České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Kuznetsova Anastasia

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: Aplikace pro zobrazení dojezdu elektrokola

Pokyny pro vypracování:

1) Udělejte rešerši problému výpočtu dojezdu elektrického kola ? vzorce pro dojezd, existující aplikace.
2) Formalizujte problém nalezení dojezdu elektrického kola ze zadaného bodu jako grafovou úlohu.
3) Navrhněte vhodný algoritmus, který řeší tento problém.
4) Implementujte algoritmus do aplikace pro chytré telefony s platformou Android.
5) Vyhodnoťte vlastnosti algoritmu a chování aplikace na realistických testovacích scénářích.

Seznam odborné literatury:

[1] A. Muetze and Y. C. Tan, "Performance evaluation of electric bicycles," Fourtieth IAS Annual Meeting. Conference Record of the 2005 Industry Applications Conference, 2005., 2005, pp. 2865-2872 Vol. 4.
[2] Fishman, Elliot, and Christopher Cherry. "E-bikes in the Mainstream: Reviewing a Decade of Research." Transport Reviews 36.1 (2016): 72-91.
[3] Rose, Geoffrey. "E-bikes and urban transportation: emerging issues and unresolved questions." Transportation 39.1 (2012): 81-96.
[4] Aslak Fyhri, Nils Fearnley, Effects of e-bikes on bicycle use and mode share, Transportation Research Part D: Transport and Environment, Volume 36, May 2015, Pages 45-52, ISSN 1361-9209

Vedoucí: Jan Hrnčíř, MSc., Ph.D.

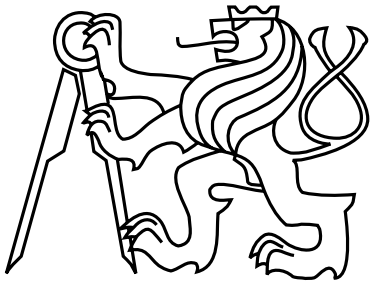
Platnost zadání do konce zimního semestru 2018/2019


prof. Dr. Michal Pěchouček, MSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 26.6.2017

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Diplomová práce

# Aplikace pro zobrazení dojezdu elektrokola

*Bc. Anastasia Kuznetsova*

Vedoucí práce: Jan Hrnčíř, Ph.D.

Studijní program: Otevřená informatika, magisterský navazující na bakalářský

Obor: Softwarové inženýrství

9. ledna 2018

# Poděkování

Ráda bych touto cestou vyjádřila poděkování Janu Hrnčířovi, Ph.D. za jeho cenné rady a pomoc při vedení mé diplomové práce. Rovněž bych chtěla poděkovat své rodině a přátelům za podporu v průběhu mého studia.

# Prohlášení

Prohlašuji, že jsem práci vypracovala samostatně a použila jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 9. ledna 2018 .....

# Abstract

The aim of this diploma thesis was to create a mobile application which would calculate electric bike range of the pedelec type based on data obtained from users and information about the terrain.

Factors influencing battery range and formula for its calculation were derived from analysis of existing solutions. The issue of range was formalized as an issue of finding the shortest paths from a given node of a cycling graph sourced from publicly available geographical data. To solve this problem, a modified version of Dijkstra's algorithm was used. Compared to an application called Bosch eBike range assistant, which was used as a reference, the precision of our application differs by 34.84% on average. Terrain testing would be necessary in order to achieve more precise analysis of results. That is beyond the time frame of this thesis.

The contribution of this thesis is within creation of a usable mobile application which is capable of fast and intuitive calculation of range of the pedelec electric bike type in Prague, Central Bohemia and the central part of Austria.

**Keywords:** electric bicycles, e-bike, pedelec, range of an electric bike, OpenStreetMap, Dijkstra's algorithm.

# Abstrakt

Cílem této diplomové práce bylo vytvořit mobilní aplikaci pro výpočet dojezdu elektrického kola typu pedelek na základě dat získaných od uživatele a informací o terénu.

Na základě analýzy existujících řešení byly odvozeny faktory ovlivňující dojezd a vzorec pro jeho výpočet. Problém nalezení dojezdu byl formalizován jako problém nalezení všech nejkratších cest z vybraného uzlu cyklistického grafu vytvořeného z veřejně dostupných geografických dat. K vyřešení problému byl použit modifikovaný Dijkstrův algoritmus. V porovnání s aplikací Bosch eBike range assistant, která posloužila jako referenční, se přesnot naší aplikace liší v průměru o 34,84 %. Pro přesnější analýzu výsledků by bylo potřeba provést testování v terénu, které přesahuje rámec této práce.

Přínosem této práce je použitelná mobilní aplikace umožňující rychlý a intuitivní výpočet dojezdu elektrického kola typu pedelek v Praze, Středočeském kraji a centrální části Rakouska.

**Klíčová slova:** elektrické kolo, e-kolo, pedelek, dojezd elektrického kola, OpenStreetMap, Dijkstrův algoritmus.

# Obsah

# Seznam obrázků

# Seznam tabulek

# Kapitola 1

# Úvod

## 1.1 Motivace

Jízdní kola poháněná elektrickou energií již nejsou tak nevšedními a nevídanými dopravními prostředky, jak tomu bylo dříve. Podle údajů Konfederace evropského bicyklového průmyslu Conebi prodej elektrických kol v Evropské unii se od roku 2013 do roku 2016 zvýšil skoro o 85 % [7]. Pojem elektrické kolo je rozsáhlý a označuje širokou škálu elektrických dopravních prostředků, které lze rozdělit do dvou skupin dle způsobu aktivace elektropohonu. Do první skupiny spadají dopravní prostředky, u kterých aktivace elektropohonu není závislá na šlapání, pro jejich popis se používá pojem e-kolo (angl. e-bike). Do druhé skupiny spadají ty prostředky, u nichž se pohon aktivuje šlapáním a je na něm přímo závislý, tyto dopravní prostředky mají název pedelek (angl. pedelec) [21].

V Evropské unii se pojem elektrické kolo obvykle používá k popisu pedeleků, neboť ty nejčastěji vyhovují evropské normě ČSN EN 15194 +A1, která říká, že na pozemních komunikacích či cyklostezkách mají povolení k pohybu ty prostředky, které jsou vybaveny elektrickým pohonem s maximálním trvalým jmenovitým výkonem 250 W a maximální rychlostí s výpomocí 25 km/h [9]. Díky tomu, že je na pedeleky pohlíženo jako na obyčejná kola bez elektropohonu a také díky jejich přístupnosti pro lidi různých věkových a výkonnostních kategorií, lze v posledních letech pozorovat nárůst zájmu o tyto prostředky. Zájem o pedeleky také zvyšuje jejich šetrnost k životnímu prostředí. Podle studie Evropské cyklistické federace z roku 2011 množství vyprodukovaného oxidu uhličitého na kilometr na osobu se u elektrických kol pohybuje kolem 21 gramů, u osobních aut se spalovacím motorem to je až 221 gramů [6].

Vzhledem k tomu, že pedeleky jsou rok od roku populárnějším dopravním i sportovním prostředkem, roste poptávka po aplikacích, které zpříjemní jejich používání. Aplikace, které budou schopny na základě dat od uživatele predikovat dojezd kola, jsou zejména zapotřebí pro příležitostné cyklisty či pro úplné začátečníky, kterým by měly usnadnit plánování trasy. Pro pokročilé uživatele budou podobné aplikace užitečné v místech, s jejichž terénem nejsou obeznámeni.

## 1.2 Cíl práce

Cílem této práce je vytvořit mobilní aplikaci, která bude schopna predikovat dojezd elektrického kola typu pedelek na základě dat získaných od uživatele a informací o terénu. K tomuto cíli se dostaneme pomocí následujících kroků.

Nejdříve vypracujeme rešerši problému výpočtu dojezdu elektrického kola. Dále si upřesníme pojmy spojené s danou problematikou a s elektrickými koly obecně. Bude následovat analýza existujících řešení. Poté se zaměříme na způsoby výpočtu dojezdu a popíšeme si faktory, které ho ovlivňují. Na základě nabytých poznatků odvodíme vzorec pro výpočet dojezdu.

Dále provedeme analýzu požadavků, na jejichž základě si problém upřesníme. Poté tento problém formalizujeme a převedeme na problém nalezení dojezdu ze zadaného bodu v grafu. Následně navrhneme a implementujeme algoritmus pro jeho řešení. Cílem práce je vytvořit mobilní aplikaci pro chytré telefony s platformou Android, která za využití navrženého algoritmu bude schopna najít dojezd pedeleku ze zadaného bodu.

Na konci vyhodnotíme vlastnosti algoritmu a chování aplikace na realistických testovacích scénářích.

## 1.3 Struktura práce

Teoretická část práce (kapitola 2) popisuje problematiku spojenou s výpočtem dojezdu a faktory, které ho ovlivňují. Následující kapitola 3 popisuje požadavky na aplikaci a návrh její architektury. Kapitola 4 specifikuje problém nalezení dojezdu pedeleku ze zadaného bodu jako grafovou úlohu. Kapitola 5 shrnuje poznatky a na základě nich popisuje proces zpracování geografických dat, získávání cyklistického grafu a algoritmus pro vyhledávání nejkratších cest z vybraného bodu. Implementace je následně popsána v kapitole 6. Popis testování aplikace obsahuje kapitola 7. Závěrečná kapitola 8 shrnuje celou práci a popisuje její možná rozšíření.

# Kapitola 2

# Analýza problému

Kapitola upřesňuje pojem elektrické kolo a jeho nejdůležitější komponenty a rozebírá proces odvození vzorce pro výpočet dojezdu. Na konci dochází k analýze existujících řešení a shrnutí celé kapitoly.

## 2.1 Jízdní kola s pomocným elektrickým pohonem

Elektrická kola lze rozdělit na dvě skupiny dle způsobu aktivace elektropohonu: e-kola a pedeleky. Tato práce je zaměřena pouze na pedeleky, které splňují platné legislativní podmínky a mají povolení k pohybu na pozemních komunikacích přístupných pro cyklisty. Pod pojmem elektrické kolo tak budeme nadále myslet právě pedelek.

Nejdůležitějšími komponentami elektrického kola jsou řídící jednotka, elektromotor a baterie. Jejich technické charakteristiky zásadně ovlivňují výkonnost kola a jeho dojezd.

### 2.1.1 Řídící jednotka

Řídící jednotka je jednou z nejdůležitějších součástek elektrického kola. Na základě informací ze snímačů, ovládacích prvků kola a baterie, řídí tato jednotka dávkování výkonu elektromotoru s ohledem na spotřebu. Přes řídící jednotku se také nastavují parametry jako intenzita asistence, maximální povolená rychlost, atd [22].

Řídící jednotka je často propojena s displejem na řídítkách kola, který zobrazuje její nastavení a další informace o stavu kola, jako jsou například zbytková kapacita baterie či dojezdová vzdálenost za ideálních podmínek.

### 2.1.2 Elektromotor

U elektromotoru velmi záleží na jeho umístění, neboť se od něj přímo odvíjí styl jízdy a výkon kola. Elektromotory rozdělujeme dle umístění na následující typy:

- **Motor v předním náboji.** Motor se instaluje do předního kola, které pak pohání. Výhodami daného typu jsou jeho cenová dostupnost, snadná demontáž a dobrá vyváženost kola. Hlavními nevýhodami jsou obtížnější řízení kola z důvodu větší rotující síly

předního kola, občasné prokluzování předního kola a nižší výkon motoru způsobený menší robustností vidlice předního kola[18] .

- **Motor v zadním náboji.** Motor se umisťuje do zadního kola, čímž poskytuje lepší přenášení točivého momentu. Díky větší robustnosti vidlice zadního kola lze na ni nainstalovat výkonnější motor než v případě kola předního. Nevýhodami jsou horší vyváženost kola a složitější demontáž v případě defektu zadního kola.

- **Středový motor.** Motor je instalován místo převodníku a středu klik a pohání zadní kolo prostřednictvím řetězu převodníku[12]. Jednou z největších výhod je možnost měnit převodový poměr pomocí přehazovačky, díky čemuž je kolo úspornější. Dalšími výhodami jsou dobrá vyváženost kola a plynulý pohon. Nevýhodami jsou vyšší cena a větší míra opotřebení hnacího ústrojí.

### 2.1.3 Baterie

Baterie je jednou z nejdůležitějších součástí elektrokola, neboť na ní závisí jeho dojezd. V současné době se nejvíce používají Li-Ion či Li-Pol baterie, které mají dobré poměry váhy, životnosti a výdrže, na rozdíl od svyh předchůdců – olověných či nikl-metal hydridových baterií [19].

U baterie jsou nejdůležitějšími parametry její kapacita (udává se v ampérhodinách (Ah)) a elektrické napětí (jednotkou je volt(V)).

## 2.2 Analýza existujících řešení

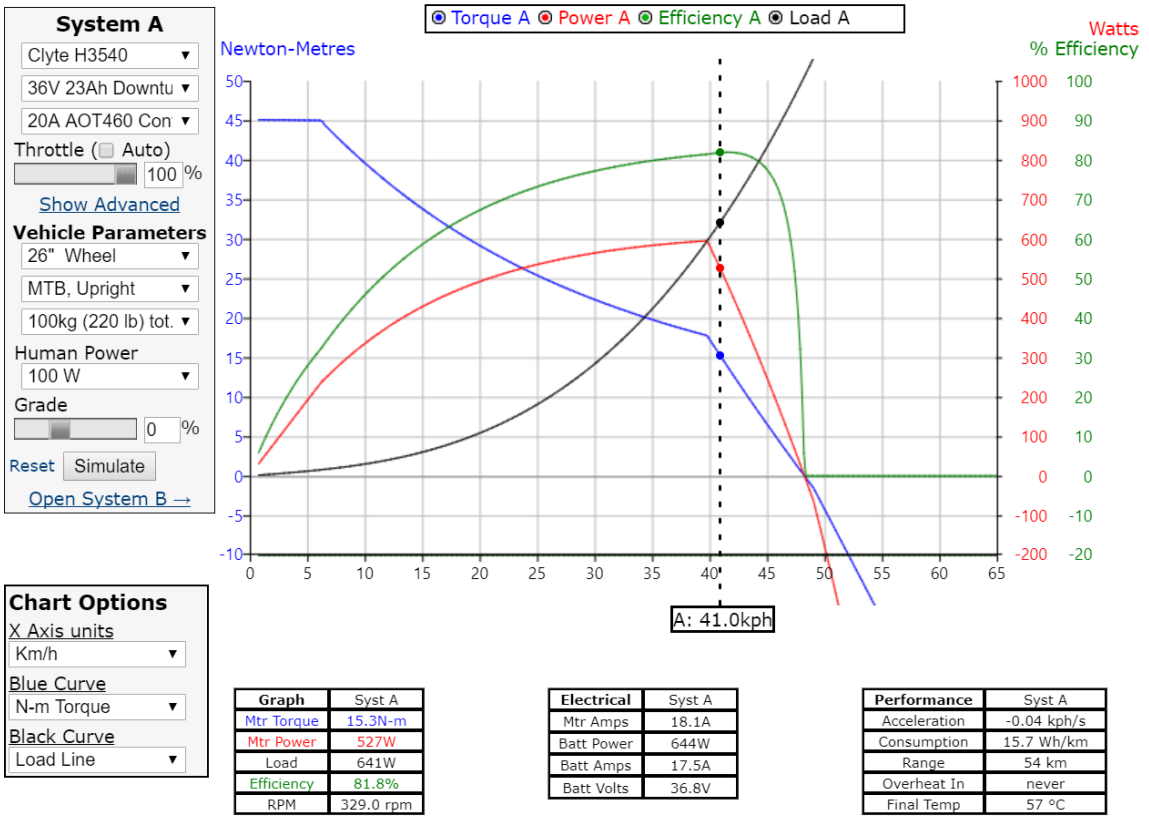
Dostupná řešení můžeme rozdělit do dvou skupin. Do první skupiny spadají nástroje určené primárně pro pokročilé uživatele, jedním z nich je například Motor Simulator[1]. Do druhé skupiny řadíme aplikace primárně určené pro méně zkušené uživatele. Příkladem takových aplikací můžou být třeba Electric Bike Simulator[2] nebo eBike range assistant[3] od společnosti Bosch. Většina existujících řešení je reprezentována webovým rozhraním a nebere v potaz reálné převýšení terénu.

### 2.2.1 Motor Simulator

Tento velmi precizní webový kalkulátor je určen zejména pro zkušené cyklisty. Uživatel musí uvést hodnoty parametrů jako typ elektromotoru, kapacita a napětí baterie, typ kontroleru, velikost kola a převýšení trasy.

Hlavními nevýhodami aplikace jsou absence výpočtu pro středový motor a nutnost počítat s konstantním převýšením terénu. Aplikaci také chybí responzivní rozhraní, které by usnadnilo její použití na mobilních zařízeních.

[1] <http://www.ebikes.ca/tools/simulator.html>
[2] <http://www.electricbikesimulator.com/>
[3] <https://www.bosch-ebike.com/en/service/range-assistant/>




Obrázek 2.1: Ukázka rozhraní aplikace *Motor Simulator*.

### 2.2.2 Electric Bike Simulator

Tato webová aplikace pro výpočet dojezdu kola má uživatelsky příjemné a intuitivní provedení. Uživatel může nastavit hodnoty parametrů, jako jsou celková hmotnost kola a cyklisty, převýšení terénu, aerodinamický koeficient, akcelerátor, výbava kola a další. Pro všechny parametry jsou uvedené výchozí hodnoty, což ulehčuje výpočet zejména pro začínající cyklisty.

Hlavní nevýhodou dané aplikace je to, že umožňuje výpočet pouze pro konstantní převýšení terénu a není přizpůsobena pro použití na mobilních zařízeních.

### 2.2.3 eBike range assistant

Firma Bosch ve snaze poskytnout zákazníkům co nejpřesnější odhad dojezdu nabízí speciální webovou kalkulačku určenou pro konkrétní modely pedeleků značky Bosch. Při výpočtu dojezdu se v této aplikaci používají převážně předdefinované konstanty, které jsou stanovené pro každý model či konfiguraci kola. Tento přístup výrazně zlepšuje odhad dojezdů. Aplikace také umožňuje uživateli nastavit až třináct vstupních parametrů ze tří skupin: řidič, výbava kola a prostředí.


Obrázek 2.2: Ukázka rozhraní aplikace *Electric Bike Simulator*.

Nevýhodou aplikace je to, že je určena pro elektrická kola značky Bosch a je nepoužitelná pro majitele jiných značek, neboť neumožňuje libovolně měnit hodnoty ve skupině parametrů „Výbava kola“.

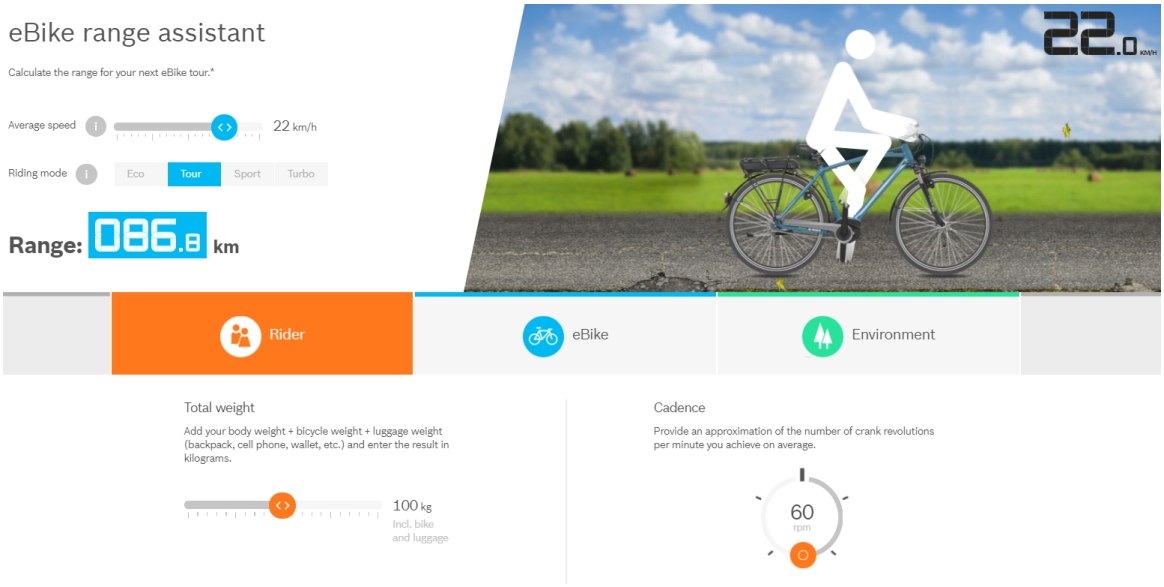
## 2.3 Výpočet dojezdu elektrického kola

Dojezd elektrického kola je vzdálenost, po kterou se projevuje výpomoc elektromotoru, než se vybije baterie[1]. Dojezd, který obvykle uvádí výrobci, lze pojmenovat jako teoretický dojezd kola. Jedná se o hodnotu, která udává dojezd za ideálních podmínek, jako jsou plochý terén, nízká průměrná rychlost, malá váha jezdce, nejnižší stupeň podpory při šlapání, atd. Například firma Bosch [8] při výpočtu dojezdu za ideálních podmínek předpokládá, že cyklista se pohybuje plochým terénem s průměrnou rychlostí 15 km/h při použití nejnižšího stupně podpory, kdy motor poskytuje 40 % energie, kterou cyklista dodává při šlapání.

Dojezd elektrického kola v reálném prostředí, neboli reálný dojezd, se může hodně lišit od teoretického. Jeho výpočet je také mnohem složitější, neboť zahrnuje více vnějších faktorů.

Faktory, které ovlivňují reálný dojezd elektrického kola, lze rozdělit do tří skupin: faktory související s výbavou kola, vnější faktory a faktory související s cyklistou[10].

Do faktorů souvisejících s výbavou kola spadají kapacita a napětí baterie, stáří baterie, tlak v pneumatice, výkon motoru, apod.




Obrázek 2.3: Ukázka rozhraní aplikace *eBike range assistant*.

Vnější faktory jsou faktory prostředí, v němž se cyklista pohybuje. Do této skupiny faktorů spadá například převýšení terénu, stav a typ vozovky, venkovní teplota, větrnost, oblast, ve které se cyklista pohybuje, nebo třeba počet semaforů či zastávek.

Poslední skupinou jsou faktory související se samotným cyklistou, jako je celková hmotnost kola a cyklisty, rychlost jízdy, styl jízdy nebo vybraný stupeň asistence.

Přesnost vypočítaného dojezdu je přímo závislá na přesnosti vstupních faktorů. Čím přesnější jsou faktory, tím věrohodnější bude odhad. Přesné zahrnutí všech faktorů do výpočtu dojezdu je ideálním případem, v praxi se ale jedná o skoro neproveditelnou úlohu. Jednak kvůli nedostatečnému počtu snímačů na elektrických kolech[11], jednak kvůli rychlé proměnlivosti některých faktorů [10]. Je tedy nutné určit takové faktory, jež budou mít největší vliv na dojezd. Pro tyto faktory (dále vstupní parametry) budeme vyžadovat přesná data z reálného prostředí. Ostatní faktory budeme považovat za méně důležité a budou nabývat konstantních hodnot za ideálních podmínek.

Určení vstupních parametrů je složitou úlohou, jak pojednává článek [10]. Je těžké přesně určit, jak a které faktory nejvíce ovlivňují dojezd. Článek [11] také zdůrazňuje, že problém navíc zesložiťuje i nedostatečné množství výzkumů zabývajících se danou problematikou.

Vstupní parametry lze určit na základě již existujících řešení, nejčastěji se používají následující parametry:

- Průměrná rychlost jízdy
- Stupeň podpory při šlapání
- Celková hmotnost kola a cyklisty
- Kapacita a elektrické napětí baterie

- Šířka pláště kola, velikost rámu
- Typ vozovky
- Převýšení trasy
- Síla a směr větru

Při odvození vzorců popsaných níže jsme uvažovali pouze vstupní parametry, které lze získat z naší aplikace a jež považujeme za nejdůležitější na základě analýzy článků a vzorců pro výpočet. Jedná se o následující parametry: průměrná rychlost cyklisty, stupeň podpory při šlapání, kapacita a napětí baterie, celková hmotnost kola a cyklisty a převýšení trasy. Pro výpočet dojezdu na základě těchto faktorů lze použít vzorec popsány níže.

Nejdříve definujeme výkon potřebný pro překonání odporu prostředí $P_{drag}[W]$, výkon potřebný k překonání úseků bez svahu $P_{flat}[W]$ a výkon potřebný k překonání úseků se svahem $P_{hill}[W]$.

$$P_{drag} = 0.5 \cdot D \cdot v_g^3 \cdot C_d \cdot A,$$

$$P_{flat} = m \cdot R_c \cdot g \cdot v_g,$$

$$P_{hill} = m \cdot \cos\left(\tan\left(\frac{\alpha}{100}\right)\right) \cdot \left(R_c + \tan\left(\tan\left(\frac{\alpha}{100}\right)\right)\right) \cdot g \cdot v_g,$$

kde $D[kg/m^3]$ je hustota vzduchu, $v_g[m/s]$ je průměrná rychlost jízdy, $C_d$ je součinitel vzdušného odporu, $A[m^2]$ je čelní plocha cyklisty, $m[kg]$ je celková hmotnost kola a cyklisty, $R_c$ je valivý odpor pláště po povrchu cesty, $g[m/s^2]$ je tíhové zrychlení a $\alpha$ je úhel mezi vodorovnou rovinou a úsekem cesty. Celkovou energii $e_{total}[Wh]$ potřebnou pro zdolání úseku délky $l[m]$ při účinnosti motoru $\eta$ budeme počítat zvlášť pro vodorovné úseky a úseky se sklonem. Vzhledem k tomu, že v reálném městě většina tras buď stoupá nebo klesá a jen stěží najdeme úseky, které jsou vodorovné, definujeme podmínky, za nichž úsek budeme považovat za vodorovný $\alpha \in \langle -0.5, 0 \rangle$, stoupající $\alpha \geq 0$ či klesající $\alpha \leq -0.5$.

Celkovou energii pak vypočítáme následovně. Pro vodorovný úsek platí

$$e_{total} = \frac{0.001 \cdot l \cdot (P_{drag} + P_{flat})}{3.6 \cdot \eta \cdot v_g},\ \alpha \in \langle -0.5, 0 \rangle, \tag{2.1}$$

pro stoupající úsek platí

$$e_{total} = \frac{0.001 \cdot l \cdot (P_{drag} + P_{hill})}{3.6 \cdot \eta \cdot v_g},\ \alpha > 0, \tag{2.2}$$

pro klesající úsek platí

$$e_{total} = 0,\ \alpha < -0.5. \tag{2.3}$$

Energii $e_{bicycle}[Wh]$, kterou musí vynaložit kolo pro zdolání úseku délky $l[m]$ s vybranou asistencí $a$, vypočítáme takto:

$$e_{bicycle} = e_{total} \cdot \frac{a}{1+a}.$$

Pro výpočet maximálního dojezdu $r[m]$ nejprve vypočítáme energii $e_{battery}[Wh]$, kterou je schopno dodat elektrické kolo s napětím baterie $V_{battery}[V]$ a kapacitou $A_{battery}[Ah]$,

$$e_{battery} = A_{battery} \cdot V_{battery}. \tag{2.4}$$

Maximální dojezd v metrech vypočítáme následovně

$$r = \frac{1000 \cdot e_{battery}}{e_{bicycle}}. \tag{2.5}$$

$$\tag{2.6}$$

## 2.4 Shrnutí

V této kapitole jsme se nejprve zaměřili na pojmy vztahující se k elektrickým kolům a popsali jsme jejich nejdůležitější komponenty. Po analýze existujících řešení jsme určili, že největším problémem všech dostupných aplikací je neuvažování terénu se stoupáním a absence responzivního rozhraní, ta výrazně zhoršuje jejich přístupnost pro cyklisty. Po prozkoumání faktorů ovlivňujících dojezd kola jsme na základě analýzy stanovili, jaké vstupní parametry budeme uvažovat při výpočtu. Nakonec jsme s ohledem na stanovené parametry odvodili vzorec pro výpočet reálného dojezdu kola.

# Kapitola 3

# Softwarový návrh aplikace

Kapitola obsahuje požadavky na mobilní aplikaci a případy jejího užití, dále popisuje návrh architektury a proces tvorby prototypu uživatelského rozhraní.

## 3.1 Analýza požadavků

Při návrhu aplikace je důležité správně určit požadavky na ni kladené, neboť se od nich odvíjí spokojenost uživatelů. Požadavky umožňují určit rozsah aplikace a identifikují nutné kroky a akce při jejím návrhu.

Při analýze požadavků definujeme vlastnosti, které aplikace musí splňovat. Rozlišujeme dva druhy požadavků: funkční a obecné (kvalitativní). Funkční požadavky popisují akce a aktivity, které aplikace musí umožňovat. Pro změnu obecné požadavky definují návrh aplikace, požadavky na efektivitu či použitelnost [13].

Při návrhu požadavků na mobilní aplikaci jsme vycházeli z poznatků nabytých v předchozí kapitole 2.

### 3.1.1 Funkční požadavky

Mobilní aplikace by měla splňovat následující funkční požadavky:

- **Výběr startovního bodu.** Aplikace umožní uživateli určit startovní bod kliknutím na mapu.
- **Vyhledávání dojezdu.** Aplikace umožní vyhledání dojezdu ve vybraném regionu na základě informací získaných od uživatele a informací o terénu.
- **Vstupní parametry.** Aplikace umožní přidávání, změnu a ukládání vstupních parametrů.
- **Výběr regionu.** Aplikace umožní uživateli výběr ze dvou regionů na mapě - centrální část Rakouska, Praha a Středočeský kraj. Uživatel bude schopen změnit vybraný region, aniž by musel aplikaci znovu instalovat.
- **Zobrazení dojezdu.** Aplikace bude zobrazovat několik oblastí dojezdu s různou spotřebou kapacity baterie.

### 3.1.2 Obecné požadavky

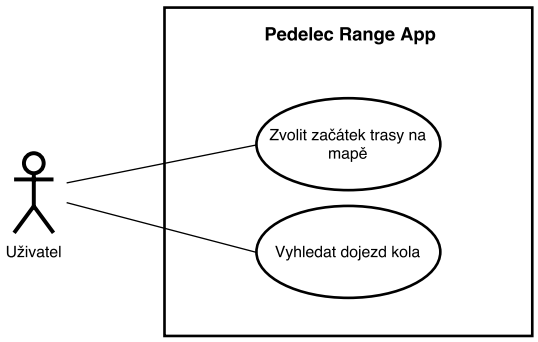
Mobilní aplikace by měla splňovat následující obecné požadavky:

- **Informace o terénu.** Aplikace musí využívat informací o terénu při určení dojezdu elektrického kola.
- **Přístupnost a použitelnost aplikace.** Aplikace musí být srozumitelná a použitelná jak pro příležitostné, tak pro pokročilé cyklisty.
- **Operační systém Android.** Aplikace musí být schopná provozu na zařízeních Android a to s verzí Android 4.1 a vyšší.

## 3.2 Případy užití

Diagram případů užití popisuje ty funkce aplikace, které přináší viditelný výsledek pro uživatele. Pomocí diagramu případů užití se určují omezení aplikace. Také nabízí jiný pohled na funkční požadavky popsané v 3.1. V diagramech níže vystupuje pouze jeden aktér, který zastupuje cyklistu – uživatele.

V mobilní aplikaci máme dva hlavní diagramy případů užití. První diagram 3.1 popisuje případy užití spojené s procesem vyhledávání dojezdu. Uživatel nejdřív musí vybrat startovní bod na mapě, pak může vyhledat dojezd kola.

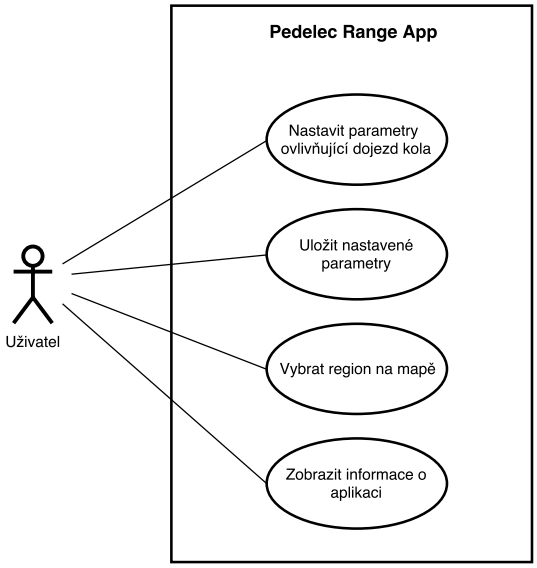



Obrázek 3.1: Diagram případů užití 1: Vyhledávání dojezdu.

Druhý diagram 3.2 znázorňuje případy užití spojené se správou nastavení aplikace. Uživatel může zadávat a upravovat nastavení aplikace, dále může změnit region pro výpočet dojezdu nebo zobrazit informace o aplikaci.

## 3.3 Architektura aplikace

Jak je vidět z diagramu nasazení 3.3, pro splnění cílů popsaných na začátku této kapitoly budou vytvořeny dvě aplikace. První desktopová aplikace provede zpracování geografických dat




Obrázek 3.2: Diagram případu užití 2: Správa nastavení aplikace.

regionu (geodata) a na základě nich sestaví cyklistický graf, který bude upravenou a zjednodušenou verzí těchto dat. Druhá mobilní aplikace pak umožní uživateli na tomto cyklistickém grafu vyhledávat dojezd kola.

Architekturu mobilní aplikace pro vyhledávání dojezdu můžeme rozdělit na tři logické komponenty:

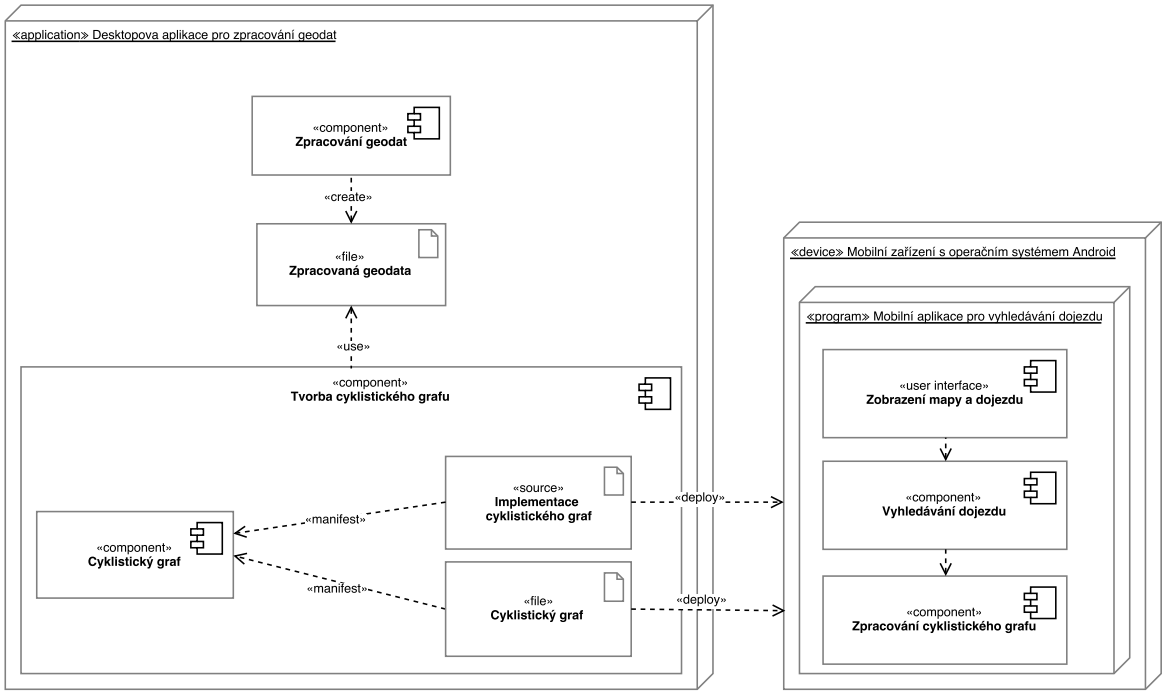
- **Zobrazení mapy a dojezdu.** Komponenta odpovídá za uživatelské rozhraní, konkrétně za zobrazení dojezdu a mapy uživateli.
- **Vyhledávání dojezdu.** V této komponentě se bude provádět vyhledávání dosažitelných uzlů.
- **Zpracování cyklistického grafu.** Tato komponenta odpovídá za inicializaci grafu.

## 3.4 Návrh prototypu uživatelského rozhraní

Prototyp uživatelského rozhraní mobilní aplikace byl navržen na základě poznatků získaných z této a předešlé kapitoly 2. Jak bylo uvedeno výše v kapitole 3.1, aplikace musí být srozumitelná jak začátečníkům, tak i pokročilým cyklistům. Hlavním požadavkem na uživatelské rozhraní je tedy jeho použitelnost. K dosažení tohoto cíle při návrhu rozhraní byla brána v potaz heuristicka pravidla (dále jen HP) použitelnosti stanovená Jakobem Nielsenem [20]:

**HP 1: Viditelnost stavu aplikace.** Aplikace by měla vždy informovat uživatele o tom, co se děje.

**HP 2: Propojení aplikace a reálného světa.** Aplikace by měla mluvit jazykem srozumitelným uživatelům a dodržovat konvence reálného světa.




Obrázek 3.3: Diagram nasazení.

**HP 3: Uživatelská kontrola a svoboda.** Uživatelé často dělají chyby, proto musí mít možnost vrátit se zpět a svou volbu zrušit nebo změnit.

**HP 4: Standardizace a konzistence.** Aplikace musí používat prvky, které jsou uživateli dobře známé. Zároveň použité zobrazovací prvky musí být konzistentní.

**HP 5: Prevence chyb.** Je lepší chybě úplně předejít, než ji oznámit uživateli. Aplikace musí zamezit tvorbě chyb.

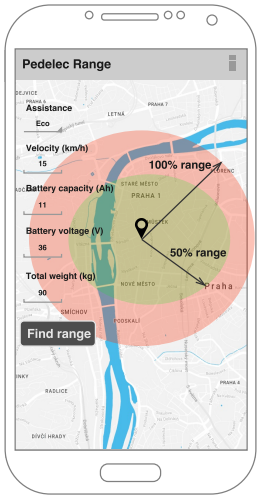
**HP 6: Estetický a minimalistický design.** Uživatelské rozhraní musí obsahovat pouze nezbytné informace. Každá další jednotka informací zabírá místo a činí aplikací nepřehlednou.

**HP 7: Rozpoznání místo vzpomínání.** Aplikace by neměla nutit uživatele k zapamatovávání si kroků, které v rozhraní udělal, či údajů, které zadal.

**HP 8: Flexibilní a efektivní použití.** Zkušený uživatel musí mít možnost urychlit některé akce nebo je přeskočit.

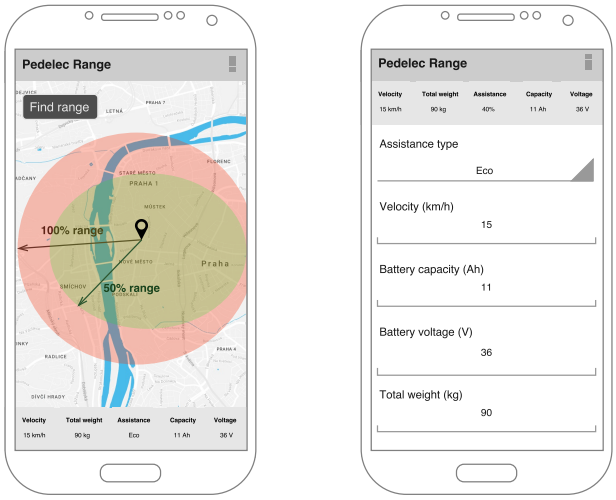
**HP 9: Pomoc uživateli pochopit, poznat a vzpamatovat se z chyb.** Pokud dojde k chybě, uživatel by měl být schopen pochopit, co k ní vedlo, a mít možnost vrátit se v aplikaci do původního stavu.

**HP 10: Návod a dokumentace.** Nápověda nebo dokumentace jsou nutnou pomůckou uživateli, musí ale být srozumitelné a lehce dohledatelné.




Obrázek 3.4: První prototyp aplikace.

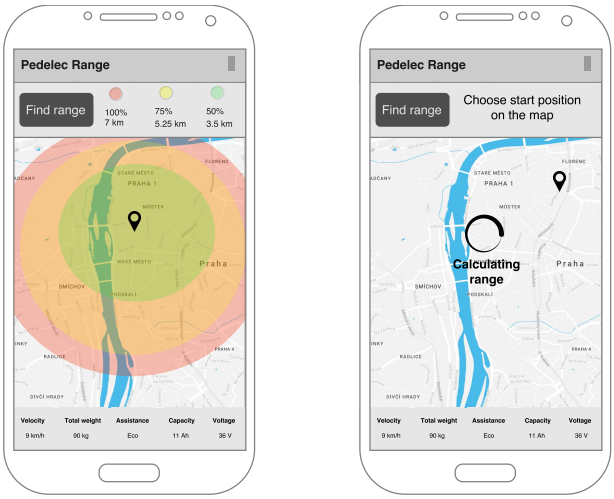
Návrh prototypu probíhal iterativně. První vytvořeny prototyp 3.4 porušoval HP 5, HP 6, HP 8, HP 9 a HP 10. Nastavitelné parametry zabíraly skoro polovinu levé strany obrazovky, a činily tak aplikaci nepřehlednou (porušen HP 6). Dalším problémem byla skutečnost, že se po spuštění aplikace nezobrazilo žádné informační okno, a tak nebylo jasné, že pro vyhledávání dojezdu musí uživatel kliknout na mapu (porušeny HP 5 a HP 9). Políčka pro vstupní paramtery neobsahovala předvyplněné hodnoty, což vedlo ke zhoršení každodenního používání aplikace (porušen HP 8). Způsob umístění políček pro vstupní parametry také měl další závažnou nevýhodu, kvůli omezenému prostoru nebylo možné do něj přidávat další vstupní paramtery, což nepřispívalo případnému rozšíření nabídky parametrů v budoucnu.




Obrázek 3.5: Druhý prototyp aplikace.

Při tvorbě druhého prototypu jsme brali v potaz chyby objevené v prvním návrhu. Jak

je vidět na obrázku výše 3.5, k nastavení vstupních parametrů se začal používat posuvný panel (zdola nahoru), přičemž ten nelze zcela skrýt, a vždy tak v dolní liště zobrazuje nastavené parametry. Díky tomuto řešení je úvodní obrazovka aplikace mnohem přehlednější, byl tak splněn HP 6. Prototyp také počítal s tím, že po spuštění aplikace budou vstupní hodnoty již předvyplněny, a to buď výchozími hodnotami, nebo hodnotami posledně zadanými uživatelem, což vedlo ke splnění HP 8. Problémem daného prototypu byla absence přesných informací o dojezdu elektrického kola, například množství kilometrů, které bylo kolo schopno ujet.




Obrázek 3.6: Třetí prototyp aplikace.

Třetí, finální prototyp je vidět na obrázku 3.6. Je minimalistický (HP 6), ale zároveň zobrazuje všechny potřebné informace (HP 4 a HP 7). Aplikace jasně a srozumitelně informuje uživatele o svem stavu (HP 1 a HP 2). Na úvodní obrazovce se také zobrazuje návod, který popisuje jaké kroky uživatel musí učinit k nalezení dojezdu (HP 10). Prototyp také zaručuje, že po otevření aplikace vstupní parametry obsahují buď výchozí hodnoty, nebo hodnoty naposledy nastavené uživatelem, což výrazně zrychlí použití aplikace (HP 8). Prototyp také informuje uživatele, pokud dojde ke špatnému vyplnění parametru, a to rovnou v průběhu vyplňování políčka (HP 5 a HP 9). Nastavený parametr lze kdykoliv změnit (HP 3).

# Kapitola 4

# Specifikace problému

Kapitola obsahuje definici grafu cyklistické sítě, formalizaci a převedení problému nalezení dojezdu kola na grafovou úlohu.

## 4.1 Graf cyklistické sítě

Pro popis našeho grafu použijeme definici cyklistického grafu z článku [2]. Ohodnocený souvislý jednoduchý graf $G = (V, E, g, h, l)$ bude vytvořen na základě zjednodušených geodat vybraného regionu. $V$ je množina uzlů grafu a $E = \{(u, v)|u, v \in V \wedge u \neq v\}$ je množina hran, které odpovídají jednotlivým segmentům mapy. Funkce $g : V \to \mathbb{R}^2$ přiřadí každému uzlu zeměpisnou šířku a délku, funkce $h : V \to \mathbb{R}$ mu přiřadí nadmořskou výšku. Za ohodnocení grafu odpovídá funkce $l : E \to \mathbb{R}^+$, která přiřadí každé hraně $(u, v) \in E$ její délku odpovídající reálné vzdálenosti mezi dvěma body na mapě.

## 4.2 Definice funkcí a symbolů k řešení problému dojezdu

Tato podkapitola podrobně popisuje funkce a symboly, které se budou používat při popisu řešení problému nalezení dojezdu.

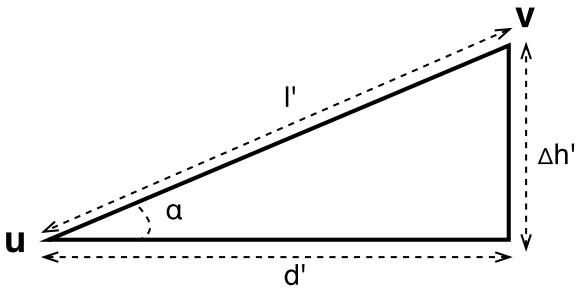
### 4.2.1 Výpočet vzdálenosti mezi dvěma uzly grafu

Jak je vidět na obrázku 4.1, mezi uzly $u$, $v \in V$ lze definovat nejkratší vzdálenost $d'$ a reálnou vzdálenost $l'$ vypočítanou s ohledem na nadmořskou výšku bodů. Při výpočtech vzdálenosti použijeme geodetický standard World Geodetic System ve verzi WGS-84, který definuje souřadnicový systém a referenční elipsoid.

Funkce $d$ pro výpočet nejkratší vzdálenosti $d'[m]$ mezi uzly $u$, $v$ vypadá následovně [23],

$$a = \sin^2\left(\frac{\phi_v - \phi_u}{2}\right) + \cos\phi_u \cdot \cos\phi_v \cdot \sin^2\left(\frac{\lambda_v - \lambda_u}{2}\right),$$

$$d = 2 \cdot R \cdot arctan2\left(\sqrt{a}, \sqrt{1-a}\right),$$




Obrázek 4.1: Úseky určující nejkratší a reálnou vzdálenost mezi body $u$, $v$.

kde $\phi_v$ je zeměpisná šířka uzlů $v$ v radiánech, $\phi_u$ je zeměpisná šířka uzlů $u$ v radiánech, $\lambda_v$ je zeměpisná délka uzlu $v$ v radiánech, $\lambda_u$ je zeměpisná délka uzlu $u$ v radiánech a $R = 6378137[m]$ je velká poloosa Země.

Pro výpočet reálné vzdálenosti $l'$ mezi dvěma uzly nejprve převedeme zeměpisné souřadnice uzlů do kartézské soustavy. Dle textu [3] prostorovou pravoúhlou souřadnici pro uzel $u$ s nadmořskou výškou $h_u[m]$, zeměpisnou šířkou $\phi_u$ v radiánech a zeměpisnou délkou $\lambda_u$ v radiánech odvodíme pomocí formulí:

$$u_x = (\nu + h_u) \cdot \cos\phi_u \cdot \cos\lambda_u,$$

$$u_y = (\nu + h_u) \cdot \cos\phi_u \cdot \sin\lambda_u,$$

$$u_z = (\nu \cdot (1 - e^2) + h_u) \cdot \sin_u\phi,$$

kde $\nu = \frac{R}{\sqrt{1-e^2 \cdot sin^2(\phi)}}$, $e^2 = 0.0066943801$ je numerická výstřednost (první excentricita). Funkce ohodnocení $l$ grafu $G$, která vrátí reálnou vzdálenost $l'$ mezi dvěma uzly $u$, $v$ v euklidovském prostoru, je definována následovně

$$l = \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2 + (u_z - v_z)^2}.$$

## 4.3 Výpočet úhlu mezi vodorovnou rovinou a úsekem cesty

Úhel $\alpha$, jak je zobrazen na obrázku 4.1, vypočítáme následovně:

$$\alpha = \arctan\left(\frac{h_v - h_u}{d'}\right),$$

kde $h_v$ je nadmořská výška uzlu $v$, $h_u$ je nadmořská výška uzlu $u$ a $d'$ je nejkratší vzdálenost mezi uzly $u$, $v$.

## 4.4 Výpočet energie potřebné pro zdolání úseku cesty

V podkapitole 2.3 jsme již definovali vstupní parametry, které se budou uvažovat při výpočtu dojezdu. Množina vstupních parametrů $P^n$ obsahuje následující položky:

- $v_g[m/s]$ průměrná rychlost jízdy.
- $a$ stupeň podpory při šlapání.
- $m[kg]$ celková hmotnost kola a cyklisty.
- $A_{battery}[Ah]$ kapacita baterie.
- $V_{battery}[V]$ napětí baterie.

Maximální množství energie, kterou je kolo schopno poskytnout, je $M[Wh]$,

$$M = A_{battery} \cdot V_{battery}.$$

Funkce $\omega$ na základě množiny vstupních parametrů $P^n$ pro každou hranu $(u, v) \in E$ vrátí číslo odpovídající množství energie, kterou musí poskytnout kolo pro její překonání. Dle podkapitoly 2.3 pro vodorovný úsek platí

$$\omega = \frac{0.001 \cdot l' \cdot (P_{drag} + P_{flat})}{3.6 \cdot \eta \cdot v_g}, \ \alpha \in \langle -0.5, 0 \rangle, \tag{4.1}$$

pro stoupající úsek platí

$$\omega = \frac{0.001 \cdot l' \cdot (P_{drag} + P_{hill})}{3.6 \cdot \eta \cdot v_g}, \ \alpha > 0, \tag{4.2}$$

pro klesající úsek platí

$$\omega = 0, \ \alpha < -0.5, \tag{4.3}$$

kde $l'$ je reálná vzdálenost mezi uzly $u$, $v$ a $P_{drag}[W]$ je výkon potřebný pro překonání odporu prostředí, $P_{flat}[W]$ je výkon potřebný k překonání úseků bez svahu a $P_{hill}[W]$ je výkon potřebný k překonání úseků se svahem. Výkony se vypočítají dle vzorců z podkapitoly 2.3. Při výpočtu budeme počítat s konstantní účinnosti motoru $\eta = 0.8$.

## 4.5 Problém nalezení dojezdu pedeleku

Problém nalezení dojezdu pedeleku definujeme jako $C = (G, M, P^n, s)$, kde $G$ je graf cyklistické sítě 4.1, $M$ je maximální množství energie, kterou je schopno poskytnout kolo, $P^n$ je množina vstupních parametrů pro výpočet dojezdu 4.4 a $s \in V$ je počáteční uzel. Při popisu řešení problému budeme využívat pojem cesta grafu, který je definován následovně, cesta $\pi(u, v)$ grafu $G$ je posloupnost hran $\pi(u, v) = \langle (u_i, u_{i+1}), \ldots, (u_{k-1}, u_k) \rangle$ taková, že $u_i = u$, $u_k = v$, $k \leq |V|$ a zároveň $\nexists u_i = u_j$, kde $i \neq j$ pro $i, j \in \langle 1, 2, \ldots k \rangle$ [14]. Množina $\Pi(u, v)$ obsahuje všechny cesty z uzlu $u$ do uzlu $v$.

Řešením problému dojezdu je množina uzlů $T$, která obsahuje uzly ležící v maximální vzdálenosti $M$ od počátečního uzlu $s$. V našem problému je vzdálenost mezi dvěma uzly množství energie, kterou musí poskytnout kolo pro překonání cesty mezi těmito uzly.

Funkce $\omega$ na základě vstupních parametrů $P^n$ pro každou hranu $(u, v) \in E$ vrátí její váhu, neboli číslo odpovídající množství energie, kterou musí poskytnout kolo pro její překonání,

$$\omega : P^n \times E' \to \mathbb{R}_0^+,$$

kde $(u, m) \in E'$, pokud $(u, m) \in E$, nebo $(u, m) \notin E$ a zároveň $(u, v) \in E$ a uzel $m$ leží na hraně $(u, v)$.

Pak číslo odpovídající nejmenšímu množství energie, kterou musí poskytnout kolo pro zdolání cesty, neboli nejkratší vzdálenost uzlu $s$ od uzlu $v$, $s, v \in V$, $\pi(s, v) \in \Pi$ definuje funkce $c : P^n \times V \times V \to \mathbb{R}_0^+$.

$$c(P^n, s, v) = min \begin{cases} c(P^n, s, v) \\ c(P^n, s, m) + \omega(P^n, (m, v)), \end{cases}$$

kde $m \in V$, $\pi(s, m) \in \Pi$ a $(m, v) \in E$.

Nejkratší cesta grafu $\pi'(u, v)$ je taková posloupnost hran $\pi'(u, v) = \langle (u_i, u_{i+1}), \ldots, (u_{t-1}, u_t) \rangle$, pro kterou platí $u, v \in V$, $t \leq |V|$, $\nexists u_i = u_j$, kde $i \neq j$ pro $i, j \in \langle 1, 2, \ldots t \rangle$, $u_i = u, u_t = v$, $\sum_{i=1}^{t-1} \omega(P^n, (u_i, u_{i+1})) = c(P^n, u, v)$.

Řešením problému dojezdu $C$ je množina $T$, která obsahuje uzly, které leží v maximální vzdálenosti $M$ od počátečního uzlu $s$. $T$ se skládá ze dvou množin $T_v$ a $T_m$. $T_v$ obsahuje všechny uzly grafu $G$, které patří do množiny uzlů $V$ a leží v maximální vzdálenosti $M$ od $s$. Množina $T_m$ obsahuje uzly, které nepatří do množiny uzlů $V$, ale leží na jedné z hran $G$ a jejich maximální vzdálenost od počátečního uzlu $s$ je $M$.

Tehdy množina $T$ je tvořena následovně:

$$T = T_v \cup T_m \tag{4.4}$$

$$T_v = \{m \mid c(P^n, s, m) = M\}, \forall m \in V \tag{4.5}$$

$$T_m = \big\{m \mid m \notin V, m \in \{u, v\}, \exists u \in \pi'(s, v), \\ c(P^n, u) < M \wedge c(P^n, v) > M \wedge c(P^n, u) + \omega(P^n, u, m) = M\big\}, \forall v \in V \tag{4.6}$$

### 4.5.1 Nalezení dosažitelných bodů na přímce

Abychom byli schopni co nejpřesněji vypočítat dojezd elektrického kola, musíme být schopni přidávat do množiny dosažitelných uzlů i uzly, které leží na hranách grafu $G$, ale nepatří do množiny $V$. Uzel $m$ patří do množiny dosažitelných uzlů $T$, pokud $m \notin V$, $m \in (u, v)$, $\exists u \in \pi'(s, v), c(P^n, u) < M \wedge c(P^n, v) > M \wedge c(P^n, u) + \omega(P^n, (u, m)) = M$.

Pro určení bodu $m$ je zapotřebí znát jeho zeměpisné souřadnice.

Nejdříve musíme vypočítat vzdálenost $dist(u, w)[m]$ od uzlu $u$ do uzlu $w$, která určuje, jakou vzdálenost bude schopno kolo projet od uzlu $u$, než dojde k úplnému vybití baterie. Dále vypočítáme spotřebu energie na jeden metr $e[Wh]$, $e = \frac{p(s,v)-p(s,u)}{dist(u,v)}$, pak

$$dist(u, w) = \frac{M - p(s, u)}{e}.$$

Pro zjištění zeměpisné šířky a délky bodu $m$ je ještě nutné vypočítat skutečný zeměpisný směrník (angl. true bearing), neboli okamžitý směr k uzlu $v$ z uzlu $u$.

$$\theta_{tr} = arctan2\left(\sin(\lambda_v - \lambda_u) \cdot \cos\varphi_v, \cos\varphi_u \cdot \sin\varphi_v - \sin\varphi_u \cdot \cos\varphi_v \cdot \cos(\lambda_v - \lambda_u)\right),$$

$$\theta_{cmp} = \frac{\left(\frac{\theta_{tr} \cdot 180}{\pi} + 360\right) \bmod 360}{180 \cdot \pi},$$

kde $\theta_{tr}$ je skutečný zeměpisný směrník v radiánech, $\theta_{cmp}$ je zeměpisný směrník v radiánech normalizovaný do kompasového, $\varphi_u$ je zeměpisná šířka uzlu $u$ v radiánech, $\lambda_u$ je zeměpisná délka uzlu $u$ v radiánech, $\varphi_v$ je zeměpisná šířka uzlu $v$ v radiánech a $\lambda_v$ je zeměpisná délka uzlu $v$ v radiánech.

Zeměpisné souřadnice v radiánech pro uzel $w$ podél velké kružnice (angl. great-circle) vypočítáme dle vzorce:

$$\varphi_w = \arcsin\left(\sin\varphi_u \cdot \cos\frac{dist(u,w)}{R} + \cos\varphi_u \cdot \sin\frac{dist(u,w)}{R} \cdot \cos\theta_{cmp}\right),$$

$$\lambda_w = \lambda_u + arctan2\left(\sin\theta_{cmp} \cdot \sin\frac{dist(u,w)}{R} + \cos\varphi_u, \cos\frac{dist(u,w)}{R} - \sin\varphi_u \cdot \sin\varphi_w\right),$$

kde $R = 6378137[m]$ je velká poloosa Země.

# Kapitola 5

# Návrh řešení

Kapitola obsahuje popis řešení problému dojezdu kola, zejména proces vytvoření cyklistického grafu z geografických dat a algoritmy k nalezení dosažitelných uzlů grafu.

## 5.1 Řešení problému dojezdu pedeleku

V předchozí kapitole 4.5 jsme matematický definovali problém dojezdu pedeleku $C = (G, M, P^n, s)$, také jsme podrobně popsali, že řešením problému je množina $T$, která obsahuje uzly, které leží ve vzdálenosti $M$ od počátečního uzlu $s$, kde $M$ je maximální množství energie, kterou je kolo schopno poskytnout. Abychom byli schopni vyřešit problém $C$, jak je popsán v předchozí kapitole, nejprve musíme popsat algoritmus nalezení počátečního uzlu $s$ v grafu $G$ a poté popsat algoritmus implementovány na základě matematické definice problému $C$.

### 5.1.1 Určení počátečního uzlu dle zeměpisných souřadnic

Dle funkčních požadavků popsaných v podkapitole 3.1.1 umožní aplikace uživateli určit počáteční uzel $s$ cyklistického grafu $G$ na základě zeměpisných souřadnic bodu vybraného na mapě. Problém nalezení uzlu $s$ na základě zeměpisných souřadnic označíme za problém nejbližšího souseda, který spočívá v nalezení množiny uzlů umístěných v metrickém prostoru uzlů blízké k danému uzlu dle funkce blízkosti, která tento prostor definuje. V našem případě funkci blízkosti zastupuje funkce, která na základě zeměpisných souřadnic dvou uzlů a jejich převýšení určí reálnou vzdálenost mezi nimi.

### 5.1.2 Hledání dosažitelných uzlů

Vzdálenost mezi uzly je číslo reprezentující nejmenší množství energie, kterou musí poskytnout kolo pro zdolání cesty mezi těmito uzly. Vzdálenost mezi $s$ a $v \in V$ nalezneme pomocí funkce $c : P^n \times V \times V \to \mathbb{R}_0^+$.

Pokud budeme považovat vzdálenost mezi dvěma uzly za hodnocení hrany, pak problém nalezení vzdálenosti lze převést na problém nalezení nejkratší cesty. Vzhledem k tomu, že naším úkolem je najít dojezd z počátečního uzlu $s$, musíme najít nejkratší cesty ke všem uzlům grafu. Protože víme, že vzdálenost mezi dvěma uzly nemůže být záporným číslem, pro

řešení problému můžeme použít Dijkstrův algoritmus, který upravíme na hledání nejkratších cest do všech uzlů.

### 5.1.3 Zobrazení dojezdu pomocí mnohoúhelníku

Pro zobrazení dojezdu pomocí mnohoúhelníku je nutné najít co nejkratší uzavřenou křivku, neboli konvexní obal množiny, uvnitř které leží všechny dosažitelné uzly z množiny $T$. Dle [15] množina $M$ se nazývá konvexní, jestliže úsečka spojující libovolné dva její body je částí této množiny. Konvexní obal množiny je nejmenší konvexní množina, která množinu $M$ obsahuje.

K nalezení konvexního obalu množiny $T$ použijeme Grahamův algoritmus [1].

## 5.2 Vytvoření grafu cyklistické sítě

Graf cyklistické sítě popsaný v podkapitole 4.1 bude sestaven na základě geodat získaných z projektu *OpenStreetMap*[2]. Vzhledem k výpočetní a paměťové náročnosti jejich zpracování bude tvorba grafu probíhat mimo mobilní aplikaci.

V této podkapitole se nejdříve zaměříme na projekt *OpenStreetMap* a formát, který používá k popisu geodat. Následně si popíšeme kroky, které učiníme ke zjednodušení dat. Na konci se podíváme na proces vytvoření cyklistického grafu a popíšeme si proces převodu obecného nesouvislého ohodnoceného grafu na graf cyklistický.

### 5.2.1 Projekt *OpenStreetMap*

*OpenStreetMap* (dále *OSM*) je projekt, jehož cílem je vytváření a distribuce volně dostupných geografických dat. Přispěvatelé využívají letecké snímky, GPS přístroje a klasické mapy, aby ověřili, že *OSM* geodata jsou přesná a aktuální. OSM data se můžou používat k libovolným účelům, pokud je uvedeno autorství OpenStreetMap a jeho přispěvatelů. Dle stránky [16] *OSM* definuje vlastní formát „.osm“ pro ukládání geodat založený na XML. Pro popis datového modelu se v OSM používají čtyři základní prvky:

- *bod (angl. node)* - popisuje body lokalizované souřadnicemi
- *cesta (angl. way)* - posloupnost uzlů popisující polygon či linií
- *relace (angl. relation)* - popisuje vztahy mezi prvky

Ke každému datovému prvků lze přidat značku (angl. tag), která popisuje vlastnosti prvku nebo sadu změn provedených nad ním. Značka se skládá z *klíče* (angl. key) a *hodnoty* (angl. value), v *OSM* má tvar *klíč=hodnota*.

[1] <http://asishm.myweb.cs.uwindsor.ca/cs557/F08/handouts/lec2.pdf>
[2] <http://www.openstreetmap.org>

### 5.2.2 Zpracování geodat

Základní geodata konkrétní oblasti získáme z webového rozhraní projektu *OSM*. Pomocí knihovny *Osmosis*[3] z nich vyřízneme požadovaný region.

V dalším kroku je zapotřebí odstranit z geodat regionu údaje o cestách nevhodných pro cyklisty nebo o bodech, které nejsou součástí prvků typu *cesta*. V případě prvku *cesta* značka s klíčem *highway* popisuje libovolný druh silnice, ulice či cesty. Hodnota klíče *highway* popisuje důležitost cesty v silniční síti. Po prozkoumání nejpoužívanějších hodnot pro klíč *highway* a po analýze cest v Praze jsme dospěli k závěru, že pro sestavení cyklistického grafu je nutné zachovat cesty uvedené níže. Typy cest jsou definované dle [17].

- *primary* Hlavní silniční tahy.
- *secondary* Silniční tahy, které spravují kraje. Mají začlenění jako silnice 2. třídy.
- *tertiary* Silniční tahy, které spravují kraje. Mají začlenění jako silnice 3. třídy.
- *motorway_link* Nájezdy a sjezdy příslušné k dálnicím.
- *trunk_link*, *primary_link*, *secondary_link*, *tertiary_link* Přípojné rampy z nebo na komunikaci.
- *unclassified* Silnice s pravidelným provozem podružného významu s pevným povrchem, mimo administrativní zatřídění.
- *residential* Místní komunikace v obci, určené pro přístup k nemovitostem, nejsou-li zatříděny do vyšších tříd silnic.
- *living_street* Obytná zóna, označená obdélníkovou modrou značkou s domem a dětmi.
- *track* Lesní a polní cesty, obvykle průjezdné.
- *pedestrian* Pěší zóny a jiné zpevněné plochy určené převážně nebo jen pro chodce.
- *cycleway* Cyklostezka.
- *path* Stezka bez konkrétního určení. Pro motorová vozidla a techniku je cesta fyzickým provedením nepoužitelná.

Odfiltrujeme také všechny prvky, které reprezentují oblasti s omezeným pohybem nebo přímo zakazují pohyb na kole. Odfiltrujeme tedy dálnice, při popisu kterých se používají značky *motorroad=yes* a *highway!=*_link*. Z geodat také odstraníme relace, které mají značku s klíčem *route*, ale její hodnota není *bicycle*. K odstranění přebytečných uzlů a cest použijeme aplikací Osmfilter[4].

Každý bod v OSM geodatech je reprezentován unikátním identifikátorem a zeměpisnou šířkou a délkou. Tyto údaje ale jsou nedostačující pro výpočet sklonu úseku, který je důležitou součástí výpočtu dojezdu elektrického kola dle funkčních požadavků popsaných v kapitole 3.1.1.

[3] <http://wiki.openstreetmap.org/wiki/Osmosis>
[4] <http://wiki.openstreetmap.org/wiki/Osmfilter>

Do geodat přidáme údaje o nadmořské výšce každého bodu pomocí rozšíření knihovny *Osmosis - SRTMPlugin*[5]. Rozšíření využívá veřejně dostupných dat z projektu *ShuttleRadarTopographyMissio* Ve výsledku tak každý bod na mapě obsahuje značku s klíčem *height* a informaci o nadmořské výšce.

### 5.2.3 Vytvoření cyklistického grafu

Cyklistický graf bude vytvořen na základě zpracovaných geodat. Pomocí Java frameworku *Osmonaut*[7], který při zpracování geodat vrací kompletní *OSM* instance, postupně zpracujeme všechny prvky typu *cesta*. Pro každý bod cesty, který neexistuje v grafu, vytvoříme nový uzel. Přidáme do grafu hrany pro všechny sousedící body a vypočítáme délku hrany jako vzdálenost mezi uzly.

Po skončení zpracování geodat obdržíme obecný nesouvislý ohodnocený graf, který je zapotřebí převést do grafu cyklistického. K vytvoření cyklistického grafu je nutné nejprve odstranit z obecného grafu následující data:

- Hrany, které vychází a končí ve stejném uzlů.
- Uzly se stupněm dva. Uzly do kterých vstupují pouze dvě hrany jsou v cyklistickém grafu přebytečné. Chceme-li graf co nejvíce zjednodušit, a zrychlit tak hledání dojezdu, odstraníme tyto uzly z grafu. Hrany, které uzel se stupněm dva propojuje, sjednotíme do jedné. Délkou sjednocené hrany bude suma délek obou hran.
- Všechny komponenty souvislosti kromě té největší. Dle podkapitoly 4.1 cyklistický graf musí být souvislý. Pro splnění daného požadavku zachováme v grafu pouze jednu komponentu souvislosti a to jednu s největším počtem vrcholů.

V dalším kroku ze zjednodušeného souvislého ohodnoceného grafu vytvoříme cyklistický graf, který bude obsahovat množinu uzlů a množinu hran pro každý uzel. Výsledkem procesu vytvoření grafu bude serializovany graf cyklistické sítě.

## 5.3 Mobilní aplikace

Podle obecných požadavků popsaných v podkapitole 3.1.1 aplikace pro zobrazení dojezdu bude určena pro mobilní zařízení s operačním systémem Android ve verzi 4.1 a vyšší. Graf cyklistické sítě bude vytvořen mimo mobilní aplikaci. Ta bude obsahovat dva serializované grafy cyklistické sítě, jeden mapující Prahu a Středočeský kraj, druhý zahrnující centrální část Rakouska. Uživatel bude mít na výběr, který z těchto dvou region chce použít. Po spuštění aplikace dojde k deserializaci regionu dle posledního výběru či podle výchozího nastavení. Součástí aplikace budou algoritmy popsané v podkapitole 5.1. Po kliknutí na mapu a zmáčknutí tlačítka „Find range“ dojde k vyhledání nejbližšího uzlu v grafu na základě zeměpisných souřadnic, jak je popsáno v podkapitole 5.1.1. Po nalezení počátečního uzlu

[5] <https://github.com/locked-fg/osmosis-srtm-plugin>
[6] <https://www2.jpl.nasa.gov/srtm/>
[7] <https://github.com/MorbZ/OSMonaut>

bude spuštěn proces hledání množiny dosažitelných uzlů, následně bude nalezen konvexní obal této množiny a výsledek bude zobrazen uživateli.

Uživatelské rozhraní mobilní aplikace bude vycházet z prototypu popsaného v podkapitole 3.4.

# Kapitola 6

# Implementace

Tato kapitola obsahuje popis aplikace *MapConverter* určené k vytvoření grafu cyklistické sítě, popis algoritmů použitých k vyřešení problému nalezení dojezdu a popis implementace mobilní aplikace *PedelecRange*. Na konci kapitoly je popsán proces ověření správnosti zpracování geodat a jednotkové testy komponenty *calculators* aplikace *MapConverter*.

## 6.1 Zpracovaní geodat a vytvoření grafu

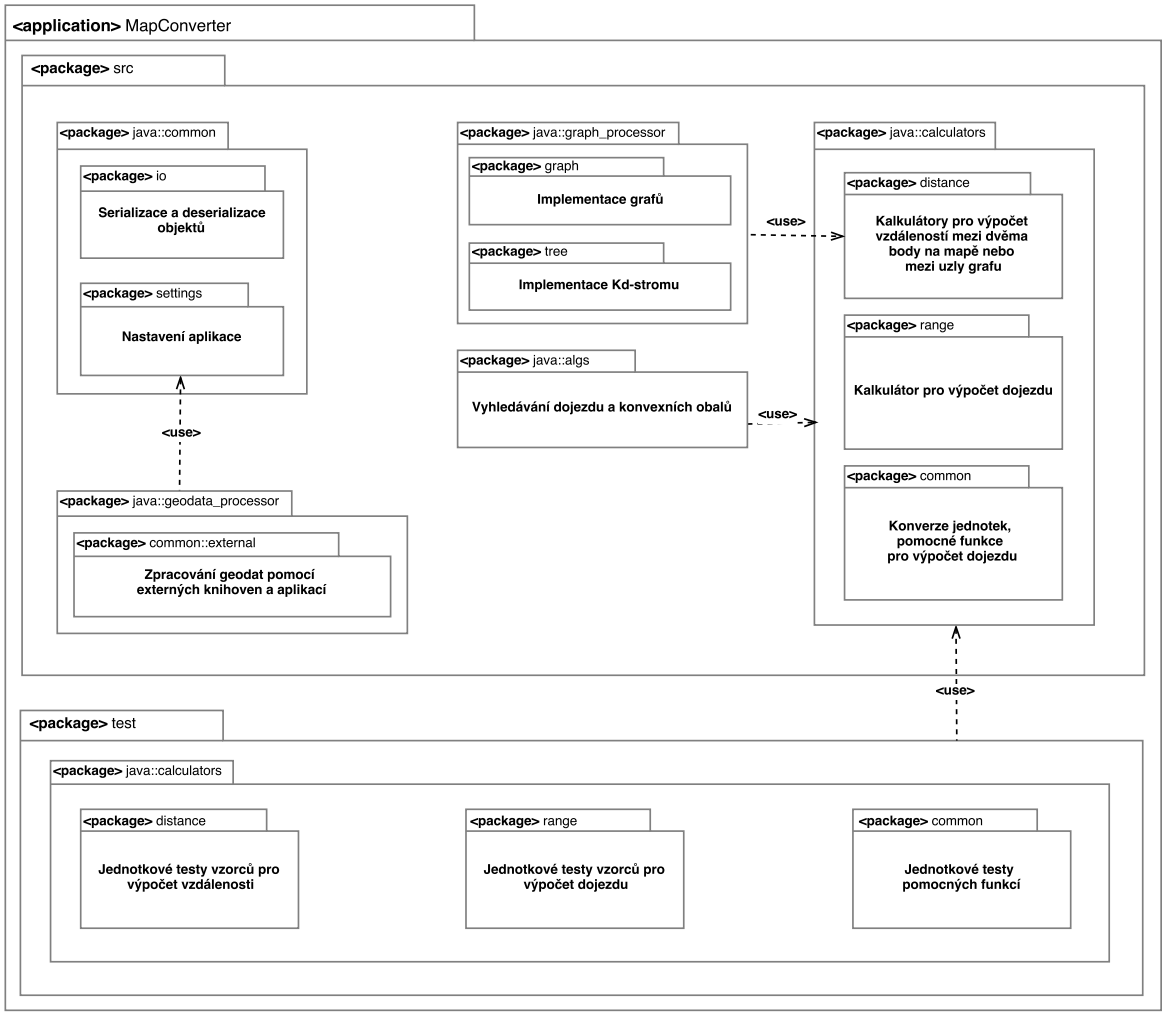
Pro zpracování geodat a sestavení cyklistického grafu dle postupů popsaných v podkapitole 5.2 byla vytvořena desktopová aplikace *MapConverter*. Za proces zpracování geodat, jak je popsán v podkapitole 5.2.2, odpovídá třída *OsmGeodataProcessor*. Za vytvoření grafu je zodpovědná třída *GraphProcessor*.

### 6.1.1 Třída *OsmGeodataProcessor*

Třída *OsmGeodataProcessor* s využitím knihovny *Osmosis* se zabudovaným pluginem *SRTM* a ve spolupráci s aplikací *OSMFilter* provede zpracování geodat pro konkrétní oblast dle nastavení definovaných ve třídě *MapConverterSettings*.

### 6.1.2 Třída *GraphProcessor*

Ke zpracování geodat, jak již bylo řečeno v podkapitole 5.2.3, byl vybrán framework *Osmonaut*. Třída *GeneralGraphCreator* implementuje metody *needsEntity*() a *foundEntity*() z rozhraní *IOsmonautReceiver*. Metoda *needsEntity*() popisuje prvky ke zpracování, t.j. prvky typu cesta (angl. way). Metoda *foundEntity*() odpovídá za zpracování vybraných prvků, v našem případě tato metoda přidává uzly a hrany se vzdálenostmi mezi uzly do obecného grafu (instance třídy *GeneralGraph*). Pro zjednodušení obecného grafu z něj nejprve odstraníme smyčky a uzly se stupněm dva. Poté nalezneme všechny komponenty souvislosti a zachováme pouze tu s největším počtem uzlů, zbytek odstraníme. Takto získáme souvislý obecný graf. K nalezení komponent souvislosti použijeme prohledávání do hloubky.



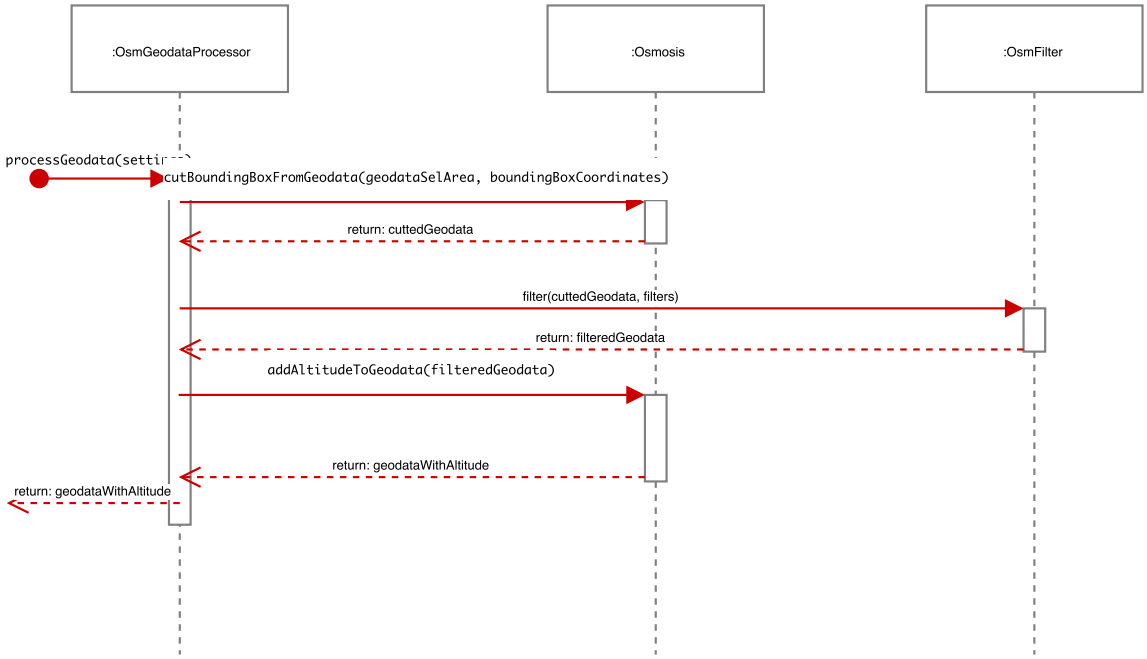

Obrázek 6.1: Diagram balíčků aplikace *MapConverter*

Cyklistický graf je reprezentován třídou *BicycleGraph*. Při jejím návrhu byl brán zřetel především na paměťovou náročnost. Graf je serializován pomocí frameworku *Kryo*[1].

## 6.2 Výpočet dojezdu

Algoritmy popsané v podkapitole 5.1 byly implementovány v aplikaci *MapConverter* z důvodu usnadnění jejich testování. Jejich implementace byla následně použita v mobilní aplikaci *PedelecRange*.

[1] <https://github.com/EsotericSoftware/kryo>




Obrázek 6.2: Proces zpracování geodat třídou *OsmGeodataProcessor*.

### 6.2.1 Hledání dosažitelných uzlů

Třída *Dijkstra* odpovídá za hledání dosažitelných uzlů grafu na základě maximální energie, kterou je schopno kolo poskytnout. Algoritmus vychází z implementace Dijkstrova algoritmu, ale je jeho modifikací. Průběh algoritmu lze popsat následovně:

```
function findAchievableVertices
Vstup: V množina uzlů cyklistického grafu V ∈ G,
  E množina hran cyklistického grafu E ∈ G,
  V_s počáteční uzel V_start ∈ V,
  A_battery kapacita baterie A_battery ∈ P,
  V_battery napětí baterie V_battery ∈ P,
  v_north nejsevernější uzel grafu se zaokrouhlenými koordináty,
  v_west nejzápadnější uzel se zaokrouhlenými koordináty,
  v_south nejjižnější uzel grafu se zaokrouhlenými koordináty,
  v_east nejvýchodnější uzel grafu se zaokrouhlenými koordináty.
Výstup: mapa T obsahující množiny dosažitelných uzlů dle oblastí dosažitelnosti.

e_total := A_battery · V_battery
e_50% := e_total · 0.5
e_75% := e_total · 0.75

K := maximální uvažované množství energie
K := e_total + e_total · 0.1

T := prázdná mapa
T_50% := prázdná množina pro ukládání uzlů dosažitelných s množstvím energie E_50%
T_75% := prázdná množina pro ukládání uzlů dosažitelných s množstvím energie E_75%
T_100% := prázdná množina pro ukládání uzlů dosažitelných s množstvím energie E_100%
```

```
24
25 P := množina vstupních parametrů od uživatele
26 Q := prázdná prioritní fronta pro ukládání nenavštívených uzlů
27 W := prázdná mapa pro ukládání hodnoty reprezentující nejmenší množství energie
        potřebné pro dosažení uzlu z V_start
28 A := prázdná mapa pro ukládání navštívených uzlů
29 S := prázdná mapa pro ukládání hran, které vedou ke konečnému uzlu v nejkratší cestě
30
31 Q.insert(V_start)
32 W.insert(V_start, 0)
33
34 while Q není prázdná do
35   V_src := Q.poll()
36   E_src := W.get(V_src)
37   A.insert(V_src, true)
38
39   if W_src > M then
40     break
41   end
42
43   if uzel V_src má pouze jednu hranu and
44   (V_src.lat > v_north.lat or V_src.lat ≤ v_south.lat or
45    V_src.lon ≤ v_west.lon or V_src.lon ≥ v_east.lon)
46     then
47
48     if E_src ≤ E_50%range then
49       savePoint(T_50%, V_src)
50     end
51     if E_src ≤ E_75%range then
52       savePoint(T_75%, V_src)
53     end
54     savePoint(T_100%, V_src)
55     continue
56   end
57
58   if S.containsKey(V_src) and E_src ≥ E_50%range then
59     E_lastFromShortestPath = S.get(V_src)
60     V_lastFromShortestPath = E_lastFromShortestPath.getDestination()
61     savePointFromEdge(V_lastFromShortestPath, V_src, E_lastFromShortestPath)
62     S.remove(V_src)
63   end
64
65   foreach hranu e uzlu V_src do
66     V_dest := e.getDestination()
67     if A.containsKey(V_dest) then
68       continue
69     end
70     W_dest := W_src + calculatePowerConsumptionForDistance(V_src, V_dest, P)
71     if not W.containsKey(V_dest) or W_dest < W.get(V_dest) then
72       W.insert(V_dest, W_dest)
73       S.insert(V_dest, e)
74     end
75     if not A.containsKey(V_dest) and not Q.contains(V_dest) then
76       Q.insert(V_dest)
77     end
78   end
```

```
79 end
80 T.insert("50% of energy", T_50%)
81 T.insert("75% of energy", T_75%)
82 T.insert("100% of energy", T_100%)
83 return T
```

Po spuštění algoritmu se nejdříve vypočítají tři hodnoty energie odpovídající 50 %, 75 % a 100 % spotřebované energie. Následně je definována hodnota $K$, která se rovná maximálnímu množství energie, které je kolo schopné poskytnout, zvýšené o 10 %. Definice hodnoty $K$ je velmi důležitá, neboť slouží jako terminační hodnota algoritmu v momentě, kdy cena nejkratší cesty do uzlu $u$, neboli množství energie potřebné pro jeho dosažení, bude větší než tato hodnota. Seznamy $T_{50\%}$, $T_{75\%}$ a $T_{100\%}$ slouží k ukládání dosažitelných bodů dle množství potřebné energie.

Metoda *savePointFromEdge*() odpovídá za ukládání uzlu grafu (metoda *savePoint*()) nebo dopočítaného uzlu ležícího na hraně, která spojuje dosažitelnou a nedosažitelnou oblast (metoda *calculateAndSavePoint*).

```
1  function savePointFromEdge
2  Vstup: V_src uzel na začátku hrany (uzel blíže k počátečnímu uzlu),
3   V_dest uzel na konci hrany,
4   E hrana z uzlu V_src do V_dest.
5
6  W_src = W.get(V_src)
7  W_dest = W.get(V_dest)
8
9  if W_src < E_100% and W_dest ≥ E_100%range then
10   calculateAndSavePoint(T_100%, V_src, V_dest, E)
11 end
12
13 if W_src < E_75% and W_dest ≥ E_75%range then
14   calculateAndSavePoint(T_75%, V_src, V_dest, E)
15 end
16
17 if W_src < E_50% and W_dest ≥ E_50%range then
18   calculateAndSavePoint(T_50%, V_src, V_dest, E)
19 end
```

### 6.2.2 Zobrazení dojezdu pomocí mnohoúhelníku

Pro zobrazení dojezdu pomocí mnohoúhelníku je nutné najít konvexní obal množiny dosažitelných uzlů. Jak již bylo popsáno v předchozí kapitole, k nalezení konvexního obalu byl použit Grahamův algoritmus.

Implementace algoritmu ve třídě *GrahamScan* vychází z implementace popsané na stránce volně dostupného projektu *GrahamScan*[2]. Konvexní obal najdeme pro každou ze tří množin (50 %, 75 % a 100 %) dosažitelných uzlů pomocí metody $findConvexHulls$.

Do množiny dosažitelných uzlů $T_k$ s určitým procentem spotřeby baterie $k$ spadají pouze takové uzly, pro které se rovná nejmenší množství energie potřebné pro jejich dosažení množství energie, kterou je kolo schopno dodat při maximální procentuální spotřebě $k$. Z toho vyplývá, že kvůli omezení grafu na určitý region (např. Praha) může dojít k situaci, kdy

[2] <https://github.com/bkiers/GrahamScan/blob/master/src/main/cg/GrahamScan.java>

žádný nebo některé potenciálně dosažitelné uzly nebudou nalezeny algoritmem. V takovém případě bude do dané množiny zařazen hraniční uzel regionu.

```
function findConvexHulls
Vstup: mapa T obsahující množiny dosažitelných uzlů dle oblastí dosažitelnosti.
Výstup: mapa C obsahující konvexní obaly pro každou množinu dosažitelných uzlů dle oblastí dosažitelnosti.

C := prázdná mapa
C_50% := konvexní obal množiny uzlů dosažitelných s 50% nabitím baterie
C_75% := konvexní obal množiny uzlů dosažitelných s 75% nabitím baterie
C_100% := konvexní obal množiny uzlů dosažitelných s 100% nabitím baterie

C_50% := findConvexHull(T_50%)
T_75%.union(C_50%)

C_75% := findConvexHull(T_75%)
T_100%.union(C_75%)

C_100% := findConvexHull(T_100%)

C.insert("50% of energy", C_50%)
C.insert("75% of energy", C_75%)
C.insert("100% of energy", C_100%)
return C
```

## 6.3 Aplikace *PedelecRange*

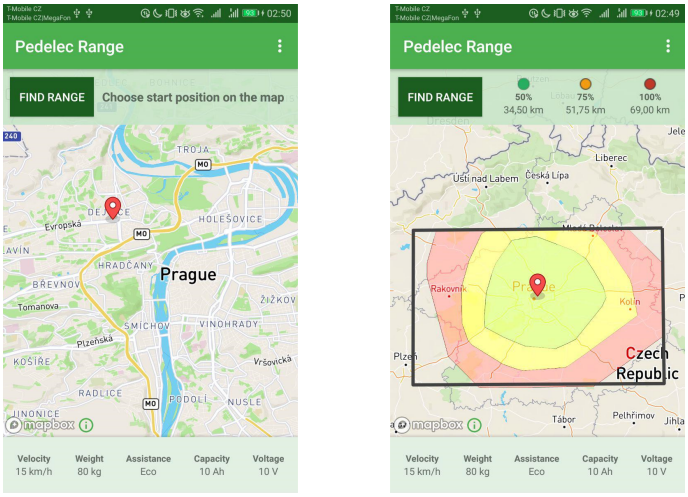
Mobilní aplikace *PedelecRange* byla implementována pro mobilní zařízení se systémem Android s verzí 4.1 a vyšší. Grafy cyklistických sítí byly vytvořeny v aplikaci *MapConverter* a následně přesunuty do mobilní aplikace v podobě dvou serializovanych objektů třídy *BicycleGraph*. Dále byly z aplikace *MapConverter* přeneseny balíčky *algs* a *calculators* (dle diagramu 6.1) obsahující algoritmy potřebné k vyhledávání dojezdu. Jejich implementace byla podrobně popsána v podkapitole 5.1.

Pro spuštění paměťově a časově náročných operací, jako jsou deserializace grafu a hledání konvexních obalů, byly implementovány třídy *DeserializeGraphTask* a *FindRangeTask*. Tyto třídy rozšiřují abstraktní třídu platformy Android *AsyncTask*, která umožňuje provádění výpočetních operací na pozadí bez blokování hlavního vlákna.

K ukládání nastavení aplikace využíváme třídu *SharedPreferences*.

Uživatelské rozhraní aplikace *PedelecRange* vychází z prototypu popsaného v podkapitole 3.4. K zobrazení mapy se použil framework *MapboxMap*[3], který obstarává nejen proces vykreslování mapy, ale i zobrazení polygonů či značek na mapě. K vytvoření posuvného panelu s nastavením byla použita knihovna *AndroidSlidingUpPanel*[4].

[3] <https://www.mapbox.com/maps/>
[4] <https://github.com/umano/AndroidSlidingUpPanel>




Obrázek 6.3: Grafické rozhraní aplikace *PedelecRange*.

## 6.4 Metodiky testování

Jak již bylo řečeno v předchozí kapitole 6, algoritmy pro vyhledávání dojezdu byly implementovány v aplikaci *MapConverter* a následně použity v mobilní aplikaci *PedelecRange* z důvodu usnadnění jejich testování.

Testování každé části aplikace *MapConverter* vyžadovalo odlišný přístup. Při ověření funkčnosti procesu zpracování geodat a vytvoření grafu jsme použili manuální testování. K ověření funkčnosti a správnosti výpočtů, například vzdálenosti mezi dvěma uzly, jsme použili jednotkové testy.

### 6.4.1 Ověření správnosti zpracování geodat a vytvoření grafu

Ověření správnosti algoritmu pro zpracování geodat bylo složitou úlohou vzhledem k jejich velikosti a provázanosti. Z tohoto důvodu se ověření správnosti procesu zpracování geodat a vytváření grafů provádělo manuálně náhodným ověřováním přítomnosti či nepřítomnosti požadovaných či nepožadovaných cest. Ke znázornění obecného grafu, instance třídy *GeneralGraph*, a cyklistického grafu *BicycleGraph* byla použita jejich konverze do formátu *GeoJson*[5]. Analýza geodat se prováděla v programu *QGIS*[6] za použití pluginu *QuickOSM*[7].

### 6.4.2 Testování jednotek

Součástí aplikace *MapConverter* jsou také jednotkové testy, které ověřují správnost a funkčnost balíčku *calculators*. K implementaci jednotkových testů byl použit framework *Junit* ve verzi 4.12[8].

[5] <http://geojson.org/>
[6] <https://www.qgis.org/en/site/>
[7] <https://plugins.qgis.org/plugins/QuickOSM/>
[8] <http://junit.org/junit4/>

Součástí balíčku *test* jsou jednotkové testy vzorců pro výpočet vzdáleností, množství spotřebované energie a testy pomocných funkcí, jako jsou například konverze mezi jednotkami či výpočet stoupání svahu.

Jako referenční hodnoty pro ověření správnosti výpočtu byly použity volně dostupné údaje z mapového portálu Mapy.cz[9] či hodnoty vypočítané pomocí kalkulaček dostupných na webové stránce Movable Type Scripts[10].

[9] <https://mapy.cz/>
[10] <https://www.movable-type.co.uk/scripts/latlong.html>

# Kapitola 7

# Vyhodnocení

Tato kapitola obsahuje kvalitativní vyhodnocení aplikace *PedelecRange*, vyhodnocení správnosti výpočtu dojezdu a kvantitavni vyhodnocení, které se zaměřuje na vyhodnocení časové náročnosti algoritmů použitých k řešení problému nalezení dojezdu.

## 7.1 Správnost výpočtu dojezdu

V ideálním případě by k ověření správnosti výpočtu dojezdu měla být provedena měření v terénu, ta ale vzhledem k jejich náročnosti nejsou součástí této práce.

Zde pouze provedeme orientační porovnání výsledné aplikace a aplikace *eBike range assistant*[1] popsané v odstavci 2.2.3.

Ovšem pro účely testování má tato kalkulačka několik nevýhod. Jednou z nejzásadnějších je absence možnosti nastavit kapacitu baterie. Z toho důvodu muela být její hodnota v průběhu testování zafixována. Další nevýhodou je to, že parametrům jako stupeň podpory při šlapání a profil terénu nelze nastavit libovolnou číselnou hodnotu a implicitní hodnoty jsou reprezentovány pouze ilustračním obrázkem, není tedy úplně jasné, jak přesně jsou tyto parametry nastaveny. Na druhou stranu se nám po podrobnější analýze aplikace podařilo získat hodnoty pro jednotlivé stupně převýšení.

Kvůli tomu, že aplikace *eBike range assistant* vyžívá při výpočtu více vstupních parametrů než vzorec pro výpočet dojezdu popsaný v této práci, musíme některé vstupní parametry zafixovat:

- **Rychlost šlapání [rpm]:** 60.
- **Typ motoru:** Active Line.
- **Typ řazení:** Derailleur systém.
- **Typ pneumatiky:** pneumatika do města.
- **Typ kola:** sportovní jízdní kolo.

[1] <https://www.bosch-ebike.com/en/service/range-assistant>

- **Zemní povrch:** asfalt.
- **Počet zastávek v průběhu cesty:** nízký
- **Větrné podmínky:** bezvětrné počasí.

Parametry jako průměrná rychlost, stupeň podpory při šlapání, celková hmotnost kola a cyklisty a profil terénu se v průběhu testování měnily. Vstupní parametry a jejich testované hodnoty jsou:

- **Průměrná rychlost [km/h]:** 10, 20, 25 (rychlost 25 km/h je maximální povolená rychlost s asistencí pro elektrická kola typu pedelek).
- **Stupeň podpory při šlapání:** ECO (40%), TOUR (100%), SPORT (150%), TURBO (250%).
- **Celková váha [kg]:** 70, 90, 150 (váhá 150 kg je maximální váha, kterou lze vybrat v aplikaci) .
- **Profil terénu:** FLAT (stoupání 2 %), SOME INCLINES (stoupání 4 %), UPLANDS (stoupání 7.5 %).

Třída $TestCorrectnessOfRangeCalculator$ implementovaná v projektu $MapConverter$ obsahuje 216 testovacích případů, které pokrývají všechny kombinace vstupních parametrů. V průběhu každého testovacího případu byl vypočítán rozdíl mezi očekávaným dojezdem kola (dojezd kola vypočítaný aplikací eBike range assistant) a vypočítanou hodnotou (dojezd kola vypočítaný naší aplikací) dle vzorečku $diff = expectedRange - calculatedRange$. Tato hodnota se uložila jak pro každou proměnnou ze seznamu výše, tak i pro všechny testovací případy. V průběhu každého testu byl také vypočítán procentuální rozdíl mezi dojezdy, $percentageError = 100 \cdot (expectedRange - calculatedRange)/expectedRange$. Tato hodnota se také uložila pro každou proměnnou ze seznamu.

Průměrný rozdíl mezi očekávaným a vypočítaným dojezdem elektrického kola pro všechny testovací případy byl 26,05 km, průměrné rozdíly pro jednotlivé hodnoty proměnných jsou uvedeny v tabulce 7.1. Průměrný procentuální rozdíl mezi očekávaným a vypočítaným dojezdem elektrického kola pro všechny testovací případy byl 34,84 %.

## 7.2 Časová náročnost aplikace

Dále jsme změřily časovou náročnost nejnáročnějších operací: deserializace grafu, hledání uzlu v grafu na základě údajů o zeměpisné šířce a délce, hledání dosažitelných uzlů a nalezení konvexních obalů. Pro každý testovací případ jsme provedli 11 měření. K testování jsme použili zařízení Huawei Nova s charakteristikami: Android 7.0, RAM 3 GB, uživatelská paměť 32 GB, počet jader procesoru 7.

Testovací případy:

- **Testovací případ 1.** Vstupní parametry: napětí baterie 24 V, kapacita baterie 4 Ah, podpora při šlapání v režimu Eco, průměrná rychlost kola 15 km/h, celková hmotnost kola a cyklisty 90 kg, region Praha a Středočeský kraj.

| **Proměnná** | **Průměrný rozdíl v kilometrech** | **Průměrný rozdíl v procentech** |
|---|---|---|
| Průměrná rychlost | | |
| 10 km/h | 25,53 km | 34,72 % |
| 20 km/h | 32,74 km | 38,99 % |
| 25 km/h | 19,88 km | 30,83 % |
| Stupeň podpory | | |
| 40 % | 43,18 km | 35,44 % |
| 100 % | 25,76 km | 36,07 % |
| 150 % | 17,93 km | 32,70 % |
| 250 % | 17,32 km | 35,18 % |
| Celková váha | | |
| 70 kg | 17,19 km | 24,94 % |
| 90 kg | 29,74 km | 34,15 % |
| 150 kg | 31,22 km | 45,45 % |
| Stoupání terénu | | |
| 2 % | 17,15 km | 23,58 % |
| 4 % | 32,71 km | 37,82 % |
| 7,5 % | 28,29 km | 43,14 % |

Tabulka 7.1: Průměrný rozdíl mezi očekávaným a vypočítaným dojezdem kola, procentuální rozdíl mezi dojezdy vzhledem ke každé proměnné.

- **Testovací případ 2.** Vstupní parametry: napětí baterie 24 V, kapacita baterie 4 Ah, podpora při šlapání v režimu Eco, průměrná rychlost kola 15 km/h, celková hmotnost kola a cyklisty 90 kg, region centrální část Rakouska.
- **Testovací případ 3.** Vstupní parametry: napětí baterie 24 V, kapacita baterie 6 Ah, podpora při šlapání v režimu Tour, průměrná rychlost kola 15 km/h, celková hmotnost kola a cyklisty 90 kg, region centrální část Rakouska.
- **Testovací případ 4.** Vstupní parametry: napětí baterie 36 V, kapacita baterie 4 Ah, podpora při šlapání je Sport, průměrná rychlost kola 15 km/h, celková hmotnost kola a cyklisty 90 kg, region Praha a Středočeský kraj.

Deserializace grafu Prahy a Středočeského kraje trvá průměrně 7102 ms, centrálního Rakouská 8129 ms.

Jak je patrné z 7.2 nejnáročnější operací je hledání dosažitelných uzlů pomocí upravené verze Dijkstrova algoritmu. Časová náročnost algoritmu je závislá na množství zpracovaných uzlů a hran, čím je větší, tím jsou operace pomalejší. To také naznačuje tabulka 7.3 zahrnující porovnání průměrné doby trvání Dijkstrova algoritmu s délkou dojezdu, kde je vidět, že vyhledávání dosažitelných uzlů trvá déle pro delší dojezdy.

Z tabulky 7.2 plyne, že průměrná doba hledání dosažitelných uzlů a konvexních obalů se mezi dvěma oblastmi skoro neliší. Časová náročnost nalezení počátečního uzlu a deserializace grafu je o něco vetší v případě regionu centrálního Rakouska z důvodu jeho větší velikosti.

| Testovací případ | Hledání uzlu [ms] | Dijkstrův alg. [ms] | Konvexní obal [ms] |
|---|---|---|---|
| 1 | 1169 | 53724 | 13 |
| 2 | 1442 | 50321 | 17 |
| 3 | 1327 | 41736 | 14 |
| 4 | 1127 | 42215 | 11 |

Tabulka 7.2: Průměrná doba trvání nejnáročnějších operací pro každý testovací případ.

| Testovací případ | Maximální dojezd [m] | Dijkstrův algoritmus [ms] | |
|---|---|---|---|
| 1 | 183 | 60920 | 2984 |
| 2 | 136 | 41736 | 3764 |
| 3 | 61 | 42215 | 3448 |
| 4 | 73 | 62740 | 4030 |

Tabulka 7.3: Porovnání průměrné doby trvání Dijkstrova algoritmu s délkou dojezdu.

## 7.3 Vyhodnocení výsledků testování

Z výsledků testování správnosti výpočtů dojezdu lze usoudit, že na základě našeho vzorce popsáného v 2.3 získáváme hodnoty, které se liší od hodnot získaných z aplikace *eBike range assistant* v průměru o 34,84 %. Rozdíly můžou být způsobené jak nepřesností aplikací, tak i růzností vstupních hodnot. Při testování totiž nebylo možné použít úplně stejná vstupní data pro obě aplikace (např. typ pneumatiky nebo typ motoru). Funkčnost aplikace, která byla vytvořena v rámci této práce, je možné přesně posoudit pouze po provedení testů v reálném prostředí.

Největší rozdíly se objevily u hodnot proměnných, jako jsou např. stupeň podpory při šlapání typu Eco s hodnotou 40 % nebo stoupání do svahu typu SOME INCLINES se stoupáním 4 %.

Měření časové náročnosti aplikace prokázalo, že nejnáročnější operací je hledání dosažitelných uzlů pomocí upravené verze Dijkstrova algoritmu. Jeho časová náročnost je závislá na množství zpracovaných uzlů a hran.

# Kapitola 8

# Závěr

V rámci této diplomové práce se podařilo vytvořit funkční mobilní aplikaci pro výpočet dojezdu elektrického kola typu pedelek dle požadavků stanovených v počáteční fázi projektu. Aplikace na základě dat dodaných uživatelem a informací o terénu spočítá a graficky znázorní maximální dojezd pedeleku při 50%, 75% a 100% spotřebě baterie. Regiony, ve kterých lze aplikaci použít, jsou Praha, Středočeský kraj a centrální část Rakouska.

Při testování aplikace se ukázalo, že její přesnost se od referenční aplikace *Bosch eBike range assistant* liší v průměru o 34,84 %. Pro přesnější vyhodnocení by bylo potřeba provést testování v terénu, které bylo nad rámec této práce.

Do budoucna by šlo aplikaci rozšířit o přesnější vzorec pro výpočet dojezdu, který by zahrnoval další informace o terénu, jakou jsou stav a typ vozovky, počet semaforů v průběhu cesty, povětrnostní podmínky, apod. Dále by šlo uvažovat například proměnlivou rychlost cyklisty v závislosti na úseku cesty, kterým se projíždí.

Tímto můžeme konstatovat, že cíl práce, kterým bylo vytvořit mobilní aplikaci pro výpočet dojezdu pedelku, se podařilo zdárně naplnit.

# Literatura


[1] Dojezd elektrokola. Svět elektokol [online]. c2017 [cit. 2017-06-27]. Dostupné z: http://svet-elektrokola.cz/dojezd-elektrokola/

[2] HRNCIR, Jan, Qing SONG, Pavol ZILECKY, Marcel NEMET a Michal JAKOB. Bicycle route planning with route choice preferences. Proceedings of the Twenty-first European Conference on Artificial Intelligence. 2014, (14), 1149-1154. DOI: 10.3233/978-1-61499-419-0-1149.

[3] BAYER, Tomáš. Matematické metody v kartografii [online]. In: . [cit. 2017-06-27]. Dostupné z: https://web.natur.cuni.cz/ bayertom/images/courses/mmk/mk0.pdf

[4] MAREŠ, Martin. Nejkratší cesty [online]. 2015 [cit. 2017-06-27]. Dostupné z: http://mj.ucw.cz/vyuka/ga/13-dijkstra.pdf

[5] MUETZE, Annette a Ying C. TAN. Electric bicycles - A performance evaluation. IEEE Industry Applications Magazine [online]. , 12-21 [cit. 2017-06-28]. DOI: 10.1109/MIA.2007.4283505.

[6] BLONDEL, Benoît, Chloé MISPELON a Julian FERGUSON. Cycle more Often 2 cool down the planet !: Quantifying CO2 savings of cycling [online]. In: . Brussels, 2011, s. 11-14 [cit. 2017-08-27]. Dostupné z: https://ecf.com/sites/ecf.com/files/ecf_co2_web.pdf

[7] European Bicycle Market: Industry & Market Profile [online]. In: MARSILIO, Manuel a Laura DE VOCHT. Brussels, 2017, s. 29 [cit. 2017-08-27]. Dostupné z: http://www.conebi.eu/facts-and-figures/

[8] Všeobecné vysvětlivky k celému EPAC. Originální provozní návod: Bosch, pohonné systémy [online]. Austria: KTM, 2013, s. 4-11 [cit. 2017-09-02]. Dostupné z: http://www.ktmelektrokola.cz/dokumenty/navod_systemu_bosch_2014.pdf

[9] Jízdní kola - Jízdní kola s pomocným elektrickým pohonem - Jízdní kola EPAC. ČSN EN 15194+A1 30 9080. Brusel, 2015.

[10] LI, Wen, Patrick STANULA, Patricia EGEDE, Sami KARA a Christoph HERRMANN. Determining the Main Factors Influencing the Energy Consumption of Electric Vehicles in the Usage Phase. Procedia CIRP [online]. 2016, 48, 352-357 [cit. 2018-01-02]. DOI: 10.1016/j.procir.2016.03.014. ISSN 22128271. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S221282711600651X

[11] GEBHARD, Lukas, Lukasz GOLAB, S. KESHAV a Hermann DE MEER. Range prediction for electric bicycles. Proceedings of the Seventh International Conference on Future Energy Systems - e-Energy '16 [online]. New York, New York, USA: ACM Press, 2016, 2016, , 1-11 [cit. 2018-01-02]. DOI: 10.1145/2934328.2934349. ISBN 9781450343930. Dostupné z: http://dl.acm.org/citation.cfm?doid=2934328.2934349

[12] Udělejte z vlastního kola elektrokolo. Turistika.cz [online]. 2015 [cit. 2018-01-02]. Dostupné z: https://www.turistika.cz/clanky/udelejte-z-vlastniho-kola-elektrokolo/detail

[13] KOMÁREK, Martin. Softwarové inženýrství: Dokončení UML diagramu aktivit a procesního modelování. Sběr požadavků. [přednáška]. České vysoké učení technické v Praze, Fakulta elektrotechnická [cit. 2018-01-02].

[14] Teorie grafů [online]. Praha, 2010 [cit. 2018-01-02]. Dostupné z: http://teorie-grafu.cz/zakladni-pojmy/cesta-a-souvislost-grafu.php. Diplomová práce. Univerzita Karlova, Matematicko-fyzikální fakulta. Vedoucí práce RNDr. Pavla Pavlíková, Ph.D.

[15] SURYNKOVÁ, Petra. Počítačová geometrie [přednáška]. Praha [cit. 2018-01-02]. Dostupné z: http://surynkova.info/dokumenty/mff/PG/Prednasky/prednaska_8.pdf. Univerzita Karlova, Matematicko-fyzikální fakulta.

[16] Cs:Prvky. OpenStreetMap Wiki [online]. 2009 [cit. 2018-01-02]. Dostupné z: http://wiki.openstreetmap.org/wiki/Cs:Prvky

[17] Cs:Key:highway. OpenStreetMap Wiki [online]. 2009 [cit. 2018-01-02]. Dostupné z: http://wiki.openstreetmap.org/wiki/Cs:Key:highway

[18] Důvody pro jízdu na elektrokole. CykloAtom [online]. Praha [cit. 2018-01-02]. Dostupné z: http://www.horska-silnicni-kola.cz/show-free.htm?fid=54

[19] Vše o bateriích pro elektrokola. Ekolo.cz [online]. Praha, c2018 [cit. 2018-01-02]. Dostupné z: https://ekolo.cz/baterie-elektrokola

[20] NIELSEN, Jakob. 10 Usability Heuristics for User Interface Design. Nielsen Norman Group [online]. 1995 [cit. 2015-06-17]. Dostupné z: http://www.nngroup.com/articles/ten-usability-heuristics/

[21] ŠKOLAŘ, Tomáš. JÍZDNÍ KOLA NA ELEKTRICKÝ POHON: ELEKTROKOLA [online]. Přerov, 2013 [cit. 2018-01-09]. Dostupné z: http://www1.fs.cvut.cz/stretech/2013/sbornik_2013/125.pdf. Střední škola technická, Přerov.

[22] Řídící jednotka. ElektroSport [online]. 2014 [cit. 2018-01-09]. Dostupné z: http://elektrosport.cz/vse-o-elektrokolech/ridici-jednotka

[23] Calculate distance, bearing and more between Latitude/Longitude points. Movable Type Ltd [online]. Cambridge [cit. 2018-01-09]. Dostupné z: https://www.movable-type.co.uk/scripts/latlong.html

# Kapitola 9

# Obsah přiloženého CD

/
- `dp_text` ..... Text práce ve zdrojovém formátu LaTex.
- `MapConverter` ..... Zdrojový kód aplikace *MapConverter*.
- `EbikeApp` ..... Zdrojový kód aplikace *PedelecRange*.
- `pedelec_range.apk` ..... APK aplikace *PedelecRange*.
- `01_Kuznetsova-thesis-2017.pdf` ..... Text diplomové práce.