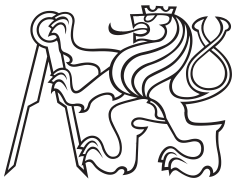
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Plánování cest s nejistou průchodností hran

Jan Macháček

Vedoucí práce: Ing. Marek Cuchý
Studijní program: Otevřená informatika
Specializace: Základy umělé inteligence a počítačových věd
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Macháček** Jméno: **Jan** Osobní číslo: **483753**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Otevřená informatika**

Specializace: **Základy umělé inteligence a počítačových věd**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Plánování cest s nejistou průchodností hran

Název bakalářské práce anglicky:

Route Planning with Uncertain Edge Traversability

Pokyny pro vypracování:

Mapové podklady pro plánování cest jsou často neúplné nebo obsahují chyby. Tyto nedostatky se nevyskytují příliš často u informací podstatných pro cesty automobilem, ale jsou důležité pro jiné způsoby dopravy, u kterých překážky na cestě můžou způsobit velké komplikace (např. pro lidi na vozíku nebo cyklisty). Běžné plánovače tuto nejistou průchodnost hran ignorují – buď hranu označí jako plně průjezdnou nebo ji vůbec neuvažují. Cílem práce je navrhnout a implementovat algoritmus, který pracuje s touto nejistotou a navrhuje cesty v závislosti na pravděpodobnosti jejich průchodnosti.

1. Vyhledejte a analyzujte existující algoritmy řešící podobné problémy.
2. Formalizujte problém plánování cest s nejistou průchodností hran.
3. Navrhněte algoritmus řešící zadaný problém.
4. Implementujte tento algoritmus.
5. Implementovaný algoritmus vyhodnoťte na testovacích instancích problému založených na reálných datech.

Seznam doporučené literatury:

[1] Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wagner, D. and Werneck, R.F., 2016. Route planning in transportation networks. In Algorithm engineering (pp. 19-80). Springer, Cham.
[2] Papadimitriou, C.H. and Yannakakis, M., 1991. Shortest paths without a map. Theoretical Computer Science, 84(1), pp.127-150.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Marek Cuchý, katedra počítačů FEL

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

Datum zadání bakalářské práce: **08.01.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

Ing. Marek Cuchý	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. Datum převzetí zadání

Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Poděkování

V prvé řadě bych rád poděkoval vedoucímu mé práce Ing. Marku Cuchému za jeho ochotu a přínosné rady při vedení této práce.

Dále bych rád poděkoval Výzkumnému centru informatiky RCI ČVUT za možnost využití výpočetní kapacity pro běh experimentů. Tato výpočetní infrastruktura je financována OP VVV projektem CZ.02.1.01/0.0/0.0/16_019/0000765 „Research Center for Informatics“.

Děkuji také mé rodině za podporu během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 21. května 2021

.....

podpis autora práce

Abstrakt

Ačkoliv mapové podklady nejsou dokonalé, většina algoritmů plánování cest nebere v úvahu průchodnost jednotlivých úseků cesty. Pro některé uživatele (např. pro lidi na vozíku nebo lidi s kočárky) přitom možné překážky na cestě mohou způsobit velké komplikace. Cílem této práce je navrhnout a implementovat algoritmus plánování cest, který bere v potaz pravděpodobnost průchodu jednotlivých hran.

Nejprve jsme zvolili vhodné kritérium průchodnosti a zformulovali dva podobné problémy. Prvním problémem je multikriteriální problém nalezení množiny všech Pareto-optimálních cest optimalizující kritéria vzdálenosti a průchodnosti. Cílem druhého problému je nalézt nejkratší cestu, která splňuje dané omezení kritéria průchodnosti.

Pro řešení obou těchto problémů jsme navrhli a implementovali podobné algoritmy založené na algoritmu NAMOA*.

K vyhodnocení obou algoritmů jsme použili grafy Prahy a Šumavy pro pěší. Dále jsme na základě informací z OpenStreetMap vytvořili 2 scénáře průchodnosti simulující rozdílné situace s ohledem na počasí, ze kterých byla určena pravděpodobnost průchodu jednotlivých hran.

V experimentech jsme porovnali oba navržené algoritmy na několika vytvořených sadách problémů. Kromě nich jsme také vyhodnotili vliv heuristik, různých implementací Pareto-množin a scénářů průchodnosti.

Výsledky experimentů ukázaly, že všechny zkoumané parametry, mohou výrazně ovlivnit dobu běhu algoritmu.

Klíčová slova: plánování cest, nejistota, nejistá průchodnost, multikriteriální problém nejkratší cesty

Vedoucí práce: Ing. Marek Cuchý

Abstract

Although the map data are not perfect, the majority of route planning algorithms do not consider the traversability of particular parts of path. At the same time, there are users (e.g. people in wheelchairs or people with prams) to whom possible obstacles on the road can cause big issues. The aim of this thesis is to propose and implement route planning algorithm which takes into account the probability of traversal of particular edges.

At first, we chose a suitable traversability criterion and formulated two similar problems. The first problem is a multicriteria problem of finding all Pareto-optimal paths optimizing distance and traversability criteria. The aim of the second one is to find the shortest path that satisfies given traversability criterion constraint.

To solve both of these problems, we proposed and implemented similar algorithms based on the NAMOA* algorithm.

To evaluate both algorithms, we used graphs of Prague and Sumava for pedestrians. Furthermore, based on OpenStreetMap information, we created 2 traversability scenarios simulating a different situations with respect to the weather. The probability of traversal of particular edges was determined from these scenarios.

In experiments, we compared both proposed algorithms on several created problem sets. Moreover, we also evaluated the impact of heuristics, different Pareto-sets implementations and traversability scenarios.

The results of experiments showed that all examined parameters can significantly affect the algorithm runtime.

Keywords: route planning, uncertainty, uncertain traversability, multiobjective shortest path problem

Title translation: Route Planning with Uncertain Edge Traversability

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Při plánování cest je obvyklé reálný svět modelovat grafem dopravní sítě, ve kterém je následně hledána optimální cesta. Graf dopravní sítě je vytvořen z mapových podkladů, které však nejsou dokonalé. Občas se v nich totiž mohou vyskytovat určité nedostatky (např. v nich některé podstatné informace chybí nebo jsou nesprávné). A tedy i graf vytvořený z těchto podkladů bude obsahovat tyto nedokonalé informace.

Pro cesty automobily se chyby či nedostatky v datech dopravní sítě, vzhledem k jejich častému používání, příliš nevyskytují. A proto se cesty naplánované s těmito nedokonalými informacemi od optimálních příliš neliší. Na druhou stranu u některých způsobů dopravy (např. u lidí na vozíku, lidí s kočárkem nebo u cyklistů), kde se nedostatků v mapových podkladech vyskytuje více, může neúplnost a nesprávnost těchto informací způsobit během cesty velké komplikace.

Jiná informace, jejíž neuvažování může vést k nedokonalosti grafu a tím ovlivnit optimalitu naplánované cesty, je způsobena vlivem počasí. Například budou-li na cestě popadané větve, stane-li se z pevné cesty nezpevněná nebo bude-li cesta pokryta sněhovou pokrývkou, průchodnost (resp. průjezdnost) cesty tím bude určitě výrazně ovlivněna.

Ačkoliv je z výše uvedených příkladů vidět, že graf dopravní sítě nemusí být vždy dokonalý a průchodnost některých jeho hran může být nejistá, většina algoritmů plánování cesty nejistotu průchodnosti hran kvůli jednoduchosti ignoruje. Hrana je pak buď považována za plně průjezdnou nebo není uvažována vůbec.

V této práci uvažujeme předpoklad, že o této nejistotě máme předem jistou znalost. Konkrétně předpokládáme, že u každé hrany grafu je předem známá určitá hodnota pravděpodobnosti určující, jestli daná hrana bude průchodná. Cílem práce je pak zformulovat problém plánování cest tak, aby u naplánovaných cest byla brána v úvahu tato pravděpodobnost. Pro porovnání jednotlivých cest bude nejprve potřeba vhodně zadefinovat samotné kritérium průchodnosti cesty. Dále je cílem navrhnout a implementovat algoritmus pro řešení tohoto problému. A na závěr tento implementovaný algoritmus vyhodnotit na testovacích instancích založených na reálných datech.

Kapitola 2
Přehled literatury

Tato kapitola obsahuje přehled několika problémů plánování optimální cesty a popis algoritmů, které tyto problémy řeší. Nejprve je v sekci 2.1 popsán problém hledání nejkratší cesty v grafu, jehož struktura i ceny hran jsou předem známy. V sekci 2.2 je popsáno rozšíření této úlohy na problém hledání nejkratší cesty optimalizující více kritérií. Na závěr jsou v sekci 2.3 popsány problémy uvažující určitou nejistotu při hledání cesty v grafu. V těchto problémech je graf sice předem známý, ale skutečné ceny jeho hran již známy nejsou.

2.1 Hledání nejkratší cesty

Hledání nejkratší cesty mezi dvěma body v grafu je dobře známým a často studovaným problémem. Cílem problému je nalézt nejkratší cestu mezi dvěma uzly, tedy cestu s minimálním součtem délek jejích hran.

Poznamenejme, že optimalizovaným kritériem nemusí být pouze délka cesty, ale jakékoli jiné kritérium, jehož hodnota lze na jednotlivých hranách grafu sčítat. Například chceme-li hledat nejrychlejší cestu, můžeme použít algoritmus hledání nejkratší cesty, pouze s tím rozdílem, že cena hrany nebude její délka, ale čas, za který se lze přes danou hranu dostat.

2.1.1 Dijkstrův algoritmus

Jedním z nejznámějších algoritmů hledání nejkratší cesty v grafu je *Dijkstrův algoritmus* [1]. Cílem tohoto algoritmu je nalézt nejkratší cestu z určitého počátečního uzlu s do všech ostatních (z s dostupných) uzlů grafu.

Během běhu algoritmu si každý uzel u udržuje délku dosud nejkratší cesty $d(u)$ vedoucí z s do něj. Dále je udržována množina tzv. *otevřených uzlů*. Otevřený uzel je takový uzel, do něhož již byla nalezena cesta, ale ještě nebyl expandován (viz dále).

Na počátku algoritmu je počáteční uzel s vložen do množiny otevřených uzlů a $d(s)$ je mu nastavena na 0. Všem ostatním uzlům je přiřazena délka ∞.

Po počáteční inicializaci jsou v cyklu postupně *expandovány* dosud nejlepší uzly. Konkrétně, v každém cyklu algoritmu je z množiny otevřených uzlů vybrán uzel u s nejkratší délkou $d(u)$. Tento uzel u je odebrán z množiny

otevřených uzlů a je expandován. Během expanze uzlu u jsou postupně navštíveny všechny uzly v_i sousedící s tímto uzlem. Pro každý uzel v_i je vypočtena $d'(v_i)$, délka cesty z s do v_i vedoucí přes uzel u, jako

$$d'(v_i) = d(u) + l(u, v_i),$$

kde $l(u, v_i)$ je délka hrany mezi uzly u a v_i. Pokud je délka $d'(v_i)$ kratší než délka dosud nejkratší cesty $d(v_i)$, je do $d(v_i)$ přiřazena délka $d'(v_i)$ a uzel v_i je přidán do množiny otevřených uzlů.

Pokud je množina otevřených uzlů prázdná, je pro každý uzel nalezena délka nejkratší cesty a algoritmus končí.

V některé literatuře (např. v [2]) je tento algoritmus označován jako tzv. *uniform-cost search*. Základní Dijkstrův algoritmus lze snadno vylepšit tím, že si budeme udržovat i množinu *uzavřených* uzlů (tj. uzlů, které již byly vybrány k expanzi), které už nebudeme dále navštěvovat. To je umožněno díky tomu, že pokud byl uzel vybrán k expanzi, byla již do něj nalezena nejkratší cesta [2]. Pokud tedy hledáme pouze cestu do konkrétního uzlu, algoritmus může skončit před jeho expanzí.

Dijkstrův algoritmus je optimální na grafu s nezápornými hranami.

2.1.2 Algoritmus A*

Zobecněním Dijkstrova algoritmu pro hledání nejkratší cesty mezi dvěma uzly grafu je algoritmus A* [3], který kromě ceny dosažení uzlu $g(u)$ využívá i odhad ceny cesty do cílového uzlu. Tento odhad je realizován heuristickou funkcí $h(u)$. Z množiny otevřených uzlů se pak nevybírá podle funkce $g(u)$, ale podle funkce

$$f(u) = g(u) + h(u).$$

Poznamenejme, že v případě, že $h(u) = 0$ pro každé u, je algoritmus A* redukován na Dijkstrův algoritmus.

Pro správné chování algoritmu je požadováno, aby heuristická funkce byla *přípustná* a ideálně i *konzistentní*.

Definice 2.1. Je-li $h^*(u)$ cena optimální cesty z bodu u do cílového uzlu. Heuristická funkce $h(u)$ je **přípustná**, pokud pro každý uzel u platí

$$0 \leq h(u) \leq h^*(u).$$

Přípustná heuristika tedy nikdy neodhaduje vyšší cenu cesty do cílového uzlu než je skutečná, což zaručuje optimalitu nalezeného řešení.

Definice 2.2. Heuristická funkce $h(u)$ je **konzistentní**, pokud pro každý uzel u a jeho následovníka v platí trojúhelníková nerovnost

$$h(u) \leq c(u, v) + h(v),$$

kde $c(u, v)$ je cena hrany mezi uzly u a v.

Konzistentní heuristika tedy zaručuje, že hodnoty $f(u)$ během cesty budou neklesající. A tudíž kdykoliv je uzel vybrán k expanzi, byla již do tohoto uzlu nalezena optimální cesta [2]. Poznamenejme, že pokud je heuristika konzistentní, tak je zároveň také přípustná.

2.2 Multikriteriální hledání nejkratší cesty

V některých typech úloh optimalizace podle jednoho kritéria nestačí a je vhodné hledat optimální řešení podle více kritérií. Například zajímá-li nás nejkratší a zároveň nejrychlejší cesta, budeme optimalizovat podle vzdálenosti a času.

Na úvod poznamenejme, že zatímco algoritmy zmíněné v předchozí sekci, které hledají nejkratší cestu podle jednoho kritéria, naleznou řešení v polynomiálním čase, hledání nejkratší cesty s více kritérii je již složitější problém. V [4] bylo s odkazem na článek [5] zmíněno, že i v případě problému se dvěma kritérii může výsledná Pareto-množina obsahovat exponenciálně mnoho řešení.

Při hledání optimální cesty podle více kritérií vzniká hned několik problémů, které se v algoritmech optimalizujících jediné kritérium nevyskytují. Největším rozdílem je, že optimalizované kritérium není pouze skalární hodnota, ale vektor hodnot jednotlivých kritérií. Tím vzniká problém s určením lepšího z vektorů cen. Proto je pro vektory cen definována relace částečně uspořádaných preferencí $\prec$ nazvaná *dominance*.

Definice 2.3. Mějme dva vektory cen $c_1, c_2 \in \mathbb{R}^k$. Řekneme, že c_1 **dominuje** c_2 (značeno $c_1 \prec c_2$) právě tehdy, když pro každé $i \in \{1, \ldots, k\}$ platí

$$c_1(i) \leq c_2(i) \wedge c_1 \neq c_2.$$

Poznamenejme, že ne pro všechny dvojice vektorů cen existuje relace $\prec$, a tedy ne vždy lze rozhodnout, který z vektorů cen je lepší. Například pro vektory $c_1 = (1, 2)$, $c_2 = (1, 3)$ a $c_3 = (2, 1)$ platí, že c_1 dominuje c_2, ale neplatí, že c_1 dominuje c_3 ani že c_3 dominuje c_1.

2.2.1 Multikriteriální A*

Algoritmus rozšiřující algoritmus A* na více kritérií je *multikriteriální A* algoritmus* (Multiobjective A*, MOA*) popsaný v článku [6]. Jako rozšíření algoritmu A* byl později v článku *A New Approach to Multiobjective A* Search* [7] také navržen algoritmus NAMOA*. Vlastnosti algoritmu NAMOA* byly dále podrobněji studovány v článku [8], kde byl navíc algoritmus NAMOA* porovnáván s algoritmem MOA*. Mimo jiné zde bylo ukázáno, že za rozumných předpokladů má algoritmus NAMOA* některé důležité vlastnosti algoritmu A*, které však algoritmus MOA* nemá. Jednou z těchto vlastností je například, že s konzistentními heuristikami lze u algoritmu NAMOA* garantovat, že k expanzi budou vybrány jen nezbytně nutné cesty.

Protože v multikriteriální optimalizaci nemusí být optimální jediné řešení. Cílem algoritmů MOA* a NAMOA* je proto nalézt množinu všech nedominovaných (Pareto-optimálních) řešení.

Vzhledem k tomu, že jsou hledány všechny nedominované cesty, je potřeba si během prohledávání v každém uzlu u udržovat množinu cen všech nedominovaných cest do u. Tato množina obsahující nedominované ceny bývá také nazývána jako tzv. *Pareto-množina*. Kdykoliv je pak nalezena nová

cesta do uzlu u, je potřeba zkontrolovat její dominanci s cenami všech dosud nedominovaných cest do u z této množiny.

Vzhledem k tomu, že značnou část běhu multikriteriálního algoritmu tvoří právě porovnávání dominance jednotlivých cest, v článku [9] byla představena technika tzv. *redukce dimenze*, která může běh algoritmu urychlit. Tuto techniku je možné použít u tzv. přesných multikriteriálních label-setting algoritmů, mezi něž patří i algoritmus NAMOA*. Redukce dimenze spočívá v tom, že v některých případech je za určitých podmínek možné při kontrole dominance vynechat první kritérium. Tyto podmínky konkrétněji popíšeme dále v podsekci 5.1.

2.2.2 Konkrétní úlohy multikriteriálního plánování cest

Konkrétním případem, kdy je při plánování cest vhodné využít multikriteriální úlohu může být plánování pro cyklisty, pro které délka nebo čas cesty nemusí být hlavním kritériem pro výběr cesty.

Plánování pro cyklisty bylo řešeno například v článku [10], ve kterém byl zformulován bikriteriální problém, kde prvním kritériem byl čas cesty a druhým kritériem byla vhodnost cesty. Jednotlivým hranám grafu byl určen čas a cena kritéria vhodnosti, což byla hodnota mezi 0 a 100 získaná jako kombinace mnoha faktorů např. bezpečnost cesty, úroveň dopravy nebo terén.

Multikriteriální plánování pro cyklisty bylo řešeno také v článku [11], kde byl zformulován problém se třemi kritérii (časem cesty, pohodlností cesty a převýšením na cestě). Protože se zvyšujícím počtem kritérií se také zvyšuje doba běhu algoritmu, v článku byly porovnávány kombinace různých urychlujících technik, které však mohou vést k nenalezení některých optimálních řešení. Na druhou stranu ale bylo ukázáno, že s některými kombinacemi těchto technik je možné nalézt optimální řešení v prakticky použitelném průměrném čase 5 sekund.

2.3 Hledání optimální cesty v grafu s nejistými cenami hran

V některých úlohách hledání nejkratší cesty nemusí být ceny některých hran v grafu předem známé nebo určené jednou konkrétní hodnotou. Možnou variantou problému, kde se vyskytuje určitá nejistota, je například problém popsaný v podsekci 2.3.1, kde ve známém grafu mohou být některé hrany zablokovány a být tak plně neprůjezdné. Jiný problém je popsán v podsekci 2.3.2, kde skutečné ceny hran nejsou předem známé, ale je známé jejich pravděpodobnostní rozdělení.

Jiná forma nejistoty se může vyskytovat při plánování cest v dopravních sítích. V [4] je například zmíněna nejistota ve formě zpoždění některého dopravního prostředku nebo nejistoty při přestupu. Tyto typy úloh se však již od naší úlohy výrazně liší.

2.3.1 Problém kanadského cestujícího

Problémem hledání optimální cesty v grafu, kdy průchodnost hran není předem známá a je zjištěna až během cesty, se zabývá tzv. *problém kanadského cestujícího* (Canadian Traveller Problem, CTP) definovaný v článku [12].

Řidič se pohybuje v známém prostředí a snaží se dostat z počátečního bodu do cíle. Během jeho cesty se však některé hrany mohou stát neprůjezdné (např. kvůli zasypání sněhem). Jestli je určitá hrana průjezdná či nikoliv lze zjistit až v s ní sousedícím uzlu. Cílem tohoto problému je nalézt optimální strategii pro cestu z počátečního bodu do cíle, která minimalizuje očekávanou cenu cesty. Kritériem minimalizace může být například tzv. *worst–case ratio*, poměr mezi délkou cesty podle dané strategie a délkou nejkratší cesty, která by byla nalezena při znalosti cen všech hran v grafu.

Existuje několik variant CTP úloh. Některé z nich jsou studovány v článku [13], kde je studována například varianta, kde zablokované hrany mohou být po určité době odblokovány, nebo varianta, kde je shora omezen počet hran, které mohou být zablokovány.

Vzhledem k tomu, že se jedná spíše o dynamický problém, neboť je uvažováno získávání nových informací během cesty, nelze ho v případě naší úlohy (tedy plánování trasy předem) jednoduše použít. Navíc CTP je spíše o hledání optimální strategie pro procházení grafu než o hledání konkrétní cesty.

2.3.2 Stochastický problém nejkratší cesty

Jiným problémem hledání optimální cesty v grafu s nejistými cenami hran se zabývá *stochastický problém nejkratší cesty* (Stochastic Shortest Path Problem with Recourse) studovaný v článku [14].

Formulace tohoto problému je velmi podobná problému CTP. V tomto případě je graf stochastický (tj. ceny hran jsou náhodné). Cena každé hrany je realizována náhodnou proměnnou s předem známým pravděpodobnostním rozdělením. Skutečné ceny hran jsou však zjišťovány až průběžně při průchodu grafem.

Jako příklad možného problému je v článku zmíněna situace jízdy autem, kdy skutečná úroveň provozu (a tedy i např. možné dopravní zácpy) je zjišťována v průběhu cesty až na jednotlivých křižovatkách.

Stejně jako CTP i tento problém je spíše o hledání optimální strategie (tj. pravidel, jak se rozhodovat s ohledem na aktuálně známé informace) než o hledání konkrétní cesty. Cílem stochastického problému nejkratší cesty je totiž nalézt strategii, která minimalizuje střední hodnotu součtu cen všech hran, po kterých vozidlo projelo než se dostalo do cílového uzlu.

Kapitola 3

Formulace problému

V této kapitole zformulujeme problém plánování cest s nejistou průchodností hran.

Pokud bychom plánovali cestu pouze podle její průchodnosti, velmi často by naplánovaná cesta byla příliš dlouhá. Domníváme se, že pro praktické účely je při plánování potřeba vzít v úvahu nejen průchodnost cest ale i jejich délku. Proto jsme se rozhodli problém zformulovat jako multikriteriální problém, kde je možné vzít v úvahu jak délku cesty, tak její průchodnost.

Nakonec jsme se rozhodli zformulovat 2 podobné problémy. Prvním problémem je obecný problém, jehož cílem je nalézt celou množinu všech Pareto-optimálních cest. Cílem druhého problému je nalézt nejkratší cestu z této množiny tak, aby bylo splněno dané omezení na její průchodnost.

Pro formulaci těchto problémů nejprve zadefinujeme několik potřebných pojmů. Začneme definováním pojmů z teorie grafů inspirované textem přednášky [15] a pojmů, které budou potřeba pro definici našich problémů. Poté popíšeme, jak jsme volili vhodné kritérium průchodnosti a v závěru kapitoly zformulujeme oba problémy.

3.1 Orientovaný graf

Orientovaný graf G je trojice (V, E, ϵ), kde

- V je neprázdná konečná množina *vrcholů (uzlů)*,
- E je konečná množina *hran*,
- *vztah incidence* ϵ je přiřazení, které každé hraně $e \in E$ přiřadí uspořádanou dvojici vrcholů.

Jestliže pro hranu $e \in E$ a vrcholy $u, v \in V$ platí $\epsilon(e) = (u, v)$, pak vrchol u nazveme *počátečním vrcholem hrany* e a vrchol v *koncovým vrcholem hrany* e.

3.2 Cesta

Mějme orientovaný graf $G = (V, E, \epsilon)$. Posloupnost vrcholů a hran

$$v_1, e_1, v_2, e_2, \ldots, v_{k-1}, e_{k-1}, v_k$$

grafu G nazveme *cestou* z uzlu v_1 do uzlu v_k, pokud pro každou hranu e_i této posloupnosti platí, že uzel v_i je jejím počátečním vrcholem a uzel v_{i+1} je jejím koncovým vrcholem.

Poznamenejme, že v teorii grafů (například v [15]) bývá v definici cesty ještě zakázáno opakování uzlů a hran. V úlohách hledání nejkratší cesty však bývají tyto podmínky vypuštěny, a proto se nevyskytují ani v této definici.

3.3 Graf dopravní sítě s nejistou průchodností hran

Graf dopravní sítě s nejistou průchodností hran G je trojice (G', d, p), kde

- $G' = (V, E, \epsilon)$ je orientovaný graf,
- d: $E \to \mathbb{R}_{>0}$ je zobrazení přiřazující každé hraně vzdálenost mezi jejím počátečním a koncovým vrcholem,
- p: $E \to \langle 0, 1 \rangle$ je zobrazení přiřazující každé hraně pravděpodobnost, že hrana bude průchodná.

3.4 Cena cesty

Nechť $\pi = v_1, e_1, v_2, e_2, \ldots, v_{k-1}, e_{k-1}, v_k$ je cesta v grafu dopravní sítě s nejistou průchodností hran $G = ((V, E, \epsilon), d, p)$, pak

- její *délku* $c_d(\pi)$ definujeme jako je součet délek jejích hran, tedy

$$c_d(\pi) = \sum_{i=1}^{k-1} d(e_i),$$

- její *váženou neprůchodnost* $c_p(\pi)$ jako součet součinů délek a pravděpodobností neprůchodu jejích hran, tedy

$$c_p(\pi) = \sum_{i=1}^{k-1} d(e_i)(1 - p(e_i)),$$

- její *cenu* $c(\pi)$ jako dvojici $(c_d(\pi), c_p(\pi))$.

Prvním kritériem v naší úloze tedy bude délka cesty a druhým bude kritérium průchodnosti. Proč jsme pro kritérium průchodnosti zvolili právě váženou neprůchodnost popíšeme dále v sekci 3.6.

3.5 Relace dominance

Jak již bylo zmíněno při popisu multikriteriálního hledání cest, pro porovnání dvou vektorů cen je potřeba zavést vztah dominance.

Dominance dvou cen cest $c_1 = (d_1, p_1)$, $c_2 = (d_2, p_2)$ přímo z definice by v našem případě byla určená vztahem

$$c_1 \prec c_2 \quad \Leftrightarrow \quad d_1 \leq d_2 \wedge p_1 \leq p_2 \wedge c_1 \neq c_2.$$

Za účelem zrychlení algoritmu je možné využít slabší definici dominance, která se liší pouze tím, že jsou navíc dominovány i stejné ceny. Dominanci tedy definujeme následovně.

Nechť $c_1 = (d_1, p_1)$ a $c_2 = (d_2, p_2)$ jsou dvě ceny cest, pak platí

$$c_1 \preceq c_2 \quad \Leftrightarrow \quad d_1 \leq d_2 \wedge p_1 \leq p_2.$$

Tato definice sice neumožní nalézt cesty se stejnou cenou, nicméně tyto cesty se většinou liší jen na velmi krátkém úseku (například v jednom uzlu a dvou hranách), a tím jsou pro výslednou množinu cest nezajímavé. Navíc v některých případech může tento přístup vést k výrazné redukci velikosti Pareto-množiny, a tedy i k výraznému zrychlení. Proto jsme se rozhodli v našem případě využít tuto slabší definici dominance.

3.6 Volba kritéria průchodnosti

Kritérium pro délku cesty jsme v našem multikriteriálním problému zvolili jednoduše jako minimalizaci součtu délek jednotlivých hran na cestě. Zdůvodnění, jak jsme uvažovali při volbě kritéria průchodnosti a proč jsme zvolili právě váženou neprůchodnost, nabízí tato sekce.

Pro hledání nejkratší cesty pomocí standardních algoritmů (Dijkstra, A*) je potřeba, aby kritérium splňovalo určité vlastnosti. Tyto vlastnosti bychom chtěli i u kritéria průchodnosti. Potřebovali bychom tedy, aby kritérium bylo **po hranách aditivní**, aby bylo **minimalizováno** a aby jeho hodnoty během cesty byly **neklesající** (tj. aby se přidáním hrany nemohla cena cesty snížit).

3.6.1 Pravděpodobnost průchodu cesty

Jedním z možných kritérií by mohla být přímo maximalizace *pravděpodobnosti průchodu cesty*, která by, za předpokladu nezávislosti pravděpodobností průchodu jednotlivých hran na cestě, byla určená jako součin pravděpodobností průchodu jednotlivých hran na cestě vzorcem

$$\prod_{e_i \in \pi} p(e_i). \tag{3.1}$$

Toto kritérium sice není přímo aditivní, ale je možné ho s využitím vlastností logaritmu na aditivní kritérium relativně jednoduše převést. Jednotlivé kroky převodu jsou přehledně popsány na analogickém případu hledání nejspolehlivější cesty v článku [16], kde jsou použity úpravy z článku [17]. Navíc by se nám při tomto převodu vyřešil i druhý problém tohoto kritéria a to, že maximalizaci bychom převedli na minimalizaci. Využitím tohoto převodu

bychom během prohledávání minimalizovali součty opačných hodnot logaritmů pravděpodobností průchodu hran na cestě, tedy $\sum_{e_i \in \pi} -\log(p(e_i))$, a až na závěr bychom toto kritérium převedli zpět, a tím získali pravděpodobnost průchodu cesty podle vzorce 3.1.

Ačkoliv toto transformované kritérium již splňuje všechny vytyčené vlastnosti, nepovažujeme ho za vhodné pro náš problém. Důvodem pro to je především neuvažování délek jednotlivých hran a závislost na jejich počtu. Tyto důvody jsou ilustrovány v následujících dvou příkladech.

Příklad 3.1 (Krátký neprůchodný úsek)**.** Mějme cestu s délkou 1 km tvořenou několika hranami. Tato cesta má jednu hranu s pravděpodobností průchodu 0.1, která tvoří jen pětimetrový úsek z celé cesty. Všechny ostatní hrany mají pravděpodobnost průchodu 1.

Použitím vzorce 3.1 by této cestě byla přiřazena průchodnost 0.1. A tedy ačkoliv je tato cesta až na malý úsek plně průchodná, krátký neprůchodný úsek by způsobil, že i průchodnost celé cesty by byla nízká. Přitom je možné, že tento krátký by úsek mohlo jít v mnoha případech nějak přejít, přenést či obejít.

Příklad 3.2 (Závislost na počtu hran)**.** Mějme 2 cesty se stejnou délkou 1 km. První cesta je tvořená jedinou hranou s pravděpodobností průchodu 0.9. Druhá cesta je tvořena 5 hranami, kde každá hrana má stejnou pravděpodobnost průchodu 0.9.

Intuice říká, že vzhledem k tomu, že obě cesty mají na stejně dlouhém úseku po celou dobu pravděpodobnost průchodu 0.9, měly by mít obě cesty i stejnou průchodnost. Nicméně, pokud by byl pro průchodnost cesty využit vztah 3.1, první cesta by měla průchodnost 0.9, zatímco cesta tvořená více hranami by měla výrazně nižší průchodnost $0.9^5 \doteq 0.59$.

3.6.2 Průměrná (ne)průchodnost cesty

Variantou, která bere ohled na délku jednotlivých hran i na jejich počet, je *průměrná průchodnost cesty*. Tou označujeme vážený aritmetický průměr pravděpodobností průchodu jednotlivých hran na cestě, kde váhy jsou délky příslušných hran, tedy

$$\frac{\sum_{e_i \in \pi} d(e_i) p(e_i)}{\sum_{e_i \in \pi} d(e_i)}, \tag{3.2}$$

kde vzhledem k tomu, že délky jednotlivých hran jsou kladné, víme, že i $\sum_{e_i \in \pi} d(e_i) > 0$.

Takto definované kritérium se chová rozumněji nejen na příkladech 3.1 a 3.2. Konkrétně pro příklad 3.1 je průměrná průchodnost 0.9955 a pro příklad 3.2 je 0.9. Na druhou stranu, jakožto průměr bohužel nesplňuje vlastnosti pro použití standardních algoritmů hledání nejkratší cesty.

Vezmeme-li pouze čitatele ze vzorce 3.2, tedy $\sum_{e_i \in \pi} d(e_i) p(e_i)$, jedná se již o neklesající kritérium, které lze jednoduše sčítat. Navíc přitom bere v úvahu délky jednotlivých úseků.

Stejná úprava byla použita i v článku [10], kde byl řešen bikriteriální problém s podobnými kritérii. Vzhledem k tomu, že tato kritéria nemají

na řešení problému vliv, pro přehlednost popíšeme použitý převod úloh v našem značení. V článku tedy byla zformulována základní úloha, kde prvním kritériem byla minimalizace délky cesty a v druhém kritériu byla maximalizace váženého aritmetického průměru, kde váhami byla právě délka jednotlivých hran. K této úloze pak byla zformulována modifikovaná úloha, kde jako první kritérium zůstala minimalizace délky a v druhém kritériu byla místo maximalizace váženého aritmetického průměru využita jen maximalizace jeho čitatele.

V tomto článku byla dále dokázána implikace, že je-li π^* optimální (nedominované) řešení základní úlohy, pak je π^* také optimálním (nedominovaným) řešením modifikované úlohy. Bohužel pro maximalizaci kritéria nelze využít algoritmy hledání nejkratší cesty. Proto pro řešení problému byl v článku využit algoritmus, který nalezne délku nejkratší cesty D_{min}, a pak hledá všechny cesty s délkou mezi D_{min} a $D_{min} + \delta$. Z nalezených řešení modifikované úlohy byla nakonec eliminována řešení, která nebyla optimální v základní úloze.

V našem případě jsme se rozhodli využít toho, že známe-li pravděpodobnost $p(e_i)$, že hrana bude průchodná, známe také pravděpodobnost opačného jevu $1 - p(e_i)$, že hrana průchodná nebude. A tedy jsme se rozhodli, že místo maximalizace průchodnosti cesty využijeme minimalizaci její neprůchodnosti. Jako *průměrnou neprůchodnost cesty* tedy označíme vážený aritmetický průměr s pravděpodobností, že hrana průchodná nebude, tedy

$$\frac{\sum_{e_i \in \pi} d(e_i)(1 - p(e_i))}{\sum_{e_i \in \pi} d(e_i)}. \tag{3.3}$$

Jakožto průměr má toto kritérium opět problém s monotonicitou, ale převedením tohoto kritéria na úlohu minimalizace jeho čitatele, tedy

$$\sum_{e_i \in \pi} d(e_i)(1 - p(e_i)), \tag{3.4}$$

již splňuje všechny 3 vytyčené vlastnosti. Je po hranách aditivní, jedná se o minimalizaci a hodnoty tohoto kritéria během cesty budou neklesající, neboť se přičítají pouze nezáporné hodnoty. Přitom jsou uvažovány i délky jednotlivých hran. Připomeňme, že toto kritérium je námi zvolená vážená neprůchodnost cesty.

Vážená neprůchodnost cesty je tedy vhodným kritériem pro naši úlohu. Jediným problémem je jeho horší interpretovatelnost. Bohužel v případě minimalizace již nelze použít výše uvedený převod úloh, který byl použit v článku [10].

Označíme-li pro přehlednost délku cesty $D(\pi) = \sum_{e_i \in \pi} d(e_i)$ a váženou neprůchodnost cesty $N(\pi) = \sum_{e_i \in \pi} d(e_i)(1-p(e_i))$, kde $D(\pi) > 0$ a $N(\pi) \geq 0$, pak tedy **neplatí**, že je-li π^* optimální (nedominované) řešení úlohy minimalizace ceny

$$(D(\pi), \frac{N(\pi)}{D(\pi)}), \tag{3.5}$$

pak je π^* také optimálním (nedominovaným) řešením úlohy minimalizace ceny

$$(D(\pi), N(\pi)). \tag{3.6}$$

Jako protipříklad lze uvést graf se dvěma paralelními hranami. První cesta má délku 100 a pravděpodobnost průchodu 0.1. Druhá cesta má délku 1000 a pravděpodobnost průchodu 0.9. Zatímco v úloze 3.6 je nalezena jen první cesta ($100 < 1000$ a $100 \cdot (1 - 0.1) = 90 < 100 = 1000 \cdot (1 - 0.9)$), v úloze 3.5 jsou optimální obě cesty ($100 < 1000$ a $\frac{100 \cdot (1-0.1)}{100} = 0.9 > 0.1 = \frac{1000 \cdot (1-0.9)}{1000}$).

Na druhou stranu však platí opačná implikace, která nám umožní interpretovat cenu kritéria vážené neprůchodnosti cesty (vzorec 3.4) jako průměrnou neprůchodnost určenou vztahem 3.3. Tuto implikaci formuluje následující tvrzení. Důkaz je založen na stejné myšlence jako důkaz tvrzení umožňující převod úloh v článku [10].

Tvrzení 3.3. *Je-li π^* optimální (nedominované) řešení úlohy minimalizace ceny 3.6, pak je π^* také optimálním (nedominovaným) řešením úlohy minimalizace ceny 3.5.*

Důkaz. Toto tvrzení dokážeme sporem. Nechť π^* je optimální řešení úlohy 3.6. Naopak předpokládejme, že π^* není optimálním řešením úlohy 3.5. Tudíž existuje cesta π' taková, že $\pi' \prec \pi^*$, a tedy

$$D(\pi') < D(\pi^*) \quad \wedge \quad \frac{N(\pi')}{D(\pi')} \leq \frac{N(\pi^*)}{D(\pi^*)} \tag{3.7}$$

nebo

$$D(\pi') \leq D(\pi^*) \quad \wedge \quad \frac{N(\pi')}{D(\pi')} < \frac{N(\pi^*)}{D(\pi^*)}. \tag{3.8}$$

Ukážeme, že oba případy (3.7, 3.8) vedou ke sporu.

1. (nerovnice 3.7 vedou ke sporu)

   Vynásobením druhé nerovnosti ze vztahu 3.7 oběma kladnými jmenovali dostaneme nerovnost

   $$N(\pi')D(\pi^*) \leq N(\pi^*)D(\pi').$$

   S využitím první nerovnosti vztahu 3.7 a faktu, že všechny výrazy v nerovnici jsou nezáporné, můžeme v případě $N(\pi') > 0$ levou stranu této nerovnosti zdola ostře omezit výrazem $N(\pi')D(\pi')$. Tím získáme nerovnice

   $$N(\pi')D(\pi') < N(\pi')D(\pi^*) \leq N(\pi^*)D(\pi'),$$

   a tedy i nerovnici

   $$N(\pi')D(\pi') < N(\pi^*)D(\pi'). \tag{3.9}$$

   Vydělíme-li tuto nerovnici kladným výrazem $D(\pi')$, získáme nerovnici

   $$N(\pi') < N(\pi^*), \tag{3.10}$$

   která je však ve **sporu** s tím, že π^* je optimální řešení úlohy 3.6.

   Během úprav jsme vynechali případ $N(\pi') = 0$. V tomto případě (z druhé nerovnosti vztahu 3.7) musí být $N(\pi^*) \geq 0$.

- Případ $N(\pi^*) > 0$ vede na nerovnici 3.10 (a tedy i ke sporu).
- Pro případ $N(\pi^*) = 0$ tvrzení platí, protože v tomto případě by se obě úlohy 3.5 a 3.6 zredukovaly na stejnou úlohu – minimalizaci $D(\pi)$.

2. (nerovnice 3.8 vedou ke sporu)

   Podobnými úpravami lze ukázat, že i nerovnosti 3.8 vedou na nerovnici 3.9, a tedy i ke sporu s tím, že π^* je optimální řešení úlohy 3.6. Konkrétně vynásobením druhé nerovnosti z 3.8 oběma kladnými jmenovali dostaneme nerovnost

   $$N(\pi')D(\pi^*) < N(\pi^*)D(\pi').$$

   S využitím první nerovnosti vztahu 3.8 a faktu, že všechny výrazy v nerovnici jsou nezáporné, můžeme levou stranu této nerovnosti zdola omezit výrazem $N(\pi')D(\pi')$. Tím získáme nerovnice

   $$N(\pi')D(\pi') \leq N(\pi')D(\pi^*) < N(\pi^*)D(\pi'),$$

   a tedy i nerovnici 3.9, jež vede ke **sporu** (viz první část důkazu).

V obou případech jsme odvodili spor s tím, že π^* je optimální řešení úlohy 3.6. Tím jsme ukázali, že π^* je také optimálním řešením úlohy 3.5. □

Pro prohledávání jsme tedy jako nejvhodnější kritérium průchodnosti zvolili minimalizaci *vážené neprůchodnosti* (vzorec 3.4). Aby bylo možné ceny nalezených cest rozumně interpretovat, budeme ceny kritéria vážené neprůchodnosti převádět na *průměrnou neprůchodnost cesty* (vzorec 3.3) a *průměrnou průchodnost cesty* (vzorec 3.2). Převod mezi průměrnou průchodností a neprůchodností je totiž možný díky následujícímu tvrzení.

Tvrzení 3.4. *Úloha minimalizace průměrné neprůchodnosti cesty (vzorec 3.3) a úloha maximalizace průměrné průchodnosti cesty (vzorec 3.2) mají stejná řešení.*

Důkaz. Označíme-li $c_{pp}(\pi)$ průměrnou průchodnost cesty (vzorec 3.2) a $c_{pn}(\pi)$ průměrnou neprůchodnost cesty (vzorec 3.3), pak protože platí následující vztah mezi $c_{pp}(\pi)$ a $c_{pn}(\pi)$

$$\begin{aligned} c_{pn}(\pi) &= \frac{\sum\limits_{e_i \in \pi} d(e_i)(1 - p(e_i))}{\sum\limits_{e_i \in \pi} d(e_i)} \\ &= \frac{\sum\limits_{e_i \in \pi} d(e_i) - \sum\limits_{e_i \in \pi} d(e_i)p(e_i)}{\sum\limits_{e_i \in \pi} d(e_i)} \\ &= 1 - \frac{\sum\limits_{e_i \in \pi} d(e_i)p(e_i)}{\sum\limits_{e_i \in \pi} d(e_i)}, \\ &= 1 - c_{pp}(\pi) \end{aligned}$$

a následující vztahy mezi funkcemi argmin a argmax

$$\operatorname*{argmax}_{\pi} c_{pp}(\pi) = \operatorname*{argmin}_{\pi} \{1 - c_{pp}(\pi)\}$$
$$\operatorname*{argmax}_{\pi} c_{pp}(\pi) = \operatorname*{argmin}_{\pi} \{-c_{pp}(\pi)\}$$

víme, že platí

$$\operatorname*{argmax}_{\pi} c_{pp}(\pi) = \operatorname*{argmin}_{\pi} c_{pn}(\pi).$$

□

3.7 Množina optimálních cest

Mějme graf dopravní sítě s nejistou průchodností hran G, počáteční uzel s a koncový uzel t. *Množina optimálních cest* $\Pi^*_{s,t}$ je množina všech cest z s do t, jejichž cena (dvojice $(c_d(\pi), c_p(\pi))$ definovaná v sekci 3.4) není dominovaná jakoukoliv jinou cestou z s do t.

Pro každé dvě cesty $\pi, \pi' \in \Pi^*_{s,t}$ tedy platí, že $c(\pi) \not\preceq c(\pi')$ a zároveň $c(\pi') \not\preceq c(\pi)$.

3.8 Multikriteriální problém plánování cest s nejistou průchodností hran

Mějme graf dopravní sítě s nejistou průchodností hran G, počáteční uzel s a koncový uzel t. *Multikriteriální problém plánování cest s nejistou průchodností hran* je úloha minimalizace ceny cesty $c(\pi) = (c_d(\pi), c_p(\pi))$, kde cílem je nalézt množinu všech optimálních cest $\Pi^*_{s,t}$ vedoucích z s do t.

3.9 Problém hledání nejkratší cesty s omezením průchodnosti

Mějme graf dopravní sítě s nejistou průchodností hran G, počáteční uzel s a koncový uzel t a horní mez vážené neprůchodnosti b. Pak *problém hledání nejkratší cesty s omezením průchodnosti* je nalézt nejkratší cestu π^* z množiny optimálních cest $\Pi^*_{s,t}$ vedoucích z s do t, jejíž vážená neprůchodnost je menší nebo rovna b, tedy

$$\pi^* = \operatorname*{argmin}_{\pi \in \Pi^*_{s,t}} c_d(\pi) \quad \text{za podmínky} \quad c_p(\pi) \leq b.$$

Kapitola 4
Řešení problému

V této kapitole popíšeme algoritmy, které řeší problémy definované v předchozí kapitole. Konkrétně tedy řeší multikriteriální problém plánování cest s nejistou průchodností hran a problém hledání nejkratší cesty s omezením průchodnosti.

Vzhledem k tomu, že cesta, jež je řešením problému s omezením, je také jedna z množiny optimálních cest, která je řešením multikriteriálního problému, jsou si oba algoritmy velmi podobné. Proto nejprve v sekci 4.1 popíšeme obecný multikriteriální algoritmus, který je společnou kostrou obou navržených algoritmů, a následně v sekcích 4.2 a 4.3 od sebe tyto algoritmy odlišíme.

4.1 Obecný multikriteriální algoritmus

Tato sekce popisuje obecný multikriteriální algoritmus, který je implementací algoritmu NAMOA* [7], jenž je zmíněn v sekci 2.2.1 a upraven na naši úlohu.

Algoritmus využívá několik datových struktur, které popíšeme v podsekci 4.1.1. Dále v podsekci 4.1.2 objasníme účel jednotlivých abstraktních funkcí, které jsou v algoritmu využity, v podsekci 4.1.3 zmíníme možné heuristické funkce a v závěrečné podsekci 4.1.4 popíšeme algoritmus.

Na úvod poznamenejme, že kvůli větší přehlednosti jsme popis algoritmu mírně zjednodušili v tom, že místo množiny nedominovaných cest nachází pouze množinu jejich cen. Způsob, jak lze tento algoritmus jednoduše upravit tak, aby nalezl přímo cesty, je ilustrován v závěru této sekce.

4.1.1 Datové struktury použité v algoritmu

V algoritmu jsou použity tyto datové struktury:

- *stav (label)* je trojice $(u, c(u), e(u))$, kde
  - u je uzel,
  - $c(u) = (c_d(u), c_p(u))$ je současná cena cesty z počátečního uzlu s do uzlu u v tomto stavu,
  - $e(u) = (c_d(u) + h_d(u), c_p(u) + h_p(u))$ je odhadovaná cena celé cesty v tomto stavu (tj. součet současné ceny cesty z počátečního uzlu s

do u a odhadu ceny cesty z u do cílového uzlu t určeného heuristickou funkcí $h(u,t) = (h_d(u), h_p(u))$, která je vysvětlena podrobněji v podsekci 4.1.3).

- prioritní fronta Q stavů $(u, c(u), e(u))$, udržující si nalezené neexpandované stavy. Stavy jsou z fronty odebírány na základě lexikografického uspořádání jejich odhadované ceny cesty $e(u)$. Obsahuje-li tedy fronta stavy $l_u = (u, c(u), e(u))$ a $l_v = (v, c(v), e(v))$ s odhadovanými cenami $e(u) = (e_d(u), e_p(u))$ a $e(v) = (e_d(v), e_p(v))$, z fronty bude dříve odebrán stav l_u právě tehdy, když $e_d(u) < e_d(v)$ nebo $e_d(u) = e_d(v) \wedge e_p(u) \leq e_p(v)$.
- množina $C_{solutions}$ obsahující nalezené nedominované ceny cest z počátečního uzlu s do cílového uzlu t.
- pro každý uzel u máme množinu $C_{open}[u]$, která obsahuje otevřené ceny cest z s do u (tj. ceny stavů, které jsou ještě ve frontě Q).
- pro každý uzel u máme množinu $C_{closed}[u]$, která obsahuje uzavřené ceny cest z s do u (tj. ceny stavů, které již byly vytaženy z fronty Q).

4.1.2 Abstraktní funkce v algoritmu

Jak již bylo zmíněno, v navrženém algoritmu 1 používáme abstraktní funkce, jejichž implementace se může lišit pro konkrétní úlohy. Tyto funkce a jejich účel vysvětlují následující odrážky.

- Funkce $isSuitableEstimate$ rozhoduje, jestli má v současném stavu prohledávání smysl dále expandovat stav s danou odhadovanou cenou cesty $e(u)$. Víme-li totiž, že z tohoto stavu již nebude možné nalézt optimální řešení, není nutné ho dále uvažovat. V závislosti na tom, jestli může být nalezena cesta s cenou $e(u)$, vrací funkce pravdivostní hodnotu $true$ nebo $false$.
- Funkce $terminationCondition$ v závislosti na současném stavu prohledávání určuje, jestli se má prohledávání ukončit (vrací hodnotu $true$) či dále pokračovat (vrací hodnotu $false$).
- Funkce $pruneAfterFoundSolution$ je volána po nalezení řešení s danou cenou $c(solution)$. Jejím cílem je odstranit z fronty Q stavy, které již nemohou vést k nalezení optimálního řešení.

4.1.3 Možné heuristické funkce

Máme-li o problému určitou znalost, je vhodné ji využít pro urychlení algoritmu. Proto je v algoritmu použita heuristická funkce $h(u,t)$, která této informace využívá. Tato funkce vrací odhadovanou cenu cesty z daného uzlu u do cílového uzlu t. Protože máme cenu složenou ze 2 složek, je i tato heuristická funkce složena ze 2 heuristik pro jednotlivá kritéria. V úvahu přichází hned několik možných heuristik.

Nejjednodušší heuristikou, je tzv. *nulová heuristika*, která vždy vrací hodnotu 0. Tato heuristika v prohledávání nepřidá žádnou informaci. Jde tedy o neinformované prohledávání, které z algoritmu dělá jistou multikriteriální verzi Dijkstrova algoritmu. Nulová heuristika tedy k zrychlení algoritmu nevede.

Pokud se týká heuristiky vzdálenosti, k odhadu délky cesty do cílového uzlu t je možné využít znalosti geografických souřadnic obou uzlů. Vzhledem k tomu, že body nejsou v rovině, není možné pro výpočet vzdálenosti mezi nimi jednoduše využít eukleidovskou vzdálenost. Místo toho je vhodné zjednodušit tvar Země na kouli, což je pro účel heuristiky dostatečně přesná aproximace, a využít tzv. *Haversinovu formuli*[1], která umožní spočítat vzdálenost mezi dvěma body na kouli z jejich geografických souřadnic.

Vzhledem k tomu, že čas běhu multikriteriálního algoritmu bývá výrazně delší než v případě jediného kritéria, je možné si předpočítat přesné ceny cest z každého uzlu grafu pro jednotlivá kritéria. Tyto ceny lze získat použitím Dijkstrova algoritmu z cílového uzlu t do všech ostatních uzlů grafu v grafu, který vznikne obrácením směru všech jeho hran. Tento přístup byl představen v článku [18] jako součást navrženého multikriteriálního algoritmu.

Jak bylo zmíněno v podsekci 2.1.2, pro správné chování heuristických funkcí je potřeba, aby byly přípustné a ideálně i konzistentní. Naštěstí všechny tyto heuristické funkce jsou jak přípustné, tak i konzistentní, protože neodhadují větší vzdálenost než je skutečná a ani neporušují trojúhelníkovou nerovnost.

Pro kritérium vzdálenosti lze použít všechny tři tyto představené heuristiky (tj. nulovou, přímou vzdálenost a předpočítanou). Pro kritérium vážené neprůchodnosti je možné použít jen nulovou a předpočítanou heuristiku. U předpočítané heuristiky pak bude jediný rozdíl a to, že předpočítaná cena cesty bude místo délky cesty její vážená neprůchodnost. Vliv jednotlivých kombinací těchto heuristik dále vyhodnocujeme v podsekci 6.5.1.

4.1.4 Popis algoritmu

V této podsekci popíšeme obecný multikriteriální algoritmus využívající abstraktní funkce *isSuitableEstimate*, *terminationCondition* a *pruneAfterFoundSolution*, jehož pseudokód je zobrazen v algoritmu 1.

Vstupem algoritmu je graf dopravní sítě s nejistou průchodností hran $G = ((V, E, \epsilon), d, p)$, počáteční uzel s a cílový uzel t. Výstupem algoritmu je množina nedominovaných cen cest vedoucích z s do t, tedy množina cen odpovídající cestám z množiny optimálních cest $\Pi^*_{s,t}$, jež byla definována v sekci 3.7.

Algoritmus začíná inicializací jednotlivých datových struktur na řádcích 1 – 4. Po jejich inicializaci je pomocí funkce *addToOpenQueue* vytvořen nový stav, odpovídající počátečnímu uzlu s a ceně $(0, 0)$, který je přidán do příslušných datových struktur.

Po počáteční inicializaci v cyklu na řádcích 6 – 26 postupně probíhá expandování vždy dosud nejlepšího stavu z fronty Q. Expandování probíhá

[1] https://en.wikipedia.org/wiki/Haversine_formula

Algoritmus 1: obecný multikriteriální algoritmus

input : graph $G = ((V, E, \epsilon), d, p)$, source node s, target node t
output : set of nondominated costs of paths from s to t
$\{c(\pi) \mid \pi \in \Pi^*_{s,t}\}$

```
1  Q := ∅
2  C_open[u] := ∅, ∀u ∈ V
3  C_closed[u] := ∅, ∀u ∈ V
4  C_solutions := ∅
5  addToOpenQueue(s, (0, 0))
6  while Q ≠ ∅ do
7      (u, c(u), e(u)) := poll(Q)
8      C_open[u] := C_open[u] \ {c(u)}
9      C_closed[u] := C_closed[u] ∪ {c(u)}
10     if u = t then
11         C_solutions := C_solutions ∪ {c(u)}          // found solution
12         if terminationCondition() then
13             break
14         end
15         pruneAfterFoundSolution(c(u))
16     else
17         foreach (e, v) ∈ E × V : ε(e) = (u, v) do
18             c(v) := c(u) + (d(e), d(e) · (1 − p(e)))
19             if ∄c ∈ C_open[v] ∪ C_closed[v] : c ⪯ c(v) then
20                 C_open[v] := C_open[v] \ {c ∈ C_open[v] | c(v) ⪯ c}
21                 C_closed[v] := C_closed[v] \ {c ∈ C_closed[v] | c(v) ⪯ c}
22                 addToOpenQueue(v, c(v))
23             end
24         end
25     end
26 end
27 return C_solutions
28
29 Function addToOpenQueue(u, c(u)) :
30     e(u) := c(u) + h(u, t)
31     if isSuitableEstimate(e(u)) then
32         Q := Q ∪ {(u, c(u), e(u))}
33         C_open[u] := C_open[u] ∪ {c(u)}
34     end
35 End Function
```

do té doby, než jsou expandovány všechny stavy z Q (a tedy než je tato fronta prázdná) nebo než je po nalezení řešení splněna ukončovací podmínka *terminationCondition*. Jednu iteraci algoritmu popisuje několik následujících odstavců.

Nejprve je z fronty Q na základě dříve popsané priority vybrán stav $(u, c(u), e(u))$. Cena tohoto stavu je odebrána z množiny otevřených cen $C_{open}[u]$ a je přidána do množiny uzavřených cen $C_{closed}[u]$.

Na řádku 10 následuje kontrola, jestli je uzel u cílový uzel t. V případě, že se jedná o cílový uzel, je nalezena cena jednoho z optimálních řešení $c(u)$, která je přidána do množiny cen $C_{solutions}$. Je-li poté splněna ukončovací podmínka *terminationCondition*, algoritmus ukončí cyklus expanze a vrací nalezenou množinu $C_{solutions}$ na řádku 27. Není-li tato podmínka splněna, je aplikována ořezávací funkce $pruneAfterFoundSolution$ a algoritmus pokračuje dalším cyklem na řádku 6.

Pokud uzel u není cílový, jsou na řádcích 17 – 24 postupně procházeny všechny hrany e, jejichž počátečním uzlem je právě uzel u, spolu s uzly v, které jsou koncovými uzly těchto hran. Pro každý sousední uzel v je vypočtena cena $c(v)$ jako součet odpovídajících si složek ceny $c(u)$ a ceny hrany e. Dále je na řádku 19 cena $c(v)$ porovnávána s cenami dosud nalezených cest do uzlu v a je studováno, jestli má být tento nový stav dále expandován.

Nebyla-li zatím nalezena cesta do uzlu v s cenou, která by dominovala cenu $c(v)$ (a tedy neobsahuje-li ani jedna z množin $C_{open}[v]$ nebo $C_{closed}[v]$ cenu, která by dominovala cenu $c(v)$), jsou z množin $C_{open}[v]$ a $C_{closed}[v]$ eliminovány všechny ceny, které jsou dominované cenou $c(v)$. Poté je na daný uzel v a cenu $c(v)$ použita funkce *addToOpenQueue*.

Funkce *addToOpenQueue*, jejíž pseudokód je zobrazen na řádcích 29 – 35, má argumenty uzel u a cenu cesty do tohoto uzlu $c(u)$. Ve funkci je nejprve součtem ceny $c(u)$ a odhadu heuristické funkce $h(u, t)$ vypočtena odhadovaná cena $e(u)$. Je-li tato odhadovaná cena vhodná k další expanzi (podle funkce *isSuitableEstimate*), je tento nový stav $(u, c(u), e(u))$ přidán do fronty Q a jeho cena je přidána do množiny otevřených cen příslušného uzlu $C_{open}[u]$.

V případě, že neplatí podmínka na řádku 19, byla již do uzlu v nalezena cesta s lepší cenou, a tak nemá smysl tento stav dále expandovat. Algoritmus pokračuje navštívením dalšího sousedního uzlu uzlu u na řádku 17.

Poté, co jsou navštíveny všechny sousední uzly uzlu u, expandování daného stavu končí a algoritmus pokračuje v dalším cyklu na řádku 6.

Je-li fronta Q prázdná, byly již expandovány všechny stavy, které by mohly vést k nalezení řešení, a algoritmus na řádku 27 vrací nalezenou množinu nedominovaných cen cest $C_{solutions}$.

Jak již bylo uvedeno, popis algoritmu byl kvůli větší přehlednosti zjednodušen na nalezení nedominovaných cen. Pokud bychom chtěli získat konkrétní cesty, jediné změny by byly v tom, že kromě množiny nalezených cen $C_{solutions}$ bychom si navíc ještě museli udržovat množinu jejich stavů. Tyto stavy bychom rozšířili o referenci na předchozí stav prohledávání, kterou bychom museli při jeho vytváření nastavovat. Cesty bychom rekonstruovali v závěru algoritmu z uložených stavů zpětným procházením přes stavy.

4.2 Řešení multikriteriálního problému plánování cest s nejistou průchodností hran

V této sekci popíšeme algoritmus řešící multikriteriální problém plánování cest s nejistou průchodností hran, který je definován v sekci 3.8. Jak již bylo avizováno, tento algoritmus je implementací obecného multikriteriálního algoritmu z předchozí sekce. Pseudokód tohoto algoritmu, který je složen z implementovaných abstraktních funkcí, je zobrazen v algoritmu 2.

Algoritmus 2: algoritmus řešící multikriteriální problém plánování cest s nejistou průchodností hran

input : graph $G = ((V, E, \epsilon), d, p)$, source node s, target node t
output : set of all nondominated costs of paths from s to t $\{c(\pi) \mid \pi \in \Pi^*_{s,t}\}$
global data: all datastructures used in algorithm 1

```
1  Function isSuitableEstimate(e(u)):
2  |  return ∄c(solution) ∈ C_solutions : c(solution) ⪯ e(u)
3  End Function
4
5  Function terminationCondition():
6  |  return false
7  End Function
8
9  Function pruneAfterFoundSolution(c(solution)):
10 |  foreach (u, c(u), e(u)) ∈ Q : c(solution) ⪯ e(u) do
11 |  |  Q := Q \ {(u, c(u), e(u))}
12 |  end
13 End Function
```

Vstup algoritmu je stejný, jako u obecného algoritmu. Vstupem algoritmu je tedy graf dopravní sítě s nejistou průchodností hran $G = ((V, E, \epsilon), d, p)$, počáteční uzel s a cílový uzel t. Výstupem algoritmu je množina všech nedominovaných cen cest vedoucích z s do t, tedy množina cen všech cest z množiny optimálních cest $\Pi^*_{s,t}$, jež byla definována v sekci 3.7.

Dále jsou v algoritmu zpřístupněny všechny datové struktury použité v obecném algoritmu. Abstraktní funkce obecného multikriteriálního algoritmu jsou implementovány následovně.

- Funkce *isSuitableEstimate* porovnává odhadovanou cenu cesty $e(u)$ s dosud nalezenými řešeními v množině $C_{solutions}$. Není-li v této množině cena, která by dominovala odhad $e(u)$, může stav s touto odhadovanou cenou vést k nalezení optimálního řešení, a tak funkce vrací hodnotu *true*. V opačném případě víme, že stav s touto odhadovanou cenou by již k nalezení optimálního řešení nevedl, protože v algoritmu jsou použity

přípustné heuristiky, které neodhadují horší cenu než je skutečná, a tak je vrácena hodnota $false$.

- Vzhledem k tomu, že cílem multikriteriálního problému plánování cest s nejistou průchodností hran je nalézt množinu všech nedominovaných řešení, nemůže být prohledávání ukončeno dříve než jsou expandovány všechny stavy z fronty Q, a proto funkce $terminationCondition$ vrací pokaždé hodnotu $false$.

- Ve funkci $pruneAfterFoundSolution$, jsou z fronty Q odstraněny všechny stavy, které mají odhadovanou cenu, jež je dominovaná cenou nalezeného řešení $c(solution)$. To je možné provést kvůli tomu, že v algoritmu jsou použity přípustné heuristiky a že po nalezení řešení s cenou $c(solution)$ již nemohou být nalezena optimální řešení, která by byla cenou $c(solution)$ dominována.

Samotný algoritmus pak spočívá ve spuštění obecného multikriteriální algoritmu s argumenty G, s a t a takto implementovanými funkcemi. Tímto způsobem spuštěný obecný algoritmus vrátí nalezenou množinu řešení, která je rovnou i výstupem tohoto algoritmu.

4.3 Řešení problému hledání nejkratší cesty s omezením průchodnosti

Tato sekce popisuje algoritmus, který hledá jedno konkrétní řešení (nejkratší cestu, která splňuje podmínku maximální vážené neprůchodnosti) z množiny všech optimálních řešení úlohy multikriteriálního plánování cest s nejistou průchodností hran. Algoritmus tedy bude řešit problém hledání nejkratší cesty s omezením průchodnosti definovaný v sekci 3.9. Navržený algoritmus pro tento problém je opět implementací obecného multikriteriálního algoritmu ze sekce 4.1. Pseudokód navrženého algoritmu, skládající se především z implementování abstraktních funkcí, je zobrazen v algoritmu 3.

Vstupem algoritmu je graf dopravní sítě s nejistou průchodností hran $G = ((V, E, \epsilon), d, p)$, počáteční uzel s, cílový uzel t a horní hranice vážené neprůchodnosti b. Pokud existuje cesta splňující podmínku průchodnosti, je výstupem algoritmu cena nejkratší cesty, která splňuje horní mez vážené neprůchodnosti. Jinak je vrácen neúspěch $failure$.

Algoritmus může využít všechny datové struktury z obecného algoritmu. Abstraktní funkce jsou implementovány následovně.

- Funkce $isSuitableEstimate$ porovnává horní hranici neprůchodnosti b s odhadovanou váženou neprůchodností e_p, která je druhou složkou dané odhadované ceny $e(u) = (e_d, e_p)$. Je-li $e_p \leq b$, stav s danou odhadovanou cenou $e(u)$ může vést k nalezení optimálního řešení, a tak funkce vrací hodnotu $true$. Pokud již naopak byla hranice maximální vážené neprůchodnosti překročena ($e_p > b$), stav s danou odhadovanou cenou $e(u)$ již nemůže vést k nalezení optimálního řešení, a tak funkce vrací $false$.

Algoritmus 3: algoritmus řešící problém hledání nejkratší cesty s omezením průchodnosti

input : graph $G = ((V, E, \epsilon), d, p)$, source node s, target node t, upper bound of weighted untraversability b
output : cost of optimal path according problem definition 3.9 or $failure$
global data: all datastructures used in algorithm 1

```
1  Function isSuitableEstimate(e(u)):
2  |  (e_d, e_p) := e(u)
3  |  return e_p ≤ b
4  End Function
5
6  Function terminationCondition():
7  |  return C_solutions ≠ ∅
8  End Function
9
10 Function pruneAfterFoundSolution(c(solution)):
   |  // no prunning
11 End Function
12
13 foundSolutions := run algorithm 1 with isSuitableEstimate,
     terminationCondition and pruneAfterFoundSolution functions
14 if foundSolutions = ∅ then
15 |  return failure                    // no solution exists
16 else
17 |  return foundSolutions.getSolution()
18 end
```

- V tomto problému hledáme jediné řešení, a proto funkce *terminationCondition* vrací hodnotu *true*, je-li nalezeno alespoň jedno řešení (a tedy není-li množina $C_{solutions}$, obsahující nalezená řešení, prázdná). V opačném případě vrací hodnotu $false$.
- Vzhledem k tomu, že po nalezení prvního řešení algoritmus končí, tělo funkce $pruneAfterFoundSolution$ je prázdné.

Samotný algoritmus pak spočívá ve spuštění obecného algoritmu s argumenty G, s a t a těmito funkcemi. Neexistuje-li cesta z s do t s cenou $c = (d, p)$, která by splňovala podmínku vážené neprůchodnosti $p \leq b$, obecný algoritmus vrátí prázdnou množinu a je vrácen neúspěch $failure$. V opačném případě obecný algoritmus nalezne množinu obsahující jedinou cenu a tato cena je vrácena.

Kapitola 5
Implementace

Aplikace plánování cest je implementována v programovacím jazyku Java 14 s využitím nástroje Maven[1] pro správu použitých knihoven. Výsledky proběhlých experimentů (viz sekce 6.5) pak byly analyzovány v jazyce Python s využitím knihoven numpy[2], pandas[3] a matplotlib[4] v interaktivním prostředí Jupyter Notebooku[5]. Některé nalezené cesty pak byly vizualizovány v open source geografickém programu QGIS[6].

V této kapitole nejdříve v sekci 5.1 popíšeme, jakým způsobem jsme implementovali Pareto-množiny a poté v sekci 5.2 popíšeme výslednou aplikaci na spouštění experimentů.

5.1 Implementace Pareto-množin

Důležitou součástí multikriteriálního algoritmu je ukládání nedominovaných cen. V našem algoritmu jsou tyto ceny ukládány v množinách $C_{open}[u]$, $C_{closed}[u]$ a $C_{solutions}$, které jsou kromě udržování nalezených optimálních cen využity i pro kontrolu dominance nově nalezených cen.

Tyto Pareto-množiny mohou být implementovány více způsoby. V našem případě jsme implementovali následující 3 varianty. Jejich možný vliv na rychlost algoritmu vyhodnotíme dále v podsekci 6.5.2.

- První implementace, kterou označujeme jako **základní**, je implementována naivně pomocí standardní Java kolekce HashSet. Aby bylo možné v tomto případě zkontrolovat, jestli je daná cena dominována nějakou cenou z této množiny, je potřeba projít všechny prvky v množině.
- **Seřazená** bikriteriální implementace si ceny uchovává seřazené podle prvního a tedy i druhého kritéria. Seřazení podle druhého kritéria je totiž způsobeno vlastností bikriteriálního problému, kdy jsou-li nedominované ceny seřazeny vzestupně podle prvního kritéria, jsou rovnou seřazeny

[1] https://maven.apache.org/
[2] https://numpy.org/
[3] https://pandas.pydata.org/
[4] https://matplotlib.org/
[5] https://jupyter.org/
[6] https://www.qgis.org/en/site/

i sestupně podle druhého kritéria. Díky tomuto seřazení není nutné při kontrole dominance procházet všechny ceny v množině, ale s využitím binárního hledání stačí kontrolovat dominanci jen s některými cenami v Pareto-množině. Proto je tato implementace efektivnější.

- Třetí implementace využívá při kontrole dominance techniku **redukce dimenze** z článku [9], kterou jsme zmínili v podsekci 2.2.1. V tomto článku je dále uvedeno, kdy je možné při kontrole dominance vynechat první kritérium. Konkrétně je možné ho vynechat u množiny uzavřených uzlů a množiny nalezených řešení za podmínek, že jsou použity konzistentní heuristiky a stavy jsou z fronty stavů k expandování odebírány v lexikografickém pořadí.

  Vzhledem k tomu, že v našem případě jsou splněny oba předpoklady jejího použití, můžeme tuto techniku využít. Konkrétně, množiny $C_{open}[u]$ jsou implementovány stejně jako v seřazené implementaci a redukce dimenze je použita pro množiny $C_{closed}[u]$ a $C_{solutions}$. V případě dvou kritérií redukce dimenze znamená, že si v množině stačí uchovávat jen jedinou cenu a dominanci porovnávat pouze podle druhého kritéria.

5.2 Výsledná aplikace

Výsledná aplikace slouží ke spouštění experimentů. Vstupem aplikace je název některého z konfiguračních souborů (viz podsekce 5.2.1) uložených ve složce *resources/experiment_configs*. Podle zadaného konfiguračního souboru jsou danou konfigurací algoritmu poté řešeny jednotlivé problémy. Ceny nalezených řešení jsou pro každý problém nakonec uloženy do samostatného JSON souboru. Kromě nich výsledný soubor obsahuje také informace o problému, konfiguraci algoritmu, která daný problém řešila, a informace o průběhu jeho řešení, kterými jsou čas řešení problému, čas inicializace heuristik a počet iterací algoritmu. Všechny tyto informace následně slouží k vyhodnocení jednotlivých experimentů.

5.2.1 Konfigurace experimentů

Jak již bylo zmíněno, experimenty jsou spouštěny na základě konfiguračních souborů. Konfigurační soubor je JSON soubor obsahující tyto 3 položky.

- Položka *problemSetName* určuje, jaká sada problémů (viz sekce 6.3) má být pro daný experiment použita. Hodnota této položky je název textového souboru ze složky *resources/problem_sets* obsahující data, podle kterých jsou vytvořeny jednotlivé instance problémů. Tento soubor obsahuje řádek určující graf, jehož se daná sada problémů týká. Tento řádek je následován řádky obsahující dvojice identifikátorů uzlů grafu.
- Hodnota položky *scenarioName* je hodnota enumu *TraversabilityScenario*, která odpovídá jednomu z scénářů průchodnosti popsaných dále v sekci 6.2.

- Položka *solverAbbrName* určuje, jaká konfigurace algoritmu má být pro daný experiment použita. Její hodnota je zkratka, ve které je tato konfigurace algoritmu zakódována. Kromě obou navržených algoritmů je daný plánovač rozlišen i podle heuristik jednotlivých kritérií a podle způsobu, jakým jsou implementovány Pareto-množiny (viz sekce 5.1). Plánovače řešící úlohu nejkratší cesty s omezením průchodnosti navíc ještě rozlišujeme podle hranice minimální průměrné průchodnosti (viz podsekce 6.5.4). To, jaká konfigurace algoritmu má být použita, je ze zkratky dekódováno ve třídě *ProblemSolverParser*.

Vytvořené konfigurační soubory byly vygenerovány ve třídě *ExperimentConfigsCreator*. Sady problémů pak byly vytvořeny pomocí třídy *ProblemSetCreator*. Vytvořené soubory pak stačilo jen přesunout do odpovídající složky ve složce *resources*.

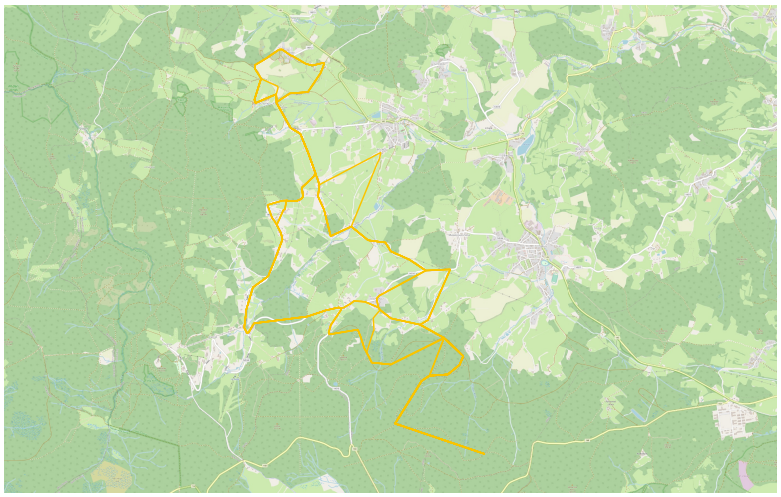
5.2.2 Vizualizování nalezených cest

Jak jsme již uvedli v úvodu této kapitoly, některé nalezené cesty jsme dále vizualizovali v geografickém programu QGIS. Abychom mohli cesty v tomto programu vizualizovat stačilo si jen nalezené cesty uložit navíc ještě v určitém formátu do CSV souboru. Pro každou cestu tak byl do vytvořeného CSV souboru přidán jeden záznam ve formátu *LineString*[7], který je tvořen ze souřadnic jednotlivých uzlů na cestě.

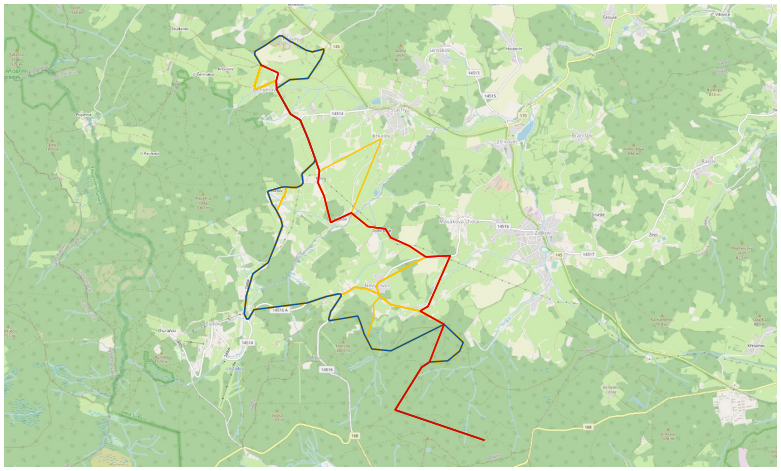
Ve výsledné aplikaci jsme kvůli velkému množství vytvářených souborů vytváření těchto souborů neponechali. Pokud by byl zájem získat z běhu experimentů kromě JSON souborů i tyto CSV soubory, stačí ve třídě *Main* přepsat hodnotu globální proměnné *SAVE_PATHS_ALSO_TO_CSV* z hodnoty *false* na *true*.

Ukázka všech nalezených cest pro jednu náhodnou instanci problému v oblasti Šumavy je zobrazena v obrázku 5.1. V obrázku 5.1a jsou stejnou barvou zobrazeny všechny cesty, které byly na daném problému nalezeny multikriteriálním algoritmem. V obrázku 5.1b jsou navíc zvýrazněny cesty s nejmenší délkou a s nejmenší váženou neprůchodností.

[7] https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry



(a) : všechny nalezené nedominované cesty (žlutá)



(b) : nejkratší cesta (červená), cesta s nejmenší váženou neprůchodností (modrá) a ostatní nedominované cesty (žlutá)

Obrázek 5.1: Vizualizace všech nalezených cest pro jednu instanci problému v oblasti Šumavy, které byly nalezeny multikriteriálním algoritmem.

Kapitola 6
Vyhodnocení

V této kapitole porovnáme a vyhodnotíme obě varianty problému a jejich řešení. Kromě toho budeme také sledovat vliv zvolení pravděpodobnosti průchodu jednotlivých hran a některých částí algoritmu jako jsou použité heuristiky nebo rozdílné implementace Pareto-množin.

Dříve než v sekci 6.5 provedeme samotné vyhodnocení jednotlivých experimentů, v následujících sekcích popíšeme parametry, které byly při experimentech použity. V sekci 6.1 objasníme, jak byly vytvořeny grafy dopravní sítě s nejistotou. V sekci 6.2 popíšeme 2 navržené scénáře, podle nichž byla jednotlivým hranám přiřazována pravděpodobnost průchodu. Poté v sekci 6.3 popíšeme sady problémů, které byly při vyhodnocení použity, a v sekci 6.4 shrneme použité parametry experimentů.

Pro vyhodnocení jsme zvolili oblast plánování pro pěší, pro které by mohlo být plánování cest s ohledem na jejich průchodnost užitečné. Konkrétním příkladem uživatelů mohou být například lidé s kočárky, kterým může nevhodný terén nebo překážka na cestě výrazně zkomplikovat cestu.

Poznamenejme však, že volba konkrétní domény nehraje pro náš algoritmus žádnou roli. Algoritmus by tedy mohl být použit i třeba při plánování pro cyklisty, kteří mohou mít během cesty (především v přírodě) podobné problémy. Jediný rozdíl by byl v použitém grafu dopravní sítě s nejistotou, kde by mohly být zahrnuty i silnice s frekventovanější dopravou a naopak by v něm nebyly zahrnuty například chodníky.

6.1 Tvorba grafu dopravní sítě s nejistotou

Pro experimenty jsme si vybrali 2 výrazně odlišné oblasti. První zvolenou oblastí je oblast Prahy, která je složena z husté sítě cest s vysokou průchodností. Druhou je oblast Národního parku Šumava a jeho blízkého okolí, která obsahuje mnoho cest s nižší průchodností.

Porovnání vytvořených grafů ukazuje tabulka 6.1, kde můžeme vidět, že graf Prahy je z hlediska počtu uzlů i hran násobně větší než graf Šumavy. Zajímavým údajem je také počet hran na 1 km^2 [1], kde se ukazuje, že síť cest

[1] Hodnoty rozlohy (a tedy i počtu hran na 1 km^2) jsou pouze orientační. Byly získány v programu QGIS jako obsah plochy mnohoúhelníku ohraničujícího všechny uzly grafu. Pro tento účel jsou však dostačující.

v oblasti Prahy je oproti síti Šumavy více než 20krát hustší.

parametr grafu	graf Prahy	graf Šumavy
počet uzlů	118559	24934
počet hran	321160	60324
přibližná rozloha v km^2	507.42	2070.48
počet hran na 1 km^2	632.93	29.14

Tabulka 6.1: Parametry použitých grafů.

K vytvoření grafu dopravní sítě s nejistotou jsme využili mapové podklady OpenStreetMap. V podsekci 6.1.1 přiblížíme, jak jsou tyto mapové podklady organizovány. Podsekce 6.1.2 obsahuje bližší informace ohledně toho, kde jsme tyto data získali a jakým způsobem jsme z nich vytvořili graf. Otázky, které informace jsme zvolili pro určení pravděpodobnosti průchodu jednotlivých hran a jak je z nich tato pravděpodobnost určena, dále objasní podsekce 6.1.3.

6.1.1 OpenStreetMap

OpenStreetMap (OSM) [2] je otevřený projekt poskytující mapové podklady celého světa. Tyto mapové podklady jsou tvořeny především třemi základními prvky [19]. Základním prvkem je:

- *uzel (node)*, který označuje určitý bod na Zemi,
- *cesta (way)*, jež je tvořena uspořádaným seznamem uzlů a označuje určitou cestu (např. silnici, chodník) nebo ohraničuje plochu (např. budovu či louku),
- *relace (relation)*, která označuje vztahy mezi více prvky (např. restrikce typu zákaz odbočení nebo označení tramvajové linky).

Velmi důležitou součástí těchto mapových podkladů je i tzv. *značka (tag)* složená z dvojice klíč a hodnota, která může být připojena k základnímu prvku. Ke každému základnímu prvku jich může být přidáno hned několik. Hlavním účelem značek je co nejvíce popsat vlastnosti příslušného prvku.

6.1.2 Parsování OSM dat na graf

Pokud se týká získání OSM dat, existuje několik nástrojů, pomocí nichž lze získat data určité oblasti. Tyto nástroje se liší formáty, ve kterých lze data získat, a maximální velikostí dat, které je možné stáhnout. V našem případě jsme k získání OSM dat Šumavy využili stahovací server *BBBike's free server*[3], který umožnuje stahovat větší OSM data v několika různých formátech. Navíc je možné si zde zvolit vlastní oblast. OSM data Prahy jsme

[2] https://www.openstreetmap.org/
[3] https://download.bbbike.org/osm/

získali stažením z webové stránky[4], která umožnuje stažení OSM dat některé z konkrétních oblastí. V obou případech jsme OSM data získali ve formátu PBF[5].

Získaná OSM data jsme následně zpracovali v aplikaci *road-graph-tool*, která je vyvíjena výzkumným týmem Smart Mobility[6] z Centra umělé inteligence FEL ČVUT[7]. Jedná se o nástroj, který umožnuje zpracování a vyfiltrování vhodných dat z OpenStreetMap. Aplikace je zatím ve fázi vývoje, a tak není veřejná. Pro naše účely zpracování OSM dat a tvorby grafu ji již však bylo možné použít.

V nástroji road-graph-tool jsme nejprve vyfiltrovali vhodné OSM cesty. Pro oba grafy jsme z dat tedy vynechali dálnice, rychlostní silnice a silnice 1. třídy, které nejsou vhodné pro nemotorovou dopravu. Protože vyhodnocení plánujeme udělat na grafu pro pěší, zakázali jsme navíc i silnice 2. třídy, které jsou pro pěší příliš rušné, a cesty, kam je pěším zakázán vstup. V grafu Prahy jsme navíc ještě vynechali silnice 3. třídy, neboť se u nich téměř vždy nachází chodník.

Po vyfiltrování vhodných tagů byly OSM cesty převedeny na hrany, čímž už vnikl graf. Z tohoto grafu jsme vybrali pouze jeho největší silně souvislou komponentu, díky níž budeme mít zaručeno, že mezi každými dvěma uzly grafu bude existovat cesta. Vzniklý graf jsme spolu s užitečnými informacemi uložili do CSV souborů. Naše aplikace pak tyto soubory načítá a na základě uložených informacích z nich vytváří graf dopravní sítě s nejistou průchodností hran.

6.1.3 Určení pravděpodobnosti průchodu jednotlivých hran

Pro určení pravděpodobnosti průchodu jednotlivých hran jsme použili informace získané z OpenStreetMap. Konkrétně jsme využili značky s klíči *highway*, *surface* a *tracktype*.

klíč	graf Prahy	graf Šumavy
highway	99	100
surface	33	22
tracktype	3	38

Tabulka 6.2: Relativní četnost (v %) klíčů zvolených OSM značek vůči celkovému počtu hran jednotlivých grafů. Hodnoty jsou zaokrouhleny na celá procenta.

Značka s **klíčem highway**[8] rozlišuje různé druhy silnic a cest. Podle její hodnoty řadí cestu do jedné z mnoha kategorií jako například silnice 3. třídy, chodník, lesní cesta nebo schody. Tato značka sice dává jistou informaci o typu cesty, nikoliv však už o její kvalitě. Důvodem, proč jsme se rozhodli

[4] http://download.openstreetmap.fr/extracts/europe/czech_republic/
[5] https://wiki.openstreetmap.org/wiki/PBF_Format
[6] https://www.aic.fel.cvut.cz/research-areas/smart-mobility
[7] https://www.aic.fel.cvut.cz/
[8] https://wiki.openstreetmap.org/wiki/Cs:Key:highway

využít tuto značku, je, že se vyskytuje téměř u každé hrany výsledných grafů. To potvrzuje i tabulka 6.2 ukazující, jaký je poměr počtu hran s daným klíčem vůči celkovému počtu hran jednotlivých grafů.

Informaci o povrchu jednotlivých cest poskytuje značka s **klíčem surface**[9]. Její hodnota dává o cestě informaci, jestli je cesta zpevněná či nikoliv. V mnoha případech je dokonce přímo uvedena informace o jejím materiálu (tj. jestli je cesta například asfaltová, štěrková nebo hliněná).

Vzhledem k tomu, že polních nebo lesních cest existuje mnoho a jejich kvalita se výrazně liší, může být k cestě přidána značka s **klíčem tracktype**[10], která rozlišuje tyto cesty do 5 typů podle jejich kvality. Cesty jsou rozlišeny od cest s pevnými povrchy až po nejméně kvalitní cesty s měkkým nezpevněným povrchem.

S ohledem na informace poskytnuté těmito značkami jsme většině možným hodnotám značek s těmito klíči přiřadili pravděpodobnost průchodu. Zvolené hodnoty pro jednotlivé scénáře průchodnosti (viz sekce 6.2) zobrazují tabulky pro příslušné OSM značky 6.3, 6.4 a 6.5. Případný vliv zvolení konkrétních hodnot na rychlost běhu algoritmu dále vyhodnocujeme v podsekci 6.5.3.

hodnota značky highway	**scénář „sucho“**	**scénář „mokro“**
tertiary	1.00	1.00
tertiary-link	1.00	1.00
residential	1.00	1.00
living_street	1.00	1.00
service	1.00	1.00
pedestrian	1.00	1.00
footway	1.00	0.99
sidewalk	1.00	1.00
crossing	1.00	1.00
cycleway	1.00	0.99
unclassified	0.99	0.95
road	0.99	0.95
corridor	0.99	0.99
path	0.95	0.80
track	0.90	0.70
bridleway	0.90	0.70
steps	0.70	0.60

Tabulka 6.3: Zvolené hodnoty pravděpodobnosti průchodu pro jednotlivé scénáře průchodnosti podle OSM značky s klíčem highway.

[9] https://wiki.openstreetmap.org/wiki/Cs:Key:surface
[10] https://wiki.openstreetmap.org/wiki/Cs:Key:tracktype

hodnota značky surface	scénář „sucho“	scénář „mokro“
paved	1.00	0.99
asphalt	1.00	1.00
concrete	1.00	0.99
paving_stones	1.00	0.99
metal	1.00	1.00
wood	1.00	1.00
concrete:lanes	0.99	0.95
concrete:plates	0.99	0.95
sett	0.99	0.95
unhewn_cobblestone	0.99	0.95
cobblestone	0.99	0.95
compacted	0.95	0.80
fine_gravel	0.95	0.80
pebblestone	0.95	0.80
grass_paver	0.95	0.80
unpaved	0.90	0.70
gravel	0.80	0.70
ground	0.80	0.70
grass	0.70	0.70
dirt	0.60	0.40
earth	0.60	0.40
sand	0.60	0.40
mud	0.40	0.30
rock	0.20	0.20

Tabulka 6.4: Zvolené hodnoty pravděpodobnosti průchodu pro jednotlivé scénáře průchodnosti podle OSM značky s klíčem surface.

hodnota značky tracktype	scénář „sucho“	scénář „mokro“
grade1	1.00	0.90
grade2	0.95	0.70
grade3	0.80	0.50
grade4	0.60	0.40
grade5	0.40	0.20

Tabulka 6.5: Zvolené hodnoty pravděpodobnosti průchodu pro jednotlivé scénáře průchodnosti podle OSM značky s klíčem tracktype.

Výslednou pravděpodobnost průchodu hrany určujeme sekvenčním způsobem, kdy postupně porovnáváme značky příslušné hrany. Podle toho, jakou informaci o kvalitě cesty jednotlivé značky přináší, jsme se rozhodli značky porovnávat v pořadí *tracktype*, *surface* a *highway*.

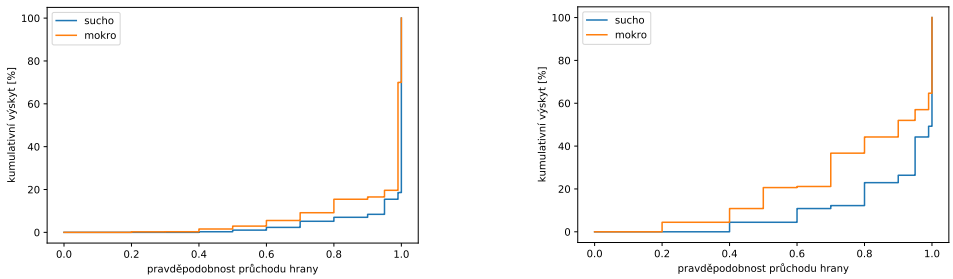
Pravděpodobnost průchodu určité hrany je tedy určena následovně. Je-li hrana označená značkou s klíčem *tracktype*, přiřadíme této hraně pravděpodobnost, která odpovídá hodnotě této značky. Jinak je analogickým způsobem přiřazena pravděpodobnost průchodu hran podle značky *surface*, případně podle značky *highway*. Není-li u hrany uvedena ani jedna z těchto 3 sledovaných značek, nemáme o této hraně žádnou informaci a pro pravděpodobnost průchodu hrany je zvolena hodnota 0.5.

6.2 Scénáře průchodnosti

Abychom mohli vyhodnotit, jaký vliv má na nalezené řešení zvolení pravděpodobnosti průchodu, vytvořili jsme 2 pravděpodobnostní scénáře simulující průchodnost cest za určitých okolností. Tyto scénáře a konkrétní hodnoty zvolených pravděpodobností průchodu jsou zobrazeny v tabulkách pro jednotlivé OSM značky 6.3, 6.4 a 6.5.

Scénář „sucho“ simuluje průchodnost cest (např. pro lidi s kočárkem) za příznivé situace, kdy několik dní v řadě bylo vhodné počasí. Pevným a hladkým cestám přiřazuje vysokou pravděpodobnost blízko 1. Nižší pravděpodobnost kolem 0.7 mají například schody, nezpevněné hlinité cesty nebo cesty s horším stupněm kvality, které na delším úseku cesty mohou odradit. Nejnižší průchodnost je přiřazena bahnitým či skalnatým cestám a také cestám s nejhorším stupněm kvality.

Scénář „mokro“ simuluje průchodnost cest (např. pro lidi s kočárkem) v nepříznivé situaci, například když v poslední době intenzivně pršelo. V tomto případě je průchodnost pevných silnic o trochu nižší než při scénáři „sucho“. Nicméně jejich průchodnost zůstává vysoká. Odlišnější je průchodnost nezpevněných cest, které se snadno mohou stát bahnitými, a tudíž je jejich průchodnost snížena výrazněji.



(a) : graf Prahy

(b) : graf Šumavy

Obrázek 6.1: Kumulativní výskyt hran s danou pravděpodobností průchodu v jednotlivých scénářích průchodnosti a grafech.

Porovnání obou vytvořených scénářů z hlediska toho, v jakém poměru se v jednotlivých grafech vyskytují hrany s danou pravděpodobností průchodu, ukazuje obrázek 6.1. Z něj vyplývá, že v grafu Prahy (obr. 6.1a) se vyskytuje menší poměr hran s nižší pravděpodobostí než v grafu Šumavy (obr. 6.1b). Například ve scénáři „mokro“ se v grafu Prahy nachází pouze přibližně 20 %

hran s pravděpodobností průchodu nejvýše 0.8, zatímco v grafu Šumavy je to více než 40 % hran. Dále se ukazuje, že pro oba grafy má scénář „mokro“ vyšší poměr hran s nižší pravděpodobností než scénář „sucho“.

6.3 Sady problémů

Pro oba grafy jsme vytvořili několik sad problémů rozlišených podle délky nejkratší cesty mezi počátečním a cílovým uzlem nalezenou pomocí A* algoritmu.[11] Každá sada problémů odpovídá určitému grafu a je vždy tvořená 100 páry identifikátorů náhodných uzlů tohoto grafu, tedy počátečního a cílového uzlu.

Konkrétně pro graf Šumavy jsme vytvořili 6 sad problémů, které jsme rozlišili podle délky nejkratší cesty do intervalů po 10 km (tj. 0–10 km, 10–20 km, 20–30 km, 30–40 km, 40–50 km, 50+ km). Pro graf Prahy jsme vytvořili 5 sad problémů rozlišených po 5 km (tj. 0–5 km, 5–10 km, 10–15 km, 15–20 km, 20+ km).

Při plánování pro pěší jsou některé z těchto intervalů možná příliš dlouhé. Nicméně jak již bylo zmíněno v úvodu kapitoly, porovnávané algoritmy lze použít například i při plánování pro cyklisty, kde jsou již tyto délky normální, a proto algoritmy vyhodnocujeme i na těchto větších instancích problému.

6.4 Konfigurace experimentů

Pro vyhodnocení jsme spouštěli několik experimentů lišících se v jednotlivých parametrech. Každý experiment je složen z grafu a scénáře průchodnosti, z nichž je sestrojen graf dopravní sítě s nejistou průchodností, sady problémů odpovídajícího grafu a konkrétní konfigurace algoritmu, který daný problém řeší. Kromě obou navržených algoritmů, u algoritmů rozlišujeme ještě heuristiky obou kritérií a konkrétní implementaci Pareto-množin.

parametr experimentu	**možné hodnoty**
graf	{Praha, Šumava}
scénář průchodnosti	{sucho, mokro}
algoritmus	{multikriteriální, s omezením}
heuristika kritéria vzdálenosti	{nulová, přímá vzdálenost, předpočítaná}
heuristika krit. průchodnosti	{nulová, předpočítaná}
implementace Pareto-množin	{základní, seřazená, redukce dimenze}

Tabulka 6.6: Uvažované hodnoty jednotlivých parametrů experimentu.

[11] Tuto metriku vzdálenosti jsme upřednostnili před přímou vzdáleností mezi uzly (tj. vzdáleností vzdušnou čarou) z důvodu, že v některých úsecích mezi počátečním a cílovým uzlem neexistuje žádná přímá cesta a délka nejkratší cesty mezi těmito uzly se od přímé vzdálenosti mezi nimi výrazně liší.

Všechny možné hodnoty jednotlivých parametrů experimentu jsou shrnuty v tabulce 6.6. Pro zvolený graf pak byly vždy řešeny všechny vytvořené sady problémů daného grafu, a proto jsou pro přehlednost z tabulky vynechány.

Kvůli velkému množství možných kombinací parametrů byly pro vyhodnocení vlivu jednoho z parametrů vybrány jen některé kombinace těchto hodnot. Konkrétní hodnoty parametrů, které byly pro dané vyhodnocení použity jsou pak shrnuty v jednotlivých podsekcích.

6.5 Vyhodnocení experimentů

V této sekci popíšeme a vyhodnotíme spouštěné experimenty.

Experimenty byly spouštěny na některém z CPU uzlů výpočetního clusteru RCI CTU[12] s těmito hardwarovými specifikacemi: 24 jader/48 vláken 3.2GHz (2 x Intel Xeon Scalable Gold 6146), 384GB RAM. Pro určitý výpočet byl použit vždy jeden výpočetní uzel, kterému byla přiřazena 1 úloha používající 4 jádra s pamětí omezenou na 10 GB. Ačkoliv naše aplikace není paralelní, pro běh experimentů bylo alokováno více jader. Důvodem pro to je, že experimenty byly spouštěny v Javě, a tak jsme chtěli, aby naměřené výsledky experimentů (především doba běhu) byly co nejméně ovlivněné během garbage collectoru.

Tato sekce je organizována následujícím způsobem. Nejprve v podsekci 6.5.1 vyhodnotíme vliv heuristik jednotlivých kritérií a v podsekci 6.5.2 porovnáme rozdílné implementace Pareto-množin. Vliv zvolení pravděpodobností průchodu budeme sledovat v podsekci 6.5.3, kde porovnáme naměřené hodnoty v obou navržených scénářích průchodnosti. Na závěr v podsekci 6.5.4 porovnáme oba navržené algoritmy z kapitoly 4.

6.5.1 Vliv heuristik

Jak již bylo zmíněno, využití heuristik odhadujících cenu cesty do cílového uzlu může vést k zrychlení algoritmu. V této podsekci se zabýváme tím, jak jednotlivé kombinace heuristických funkcí navržených v podsekci 4.1.3, ovlivní běh algoritmu.

parametr experimentu	**hodnoty**
graf	{Praha, Šumava}
scénář průchodnosti	{sucho}
algoritmus	{multikriteriální}
heuristika kritéria vzdálenosti	{nulová, přímá vzdálenost, předpočítaná}
heuristika krit. průchodnosti	{nulová, předpočítaná}
implementace Pareto-množin	{redukce dimenze}

Tabulka 6.7: Hodnoty jednotlivých parametrů experimentu použité při vyhodnocování vlivu heuristik.

[12] https://login.rci.cvut.cz/wiki/start

heuristika vzdálenosti	**heuristika průchodnosti**	
	nulová (z)	předpočítaná (pc-p)
nulová (z)	(z, z)	(z, pc-p)
přímá vzdálenost (d)	(d, z)	(d, pc-p)
předpočítaná (pc-d)	(pc-d, z)	(pc-d, pc-p)

Tabulka 6.8: Kombinace porovnávaných heuristik s jejich zkratkami.

Hodnoty jednotlivých parametrů experimentu, které byly použity při porovnávání heuristik jsou zobrazeny v tabulce 6.7. Vzhledem k tomu, že naměřené hodnoty na obou grafech ukazují podobný trend, v této podsekci popíšeme jen naměřené hodnoty na grafu Šumavy. Naměřené hodnoty na grafu Prahy jsou zobrazeny v příloze A.1.

Pokud se týká porovnávaných heuristik, porovnáváno bylo všech 6 jejich kombinací. Tyto kombinace spolu se zkratkami, pomocí nichž se na ně budeme v této podsekci odkazovat, jsou zobrazeny v tabulce 6.8.

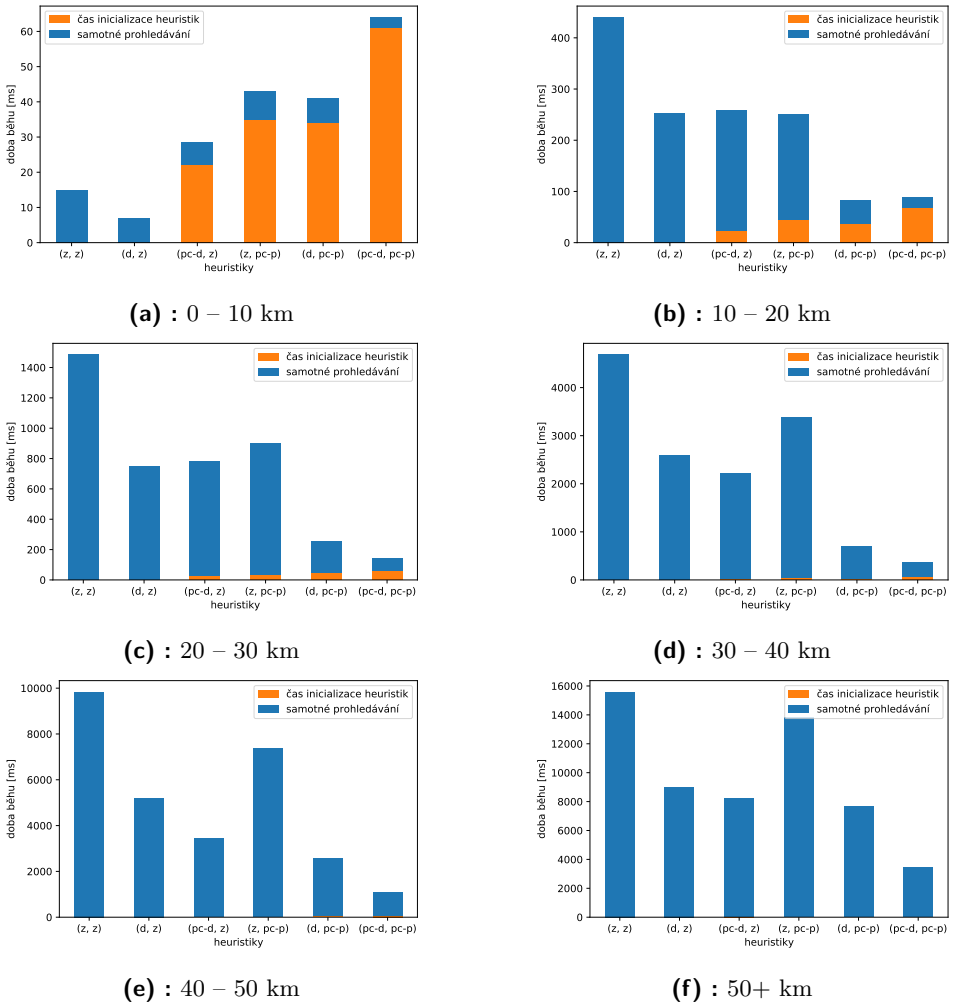
Předpočítaná heuristika, která využívá ideální odhad ceny, vyžaduje počáteční inicializaci, jež stojí jistý čas z doby prohledávání. Na druhou stranu nulová heuristika a heuristika přímé vzdálenosti, které tak přesný odhad nemají, se inicializovat nemusí. Proto jsme se rozhodli sledovat, jestli se vyplatí strávit čas touto inicializací.

Grafy, ve kterých je porovnáván medián[13] doby běhu algoritmu s časem inicializace heuristik, jsou zobrazeny v obrázku 6.2. Naměřené hodnoty ukazují, že pro sadu problémů 0–10 km (obr. 6.2a) se inicializace nevyplatí, neboť trvá déle než je doba běhu algoritmu s nulovými heuristikami. Medián času inicializace alespoň jedné z předpočítaných heuristik je totiž větší než 20 ms, zatímco medián doby běhu algoritmu s nulovými heuristikami je jen 15 ms. Nicméně již od sady problémů s délkou nejkratší cesty nad 10 km je celkový čas běhu algoritmu s předpočítanými heuristikami nižší než bez nich. Zajímavým údajem je, že ačkoliv v sadě problémů 10–20 km trvá předpočítání obou heuristik (sloupec (pc-d, pc-p)) většinu (konkrétně 77 %) času běhu algoritmu, celková doba běhu algoritmu patří k nejnižším.

Pro sady problémů s délkou nejkratší cesty větší než 20 km (obr. 6.2c – 6.2f), již doba inicializace heuristik zabírá menší část doby běhu algoritmu (např. na sadě problémů 50+ km u algoritmu s oběma předpočítanými heuristikami (sloupec (pc-d, pc-p)) zabírá inicializace heuristik méně než 2 % celkové doby běhu algoritmu). Dále se ukazuje, že algoritmy, které mají pouze heuristiku délky (sloupce (d, z) a (pc-d, z)), doběhnou pro větší instance problémů rychleji než algoritmus, který využívá pouze heuristiku pro kritérium průchodnosti (sloupec (z, pc-p)).

Protože spolu s časem ovlivňují heuristiky také počet iterací algoritmu, sledujeme i tento ukazatel. Navíc oproti času je počet iterací algoritmu nezávislý na prostředí, kde byly problémy řešeny.

[13] Medián jsme se rozhodli upřednostnit před průměrem kvůli tomu, že není ovlivněn vychýlenými hodnotami, které by v případě průměru zkreslily naměřené údaje.



(a) : 0 – 10 km

(b) : 10 – 20 km

(c) : 20 – 30 km

(d) : 30 – 40 km

(e) : 40 – 50 km

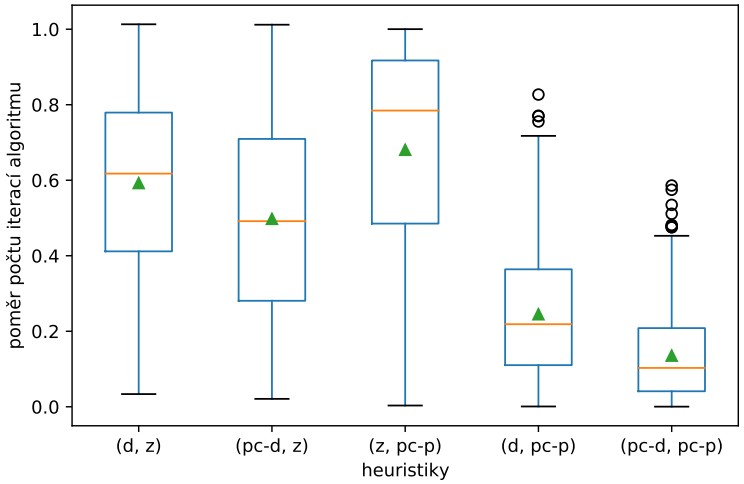
(f) : 50+ km

Obrázek 6.2: Medián času inicializace heuristik a celkové doby běhu algoritmu pro jednotlivé kombinace heuristik na sadách problémů grafu Šumavy.

Poměry počtu iterací algoritmu s jednotlivými heuristikami vůči algoritmu s nulovými heuristikami pro všechny odpovídající si problémy ze všech 6 sad problémů grafu Šumavy jsou zobrazeny tzv. boxploty[14] v obrázku 6.3 Naměřené údaje ukazují, že algoritmus s oběma předpočítanými heuristikami (sloupec (pc-d, pc-p)) ze srovnání vychází nejlépe, když ve většině problémů má méně než pětinu iterací oproti algoritmu s nulovými heuristikami. Dále se potvrzuje, že algoritmy s heuristikami pro obě kritéria (sloupce (d, pc-p) a (pc-d, pc-p)) mají méně iterací než algoritmy, které mají pouze jednu informovanou heuristiku (např. na sadě problémů 0–10 km se počet iterací v algoritmech s oběma heuristikami oproti počtu iterací v algoritmech vyu-

[14]Boxplot je typ grafu, který nabízí větší vhled do rozložení dat. Zobrazeny jsou medián (oranžová čára), průměrná hodnota (zelený trojúhelník), hodnoty mezi kvartily $Q_{0.25}$ a $Q_{0.75}$ (ohraničené obdélníkem), tzv. vousy ukazující rozsah nevychýlených hodnot mezi $Q_{0.25} - 1.5$ IQR a $Q_{0.75} + 1.5$ IQR, kde IQR je mezikvartilové rozpětí $Q_{0.75} - Q_{0.25}$, a vychýlené hodnoty (černé kružnice).

žívajících pouze jednu informovanou heuristiku liší řádově o tisíce, na sadě problémů 50+ km jsou to již dokonce miliony iterací). V neposlední řadě naměřené údaje ukazují, že algoritmy, které mají pouze heuristiku délky (sloupce (d, z) a (pc-d, z)), mají méně iterací než algoritmus, který má pouze heuristiku pro kritérium průchodnosti (sloupec (z, pc-p)).



Obrázek 6.3: Poměr počtu iterací algoritmu s jednotlivými kombinacemi heuristik vůči algoritmu s nulovými heuristikami na grafu Šumavy.

6.5.2 Vliv implementace Pareto-množin

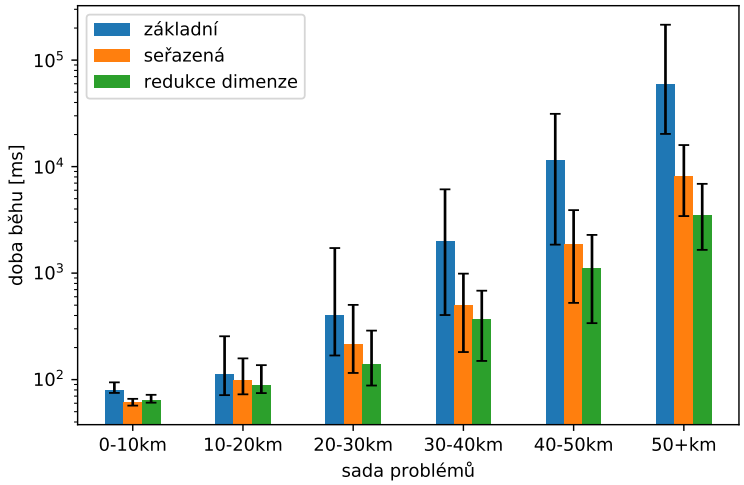
Jak ukládání nedominovaných cen, tak i kontrolování dominance nových cen probíhá během algoritmu velmi často. Proto i implementace Pareto-množin může ovlivnit výslednou dobu běhu algoritmu. V této podsekci porovnáváme 3 rozdílné implementace (základní, seřazená a redukci dimenze), jež byly představeny v sekci 5.1.

parametr experimentu	**hodnoty**
graf	{Praha, Šumava}
scénář průchodnosti	{sucho}
algoritmus	{multikriteriální}
heuristika krit. vzdálenosti	{předpočítaná}
heuristika krit. průchodnosti	{předpočítaná}
implementace Pareto-množin	{základní, seřazená, redukce dimenze}

Tabulka 6.9: Hodnoty jednotlivých parametrů experimentu použité při vyhodnocování vlivu implementace Pareto-množin.

Hodnoty jednotlivých parametrů experimentu, které byly použity při porovnávání jednotlivých implementací Pareto-množin jsou zobrazeny v tabulce 6.9. V této podsekci popíšeme opět jen naměřené údaje na grafu Šumavy. Obrázky

zobrazující údaje získané na grafu Prahy jsou zobrazeny v příloze A.2. Bohužel celková doba běhu algoritmu na některých menších instancích problémů na grafu Prahy byla pravděpodobně ovlivněna nějakým faktorem (např. během garbage collectoru nebo spíše větším vytížením výpočetního uzlu), protože čas inicializace heuristik se občas výrazně liší (řádově o stovky ms). Přitom vzhledem k tomu, že ve všech experimentech byla použita stejná kombinace heuristik, čas inicializace heuristik by měl být pro stejné sady problémů velmi podobný. Pokud bychom odhlédli od těchto výkyvů na menších instancích problémů, lze i z dat na grafu Prahy dojít k podobnému závěru jako z dat na grafu Šumavy.

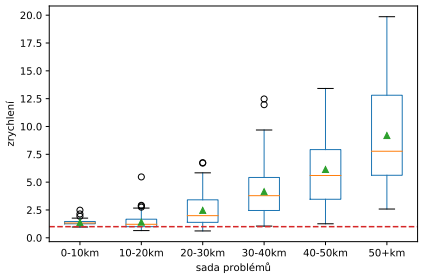


Obrázek 6.4: Porovnání mediánu doby běhu (v logaritmickém měřítku) algoritmu s jednotlivými implementacemi Pareto-množin pro jednotlivé sady problémů grafu Šumavy. Černě je vyznačeno rozpětí hodnot mezi kvartily $Q_{0.25}$ a $Q_{0.75}$.

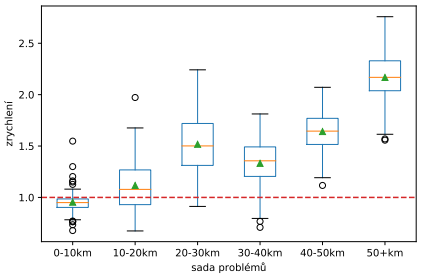
Porovnání doby běhu algoritmu s jednotlivými implementacemi Pareto-množin pro jednotlivé sady problémů je zobrazeno v obrázku 6.4. Poznamenejme, že kvůli zpřehlednění je doba běhu zobrazena v logaritmickém měřítku. Naměřené hodnoty jsou očekávané. To znamená, že prohledávání se základní implementací trvá déle než se seřazenou implementací a že využití redukce dimenze dobu běhu algoritmu ještě sníží.

Zajímavějším údajem je, jak jednotlivé implementace zrychlí běh algoritmu. Porovnání zrychlení běhu algoritmu (tj. poměr doby běhu algoritmu na odpovídajících si instancích problému) pro jednotlivé sady problémů ukazují boxploty v obrázku 6.5. Graf (obr. 6.5a), zobrazující poměr doby trvání mezi základní a seřazenou implementací, ukazuje, že samotné řazení cen v Pareto-množině pro větší instance problémů dobu běhu algoritmu několikanásobně sníží (na sadě problémů 20–30 km přibližně dvojnásobně, na sadě problémů 50+ km je zrychlení u většiny problémů více než sedminásobné). Porovnání poměru doby běhu mezi seřazenou implementací a implementací s redukcí dimenze (obr. 6.5b) potvrzuje, že použití techniky redukce dimenze běh algoritmu ještě zrychlí (např. pro většinu problémů sady problémů 50+ km

více než dvojnásobně, což na této sadě problémů znamená rozdíl v době běhu přibližně 5 s).



(a) : seřazená vs. základní (červená čárkovaná čára)



(b) : redukce dimenze vs. seřazená (červená čárkovaná čára)

Obrázek 6.5: Porovnání zrychlení (tj. poměr času trvání) běhu algoritmu mezi vybranými dvojicemi implementací Pareto-množin pro jednotlivé sady problémů grafu Šumavy.

6.5.3 Vliv zvolení pravděpodobnosti průchodu jednotlivých hran

V sekci 6.2 jsme na základě informací z OpenStreetMap navrhli 2 scénáře průchodnosti simulující průchodnost cest za určitých situací. Cílem této podsekce je tyto 2 scénáře porovnat a zjistit, jestli má zvolení hodnot pravděpodobnosti průchodu vliv na rychlost algoritmu.

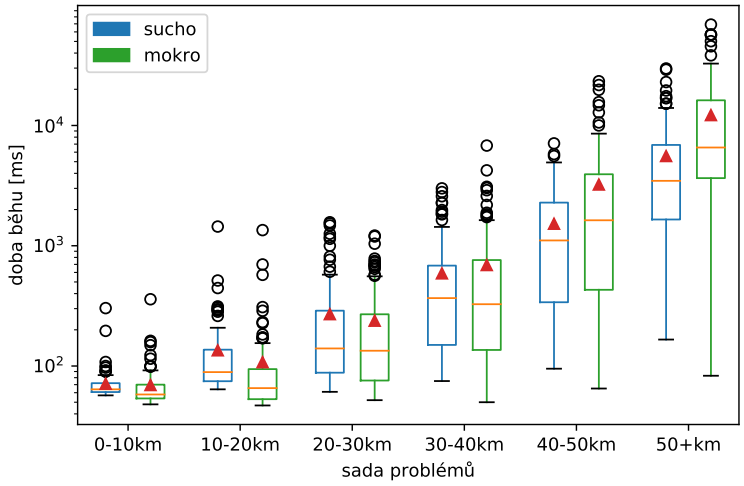
parametr experimentu	**hodnoty**
graf	{Praha, Šumava}
scénář průchodnosti	{sucho, mokro}
algoritmus	{multikriteriální}
heuristika kritéria vzdálenosti	{předpočítaná}
heuristika kritéria průchodnosti	{předpočítaná}
implementace Pareto-množin	{redukce dimenze}

Tabulka 6.10: Hodnoty jednotlivých parametrů experimentu použitých při vyhodnocování vlivu zvolení pravděpodobnosti průchodu jednotlivých hran.

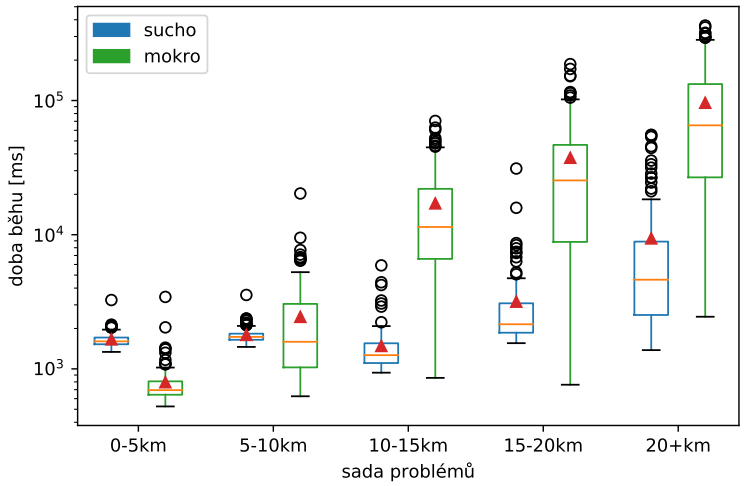
Hodnoty jednotlivých parametrů experimentu, které byly použity při porovnání obou scénářů jsou zobrazeny v tabulce 6.10.

Porovnání doby běhu algoritmu v obou scénářích průchodnosti zobrazené boxploty nabízí obrázek 6.6. Obrázek ukazuje naměřená data pro jednotlivé sady problémů grafu Šumavy (obr. 6.6a) i grafu Prahy (obr. 6.6b). Doba běhu algoritmu je pro přehlednost opět zobrazena v logaritmickém měřítku.

Při porovnání obou scénářů průchodnosti na grafu Šumavy (obr. 6.6a), je vidět, že doba běhu algoritmu se mezi jednotlivými scénáři (až na největší sady problémů) nějak výrazně nemění. Naměřené údaje ukazují, že pro sady problémů 0–10 km a 10–20 km je ve většině problémů doba běhu algoritmu



(a) : graf Šumavy



(b) : graf Prahy

Obrázek 6.6: Porovnání doby běhu algoritmu mezi oběma scénáři průchodnosti na jednotlivých sadách problémů grafu Šumavy i Prahy.

ve scénáři „sucho“ nepatrně (cca o 10 ms) delší. Pro sady problémů 40–50 km a 50+ km je naopak doba běhu delší ve scénáři „mokro“. Pro tyto větší instance problémů je již rozdíl mezi scénáři významnější (např. v sadě problémů 50+ km je medián doby běhu ve scénáři „sucho“ přibližně 3.5 s, zatímco ve scénáři „mokro“ je medián 6.5 s).

Výraznější rozdíly v době běhu algoritmu můžeme vidět na grafu Prahy (obr. 6.6b). Na sadě problémů 0–5 km je ve scénáři „sucho“ doba běhu algoritmu téměř ve všech problémech delší než ve scénáři „mokro“. Pro sady problémů s délkou nejkratší cesty nad 10 km je to výrazně naopak, kdy doba běhu algoritmu ve scénáři „mokro“ je ve většině problémů desetinásobně delší. Konkrétně, porovnáme-li mediány doby běhu algoritmu pro sadu problémů

20+ km, pro scénář „sucho“ je medián přibližně 4.5 s, zatímco pro scénář „mokro“ je medián 65 s.

Z porovnání obou grafů vyplynulo, že pro nejmenší instance problémů je ve většině případu o trochu rychlejší scénář „mokro“. Tento výkyv je způsobem dobou inicializace předpočítaných heuristik, které je pro scénář „mokro“ o trochu rychlejší (na grafu Šumavy o desítky ms, na grafu Prahy o stovky ms) než u scénáře „sucho“.

Mnohem významnější je rozdíl v době běhu algoritmu na větších instancích problémů, kde byl v obou grafech rychlejší scénář „sucho“. To je nejspíš způsobeno počtem optimálních řešení v jednotlivých úlohách, protože pro většinu problémů bylo ve scénáři „mokro“ nalezeno více řešení než ve scénáři „sucho“.

Především na větších instancích problémů je vidět, že i zvolení konkrétních hodnot pravděpodobností průchodu má nezanedbatelný vliv na dobu běhu algoritmu.

6.5.4 Porovnání multikriteriálního algoritmu s algoritmem řešící úlohu s omezením průchodnosti

V kapitole 4 jsme navrhli dva algoritmy. Prvním je multikriteriální algoritmus popsaný v sekci 4.2, který nalezne množinu všech Pareto-optimálních cest. Druhý algoritmus, jenž byl popsán v sekci 4.3, řeší jednodušší úlohu, kterou je nalezení nejkratší cesty tak, že je splněna podmínka maximální vážené neprůchodnosti cesty. V této podsekci tyto dva algoritmy porovnáme.

Vzhledem k tomu, že algoritmus s omezením nalezne pouze jediné řešení z množiny řešení nalezených multikriteriálním algoritmem, je zřejmé, že pro nalezení tohoto řešení stačí algoritmu s omezením méně iterací než by potřeboval multriteriální algoritmus. Což by mělo vést k tomu, že algoritmus s omezením bude rychlejší. Jelikož má algoritmus s omezením oproti multikriterálnímu algoritmu jeden argument navíc, je zajímavé také sledovat, jestli má zvolení hranice vážené neprůchodnosti vliv na rychlost nalezeného řešení.

Bohužel hranice maximální vážené neprůchodnosti, která je použita v algoritmu, je hůře interpretovatelná a bylo by obtížné jí rozumně zvolit pro jednotlivé instance problémů. Proto jsme se rozhodli porovnávané varianty algoritmu s omezením rozlišit podle hranice minimální průměrné průchodnosti (vzorec 3.2), ze které hranici maximální vážené neprůchodnosti určíme individuálně pro jednotlivé instance problémů. Hranice maximální vážené neprůchodnosti b je pak pro konkrétní instanci problému určena následovně. Daná hranice průměrné průchodnosti cesty $c_{pp}(\pi)$ je převedena na průměrnou neprůchodnost cesty $c_{pn}(\pi) = 1 - c_{pp}$, která je poté vynásobena délkou nejkratší cesty, jež je nalezena pomocí A* algoritmu. Tím je získaná potřebná hranice pro maximální váženou neprůchodnost b.

Například chceme-li cestu s minimální průměrnou průchodností 0.9 a víme-li, že délka nejkratší cesty mezi danými uzly je 1000 m, hranice maximální vážené neprůchodnosti pro úlohu s omezením bude $b = (1 - 0.9) \cdot 1000 = 100$.

parametr experimentu	hodnoty
graf	{Praha, Šumava}
scénář průchodnosti	{sucho}
algoritmus	{multikriteriální, s omezením}
heuristika kritéria vzdálenosti	{předpočítaná}
heuristika kritéria průchodnosti	{předpočítaná}
implementace Pareto-množin	{redukce dimenze}
maximální průměrná průchodnost	{0.75, 0.85, 0.9, 0.92, 0.95, 0.97, 0.99}

Tabulka 6.11: Hodnoty jednotlivých parametrů experimentu použitých při porovnávání obou algoritmů.

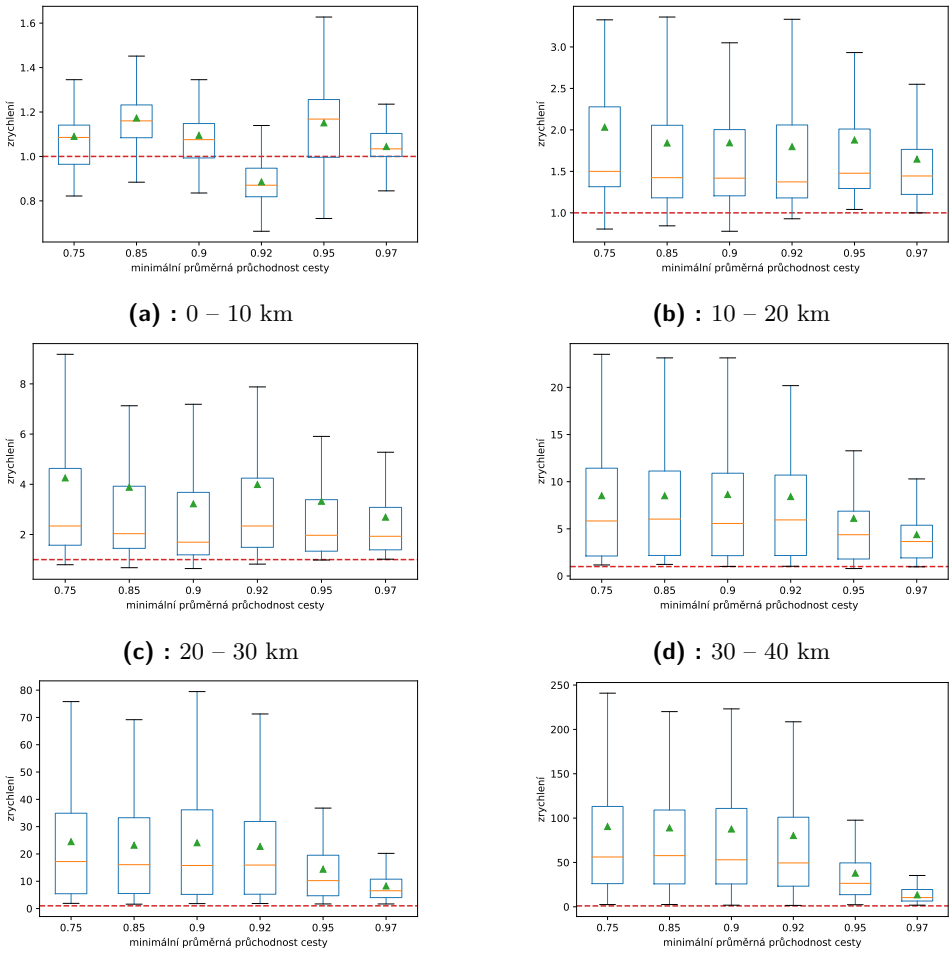
Hodnoty jednotlivých parametrů experimentu, které byly použity při porovnávání obou algoritmů, jsou zobrazeny v tabulce 6.11. Oproti předchozím experimentům jsou v tabulce navíc ještě hodnoty maximální průměrné průchodnosti cesty. Z nich pro graf Šumavy nebyla využita hodnota 0.99, protože v mnoho případech by řešení neexistovalo, a pro graf Prahy zase nebyly použity nízké hodnoty 0.75 a 0.85, které by ve většině problémů vedly na stejná řešení jako v úloze s hodnotou 0.9.

Vzhledem k tomu, že vyhodnocení na obou grafech vede k podobným závěrům, v této sekci popíšeme jen naměřená data na grafu Šumavy. Obrázek zobrazující naměřená data na grafu Prahy je zobrazen v příloze A.3.

Porovnání, kolikrát se zrychlil běh algoritmu oproti multikriteriálnímu algoritmu v jednotlivých sadách problémů, nabízí obrázek 6.7. Zrychlení je opět poměr doby běhu jednotlivých algoritmů pro odpovídající si instance problémů. V obrázku je rovněž porovnáváno několik algoritmů s omezením rozlišených podle zadané minimální průměrné průchodnosti. Poznamenejme, že kvůli přehlednosti grafů v nich nebyly zobrazeny vychýlené hodnoty.

Naměřené údaje potvrzují, že doba běhu algoritmu s omezením je téměř ve všech instancích problémů nižší než je doba běhu multikriteriálního algoritmu. Dokonce u algoritmů s hranicí minimální průměrné průchodnosti pod 0.92 byla na sadě problémů 50+ km většina problémů vyřešena více než 50krát rychleji. Naměřená data na větších instancích problému dále ukazují, že čím je podmínka minimální průměrné průchodnosti vyšší (a tedy i přísnější), tím menší je zrychlení, a tedy i tím delší je doba běhu algoritmu. To je nejspíš způsobeno tím, že kritérium průchodnosti je v algoritmu až druhým kritériem, což vede k tomu, že dříve jsou expandovány stavy s nižší délkou cesty a vyšší váženou neprůchodností, která odpovídá nižší průměrné průchodnosti.

Výsledky experimentů tedy potvrzují, že algoritmus s omezením je v téměř všech porovnávaných variantách rychlejší než multikriteriální algoritmus. Jedinou výjimkou jsou naměřené hodnoty jedné konfigurace algoritmu na nejmenší sadě problémů, kde je však rozdíl v době běhu menší než 10 ms. Na druhou stranu nevýhodou algoritmu s omezením je právě nalezení jediného řešení, které o problému nedá takovou informaci jakou je celá množina nedominovaných řešení. Nemáme-li však tolik času, aby byla nalezena všechna nedominovaná řešení, může být úloha s omezením vhodnou alternativou.



(a) : 0 – 10 km

(b) : 10 – 20 km

(c) : 20 – 30 km

(d) : 30 – 40 km

(e) : 40 – 50 km

(f) : 50+ km

Obrázek 6.7: Porovnání zrychlení doby běhu algoritmů s omezením oproti multikriteriálnímu algoritmu (červená čárkovaná čára) v závislosti na zadané minimální průměrné průchodnosti cesty na jednotlivých sadách problémů grafu Šumavy.

Kapitola 7

Závěr

V této práci jsme se zabývali problémem plánování cest s ohledem na znalost informace, že některé úseky cesty mohou být s určitou pravděpodobností neprůchodné. Cílem práce bylo navrhnout a implementovat algoritmus, který by bral v úvahu tuto pravděpodobnost průchodu jednotlivých hran.

Vzhledem k tomu, že při plánování cest, které by plánovalo cesty pouze podle kritéria průchodnosti, by navržené cesty mohly být často příliš dlouhé, problém jsme formulovali jako multikriteriální problém se dvěma kritérii – délkou cesty a její průchodností. Důležitou součástí této formulace pak byla samotná volba kritéria průchodnosti, kde jsme brali v potaz jak jeho vhodnost pro použití v algoritmu, tak rozumnost jeho interpretace.

Kromě tohoto multikriteriálního problému plánování cest s nejistou průchodností hran, kterým bylo nalézt množinu všech nedominovaných řešení, jsme zformulovali ještě problém hledání nejkratší cesty s omezením průchodnosti. Cílem tohoto problému s omezením bylo nalézt jediné řešení (nejkratší cestu, která splňuje podmínku pro kritérium průchodnosti) z této množiny nedominovaných řešení.

Pro řešení obou těchto problémů jsme navrhli a implementovali podobné algoritmy založené na algoritmu NAMOA*, který byl nejvhodnějším řešením pro náš problém.

Pro vyhodnocení jsme vytvořili 2 grafy pro pěší pokrývající výrazně odlišné oblasti Prahy a Šumavy. K obou grafům jsme vytvořili několik sad problémů, které jsme rozlišili podle délky nejkratší cesty mezi uzly. Každá sada problémů byla tvořena 100 instancemi problémů tvořených dvojicemi náhodných uzlů.

Při zvolení pravděpodobnosti průchodu jednotlivých hran grafu jsme využili informací z OpenStreetMap. Konkrétně jsme vytvořili 2 scénáře průchodnosti, které simulují rozdílné situace s ohledem na počasí. Podle těchto scénářů jsme pak, podle toho jakými značkami jsou označeny v OpenStreetMap, jednotlivým hranám přiřadili určitou pravděpodobnost průchodu.

7.1 Shrnutí výsledků experimentů

Kromě porovnání obou algoritmů a vlivu zvolení pravděpodobnosti průchodu jsme v experimentech dále vyhodnocovali i vliv navržených heuristik a vliv

různých implementací Pareto-množin na běh algoritmu. Naměřené výsledky ukazují, že všechny tyto zvolené faktory mohou běh algoritmu výrazně ovlivnit.

Pokud se týká heuristik, pro většinu problémů dosáhly nejlepšího času předpočítané heuristiky. Pouze pro malé instance problémů (na grafu Šumavy problémy s délkou nejkratší cesty do 10 km) byly tyto heuristiky, především z důvodu jejich nutné inicializace, překonány heuristikami, které nemusí být inicializovány.

Ukázalo se, že i implementace Pareto-množin může mít vliv na rychlost běhu algoritmu a pro větší instance problémů se doba běhu algoritmu může až násobně lišit. Téměř ve všech sadách problémů byla nejrychlejší implementace využívající redukci dimenze. Na grafu Šumavy bylo například na největších instancích problémů zrychlení mezi seřazenou a základní implementací více než sedminásobné. Při porovnání redukce dimenze se seřazenou implementací pak došlo ještě k dvojnásobnému zrychlení.

Porovnání vytvořených scénářů ukázalo, že i zvolení pravděpodobnosti průchodu má vliv na dobu běhu algoritmu. Pro malé instance problémů byl (o desítky až stovky ms) rychlejší scénář „mokro“, který měl více hran s nižší pravděpodobností průchodu. Naopak pro větší instance problémů, kde byl rychlejší scénář „sucho“, byl rozdíl v době běhu algoritmu výraznější (na grafu Prahy více než desetinásobný).

Z porovnání multikriteriálního problému a problému s omezením vyplývá, že běh algoritmu řešícího problém s omezením je rychlejší než běh multikriteriálního algoritmu. Na největší sadě problémů grafu Šumavy byla některými konfiguracemi algoritmu s omezením většina problémů vyřešena i více než 50krát rychleji. A proto v případě, kdy je potřeba rychle nalézt jedno řešení, může být problém s omezením vhodnější.

7.2 Budoucí práce

V této práci jsme pro účely experimentů zvolili pravděpodobnosti průchodu jednotlivých hran grafu na základě informací z OpenStreetMap. Tyto informace nemusí vždy úplně přesně popisovat stav jednotlivých cest. Budoucí prací by mohlo být nějaké sofistikovanější získání hodnot tak, aby byly pravděpodobnosti průchodu jednotlivých hran přesnější, a tím i kvalitnější navržené cesty.

Navržený multikriteriální algoritmus řeší úlohu s minimalizací vážené neprůchodnosti, jejíž řešení jsou jen některými řešeními úlohy 3.5, která minimalizuje průměrnou průchodnost. Proto by mohlo být také zajímavé vytvořit algoritmus řešící tuto úlohu s průměrnými hodnotami a následně oba algoritmy porovnat, jak z hlediska doby běhu algoritmu, tak i z pohledu nalezených řešení.

Oba tyto návrhy jsou již však za rozsahem této práce.

Literatura

[1] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[2] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, chapter 3. Prentice Hall, 3rd edition, 2009.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[4] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. Route planning in transportation networks. In *Algorithm engineering*, pages 19–80. Springer, 2016.

[5] P. Hansen. Bicriterion path problems. In *Multiple Criteria Decision Making Theory and Application*, pages 109–127. Springer Berlin Heidelberg, 1980.

[6] B. S. Stewart and C. C. White. Multiobjective A*. *Journal of the ACM*, 38(4):775–814, 1991.

[7] L. Mandow and J. L. Pérez de la Cruz. A New Approach to Multiobjective A* Search. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 218–223, 2005.

[8] L. Mandow and J. L. Pérez de la Cruz. Multiobjective A* search with consistent heuristics. *Journal of the ACM*, 57(5):1–25, 2008.

[9] F. J. Pulido, L. Mandow, and J. L. Pérez de la Cruz. Dimensionality reduction in multiobjective shortest path search. *Computers & Operations Research*, 64:60–70, 2015.

[10] M. Ehrgott, J. Y. T. Wang, A. Raith, and C. van Houtte. A bi-objective cyclist route choice model. *Transportation Research Part A: Policy and Practice*, 46(4):652–663, 2012.

[11] J. Hrnčíř, P. Žilecký, Q. Song, and M. Jakob. Practical multicriteria urban bicycle routing. *IEEE Transactions on Intelligent Transportation Systems*, 18(3):493–504, 2017.

[12] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.

[13] A. Bar-Noy and B. Schieber. The Canadian Traveller Problem. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 261–270, 1991.

[14] G. H. Polychronopoulos and J. N. Tsitsiklis. Stochastic Shortest Path Problems with Recourse. *Networks*, 27(2):133–143, 1996.

[15] M. Demlová. Text dvanácté přednášky předmětu Logika a grafy. `https://math.fel.cvut.cz/demlova/teaching/lgr/p-lgr812.pdf`, 2018. [Online; accessed 11. 1. 2021].

[16] A. Keilhauer. Problem Reduction – The Most Reliable Path is Just Another Shortest Path. `https://whatsoftwarecando.org/en/problem-reduction-the-most-reliable-path-is-just-another-shortest-path/`, 2015. [Online; accessed 6. 4. 2021].

[17] M. Roosta. Routing through a network with maximum reliability. *Journal of Mathematical Analysis and Applications*, 88(2):341–347, 1982.

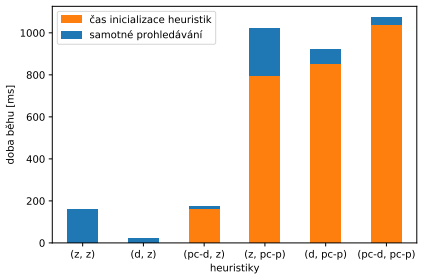
[18] C. T. Tung and K. L. Chew. A multicriteria Pareto-optimal path algorithm. *European Journal of Operational Research*, 62(2):203–209, 1992.

[19] OpenStreetMap Wiki contributors. Cs:Prvky — OpenStreetMap Wiki. `https://wiki.openstreetmap.org/wiki/Cs:Prvky`, 2021. [Online; accessed 29. 3. 2021].

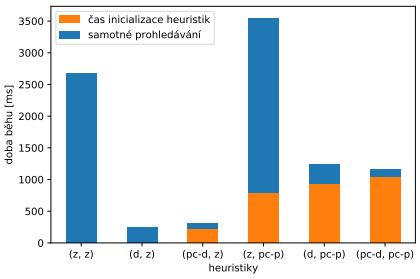
Příloha A

Naměřené hodnoty na grafu Prahy

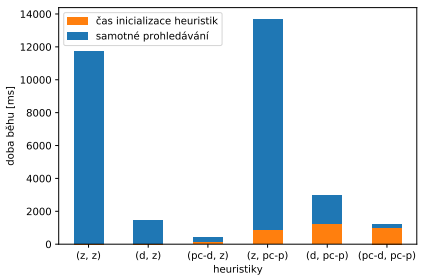
A.1 Vliv heuristik



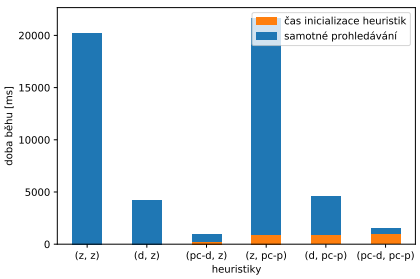
(a) : 0 – 5 km



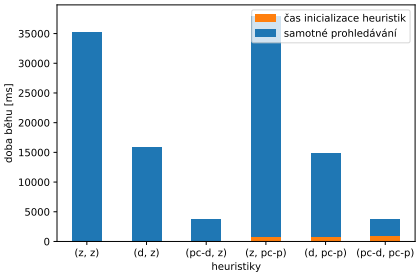
(b) : 5 – 10 km



(c) : 10 – 15 km

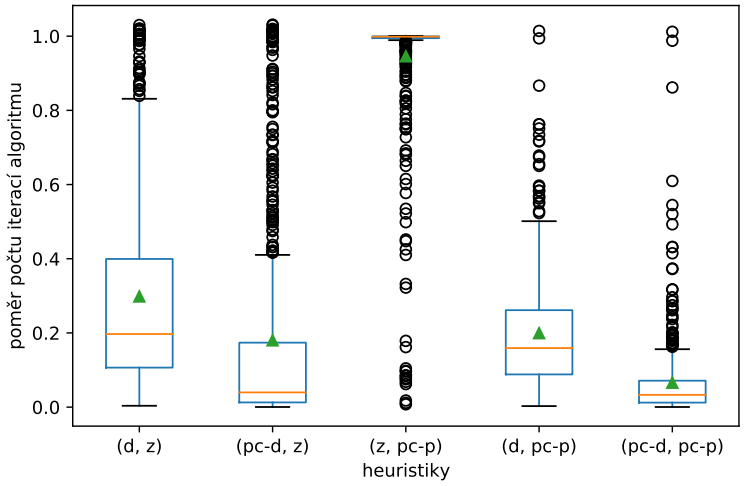


(d) : 15 – 20 km



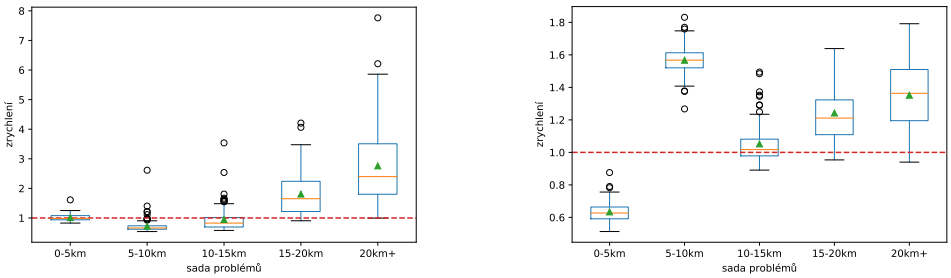
(e) : 20+ km

Obrázek A.1: Medián času inicializace heuristik a celkové doby běhu algoritmu pro jednotlivé kombinace heuristik na sadách problémů grafu Prahy.



Obrázek A.2: Poměr počtu iterací algoritmu s jednotlivými kombinacemi heuristik vůči algoritmu s nulovými heuristikami na grafu Prahy.

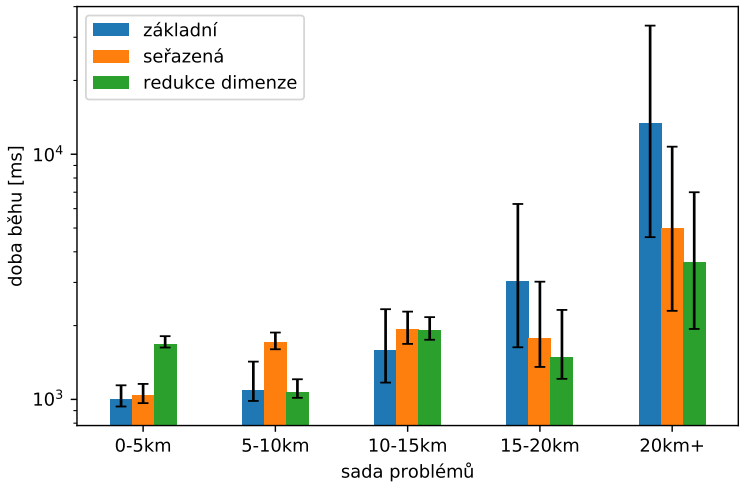
A.2 Vliv implementace Pareto-množin



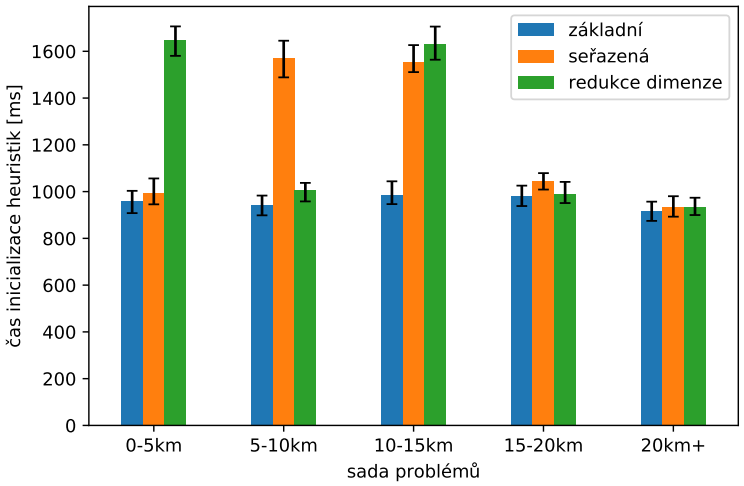
(a) : seřazená vs. základní (červená čárkovaná čára)

(b) : redukce dimenze vs. seřazená (červená čárkovaná čára)

Obrázek A.3: Porovnání zrychlení (tj. poměr času trvání) běhu algoritmu mezi vybranými dvojicemi implementací Pareto-množin pro jednotlivé sady problémů grafu Prahy.

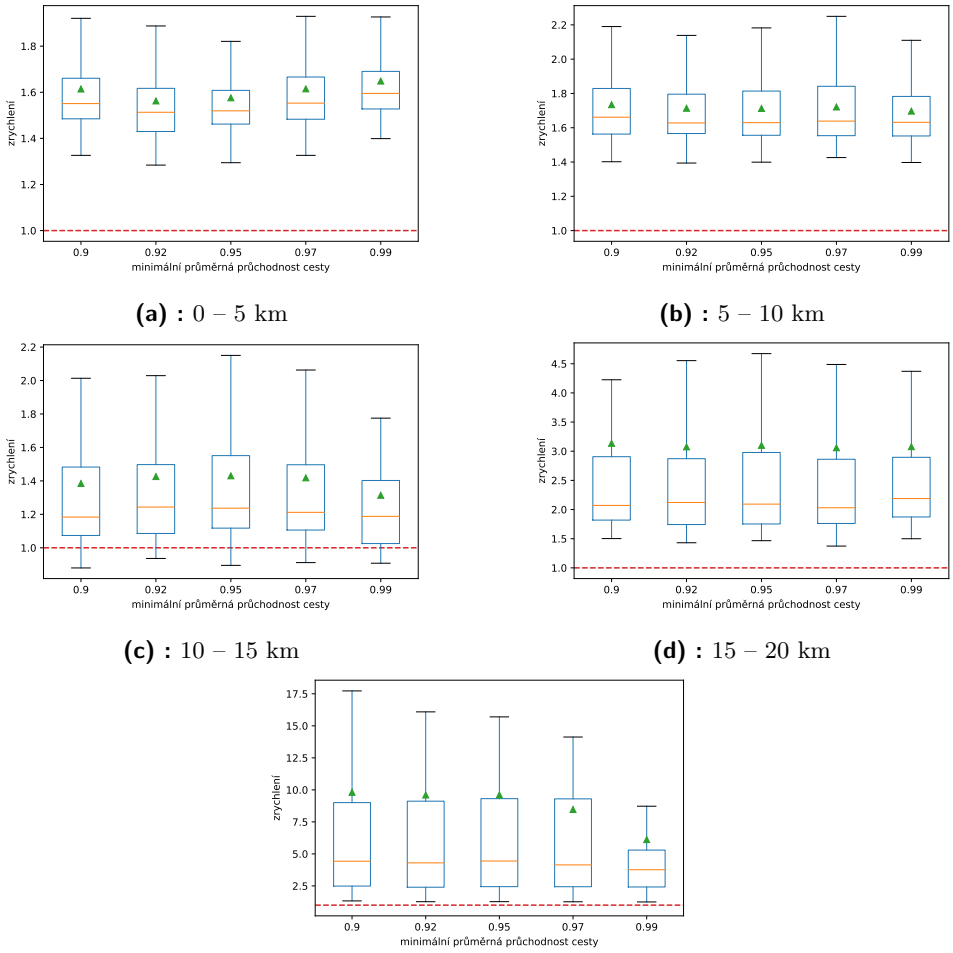


Obrázek A.4: Porovnání mediánu doby běhu (v logaritmickém měřítku) algoritmu s jednotlivými implementacemi Pareto-množin pro jednotlivé sady problémů grafu Prahy. Černě je vyznačeno rozpětí hodnot mezi kvartily $Q_{0.25}$ a $Q_{0.75}$.



Obrázek A.5: Porovnání mediánu času inicializace heuristik algoritmu s jednotlivými implementacemi Pareto-množin pro jednotlivé sady problémů grafu Prahy. Černě je vyznačeno rozpětí hodnot mezi kvartily $Q_{0.25}$ a $Q_{0.75}$.

A.3 Porovnání multikriteriálního algoritmu s algoritmem řešící úlohu s omezením průchodnosti



(a) : 0 – 5 km

(b) : 5 – 10 km

(c) : 10 – 15 km

(d) : 15 – 20 km

(e) : 20+ km

Obrázek A.6: Porovnání zrychlení doby běhu algoritmů s omezením oproti multikriteriálnímu algoritmu (červená čárkovaná čára) v závislosti na zadané minimální průměrné průchodnosti cesty na jednotlivých sadách problémů grafu Prahy.