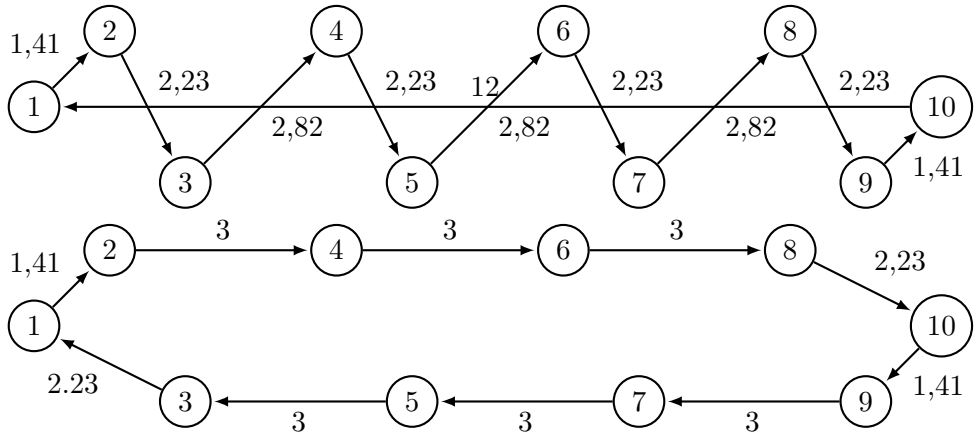
Obrázek 2.2: Pokud je počátečním uzlem uzel číslo jedna (vlevo), je trasa delší, než pokud je počátečním uzlem uzel číslo tři(vpravo)

Časová složitost tohoto algoritmu je $\mathcal{O}(|U|^2)$, protože pro každý uzel je hledán uzel, který je tomuto uzlu nejblíže a dosud nebyl zahrnut do řešení.

Nevýhodou tohoto algoritmu je, že na konci jsou obvykle přidávány ty nejvzdálenější uzly od uzlu počátečního. Z naposledy přidaného uzlu je třeba se opět vrátit do počátečního uzlu, aby vznikla kružnice. Hrana spojující počáteční a naposledy přidaný uzel je obvykle velice dlouhá. Výhodnější by bylo, kdyby se nejvzdálenější uzel nacházel přibližně v polovině trasy. Situace je znázorněna na obr. 2.1, kde hladový algoritmus nalezl trasu 1-2-3-4-1. Jako poslední byl přidán uzel číslo čtyři, který je nejdále od počátečního uzlu číslo jedna, a po jeho přidání bylo třeba se opět vrátit do uzlu číslo jedna. Tato trasa je dlouhá přibližně 23,45 a je neoptimální. Kratší trasu tvoří např. kružnice 1-2-4-3-1, která je dlouhá 21,64.

Výsledné řešení silně závisí na volbě počátečního uzlu. Může být výhodnější začít v uzlu, který se nachází „na kraji“ množiny uzlů, než v uzlu, který se nachází „uprostřed“ množiny uzlů. Rozdílné trasy při volbě různých počátečních uzlů jsou znázorněny na obr. 2.2. Pokud je zvolen počátečním uzlem uzel číslo jedna, je trasa delší, než pokud je počátečním uzlem uzel číslo tři.

Pro zlepšení výsledků této heuristiky je možné algoritmus opakovat s tím, že je pokaždé zvolen jiný počáteční uzel.



Obrázek 2.3: Srovnání špatného řešení (nahoře), které nalezne hladový algoritmus, s optimální cestou (dole)

Existují zadání, kde hladový algoritmus nachází velice špatnou cestu, příklad takového zadání je na obr. 2.3. V tomto případě hladový algoritmus nalezne řešení délky 31,66, přestože optimální řešení, kde uzly jsou procházeny v pořadí 1-2-4-6-8-10-9-7-5-3-1, má délku 25,28.

Judith Brecklinghaus a Stefan Hougardy v [9] dokázali, že hladový algoritmus má, pro zadání splňující trojúhelníkovou nerovnost, aproximační faktor $\Theta(\log n)$, kde n je počet uzlů. Tudíž nalezené řešení je v nejhorším případě $\Theta(\log |U|)$krát horší než optimální řešení.

2.2.2 2-aproximační algoritmus

2-aproximační algoritmus produkuje řešení, které je v nejhorším případě dvakrát delší než optimální řešení. Tento algoritmus je, jak již bylo řečeno, validní pouze pro metrický problém obchodního cestujícího.

Pro navržení aproximačního algoritmu je třeba znát spodní mez optimálního řešení. Jen tak lze určit, kolikanásobku optimálního řešení je navrženým algoritmem dosaženo. Při řešení problému obchodního cestujícího spodní mez optimálního řešení určuje minimální kostra grafu(MST). Minimální kostra grafu $G = \langle H, U, \varrho \rangle$, ve kterém délka hrany e je rovna $w(e)$, je takový souvislý podgraf $G' = \langle H', U, \varrho \rangle$, kde hodnota $w(H')$

$$w(H') = \sum_{e \in H'} w(e)$$

je minimální.

Pokud by z optimálního řešení TSP byla odebrána jedna hrana, vznikla by kostra grafu. Tudíž je očividné, že délka optimální trasy není nikdy kratší než délka MST, protože MST obsahuje $|U| - 1$ minimálních hran, zatímco Hamiltonovská kružnice jich obsahuje $|U|$.

Definice 2.1. *Eulerovým grafem* je takový neorientovaný souvislý graf, jehož každý uzel má sudý stupeň [1].

Definice 2.2. *Eulerovým tahem* je uzavřený nebo otevřený tah obsahující všechny hrany grafu [1].

2-aproximační algoritmus je popsán v [8] následovně:

1. Je nalezena minimální kostra G.
2. Každá hrana minimální kostry je zduplikována, takže vznikne Eulerův graf.
3. Je nalezen Eulerův tah J vzniklého Eulerova grafu.
4. Z tahu J je vytvořena kružnice T tak, že jsou procházeny uzly ve stejném pořadí, jako se vyskytují v J, s tím, že jsou přeskočeny uzly, které již byly navštíveny.

Není složité ukázat, že uvedený algoritmus je 2-aproximační pro metrický TSP. Jak bylo řečeno, délka MST $\leq$ délka optimálního řešení. Protože tah J obsahuje každou hranu MST dvakrát, délka tahu J je rovna dvojnásobku délky MST. Protože platí trojúhelníková nerovnost, přeskakování již nalezených uzlů nezvětší délku řešení. Tudíž pro délku kružnice T platí

$$\text{délka } T \leq 2 \times \text{délka MST} \leq 2 \times \text{délka optimálního řešení},$$

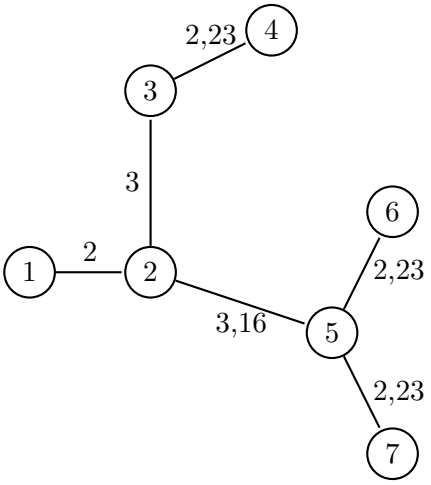
proto má uvedený algoritmus aproximační faktor dva.

Eulerův tah lze nalézt tak, že kostra grafu je projita z libovolného uzlu do hloubky a jsou poznamenány všechny průchody přes vrcholy, tzn. je poznačen každý uzel, kterým je procházeno, i když už jím je procházeno poněkolikáté.

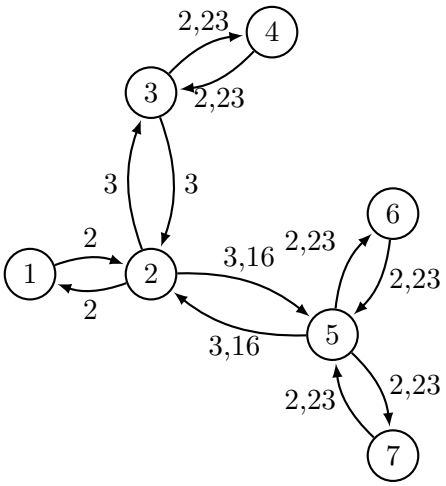
Projitím do hloubky se rozumí takový algoritmus, který zvolí libovolný uzel a označí jej jako otevřený, zavolá sám sebe na všechny dosud neotevřené sousedy daného uzlu a po návratu z rekurze uzel označí jako uzavřený.

Časová složitost 2-aproximačního algoritmu je polynomiální. Nalezení minimální kostry může mít, v závislosti na zvoleném algoritmu a implementaci, složitost $\mathcal{O}(|H|log|U|)$ [1]. Druhý krok algoritmu má lineární složitost v závislosti na počtu hran. Nalezení Eulerova tahu má složitost $\mathcal{O}(|U|+|H|)$, protože ke každé hraně i ke každému uzlu je přistupováno právě jednou. Složitost čtvrtého kroku je také lineární, protože prochází uzly obsažené v Eulerově tahu, kterých je o jedna více, než kolik bylo v Eulerově grafu hran.

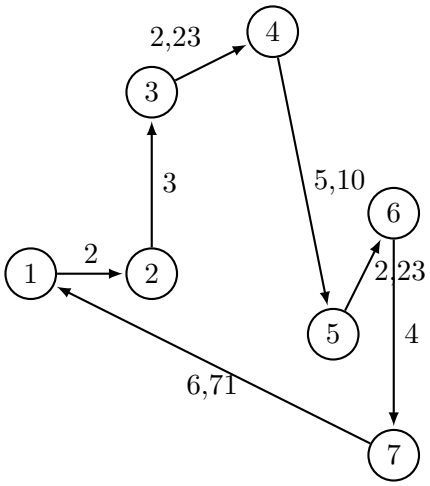
Průběh 2-aproximačního algoritmu je zobrazen na obr. 2.4.



(a) Minimální kostra grafu



(b) Eulerův tah projde uzly v pořadí 1-2-3-4-3-2-5-6-5-7-5-2-1



(c) Kružnice vzniklá po přeskočení opakujících se uzlů v Eulerůvě tahu

Obrázek 2.4: Vizualizace 2-aproximačního algoritmu

2.2.3 Další aproximační algoritmy

Aproximačních algoritmů pro řešení metrického problému obchodního cestujícího existuje velké množství. Bylo dokázáno, že aproximační faktor nemůže být, za předpokladu, že $P \neq NP$, menší než $\frac{123}{122}$ [10].

Vylepšením 2-aproximačního algoritmu je Christofidesův algoritmus, který má aproximační faktor $\frac{3}{2}$. Změna spočívá v lepší konstrukci Eulerova grafu ve druhém kroku algoritmu, kdy se neduplikují všechny hrany, ale pouze se přidají hrany tak, aby každý uzel byl sudého stupně [8].

Zajímavým způsobem, jak řešit metrický TSP, jsou genetické algoritmy či neuronové sítě. Tyto přístupy se blíží optimálnímu řešení o něco více než dosud uvedené algoritmy, ale jsou časově mnohem náročnější [11], tudíž jsou pro implementační část této práce nevhodné.

2.2.4 K-opt heuristika

K-opt heuristika je iterativní optimalizační algoritmus, který vylepšuje řešení nalezené jiným algoritmem.

Algoritmus této heuristiky, jak je popsán v [2], se skládá ze dvou hlavních kroků:

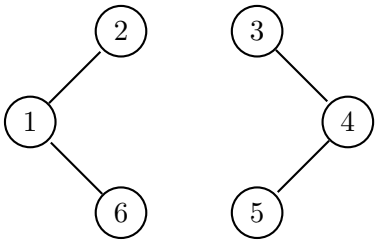
1. Je dána počáteční kružnice (trasa nejlepšího řešení nalezená jiným algoritmem).

2. Z kružnice je odebráno k hran. Tím vznikne k fragmentů původní kružnice. Tyto fragmenty jsou spojeny všemi možnými způsoby tak, aby v grafu vznikla právě jedna kružnice, která prochází všemi uzly grafu. Pokud nějaký způsob spojení fragmentů vytvořil kružnici kratší, než byla počáteční, je tato kružnice považována za počáteční kružnici a je opakován krok 2. Pokud není možné dosáhnout žádného zlepšení, algoritmus končí.

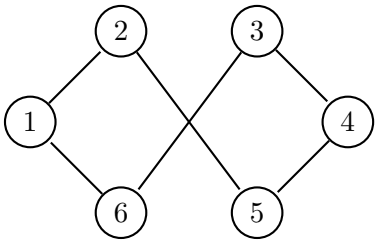
V jednom kroku algoritmu je prozkoumáno řádově $|U|^k$ možných řešení, protože možností, jak z $|U|$ uzlů vybrat k uzlů, je $\binom{|U|}{k}$ a možností, v jakém pořadí se bude k fragmentů vyskytovat ve výsledné kružnici, je řádově $k!$ [2].

S rostoucím k se exponenciálně zvyšuje výpočetní náročnost algoritmu. Proto jsou nejpoužívanějšími k-opt heuristikami 2-opt a 3-opt, protože značně vylepšují řešení a zároveň jsou dostatečně rychlé.

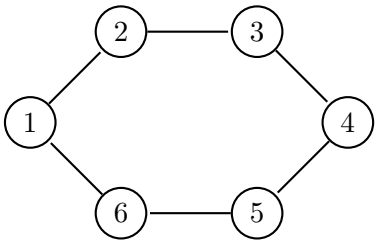
2-opt heuristika tedy zkusí vždy odebrat dvě hrany a vzniklé dva fragmenty spojit jiným způsobem, než byly spojeny předtím. Možnosti, jak znovu vložit dvě odebrané hrany, jsou tři, avšak jednu z nich nemá smysl uvažovat, protože by vznikly dva oddělené cykly, jak je znázorněno na obr. 2.5. Protože je zároveň vyžadováno jiné spojení, než bylo původně, je způsob přepojení dán jednoznačně.



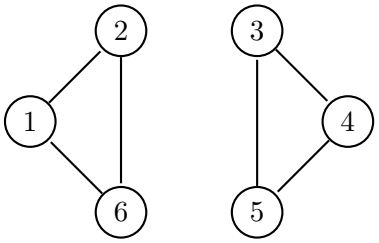
(a) Dva fragmenty grafu, které jsou spojovány



(b) První možnost spojení fragmentů



(c) Druhá možnost spojení fragmentů



(d) Vložení hran tímto způsobem není validní, protože vznikly dva cykly

Obrázek 2.5: Možnosti spojení dvou fragmentů grafu

K-opt heuristika tedy často dokáže zamezit tomu, aby se výsledná trasa kdekoliv křížila, což se například při použití hladového algoritmu stává relativně často (např. na obr. 2.2).

2.3 Výběr podmnožiny uzlů

V praxi se ukazuje, že často je třeba řešit spíše problém orientačního běžce než problém obchodního cestujícího. Tedy že požadujeme, aby hledaná trasa měla maximálně délku B a maximalizovala počet vrcholů, kterými tato cesta prochází. To je případ i nacházení optimální trasy pro hru Geocaching, protože je dána velká množina keší a cílem je nalézt trasu obsahující pouze určitou podmnožinu těchto keší.

Jedním možným způsobem řešení tohoto problému je nějakou heuristikou vybrat podmnožinu uzlů a nad ní následně spustit algoritmus pro řešení problému obchodního cestujícího. Druhou možností je rovnou řešit problém orientačního běžce.

2.3.1 Hladový výběr

Protože je tato práce zaměřena na hladové algoritmy, jedním z uvažovaných způsobů pro výběr vhodné podmnožiny uzlů je hladový výběr.

Hladový výběr je možno udělat například tak, že je vybráno k nejbližších uzlů vzhledem k počátečnímu uzlu. Vhodné k, vzhledem k zadané maximální délce B, lze určit binárním vyhledáváním.

Jiným způsobem, jak provést hladový výběr, je spustit hladový algoritmus (popsaný v sekci 2.2.1). Pokaždé, když je do trasy přidán nový uzel, je zkontrolováno, zda součet dosavadní délky trasy a vzdálenosti do počátečního uzlu není větší než zadaná maximální délka B.

Oba výše zmíněné hladový výběry snadno uváznou v lokálním optimu, a tak se ukazuje jako výhodné spustit hladový výběr vícekrát a pokaždé zakázat určitou podmnožinu uzlů. Například uzel, který byl vybrán ihned po počátečním uzlu. Nebo povolit jen uzly, které se nachází na určité světové straně od počátečního uzlu.

2.3.2 Řešení problému orientačního běžce

Chekuri v [5] dokázal, že pro libovolné pevné $\varepsilon > 0$ existuje algoritmus, který má časovou složitost $n^{\mathcal{O}(1/\varepsilon^2)}$ a na neorientovaných grafech aproximuje optimální řešení problému orientačního běžce s aproximačním faktorem $(2 + \varepsilon)$.

Pro praktické řešení je postačující 3-aproximační algoritmus řešení problému orientačního běžce, který je uveden v [12]. Tento algoritmus postupně zkouší všechny možné dvojice druhého a předposledního uzlu trasy, mezi které vkládá další uzly. Tyto další uzly jsou vybírány pomocí $(2+\epsilon)$-aproximačního algoritmu, který řeší problém minimálního přebytku [13]. Zároveň je určen maximální povolený přebytek, tedy o kolik může být trasa přes vložené uzly delší než přímá trasa z druhého do předposledního uzlu, tak, aby nebyla překročena stanovená maximální délka kompletního řešení B.

Jestliže je dán počáteční uzel s a koncový uzel t cesty a dále je dána norma k, algoritmus řešící *problém minimálního přebytku* nalezne cestu z s do t, která prochází minimálně dalšími k uzly a zároveň má minimální přebytek.

Přebytkem se rozumí rozdíl délky nově vzniklé cesty a vzdálenosti uzlů s a t.

Pro užití algoritmu minimálního přebytku je třeba znát hodnotu normy k, která musí být maximální možná a zároveň taková, aby přebytek nebyl větší, než je maximální povolený přebytek B daný 3-aproximačním algoritmem. Toto k je nalezeno pomocí binárního vyhledávání.

To, že algoritmus pracuje s počátečním a koncovým uzlem, není překážkou pro vytváření kružnic, protože nikde není řečeno, že by počáteční uzel u musel být různý od koncového uzlu v.

Důležité je zmínit, že algoritmus pro řešení OP je aproximační co do počtu navštívených uzlů, tedy výsledná trasa prochází minimálně třetinou uzlů, kterými prochází optimální trasa. Maximální délka B není nikdy překročena.

Kapitola 3

Geocaching

Geocaching je hra spojující turistiku s dobrodružným hledáním schránek. Těmto schránkám se také říká keše.

Základní principem hry je nalézt, neboli také ulovit, ukrytou keš. Její poloha je určena zeměpisnými souřadnicemi zveřejněnými na internetu. Pokud hráč keš nalezne, zapíše svůj nález do sešitku (logbooku) přímo v keši a následně ho zaznamená i na internetu.

Díky různorodosti keší je Geocaching zábavou pro různé skupiny osob. Některé schránky jsou umístěny na turisticky zajímavých místech. Jiné jsou na těžko dosažitelných místech a jejich odlov vyžaduje podání fyzicky náročného výkonu. Milovníky šifer a přemýšlení potěší keše, jejichž souřadnice nejsou přímo dané, ale musí být nejprve vyluštěny. Hledání keší je oblíbeno také u dětí, kterým Geocaching ozvláštňuje klasické výlety, které jsou jimi často považovány za nudné. Schránky mohou navíc obsahovat vyměnitelné předměty, což je pro děti velmi lákavé.

Historicky první keš vznikla 3. května 2000 v Oregonu ve Spojených státech amerických, kdy Dave Ulmer ukryl v lese schránku, její souřadnice uveřejnil na internetu a vyzval ostatní, aby se ji pokusili najít. Tím položil základy Geocachingu, který na prakticky shodném principu funguje dodnes. [14]

Postupem času se Geocaching částečně komercializoval. Existují keše, které mohou nalézt pouze platící hráči, ostatním jsou informace o těchto keších, například jejich přesné souřadnice, skryty. Z tohoto důvodu vzniklo několik jiných variant hry, například Opencaching [15], který je založen na principu opensource.

Geocachingu se dnes věnuje přibližně patnáct milionů lidí na celém světě [14]. Hráči si musí při hledání počínat nenápadně, aby neprozradili existenci keše lidem, kteří hru nehrají. Ti by mohli schránku schválně poničit či odcizit.

3.1 Navigace

Základním prvkem Geocachingu je hledání míst na základě zadaných zeměpisných souřadnic. Hráč se obvykle orientuje pomocí signálu, který vysílají družice na oběžné dráze Země. Může se také orientovat podle mapy, do které jsou zaneseny souřadnice keší. Tato metoda je vhodná zejména pro hledání ve městech, kde bývá špatný družicový signál. Použití klasické mapy naopak není vhodné při hledání mystery či multi keší (tyto pojmy jsou popsány v sekci 3.2.1), u kterých se hráč dozví finální souřadnice umístění schránky až v průběhu lovu. Hledání zeměpisných souřadnic na mapě je velmi obtížné a pro Geocaching příliš pomalé a pracné.

3.1.1 Zeměpisné souřadnice

K jednoznačnému určení polohy na povrchu Země slouží zeměpisné souřadnice. Jak uvádí [16], skládají se ze tří složek, přičemž ta první, vzdálenost od středu Země, se obvykle nepoužívá. Druhou složkou je *zeměpisná šířka*, což je úhlová vzdálenost od rovníku, tedy úhel, který svírá rovina rovníku s normálou referenční plochy v příslušném bodě na povrchu Země. Třetí složkou je *zeměpisná délka*, která udává úhlovou vzdálenost od nultého poledníku.

3.1.2 Vzdálenost dvou bodů

Nejkratší spojnicí dvou bodů na kulové ploše, tedy i na Zemi, je *ortodroma* [16]. Ortodromu tvoří kratší oblouk hlavní kružnice. Hlavní kružnice má střed ve středu Země a prochází oběma body, mezi kterými je hledána nejkratší vzdálenost. Délku ortodromy d mezi body $[\phi_1, \lambda_1]$ a $[\phi_2, \lambda_2]$ lze vyjádřit vzorcem

$$d = \frac{2\pi}{360}\sigma r,$$

kde r je poloměr Země a σ je středový úhel. Středový úhel σ lze vypočítat ze souřadnic bodů:

$$\sigma = \arccos(\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos(\lambda_2 - \lambda_2)).$$

3.1.3 Globální družicový polohový systém

Globální družicový polohový systém (GNSS) [17] je služba, která umožňuje určit polohu na Zemi. Na základě signálu vyslaného z družic na oběžné dráze Země lze vypočítat polohu s přesností na jednotky metrů. Armádní zařízení jsou schopna určit polohu s přesností až na několik centimetrů.

GNSS se skládá ze tří složek: kosmické (družice), řídící (pozemní stanice) a uživatelské (přijímače).

Řídící složka monitoruje družice a zasílá jim povely, např. změnu letové dráhy či korekci hodin v družici. V současné době jsou plně funkční dva GNSS, a to americký GPS a ruský GLONASS.

GPS je plně funkční od roku 1995 a jeho součástí je 32 družic, které obíhají ve výšce 20 350 km nad povrchem Země po šesti kruhových dráhách. GLONASS je plně funkční od roku 2011 a zahrnuje 24 družic, které obíhají pouze po třech dráhách. [17]

Běžný uživatel může využívat družic z obou systémů zároveň, což má za následek rychlejší a přesnější určení polohy.

Za zmínku stojí evropský navigační systém Galileo, který je vyvíjen, aby evropské státy nebyly závislé na americkém GPS. Galileo momentálně není provozuschopný, protože jeho součástí je pouze dvanáct družic. Tento systém by měl být přesnější než GPS a jeho plná funkčnost se očekává do roku 2020.

3.1.4 Výpočet polohy

Kolem Země obíhají družice, které vysílají rádiové vlny s informacemi o své poloze. Součástí vysílaného signálu je i časová značka udávající dobu jeho odeslání. V místě, kde se měří poloha, je umístěn přijímač. Ten přijme a dekóduje signál vyslaný z družice a poznamená si čas přijetí signálu. Přijímač vypočte svoji vzdálenost od družice d jako součin rychlosti šíření rádiových vln (300 000 km/h) a času, který uběhl od vyslání dat z družice v okamžiku přijetí.

Pro určení polohy v prostoru, jak uvádí [18], je třeba určit tři souřadnice X, Y a Z. Protože však hodiny v přijímači nejsou pro potřeby výpočtu dostatečně přesné a nejsou plně synchronizované s hodinami družice, je čas uživatele T také chápan jako proměnná. Je známa poloha (x, y, z) a časová značka (t) družice. Také je známa rychlost šíření rádiových vln c. Pro výpočet vzdálenosti přijímače od družice je porovnána euklidovská vzdálenost dvou bodů v prostoru (družice a přijímače) a dráha d, která byla vyjádřena výše:

$$\sqrt{(X-x)^2 + (Y-y)^2 + (Z-z)^2} = (T-t)c.$$

Tato rovnice obsahuje čtyři neznámé. Pokud jsou známy hodnoty x, y, z a t pro čtyři družice, lze sestavit a vyřešit soustavu čtyř rovnic a zjistit polohu a čas přijímače. Přesnost hodin je velice důležitá. Změna časové značky o jednu mikrosekundu znamená odchylku přibližně 300 metrů.

3.1.5 Zdroje nepřesnosti navigace

Zdroje nepřesností GNSS je možné rozdělit na umělé a přirozené.

Přirozené zdroje nepřesností jsou především fyzikální a meteorologické vlivy, které nemohou být eliminovány.

Systémy však často obsahují i uměle zavedené nepřesnosti. Systém GPS až do roku 2000 obsahoval umělou chybu, která zapříčiňovala průměrnou odchylku při určování polohy pro běžné uživatele přibližně padesát metrů. Tato chyba byla odstraněna, avšak dodnes mají běžní uživatelé k dispozici méně přesný čas než armáda Spojených států amerických. Přijímače vyrobené v USA

musí mít nastavená omezení do výšky 18 kilometrů a rychlosti do 515 m/s. Tato opatření zabraňují zneužití systém GPS pro navigaci balistických raket či střel s plochou dráhou letu [19].

Přesnost výpočtu polohy přijímače je negativně ovlivňována mnoha vlivy [17]. Mezi nejvýznamnější patří:

1. Drobné změny dráhy družice vlivem kolísání tíhových sil Země, Slunce a Měsíce a vlivem sluneční jaderné aktivity.

2. Nepřesnost družicových hodin. Hodiny v družicích je třeba synchronizovat s časem na Zemi, protože kvůli vysoké rychlosti družic se hodiny v nich zpomalují (dle speciální teorie relativity). Zároveň se družicové hodiny zrychlují oproti hodinám na Zemi (dle obecné teorie relativity), protože intenzita gravitačního pole Země na oběžné dráze je značně menší než na povrchu.

3. Změna rychlosti rádiového signálu v závislosti na počtu elektronů v ionosféře.

4. Vliv troposféry, kde rychlost signálu ovlivňují meteorologické vlivy, například teplota, tlak nebo vlhkost.

5. Vícecestné šíření signálů. Pokud jsou blízko antény přijímače odrazové materiály, mohou být zachyceny signály odražené například od země nebo od skleněné budovy.

Dopady těchto vlivů mohou být částečně eliminovány pomocí *referenčních stanic* [17]. Referenční stanice jsou zařízení na Zemi, která znají svoji přesnou polohu. Stanice změří svoji polohu pomocí signálu z družic, vyhodnotí odchylku od skutečného stavu a vypočítá korekce. Zjištěné korekce odešle uživateli prostřednictvím internetu. Pokud je uživatel dostatečně blízko referenční stanice, může pomocí spočtených korekcí určit svoji polohu s přesností i na několik centimetrů.

3.2 Hledané schránky

Typickou schránkou je malá, voděodolná krabička, kterou ukryje jiný hráč Geocachingu. Ten umístí na internet souřadnice úkrytu společně s doprovodnými informacemi, tzv. listingem. V něm jsou uvedeny všechny informace o keši, především její souřadnice, typ, velikost, obtížnost a terén. V listingu také bývá vysvětleno, čím je místo uložení schránky zajímavé a proč zde byla keš ukryta. Schránka by měla být vždy umístěna na místě, které je z nějakého důvodu zajímavé, například se blízko nachází přírodní či historická památka nebo je spjato s životem známé osobnosti.

Keše lze třídit do skupin podle různých vlastností. Základní rozdělení je popsáno na oficiálních webových stránkách hry [14].

3.2.1 Typ schránky

Rozlišuje se přes dvacet typů schránek. Jednotlivé typy se liší obtížností, časovou náročností i způsobem odlovu. 95% keší patří do tří nejběžnějších skupin:

- *Tradiční keš* – nejčastější typ, nachází se přímo na souřadnicích uvedených v listingu.
- *Mystery keš* – finální souřadnice keše nejsou uvedeny v listingu a hráč je musí získat vyluštěním šifry, výpočtem, nebo jiným způsobem.
- *Multi keš* – při hledání této schránky musí hráč projít několik míst, neboli stagí. Na souřadnicích uvedených v listingu hráč získá informace, jak se dostat na druhou stage. Tam hráč nalezne indicie, které ho dovedou na třetí místo... Postupně hráč projde několik stagí, než se dostane k finálnímu umístění schránky. Tento typ se používá, pokud je účelem provést hráče po více zajímavých místech. Hledání obvykle zabere řádově více času, než hledání tradiční keše.

3.2.2 Velikost schránky

Schránka může být prakticky libovolně velká, jedinou podmínkou je, aby se do ni vešel kus papíru sloužící jako logbook. Z hlediska velikosti se rozlišuje pět typů keší:

- *Micro* – lze ji nenápadně ukrýt téměř všude, proto se používá hlavně ve městech, kde není prostor pro umístění větších schránek. Její objem je přibližně do 100 ml, obvykle se jedná o krabičku od 35 mm filmu, krabičku od léků či PET prefabrikát. Do této schránky se obvykle vejde pouze malý logbook a tužka.
- *Small* – menší schránka o objemu sto mililitrů až jeden litr. Lze do ni vložit menší předměty, např. samolepky a figurky z Kinder vajíček. Typicky se jedná o svačinový plastový box.
- *Regular* – dostatečně velká schránka, do které lze uložit většinu vyměňovatelných a sledovatelných předmětů. Její objem je přibližně jeden až dvacet litrů. Nevýhodou této schránky je, že se hůře hledá dostatečně velké místo na její ukrytí.
- *Large* – největší schránka, jejíž objem přesahuje dvacet litrů. Na většině míst není možné nalézt vhodné místo pro ukrytí takto velké schránky, proto nejsou large keše tolik rozšířené. Obvykle však obsahují zajímavé předměty, protože se do nich vejde prakticky cokoliv.
- *Other* – jedná se o keš atypického tvaru. Může mít podobu například magnetické fólie s připevněným listem papíru jako logbookem. Do těchto schránek obvykle není možné vkládat předměty.

3.2.3 Obtížnost schránky

Obtížnost udává, jak těžké je objevit, kde je keš ukryta. U netradičních keší zároveň vypovídá o náročnosti rozluštění cílových souřadnic. Určení obtížnosti schránky je diskutabilní, protože každý má na obtížnost úkrytu či šifry jiný názor.

- Obtížnost 1 – snadno identifikovatelná skrýš, vhodná i pro úplné začátečníky.
- Obtížnost 2 – jednoduchá multi keš nebo tradiční keš se standardním ukrytím.
- Obtížnost 3 – složité multi keše, mystery vyžadující domácí přípravu či tradiční keše s důmyslným úkrytem.
- Obtížnost 4 – keše zahrnující složité šifry nebo velice dlouhé multi keše.
- Obtížnost 5 – nejtěžší keše s vysokou náročností na intelekt lovce.

3.2.4 Terén schránky

Terén je veličina vyjadřující obtížnost pohybu při cestě ke schránce na stupnici od jedné do pěti. Její hodnota se uvádí pro průměrného člověka a za ideálních podmínek. Dělení do pěti kategorií je následující:

- Terén 1 – silnice a chodníky, zpevněné cesty. Vhodné i pro maminky s kočárky a vozíčkáře. Keš je umístěna ve výšce 60–100 cm nad zemí, aby na ni bylo možné snadno dosáhnout z vozíku.
- Terén 2 – nezpevněné cesty, vyšlapané cestičky, schody. Terén nečiní problém zdravému jedinci ani dětem.
- Terén 3 – obtížnější pohyb, např. strmý kopec, hustá vegetace. Většina lidí je schopna projít tento terén s drobnými obtížemi.
- Terén 4 – velmi strmé svahu, skalky, močály. Terén je přístupný pouze zdatnějším jedincům. Do této kategorie patří i všechny keše, které jsou umístěny výše než dva metru nad zemí a pro které se musí vylézt například na strom.
- Terén 5 – velmi náročný terén vyžadující použití speciálního vybavení, například horolezeckého či potápěčského.

3.2.5 Hodnocení schránky

Uživatelé mohou ohodnotit kvalitu schránky prostřednictvím doplňku pro webovou stránku Geocaching. Tento doplněk se jmenuje GCVote [20] a umožňuje hodnocení na pětihvězdičkové stupnici.

Dalším ukazatelem kvality keše je počet tzv. *bodů oblíbenosti*. Platící uživatelé získají za každých deset nalezených schránek jeden bod oblíbenosti. Ten mohou věnovat keši, kterou v minulosti nalezli a která jim připadala výjimečně dobrá.

Obě zmíněná hodnocení jsou přístupná všem uživatelům. Ti si na jejich základě mohou například vybírat k odlovu pouze keše s dobrým hodnocením. Dobré hodnocení zpravidla vypovídá o zajímavém provedení schránky, její čistotě a obsahu nebo o kvalitně napsaném listingu.

3.2.6 Obsah schránky

Ve schránce se obvykle nachází logbook, tužka a další předměty. Pokud hráč keš nalezne, zapíše tuto skutečnost do logbooku. Hráč může do keše vložit nějaký předmět a na oplátku z ní vyjmout jiný. Vložený předmět by měl mít hodnotu vyšší nebo rovnu hodnotě vyjmutého předmětu, jinak dochází ke znehodnocování keše.

Mezi obvykle vyměňované předměty patří drobné hračky, například figurky z Kinder vajíček, sběratelské obrázky a samolepky. Velké schránky mohou obsahovat větší předměty, například knihy a kompaktní disky.

Další skupinou předmětů jsou sledovatelné předměty. Sledovatelným předmětem je obvykle drobná hračka, která má svůj identifikační kód a hráči ji přenášejí mezi kešemi. Na internetu je zaznamenáváno, kterými schránkami předmět putuje. Obvykle je definováno,do jakých schránek má být předmět umisťován, nebo kam má doputovat. Například má být vkládán pouze do schránek, které jsou umístěny na stromech, nebo je jeho cílem doputovat do Japonska.

3.3 Hardware a software pro Geocaching

Hráči Geocachingu se bez výpočetní techniky neobejdou. Informace o schránkách, které mohou hledat, jsou dostupné na internetovém serveru Geocaching.com [14]. Nejdostupnější způsob, jak vyhledat keše, je použití webového prohlížeče. Pokud hráč nevlastní žádný vhodný hardware, může si polohu schránek poznamenat do klasické papírové mapy.

3.3.1 Přijímač signálu

Pokud se hráč nespoléhá pouze na klasickou mapu, potřebuje zařízení schopné přijímat signál GNSS. Tímto zařízením může být automobilová či turistická

navigace, avšak naprostá většina hráčů používá svůj mobilní telefon, protože je ze všech navigačních zařízení nejflexibilnější. Umožňuje zobrazování schránek i pozice hráče na mapě. Uživatel v něm může mít uložené listingy keší, nebo si na něm může dohledat informace, které například potřebuje pro řešení mystery keší. Nespornou výhodou také je, že mobilní telefon dnes nosí každý neustále při sobě. Naprostá většina současných mobilních telefonů je schopna přijímat GPS signál. Většina z těch, které byly vyrobeny po roce 2013, je schopna pracovat i se signály GLONASS.

3.3.2 Webové stránky

Nejobsáhlejší webovou stránkou pro Geocaching je Geocaching.com [14], která sdružuje miliony hráčů z více než dvou set zemí světa. Všem uživatelům je umožněno přistupovat k souřadnicím a popisům většiny keší. Za poplatek jsou zpřístupněny další schránky, lepší vyhledávací nástroje, možnost upozornění na nově vzniklé keše... Prostřednictvím tohoto serveru jsou také schvalovány nové schránky.

3.3.3 Mobilní aplikace

Pokud hráč používá mobilní telefon, může využít jednu z mnoha aplikací, které jsou k dispozici pro Android, iOS i Windows Phone. Nejpoužívanější je bezplatná aplikace *c:geo* [21]. Druhou nejpoužívanější je oficiální placená mobilní aplikace *Geocaching* [22]. S těmito aplikacemi je možné např. najít nejbližší keše vzhledem k aktuální poloze, zobrazit je na mapě nebo uložit informace o schránkách, aby bylo možné hrát bez připojení k internetu.

3.3.4 Aplikace pro osobní počítače

Většina uživatelů si vystačí s webovým rozhraním serveru Geocaching.com nebo s mobilními aplikacemi. Desktopové aplikace nejsou tolik využívány, protože při hledání keší s sebou hráči obvykle nenosí počítač. Přesto existuje několik aplikací pro osobní počítače. Zaměřují se především na práci s GPX soubory (popsané v sekci 3.3.5), přenos stažených dat do navigačních zařízení, zobrazování keší na mapách nebo na udržování offline databáze keší. V desktopových aplikacích mají obvykle i neplatící hráči k dispozici některé funkce, které jsou běžně dostupné pouze platícím uživatelům. Příkladem může být pokročilé vyhledávání a filtrování schránek. Za zmínku stojí například program *MyGC* [23], sloužící k udržování offline databáze keší, nebo komplexní program *GeoGet* [24], který je podrobněji rozebrán v kapitole 3.4.

3.3.5 Formát GPX

Zkratka GPX vychází ze slovního spojení GPS exchange format a označuje soubor, který obsahuje GPS data.

Tento soubor může obsahovat jednotlivé body nebo celé trasy. Položkám v GPX souboru jsou přiřazovány atributy. Příkladem atributu bodu může být jeho název nebo zeměpisné souřadnice.

GPX je XML datový formát, tedy formát, který je možno rozšířit o vlastní značky. Pokud GPX soubor slouží hráčům Geocachingu k ukládání keší, je možné jednotlivým bodům přidat i konkrétní atributy vztahující se ke Geocachingu, například obtížnost či terén keše.

3.4 GeoGet

GeoGet je bezplatná aplikace určená pro hráče Geocachingu. Slouží ke správě vlastní databáze keší. Data v této databázi lze efektivně vyhledávat, filtrovat, zobrazovat na mapách či exportovat v různých formátech. Při popisu této aplikace je čerpáno z její dokumentace [25].

Primárně je GeoGet určen pro osobní počítače s operačním systém Windows. Vznikl roku 2006 jako jednoduchá jednoúčelová aplikace. Je vysoce modulární, takže jeho funkčnost lze snadno rozšiřovat pomocí doplňků (pluginů) a dalších pomocných skriptů. Díky tomu se rychle vyvinul v komplexní nástroj obsahující mnoho funkcí.

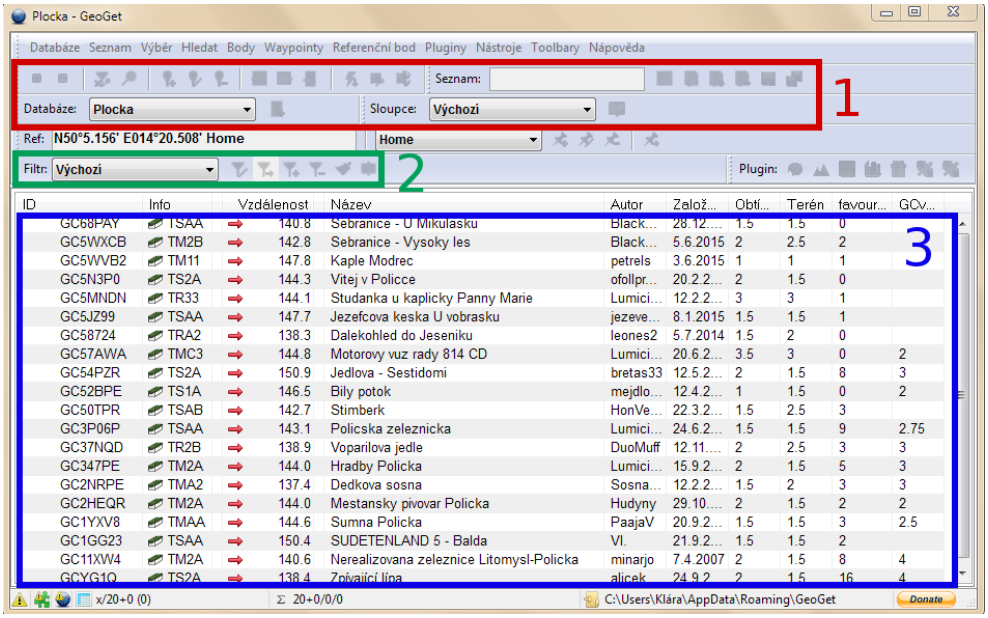
GeoGet je napsán ve vývojovém prostředí Delphi, které je určeno pro tvorbu aplikací v jazyce Object Pascal.

3.4.1 Funkce aplikace GeoGet

GeoGet je rozsáhlý nástroj pro Geocaching a zahrnuje mnoho funkcí. Mezi ty základní patří:

- import dat ze serveru Geocaching.com nebo z GPX souborů
- vyhledávání dat v databázi
- filtrování dat
- export dat v různých formátech
- zobrazení keší na mapě
- uložení keší pro použití bez internetového připojení
- možnost rozšiřování pomocí skriptů

Oproti funkcím webového serveru Geocaching.com umožňuje GeoGet lépe filtrovat keše a vyhledávat je. Nespornou výhodou je, že dokáže pracovat bez připojení k internetu (pokud má uživatel předem stažené keše).



Obrázek 3.1: Hlavní okno aplikace GeoGet

3.4.2 Základy práce s aplikací GeoGet

Základním pojmem je bod, který má (kromě dalších atributů) určeny zeměpisné souřadnice a unikátní identifikátor. Bodem může být například keš. Každý bod může mít přidruženy pomocné body, tzv. waypointy, které s ním určitým způsobem souvisejí. Waypointem mohou být například souřadnice parkoviště v blízkosti schránky, souřadnice zajímavého místa, na které keš odkazuje apod.

Body a waypointy se ukládají do databáze. Databází je možno mít více a přepínat mezi nimi. Zobrazený výběr bodů z databáze, nad kterým je možné vykonávat další operace, se nazývá seznam. Ten vznikne například jako výsledek vyhledávání nebo filtrování dat (filtrování je popsáno v sekci 3.4.3) a může být uložen pro pozdější použití.

Základní vzhled hlavního okna aplikace je znázorněn na obr. 3.1, kde jsou vyznačeny prvky zmíněné v předchozím odstavci:

1. výběr seznamu a databáze
2. filtr sloužící k výběru podmnožiny bodů z databáze
3. seznam bodů

3.4.3 Filtrování dat

Vybrat pouze určitou podmnožinu bodů z databáze umožňují filtry. Ty také ovlivňují seznam zobrazených bodů. Tento seznam je následně využíván mnohými skripty.

Filtry lze nastavit například podle typu, velikosti, terénu a obtížnosti keše. Lze vyfiltrovat pouze schránky na určitém území, nebo v blízkosti zvolené trasy či bodu. Je možné určit, zda mají být v seznamu uvedeny waypointy.

3.4.4 Doplňky

Aplikace GeoGet poskytuje rozhraní, pomocí kterého může kdokoliv rozšířit její funkce přidáním doplňků. Ty jsou psány formou skriptů. GeoGet obsahuje skriptovací engine Pascal Script [26], který umožňuje interpretovat doplňky za běhu programu, takže nemusí být kompilovány. To dovoluje uživateli měnit GeoGet, aniž by ho musel znovu kompilovat. Pascal Script umožňuje použití většiny konstrukcí Object Pascalu. Jazykem, ve kterém jsou doplňky psány, je tedy omezený Object Pascal.

Existuje několik druhů skriptů, z nichž jsou zmíněny čtyři nejdůležitější. Download skripty slouží ke stahování dat. Vizualizační skripty dovolují upravovat zobrazení výpisu keší v seznamu v hlavním okně aplikace, například přidávají obrázky či upravují zobrazené texty. Exportní skripty slouží k definování vlastních exportních formátů.

Nejobsáhlejší skupinou jsou programové skripty, neboli také pluginy, které rozšiřují funkce aplikace GeoGet a mohou i volat externí programy. Mezi tyto pluginy patří například doplňky, které:

- generují poutavé statistiky uživatele
- umožňují stahovat hodnocení a hodnotit keše přes CGVote
- zjišťují nadmořskou výšku keší
- vylepšují filtrování keší
- pomáhají řešit mystery keše ...

3.5 Současné možnosti plánování tras

V současné době neexistuje aplikace, která by navrhovala výlety spojené se hrou Geocaching.

Nejpodobnějšími aplikacemi jsou ty, které navrhují trasy s turistickými zajímavostmi. Tyto aplikace hledají trasy dané délky s co možná nejvíce turistickými zajímavostmi. Nelze je však využít k tvorbě výletů spojených se hrou Geocaching a uživatel nemá moc možností, jak upravit vyhledávání dle svých preferencí. Příkladem tohoto typu aplikací je plánovač tras na webových

stránkách Mapy.cz [27]. Uživatel může nastavit pouze výchozí bod trasy a její délku. Následně mu je nabídnuta jediná trasa.

Dále existují aplikace, které umí vyhledat všechny keše do určité vzdálenosti od zvolené trasy. Pokud hráč ví, kudy povede jeho výlet, není pro něj složité ručně nalézt keše v okolí. Obvykle však hráč trasu výletu danou nemá a hledá, kudy by se měl vydat. Toto řešení je vhodné například pro automobilisty, kteří cestují daleko a pouze se chtějí zastavit pro několik keší.

Poslední skupinu aplikací tvoří ty, které umí nalézt mezi zvolenými kešemi nejkratší trasu. Toto řešení však vyžaduje, aby hráč již věděl, které keše chce najít.

Z důvodu neexistence vhodné aplikace dnes většina hráčů plánuje své výlety ručně tak, že si zobrazí na mapě keše v okolí a na základě svých zkušeností si zvolí keše, které chtějí najít, a najdou mezi nimi sami nějakou trasu. Tento způsob plánování je časově náročný a často hráč nezvolí optimální trasu, nebo neodhadne správně časovou náročnost zvolené trasy.

Kapitola 4

Implementace

Cílem implementační části této práce bylo navrhnout a implementovat modul do aplikace GeoGet [24], který podporuje tvorbu výletů spojených se hrou Geocaching. Vytvořený modul byl pojmenován *TripMaker*.

Modul na základě požadavků uživatele navrhne trasu výletu spojeného s hledáním schránek. Hráči je nabídnuto více tras, aby se mohl rozhodnout, která mu vyhovuje nejvíce. Navrhované výlety jsou zobrazovány na mapě.

Při hledání vhodných tras TripMaker spolupracuje s aplikací GeoGet, ze které získává informace o keších. Modul dále používá knihovnu Qt [28], pomocí které je vytvořeno grafické rozhraní. Trasy jsou zobrazovány na mapách OpenStreetMap [29], práci s nimi usnadňuje použitá knihovna Marble [30].

V této kapitole jsou popsány výše zmíněné knihovny. Následně je rozebrána samotná implementace doplňku TripMaker. Uživatelská příručka je uvedena v příloze B.

4.1 GeoGet

GeoGet je software pro Geocaching a byl popsán v kapitole 3.4. Tento software je modulární a lze jej rozšiřovat například doplňky popsanými v kapitole 3.4.4. TripMaker patří do kategorie programových skriptů.

4.2 Použité nástroje

4.2.1 OpenStreetMap

OpenStreetMap (OSM) je volně dostupná mapa celého světa. Mapu je možné prohlížet na internetu nebo může být stažena jako XML soubor. Ten obsahuje základní prvky potřebné k vykreslení mapy. Mezi základní prvky patří uzly, představující body v prostoru se zadanými souřadnicemi a jednoznačným identifikátorem, a cesty, které jsou tvořeny posloupností uzlů. Dalším základním prvkem jsou relace. Relace je uspořádaný seznam uzlů nebo cest

popisující jejich geografickou či logickou závislost. Typickou relací může být multipolygon, popisující komplexní plochy (například budovy) nebo hranice (států či pozemků) [29].

Pokud jsou vynechány relace, tak OSM je graf, který je možné prohledávat klasickými algoritmy.

4.2.2 Marble

Marble je knihovna funkcí [31] napsaná v programovacím jazyce C++ usnadňující práci s mapami. Tato knihovna je závislá na knihovně Qt (která je popsána v sekci 4.2.3).

Základním prvkem je MarbleWidget, ovládací prvek vycházející z Qt widgetu, jenž zobrazuje mapu. MarbleModel je třída ukládající data, která jsou zobrazena v MarbleWidgetu. Jsou v ní uloženy například mapové podklady, hranice států nebo tzv. placemarks, význačné body. V této práci jsou keše na mapě reprezentovány právě pomocí placemarks.

Marble umožňuje zobrazovat mapové podklady, které získává v reálném čase z internetu, nebo může zobrazovat stažené mapové podklady ve formátu OSM XML.

Tato knihovna je velmi rozsáhlá, obsahuje přes 1000 tříd, tudíž není jednoduché se v ní zorientovat. Na druhou stranu umožňuje pohodlnou práci s mapami, proto byla vybrána jako vhodná pro tuto práci.

4.2.3 Qt

Qt je knihovna pro vytváření programů s grafickým uživatelským rozhraním. Tato knihovna byla zvolena díky přehledně zpracované dokumentaci [32] a také proto, že je vyžadována knihovnou Marble.

Zajímavým prvkem knihovny Qt jsou signály a sloty sloužící ke komunikaci mezi objekty. Signál je vyslán objektem, pokud došlo k určité události (bylo kliknuto na tlačítko, dokončil se výpočet atp.). Slot je metoda, která je volána v reakci na určitý vyslaný signál. V programu musí být definováno, které sloty reagují na jaké signály.

Signál může, kromě svého identifikátoru, obsahovat také parametry. Ty jsou předány slotu jako normální argumenty funkce. Například pokud je kliknuto na mapu, mohou být předávaným parametrem souřadnice místa, na které bylo kliknuto.

4.3 TripMaker

TripMaker je doplněk aplikace GeoGet, který navrhuje výlety spojené se hrou Geocaching.

Při jeho vývoji bylo dbáno na modularitu. Použité algoritmy, sloužící k výběru keší a hledání nejkratší cesty mezi nimi, lze jednoduše nahradit jinými algoritmy.

Dále bylo dbáno na jednoduchost ovládání a praktickou využitelnost.

Jazykem, ve kterém jsou psány doplňky aplikace GeoGet, je omezený Object Pascal. Protože tento jazyk není vhodný pro práci s mapami, je v něm napsána pouze menší část doplňku TripMaker. Ta zajišťuje spolupráci s aplikací GeoGet, především slouží ke spouštění doplňku z hlavního okna aplikace a ke získávání informací o keších. Z této části je spouštěna hlavní část programu TripMaker, která je napsána v jazyce C++.

4.3.1 Vstupní data

Vstupními daty jsou keše zobrazené v seznamu v hlavním okně aplikace GeoGet. Filtrování keší nebylo v modulu implementováno, protože výběr keší je možný za použití filtrů popsaných v sekci 3.4.3. Bylo by zbytečné implementovat funkci, která již implementována v aplikaci je, navíc velmi dobře.

TripMaker je primárně navržen pro hledání tradičních keší. Mystery keše obvykle vyžadují domácí přípravu nebo několikahodinové přemýšlení. Časová náročnost multi keší je značná, často vystačí jedna keš na celý výlet. Proto je doporučeno používat tento doplněk pouze na hledání tradičních schránek.

4.3.2 Funkce

Vhodná trasa výletu je nalezena na základě zvolených parametrů. Těmito parametry jsou výchozí bod výletu, požadovaná doba trvání a rychlost pohybu hráče.

Dalším parametrem, kterým hráč může ovlivnit trasu výletu, je tzv. *hunger rate*. Hunger rate nabývá hodnot od jedné do sta a udává, nakolik hráč preferuje kvantitu nalezených keší před jejich kvalitou. Jestliže je hunger rate nastaven na hodnotu 100, navrhovaná trasa obsahuje maximální možný počet keší. Naopak, pokud je hunger rate nastaven na hodnotu 1, jsou do trasy vybrány ty nejlepší keše z okolí, i když mohou být vzdálenější a celkově jich může být méně.

Po vygenerování tras je uživateli jedna doporučena a ta je vykreslena na mapě. Na mapě je zobrazena trasa s kešemi, které jsou znázorněny jako body s názvy. Po kliknutí na keš jsou zobrazeny podrobnější informace o ní (obtížnost, terén, velikost, hodnocení GCVote, souřadnice a nadmořská výška). Zároveň je zobrazen popis výletu, kde je napsána jeho doba trvání, délka a počet keší na trase. Následně je vypsán seznam keší zahrnutých ve výletu, včetně unikátního identifikátoru keše. Tento identifikátor je zároveň hypertextovým odkazem odkazujícím na kompletní listing keše na serveru Geocaching.com.

TripMaker vygeneruje více tras, aby si hráč mohl vybrat výlet dle svých preferencí. Počet navrhovaných tras se pohybuje od jedné do sedmi a závisí

na rozložení keší v okolí a na zadaných parametrech. Alternativní trasy jsou zobrazeny po vybrání z rozbalovacího menu.

Zobrazenou trasu lze exportovat ve formátu GPX upraveném pro Geocaching. Kromě základních atributů jsou v něm navíc uvedeny informace o schránkách.

Také je možné uložit aktuální pohled na mapu s trasou, což je vhodné např. pro tisk trasy výletu.

Mezi další funkce patří možnost zobrazení výškového profilu mapy, kopírování souřadnic z mapy, měření vzdálenosti na mapě, vyhledávání trasy mezi dvěma zvolenými body...

Na mapě je možné se pohybovat pomocí myši nebo klávesnice.

TripMaker je možné používat i v offline režimu, zcela bez mapových podkladů. Délka cesty mezi kešemi je pak odhadována na základě jejich vzdálenosti vzdušnou čarou.

4.3.3 Použité algoritmy

Z hlediska teorie grafů jsou keše reprezentovány uzly a cesty mezi nimi tvoří hrany.

Výběr podmnožiny keší byl realizován hladovým výběrem, protože tato práce se zaměřuje na hladové algoritmy. Implementovány byly obě metody popsané v sekci 2.3.1. Jejich výsledky byly v obecném případě srovnatelné. Ve výsledné aplikaci byla použita první metoda, výběr k nejbližších sousedů.

Výběr podmnožiny je též ovlivněn hodnotou hunger rate, hodnocením keší GCVote a body oblíbenosti. Skutečná vzdálenost je násobena koeficienty stanovenými dle hodnocení keše a počtu bodů oblíbenosti. Rozptyl těchto koeficientů je dán hodnotou hunger rate, čím nižší hodnota, tím větší rozptyl. Pokud je hunger rate nastaven na 100, rozptyl je nulový, tudíž hodnocení nemá na nalezenou trasu vliv.

Výpočet nejkratší trasy byl zkusmo realizován pomocí exaktního algoritmu s využitím metody větví a hranic, která byla uvedena v sekci 2.1.3. Toto řešení bylo příliš pomalé, ačkoliv pro malé vstupy nacházelo optimální řešení.

Implementován byl také hladový a 2-aproximační algoritmus (popsané v sekcích 2.2.1 a 2.2.2). Po vylepšení získaného řešení 2-opt heuristikou nacházely oba algoritmy podobně dobrá řešení. Není možné jednoznačně určit, který z algoritmů je lepší, protože přibližně pro polovinu případů je lepší 2-aproximační algoritmus a pro druhou polovinu hladový algoritmus. Ve výsledné implementaci doplňku byl zvolen hladový algoritmus, protože vykazoval v průměrném případě o trochu lepší výsledky. Podrobné porovnání je uvedeno v sekci 5.2.

Aby byl doplněk použitelný v praxi, musí dodávat řešení v rozumném čase, řádově do několika desítek vteřin. Použitá 2-opt heuristika je však časově náročná (jak je popsáno v sekci 2.2.4), a proto byl stanoven časový limit,

který zaručuje, že doba běhu programu není nepřijatelně dlouhá, i za cenu, že nalezené řešení není nejlepší možné.

Odhad času potřebného na nalezení keše vychází ze zkušeností hráčů a závisí na obtížnosti a terénu schránky. Odhad je relativně nepřesný, protože každému hráči trvá odlov různou dobu. Blíže je tento problém popsán v sekci 5.1.1.

Kapitola 5

Testování

Tato kapitola popisuje způsob testování modulu TripMaker, interpretuje výsledky testování a srovnává je s konkurenční implementací modulu od Jana Staňka z roku 2016 [33].

Cílem testování bylo zjistit, nakolik doplněk odpovídá definovaným požadavkům. Hlavním požadavkem bylo nalezení nejvhodnější trasy dle zadaných kritérií. Míra naplnění tohoto požadavku nelze zcela objektivně změřit. Také neexistuje žádná podobná aplikace, která by umožňovala přesné srovnání a ověření správnosti výsledků.

Z těchto důvodů byl modul podroben sérii rozmanitých testů, které se zaměřovaly na jednotlivé oblasti řešení:

1. Porovnání implementovaných algoritmů řešících problém orientačního běžce (výběr množiny bodů a nalezení nejkratší cesty mezi nimi).
2. Srovnání délky nejkratší nalezené Hamiltonovské kružnice pro danou množinu bodů s délkou nalezenou jinou aplikací.
3. Vliv hodnoty hunger rate na výslednou trasu.
4. Srovnání s konkurenční implementací od Jana Staňka.
5. Porovnání výsledků, převážně odhadu časové náročnosti, s historickými daty.
6. Ověření správnosti řešení experimentem.
7. Míra zlepšení oproti ručnímu plánování.

V testech byly testovány vstupy reálné a extrémní. Reálné odpovídají předpokládaným požadavkům uživatelů. Předpokládá se, že průměrný hráč se pohybuje rychlostí 3 až 20 km/h a požaduje dobu trvání výletu od 30 minut do 8 hodin. Extrémní vstupy zahrnují větší nebo nepříznivé vstupy a testují výkonnost a kvalitu použitých algoritmů.

5.1 Kvalita nalezené trasy

Aby mohla být vybrána nejvhodnější trasa, bylo třeba stanovit metriky pro určení kvality nalezených tras. Kvalita závisí hlavně na počtu keší zahrnutých do výletu, jejich hodnocení a předpokládané časové náročnosti výletu. Ta by se neměla příliš lišit od požadované doby trvání.

Pro určení kvality trasy bylo zjištěno skóre. Čím vyšší má hodnotu, tím je trasa kvalitnější. Skóre S lze vyjádřit jako součin váženého počtu keší $vazenyPocet$ a časového faktoru $casovyFaktor$:

$$S = vazenyPocet \times casovyFaktor.$$

Časový faktor penalizuje výlety, jejichž doba trvání ($casVyletu$) se liší od požadované časové náročnosti ($pozadovanyCas$). Je dán vztahem

$$casovyFaktor = 1 - min(\frac{9}{10}, k \ \times |1 - \frac{casVyletu}{pozadovanyCas}|).$$

Pokud je časová odchylka do 20 % požadované délky trvání výletu, je považována za akceptovatelnou a hodnota k je rovna $\frac{1}{10}$. Pokud je časová odchylka větší než 20 %, hodnota k je 1.

Vážený počet keší vyjadřuje kvantitu a kvalitu keší zahrnutých do výletu. Závisí také na hodnotě hunger rate, pokud je rovna 100, nemá hodnocení schránek na kvalitu trasy vliv. Čím nižší hunger rate, tím větší vliv má hodnocení keší na kvalitu trasy.

$$vazenyPocet = \sum_{\forall keše} (1 + (GCVote - 1) \times (\frac{100 - hungerRate}{20}) \times f)$$

Hodnota proměnné f vychází z bodů oblíbenosti keše. Pokud počet bodů oblíbenosti patří mezi nejvyšších 20 % počtů v rámci zkoumané množiny schránek, je proměnná f rovna hodnotě $1,2$. Pokud naopak patří počet bodů oblíbenosti mezí nejnižších 20 %, je hodnota f rovna $0,8$.

5.1.1 Časová náročnost výletu

Kvalita nalezené trasy závisí na odhadu časové náročnosti výletu. Čím rychleji se hráč pohybuje, tím větší část času tráví hledáním schránek a tím více je důležitý správný odhad času potřebného k nalezení schránek.

Časová náročnost schránky nemůže být přímo určena objektivními parametry. Doba potřebná pro nalezení dvou keší se stejnými hodnotami obtížnosti i terénu se může značně lišit. Záleží například na důmyslnosti úkrytu nebo zkušenostech hráče. Obecně platí, že začátečníkům trvá nalezení schránky výrazně delší dobu než zkušeným hráčům. Záleží i na vnějších okolnostech, například pokud v blízkosti keše stojí cizí lidé, hráč musí počkat, než odejdou.

Nejhorší případ, z hlediska časového odhadu, nastává, pokud se hráči nepodaří keš nalézt. Někteří hráči mají stanovený časový limit a jestliže keš nenaleznou v tomto limitu, přesouvají se k další keši. Jiní hráči jsou schopni hledáním jedné keše strávit hodinu.

Z těchto důvodů nelze algoritmicky testovat správnost odhadu délky trvání výletu, a proto není při algoritmickém testování příliš zohledněn časový faktor. Správnost odhadu časové náročnosti výletu je ověřována praktickými testy v sekcích 5.6 a 5.7.

5.2 Srovnání implementovaných algoritmů

Cílem tohoto testu bylo porovnání implementovaných algoritmů, které řeší problém orientačního běžce. Jak bylo popsáno v kapitole 4.3.3, tyto algoritmy vyberou podmnožinu keší a nalezenou mezi nimi nejkratší trasu.

Srovnávány byly čtyři algoritmy kombinující možné způsoby nalezení podmnožiny keší a nejkratší cesty. Všechny tyto algoritmy byly vylepšeny 2-opt heuristikou:

1. Podmnožina k nejbližších keší vzhledem k počáteční pozici, trasa nalezena hladovým algoritmem. Výsledky jsou uvedeny v tabulce 5.1.

2. Podmnožina získána postupným výběrem, trasa nalezena hladovým algoritmem. Výsledky jsou uvedeny v tabulce 5.2.

3. Podmnožina k nejbližších keší vzhledem k počáteční pozici, trasa nalezena 2-aproximačním algoritmem. Výsledky jsou uvedeny v tabulce 5.3.

4. Podmnožina získána postupným výběrem, trasa nalezena 2-aproximačním algoritmem. Výsledky jsou uvedeny v tabulce 5.4.

Počátečním bodem výletu bylo místo se souřadnicemi 14.4402458E 50.0562744N, hunger rate byl nastaven na 100 a rychlost hráče na 12 km/h. Každý algoritmus byl otestován na sedmi instancích, které se lišily požadovanou dobou trvání výletu.

U každé instance jsou uvedeny parametry nalezené trasy (počet keší, délka výletu a jeho časová náročnost) a doba výpočtu. Protože je doba výpočtu ovlivněna momentálním zatížením počítače, bylo její měření uskutečněno třikrát a uveden je průměr těchto hodnot.

Srovnání délek nalezených tras je uvedeno na obrázku 5.1

Postupný výběr podmnožiny za běhu algoritmu byl pomalejší, než výběr k nejbližších keší, a nedosahoval lepších výsledků. Hladový algoritmus dosahoval lepších výsledků než 2-aproximační algoritmus.

Nalezená řešení jednotlivých algoritmů byla ohodnocena dle metriky uvedené v sekci 5.1. Vypočtné hodnoty skóre jsou uvedeny v tabulce 5.5. V průměrném případě nejlepších výsledků dosahoval algoritmus číslo jedna.

Tabulka 5.1: Naměřené hodnoty algoritmu řešícího problém orientačního běžce č. 1, podmnožina k nejbližších keší vzhledem k počáteční pozici, trasa nalezena hladovým algoritmem.

Požadovaný čas	Doba výpočtu (s)	Trvání (min)	Délka (km)	Počet keší
60	3,47	62	5,30	6
120	4,37	126	10,08	13
180	4,92	191	15,50	20
250	5,26	248	20,20	26
500	8,62	501	37,98	55
1000	15,96	1068	82,05	109
3000	35,46	3049	294,18	270

Tabulka 5.2: Naměřené hodnoty algoritmu řešícího problém orientačního běžce č. 2, podmnožina získána postupným výběrem, trasa nalezena hladovým algoritmem.

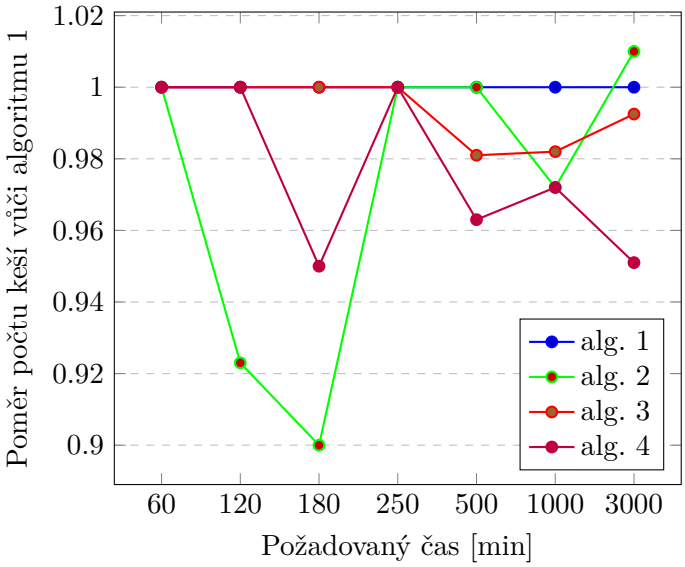
Požadovaný čas	Doba výpočtu (s)	Trvání (min)	Délka (km)	Počet keší
60	4,06	64	5,81	6
120	4,51	117	9,40	12
180	5,26	174	14,12	18
250	8,26	253	21,10	26
500	11,82	501	40,07	53
1000	20,64	1011	79,99	106
3000	35,10	3185	320,86	271

Tabulka 5.3: Naměřené hodnoty algoritmu řešícího problém orientačního běžce č. 3, podmnožina k nejbližších keší vzhledem k počáteční pozici, trasa nalezena 2-aproximačním algoritmem.

Požadovaný čas	Doba výpočtu (s)	Trvání (min)	Délka (km)	Počet keší
60	3,71	62	5,30	6
120	4,12	126	10,08	13
180	4,96	193	16,03	20
250	5,67	250	20,55	26
500	8,59	500	38,74	54
1000	16.42	1042	84,99	107
3000	31,98	2975	282,17	268

Tabulka 5.4: Naměřené hodnoty algoritmu řešícího problém orientačního běžce č. 4, podmnožina získána postupným výběrem, trasa nalezena 2-aproximačním algoritmem.

Požadovaný čas	Doba výpočtu (s)	Trvání (min)	Délka (km)	Počet keší
60	3,36	58	5,01	6
120	4,29	123	9,46	13
180	5,45	179	14,12	19
250	7,01	250	20,65	26
500	11,07	498	39,34	53
1000	25,26	1005	82,64	103
3000	36,94	2979	295,64	257



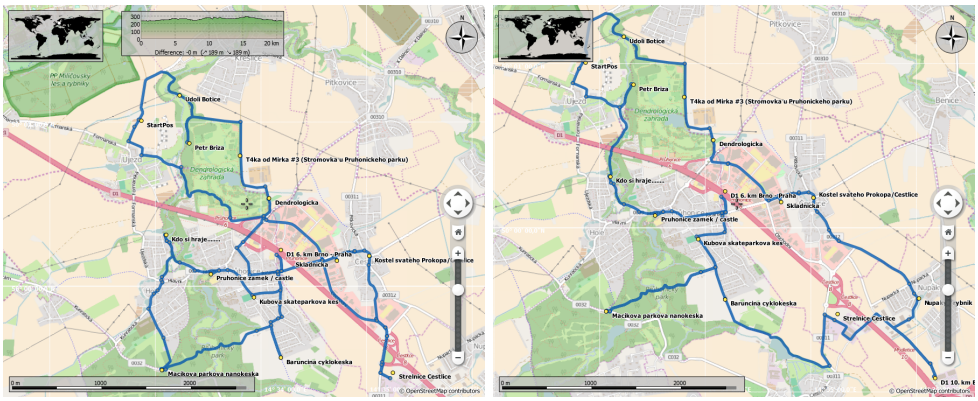
Obrázek 5.1: Poměr počtu nalezených keší algoritmů řešících problém orientačního běžce

Tabulka 5.5: Skóre tras získaných různými algoritmy dle stanovené metriky

Algoritmus\Požadovaný čas	60	120	180	250	500	1000	3000
Algoritmus 1	6.02	13.06	20.12	25.97	55.01	109.7	270.4
Algoritmus 2	6.04	11.97	17.94	26.03	53.01	106.1	272.6
Algoritmus 3	6.02	13.06	20.14	26.00	54.00	107.4	267.7
Algoritmus 4	5.98	13.03	18.98	26.00	52.97	103.0	256.8
Algoritmus 1 bez 2-opt	6.02	12.03	19.01	24.90	51.97	99.04	263.4

5.2.1 Vliv 2-opt heuristiky

2-opt heuristika značně zlepšuje získaná řešení, což je vidět např. na obrázku 5.2. Pro zadané vstupní parametry byla nejprve nalezena trasa bez použití 2-opt heuristiky. Délka této trasy je 20,56 km a zahrnuje 13 keší. Po přidání 2-opt heuristiky je délka trasy srovnatelná (20,92 km), ale obsahuje 15 keší.

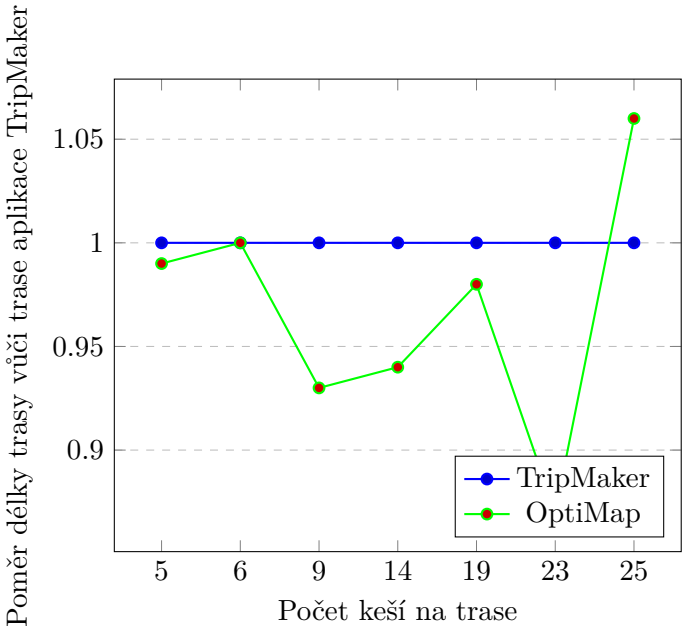


Obrázek 5.2: Bez použití 2-opt heuristiky (vlevo) je trasa horší než s použitím 2-opt heuristiky (vpravo)

Doba běhu programu bez použití 2-opt heuristiky je výrazně kratší, ale počet nalezených schránek je menší, jak ukazuje tabulka 5.6 vycházející z dat užitých pro srovnání algoritmů v předchozí sekci. Pro měření byl použit algoritmus jedna uvedený v předchozí sekci, ale nebyla použita 2-opt heuristika.

Tabulka 5.6: Naměřené hodnoty algoritmu řešícího problém orientačního běžce č. 1, podmnožina získána postupným výběrem, trasa nalezena hladovým algoritmem, bez použití 2-opt heuristiky

Požadovaný čas	Doba výpočtu (s)	Trvání (min)	Délka (km)	Počet keší
60	3,93	62	5,30	6
120	4,12	123	10,58	12
180	4,82	181	14,67	19
250	5,86	240	19,75	25
500	6,63	498	41,32	52
1000	9,66	1005	88,84	99
3000	18,19	2943	249,51	264



Obrázek 5.3: Poměr délek tras navržených doplňkem TripMaker a aplikací OptiMap

5.3 Délka Hamiltonovské kružnice

Cílem tohoto testu bylo zjistit, nakolik se nalezená trasa procházející danou množinou keší liší od nejkratší možné trasy. Ověřována byla výkonnost algoritmů řešících problém obchodního cestujícího.

Většina programů pro řešení problému obchodního cestujícího pracuje s grafy zadanými množinou uzlů a ohodnocených hran. Vstupem modulu TripMaker jsou však souřadnice keší, cesty nejsou explicitně dány, musí být nalezeny v mapě. Proto byla pro porovnání zvolena online aplikace pro řešení TSP OptiMap [34].

Vstupem této aplikace jsou GPS body a výstupem je nejkratší nalezená kružnice mezi nimi. Nevýhodou aplikace OptiMap je, že při hledání cest neuvažuje menší cesty, například lesní pěšiny, i když je zvoleno, že trasa má být pro pěší. Z tohoto důvodu nacházel TripMaker kratší trasy, zvláště mimo města. Proto byly pro porovnání vybrány trasy, na kterých TripMaker nevyužíval menších cest a pěšin.

OptiMap dosahoval o něco lepších výsledků než TripMaker. Na druhou stranu doba potřebná k výpočtu byla mnohonásobně delší, pro 25 keší řádově několik minut. Pro více než 40 keší nebylo výsledku v rozumném čase dosaženo.

Naměřené hodnoty jsou uvedeny v tabulce 5.7. Relativní porovnání délek naměřených tras je uvedeno na obrázku 5.3.

Tabulka 5.7: Délky tras (v km) nalezených aplikacemi OptiMap a TripMaker

Aplikace	5 keší	6 keší	9 keší	14 keší	19 keší	23 keší	25 keší
Optimap	5,07	12,42	10,90	6,85	18,82	14,01	265
TripMaker	5,08	12,42	11,72	7,28	19,16	15,96	249

Tabulka 5.8: Vliv hunger rate na nalezenou trasu

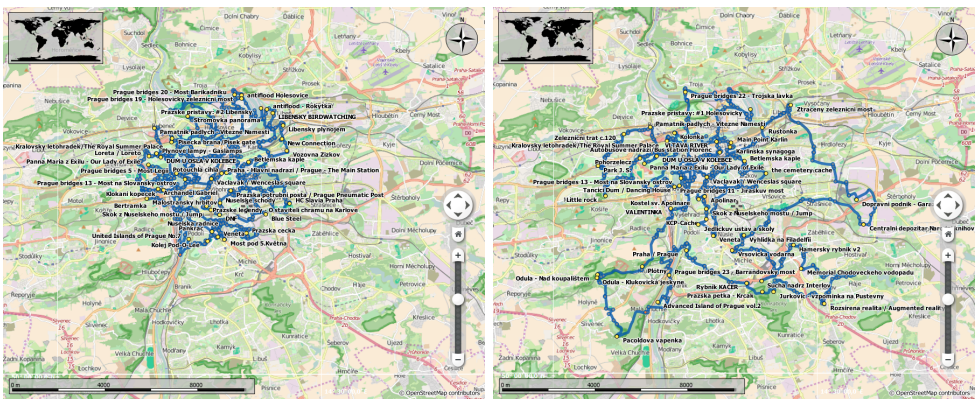
Hunger rate	Počet keší	Trvání (min)	Délka (km)	Průměr GCVote
1	4	380	24,3	5,00
25	12	367	21,1	3,66
50	20	382	20,3	3,24
75	24	401	19,0	3,06
100	29	389	17,5	2,95

5.4 Vliv hunger rate

Cílem tohoto testu bylo ověření, že hodnota parametru hunger rate (HR) ovlivňuje trasu a průměrnou kvalitu vybraných keší. Pro otestování byl zvolen počáteční bod se souřadnicemi 14.4402458E 50.0562744N, rychlost hráče 4,5 km/h a požadovaná doba trvání výletu 400 minut. Následně bylo provedeno měření pěti instancí, které se lišily hodnotou HR.

Instance se zadanou nižší hodnotou HR nacházely delší trasy obsahující méně schránek. Průměrné hodnocení GCVote těchto keší však bylo vyšší než v ostatních instancích. Porovnání instancí je uvedeno v tabulce 5.8.

Testy ukázaly, že hodnota parametru hunger rate ovlivňuje nalezená řešení dle očekávání. Efekt opačných hodnot HR je zobrazen na obrázku 5.4. V případě, že byl HR nastaven na hodnotu 100, počet nalezených keší na trase byl 168. Pokud byl HR nastaven na 1, keše byly od sebe vzdálenější a bylo jich pouze 113, avšak měly vyšší průměrné hodnocení.



Obrázek 5.4: Rozdílné trasy pro hodnoty hunger rate 100 (vlevo) a 1 (vpravo)

5.5 Porovnání s konkurenční implementací

V této sekci byl porovnán TripMaker (TM) s doplňkem GeoTrip (GT), který vytvořil Jan Staněk v rámci své bakalářské práce v roce 2016 [33]. GeoTrip má stejný primární účel jako TripMaker, tedy slouží k nacházení vhodných tras pro Geocaching.

Z uživatelského hlediska se GeoTrip liší především v následujících bodech:

- Je nutné mít předem stažené mapy.
- Je možné filtrovat keše přímo v doplňku.
- Na mapě jsou neustále zobrazeny všechny keše vyhovující filtrům. Výhodou tohoto řešení je, že hráč má přehled o rozložení schránek v oblasti. Na druhou stranu je toto řešení někdy nevyhovující, protože se zdá, že trasa prochází keší, ale ta ve skutečnosti není do plánované trasy zahrnuta. Doplněk neříká, které schránky patří do nalezené trasy, je znám pouze jejich počet.
- Nemožnost za běhu měnit výchozí bod výletu, nebo jakkoliv ovlivnit uvažovanou rychlost pohybu
- Možnost volby profilu trasy - chůze, kolo či auto.
- Délka výletu je ovlivněna zadanou maximální délkou výletu, nikoliv dobou trvání.
- Nemožnost výběru alternativních tras.

Porovnání modulů nebylo jednoduché, protože každý modul počítá s jinou rychlostí hráče, jiným časem potřebným k odlovu keše, GeoTrip nachází trasu dle zadané maximální délky, TripMaker dle požadované doby trvání výletu...

Z těchto důvodů byl porovnávací testování provedeno následujícími třemi způsoby:

1. Délka nalezené trasy obsahující právě danou množinu keší.
2. Testy zahrnující výběr podmnožiny schránek.
3. Testování na reálných datech.

Aby byly podmínky co možná nejvyrovnanější, byla zvolena trasa pro pěší a cílem bylo získat co nejvíce schránek, bez ohledu na jejich kvalitu.

Testování na reálných datech proběhlo na historických datech a provedením praktického experimentu. Výsledky těchto metod jsou popsány v kapitolách 5.6 a 5.7.

5.5.1 Délka nalezené trasy

V těchto testech byla dána množina keší a cílem bylo porovnat délky nalezených tras obsahujících všechny tyto keše. Aby byly do trasy zahrnuty všechny schránky z množiny, byla u modulu GeoTrip nastavena dostatečně velká maximální délka a u modulu TripMaker dostatečně dlouhý požadovaný čas výletu.

Testované množiny schránek byly schválně různorodé, některé se nacházely ve městech, jiné v hlubokých lesích, některé byly rozmístěny na malé ploše, jiné na velké.

Nebylo možné otestovat množinu rozmístěnou na extrémně velké ploše (např. keše v různých státech), protože u modulu GeoTrip nefungoval import a předzpracování mapy, bez kterého ho není možné používat. Proto bylo testování provedeno pouze na území České republiky, jejíž již předzpracovaná mapa byla dodána autorem.

Hlavní testovanou položkou byla délka trasy. Uvedený čas výpočtu trasy je spíše orientační, protože byl ovlivněn momentálním zatížením počítače. Čas výpočtu je vypovídající hlavně u velkých instancí. Výsledky měření jsou uvedeny v tabulce 5.9. Porovnání délek tras je uvedeno na obrázku 5.5.

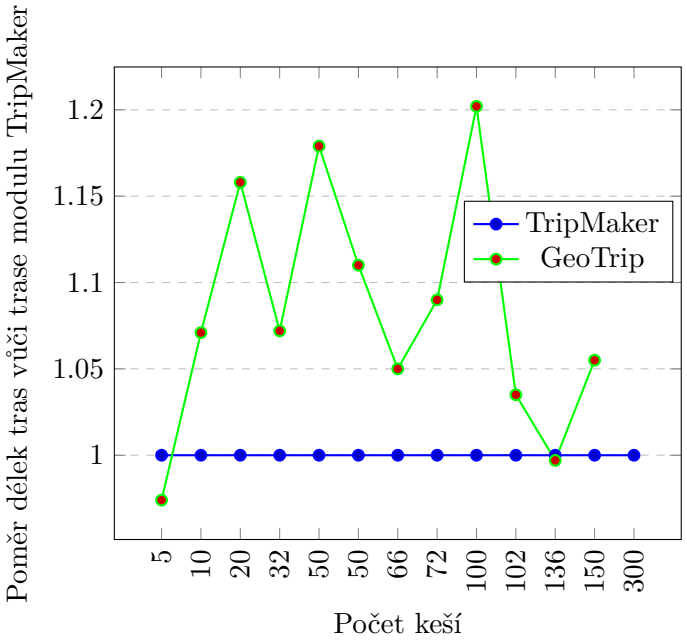
Z výsledků testování plyne, že GeoTrip řeší (pro malé množiny) danou úlohu obvykle rychleji než TripMaker (o několik vteřin). Naopak TripMaker nachází téměř vždy výrazně kratší trasu. TM je schopen řešit i větší instance, které GT není schopen vyřešit v rozumném čase.

Tabulka 5.9: Porovnání tras (délek a času potřebného k nalezení) nalezených aplikacemi GeoTrip (GT) a TripMaker (TM) nad danou množinou keší

Počet keší	GT délka (km)	TM délka (km)	GT čas výpočtu (s)	TM čas výpočtu (s)
5	7,6	7,8	4	6
10	7,7	7,1	10	8
20	394,5	340,4	6	7
32	89,3	83,2	16	13
50	402,1	340,9	10	11
50	434,1	390,7	14	16
66	46,0	43,8	8	9
72	227,7	207,3	6	9
100	1303	1084	13	15
102	352,1	340,2	13	11
136	547,5	549,0	49	14
150	112,1	106,2	125	16
300	-	272,0	více než 2000	17

5.5.2 Výběr podmnožiny schránek

Protože GeoTrip hledá výlety na základě maximální možné délky a TripMaker na základě požadované doby trvání výletu, je obtížné najít pro oba doplňky vhodné vstupní parametry, které by umožňovaly poměřit jejich výkon. Bylo



Obrázek 5.5: Poměr délek tras navržených doplňky TripMaker a GeoTrip

vyzkoušeno zadat určitou maximální délku modulu GeoTrip a jeho časový odhad použít jako vstupní data TM. Při tomto testování bylo zjištěno, že doplňky různě počítají dobu potřebnou k nalezení keší. Pro stejné trasy se odhadovaná dobu trvání výrazně liší. GT odhaduje dobu trvání delší než TM.

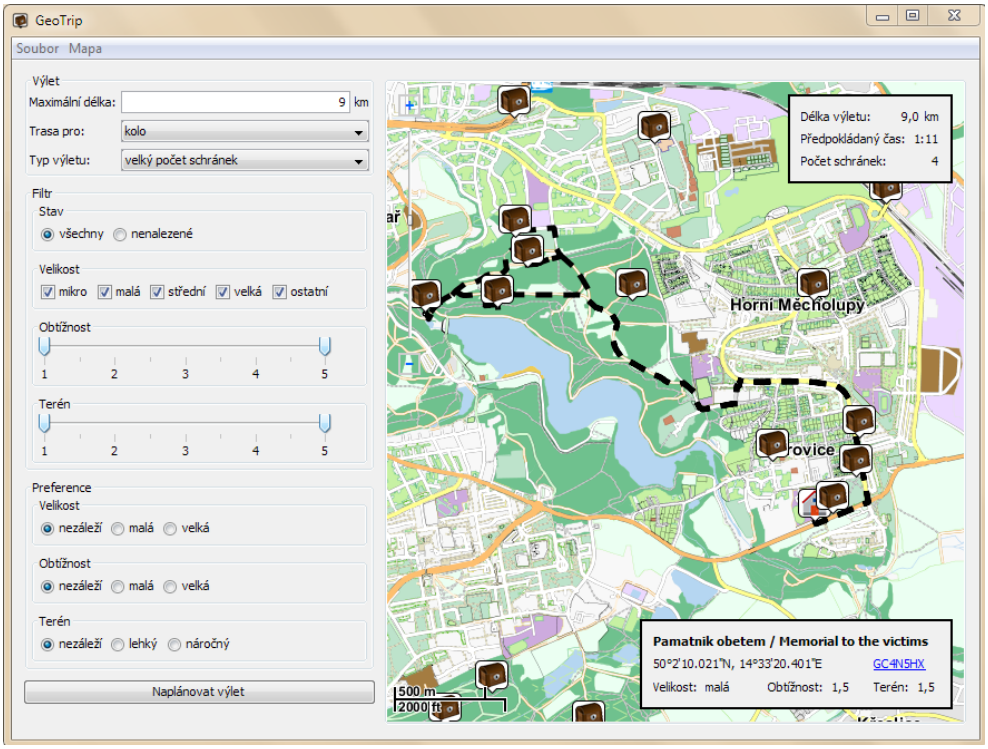
GeoTrip také bere zadanou délku pouze jako maximální, tudíž není výjimkou, že nalezená trasa zdaleka není tak dlouhá, jako maximální zadaná délka. GT nelze hodnotit metrikou stanovenou v kapitole 5.1, protože není možné spočítat odchylku od požadované délky výletu.

Při testech bylo zjištěno, že TripMaker nenachází nejvhodnější podmnožiny, pokud má abnormální nedostatek cest (například keš je na druhé straně řeky, přes kterou nevedou žádné mosty).

GeoTrip někdy vybírá velmi nevhodnou množinu keší. Jedna z těchto situací je znázorněna na obrázku 5.6. Nalezená trasa obsahuje čtyři schránky. Přestože se hned u výchozího bodu nachází čtyři schránky, byly vybrány čtyři výrazně vzdálenější. A to bylo nastaveno, že výlet má obsahovat co největší množství keší.

5.6 Úspěšnost na historických datech

Úspěšnost zejména časového odhadu náročnosti trasy, který se prakticky nedá ověřit měřením na počítači, byla testována na známých historických datech. Vybrány byly dva výlety, které svou délkou a zahrnutými kešemi odpovídají



Obrázek 5.6: Nevhodný výběr podmnožiny doplňkem GeoTrip

běžným Geocachingovým trasám. Jeden výlet byl uskutečněn na kole, druhý pěšky.

V tomto testu byla doplňkům dána množina schránek, které byly zahrnuty do uskutečněného výletu. Cílem doplňků bylo nalézt mezi nimi nejkratší trasu a odhadnout časovou náročnost.

5.6.1 Trasa A

První hodnocený výlet se uskutečnil v Kolovratech, počátečním bodem byla vlaková stanice Praha-Kolovraty. Dopravním prostředkem bylo jízdní kolo. Průměrná rychlost pohybu byla získána z tachometru. Relativně nízká rychlost je způsobena tím, že jsou keše blízko u sebe a mnoho času hráč stráví pomalým popojížděním a hledáním cesty.

Informace o trase A:

- Počáteční souřadnice: 50.0120911N, 14.6261750E
- Rychlost pohybu: 11,2 km/h
- Skutečná časová náročnost: 168 minut
- Délka: 9,8 km

- Počet keší na trase: 16

Keše včetně času potřebného k jejich nalezení, jejich obtížnosti a terénu jsou uvedeny v tabulce 5.10.

Tripmaker nalezl stejnou trasu, která byla zvolena hráčem a odhadl její časovou náročnost na 147 minut. GeoTrip nalezl o kilometr delší trasu než TM a odhadl časovou náročnost na 210 minut. TM dobu trvání výletu podhodnotil, GT naopak nadhodnotil.

Tabulka 5.10: Keše a čas potřebný k jejich nalezení na trase A

Název keše	ID	Obtížnost	Terén	Čas (min)
Motorová lokomotiva "Bardotka"	GC5W6YZ	2,5	1,5	3
Zoo kes - Pagekon rasnaty	GC66DY6	2	2	3
Tehovicka krcma	GC6B91M	1,5	1,5	3
Krizek	GC1WZA5	2	1,5	4
Prknovka	GC1PDE9	2	1,5	9
Exkrement Transformers	GC6AW84	2	1,5	4
Slachty	GC6B921	2	2	4
Brod	GC6AW75	2	2	6
Tehovicky	GC67A2F	2,5	2	10
Bývalý zimní stadion	GC1WC3V	2	2	18
Vychazkove okruhy v Kolovratech	GC1KWM1	1,5	2	6
Na Cisarske	GC66TDX	2	2	5
Kolovraty	GC67A30	2,5	1,5	5
Mladota	GC693VD	1,5	2	2
Sadovnictvi	GC6A704	1,5	2	2
Donat	GC5YPVV	2	1,5	12

5.6.2 Trasa B

Druhá zkušební trasa byla v Praze a výchozím bodem byla tramvajová zastávka Národní třída se souřadnicemi 50.0819783N, 14.4193919E. Tento výlet byl realizován pěšky. Průměrná rychlost pohybu byla získána z GPS záznamu. Informace o trase B:

- Počáteční souřadnice: 50.0819783N, 14.4193919E
- Rychlost pohybu: 4,5 km/h
- Skutečná časová náročnost: 197 minut
- Délka: 10,8 km
- Počet keší na trase: 10

Schránky včetně času potřebného k jejich nalezení, jejich obtížnosti a terénu jsou uvedeny v tabulce 5.11.

GeoTrip nalezl o 300 metrů kratší trasu než TripMaker a odhadl časovou náročnost na 215 minut. TripMaker odhadl, že na uskutečnění výletu stačí 202 minut, čímž se více přiblížil reálné hodnotě.

Tabulka 5.11: Keše a čas potřebný k jejich nalezení na trase B

Název keše	ID	Obtížnost	Terén	Čas (min)
Prazske legendy - O zkamenelem...	GC1GQ2T	2	1	4
Sametova revoluce	GC4Q15N	2	1,5	6
Rotunda sv. Krize Mensiho	GC1B418	1	1	2
United oslands of Prague No. 1	GCJC2D	2	1	3
Travnik	GC63KCA	1	1,5	15
Ruzovy tank	GCZX78	2	1,5	2
Spirala: Logaritmicka	GC6NA1V	3	3	9
Spirala: Archimedova	GC5N9ZX	1,5	2	5
Panna Maria z Exilu	GC47BHR	2	2,5	3
Pohorzelecz	GC19KD4	2,5	1	17

5.7 Praktický experiment

Tento test byl opakem testu na historických datech. Byly navrženy dvě cyklistické trasy, trasa C modulem TripMaker a trasa D modulem GeoTrip. Nad množinou bodů z obou tras byla zjištěna nejkratší cesta a odhad časové náročnosti oběma doplňky. Zde vznikl jeden problém, trasa C obsahovala schránku, kterou GeoTrip za žádných podmínek nezahrnul do výsledné trasy. Z tohoto důvodu musela být trasa C pozměněna.

Naplánované výlety byly následně uskutečněny. Cílem tohoto testu bylo ověřit uskutečnitelnost navrhovaných výletů v praxi a zkontrolovat správnost časového odhadu.

Trasy výletů jsou znázorněny na obrázcích 5.7 a 5.8. Keše zahrnuté do výletů jsou uvedeny v tabulkách 5.13 a 5.14.

Z odhadovaných časů a reálné doby trvání výletů uvedených v tabulce 5.12 je zřejmé, že odhad obou modulů byl relativně dobrý. TripMaker v obou případech odhadl kratší dobu trvání, naopak GeoTrip v obou případech odhadl delší dobu trvání.

Tabulka 5.12: Odhadovaná a reálná časová náročnost tras C a D (v minutách)

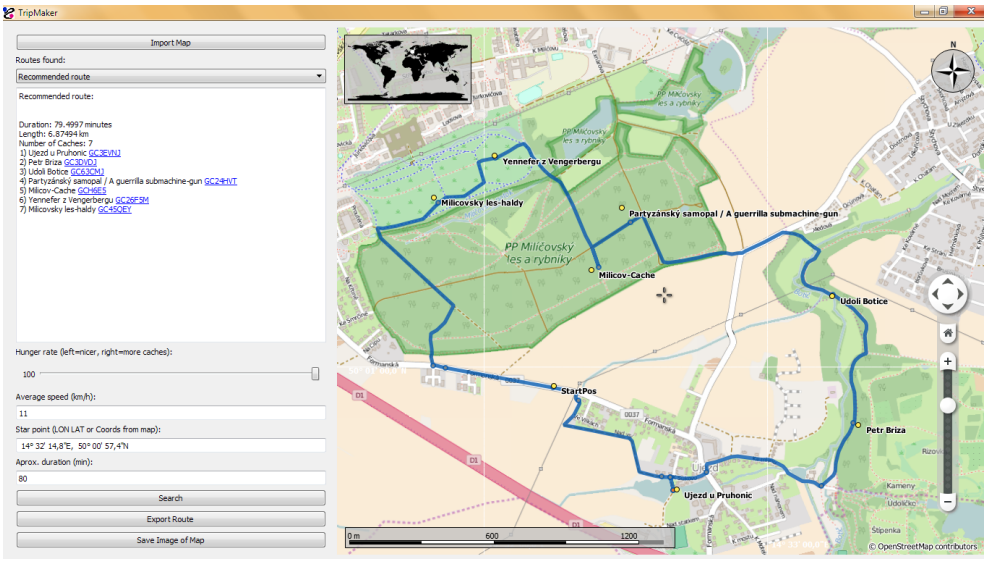
Trasa	Odhad Tripmaker	Odhad GeoTrip	Reálná časová náročnost
Trasa C	79	113	85
Trasa D	58	75	65

Tabulka 5.13: Keše a čas potřebný k jejich nalezení na trase C

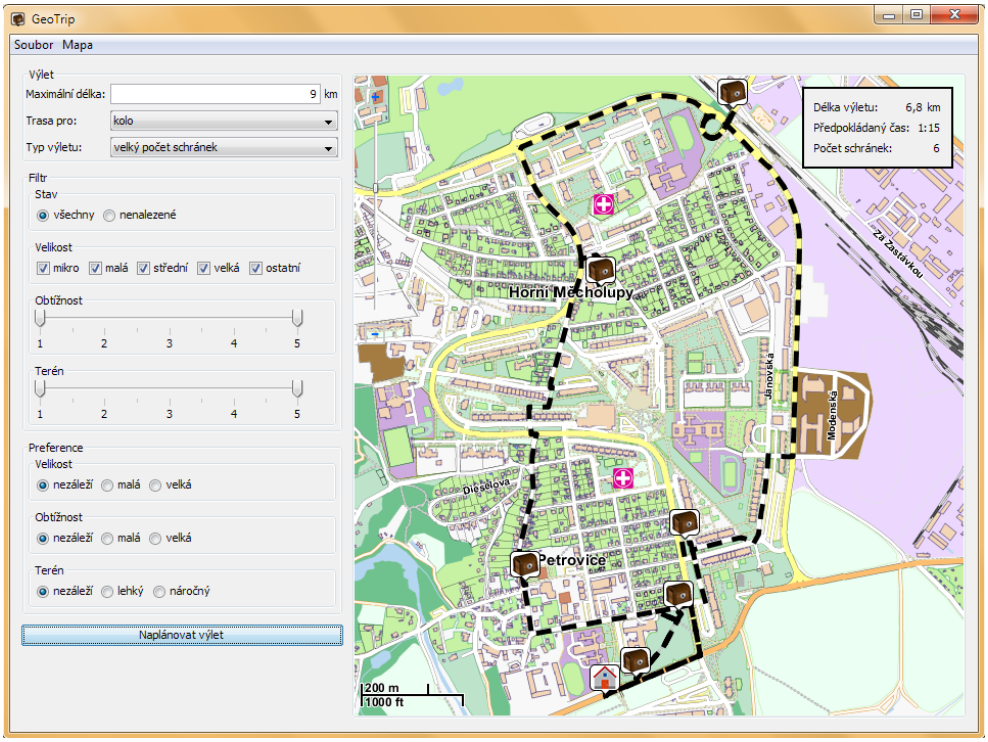
Název keše	ID	Obtížnost	Terén	Čas (min)
Ujezd u Pruhonic	GC3EVNJ	1,5	2	5
Petr Briza	GC3DVDJ	2	3	6
Udoli Botice	GC63CMJ	1,5	2,5	6
Partyzánský samopal	GC24HVT	1	1,5	2
Milicov-Cache	GCH6E5	3	1	6
Yennefer z Vengerbergu	GC26F5M	1,5	2	2
Milicovsky les-haldy	GC45QEY	1,5	2,5	6

Tabulka 5.14: Keše a čas potřebný k jejich nalezení na trase D

Název keše	ID	Obtížnost	Terén	Čas (min)
Most Petrovice	GC2WJ1J	1,5	2	4
Dinoskola	GC6C0XJ	1,5	1,5	3
Pamatnik obetem	GC4N5HX	1,5	1,5	2
Edikova keska	GC38XE7	1,5	2	3
Horni Mecholupy	GC4D7FV	1,5	1,5	3
STAR WARS cz [003]	GC59DHB	1	1	4



Obrázek 5.7: Navržená trasa výletu C modulem TripMaker



Obrázek 5.8: Navržená trasa výletu D modulem GeoTrip

5.8 Vylepšení ručního plánování

V posledním testu byla porovnávána efektivita ručního plánování tras s efektivitou plánování pomocí doplňku TripMaker.

Testu se zúčastnily čtyři osoby. Postupně jim bylo předloženo pět map se zakreslenými schránkami a startovní pozicí. Jejich cílem bylo nalézt mezi těmito schránkami trasu. Měřena byla délka trasy a rychlost jejího nalezení.

Následně byly porovnány délky nalezených tras s trasou, která byla navržena modulem. Osoby dosahovaly na malých instancích (do 7 keší) stejně dobrých výsledků, jako modul, a navíc nalézaly řešení rychleji. S rostoucím počtem keší trvalo osobám hledání nejkratší cesty déle, a navíc nacházely výrazně horší řešení než TripMaker.

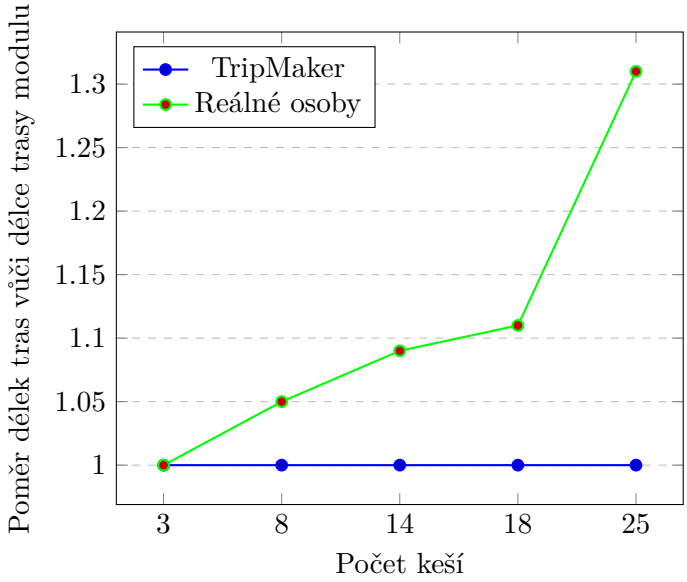
Naměřené časy hledání řešení jsou uvedeny v tabulce 5.15. Porovnání průměrné délky osobami nalezených tras s délkou trasy nalezenou modulem je zobrazeno na obr. 5.9.

5.9 Zhodnocení výsledků

Testy ukázaly, že plánování tras pomocí modulu je v průměrném případě efektivnější než ruční plánování. Odhadovaná doba trvání výletu bývá mírně

Tabulka 5.15: Srovnání času potřebného k nalezení trasy reálnými osobami a aplikací TripMaker (v sekundách)

Osoba	3 keše	8 keší	14 keší	18 keší	25 keší
Osoba 1	3	9	13	26	67
Osoba 2	2	9	9	14	32
Osoba 3	2	8	12	27	51
Osoba 4	4	7	13	26	58
TripMaker	5	7	8	9	12



Obrázek 5.9: Poměr průměrných délek tras navržených osobami a modulem TripMaker

podhodnocena. Nejlepších výsledků modul dosahuje, jestliže pro výběr podmnožiny schránek je použit výběr k nejbližších keší (vzhledem k počátečnímu bodu) a výsledná trasa je nalezena hladovým algoritmem a následně vylepšena 2-opt heuristikou.

Nalezená trasa může být delší než trasa nalezená aplikací OptiMap, avšak její nalezení trvá řádově kratší dobu. Konkurenční modul GeoTrip dodává na malých instancích řešení o pár sekund rychleji. Trasy nalezené TM však jsou kratší. TripMaker, na rozdíl od aplikací OptiMap a GeoTrip, umožňuje nalezení trasy i pro větší množiny schránek.

Závěr

Tato práce se zabývá navrhováním tras výletů spojených se hrou Geocaching. V teoretické části byly popsány algoritmy sloužící k řešení problému obchodního cestujícího a problému orientačního běžce, především hladové algoritmy a 2-aproximační algoritmus. Tyto algoritmy byly následně použity při implementaci modulu TripMaker do aplikace GeoGet, který na základě zvolených kritérií navrhuje vhodné trasy pro Geocaching.

Modul byl podroben sérii rozmanitých testů. Pro výběr podmnožiny schránek, které budou zahnuty do výletu, dosahoval nejlepších výsledků hladový algoritmus (výběr k nejbližších schránek vůči startovní pozici). Nalezení nejkratší trasy obsahující danou množinu keší bylo realizováno hladovým algoritmem vylepšeným 2-opt heuristikou, který dosahoval nejlepších výsledků z porovnávaných algoritmů.

Ve srovnání s konkurenční implementací modulu od Jana Staňka TripMaker nacházel výrazně kratší trasy. Byl také schopen nalézt trasu i pro větší množiny keší, avšak pro malé vstupy byl TripMaker o dvě až tři sekundy pomalejší než konkurenční modul.

Praktické testy ukázaly, že TripMaker dokáže dobře odhadovat časovou náročnost trasy a že je využitelný k reálnému plánování výletů spojených se hrou Geocaching.

Budoucí vývoj

V budoucnosti by bylo možné řešení rozšířit o textovou analýzu logů u keší, ze kterých by bylo možné získat informace o kvalitě keše a času potřebného k jejímu nalezení. Také by bylo možné plánovat nejen okružní výlety, ale také výlety s různým počátečním a koncovým bodem.

Na práci by bylo možné navázat vytvořením mobilní aplikace s podobnou funkcionalitou.

Literatura

[1] Kolář, J.: *Teoretická informatika*. Praha: ČVUT, 2009, ISBN 978-80-01-04331-8.

[2] Laporte, G.: The Traveling Salesman Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, ročník 59, č. 2, 1992: s. 231–247. Dostupné z: http://web.ist.utl.pt/~ist11038/CD_Casquilho/TSP1992EJOR_Laporte.pdf

[3] Skiena, S. S.: *The Algorithm Design Manual*. Springer Publishing Company, Incorporated, druhé vydání, 2008, ISBN 978-1848000698.

[4] Hilton, R.: Traveling Salesperson: The Most Misunderstood Problem [online]. 2012, [cit. 2016-04-30]. Dostupné z: http://www.nomachetejuggling.com/2012/09/14/traveling-salesman-the-most-misunderstood-problem/

[5] Chekuri, C.; Korula, N.; Pál, M.: Improved Algorithms for Orienteering and Related Problems. *ACM Trans. Algorithms*, ročník 8, č. 3, Červenec 2012: s. 23:1–23:27, ISSN 1549-6325. Dostupné z: http://doi.acm.org/10.1145/2229163.2229167

[6] Bellman, R.: Dynamic Programming Treatment of the Travelling Salesman Problem. *J. ACM*, ročník 9, č. 1, Leden 1962: s. 61–63, ISSN 0004-5411. Dostupné z: http://www.akira.ruc.dk/~keld/teaching/algoritmedesign_f08/Artikler/05/Bellman61.pdf

[7] Narahari, Y.: Optimal Solution for TSP using Branch and Bound [online]. 1999, [cit. 2016-05-01]. Dostupné z: http://lcm.csa.iisc.ernet.in/dsa/node187.html

[8] Tong, C.: Aproximation Algorithms for the Traveling Salesman Problem [online]. 2014, [cit. 2016-05-02]. Dostupné z: https://people.orie.cornell.edu/dpw/orie6300/Recitations/Rec12.pdf

[9] Brecklinghaus, J.; Hougardy, S.: The approximation ratio of the greedy algorithm for the metric traveling salesman problem. *Operations Research Letters*, ročník 43, č. 3, Květen 2015: s. 259–261. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0167637715000280

[10] Karpinski, M.; Lampis, M.; Schmied, R.: New Inapproximability Bounds for TSP. *ArXiv e-prints*, Březen 2013. Dostupné z: http://adsabs.harvard.edu/abs/2013arXiv1303.6437K

[11] Kim, B.; Shim, J.; Zhang, M.: Comparison of TSP Algorithm. *Project for Models in Facilities Planning and Materials Handlin*, 1998. Dostupné z: https://louisville.edu/speed/faculty/sheragu/Research/Intelligent/tsp.PDF

[12] Bansal, N.; Blum, A.; Chawla, S.: Approximation Algorithms for deadline-TSP and Vehicle Routing with Time-windows. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, New York, NY, USA: ACM, 2004, ISBN 1-58113-852-0, s. 166–174. Dostupné z: http://pages.cs.wisc.edu/~shuchi/papers/timewindows.pdf

[13] Blum, A.; Chawla, S.; Karger, D. R.: Approximation algorithms for orienteering and discounted-reward TSP. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, Oct 2003, ISSN 0272-5428, s. 46–55. Dostupné z: http://www.cs.cmu.edu/~avrim/Papers/orienteering-sicomp.pdf

[14] Geocaching : The Official Global GPS Cache Hunt Site [online]. ©2000-2016, [cit. 2016-05-05]. Dostupné z: https://www.geocaching.com

[15] Čermák, J.: Geocaching :: Opencaching [online]. ©2006-2016, [cit. 2016-05-05]. Dostupné z: https://www.opencaching.cz

[16] Krtička, L.: *Úvod do kartografie*. Ostravská univerzita v Ostravě, 2007, ISBN 978-80-7368-344-3.

[17] Jeffrey, C.: *An Introduction to GNSS*. NovAtel, Canada, 2010, ISBN 978-0-9813754-0-3.

[18] Thompson, R.: Global Positioning System:The Mathematics of GPS Receiver. *Mathematics magazine*, ročník 71, č. 4, Říjen 1998: s. 260–269. Dostupné z: http://www.maa.org/sites/default/files/pdf/cms_upload/Thompson07734.pdf

[19] COCOM GPS Tracking Limits [online]. 2010, [cit. 2016-05-06]. Dostupné z: http://ravtrack.com/GPStracking/cocom-gps-tracking-limits/469/

[20] Guido: GCVote - Let geocachers know your appreciation of a cache! [online]. ©2010-2016, [cit. 2016-05-06]. Dostupné z: http://gcvote.com

[21] C:geo team: c:geo 2016.04.12. [software]. Dostupné z: http://www.cgeo.org

[22] Groundspeak Inc. : Geocaching 4.1. [software]. Dostupné z: https://www.geocaching.com/mobile/

[23] MyGC 0.3.4.34. [software]. Dostupné z: http://www.instaluj.cz/mygc

[24] Gebauer, L.: GeoGet 2.9.6. [software]. Dostupné z: http://geoget.ararat.cz

[25] Gebauer, L.: Dokumentace programu GeoGet [online]. ©2006-2016, [cit. 2016-05-07]. Dostupné z: http://geoget.ararat.cz/doku.php/user

[26] RemObjects Software: Pascal Script for Delphi [online]. ©2001-2016, [cit. 2016-05-07]. Dostupné z: http://www.remobjects.com/ps.aspx

[27] Seznam.cz: Mapy.cz [online]. ©2000-2016, [cit. 2016-05-06]. Dostupné z: https://mapy.cz

[28] Qt Company: Qt 5.6.0. [software]. Dostupné z: https://www.qt.io

[29] OpenStreetMap contributors: OpenStreetMap [online]. ©2006-2016, [cit. 2016-05-07]. Dostupné z: https://www.openstreetmap.org

[30] KDE: Marble 1.9.1. [software]. Dostupné z: https://marble.kde.org/

[31] KDE: Marble [online]. ©1996-2016, [cit. 2016-05-07]. Dostupné z: http://api.kde.org/stable/kdeedu-apidocs/marble/html/

[32] Qt Company: Qt Documentation [online]. ©2016, [cit. 2016-05-06]. Dostupné z: http://doc.qt.io/

[33] Staněk, J.: *Problém cestujícího lovce hry Geocaching II.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

[34] Engdahl, G.: OptiMap - Fastest Roundtrip Solver [online]. ©2016, [cit. 2016-05-13]. Dostupné z: http://gebweb.net/optimap/

Příloha **A**

Seznam použitých zkratek

B&B Branch and Bound, metoda větví a hranic

GLONASS Globalnaja navigacionnaja sputnikovaja sistěma, ruský globální družicový polohový systém

GNSS Global Navigation Satellite System, globální družicový polohový systém

GPS Global Positioning System, globální polohovací systém

GPX GPS exchange format, soubor ukládající GPS data

GT GeoTrip

HC Hamiltonovská kružnice

HR hunger rate

MST Minimum Spanning Tree, minimální kostra grafu

NP nedeterministicky polynomiální

OP Orienteering problem, problém orientačního běžce

OSM OpenStreetMap, volně dostupná mapa světa

PET polyethylentereftalát

TM TripMaker

TSP Travelling salesman problem, problém obchodního cestujícího

XML Extensible Markup Language, rozšířený značkovací jazyk

Příloha B

Uživatelská příručka

B.1 Požadavky

TripMaker je doplňkem aplikace GeoGet. GeoGet je nativní WIN32 aplikace. Proto je nutné používat operační systém Windows (verze XP nebo vyšší) a mít nainstalován program GeoGet. Je zaručena kompatibilita s verzí aplikace GeoGet 2.9.5.

Pro plnou funkčnost je třeba mít instalován doplněk aplikace jménem GCVote.

B.2 Instalace

K instalaci modulu Tripmaker je potřeba instalační balíček TripMaker.gip. Tento balíček je k dispozici na CD, které je součástí této práce. Instalace proběhne automaticky po dvojitém kliknutím na zmíněný balíček.

B.3 Spuštění

Doplněk je možné spustit kliknutím na položku TripMaker v záložce menu Pluginy.

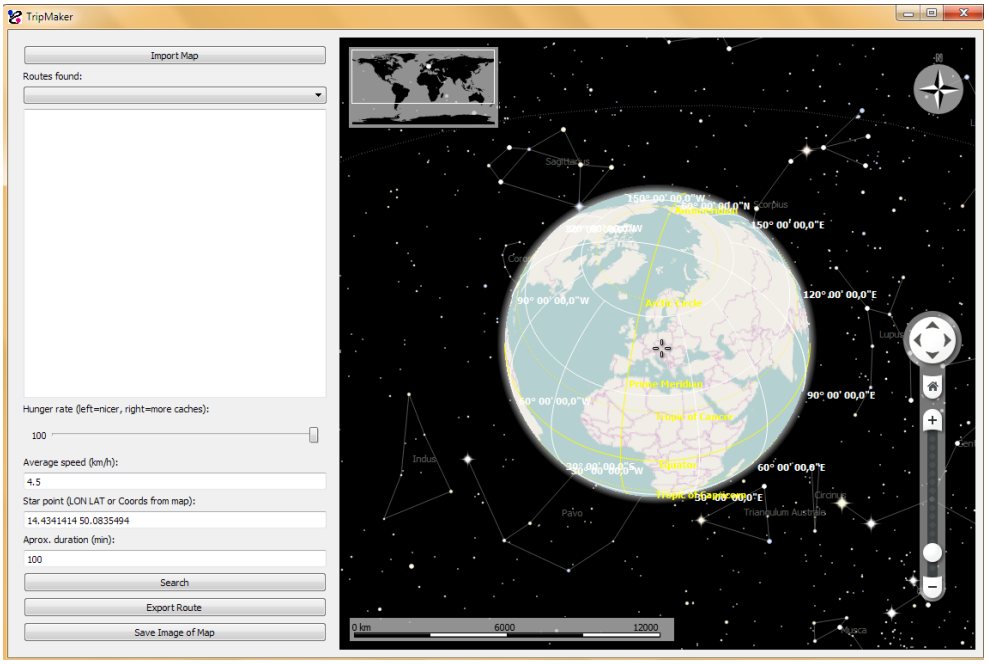
B.4 Základní prvky ovládání

V levé části okna modulu se nachází ovládací prvky a v pravé části je zobrazena mapa (viz obr. B.1 a B.2). Význam jednotlivých prvků (směrem shora dolů):

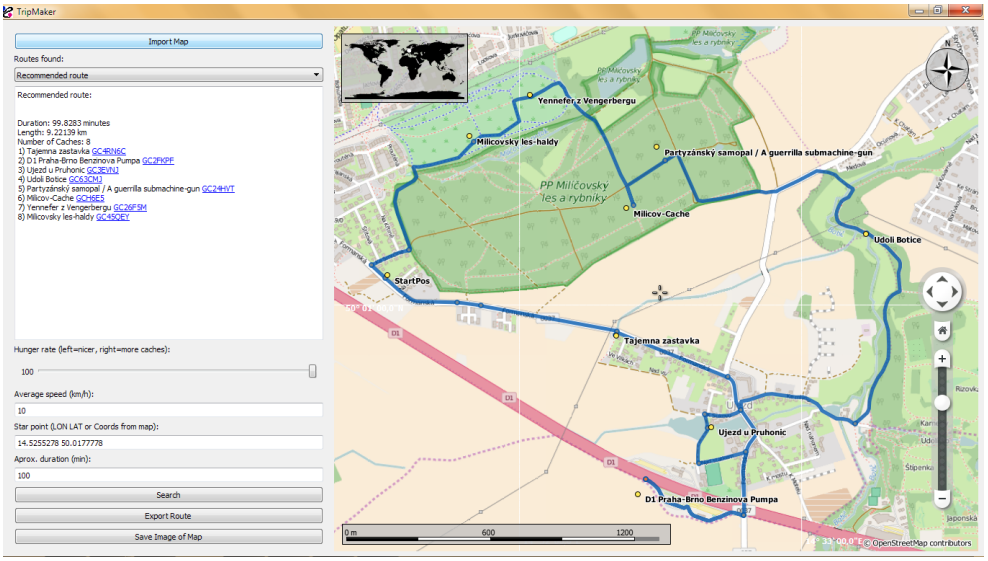
- Tlačítko Import Map umožňuje načtení mapového podkladu ve formátu *.osm.

- V rozbalovacím menu Routes found lze přepínat mezi navrhovanými trasami. Po zvolení trasy je tato trasa zobrazena na mapě a její popis je zobrazen v textovém poli.
- Textové pole slouží k zobrazování informací o navržené trase. Zobrazuje odhad doby trvání výletu, jeho délku a seznam keší, které jsou součástí výletu.
- Pomocí posuvného tlačítka (slideru) je možné nastavit hunger rate (požadovaný poměr kvantity a kvality keší).
- V políčku Average speed lze nastavit průměrnou rychlost pohybu hráče.
- Políčko Start point slouží k určení výchozích souřadnic hráče.
- Poslední nastavitelná hodnota Aprox. duration udává požadovanou dobu trvání výletu.
- Tlačítkem Search se spustí vyhledávání tras dle zadaných parametrů.
- Po stisknutí tlačítka Export Route je nalezená trasa uložena ve formátu GPX (do zvoleného souboru).
- Tlačítko Save Image of Map slouží k uložení aktuálního stavu mapy (včetně trasy) do obrázkového souboru.

Na mapě je zobrazeno měřítko a ovládání přiblížení mapy. Mapu je možné posouvat, přibližovat a oddalovat pomocí myši nebo klávesnice (šipkami a klávesami plus a mínus). Po kliknutí pravým tlačítkem na mapu se zobrazí menu. Pomocí Measure Points lze měřit na mapě vzdálenosti. Dále je možné nalézt nejkratší trasu mezi dvěma body, zkopírovat souřadnice daného místa nebo zobrazit výškový profil nalezené trasy.



Obrázek B.1: Výchozí okno modulu TripMaker



Obrázek B.2: Okno modulu TripMaker s nalezenou trasou

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Schovankova_Klara_2016.pdf` ..... text práce ve formátu PDF