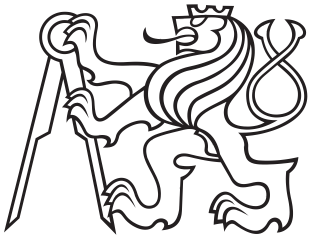
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Plánování cest pro elektrická auta s neúplnou informací

Bc. Lukáš Hromadník

Vedoucí: Ing. Marek Cuchý
Obor: Umělá inteligence
Studijní program: Otevřená informatika
Květen 2020

Poděkování

Děkuji vedoucímu Ing. Marku Cuchému za ochotu, podporu a věcné připomínky během celé práce. Děkuji rodině a přítelkyni za podporu během mého dlouhého studia. A na závěr děkuji Martinu Votrubovi a Šimonu Mandlíkovi za podporu v těžkých chvílích během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 22. května 2020

Abstrakt

V této práci jsme se zaměřili na řešení problému nalezení optimální cesty pro elektromobily při zadaném konci cesty a při známé poptávce po jednotlivých nabíjecích stanicích. Elektromobily je však při delší cestě potřeba nabíjet, vzniká zde tak poptávka po nabíjecích stanicích. Délka fronty, která může na nabíjecí stanici vzinkout, ale není dopředu známá. Vzniká zde tak neurčitost, se kterou je potřeba při plánování počítat. V práci řešíme dva problémy – v prvním problému hledáme optimální cestu a optimální začátek cesty při daném konci cesty a v druhém problému hledáme optimální cestu při zadaném začátku a konci cesty. Oba problémy uvažují omezení plynoucí z baterie v elektromobilu. Vytvořili jsme dva algoritmy, každý pro řešení jednoho z problémů. Algoritmy počítají nejspolehlivější cesty na minimalizací očekávané ceny cesty, kde cenu cesty je kvadratická. Funkčnost implementace byla experimentálně ověřena.

Klíčová slova: optimální cesta, elektromobilita, poptávka, neurčitost

Vedoucí: Ing. Marek Cuchý

Abstract

In this thesis we've focused on solving a problem of finding the optimal journey for electric cars for the specificed end of the journey and for the specified demand on charging stations. With the need to recharge electric cars for longer journey there is a demand for charging stations. And the size of the queue that can occur on the station is not known beforehand. Therefore we need to address this uncertainty during the planning. We proposed a definition for two types of the problem. The first problem is a problem of finding an optimal journey and optimal start of the journey. The second one is to find an optimal journey for a given start. Both problems consider constraints on the battery in the electric car. We've come up with a separate solution for each problem. Both our algorithms compute an optimal path with the smallest total variance. With that the realization of the final path has smaller cost then algorithms that do not consider demand on charging stations. Algorithms were implemented in Swift and their functionality was experimentally verified.

Keywords: optimal path, eletric cars, demand, uncertainty

Title translation: Car route planning for eletrical vehicles with uncertainty

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Tlak na upuštění od fosilních paliv v dopravě je v posledních letech obrovský. Automobily či jiné motorové dopravní prostředky používáme prakticky denně. Ať už se jedná o cestu do práce, do školy nebo za přáteli. S přibývajícím množstvím automobilů stoupají i jejich emise. Vlády jednotlivých států či státních uskupení se snaží vytvářet stále nové a přísnější emisní limity pro nejnovější modelové řady, avšak podíváme-li se na průměrný věk vozového parku České republiky [1], zjistíme, že toto číslo je přibližně 15 let.

Nebylo by však lepší místo přitvrzování emisních norem přejít úplně na jiný druh paliva?

Elektřina by mohla být řešením. Většina věcí, které během každodenního života používáme nějakým způsobem elektřinu využívá, tak proč ne i automobil?

Elektromobily jsou v posledních letech na vzestupu. Příkladem může být americká společnost Tesla, která velkou měrou elektromobilitu popularizuje. Určitě každý zná poslední počiny této firmy – elektrický nákladní tahač Semi nebo pick-up Cybertruck.

Avšak elektromobilů jezdí po našich silnících v poměru k ostatním automobilů stále malé množství. Jedním z důvodů je pořizovací cena elektromobilu, která je stále vysoká a dělá tak elektromobil nedostupný pro běžné občany. Druhým nedostatkem je dojezd elektromobilu. Většina aktuálních modelů má dojezd

okolo 200–250 km. Opravdu jenom malá část z nich má dojezd vyšší. Reálný dojezd však velmi závisí na provozu, kde rychlá cesta po dálnici je pro baterii v elektromobilu velice náročná a reálný dojezd se tak rychle snižuje. Z tohoto důvodu je potřeba elektromobil během delších cest i několikrát nabíjet.

Čímž se dostáváme k poslednímu nedostatku – síti nabíjecích stanic. Čerpací stanice lze najít prakticky na každém rohu, ale nabíjecích stanic zatím tolik není. A samotné nabíjení není otázka několika minut, jako je tomu u čerpacích stanic. Nabíjení automobilu z 0 % na 100 % může trvat i několik hodin v závislosti na výkonu nabíjecí stanice. Celkový čas se může ještě prodloužit, pokud bychom narazili na plně obsazenou stanici a museli bychom čekat, než se nějaké její místo uvolní. Tento scénář je v současné době spíše futuristický, ale bude potřeba ho začít řešit.

Poslední jmenované úskalí, tedy čekání na nabíjecí stanici, zkusíme v rámci naší práce optimalizovat. Čekání budeme modelovat náhodnou proměnnou. S využitím reálných se pokusíme tuto proměnnou namodelovat co nejvěrněji. Pokusíme se navrhnout algoritmy, které dokáží s touto neurčitostí pracovat a zahrnout ji do plánování. Současně řešení bude schopné pracovat s omezením v podobě stavu nabití baterie. Stav baterie je potřeba během plánování stále držet v určitých mezích, abychom zaručili proveditelnost naplánované cesty.

Cílem práce je přinést řešení pro dva typy problémů. Prvním typem problému je nalezení optimální cesty a optimálního času odjezdu cesty, pokud známe cílový čas. Druhým problémem je nalezení optimální cesty, pokud máme dopředu znám čas odjezdu i příjezdu.

V obou případech nás bude zajímat dojezd na čas. Budeme penalizovat brzký i pozdní příjezd do cílové destinace stejně. Takto nastavený problém je využíván například v donáškových službách, kde z hlediska efektivity není dobré dlouho čekat na jednom místě, ale je potřeba dorazit na místo, předat zásilku a pokračovat k dalšímu klientovi.

Kapitola 2

Analýza existujících algoritmů pro plánování cest

Problém plánování nejkratších cest pro elektrická auta s neurčitostí lze rozdělit na několik dílčích podproblémů. Tyto podproblémy mají svá specifika a řešení. V analýze se postupně od nejzákladnějšího obecného plánování nejkratších cest dostaneme až k multikritériálnímu plánování nejkratších cest s omezeními a neurčitostí, což lze považovat za nejvhodnější popis našeho problému.

2.1 Plánování nejkratších cest

Klasické plánování nejkratších cest se ve většině případů zabývá optimalizací jednoho kritéria, které je pro nás zajímavé. Může to být minimalizace času stráveného na cestě, délka cesty nebo například energetická náročnost.

Základním algoritmem pro nalezení nejkratší cesty je Dijkstrův algoritmus [2]. Pro vážený graf $G = (V, E, f)$ a počáteční uzel $u \in V$ Dijkstrův algoritmus nalezne délku nejkratší cesty mezi počátečním uzlem u a daným koncovým uzlem v. Pokud necháme algoritmus projít celý graf, získáme jako výsledek vypočtené nejkratší cesty a jejich délky mezi počátečním uzlem a všemi ostatními uzly grafu.

Výpočet algoritmu je velice jednoduchý. Na začátku algoritmu je každý uzel

Algorithm 1: Dijkstrův algoritmus

```
Result: dist
queue.init();
forall v ∈ V do
    dist[v] = ∞;
end
dist[S] = 0;
queue.enqueue(S);
while queue.notEmpty() do
    v = queue.dequeue();
    forall open neighbors v' of v do
        alt = dist[v] + distanceBetween(v, v');
        if alt < dist[v'] then
            dist[v'] = alt;
        end
    end
end
```

označen jako **otevřený**. Otevřený uzel znamená, že tento uzel ještě nebyl algoritmem zpracován a na zpracování čeká. Po zpracování je uzel označen jako **uzavřený**. Uzavřené uzly nejsou přidávány do fronty a nejsou nijak dále zpracovávány.

V hlavní smyčce algoritmu jsou jednotlivé uzly zpracovávány podle jejich vzdálenosti od počátečního uzlu. Prvním zpracovávaným uzlem je tedy počáteční uzel S, jelikož jeho vzdálenost $d(S)$ je rovna 0. Každý zpracovávaný uzel u je na začátku iterace **expandován**, neboli z grafu G nalezneme všechny jeho otevřené následovníky. Každému následovníkovi je vypočtena jeho vzdálenost od počátku. Na konci iterace označíme uzel u jako uzavřený, aby nebyl v dalších iteracích uvažovaný.

K ukončení hlavní smyčky algoritmu dojde ve chvíli, kdy jsou všechny uzly označeny jako uzavřené.

Pro správné fungování algoritmu je potřeba dodržet podmínku, že všechny hrany v grafu mají nezáporné ohodnocení. V případě existence hran se záporným ohodnocením nemusí při expanzi uzlů dojít ke správnému pořadí zpracování uzlů.

Vezměme si graf na obrázku []. Pokud bychom spustili Dijkstrův algoritmus

z uzlu A, jeho expanzí přidáme uzly B a C do fronty a uzel A označíme jako uzavřený. V další iteraci vybereme menší ze dvou otevřených uzlů ve frontě – C. Jeho vzdálenost od počátku je 3. Uzel C nemá následovníky, takže na konci iterace je pouze označen jako uzavřený. Další uzel ve frontě je uzel B. Jeho expanzí narazíme na následovníka C. Zjistíme, že aktuální napočítaná vzdálenost do C je 3. Ale pokud bychom vytvořili cestu do uzlu C přes uzel B, získáme cestu délky -5. A zde vzniká problém. Jelikož uzel C již byl zpracovaný, je aktuálně označen jako uzavřený, nemůžeme upravovat délku cesty vedoucí do C ani potenciální následovníky uzlu C a tedy délka nejkratší délky cesty do uzlu C je špatná.

Pokud tedy graf obsahuje záporně ohodnocené hrany a chceme-li na něm spustit Dijsktrův algoritmus, můžeme přepočítat ohodnocení na hranách tak, aby zůstaly zachovány nejkratší cesty z původního grafu a současně aby všechna záporná ohodnocení hran byla nahrazena nezápornými ohodnoceními. Pro tuto transformaci ohodnocení můžeme využít Johnsonův algoritmus [3], který slouží k výpočtu nejkratších cest mezi všemi uzly.

Johnsonův algoritmus pracuje ve 3 krocích:

1. Přidáme k původnímu grafu nový uzel q a spojíme ho se všemi ostatními. Nové hrany budou mít ohodnocení 0.
2. Z uzlu q spustíme Bellman-Fordův algoritmus, který napočítá vzdálenost $h(u)$ pro všechny uzly grafu od uzlu q. Dosáhne pomocí následujícího pseudokódu

```
forall i from 1 to |V| − 1 do
    forall (u, v) ∈ E do
        if h[u] + w[u, v] < h[v] then
            h[v] = h[u] + w[u, v]
```

3. V posledním kroku dojde k převážení ohodnocení původního grafu

$$w(u, v) = w(u, v) + h(u) - h(v).$$

Po převážení stačí odebrat přidaný uzel q a máme graf s nezápornými hranami, který je připravený pro použití s Dijkstrovým algoritmem.

Druhým základním algoritmem pro plánování nejkratších cest je algoritmus A* [4]. Tento algoritmus funguje na velice podobném principu jako Dijkstrův algoritmus. Avšak uzel x není k expanzi vybrán pouze na základě jeho vzdálenosti od počátečního uzlu $g(x)$, ale k této hodnotě je připočtena ještě hodnota heuristické funkce. Pomocí heuristické funkce $h(x)$ přidáváme uzlu x druhé ohodnocení, které odhadhuje jeho vzdálenost k cílovému uzlu. K expanzi je tak vybrán ten uzel x, jehož ohodnocení $f(x) = g(x)+h(x)$ vzniklé součtem vzdálenosti $g(x)$ od počátečního uzlu a hodnotou heuristické funkce $h(x)$ je nejmenší.

Abychom měli zaručený optimální výsledek algoritmu A*, je potřeba, aby použitá heuristická funkce splňovala určitá kritéria. Prvním kritériem je přípustnost funkce. Heuristická funkce je přípustná právě tehdy, když hodnota heuristické funkce je pro každý uzel x menší nebo rovna nejmenší možné vzdálenosti k cíli z tohoto uzlu, či-li $h(x) \leq h^*(x), \forall x$. Pak máme zaručený správný chod algoritmu a jeho optimální výsledek. Například pro plánování silničních cest lze použít jako heuristickou funkci vzdušnou vzdálenost mezi daným uzlem a cílovým uzlem.

Pokud je navíc heuristická funkce $h(x)$ konzistentní, tedy splňuje podmínku $h(x) \leq d(x, y)+h(y)$, pro každou hranu (x, y), kde $d(x, y)$ označuje ohodnocení hrany, navštíví algoritmus každý uzel maximálně jednou [5].

Dijkstrův algoritmus a A* mají mnoho společného. Hlavní odlišnost mezi nimi je heuristická funkce použitá v algoritmu A*. Pokud však heuristická funkce $h(x)$ bude pro každý uzel x vracet hodnotu nula, tedy $h(x) = 0$, pak jsou průběhy obou algoritmů shodné. Pokud ale heuristická funkce vrací hodnoty různé od nuly, změní se průběh algoritmu A* na tolik, že již není možné ho využít k vypočtení nejkratších vzdáleností z počátečního uzlu ke všem ostatním uzlům.

2.2 Plánování nejkratších cest pro automobily

Plánování cest pro automobily vyžaduje použití silniční sítě převedené do grafu. Silniční síť obsahuje velké množství informací, které je potřeba reprezentovat v grafu. Díky tomu silniční graf, nazvěme tak silniční síť převedenou do grafu, obsahuje spoustu vrcholů a hran mezi nimi.

2.2.1 Urychlující techniky

Velikost silničního grafu roste velice rychle s velikostí území, které má silniční graf reprezentovat. Vzniká zde tak potřeba použít zrychlující techniky, které využívají specifické vlastnosti silničního grafu, aby výpočet plánovacích algoritmů byl co nejrychlejší. Tyto zrychlující techniky využívají třeba faktu, že jednotlivé hrany silničního grafu lze hierarchicky roztřídit, jako například v článku [6], a při plánování tohoto rozdělení využít.

Dijkstrův algoritmus je hojně využívaný v plánování nejkratších cest nad silničním grafem, díky čemuž se pro něj v průběhu let podařilo vytvořit několik zrychlujících technik. Článek [7] poskytuje ucelený přehled o některých z nich

- obousměrné vyhledávání,
- goal-directed vyhledávání,
- metody využívající hierarchie.

Současně článek [7] popisuje i možnosti jakými je možné jednotlivé urychlující techniky spojovat / kombinovat a dosáhnout tak lepších výsledků. V článku [7] možné se též dozvědět, že výpočetní náročnost některých zrychlujících technik může být ve výsledku vyšší, než je samotné zrychlení vyhledávacího algoritmu.

První technikou je obousměrné vyhledávání. Obousměrné vyhledávání patří mezi způsoby, při kterém dochází ke snížení výpočetní náročnosti díky snížení počtu iterací potřebných k dosažení výsledku. Během obousměrného vyhledávání je algoritmus spuštěn z počátečního i koncového uzlu současně.

Vyhledávání z koncového uzlu navíc používá hrany opačně orientované, aby bylo možné nejkratší cestu nalézt. Samotné vyhledávání skončí ve chvíli, kdy dojde k průniku mezi množinami uzavřených uzlů.

Goal-directed vyhledávání přidává do běhu algoritmu funkci, která pomáhá lépe ohodnotit uzly grafu. Algoritmus A* s jeho heuristickou funkcí můžeme zařadit mezi zrychlující techniky Dijkstrova algoritmu.

V knize [8] lze narazit na dvě nejznámější zrychlující techniky využívající hierarchii silničního grafu

- highway hierarchies a
- contraction hierarchies.

Highway hierarchies [9] je zrychlující technika Dijkstrova algoritmu založená na předpočítání určitého okolí každého uzlu a následné využití těchto okolí k prořezávání grafu. Výsledek této techniky dokáže velice urychlit samotný průběh Dijkstrova algoritmu.

Contraction hierarchies [6] je druhá speed-up technika postavená na hierarchii uzlů, kde nejprve jsou uzly seřazeny podle důležitosti. Vezmeme-li například silniční síť, kde uzly budou reprezentovat křižovatky a hrany budou reprezentovat cesty mezi nimi, pak uzel spojující silnice nižší třídy bude méně důležitý než uzel spojující silnice vyšší třídy. Následně jsou mezi uzly vytvářeny zkratky (*shortcuts*) tak, že cesta skrze nejméně důležitý uzel je nahrazena právě touto zkratkou. Výsledné hledání probíhá nad takto upraveným grafem.

V současnosti nejvíce používanou urychlující technikou využívající hierarchii silniční sítě je transit-node routing z článku [10]. V rámci této techniky je potřeba pro daný graf předpočítat tzv. transitní uzly. Dále je mezi všemi těmito uzly napočítána vzdálenost cesty. Pro vytvoření transitních uzlů je využita technika highway hierarchies, kde uzly v nejvyšší úrovni jsou považovány za transitní. Samotné plánování pak probíhá právě nad těmito transitními uzly, kterých je několikanásobně méně než uzlů v původním grafu.

Dle [6] lze i contraction hierarchies použít jako techniku pro nalezení transitních uzlů a tím ještě více urychlit běh algoritmu.

2.3 Plánování nejkratších cest pro elektrické automobily

Plánování pro elektrické automobily s sebou přináší jistá specifika. Během plánování jsou uvažována omezení na baterie v elektrickém automobilu, u kterých by během cesty nemělo dojít k úplnému vybití, popřípadě k poklesu nabití pod předem definovanou úroveň.

Z pohledu plánování se lze na toto specifikum dívat různě. První možností je změnit typ plánování. U elektrického automobilu můžeme chtít plánovat cestu energeticky nejoptimálnější, což je případ v článku [11]. Bylo toho docíleno vytvořením energetického grafu, kde každé hranové ohodnocení udávalo spotřebu elektrické energie při přechodu po této hraně. Díky této úpravě se z optimalizace časové stala optimalizace energetická.

Elektrický automobil, stejně jako další elektricky poháněné stroje, disponuje rekuperací elektrické energie. To znamená, že například při jízdě z kopce je automobil schopný částečně své baterie dobíjet brzděním. V praxi to však ale znamená, že pokud máme hrany grafu ohodnoceny spotřebou elektrické energie potřebnou k překonání dané hrany, vzniknou nám v grafu hrany, jejichž ohodnocení bude záporné, a tedy pro tento typ grafu nelze použít základní verzi Dijkstrova algortimu.

Toto omezení lze však změnit. Článek [12] navazuje na [11] a upravuje ho tak, aby se pro takto definovaný problém s energetickým grafem na vstupu dalo využít Dijkstrova algoritmu. Pro převážení lze využít Johnsonova algoritmu. Díky tomu lze při plánování uvažovat i zrychlující techniky diskutované výše.

Druhou možností, jak se dívat na plánování s baterií, je koukat se na baterii jako na omezení při plánování. Omezení jenž jsou kladena na baterii jsou z praxe jasná – baterie se nesmí během jízdy vybít pod předem danou určitou úroveň, kde tato úroveň může být i 0. Baterie má také kapacitu, kterou nelze překročit. Plánování se pak může zabývat stále nejkratší cestou, avšak během plánování je možné v některých uzlech zastavit další expanzi právě díky porušení těchto omezení.

Přidáním omezení do plánování se stává z unikriteriálního problému problém multikriteriální. Nejkratší cesta s omezením na rozdíl od klasického problému nejkratších cest patří do třídy NP-těžkých problémů [13].

V článku [13] se autor zabývá rozšířením klasického unikriteriálního algoritmu pro plánování nejkratších cest na bikriteriální. Ohodnocení hran grafu se tak sestává ze dvou částí, pomocí kterých lze lépe popsat určité problémy. Rozšířením ohodnocení hrany o další prvek se však z jednoduchého lineárního problému stává problém s exponenciální časovou složitostí. Algoritmus popsaný v článku [13] popisuje správný výběr uzlu k expanzi a způsob porovnání jednotlivých ohodnocení hran. Současně zde také vzniká potřeba neukončovat algoritmus při nalezení prvního řešení. První nalezený výsledek je správný, avšak nemusí být optimální.

Řešení problému z článku [13]výše lze najít v článcích [14][15]. Autoři zde představují implementaci multikriteriálního A* algoritmu, pomocí kterého lze do plánování převést situaci, kdy nestačí použít pouze jedno optimalizační kritérium. V článku je podrobně popsána implementace algoritmu a potřeba použití přípustné heuristiky během plánování.

V reálné aplikaci plánování nejkratších cest pro elektrické automobily, kde vyhledáváme časově nejkratší cestu, nelze vypustit řešení, které má lepší čas ale horší stav baterie, jelikož může snadno dojít k situaci, kdy nižší stav baterie znemožní nalézt přípustné řešení dané instance problému.

2.4 Plánování s časově závislými hranami

Plánováním s časově závislými hranami se zabývá problémy, kdy ohodnocení na hranách je je závislé na čase, ve kterém danou hranu navštívíme. Vzhledem k povaze článků k danému problému lze rozdělit podkapitolu na dvě hlavní části.

První část tvoří články [16][17][18][19], které se zabývají diskrétním přístupem k plánování s časově závislými cenami hran. Tento přístup se zakládá na rozdělení časové závislosti na M stejně dlouhých časových okamžiků, se

kterými se poté plánuje. Díky aplikaci diskretizace lze problém formulovat tak, že k jeho řešení můžeme využít Dijkstrův algoritmus.

Tyto články dále rozvíjí [20], kde velikost hrany není v daném časovém okamžiku tvořena pouze jednou hodnotou ale dvěma. Řešení tohoto problému je pak ovšem shodné s řešením bikriteriálního problému v [13].

Druhou část tvoří články [21][22], které se zabývají problémem kanadského cestovatele. Problém je definován na cestovali po Kanadě, který při příchodu na křižovatku zjistí, jak dlouho mu bude trvat překonat následující část cesty. Touto definicí se dostáváme k problému, kdy není dopředu jasné, jakou cenu budou mít některé hrany před začátkem běhu algoritmu, a jednotlivá ohodnocení hran jsou tedy časově závislá. Tento problém lze formulovat jako Markovův rozhodovací proces, jehož řešení lze najít pomocí lineárního programování nebo dynamického programování.

2.5 Plánování s neurčitostí

Plánováním s neurčitostí se zabývají autoři článku [23], kde vyhledávací graf obsahuje hrany jejichž hodnota je dána nějakým pravděpodobnostním rozdělením. V jejich případě je pravděpodobnostní rozdělení použito k modelaci času potřebného k překonání dané hrany. Toto rozšíření běžného plánování umožňuje lépe zachytit realitu a ve výsledku naplánovat cestu vhodněji díky znalosti více informací.

Autoři článku se zabývali dvěma základními problémy v plánování s neurčitostí při daném cílovém času (*deadline*):

- plánování optimální cesty a optimálního času startu,
- plánování optimální cesty při pevně daném času startu.

První problém se týká nalezení nejvhodnějšího času odjezdu a optimální cesty při pevně daném času dojezdu. Autoři článku [23] hledají optimální cestu tak, že minimalizují cenu cesty. Jako funkci ceny zvolili nejdříve

$$C(t) = t^2,$$

kde t je čas příjezdu do cíle. Pro jednoduchost uvažují koncový čas $t = 0$. Pokud koncový čas různý od nuly, budeme uvažovat jako funkci ceny cesty uvažovat upravenou formuli

$$C(t) = (t_{end} - t)^2,$$

kde t_{end} je předem daný čas v cíli. Cena cesty jako kvadratická funkce slouží jako penalizace pro pozdější i předčasné příjezdy do cíle.

Tuto kvadratickou funkci lze navíc rozšířit o exponenciální člen

$$C(t) = t^2 + \lambda e^{kt},$$

kde pomocí ladícího parametru k můžeme lépe upravit algoritmus, aby penalizoval více pozdější dojezd před dřívějším a naopak.

V článku [23] je dokázáno, že nalézt exaktní optimální cestu a optimální čas startu

- s použitím kvadratické funkce jako funkce ceny cesty a s použitím obecného pravděpodobnostního rozdělení je možné s využitím deterministického algoritmu pro nalezení nejkratší cesty,
- s použítím kvadratické cenové funkce rozšířené o exponenciální člen a s použitím normálního rozdělení je možné s využitím deterministického algoritmu pro nalezení nejkratších cesty,
- s použitím kvadratické cenové funkce rozšířené o exponenciální člen a s použitím obecného rozdělení není možné s využitím deterministického algoritmu pro nalezení nejkratší cesty, protože toto nastavení může porušit optimalitu na vnitřních částech cesty.

Na druhé straně problém plánování optimální cesty při pevně daném času startu patří do třídy NP-obtížných problémů, což je v článku [23] dokázáno.

Tento problém lze řešit pomocí dynamického programování s pseudopolynomiální složitostí závisející polynomiálně na největší střední hodnotě času cesty. Problém nalezení optimální cesty a optimálního začátku lze řešit s polynomiální složitostí.

Články [24] a [25] se zabývaly také plánováním s neurčitostí. V obou článcích autoři vytvořili definice pro různé problémy s nalezením optimální cesty. Byly vytvořeny celkem 3 definice, kde jednotlivé definice postupně rozšiřovaly předcházející.

Prvním problémem je minimalizace středního hodnoty času cesty. Optimální cesta je v tomto problému nalezena tak, že ohodnocení jednotlivých hran jsou nahrazena střední hodnotou pravděpodobnostního rozdělení, které je ke každé hraně přidruženo. Tento problém lze vyřešit pomocí Dijkstrova algoritmu, avšak výsledná cesta nemusí být optimální z důvodu velkého rozptylu na některých hranách cesty.

Druhým problémem je problém nejspolehlivější cesty. Tento problém je definován s pomocí předem dané určené maximální ceny cesty. Ve výpočtu se pak snažíme maximalizovat pravděpodobnost, že cena výsledné cesty bude menší než tato stanovená maximální cena.

Třetí problém rozšiřuje předcházející o parametr α, pomocí kterého definujeme s jakou minimální pravděpodobností chceme dorazit do cíle, přičemž současně se snažíme maximalizovat pravděpodobnost, stejně jako u předchozího problému, že výsledná cena cesty nepřesáhne předem stanovou maximální cenu.

Článek [25] představil řešení posledního jmenovaného problému založené na genetickém algoritmu. Použití genetického algoritmu je v této situaci možné, avšak nalezení optimální cesty a rychlost jejího nalezení velice záleží na počátečním nastavení a náhodě.

Na druhé straně článek [24] přinesl řešení stejného problému pomocí upraveného Dijkstrova algoritmu. Ohodnocení jednotlivých hran využívá kumulovanou distribuční funkci a jednotlivé cesty jsou v průběhu aktualizovány a rozšiřovány. Díky tomu lze během plánování projít všechny možné cesty, i ty,

které se během první expanze nezdály jako hodnotné.

Dále je v článku [24] vytvořený zrychlující technika v podobě heuristiky, díky které lze k vyhledávání použít algoritmus A* a tím zrychlit běh algoritmu.

Jediné úskalí článku je možné vidět v tom, že se jeho metody dají aplikovat pouze na hrany s normálním rozdělení. Ostatní pravděpodobností rozdělení nejsou v článku uvažována a je potřeba pro ně udělat další výzkum.

Kapitola 3

Terminologie

V této kapitole nejdříve představíme potřebné definice na základě kterých pak budeme definovat samotné problémy, kterými se tato práce zabývá.

3.1 Neorientovaný graf

Nechť V a E jsou libovolné disjunktní množiny. **Neorientovaným grafem** nazýváme uspořádanou dvojici $G = (V, E)$, kde V je neprázdná množina vrcholů a $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ je množina dvouprvkových množin vrcholů, tzv. hrany.

3.2 Orientovaný graf

Aby mělo plánování smysl, je potřeba specifikovat, odkud kam se má plánování provést. Toto rozšíření neorientovaného grafu se nazývá graf orientovaný. Orientovaný graf se od neorientovaného liší tím, že jednotlivé hrany mají danou orientaci, jinak řečeno lze jednoznačně určit, ve kterém uzlu hrana začíná a ve kterém končí.

Nechť V a E jsou libovolné disjunktní množiny. **Orientovaným grafem** nazýváme uspořádanou dvojici $G = (V, E)$, kde V je neprázdná množina

vrcholů a $E \subseteq \{(u, v\ ) \mid u, v\ \in V, u\ \neq v\ \}$ je množina uspořádaných dvojic vrcholů, tzv. hrany.

Uzel u nazýváme počátečním uzlem hrany a uzel v nazýváme koncovým uzlem hrany.

3.3 Vážený orientovaný graf

Pro potřeby plánování je dále vhodné nějakým způsobem jednotlivé hrany grafu vážit – přiřadit jim hodnotu. Díky tomu lze v grafu vyhledávat např. nejkratší, resp. nejdelší, cesty.

Nechť V a E jsou libovolné disjunktní množiny a $\delta : E \to R$ je zobrazení. **Váženým orientovaným grafem** nazýváme trojici $G = (V, E, \delta)$, kde V je neprázdná množina vrcholů, $E \subseteq \{(u, v\ ) \mid u, v\ \in V, u\ \neq v\ \}$ je množina uspořádaných dvojic vrcholů, tzv. hrany, a δ je zobrazení, které každé hraně z množiny E přiřazuje reálné ohodnocení.

3.4 Cesta

Nyní si formálně zadefinujeme cestu grafem. Cesta nám dovoluje zaznamenat určitou posloupnost uzlů, které jsme navštívili, a ve spojení s váhami na hranách lze pak provádět nad těmito cestami operace, např. porovnání.

Nechť pro danou dvojici uzlů u a v orientovaného grafu $G = (V, E)$ existuje posloupnost uzlů a hran

$$J = (v_0, e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n)\,,$$

kde $e_i \in E, e_i = (v_{i-1}, v_i)$, pro $i = 1, 2, \ldots, n$, $v_i \in V$, $v_0 = u, v_n = v$. Pak tuto posloupnost J nazveme orientovanou **cestou** grafu G mezi uzly u a v.

Cesty mohou být obecně i neorientované. Tento případ však v práci není uvažován z důvodu nepotřebnosti. Dále bude pro jednoduchost cestou vždy

myšlena cesta orientovaná, pokud nebude v textu specifikováno jinak.

Nyní můžeme zadefinovat délku cesty, abychom mohli jednotlivé cesty porovnávat, popř. uspořádat.

3.5 Délka cesty

Nechť $G = (V, E, \delta)$ je vážený orientovaný graf a nechť J je cesta tohoto grafu. **Délkou cesty** nazýváme funkci $w(J)$ definovanou

$$w(J) = \sum_{i=0}^{n} \delta(e_i), \ e_i \in J.$$

Během plánování nás bude zajímat pouze specifická cesta. Délka této cesty bude optimalizovat dané zobrazení δ grafu G. Pro náš případ se může jednat o optimalizaci například délky trasy, času cesty nebo spotřeby elektrické energie.

3.6 Nejkratší cesta

Nechť $G = (V, E, \delta)$ je vážený orientovaný graf a nechť máme množinu M všech možných cest grafu G. **Nejkratší cestou** nazýváme cestu J^* takovou, že

$$J^* = \arg\min_J w(J),$$

kde $J \in M$.

3.7 Cesta s omezeními

V rámci plánování pro elektrická auta je potřeba počítat s baterií v autě, která vnáší do plánování určitá omezení. Aktuální stav nabití baterie nemůže přesáhnout maximální možnou kapacitu a současně se baterie nemůže

vybít více než pod předem danou dolní hraniční hodnotu. Je tedy potřeba zadefinovat i cestu, která dokáže tato omezení splnit.

Nechť $G = (V, E, \delta)$ je vážený orientovaný graf a C je množina omezení. **Cestou s omezeními** nazýváme takovou cestu J_C, která začíná v uzlu $u \in V$, končí v $v \in V$, a všechna omezení z množiny C jsou v rámci této cesty splněna.

Spojením předcházejících dvou definic získáme nejkratší cestu s omezeními, které bude tvořit základ pro řešení.

3.8 Nejkratší cesta s omezeními

Nechť $G = (V, E, \delta)$ je vážený orientovaný graf, C je množina omezení a M je množina všech možných cest s omezeními J_C grafu G. **Nejkratší cestou s omezeními** nazýváme cestu J_C^* takovou, že

$$J_C^* = \arg\min_{J_C} w(J_C),$$

kde $J_C \in M$.

Pro nejkratší cestu s omezeními v grafu $G = (V, E, \delta)$ tedy platí, že její délka, jenž je dána funkcí $w(J_C)$, je nejmenší a současně všechna omezení, která jsou na cestu kladena, jsou splněna.

3.9 Silniční graf

Chceme-li plánovat cesty pro automobily, ať už elektrické či se spalovacím motorem, je potřeba určitým způsobem přenést běžnou silniční síť do nějaké vhodné datové reprezentace. Většina plánovacích algoritmů pracuje s grafy a teorií kolem nich. Přenést silniční síť do grafového prostoru je proto velice výhodné a současně prosté a jednoduché.

Silniční graf je vážený orientovaný graf $G_R = (V, E, \delta)$, kde

- V je množina všech vrcholů silničního grafu, vrcholy budou tvořit významné body silniční sítě, např. křižovatky, přechody pro chodce apod.
- E je množina hran silničního grafu, které reprezentují silnice mezi jednotlivými významnými body silniční sítě.
- $\delta : E \to \mathbb{R}_0^+$ je zobrazení udávající čas potřebný k překonání dané hrany.

Ke každému vrcholu grafu je navíc přidružená GPS souřadnice a jeho nadmořská výška. Tyto informace budou využity při řešení problému.

Kapitola 4

Definice problému

Problémem, kterým se zabýváme v této práci, je nalezení optimální cesty s neurčitostí pro elektrický automobil, u kterého je potřeba během plánování uvažovat i omezení plynoucí z baterií umístěných v něm, vyjíždějícího z bodu A do bodu B a s časem příjezdu do cíle t_{end}. Samotné plánování může být prováděno pro cesty energeticky náročnější, než jaké jsou možnosti baterie v autě. Výsledkem algoritmu tak může být cesta, která nelze celá překonat na jedno plné nabití baterie. Během cesty je tedy potřeba zastavit na jedné či více nabíjecích stanicích a baterii nabít. Jednotlivé nabíjecí stanice mohou mít různou délku fronty (obsazenost) v závislosti na denní době. Tím je ovlivněn čas, který musíme na stanici vyčkat, než dojde k uvolnění některé z přípojek.

Silniční graf zadefinovaný v předchozí kapitole není vhodný pro plánování pro elektromobil. Informace obsažená v tomto grafu je velmi obecná a pro potřeby plánování cest pro elektromobily je potřeba informaci v grafu rozšířit.

4.1 Nabíjecí stanice

Důležitým rozšířením silničního grafu jsou nabíjecí stanice. Nabíjecí stanicí pro elektrický automobil rozumíme uzel $v \in V$ silničního grafu, jehož navštívením se zvýší aktuální stav nabití baterie b_{curr} a čas cesty se posune o dobu potřebnou k dosažení tohoto stavu nabití.

V rámci nabíjení lze baterii v automobilu nabít úplně, ale i částečně. Součástí našeho problému bude i toto částečné nabíjení, díky čemuž získáme více možností jak naplánovat cestu.

V rámci plánování nedává smysl vždy zkoumat všechny částečné úrovně nabití. Pokud například automobil dorazí na nabíjecí stanici a aktuální stav nabití baterie bude mít hodnotu 33 % její maximální kapacity, hodnoty pod 33 % jsou pro plánování zbytečné.

Čas nabíjení definujeme jako funkci

$$\eta(b_{curr}, ch_{max}, b_{max}) = \frac{b_{max} - b_{curr}}{ch_{max}}$$

kde b_{curr} je aktuální hodnota nabití baterie v automobilu při příjezdu na nabíjecí stanici, ch_{max} je nabíjecí výkon nabíjecí stanice a b_{max} je koncová hodnota elektrické energie, na kterou chceme baterii nabít.

Čistě formálně lze o nabíjení uvažovat jako o smyčce na grafu, kdy automobil se na jednom vrcholu grafu zastaví po nějaký určitý čas a jeho interní stav se určitým způsobem změní. Tato formální definice může ale některým algoritmům způsobit problémy. Pokud smyčky na grafu nejsou přípustné pro daný algoritmus, lze je jednoduše odstranit tak, že se ohodnocení z hrany $e = (u, v)$, kde uzel v je nabíjecí stanice, spojí s ohodnocením z nabíjecí smyčky e'. Budeme-li uvažovat i částečné nabíjení baterie, vznikne takto několik nových hran $ee' = (u, v) \circ (v, v)$, ve kterých bude spojena jak cesta do nabíjecí stanice, tak samotné nabíjení.

4.2 Čas čekání

Oba dva naše problémy obsahují neurčitost. Neurčitostí modelujeme jako čas potřebný k uvolnění místa na nabíjecí stanici. Představme si, že přijedeme na nabíjecí stanici a poptávka v daný časový okamžik t bude vysoká. To způsobí, že se na nabíjecí stanici vytvoří fronta délky k. V rámci práce modelujeme neurčitost jako čas, který je potřeba vyčkat, než je před námi odbaveno k

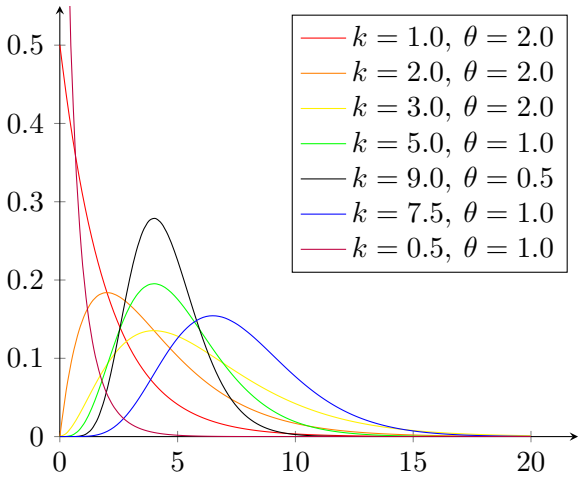
zákazníků z fronty.

Z důvodu této formulace se jako nejvhodnější pravděpodobností rozdělení jeví Gamma rozdělení.

4.2.1 Gamma rozdělení

Gamma rozdělení patří do rodiny exponenciálních pravděpodobnostních rozdělení a přebírá dva parametry:

- paremetr k, pomocí kterého se definuje tvar rozdělení, a
- parametr θ, pomocí kterého se definuje rozměr rozdělení.



Obrázek 4.1: Distribuční funkce Gamma rozdělení

Distribuční funkce Gamma rozdělení má následující tvar

$$\frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{x}{\theta}}$$

kde Gamma funkce $\Gamma(x)$ je

$$\Gamma(z) = \int_0^\infty t^{z-1} e^t \, \mathrm{d}t.$$

Pro přirozená čísla a nulu lze Gamma funkci vyjádřit jako $\Gamma(x) = (x-1)!$. Obecná formule Gamma funkce je výpočetně velmi náročná. Zjednodušená formule využívající faktoriál je pro použití s přirozených čísel a nuly výpočetně méně náročnější, avšak přináší sebou velké omezení během plánování, kdy je potřeba během plánování uvažovat pouze čas s diskrétními hodnotami.

Pokud se však podíváme na střední hodnotu a rozptyl Gamma rozdělení

$$\mu = k\theta, \quad \sigma = k\theta^2$$

získáme dvě hodnoty, jejichž výpočetní náročnost je konstantní a jejich využití je během plánování vhodnější.

Gamma rozdělení se používá k modelování času, který je potřeba vyčkat, než nastane k událostí. V našem případě pouze čekáme, než se uvolní fronta k zákazníků před námi na nabíjecí stanici a my budeme moci automobil nabít. Exponenciální rozdělení, které se používá k modelaci času, který je potřeba vyčkat, než nastane určitá událost, je speciálním případem Gamma rozdělení.

4.3 Elektrický silniční graf

Formálně zadefinujeme elektrický silniční graf se všemi rozšířeními jako $G_e = (V, E, EC, \delta, \varphi, \varepsilon)$, kde

- V je množina všech uzlů grafu, kde uzly představují uzly silniční sítě nebo nabíjecí stanice,
- E je množina všech hran grafu, kde každá hrana popisuje spojnici mezi dvěma uzly grafu,
- EC je model elektrického auta definovaný níže,
- $\delta : E \to \mathbb{R}^+$ je zobrazení přiřazující každé hraně její velikost (délku),
- $\varphi : E \to \mathbb{R}^+$ je zobrazení přiřazující hraně čas potřebný k jejímu překonání,
- $\varepsilon : E \to \mathbb{R}$ je zobrazení přiřazující každé hraně spotřebu elektrické energie potřebnou k překonání dané hrany.

4.4 Model elektrického automobilu

Pro pohyb po takto rozšířeném silničním grafu je potřeba využít určitý model, který bude schopný nově zadefinované informace zpracovat. Pro tyto účely definujeme **model elektrického automobilu** EC jako uspořádanou dvojici

$$EC = (b_{min}, b_{max}),$$

kde b_{min} je hodnota nabití elektrické energie, pod kterou nesmí hodnota na baterii klesnout. Druhou složkou je b_{max}, která určuje kapacitu baterie. Hodnota kapacity baterie b_{max} slouží pouze jako nabíjecí limit. Pokud během plánování dojde k překonání hrany, při kterém nabití baterie překročí hodnotu b_{max}, nebude tato nadbytečná energie uvažována a v plánování bude vždy nahrazena hodnotou b_{max}.

4.5 Cena cesty

Abychom mohli správně naplánovat cestu, je potřeba zadefinovat její cenu. Cena cesty se bude odvíjet od toho, jakou veličinu budeme během plánování optimalizovat. Může to být čas, může to být energie anebo vektor hodnot.

Vezmeme-li v úvahu čas potřebný k čekání, může cenová funkce cesty obsahovat jak čas strávený na cestě, tak rozptyl spojený s časem čekání této cesty. Na základě důkazů z článku [23] se dozvídáme, že pro potřeby plánování optimální cesty lze při nastavení správné cenové funkce a správné distribuční funkce na hranách řešit úlohu jako nalezení nejkratší cesty. Nejkratší cesta v tomto kontextu znamená nejspolehlivější cesta, tedy s nejmenší variancí. Výsledná cesta ale nemusí být z hlediska absolutního času nejvýhodnější, avšak výsledný čas po její realizaci bude mít nejmenší odchylku od koncového času a tedy i nejvyšší pravděpodobnost, že daný čas se bude blížit naplánované cestě.

Jelikož neurčitost je pouze na nabíjecích hranácj, časy přechodu po ostatních

hranách e budou konstantní, tedy bude pro ně platit, že $\mu_e = \varphi(e)$ a $\sigma_e^2 = 0$.

Díky definici ceny cesty v 3.5 je možné cenu cesty nestanovit pouze jako jedno číslo, ale i jako vektor. Díky tomu může mít cena více složek, přičemž optimalizace během plánování se může soustředit na jednu složku a výsledkem může být složka jiná.

Nyní je možné se všemi předešlými definicemi zadefinovat i problémy a jejich vstupy, se kterými se zabýváme v této práci.

4.6 Problém optimální cesty a optimálního času odjezdu

Definujme **cestovní dotaz** pro problém optimální cesty a optimálního času odjezdu $q_1 = (u, v, t_{end}, EC, b_{init})$, kde

- $u \in V$ je počáteční uzel v G_e,
- $v \in V$ je koncový uzel v G_e,
- $t_{end} \in \mathbb{R}_0^+$ je čas příjezdu,
- $EC = \{b_{min}, b_{max}\}$ je model elektrického auta,
- $b_{init} \in [b_{min}, b_{max}]$ je počáteční stav nabití baterie.

Máme daný elektrický silniční graf G_e a máme daný cestovní dotaz q_1. Problém optimální cesty a optimálního času odjezdu se snaží nalézt optimální cestu J a optimální čas odjezdu t_{start} takové, že cesta J je přípustná pro elektrické vozidlo. Cesta J začíná v uzlu u s počátečním stavem baterie b_{init} a končí v uzlu v s minimální cenou cesty $w(J)$. Pro jednotlivé nabíjecí stanice jsou k dispozici distribuce času čekání.

4.7 Problém optimální cesty při pevně daném času odjezdu

Definujme **cestovní dotaz** pro problém optimální cesty a optimálního času odjezdu $q_2 = (u, v, t_{start}, t_{end}, EC, b_{init})$, kde

- $u \in V$ je počáteční uzel v G_e,
- $v \in V$ je koncový uzel v G_e,
- $t_{start} \in \mathbb{R}_0^+$ je čas odjezdu,
- $t_{end} \in \mathbb{R}_0^+$ je čas příjezdu,
- $EC = \{b_{min}, b_{max}\}$ je model elektrického auta,
- $b_{init} \in [b_{min}, b_{max}]$ je počáteční stav nabití baterie.

Máme daný elektrický silniční graf G_e a mamé daný cestovní dotaz q_2. Problém optimální cesty při pevně daném času odjezdu t_{start} se snaží nalézt optimální cestu J takovou, že cesta J je přípustná pro elektrické vozidlo. Cesta J začíná v uzlu u s počátečním stavem baterie b_{init} a končí v uzlu v s minimální cenou cesty $w(J)$. Pro jednotlivé nabíjecí stanice jsou k dispozici distribuce času čekání.

Kapitola 5

Popis řešení

V této kapitole si popíšeme konkrétní řešení našich problémů. Oba naše problémy jsou si velice podobné a navzdory tomu, že liší se pouze v tom, zda-li je na začátku výpočtu čas odjezdu zafixovaný nebo ne, jejich řešení se velice liší.

Obě řešení však mají i části, které jsou sdílené. Díky tomu lze tyto části popsat na začátku kapitoly a v průběhu se na ně odkazovat.

5.1 Vyhledávací graf

Elektrický silniční graf je velice šikovný vstupní bod do plánování pro elektrické automobily. Pro naše problémy je však stále nedostačující a je potřeba ho rozšířit o další informace.

Mějme elektrický silniční graf G_e zadefinovaný v 4.3. Definujme **vyhledávací graf** pro řešení naší práce jako

$$G = (V, E, C, EC, \delta, \varphi, \varepsilon, wt, cht, \sigma_e, \mu_e),$$

kde

- V je množina všech uzlů grafu, kde uzly představují uzly silniční sítě

nebo nabíjecí stanice,

- E je množina všech hran grafu, kde každá hrana popisuje spojnici mezi dvěma uzly grafu,
- $C \subseteq V$ je množina všech nabíjecích stanic,
- EC je model elektrického auta,
- $\delta : E \to \mathbb{R}^+$ je zobrazení přiřazující každé hraně její velikost (délku),
- $\varphi : E \to \mathbb{R}^+$ je zobrazení přiřazující každé hraně čas potřebný k jejímu překonání,
- $\varepsilon : E \to \mathbb{R}$ je zobrazení přiřazující každé hraně spotřebu elektrické energie potřebnou k překonání dané hrany,
- $cht : V \times \mathbb{R}_0^+ \times b_{max} \to \mathbb{R}_0^+$ je zobrazení přiřazující uzlu v čase $t \in \mathbb{R}_0^+$ čas, který je potřeba vyčkat, než se baterie v automobilu nabije na úroveň b_{max},
- $\sigma_e : E \times \mathbb{R}_0^+ \to \mathbb{R}_0^+$ je zobrazení přiřazující hraně rozptyl času čekání na nabíjecí stanici v čase $t \in \mathbb{R}_0^+$,
- $\mu_e : E \times \mathbb{R}_0^+ \to \mathbb{R}_0^+$ je zobrazení přiřazující hraně střední hodnotu času čekání na nabíjecí stanici v čase $t \in \mathbb{R}_0^+$.

5.2 Problém nejkratší cesty s omezením

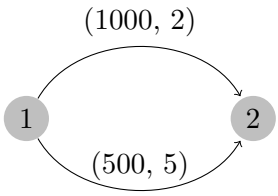
Dijkstrův algoritmus[2] v základní implementaci dokáže nalézt nejkratší cestu při minimalizaci jednoho kritéria. Pro případ plánování nejkratší cesty pro elektrické auto je potřeba uvažovat i baterii. Baterie přináší omezení kladená na nejkratší cestu, která musejí být v každém uzlu cesty splněna. Cesta s neplatným omezením není přípustná cesta a tedy ani přípustné řešení.

Při minimalizaci jednoho kritéria je reprezentace tohoto kritéria jednoduchá. Ve velké většina případů se jedná o číslo, kde porovnání tohoto čísla patří mezi fundamentální matematické dovednosti.

Pokud však kritérium rozšíříme o další prvek, vytvoříme dvojici, v obecném případě n-tici, na které běžné operace porovnání nejsou definované. Je tedy potřeba operace porovnání zadefinovat, abychom mohli tuto n-tici použít.

Dále vznikne s rozšířením z unikritériální optimalizace na multikriteriální

možnost mít více řešení pro jeden daný uzel grafu.



Obrázek 5.1: Příklad dvou multikriteriálních řešení

Vezměme si následující příklad na obrázku 5.1. Mějme dva uzly, počáteční a koncový. A mějme dvě hrany mezi těmito dvěma uzly, každá vedoucí z počátečního uzlu do uzlu koncového. Každá hrana má k sobě přiřazená dvě ohodnocení – první udává spotřebu elektrické energie potřebnou pro překonání hrany a druhá čas potřebný k překonání hrany. Předpokládejme, že první hrana bude vézt po dálnici a druhá povede skrz vesnice po okresních silnicích. Pro překonání první hrany bude potřeba velké množství energie, protože jízda po dálnici je pro elektromobil velice energeticky náročná. Naopak druhá cesta vedoucí přes okresní silnice je energeticky méně náročná, avšak čas potřebný k jejímu překonání je větší než pro první hranu.

Je vidět, že obě dvě cesty tvoří validní řešení. Jedna hrana je časově výhodnější a druhá je energeticky výhodnější. Vzniká zde tak potřeba udržovat pro každý uzel grafu množinu ohodnocení. Tuto množinu budeme nazývat Pareto množinu.

V našem případě se ohodnocení skládá z uspořádané dvojice (t, b) obsahující čas t strávený na cestě od počátečního uzlu do aktuálně vybraného uzlu a aktuální stav nabití baterie b.

Během průchodu algoritmu budou jednotlivé prvky Pareto množin umístěny na prioritní frontu. Aby algoritmus dokázal nejmenší prvek v prioritní frontě, je potřeba zadefinovat pro tuto uspořádanou dvojici operaci porovnání.

Chceme-li vybrat nejmenší ohodnocení (t, b) z množiny, začneme porovnáním jejich prvních prvků. První prvek t představuje čas strávený na cestě. Chceme-li nalézt ten menší, vybereme řešení s nejmenším časem. Pokud však prvky mají shodnou hodnotu u prvního prvku, je potřeba je porovnat pomocí druhého prvku, které udává aktuální stav nabití baterie.

Zde je baterie zajímavým prvkem. Ohodnocení, jehož baterie bude mít nižší stav nabití, je v tomto případě považován za horší, méně výhodné. Chceme-li vybrat nejmenší ohodoncení, je potřeba nalézt to, jehož první prvek (čas) je nejmenší a druhý prvek (baterie) je naopak největší.

Zamyslíme-li se nad počtem možných ohodnocení pro daný uzel, dojdeme k závěru, že takovýchto ohodnocení je exponeciálně mnoho k velikosti grafu. Pokud by plánování probíhalo nad celou touto množinou velice rychle by rostla výpočetní náročnost.

5.2.1 Dominance

Aby bylo vůbec možné upočítat multikriteriální algoritmus, je potřeba si ukládat pouze „zajímavá“ ohodnocení. K tomu slouží pojem **dominance** mezi jednotlivými ohodnoceními. Během plánování budou v rámci Pareto množiny jednoho uzlu ukládána pouze taková ohodnocení, která nejsou dominována žádným jiným ohodnocením z dané Pareto množiny. Hledáme-li nejmenší cestu, prvek v Pareto množině, jehož čas bude horší (větší) než všechny ostatní a současně baterie bude nižší než u všech ostatních, není přínosný.

Mějme tedy dvě uspořádané dvojice $A = (t_1, b_1)$ a $B = (t_2, b_2)$, kde prvky t_1, t_2 představují čas a prvky b_1, b_2 představují stav nabití energie. Říkáme, že prvek A **dominuje** prvek B, pokud platí, že

$$t_1 < t_2 \quad \wedge \quad b_1 >= b_2$$

Pro urychlení běhu algoritmu je vhodné použít upravenou dominanci, kde prvek A dominuje prvek B i v případě, že jsou schodné na všech položkách, tedy že

$$t_1 <= t2 \quad \wedge \quad b_1 >= b_2.$$

Díky tomu zabráníme dalšímu nárůstu Pareto množiny a snížíme tím výpočetní nároky.

5.3 Vývoj ohodnocení při průchodu grafem

Pro každý uzel si ukládáme Pareto set jednotlivých ohodnocení, kde každé z ohodnocení (t, b) obsahuje čas t a stav nabití b. Při expanzi uzlu nedochází přímo k expanzi uzlu, ale vybraného nejmenšího ohodnocení. Vývoj ohodnocení je dán následujícím způsobem.

Mějme elektrický silniční graf G_e, libovolný uzel u, množinu jeho následovníků $V' = \{v_1, v_2, ..., v_n\}$ a aktuální nejmenší ohodnocení (t_{min}, b_{min}) přiřazené uzlu u. Při expanzi uzlu u ve stavu (t_{min}, b_{min}) dojde k vytvoření nových ohodnocení $\{(t_1, b_1), (t_2, b_2), ..., (t_n, b_n)\}$ pro uzly z V' tak, že

$$t_i = t_{min} + \varphi(e_{u,v_i})$$
$$b_i = b_{min} + \varepsilon(e_{u,v_i}),$$

pro všechna $i = 1, 2, ..., n$, kde e_{u,v_i} označuje hranu vedoucí z u do v_i.

Vývoj času řešení při průchodu grafem je přímočarý. Při průchodu pro libovolné hraně e změně dojde k jeho navýšení o hodnotu $\varphi(e)$.

5.4 Vývoj baterie při průchodu grafem

Vývoj baterie je zajímavější. Během průchodu může dojít ke třem možnostem jejího vývoje

1. při přechodu po hraně dojde k jejímu snížení,
2. při přechodu po hraně dojde k jejímu navýšení,
3. při zastavení na nabíjecí stanici a dobití automobilu dojde k jejímu navýšení.

Pokud automobil pojede po rovině nebo do kopce bude při průchodu docházet

k vybíjení baterie, tedy ke snižování aktuálního stavu nabití baterie. Pokud pojede z kopce, bude docházet k rekuperaci energie a baterie se bude částečně dobíjet brzděním.

5.4.1 Funkce spotřeby elektrické energie

Pro první dvě možnosti je potřeba zadefinovat funkci spotřeby elektrické energie. Abychom to mohli udělat, nejprve je ale potřeba zadefinovat množství elektrické energie, o které se změní stav nabití baterie, při průchodu po hraně $e = (u, v)$. Toto množství je závislé na vzdálenosti mezi uzly u a v a na rozdílu jejich nadmořských výšek. Samotné množství elektrické energie může být jak kladné, tak záporné. V případě, že automobil bude mezi uzly u a v cestovat pouze směrem dolů, tedy z vyšší nadmořské výšky do nižší, bude docházet k tzv. rekuperaci, baterie v automobilu se bude touto jízdou z kopce částečně dobíjet. V případě cesty z nižší nadmořské výšky do vyšší bude docházet k vybíjení, jelikož baterie bude automobilu energii dodávat.

Uvažujme nyní dva uzly u a v, nechť nadmořská výška uzlu u bude vyšší než nadmořská výška uzlu v, tedy cesta z uzlu u do uzlu v bude směřovat dolů. Uvažujme nyní dvě cesty, jedna ve z uzlu u do uzlu v a druhou v opačném směru, tedy z uzlu v do uzlu u. Obě cesty mají stejnou délku a jsou pouze opačně orientované. Avšak změny množství elektrické energie, ke kterých na nich dojde, nebudou v absolutních hodnotách stejné. Pokud by to byla pravda, získali bychom *perpetuum mobile*.

Zadefinujme **funkci spotřeby elektrické energie** jako zobrazení $\varepsilon : E \to \mathbb{R}$, které každé hraně $e = (u, v)$ přiřazuje hodnotu spotřeby elektrické energie při cestě z uzlu u do uzlu v. Hodnota spotřeby je určena následujícím způsobem

$$\varepsilon(u, v) = \begin{cases} \kappa \cdot \delta(e) + \lambda \cdot \Delta elev_e, & \text{pokud } \Delta elev_e \geq 0, \\ \kappa \cdot \delta(e) + \alpha \cdot \Delta elev_e, & \text{jinak} \end{cases}$$

kde κ, λ a α jsou ladící parametry algoritmu, $\Delta elev_e = elev_u - elev_v$ určuje výškový rozdíl nadmořských výšek mezi uzly u a v a $\delta(e)$ určuje velikost (vzdálenost) hrany e.

5.4.2 Nabíjení

Třetí případ vývoje batrie je nabíjení baterie na nabíjecích stanicích. Nabíjení je na grafu G_e modelováno jako smyčka na uzlu. Při průchodu po této smyčce se zvýší čas strávený na cestě a současně se i zvýší stav nabití baterie v automobilu.

Mějme ohodnocení (t, b), kde t označuje čas ohodnocení, b stav nabití baterie tohoto řešení, nabíjecí stanici v a nový stav pro nabití baterie b_{new}. Nabíjením označujeme přechod k novému ohodnocení (t', b') tak, že

$$t' = t + cht(v, t, b_{new})$$
$$b' = b_{new}$$

Jelikož při nabíjení dochází kromě času i k navýšení stavu nabití baterie, nebude toto ohodnocení dominováno předchozími ohodnoceními.

Nabíjecí stanice může nabíjet baterii v automobilu určitým výkonem. Tento výkon je zadefinován jako ch_{max}. V našem případě bude hodnota ch_{max} nastavena pro všechny stanice stejně. Avšak v rámci zpřesnění výsledků by bylo vhodné tuto hodnotu určit pro každou nabíjecí stanici zvlášť.

5.5 Nejkratší cesta a optimální začátek cesty s neurčitostí

Oba naše problémy v sobě obsahují neurčitost představující dobu čekání na nabíjecí stanici. Prvním a jednodušším problémem s neurčitostí je nalezení optimální cesty a optimálního času odjezdu při daném cílovém času. Optimální cesta je v našem problému myšlena cesta s nejmenší variancí. Optimální cesta lze v tomto kontextu vyjádřit i jako nejjistější cesta. Optimalizace variance cesty znamená, že čas realizace cesty by se měl lišit co nejméně od času naplánované cesty.

Abychom mohli reprezentovat neurčitost v tomto problému je potřeba rozšířit ohodnocení na uzlech grafu. Předchozí ohodnocení mělo tvar (t, b), kde t je čas strávený na cestě a b je stav nabití baterie. Toto ohodnocení rozšíříme o varianci a získáme nové ohodnocení jako uspořádanou trojici (σ^2, t, b).

V problému budeme minimalizovat cenu cesty danou funkcí

$$C(t) = (t_{end} - t)^2,$$

kde čas dojezdu je pro jednoduchost a z důvodu následující úvahy na základě článku [23] označen $t_{end} = 0$. Uvažujme nyní, že cesta z počátečního uzlu do koncového obsahuje pouze jednu hranu, a čas cesty přes tuto hranu je náhodná proměnná s hustotou pravděpodobnosti $f(\cdot)$ a variancí σ^2. Potom střední hodnotu ceny cesty získáme z článku [23] jako

$$\begin{aligned} EC(t) &= \int_0^\infty f(y)(t+y)^2 \,\mathrm{d}y \\ &= t^2 + 2tE[Y] + E[Y^2] \\ &= (t+\mu)^2 + \sigma^2 \end{aligned}$$

Pokud cesta bude obsahovat n hran, kde čas cesty každé hrany bude nezávislá náhodná proměnná s hustotou $f_i(\cdot)$ a variancí σ_i^2 získáme střední hodnotu cenu cesty z článku [23] jako

$$EC(t) = \left(t + \sum_{i=0}^{n} \mu_i\right)^2 + \sum_{i=0}^{n} \sigma_i^2$$

Tedy cena cesty je minimalizována v $t = \sum_{i=0}^{n} \mu_i$ a v tomto optimu je střední cena cesty rovna $EC_{min} = \sum_{i=0}^{n} \sigma_i^2$.

K nalezení cesty s nejmenší variancí lze použít Dijkstrův algoritmus, kde ohodnocení na hranách budou odpovídat rozptylu pravděpodobnostního rozdělení, které je k dané hraně přidružené. Výsledná cesta bude mít optimální

(minimální) varianci a optimální start cesty bude dán střední hodnotou jednotlivých časů příslušných hran. Při použití kvadratické ceny cesty a obecného pravděpodobnostního rozdělení je možné optimální cestu získat využitím deterministického algoritmu pro nalezení nejkratších cest. Důkaz lze nalézt v [23].

V našem problému budou náhodné proměnné modelovat pouze čas čekání na nabíjecí stanici, než se uvolní místo a bude možné auto nabíjet.

5.5.1 Poptávka po nabíjecích stanicích

Celkový čas, který automobil stráví na nabíjecí stanici se neskládá pouze z času potřebného pro samotné nabití baterie na danou úroveň. Je potřeba také počítat s tím, že na dané nabíjecí stanici mohou být všechna nabíjecí místa obsazena dříve příchozími zákazníky.

Abychom mohli čas čekání na uvolnění nabíjecího místa modelovat co nejvěrněji, je potřeba zkoumat poptávku po nabíjecích stanicích v jednotlivých časových intervalech během dne.

Model poptávky po nabíjecích stanicích je naučený model z projektu ElaadNL Open Datasets for Electric Mobility Research[1].

Poptávka je funkce

$$\pi : V \times \mathbb{R} \to \mathbb{R},$$

jejíž parametry jsou nabíjecí stanice a časový okamžik t ve kterém nás poptávka po danou nabíjecí stanici zajímá. Výsledkem poptávkové funkce je procentuální hodnota aktuálně běžících nabíjení v čase t vzhledem k celkovému počtu všech nabíjení během dne.

Jelikož je čekání modelováno pomocí Gamma rozdělení, je potřeba získat dva parametry, abychom mohli toto rozdělení správně sestavit. Pro parametr k

[1] https://platform.elaad.io/analyses/ElaadNL_opendata.php

využijeme předchozí funkci poptávky, jejíž výsledek bude použit právě jako tento parametr.

Pro získání parametru θ je potřeba zadefinovat další funkci. **Rozptyl poptávky** definujeme jako funkci

$$\sigma : V \times \mathbb{R} \to \mathbb{R}$$

Funkce přebírá nabíjecí stanici, jejíž poptávkový rozptyl nás zajímá a daný časový okamžik.

Spojením poptávkové funkce a funkce rozptylu poptávky získáme dva parametry, se kterými lze sestavit Gamma rozdělení a využít jeho střední hodnoty μ a rozptylu σ.

Nyní již máme zadefinované nejdůležitější součásti pro plánování pro elektrické automobily a můžeme zadefinovat rozšíření silničního grafu pro toto plánování s využitím modelu elektrického automobilu.

Hodnoty parametrů k a θ byly experimentálně určeny.

5.6 Pseudopolynomiální algoritmus pro výpočet optimální cesty při daném začátku cesty

Druhým problémem, který řešíme v práci, je problém nalezení optimální cesty při pevně daném startu. Pro řešení tohoto problému jsme vytvořili algoritmus využívající dynamického programování, který závisí polynomiálně na největší střední hodnotě času cesty M vycházející algoritmu v článku [23].

5.6.1 Základní verze algoritmu

Algoritmus v článku [23] slouží k výpočtu optimální cesty při daném začátku cesty. Pro potřeby naší práce je však potřeba ho rozšířit o zpracování baterie a

o nabíjení. Základní verze algoritmu ve své hlavní smyčce postupně pro každý časový okamžik m od $1, ..., M$ a pro každý uzel u vypočítává nejspolehlivější cestu se střední dobou m, kterou se lze dostat do daného uzlu u.

Algorithm 2: Základní verze algoritmu pro nalezení optimální cesty při daném začátku cesty

Φ(S, 0) = 0;
π(S, 0) = S;
forall $v \in V$ **do**
  if *v is a neighbor of S and* $\mu_{sv} = 1$ **then**
    Φ(v, 0) = σ^2_{sv};
    π(v, 0) = S;
  else
    Φ(v, 0) = null;
    π(v, 0) = null;
  end
end
for $m = 1$ *to* M **do**
  forall $v \in V$ *with neighbors v'* **do**
    $\Phi(v, m) = \min_{v' \sim v}[\Phi(v', m - \mu_{v'v}) + \sigma^2_{v'v}]$;
    $\pi(v, m) = \arg\min_{v' \sim v}[\Phi(v', m - \mu_{v'v}) + \sigma^2_{v'v}]$;
  end
end
$m_{opt} = argmin_{m \in \{0,...,M\}}\{(t + m)^2 + \Phi(T, v)\}$;
$\pi_{opt} = \pi(T, m_{opt})$;
$EC_{min}(t) = (t + m_{opt})^2 + \Phi(T, m_{opt})$;

Algoritmus během svého běhu využívá dvě tabulky

1. $\pi(v, m)$ určující předchůdce uzlu v na cestě z počátečního uzlu S se střední hodnotou m a nejmenší variancí,
2. $\Phi(v, m)$ určující varianci cesty v uzlu u na cestě z počátečního uzlu S se střední hodnotou m.

Iniciální nastavení pro běh algoritmu jsou dány pro počáteční uzel S následovně

$$\Phi(S,0) = 0 \quad \wedge \quad \pi(S,0) = S.$$

Hlavní smyčka iteruje přes všechna $m = 1, ..., M$, kde M je největší střední hodnota času cesty.

Během každé iterace algoritmus hledá pro každý uzel v jeho nejvhodnějšího předchůdce v', který minimalizuje varianci cesty do uzlu v se střední dobou m

$$\Phi(v,m) = \min_{v' \sim v} \left[\Phi(v', m - \mu_{v'v}) + \sigma^2_{v'v} \right]$$
$$\pi(v,m) = arg \min_{v' \sim v} \left[\Phi(v', m - \mu_{v'v}) + \sigma^2_{v'v} \right]$$

Po dokončení běhu algoritmu se hledá optimální hodnota parametru m minimalizující cenu cesty pro daný koncový uzel T a pro čas odjezdu t

$$m_{opt} = arg \min_{m \in \{0,...,M\}} \left\{ (t+m)^2 + \Phi(T,m) \right\}$$

Po nalezení m_{opt} už lze jednoduše pomocí tabulky předchůdců $\pi(v,m)$ sestavit celou naplánovanou cestu.

5.6.2 Rozšíření pro elektrické automobily

Aby mohl předchozí algoritmus fungovat správně v kontextu plánování cest pro elektrické automobily, je potřeba jeho výpočet rozšířit. Algoritmus musí uvažovat omezení plynoucí z baterie v automobilu a musí umět pracovat s nabíjením baterie na nabíjecích stanicích.

Přidání baterie do výpočtu však není přímočaré. Pokud bychom přidali baterii pouze jako další tabulku, která bude určovat její hodnotu v uzlu v na cestě se střední hodnotou m a nejmenší variancí, nedokážeme tímto přístupem modelovat správně situace, kdy pro aktuálně počítané pole tabulky budeme

mít z předcházející iterace dvě hodnoty, kde jedna z nich bude mít velkou varianci a vysoký stav nabití baterie a druhou s malou varianci ale současně i malý stav nabití baterie. Vzniká zde tak potřeba mít pro každé pole tabulky Pareto množinu možných hodnot, kde položka v množině bude obsahovat uspořádanou trojici (σ^2, t, b), kde σ^2 je variance ohodnocení, t je střední hodnota času cesty a b je stav nabití baterie.

Nastavení algoritmu je potřeba upravit následovně. Pro běh algoritmu jsou vytvořeny dvě tabulky:

1. $\pi(v, m, i)$ určující předchůdce uzlu v na cestě z počátečního uzlu S se střední hodnotou m a pro položku z Pareto množiny i,
2. $\Phi(v, m, i)$ určující varianci cesty, střední hodnotu času cesty a aktuální stav nabití baterie pro položku z Pareto množiny i v uzlu u na cestě z počátečního uzlu S se střední hodnotou m.

Nadále již není možné vybírat pouze jednoho předchůdce, který minimalizuje varianci do aktuálně zpracovávaného uzlu. Pro každý uzel v na cestě se středního hodnotou m je potřeba vybrat Pareto množinu ohodnocení ze všech jeho předchůdců v', kde jednotlivé položky mají čas a baterii posunuté o velikost hrany z daného uzlu do aktuálního.

Druhé potřebné rozšíření je nabíjení. Nabíjení je na grafu modelováno jako smyčka na uzlu, během které se čas posune o dobu, kterou zákazník čeká na nabíjecí stanici, než se uvolní místo, a o dobu potřebnou k nabití automobilu. Aby byl algoritmus schopný s nabíjením pracovat, je potřeba při zpracování Pareto množiny $\Phi(v, m)$ uzlu v na cestě se střední dobou m posunout jednotlivé položky množiny o čas t potřebný k nabití baterie do stavu nabití b.

Mějme položku i z Pareto množiny $\Phi(v, m, i)$ uzlu v na cestě se střední hodnotou m jako uspořádanou dvojici $(\sigma^2_{curr}, t_{curr}, b_{curr})$. Nechť t_{ch} je čas potřebný k nabití baterie do stavu nabití b_{new}. Potom nabití položky i znamená

$$\Phi(v, m + t_{ch}, i) = (\sigma^2_{curr}, t_{curr} + t_{ch}, b_{new})$$

Neurčitost v našem problému je reprezentována pouze na nabíjecích stanicích, kdy nevíme, jak dlouho bude zákazník na dané nabíjecí stanici čekat. Čekání na nabíjecí stanici tak lze zakomponovat do procesu nabíjení.

Mějme uzel v, který je nabíjecí stanicí. Nechť $\sigma^2_{v,m}$ je rozptyl času čekání na dané stanici v na cestě se střední hodnotou m, nechť $\mu_{v,m}$ je střední hodnota tohoto času a nechť $(\sigma^2_{curr}, t_{curr}, b_{curr})$ je i-tá položka Pareto množiny $\Phi(v, m, i)$. Potom čekat na nabíjecí stanici znamená

$$\Phi(v, m + \mu_{v,m}, i) = (\sigma^2_{curr} + \sigma^2_{v,m}, t_{curr} + \mu_{v,m}, b_{curr})$$

5.7 Zrychlení běhu algoritmů

Oba problémy a jejich algoritmy jsou multikriteriální. To znamená, že v každém časovém okamžiku během běhu algoritmu je pro daný stav přidružena množina možných stavů, kde každý z nich může být součástí optimálního řešení. Algoritmy zpracovávající více kritérii jsou náročné na výpočetní a paměťové nároky.

5.7.1 ε-relaxace

ε-relaxace [26] je zrychlující technika běhu multikriteriálního algoritmu. ε se používá při určování dominance Pareto množin, kde definuje na kolik procent jsou si jednotlivé prvky Pareto množiny podobné.

Mějme dvě uspořádané dvojice $A = (t_1, b_1)$ a $B = (t_2, b_2)$, kde prvky t_1, t_2 představují čas a prvky b_1, b_2 představují stav nabití energie a mějme $\varepsilon \in \langle 0, 1 \rangle$. Říkáme, že prvek A **ε-dominuje** prvek B, pokud platí, že

$$t_1 < t_2 \quad \wedge \quad b_1 >= b_2 * \varepsilon$$

Je vidět, že rozdíl oproti základní definici dominance je v tom, že druhý prvek,

zde představující stav nabití baterie, je relativně posunut o hodnotu ε. Díky tomu dojde k dominanci i u prvků, u nichž striktní dominance není. Současně s pomocí ε-dominance dojde k dominanci na velice blízkáých ohodnoceních. Důsledkem toho je snížení výpočetních nároků díky zmenšení velikost Pareto množin na úkor možného zhoršení možného zhoršení optimálního řešení.

Kapitola 6

Implementace

V této kapitole popíšeme implementační detaily práce. Samotná práce se skládá ze 3 částí

1. převedení dat z OSM formátu do interního formátu,
2. předzpracování interního formátu pro potřeby řešení problémů,
3. samotná implementace algoritmů.

6.1 Převedení dat z OSM formáto do interního formátu

Samotná implementace práce je napsána v jazyce Swift. Bohužel v současné době neexistuje knihovna pro práci s daty z OpenStreetMap napsaná v tomto jazyce. Z tohoto důvodu byl naimplementován převodník v jazyce C++, který surová data z OpenStreetMap převedl do interního formátu, který lze dále lehce zpracovat libovolným jazykem.

V tomto převedení byly z dat odstraněny nepotřebné informace o uzlech a hranách, které nejsou podstatné pro řešení problémů plánování.

6.2 Předzpracování interního formátu pro potřeby řešení problémů

Náš interní formát byl dále zpracováván. Toto zpracování už bylo napsáno v jazyce Swift. Nad daty byl vytvořen skript, jehož výsledkem je předzpracovaný graf sestavený pouze z nabíjecích stanic a jednotlivé napočítané cesty mezi těmito stanicemi. Jelikož se jedná o poměrně velké množství dat, z důvodu paměťových nároků bylo po několika iteracích mezi různými formáty zvolen binární formát pro uložení dat. Díky tomu se zmenšila datová stopa uložených dat a jejich načítání v programu se zrychlilo.

6.3 Implementace algoritmů

Implementace algoritmů byla také provedena v jazyce Swift. Swift je moderní a velice mladý jazyk. Je to jeho výhoda, ale i nevýhoda. Jeho použití pro akademické účely je poměrně obtížné, protože pro něj neexistují žádné udržované knihovny obsahující základní datové strukty běžně používané v algoritmech. Během implementace tak byla vytvořena pro účely práce vlastní implementace haldy a prioritní fronty.

Druhým velkým omezením byly možnosti spuštění evaluace. Jelikož Swift je silně spjatý s Apple počítači, nelze naší implementaci sestavit na žádném jiném operačním systému než na operačním systému macOS přímo od společnosti Apple. Nešlo tak využít možností serverů školy.

Implementace byla rozdělena do dvou základních modulů:

1. interní modul obsahující implementaci multikriteriálního algoritmu a algoritmu pro řešení optimální cesty s daným startem
2. hlavní spuštěcí modul, ve kterém docházelo k implementaci jednotlivých scénářů s využitím tříd z interního modulu.

Toto rozdělení bylo potřeba z důvodu testování. Abychom mohli vytvořit unit

testy pro jednotlivé třídy, bylo nutné je oddělit do samostatného interního modulu, pro který lze vytvořit testovací modul.

Kapitola 7

Evaluace

V následující kapitole ukážeme výsledky implementace algoritmů pro oba zkoumané problémy. Výsledky obou algoritmů budou porovnávány se základní implementací problému nejkratších cest, která však nebude uvažovat rozptyl času čekání na nabíjecí stanici, pouze střední dobou čekání.

7.1 Zpracování vstupních dat

7.1.1 Mapové podklady

Pro evaluaci algoritmu jsme rozhodli použít silniční síť Bavorska. Tato síť je dostatečně rozsáhlá k testování větších instancí problémů a současně není příliš rozlehlá, aby jednotlivé instance bylo možné upočítat v reálném čase. Data o silniční síti byla získána z projektu OpenStreetMap[1], který data poskytuje volně k dispozici. Jednotlivé údaje o silniční síti jsou doplňovány přímo koncovými uživateli, díky čemuž nejsou data vždy a všude plně kompletní. Záleží na uživatelích a správcích, která data do projektu vloží a která ne.

Silniční síť je do grafu převedena velice podrobně. Uzly grafu silniční sítě představují významné body. Může se jednat o křižovatku dvou silnic, přechod pro chodce, železniční přejezd, změnu mezi dvěma typy silnic apod. Uzly

[1] https://www.openstreetmap.org/

jsou však využívány i k přesnému modelování vzhledu silnice. Tedy pokud je v silniční síti zatáčka, tak tento úsek je modelován několika uzly, aby při vykreslení mohlo dojít k co nejpřesnější vizualizaci úseku. Převedení silniční sítě do grafu s sebou přináší velké požadavky na paměť. Avšak spousta z těch významných uzlů, či uzlů zachycujících tvar silniční sítě, jsou pro potřeby plánování nepodstatné.

Pokud uvažujeme plánování trasy, není potřeba uvažovat tvar jednotlivých úseků. Stačí jednotlivé nezajímavé úseky nahradit zkratkou, která nepodstatný uzel přeskočí, a napočítat si data potřebná pro plánování. K našemu plánování je potřeba znát délku trasy, maximální rychlost na trase a množství elektrické energie potřebné k překonání daného úseku.

7.1.2 Určení nadmořské výšky

Ve funkci spotřeby elektrické energie 5.4.1 je potřeba znát rozdíl nadmořských výšek $\Delta elev_e$ mezi jednotlivými uzly. Tuto informaci ale velké množství uzlů z OpenStreetMap nemá. K vypočtení nadmořské výšky jsme použili další otevřený projekt Open-Elevation[2]. Projekt poskytuje veřejné aplikační rozhraní (API), pomocí kterého lze pro zadanou zeměpisnou souřadnici získat přibližnou nadmořskou výšku. Pro získání velkého množství informací poskytují veřejně používaný dataset[3]. Koncový uživatel si pak může vytvořit vlastní instanci projektu na lokálním serveru. Díky čemuž se velice zrychlí získávání dat z datasetu a současně není potřeba připojení k internetu.

7.1.3 Redukce grafu silniční sítě

Pro zmenšení velikosti grafu silniční sítě je potřeba zadefinovat uzly, které jsou podstatné pro naše plánování. Přechody pro chodce, železniční přejezdy, ani uzly sloužící k vykreslení zatáčky potřeba nejsou. Podstatné uzly jsou pouze křižovatky a uzly reprezentující změnu typu silnice, např. vjezd do obce z okresní silnice. Při změně typu silnice dochází ke změně maximální

[2] https://open-elevation.com
[3] https://www2.jpl.nasa.gov/srtm/

dovolené rychlosti a znalost maximální rychlosti je potřeba k vypočtení času potřebného k překonání daného úseku.

Díky této redukci se podařilo z grafu o několika miliónech uzlů vytvořit zmenšenou verzi vhodnou pro naše plánovací algoritmy o velikosti přibližně 100 000 uzlů.

7.1.4 Nabíjecí stanice

Poloha nabíjecích stanic není z OpenStreetMap dostupná. K získání této informace bylo potřeba stáhnout dataset z projektu OpenChargeMap[4]. Dataset obsahuje pozice nabíjecích stanic a základní informace o nich. Neposkytuje však informace o aktuální obsazenosti stanice.

Informace o nabíjecích stanicích rozšířily současné informace v redukovaném grafu. Pro každou nabíjecí stanici byl nalezen nejbližší uzel v grafu a ten byl potom prohlášen za nabíjecí stanici.

Počet nabíjecích stanic na území Bavorska je přibližně 2000.

7.1.5 Redukce stavového prostoru

I přes všechna předchozí zmenšení stavového prostoru, pro pseudopolynomiální algoritmus nalezení optimální cesty při pevně daném začátku cesty, jehož paměťová náročnost je lineárně závislá na počtu uzlů grafu a horním odhadu M, bylo potřeba graf zredukovat na úplné minimum. Výsledný zredukovaný graf obsahuje pouze nabíjecí stanice a napočítané cesty mezi nimi.

Příkladem může být cesta přes Bavorsko. Uvažujme nyní unikriteriální optimalizaci, kde nás bude zajímat pouze nějaká jedna hodnota, která bude reprezentována desetinným číslem o velikosti 8 bajtů. Pokud uvažujeme graf obsahující 100 000 uzlů a horní odhad střední doby cesty $M = 1000$, získáme velikost paměťového prostoru potřebného k uchování a vypočítání této informace jako

[4] https://openchargemap.org/site

$$100\,000 * 1\,000 * 8 = 800\,000\,000 \text{ B} = 800 \text{ MB}$$

Nyní uvažujme, že algoritmus není unikriteriální ale multikriteriální, tedy že v každém uzlu může dojít až k exponenciálnímu nárůstu potřebné paměti, z potřebných 800 MB se tak velice lehce může stát několik desítek až stovek GB.

7.1.6 Problém první a poslední míle

Aby bylo možné plánovat trasu i mimo nabíjecí stanice, jsou na začátku každého plánování ke grafu připojeny dva uzly – počáteční a koncový. Jejich připojení je vytvořeno pomocí Dijkstrova algoritmu[2], kterým jsou napočítány jejich cesty k ostatním uzlům grafu. Z počátečního uzlu je spuštěno dopředné vyhledávání a z koncového zpětné. Získáme tak nejkratší cesty ke všem ostatním uzlům.

Zpětné vyhledávání je také použito jako hodnota heuristické funkce v základním algoritmu pro hledání nejkratších cest.

7.2 Nastavení algoritmu

Oba plánovací algoritmy obsahují množství parametrů, které je potřeba před spuštěním plánováním nastavit tak, aby byl zajištěn co nejhladší běh algoritmu.

7.2.1 Model elektromobilu

Jako model automobilu lze vzít libovolný elektromobil a nastavit jeho hodnoty (b_{min}, b_{max}). Pro naše problémy jsme zvolili kopii elektromobilu e-Golf, jehož baterie má kapacitu 32 kWh. Dojezd tohoto automobilu je kolem 220 km. Minimum, pod které nesmí klesnout nabití baterie, bylo stanoveno na 0.

Dovolíme tedy automobilu jet až do úplného vyčerpání. Díky plánování pouze mezi nabíječkami to není problém.

7.2.2 Nabíjecí stanice

Nabíjecí stanice uvažujeme všechny stejně výkonné s lineární nabíjecí funkcí. Nabíjecí stanice pracují s výkonem 24 kW.

7.2.3 Funkce spotřeby elektrické energie

Funkce spotřeby elektrické energie je převzatá včetně jejich ladících parametrů z [27], kde ladící parametry mají hodnotu

$$\kappa = 0.2, \quad \lambda = 2, \quad \alpha = 1.5.$$

7.2.4 Mapové podklady

Jednotlivé segmenty silniční sítě mohou, ale nemusejí mít specifikovanou jejich maximální povolenou rychlost. Pokud tata informace v datech chybí, jednotlivé rychlosti jsou přiřazeny podle tabulky 7.1.

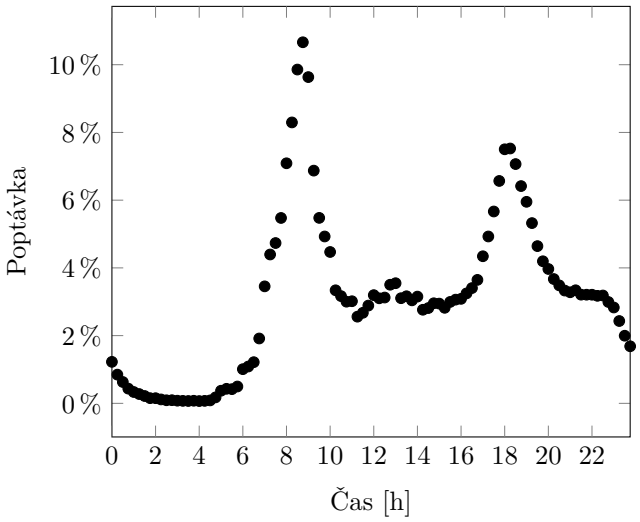
Typ silnice	Maximální rychlost v km/h
Motorway	130
Motorway link	50
Trunk	110
Trunk link	50
Primary	90
Primary link	50
Secondary	50
Secondary link	50
Tertiary	50
Tertiary link	50

Tabulka 7.1: Maximální rychlosti podle typu silnice z OSM[5]

[5] https://wiki.openstreetmap.org/wiki/Key:highway

7.2.5 Poptávka

Poptávka po nabíjecích stanicích je závislá na čase. Model poptávky byl vytvořen na základě reálných dat[6]. Procentuální rozdělení poptávky během dne je vidět na obrázku 7.1.



Obrázek 7.1: Model poptávky během dne

Hodnota poptávky v daném čase slouží jako parametr k Gamma rozdělení času čekání na nabíjecí stanici. Druhý parametr tohoto rozdělení je dán typem nabíjecí stanice. V práci jsou celkem 4 typy nabíjecích stanic. Každá stanice má přidělený jeden typ na základě její pozice v rámci silniční sítě. Jednotlivé typy stanic jsou vidět v tabulce 7.2. Spojením typu nabíjecí stanice a aktuálního času získáme parametry k a θ Gamma rozdělení, ze kterých lze vypočítat střední hodnotu a rozptyl rozdělení. Rozdělení nabíjecích stanic do těchto 4 typů probíhá na základě hodnot z okolních uzlů. Kde například okolní uzly u nabíjecí stanice, které mají v průměru přidělenou maximální rychlost přesahující 110 km / h, jsou velice pravděpodobně blízko dálnice. Taková nabíjecí stanice je pak označena typem „Stanice na dálnici“.

Určení jednotlivých používaných hodnot bylo určeno experimentálně.

[6] https://platform.elaad.io/analyses/ElaadNL_opendata.php

Typ nabíjecí stanice	Hodnota θ
Malá stanice ve městě	1
Velká stanice ve městě	2
Stanice u okresní silnice	0.5
Stanice na dálnici	1.5

Tabulka 7.2: Typy nabíjecích stanic a přidružené hodnoty parametry θ

7.3 Základní algoritmus pro porovnání

Během evaluace budeme používat základní algoritmus pro hledání nejkratších cest. Tento algoritmus uvažuje během plánování omezení kladená na baterku. Neuvažuje ale neurčitost. Pokud dorazí na nabíjecí stanici, hodnota času je posunuta o střední hodnotu času čekání pro daný čas příjezdu. Současně je během plánování využita heuristická funkce, která pomáhá výpočet algoritmu urychlit. Jako heuristická funkce se berou napočítané vzdálenosti z řešení problému Poslední míle 7.1.6. Díky tomu algoritmus získá velice přesný odhad o vzdálenosti z daného uzlu do koncového.

7.4 Experimenty

Všechny experimenty byly spuštěny na vlastním serveru, jehož technická specifikace je možná vidět v tabulce 7.3. Využití vlastního výpočetního stroje bylo způsobeno výběrem programovacího jazyka a knihoven, pomocí nichž byla práce naprogramována.

Procesor	i7-6700k 4 GHz
Počet jader	4 (8)
RAM	16 GB DDR4

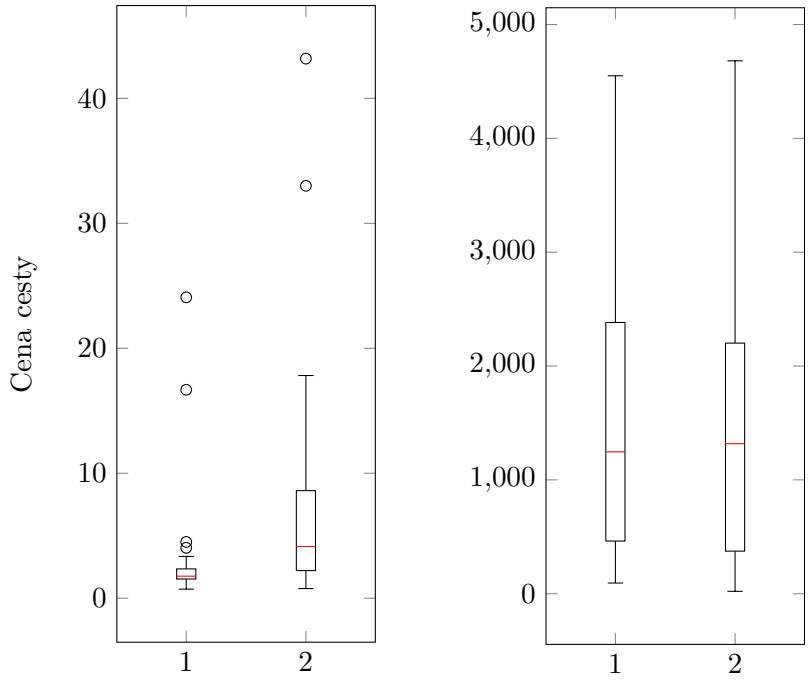
Tabulka 7.3: Hardwarová specifikace testovacího stroje

Každý z experimentů byl prováděn jako porovnání základního algoritmu pro nalezení nejkratších cest grafu a našeho algoritmu pro vybraný problém.

7.4.1 Optimální cesta při daném začátku cesty na náhodných datech

Prvním experimentem je porovnání výsledků algoritmu pro nalezení optimální cesty při daném začátku cesty z 5.6. Porovnávaným kritériem byla střední hodnota cena cesty $EC(t) = (t_{end} - t_{start} + \sum_i \mu_i)^2 + \sum_i \sigma_i^2$.

Porovnání probíhalo na 100 náhodně vygenerovaných scénářích. Každý scénář obsahoval počáteční uzel a koncový uzel. Čas začátku cesty byl náhodně rovnoměrně distribuovaný v rozmezí mezi 7:00 a 9:00. Konec cesty byl odhadnut jako čas potřebný pro překonání celé cesty vypočítaný Dijkstrovým algoritmem spuštěným zpětně z koncového uzlu a doby potřebné k nabití baterie do stavu, kdy by teoreticky bylo možné přejet celou cestu bez nabíjení. Tento výsledný koncový čas byl opět náhodně posunut v rozmezí ± 1 hodina.



(a) : Spodní polovina cen cest **(b) :** Horní polovina cen cest

Obrázek 7.2: Porovnání našeho algoritmu (1) a základního modelu (2) na náhodných datech

Z důvodu použití kvadratické ceny bylo dobré pro lepší vizualizaci rozdělit výsledky na dvě části. Výsledky byly seřazené vzestupně podle hodnoty ceny cesty a rozděleny na půl. Díky tomu se podařilo oddělit scénáře, jejichž

koncový čas měl vysokou pravděpodobnost splnění, neboli cena cesty se blížila nule, a scénáře jejichž splnitelnost je méně pravděpodobná a hodnota ceny cesty v těchto scénářích se pohybovala v řádek stovek až tisíců.

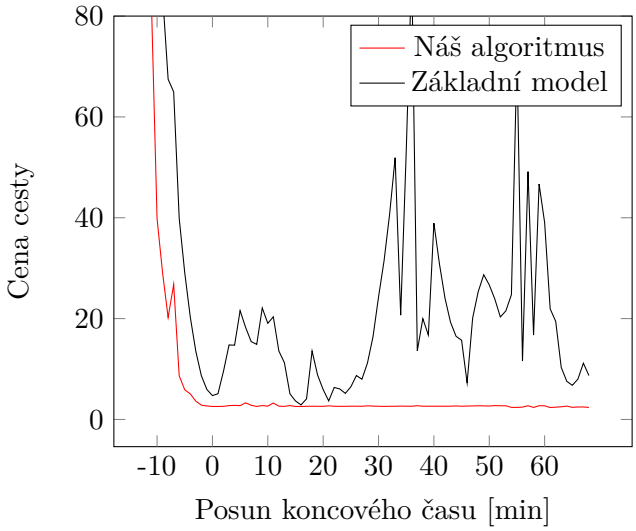
Z grafu 7.2 je vidět, že v případě splnitelného cílového času, náš algoritmus vrací výrazně lepší výsledky na náhodných datech. V případě obtížně splnitelného cílového času, tedy scénáře, jenž měl cílový čas příliš blízko počátečnímu času, se náš algoritmus zachoval velice podobně jako základní model pro nalezení nejkratší cesty. Z tohoto grafu lze vyvodit, že při nastavení cílového času velmi blízko času odjezdu se náš algoritmus zachová stejně jako nalezení základní algoritmus pro nalezení nejkratší cesty.

7.4.2 Vliv koncového času na optimální cestu při daném začátku cesty

Druhým experimentem je porovnání vlivu koncového času na optimální cestu při daném začátku cesty. Předpoklad je, že pokud velmi přiblížíme koncový čas k času začátku cesty, bude se náš algoritmus chovat spíše jako algoritmus pro vyhledání nejkratší cesty. Pokud se koncový čas přehoupne přes hranici odpovídající času nejkratší cesty, pak by měl náš algoritmu základní model vždy překonat. To je dáno tím, že náš algoritmus je schopný nalézt i nejdelší cestu. Kdežto základní model dokáže nalézt pouze Pareto množinu řešení pro problém nejkratších cest.

Experiment byl spuštěn na vybrané trase napříč Bavorskem. Během cesty je potřeba alespoň 3x nabíjet. Trvání cesty je přes 10,5 hodiny. Začátek cesty byl nastaven na 8:00. Koncový čas byl postupně posouván okolo času nejkratší cesty.

V grafu 7.3 jsou vidět absolutní hodnoty ceny cesty a je vidět, že náš předpoklad byl správný. Čas nejkratší cesty je vyznačen jako 0 na ose X. Do hodnoty 0 se náš algoritmus chová velice podobně jako základní model a od 0 dále je schopný nalézt řešení, jehož cena je velmi blízká 0. Základní model má od 0 dále jasně viditelný nárůst ceny cesty. Současně ale nestoupá cena cesty pro základní model kvadraticky, což by se mohlo na první pohled zdát jako špatně. Vývoj od 0 napravo pro základní model je dán tím, že nalezl Pareto



Obrázek 7.3: Evaluace vlivu koncového času na cenu optimální cesty při daném začátku cesty

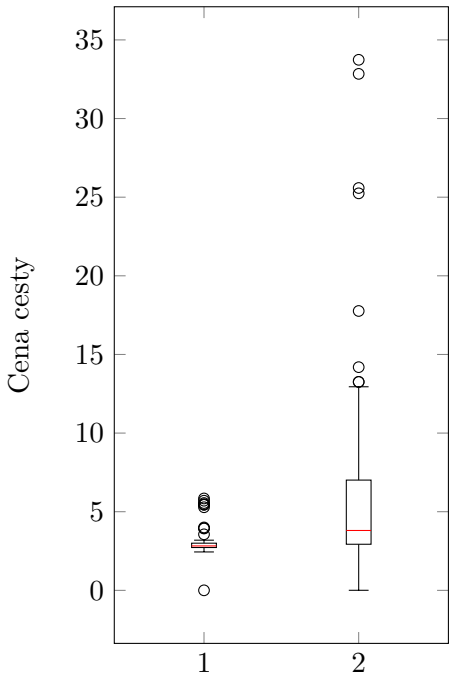
množinu řešení, tedy jednotlivým řešením z Pareto množiny se postupně se zvyšujícím se koncovým časem snižovala jejich cena, což způsobila takovýto vývoj grafu. To je důvod, proč pravá strana od 0 nestoupá kvadraticky.

7.4.3 Optimální cesta a čas cesty

Třetím experimentem a prvním experimentem pro problém nalezení optimální cesty a optimálního začátku cesty bylo nalezení ceny cesty pro náhodně vygenerované scénáře. Scénáře byly generovány tak, aby na cestě bylo nutno alespoň jednou baterii v elektromobilu dobít. Výsledek byl porovnán se základním modelem, který našel pouze nejkratší cestu a nebere v úvahu rozptyl na hranách.

Bylo vygenerováno 100 scénářů, pro každý scénář byl nalezeno řešení naším algoritmem a základním modelem. Toto řešení bylo následně 1000krát realizováno. Výsledkem pro každý scénář byla průměrná cena cesty z 1000 realizací. Rozložení hodnot je vidět v grafu 7.4.

Z grafu 7.4 je vidět, že medián obou dvou řešení je velmi podobný, náš algoritmus je o trošku lepší. Pokud se však podíváme na rozptyl hodnot, je vidět, že náš algoritmus velice překováná základní model. V absolutních



Obrázek 7.4: Porovnání našeho algoritmu (1) a základního modelu (2) na náhodných datech

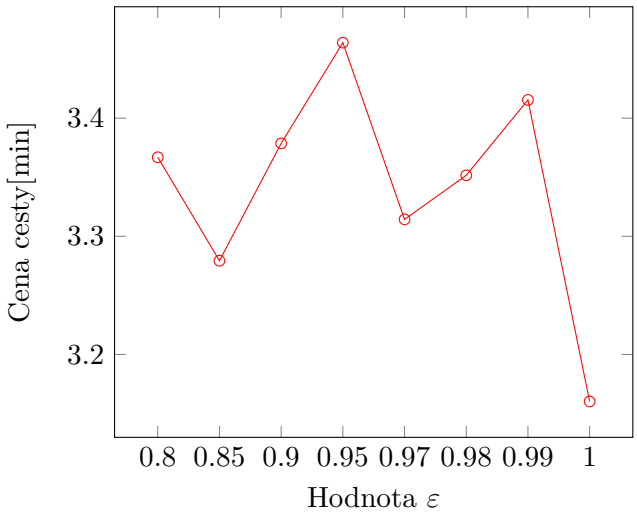
hodnotách dokázal náš algoritmus překonat základní model v 80 % případů, tedy v 80 % průměrná hodnota ceny cesty byla lepší u základního modelu.

7.4.4 Vliv ε-relaxace na cenu cesty a výpočetní čas

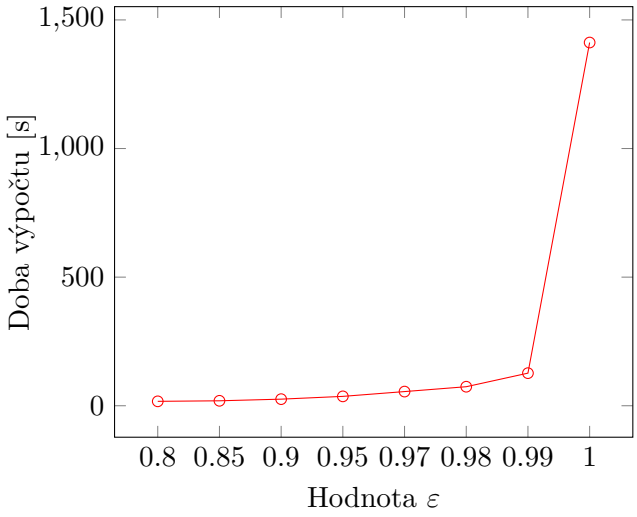
Poslední experiment ukazuje vliv hodnoty ϵ na celkovou cenu cesty a čas výpočtu. Experiment byl spuštěn na 10 náhodně vybraných trase v Bavorsku. Během cesty je potřeba alespoň 2x nabíjet. K výsledku se došlo realizací jednotlivých scénářů, kde každý scénář byl realizován 1000x pro danou hodnotu paramtru ε. Pro každou hodnotu pak byly všechny ceny cest z jednotlivých realizací sečteny a způměrovány. Hodnoty parametru ε byly nastaveny na [0.8, 0.85, 0.9, 0.95, 0.97, 0.98, 0.99, 1].

Z grafu 7.5 je vidět, že nejmenší chybu mělo řešení při nastavení $\varepsilon = 1$. Avšak zbytek grafu je velice zvláštní. Předpoklad byl, že dojde k postupnému nárůstu ceny cesty se zmenšující se hodnotou parametru ε. To se však nestalo.

Na druhé straně graf 7.6 vlivu parametru na dobu výpočtu dopadnul přesně



Obrázek 7.5: Vliv hodnoty ε na cenu cesty



Obrázek 7.6: Vliv hodnoty ε na výpočetní čas

podle očekávání. Se snižující se hodnotou parametru ε došlo i ke snížení výpočetního času.

Kapitola 8

Závěr

Zadefinovali jsme problémy pro nalezení optimální cesty a optimální začátku cesty při daném koncovém času a pro nalezení optimální cesty při daném začátku a konci cesty. Oba problémy uvažovaly omezení plynoucí z baterií v elektromobilu, nabíjení automobilu a uvažovali neurčitost v podobě fronty na nabíjecích stanicích. Vytvořili jsme dva algoritmy, každý pro řešení jednoho ze těchto dvou problémů. Oba algoritmy minimalizovali kvadratickou cenu naplánované cesty. S takto definovanou cenou cesty bylo možné nalézt řešení s minimální variancí. Cena cesty po realizaci tak je velice blízká zadanému koncovému času. Pro oba algoritmy bylo potřeba z důvodů výpočetních a paměťových nároků vytvořit zrychlující techniky. Pomocí evaluace bylo dokázáno, že výsledky našich algoritmů jsou lepší než výsledky základního algoritmu, který neurčitost během plánování nepoužívá a vyhledával pouze nejkratší cestu mezi dvěma uzly.

Použitá literatura

[1] *V Česku je 5,8 milionu aut, která stárnou. V průměru mají téměř patnáct let*. B.m.: `https://zpravy.aktualne.cz/ekonomika/auto/v-cesku-je-5-8-milionu-aut-ktera-starnou~r~36d08318208911e9b869ac1f6b220ee8/`. Accessed: 2020-05-22

[2] DIJKSTRA, Edsger W. A note on two problems in connexion with graphs. *Numerische mathematik*. 1959, **1**(1), 269–271.

[3] JOHNSON, Donald B. Efficient Algorithms for Shortest Paths in Sparse Networks. *J. ACM* [online]. 1977, **24**(1), 1–13. ISSN 0004-5411. Dostupné z: doi:10.1145/321992.321993

[4] HART, Peter E, Nils J NILSSON a Bertram RAPHAEL. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*. 1968, **4**(2), 100–107.

[5] RUSSELL, Stuart a Peter NORVIG. *Artificial Intelligence: A Modern Approach*. 3rd vyd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN 0136042597, 9780136042594.

[6] GEISBERGER, Robert, Peter SANDERS, Dominik SCHULTES a Daniel DELLING. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In: *International Workshop on Experimental and Efficient Algorithms*. B.m.: Springer, 2008, s. 319–333.

[7] WAGNER, Dorothea a Thomas WILLHALM. Speed-Up Techniques for Shortest-Path Computations. In: [online]. 2007, s. 23–36. Dostupné z: doi:10.1007/978-3-540-70918-3_3

[8] DELLING, Daniel, Peter SANDERS, Dominik SCHULTES a Dorothea WAGNER. Engineering Route Planning Algorithms. In: Jürgen LERNER, Dorothea WAGNER a Katharina A. ZWEIG, ed. *Algorithmics of Large and Complex Networks: Design, Analysis, and Simulation* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, s. 117–139. ISBN 978-3-642-02094-0. Dostupné z: doi:10.1007/978-3-642-02094-0_7

[9] SANDERS, Peter a Dominik SCHULTES. Highway Hierarchies Hasten Exact Shortest Path Queries. In: Gerth Stølting BRODAL a Stefano LEONARDI, ed. *Algorithms – ESA 2005*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, s. 568–579. ISBN 978-3-540-31951-1.

[10] BAST, Holger, Stefan FUNKE, Peter SANDERS a Dominik SCHULTES. Fast Routing in Road Networks with Transit Nodes. *Science* [online]. 2007, **316**(5824), 566–566. ISSN 0036-8075. Dostupné z: doi:10.1126/science.1137521

[11] ANDREAS ARTMEIER, Martin Leucker, Julian Haselmayr a Martin SACHENBACHER. The optimal routing problem in the context of battery-powered electric vehicles. In: *CPAIOR Workshop on Constraint Reasoning and Optimization for Computational Sustainability (CROCS)*. 2010.

[12] EISNER, Jochen, Stefan FUNKE a Sabine STORANDT. Optimal route planning for electric vehicles in large networks. In: *Twenty-Fifth AAAI Conference on Artificial Intelligence*. 2011.

[13] CLIMACO, João Carlos Namorado a Ernesto Queirós Vieira MARTINS. A bicriterion shortest path algorithm. *European Journal of Operational Research* [online]. 1982, **11**(4), 399–404. ISSN 0377-2217. Dostupné z: doi:https://doi.org/10.1016/0377-2217(82)90205-3

[14] MANDOW, Lawrence a José Luis Pérez DE LA CRUZ. Multiobjective A* search with consistent heuristics. *Journal of the ACM (JACM)*. 2010, **57**(5), 27.

[15] STEWART, Bradley S a Chelsea C WHITE III. Multiobjective A*. *Journal of the ACM (JACM)*. 1991, **38**(4), 775–814.

[16] DREYFUS, Stuart E. An appraisal of some shortest-path algorithms. *Operations research*. 1969, **17**(3), 395–412.

[17] COOKE, Kenneth L a Eric HALSEY. The shortest route through a network with time-dependent internodal transit times. *Journal of mathematical analysis and applications*. 1966, **14**(3), 493–498.

[18] CHABINI, Ismail. Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. *Transportation research record*. 1998, **1645**(1), 170–175.

[19] HALL, ECJ. Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications. 1993.

[20] HAMACHER, Horst W, Stefan RUZIKA a Stevanus A TJANDRA. Algorithms for time-dependent bicriteria shortest path problems. *Discrete optimization*. 2006, **3**(3), 238–254.

[21] EVDOKIA NIKOLOVA, David R. Karger. Route Planning under Uncertainty: The Canadian Traveller Problem. In: *Proceedings of the Twenty-Third AAAI Conference on Artifical Intelligence (2008)*. 2008, s. 969–974.

[22] PAPADIMITRIOU, Christos H a Mihalis YANNAKAKIS. Shortest paths without a map. *Theoretical Computer Science*. 1991, **84**(1), 127–150.

[23] NIKOLOVA, Evdokia, Matthew BRAND a David R KARGER. Optimal Route Planning under Uncertainty. In: *icaps*. 2006, s. 131–141.

[24] CHEN, Bi Yu, William LAM, Agachai SUMALEE, Qingquan LI, Hu SHAO a Zhixiang FANG. Finding Reliable Shortest Paths in Road Networks Under Uncertainty. *Networks and Spatial Economics* [online]. 2013, **13**, 123–148. Dostupné z: doi:10.1007/s11067-012-9175-1

[25] CHEN, Anthony a Zhaowang JI. Path finding under uncertainty. *Journal of Advanced Transportation* [online]. 2005, **39**, 19–37. Dostupné z: doi:10.1002/atr.5670390104

[26] BATISTA, L. S., F. CAMPELO, F. G. GUIMARÃES a J. A. RAMÍREZ. A comparison of dominance criteria in many-objective optimization problems. In: *2011 IEEE Congress of Evolutionary Computation (CEC)*. 2011, s. 2359–2366.

[27] FIŠER, Bc. Tomáš. *Integrated Route and Charging Planning for Electric Vehicles*. B.m., 2017. Master's thesis. Czech Technical University in Prague.

Příloha A

Obsah CD

CD v příloze obsahuje:

- **App** – zdrojové soubory k implementaci
- **Preprocessing** – zdrojové soubory k převodu OSM map do interní reprezentace k dalšímu zpracování
- **Text** – zdrojové soubory k textu práce
- **Diplomová práce.pdf** – text práce ve formátu PDF

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hromadník** Jméno: **Lukáš** Osobní číslo: **420671**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Plánování cest pro elektrická auta s neúplnou informací

Název diplomové práce anglicky:

Route Planning for Electric Vehicles with Incomplete Information

Pokyny pro vypracování:

Plánování cest pro elektrická auta rozšiřuje klasické plánování cest o restrikce vyplívajících z omezené kapacity baterie a o plánovaní kdy, kde a jak dlouho dobíjet na dobíjecích stanicích. Avšak veškeré informace potřebné k plánování nebývají obvykle k dispozici (např. obsazenost dobíjecích stanic) a je tedy nutné pracovat s neúplnou informací. Cílem práce je vytvořit algoritmus, který bude plánovat cesty pro elektrická auta a bude pracovat s neúplnou informací, tak aby se zvýšila pravděpodobnost úspěšného provedení plánu.

1. Vyhledejte a analyzujte existující algoritmy pro plánování cest a algoritmy pracující s neúplnou informací.
2. Definujte problém plánování cest pro elektrická auta s neúplnou informací.
3. Navrhněte algoritmus, který tento problém řeší.
4. Navržený algoritmus implementujte.

Seznam doporučené literatury:

[1] Baum, M., Dibbelt, J., Gemsa, A., Wagner, D., & Zündorf, T. (2015, November). Shortest feasible paths with charging stops for battery electric vehicles. In Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems (p. 44). ACM.

[2] Eisner, J., Funke, S., & Storandt, S. (2011, August). Optimal Route Planning for Electric Vehicles in Large Networks. In AAAI (pp. 1108-1113).

[3] Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., ... & Werneck, R. F. (2016). Route planning in transportation networks. In Algorithm engineering (pp. 19-80). Springer, Cham.

[4] Nikolova, E., & Karger, D. R. (2008, July). Route Planning under Uncertainty: The Canadian Traveller Problem. In AAAI (pp. 969-974).

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Marek Cuchý, katedra počítačů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **28.01.2019** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **20.09.2020**

Ing. Marek Cuchý podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání

Podpis studenta