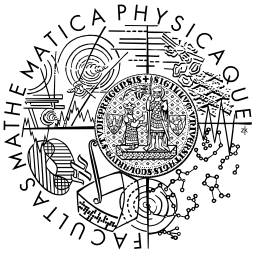


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Lada Kudláčková

Heuristiky pro cesty v mapách

Katedra aplikované matematiky

Vedoucí bakalářské práce: Mgr. Mareš Martin, Ph.D.

Studijní program: Informatika

Studijní obor: Obecná informatika

Praha 2019

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Chtěla bych poděkovat Mgr. Martinu Marešovi, Ph.D., za odborné vedení bakalářské práce, za jeho čas, cenné připomínky a trpělivý přístup.

Název práce: Heuristiky pro cesty v mapách

Autor: Lada Kudláčková

Katedra: Katedra aplikované matematiky

Vedoucí bakalářské práce: Mgr. Mareš Martin, Ph.D., Katedra aplikované matematiky

Abstrakt: Obsahem práce je popis heuristických postupů, které slouží pro hledání nejkratších cest v grafech, a ověření jejich účinnosti na skutečných datech. Věnuje se heuristikám pro Dijkstrův algoritmus, a to především algoritmu A*, který využívá dolní odhad na vzdálenost do cíle. Heuristiky jsou implementovány a testovány na silniční síti České republiky.

Klíčová slova: hledání nejkratších cest, Dijkstrův algoritmus, heuristiky

Title: Heuristics for paths in maps

Author: Lada Kudláčková

Department: Department of Applied Mathematics

Supervisor: Mgr. Mareš Martin, Ph.D., Department of Applied Mathematics

Abstract: The content of the thesis is a description of heuristic procedures, which are used to find the shortest paths in the graphs and verify their effectiveness on the actual data. It deals with heuristics for Dijkstra's algorithm, especially the A* algorithm, which uses a lower distance-to-target estimate. Heuristics are implemented and tested on the road network of the Czech Republic.

Keywords: shortest path searching dijkstra's algorithm heuristics

Obsah

1. Úvod

1.1 Základní pojmy

1.1.1 Graf

Grafem $G = (V,E)$ rozumíme množinu vrcholů V a množinu hran E, kde hrany jsou dvojice vrcholů grafu.

Graf je neorientovaný, pokud je každá jeho hrana neuspořádaná dvojice. V takovém grafu můžeme hranu e mezi vrcholy u a v označit výrazem $e = \{u,v\}$.

V orientovaném grafu jsou hrany uspořádané dvojice a hranu e vedoucí z u do v zapisujeme jako $u = (u,v)$. Skutečnost, že V je množina vrcholů grafu G, vyjadřujeme výrazem $V(G)$, výrazem $E(G)$ značíme hrany G.

Podgraf grafu $G = (V,E)$ je graf, jehož vrcholy jsou podmnožinou $V(G)$ a jehož hrany jsou podmnožinou $E(G)$. Podgraf indukovaný množinou vrcholů W obsahuje právě ty hrany z $E(G)$, které vedou mezi vrcholy z W.

Hrana opačná k hraně (u,v) je hrana (v,u). Symetrizace orientovaného grafu G je neorientovaný graf H, který vznikne z G rozšířením množiny hran tak, že pro každou hranu e přidáme k hranám H i hranu k ní opačnou, pokud ji graf H už neobsahuje.

Konečný graf je graf s konečnou množinou vrcholů a hran.

Budeme se zabývat pouze konečnými obyčejnými grafy, tedy konečnými grafy bez smyček a násobných hran. Smyčka je hrana, která je tvořena jediným vrcholem. Pro obyčejný neorientovaný graf platí, že pro každou dvojici vrcholů existuje nejvýše jedna hrana e vedoucí mezi nimi. V obyčejném neorientovaném grafu máme pro vrcholy u a v nejvýše jednu hranu vedoucí z u do v.

1.1.2 Cesta v grafu

Cesta mezi vrcholy u a v v neorientovaném grafu $G = (V,E)$ je posloupnost jeho hran $(\{u,w_0\},\{w_0,w_1\},\ldots,\{w_n,v\})$, ve které se každý vrchol vyskytuje nejvýše jednou. Na množině $V(G)$ můžeme zavést relaci dosažitelnosti: vrchol v je dosažitelný z vrcholu u, pokud v G existuje cesta vedoucí mezi u a v. Relace dosažitelnosti v neorientovaném grafu je symetrická, tzn. vrchol v je dosažitelný z u právě tehdy, když je vrchol u dosažitelný z v.

Podobně lze definovat cestu a dosažitelnost v orientovaném grafu G: orientovaná cesta z u do v je složená z orientovaných hran G, relace dosažitelnosti zde ale není symetrická.

1.1.3 Souvislost grafu

Neorientovaný graf je souvislý, pokud v něm pro každý vrchol u a v existuje cesta mezi u a v. Komponentou grafu G nazýváme množinu vrcholů $W \subseteq V(G)$, pokud je podgraf indukovaný množinou W souvislý.

Na orientovaném grafu zavedeme pojem slabé a silné souvislosti: Slabě souvislý je takový graf, jehož symetrizace je souvislá. Silně souvislý je graf, ve kterém existuje pro každý vrchol u a v cesta z u do v i cesta z v do u.

1.1.4 Vzdálenost a nejkratší cesta

Ohodnocení hran E je funkce ℓ, která každé hraně v E přiřadí její délku $\ell(e)$. Délkou cesty P rozumíme součet délek hran tvořících cestu.

Obecně může být $\ell(e)$ libovolné reálné číslo, budeme se ale věnovat hledaní cest v grafech s nezápornými délkami hran. V následujícím textu uvažujeme jen konečné orientované grafy.

V grafu s ohodnocením hran můžeme definovat pojem vzdálenosti d_G mezi vrcholy v grafu G: $d_G(u,v)$ je nejmenší možná délka cesty z u do v v G. Pokud vrchol v není z u dosažitelný, bude $d_G(u,v)$ rovno ∞.

Nejkratší cesta mezi u a v v G je taková cesta P, jejíž délka se rovná $d_G(u,v)$.

2. Prohledávání grafu

2.1 Prohledávání neohodnoceného grafu

V grafu G, který nemá ohodnocené hrany, lze délku cesty vyjádřit počtem jejich hran.

Pokud chceme pro vrchol $v \in G$ najít nejkratší cesty do všech ostatních vrcholů v G, můžeme použít algoritmus prohledávání grafu do šířky (Breadth-first search, BFS).

Od algoritmu prohledávání do hloubky (Depth-first search, DFS) se BFS liší datovou strukturou, do které se ukládají otevřené vrcholy. K ukládání nám slouží fronta, tj. struktura, ze které objekty vystupují ve stejném pořadí, v jakém byly vloženy. DFS ukládá vrcholy do zásobníku, ze kterého jsou vyjímány v opačném pořadí.

Vrchol v, ve kterém hledání začíná, budeme obvykle nazývat zdrojem a označovat ho s (source). Pro vrcholy zavedeme stavy: *nenavštíven*, *otevřen* a *uzavřen*. Všem vrcholům přiřadíme stav *nenavštíven*, vrchol s vložíme do fronty a jeho stav změníme na *otevřen*.

Dokud fronta není prázdná, opakujeme: vyjmeme z fronty první vrchol u, nastavíme ho na *uzavřen* a otevřeme všechny jeho nenavštívené sousedy, tzn. každý vrchol v, do kterého vede hrana z u, vložíme do fronty a jeho stav změníme z *nenavštíven* na *otevřen*.

V případě, že v byl při prohledávání grafu otevřen z vrcholu u, říkáme, že u je předchůdce v a v je následník u.

Je-li v následníkem u, platí v neohodnoceném grafu G rovnost $d_G(s,v) = d_G(s,u) + 1$.

Algoritmus vytvoří strom nejkratších cest.

2.2 Prohledávání ohodnoceného grafu

Pro hledání nejkratších cest na ohodnoceném grafu postupujeme podobně jako při prohledávání do hloubky, musíme si ale pro každý vrchol u udržovat jeho aktuální vzdálenost od zdroje (tj. délka zatím nejkratší objevené cesty z s do u).

Aktuální vzdálenost vrcholu u budeme značit $h(u)$. Na začátku nastavíme všechny vrcholy u na *nenavštíven* a $h(u)$ na ∞, kromě zdroje, pro který bude $h(u)$ rovno 0.

V průběhu algoritmu zlepšujeme $h(u)$ pomocí operace relaxace.

V každém kroku zvolíme otevřený vrchol v, uzavřeme ho a provedeme relaxaci v: Pro každého souseda w vrcholu v zjistíme, zda platí, že $h(w) \geq h(v) + \ell((v,w))$. Pokud ano, otevřeme vrchol w a položíme $h(w) = h(v) + \ell((v,w))$.

Postup opakujeme, dokud existují nějaké otevřené vrcholy. Po skončení algoritmu je pro každý uzavřený vrchol v splněna rovnost $h(v) = d_G(s,v)$. Vrcholy nedosažitelné ze zdroje jsou ve stavu *nenavštíven*.

2.3 Asymptotická časová složitost

Asymptotická časová složitost algoritmu vyjadřuje, jak se algoritmus chová na velkých datech. Umožňuje porovnání algoritmů a jejich zařazení do kategorií. Časovou složitost chápeme jako maximální počet kroků, který algoritmus vykoná na datech velikosti n.

2.3.1 Složitost prohledávání grafu

Pokud mluvíme o složitosti grafových algoritmů, budeme používat obvyklé značení pro velikosti množin vrcholů a hran.

Pro $G = (V,E)$ je $n = |V|$ a $m = |E|$. Velikost vstupu algoritmu vyjadřujeme pomocí m a n.

Složitost algoritmu závisí mimo jiné na způsobu, jak reprezentujeme hrany. Informace o hranách lze udržovat v matici řádu n, ke zjištění všech sousedů vrcholu u je potom potřeba n kroků. Hrany můžeme reprezentovat i pomocí seznamu sousedů ve vrcholech.

Při reprezentaci pomocí seznamu sousedů je asymptotická složitost BFS $O(n+m)$).

Asymptotická složitost prohledávání neohodnoceného grafu závisí na tom, v jakém pořadí vybíráme vrcholy k relaxaci.

3. Dijkstrův algoritmus

Dijkstrův algoritmus používáme k efektivnímu vyhledávání nejkratších cest v grafech s nezáporným ohodnocením hran. Tento postup popsal nizozemský informatik Edsger Wybe Dijkstra v roce 1959.

Jedná se o relaxační algoritmus a jeho efektivita je dána pravidlem pro výběr vrcholu, který chceme relaxovat.

V každém kroku zvolíme k relaxaci takový vrchol u, jehož aktuální vzdálenost $h(u)$ je nejmenší.

Věta 1. *Dijkstrův algoritmus uzavírá vrcholy v pořadí podle neklesající vzdálenosti od u. Každý dosažitelný vrchol je uzavřen právě jednou.*

Důkaz. V každém kroku je $h(u)$ uzavřeného vrcholu u nejvýše rovno $h(v)$ pro libovolný otevřený vrchol $h(v)$. Na začátku prohledávání je aktuální vzdálenost všech vrcholů kromě zdroje nekonečná. Zdroj je uzavřen jako první, protože $h(s)$ je rovno nule.

V každém dalším kroku relaxujeme vrchol u s minimálním $h(u)$ z otevřených vrcholů. Při relaxaci aktuální vzdálenost jeho sousedů neklesne pod $h(u)$, protože zlepšená aktuální vzdálenost je rovna $h(u) + \ell(e)$ a $\ell(e)$ je nezáporná pro každou hranu e .

□

3.1 Asymptotická složitost Dijkstrova algoritmu

Každý z n vrcholů je uzavřen nejvýše jednou a jeho relaxace proběhne v čase $O(n)$. Pokud budeme při vybírání vrcholu s nejnižší aktuální vzdáleností procházet všech n vrcholů, bude celý průběh algoritmu trvat $O(n^2)$.

Všechny relaxace dohromady vyžadují $O(m)$ kroků, proto můžeme celkovou složitost snížit výběrem vhodné datové struktury pro ukládání otevřených vrcholů. Při použití haldy máme pro operaci vkládání vrcholu do haldy, operaci změny aktuální vzdálenosti vrcholu a operaci vyjmutí minima zaručenou složitost $O(\log n)$. Algoritmus proto proběhne v čase $O(m \log n)$.

3.2 Heuristické algoritmy

Heuristický algoritmus je algoritmus, který využívá nějakou strategii umožňující rychlejší vyřešení problému. Nezaručuje nalezení nejrychlejšího řešení, ale nalezené řešení je vždy korektní.

Pro hledání nejkratší cesty mezi dvěma vrcholy můžeme vycházet ze znalosti struktury grafu a pokusit se odhadnout, které vrcholy máme prohledávat přednostně.

Heuristika poskytuje možnost snížit počet otevřených i uzavřených vrcholů. Při použití pro hledání nejkratších cest na mapě si takové zlepšení můžeme představit jako velikost prohledané plochy.

4. Použití heuristik pro hledání cest na silniční síti

4.1 Vstupní data

4.1.1 Získání vstupních dat

Pro zkoumání vhodnosti heuristik pro hledání v mapách byla použita data z projektu OpenStreetMap [1].

Shell skript umožňující stažení dat ve formátu JSON pomocí nástroje Overpass Turbo je přiložen v adresáři se zdrojovými kódy.

Data reprezentují silniční síť v ČR, která je složena z cest obsahujících přibližně 3500000 bodů. Jsou stažena ve formátu JSON.

4.1.2 Zpracování dat

Mapa je převedena do orientovaného grafu. Vrcholy odpovídají bodům a hrany jsou odvozeny z cest.

Poloha bodů na mapě je určena zeměpisnou délkou a šířkou, pro zjednodušení výpočtu jsou ale konvertována do Univerzálního transverzálního Mercatorova systému souřadnic (UTM). Délku hrany pak můžeme spočítat pomocí Pythagorovy věty.

V další fázi jsou z grafu odstraněny vrcholy, které leží uvnitř nějaké cesty a nemají žádné sousedy mimo tuto cestu. Cestu složenou z několika hran tak nahradíme jedinou hranou mezi jejími koncovými vrcholy. Její délka bude součtem délek původních hran.

4.2 Přehled heuristik

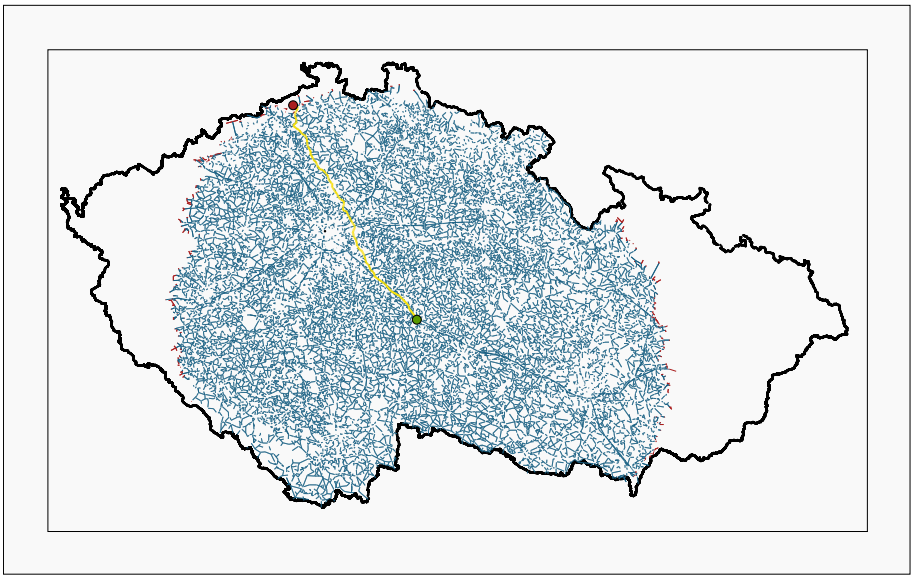
4.2.1 Dijkstrův algoritmus bez heuristiky

Na grafu s pravidelným rozmístěním vrcholů v rovině, např. na mřížkovém grafu, postupuje Dijkstrův algoritmus ze zdroje rovnoměrně všemi směry do ostatních vrcholů. Pokud nalezne nejkratší cestu do cíle, tvoří prohledaná plocha kruh se středem ve zdroji o poloměru rovnajícím se přibližně vzdálenosti nejkratší cesty.

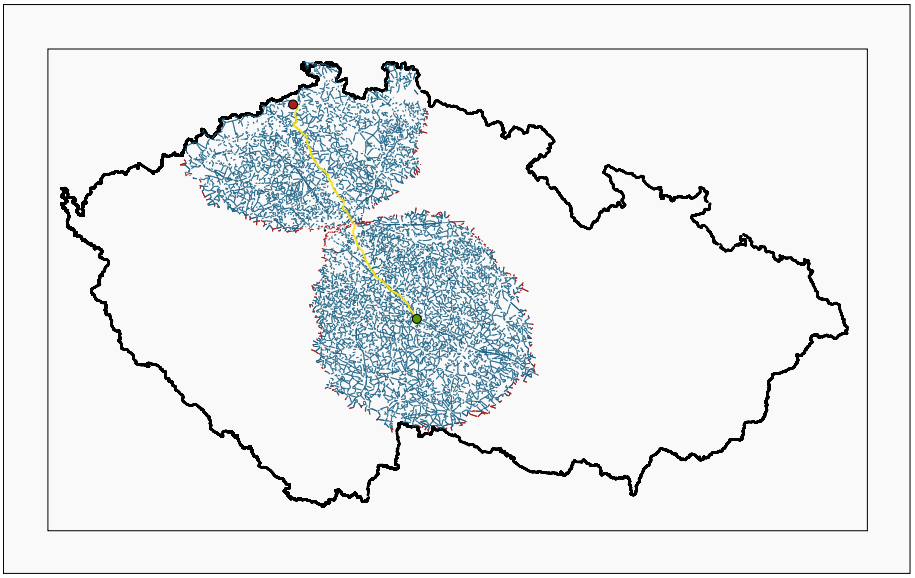
Ukázka průběhu Dijkstrova algoritmu je na obrázku 4.1. Zdroj je znázorněn zeleně, cíl červeně. Hrany mezi uzavřenými vrcholy jsou modré, hrany mezi otevřenými vrcholy jsou červené a ostatní se nezobrazují.

4.2.2 Obousměrný Dijkstrův algoritmus

Velikost prohledané plochy můžeme zmenšit, pokud procházíme graf střídavě nejen ve směru ze zdroje, ale i ve směru opačném. V opačném směru postupujeme z cílového vrcholu do ostatních vrcholů po zpětných hranách a hledáme zdroj. Hledání skončí, když je některý vrchol uzavřený v obou směrech.



Obrázek 4.1: Plocha prohledaná Dijkstrovým algoritmem bez heuristiky



Obrázek 4.2: Plocha prohledaná Obousměrným Dijkstrovým algoritmem

4.2.3 Algoritmus A*

Algoritmus A* používá heuristickou funkci pro dolní odhad vzdálenosti do cíle. Prohledává přednostně vrcholy, o kterých se předpokládá, že jsou blízko cíle.

Potenciálová funkce π je funkce přiřazující vrcholům reálná čísla. Pro danou potenciálovou funkci π můžeme definovat *redukovanou cenu hrany* předpisem: $\ell_\pi((u,v)) = \ell((u,v)) - \pi(u) + \pi(v)$.

Potenciálová funkce je přípustná, pokud je $\ell_\pi(e) \geq 0$ pro každou hranu e.

Délka každé cesty ze zdroje s do cíle t je v grafu s *redukovanou cenou hran* změněna právě o $\pi(t) - pi(s)$ vzhledem k původnímu ohodnocení cesty, proto je nalezená nejkratší cesta nejkratší cestou i v původním grafu.

V souvislosti s algoritmem A* budeme $\pi(u)$ považovat za dolní odhad vzdálenosti z u do cíle. Při vybírání vrcholu k relaxaci volíme vrchol s nejnižší hodnotou $h(u) + \pi(u)$.

Euklidovská vzdálenost jako dolní odhad

Pro vyhledávání v mapách můžeme jako potenciálovou funkci použít euklidovskou vzdálenost. Potenciálová funkce je přípustná, což lze snadno dokázat z trojúhelníkové nerovnosti.

Přednostně jsou prohledávány vrcholy ležící mezi zdrojem a cílem. Pro hledání nejkratších cest na silniční síti ČR tato heuristika není vhodná v případě, že úsečka mezi zdrojem a cílem prochází mimo území ČR, nebo pokud se jedná o vrcholy v menší vzdálenosti, mezi nimiž se nachází přirozená překážka (např. vrcholy na protějších březích řeky).

Vzdálenost k landmarkům jako dolní odhad

Jiný přístup zvolili Goldberg a spol. [2]. Zvolíme pevně množinu vrcholů, které budeme nazývat landmarky. Pro každý vrchol u si předpočítáme vzdálenosti mezi s a všemi landmarky.

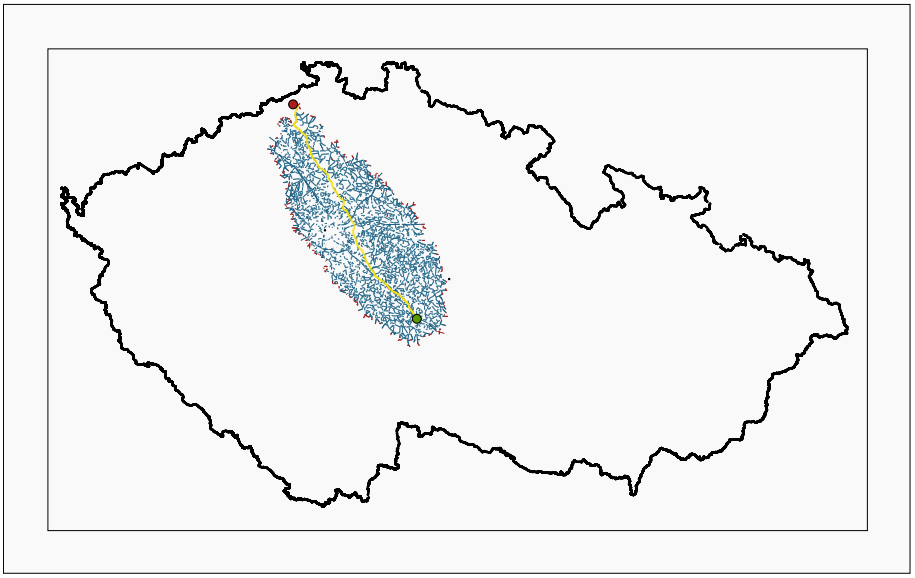
Při hledání cesty ze zdroje do cíle vybereme jeden z landmarků L (může se měnit v průběhu algoritmu) a vzdálenost k němu společně s trojúhelníkovou nerovností nám poskytne vhodnou potenciálovou funkci. Landmark L chceme zvolit takový, aby bylo možné předpokládat, že nejkratší cesta z s do L a nejkratší cesta z s do t mají společný počáteční úsek. Pokud hrana (u,v) leží na nejkratší cestě z u do L, je její *redukovaná cena* nulová, protože nejkratší cesta z u do L je složena právě z hrany (u,v) a nejkratší cesty z v do L. Proto platí:

$\ell_\pi((u,v)) = \ell((u,v)) - \pi(u) + \pi(v) = 0$.

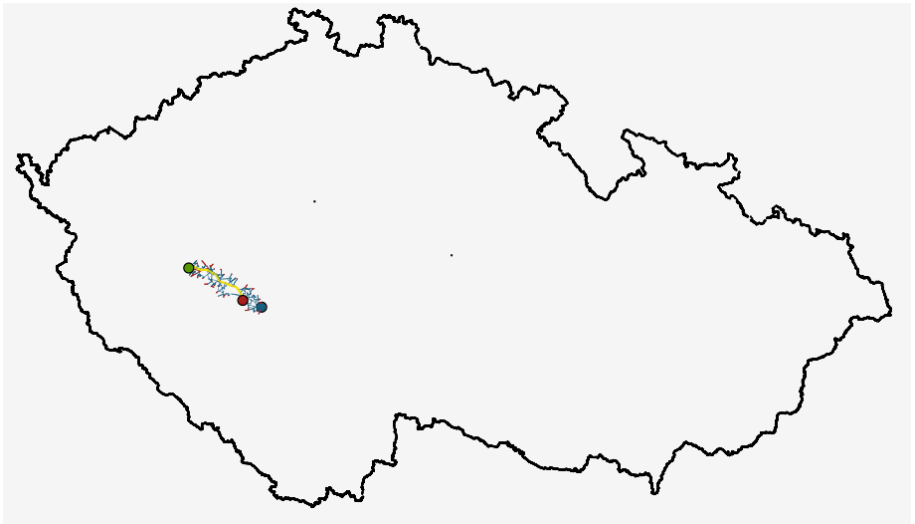
Úspěšnost heuristiky závisí nejen na počtu landmarků. Výrazně ji ovlivňuje i vhodný výběr množiny landmarků. Pro silniční síť byly landmarky nalezeny tak, aby rovnoměrně pokryly území ČR. Území bylo rozděleno čtvercovou mřížkou a z každé oblasti byl landmark vybrán náhodně.

Takový způsob volby landmarků nezohledňuje hustotu grafu v různých částech mapy. Zkusila jsem se najít rozmístění, ve kterém by počet landmarků byl úměrný počtu vrcholů v dané oblasti. Graf byl převeden do planárního grafu vnořením do roviny a odstraněním křížících se hran. Byl vytvořen strom nejkratších cest a graf byl rozložen na komponenty pomocí cest vedoucích z kořene stromu do vrcholů

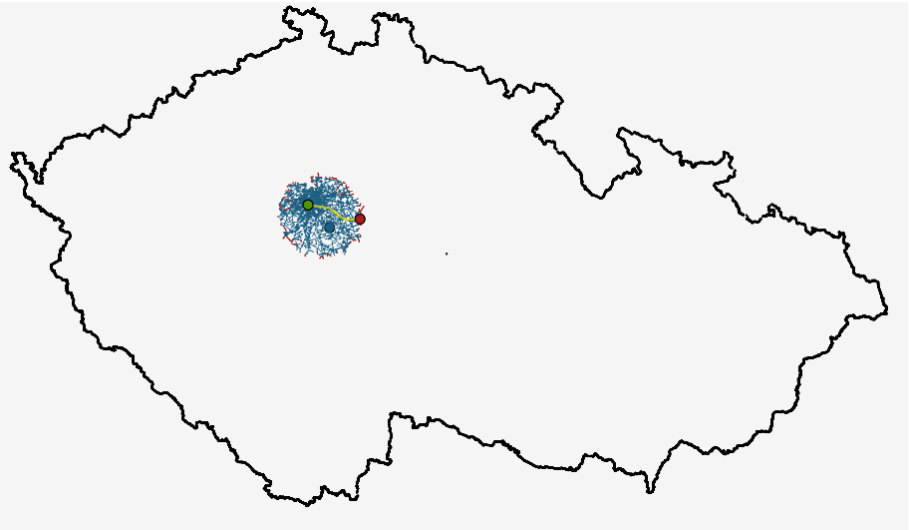
ležících na hranicích ČR. Nepodařilo se mi ale najít vhodný rozklad, který by poskytl lepší výsledky než rozmístění podle mřížky.



Obrázek 4.3: Plocha prohledaná euklidovským A*



Obrázek 4.4: Plocha prohledaná algoritmem A* s landmarkem vhodně zvoleným pro danou dvojici.



Obrázek 4.5: Plocha prohledaná algoritmem A* s nevhodným landmarkem.

5. Závěr

5.1 Porovnání algoritmů

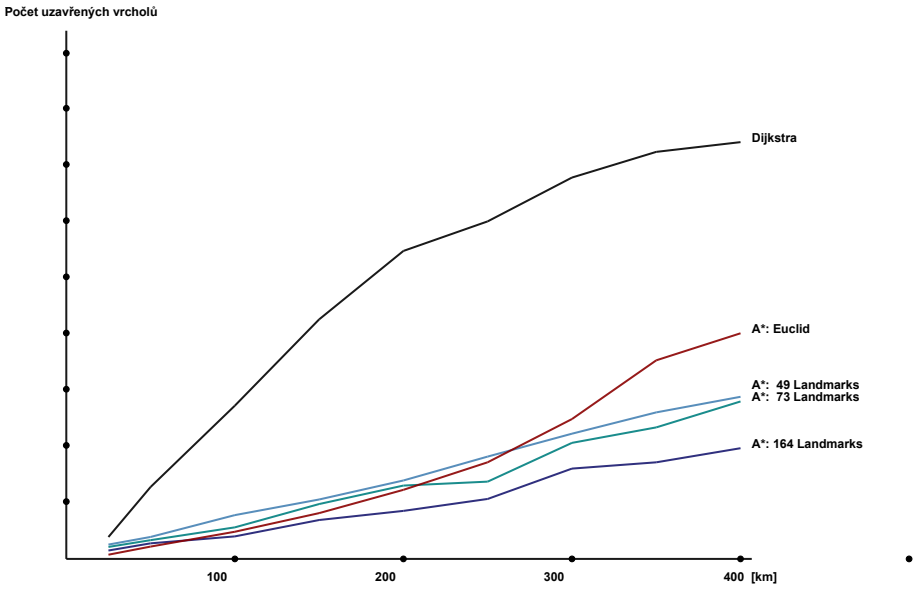
Náhodně bylo vybráno 5000 dvojic vrcholů. Porovnávali jsme velikost prohledané plochy, vyjádřenou jako počet uzavřených yrcholů, při použití heuristik s dolním odhadem. Heuristiky mají při hledání nejkratších cest na silniční síti zřejmě lepší výsledky než samotný Dijkstrův algoritmus.

Porovnávali jsme algoritmus A* s 49, 73 a 164 landmarky umístěnými rovnoměrně na území ČR. Počet uzavřených vrcholů je přímo úměrný počtu landmarků a algoritmus má dobré výsledky bez ohledu na to, jaká je vzdálenost mezi zdrojem a cílem.

Euklidovský A* se nejlépe uplatní při menších vzdálenostech. Pro vrcholy ve vzdálenosti menší než 50 km je prohledaná plocha menší než u zmíněných tří algoritmů s landmarky, naopak pro vzdálenosti přes 300 km je hledání méně efektivní.

5.2 Shrnutí

Byla ověřena vhodnost heuristických rozšíření Dijkstrova algoritmu pro hledání nejkratších cest v silniční síti, zejména účinnost algoritmu A*, pokud za heuristickou funkci zvolíme dolní odhad vzdálenosti k landmarkům.



Obrázek 5.1: Porovnání algoritmů

Literatura

[1] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org , 2017.

[2] Andrew V. Goldberg, Haim Kaplan, and Renato F. Werneck. Better landmarks within reach. pages 38–51, 2007.

Seznam obrázků