**ÚVOD**

Dopravná infraštruktúra je neoddeliteľnou súčasťou hlavného mesta najmä vtedy ak mesto nemá vybudovanú podzemnú dráhu.

MHD slúži všetkým ľudom, ktorí nevlastnia osobné vozidlá, sú zdravotne znevýhodnení, ale účastníkom cestovného ruchu.

/

MHD slúži všetkým ľudom, najmä tým, ktorí nevlastnia osobné vozidlá, sú zdravotne znevýhodnení....

Navyše minimalizuje počet vozidiel v meste, čím odbremeňuje mesto od dopravných zápch. Prispieva k ekologickej preprave v meste a v niektorých prípadoch môže byť mestská hromadná doprava najrýchlejším a najekonomickejším spôsobom prepravy.

Aby bolo cestovanie verejnou dopravou komfortné a cestujúci viac motivovaní využívať tento spôsob prepravy, kľúčovú rolu zohráva presnosť spojov, /ich vysoká frekvencia/ vysoká frekvencia spojov, hustota rozsadenia /rozmiestnenia/ zastávok, minimálna nutnosť peších presunov medzi zastávkami, pohodlnosť cestovania a mnohé iné faktory.

Študent informatiky je schopný prispieť k zefektívneniu MHD vytvorením aplikácie, ktorá bude ponúkať čo najpresnejšie vyhľadávanie spojov, okolitých zastávok a prispeje k celkovému sprehľadneniu hromadnej dopravy.

Študent informatiky je schopný prispieť k zefektívneniu MHD vytvorením aplikácie, ktorá bude ponúkať čo najpresnejšie vyhľadávanie spojov a okolitých zastávok, aby tak prispel k celkovému sprehľadneniu mestskej hromadnej dopravy.

Aktuálne sú dostupné mnohé aplikácie, ktoré poskytujú spomínané funkcionality. Okrem možnosti vyhľadávania spojov ponúkajú aj priamu kúpu cestovného lístka, sledovanie pohybu vozidiel v reálnom čase, ako aj informáciu o možnosti prestupu na prímestskú dopravu. Avšak vyhľadávanie je realizované nad statickými cestovnými poriadkami.

Dopravný podnik Bratislava disponuje informáciami o aktuálnych polohách vozidiel, z čoho vie následne určiť meškanie jednotlivých spojov. Tieto informácie sú zobrazené na elektronických tabuliach na vybraných zastávkach. Prečo teda nevyužiť tieto dáta na spresnenie vyhľadávania spojení?

Rozhodli sme sa vytvoriť aplikáciu, ktorá bude pri vyhľadávaní cesty /ciest, spojení/ prihliadať na aktuálnu dopravnú situáciu. Zároveň chceme vytvoriť túto aplikácie pre široké spektrum cestujúcich. Vieme, že existujú cestujúci, ktorí preferujú minimálny počet prestupov, iní hľadajú najrýchlejšiu cestu do cieľa a sú aj takí, ktorí potrebujú na svoj presun nízkopodlažné vozidlá. Preto táto aplikácia bude poskytovať aj alternatívne optimálne cesty.

Práca popisuje priebeh vývoja aplikácie a je rozdelená do štyroch kapitol.

V prvej kapitole sú zhrnuté poznatky, ktoré sme nadobudli štúdiom /pri štúdiu/ zaujímavých vedeckých článkov,

ako aj skúmaním funkcií už existujúcich aplikácií na vyhľadávanie ciest v bratislavskej mestskej hromadnej doprave. V literatúre sa vyhľadávanie ciest často spája s Dijkstrovým algoritmom.

Spomíname aj algoritmus A\* spolu s jeho rôznymi optimalizáciami a modifikáciami. Jedna z modifikácií sa dokonca zaoberá časovými dátami a počíta s multimodalitou dopravnej siete. Pri grafových algoritmoch sú často spomínané aj minimalizácie prehľadávaného priestoru, ktoré by mohli zrýchliť beh algoritmu. Zmieňujeme sa aj o takej literatúre, ktorá by nám mohla pomôcť s vyhľadávaním alternatívnych ciest. Spomedzi všetkých algoritmov na vyhľadávanie ciest nás napokon najviac zaujal negrafový algoritmus RAPTOR, čo je skratkou pre Round bAsed Public Transit Optimal Router. Ako vyplýva už z názvu, je určený na vyhľadávanie ciest v sieti verejnej dopravy. V tejto kapitole popisujeme jeho základnú verziu, ale aj vylepšenú verziu, ktorá sa zaoberá vyhľadávaním alternatívnych ciest. Výhodou algoritmu je, že je prispôsobiteľný tak, aby spracovával aj dynamické dáta.

Druhou kapitolou je kapitola Návrh, v ktorej vysvetľujeme, ako sme sa rozhodli vytvoriť našu aplikáciu z pohľadu funkcií, vyhľadávacieho algoritmu, dát a architektúry. Popisujeme funkcie, ktoré bude aplikácia cestujúcim ponúkať, pričom sme sa inšpirovali existujúcimi aplikáciami a vlastnými skúsenosťami. V tejto kapitole objasňujeme výber algoritmu na vyhľadávanie ciest. Vybraným algoritmom je RAPTOR algoritmus, ktorý modifikujeme, aby sme dosiahli požadované funkcie aplikácie. Ďalej po podrobnej analýze približujeme návrh spracovania dát a popíšeme dáta statických cestovných poriadkov uložených vo formáte GTFS. Dopravný podnik Bratislava nám poskytol aj dáta o meškaní vozidiel. Opisujeme, ako ukladáme dáta do databázy a do dátovej štruktúry pre algoritmus. V neposlednom rade popisujeme aj návrh architektúry systému.

V kapitole Implementácia približujeme technológie použité pri vývoji aplikácie. Spomíname riešenie problémov vzniknutých pri importovaní a spracovávaní statických dát a dát o meškaní. Popisujeme, akým spôsobom je implementovaná dátová štruktúra a ako do nej ukladáme dáta o meškaní vozidiel. Pre správne fungovanie aplikácie bolo potrebné simulovať čas v minulosti, keďže poskytnuté dáta mali platnosť v roku 2018. Ďalej ukazujeme implementáciu niektorých kľúčových častí RAPTOR algoritmu. Na ukážku poskytujeme časti kódu, ktorými sme obohatili algoritmus, aby prihliadal na preferencie zadané používateľom. Približujeme taktiež problémy, ktoré vznikali pri implementácii na klientskej strane.

V poslednej kapitole spomíname testovacie dáta, ktoré sme vytvorili na overenie správnosti vyhľadávacieho algoritmu. Na ukážkach vyhľadaných ciest približujeme, ako sa algoritmus postupne vyvíjal a ako sa podľa toho menil zoznam vyhľadaných ciest. /Na ukážkach vyhľadaných ciest približujeme postupný vývoj algoritmu a v nadväznosti na jeho vývoj aj zmenu vyhľadaných ciest./ ~~Ukazujeme, ako sa algoritmus vysporiadal s pridanými používateľskými preferenciami a ako dokázal prihliadnuť pri výpočte na meškanie spojov. Keďže nás nezaujíma jedna najkratšia cesta, ale viacero optimálnych alternatív, ukazujeme aj, ako vyzerá zoznam vyhľadaných ciest po filtrovaní.~~

**ABSTRAKT**

Diplomová práca popisuje vývojprogresívnej webovej aplikácie na vyhľadávanie ciest v bratislavskej mestskej hromadnej doprave. Aplikácia pri vyhľadávaní ciest prihliada na aktuálne meškanie vozidiel. Nehľadá len jednu najkratšiu cestu, ale ponúka cestujúcim alternatívne optimálne cesty a možnosť navolenia rôznych preferencií. Medzi jej ďalšie funkcionality patrí vyhľadávanie cesty z aktuálnej polohy cestujúceho. Zobrazenie všetkých liniek a ich trás je tiež súčasťou tejto aplikácie, ako aj možnosť zvolenia zastávok priamo z mapy.

/Diplomová práca popisuje vývojprogresívnej webovej aplikácie na vyhľadávanie ciest v bratislavskej mestskej hromadnej doprave. Aplikácia pri vyhľadávaní ciest prihliada na aktuálne meškanie vozidiel. Nehľadá len jednu najkratšiu cestu, ale ponúka cestujúcim alternatívne optimálne cesty a možnosť navolenia rôznych preferencií. Medzi jej ďalšie funkcionality patrí vyhľadávanie cesty z aktuálnej polohy cestujúceho, ako aj možnosť voľby zastávky priamo z mapy. Súčasťou tejto aplikácie je taktiež zobrazenie všetkých liniek a ich trás./

Kľúčovou časťou diplomovej práce je výber vhodného vyhľadávacieho algoritmu, ktorý sa dokáže vysporiadať s časovo závislými dátami, alternatívnymi cestami a multimodalitou hromadnej dopravy.

Našou voľbou je negrafový RAPTOR algoritmus, ktorý sme prispôsobili tak, aby vedel pracovať s dynamickými dátami a aby bol schopný spracovať preferencie vyhľadávania.

Kľúčové slová: progresívna webová aplikácia, hľadanie optimálnej cesty, alternatívne cesty, multimodalita, RAPTOR algoritmus

**ABSTRACT**

Diploma thesis ...

Application take into consideration actual delay of vehicles while searching paths.

Application is not searching for only one shortest path, but it is providing alternative optimal paths as well as possibility to choose between variety of searching preferences.

Key part of diploma thesis is also selection of applicable searching algorithm which can deal with time dependant data and searching alternative paths.

Our choice is non-graph RAPTOR algorithm, which we modified to handle dynamic data and searching preferences.

Key words: RAPTOR algorithm, optimal path finding, multimodality, alternative paths, progressive web application

**POĎAKOVANIE**

Chcela by som sa poďakovať vedúcemu mojej diplomovej práce doc. RNDr. Milanovi Ftáčnikovi, CSc. za jeho cenné rady, pripomienky, ústretový prístup a usmerňovanie pri tvorbe diplomovej práce. Moja vďaka patrí aj Mgr. Ľuborovi Illekovi za jeho čas a rady. Za poskytnutie dát ďakujem Dopravnému podniku Bratislava.

**TESTOVANIE**

V kapitole Testovanie sa budeme snažiť ukázať schopnosť aplikácie vyhľadávať optimálne cesty s prihliadnutím na meškanie. Správnosť vyhľadaných ciest sa dokazuje ťažko, keďže nemáme k dispozícii porovnateľné dáta. Existujúce aplikácie vyhľadávajú cesty bez prihliadnutia na meškanie.

Keďže máme k dispozícii dáta s platnosťou cestovných poriadkov v minulosti a existujúce aplikácie neumožňujú vyhľadávanie v ďalekej minulosti, nevieme porovnať ani vyhľadané cesty bez prihliadnutia na meškanie.

Vyhľadávací algoritmu sme testovali postupne. Najskôr algoritmus hľadal len jednu najkratšiu cestu nad statickými cestovnými poriadkami. Potom sme algoritmus vylepšili, aby prihliadal aj na meškanie vozidiel. Následne sme upravili vyhľadávanie, aby výsledná cesta nebola len jedna najkratšia, ale viacero optimálnych. Na záver sme upravili algoritmus, aby prihliadal aj na používateľské preferencie zadané pri vyhľadávaní.

Dáta získané od Dopravného podniku Bratislava sú príliš robustné a kontrola algoritmu človekom nad týmito dátami bola nemožná. Vytvorili sme preto testovaciu sadu dát, nad ktorými sme spúšťali vyhľadávanie. Testovacie dáta sme bližšie spomínali v ..... . Pre lepší prehľad v testovacích dátach sme nakreslili schému liniek a vygenerovali cestovný poriadok. Algoritmus, ktorý vyhľadával nad testovacími dátami kontrolovali traja cestujúci. Podľa ich názorov a skúseností sme vytvárali a upravovali filtre. Filtre slúžia na minimalizovanie množiny všetkých ciest, ktoré vedú zo skupiny začiatočných zastávok do konečnej skupiny zastávok na také cesty. Všetky vyfiltrované cesty však musia byť optimálne.

V nasledujúcich ukážkach ukážeme ako sa menila množina vyhľadaných ciest v závislosti od vývoja algoritmu.

**MEŠKANIE**

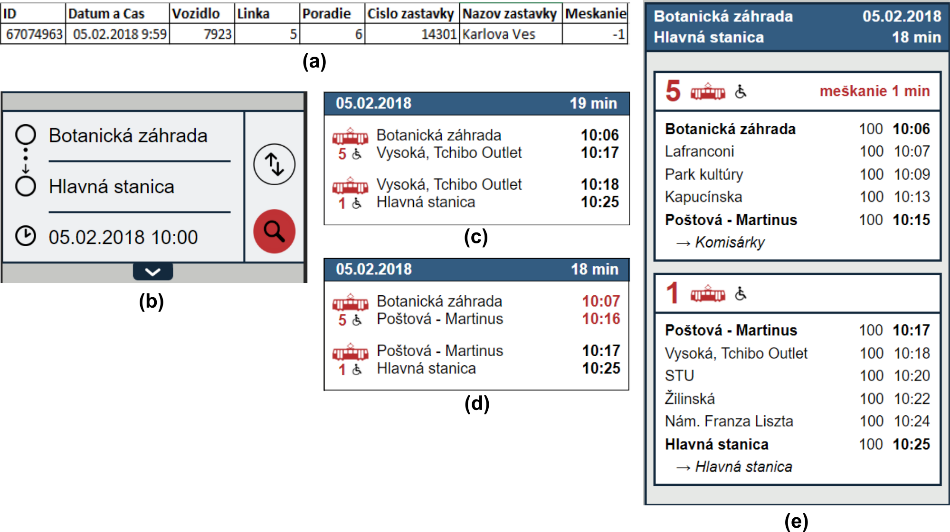
V tejto ukážke môžeme pozorovať, ako sa zmenila vyhľadaná cesta v závislosti od meškania.

Na obrázku (a) môžeme vidieť riadok z dát o meškaní vozidiel, ktorý poukazuje na to, že v deň 5.2.2018 v čase 9:59 nadobudla linka 5 na zastávke Karlova Ves meškanie 1 minútu. Vyhľadávanie sme spustili 5.2.2018 v čase 10:00. Deň 5.2.2018 je v konfigurácii nastavený ako aktuálny dátum.

Hľadáme cestu s vyhľadávacími parametrami na obrázku (b). Najskôr algoritmus hľadal len nad statickými cestovnými poriadkami. Našiel preto jednu najrýchlejšiu cestu (obrázok (c)).

Po tom ako sme v algoritme začali prihliadať na meškanie vozidiel, zmenil sa aj výsledok vyhľadávania (obrázok (d)).

Jazda linky 5 príde na zastávku Botanická záhrada s meškaním a z toho dôvodu sa mení aj prestupná zastávka medzi jazdami. Predvolený minimálny čas na prestup medzi linkami je 1 minúta. Keďže jazda linky 5 má meškanie, cestujúci by nemal žiadny čas na prestup na zastávke Vysoká, Tchibo Outlet. Prestupnou zastávkou sa preto stala zastávka Poštová, MARTINUS.



Všimnime si ešte označenie meškania.

Meškajúci spoj má zafarbené časy červenou farbou.

Po kliknutí na detail (obrázok (e) ) vidíme, aj hodnotu meškania spoja. Časy z druhého spoja sú čiernou farbou, pretože tento spoj ešte nevyrazil zo začiatočnej zastávky v čase vyhľadávania (10:00).

**ALTERNATÍVNE CESTY**

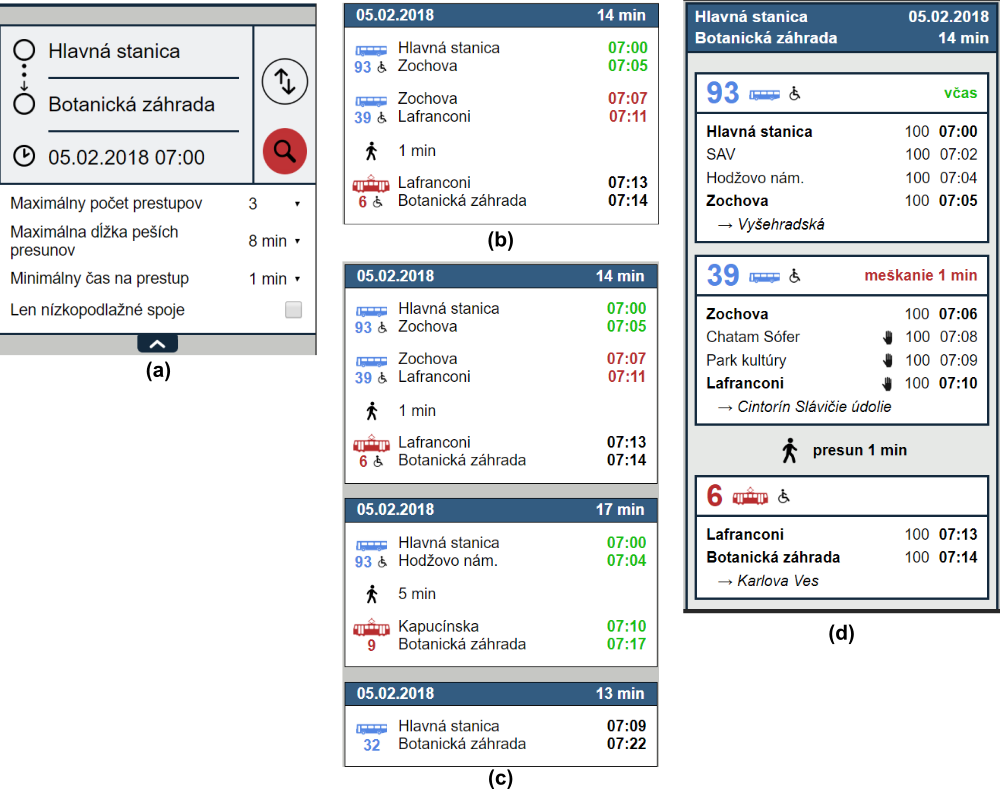
Následne sme vylepšili vyhľadávanie tak, aby aplikácia cestujúcemu ponúkla vždy alternatívne optimálne cesty. Prihliadame už aj na meškanie spojov. Spustili sme vyhľadávanie s parametrami na obrázku (a). Povolili sme až 3 prestupy, aby sme mohli ukázať ako funguje filtrovanie ciest.

Doteraz aplikácia vracala iba jednu najrýchlejšiu cestu a so zadanými parametrami by bola vyhľadaná cesta na obrázku (b).

Po vylepšení vyhľadávania nám aplikácia ponúkla množinu ciest, ktoré sú na obrázku (c). Vidíme prvú najrýchlejšiu cestu s 2 prestupmi, druhú pomalšiu s jedným prestupom a tretiu s najneskorším časom príchodu do cieľa, ale bez nutnosti prestupu.

Vo filtrovacom mechanizme ciest je podmienka, že najrýchlejšiu cestu zobrazujeme vždy. Cesty z neskoršími príchodmi do cieľa zobrazujeme len vtedy, ak majú menej prestupov. Avšak nemôžu prísť do cieľa s veľkým oneskorením po najrýchlejšej ceste. Rozdiel v časoch musí byť menší alebo rovný, ako rozdiel počtu prestupov \* x minút. V projekte sme si určili x = 5 minút. Preto je prvá cesta poskytnutá cestujúcemu.

Najrýchlejšou cestou je prvá cesta. Rozdiel v prestupoch prvej a druhej cesty je 1 a rozdiel v časoch príchodu do cieľa sú 3 minúty. (3 minúty <= 1\*5 minút).

Rozdiel v prestupoch medzi prvou a treťou cestou sú 2 prestupy a rozdiel v časoch príchodu do cieľa je 8. (8 minút <= 2\*5 minút).

Na obrázkoch (b) a (c) si môžeme všimnúť zafarbené časy zelenou farbou. Čo znamená, že jazda linky už vyrazila z prvej zastávky a nemá žiadne meškanie. V detaile prvej cesty (obrázok (d)) môžeme vidieť označenie „včas“ pri nemeškajúcej jazde.

**MEŠKANIE A ALTERNATÍVNE CESTY**

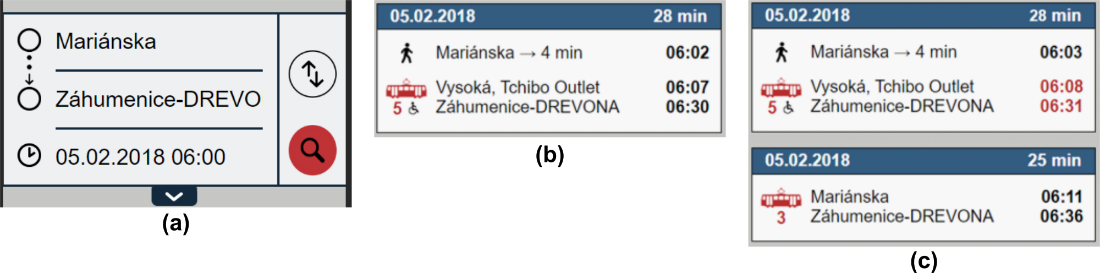
Pomocou nasledujúcej ukážky znázorníme, aký má vplyv meškanie na zobrazenie optimálnych ciest

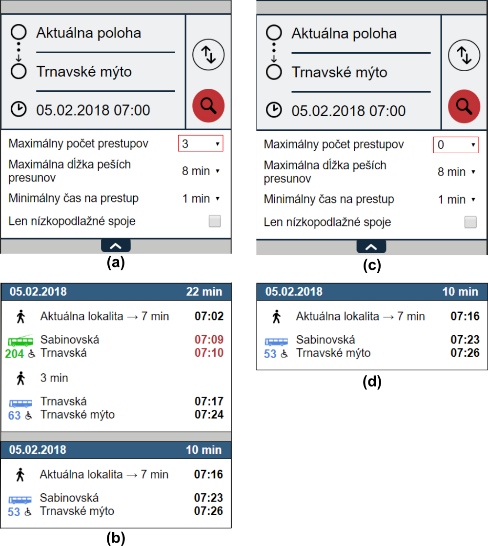
Na obrázku (a) vidíme vyhľadávacie parametre. Najskôr sme definovali v systéme aktuálny deň 4.2.2018, aby získané meškania pri vyhľadávaní 5.2.2018 nemali vplyv na vyhľadávanie. Našli sme jednu cestu najviac vyhovujúcu a žiadne alternatívne optimálne cesty.

Potom sme zmenili dátum v systéme na 5.2.2018 a čas 6:00. Teraz už algoritmus prihliada na meškania, ktoré vznikli do 6:00 hodiny. Jazda linky 5 má minútové meškanie (označené červenou farbou). Zaujímavé však je, že pribudla aj alternatívna cesta, ktorá nevyžaduje začiatočný peší presun. V predchádzajúcom hľadaní bola táto cesta odfiltrovaná. Potrebný peší presun na začiatku tiež penalizujeme ako prestup.

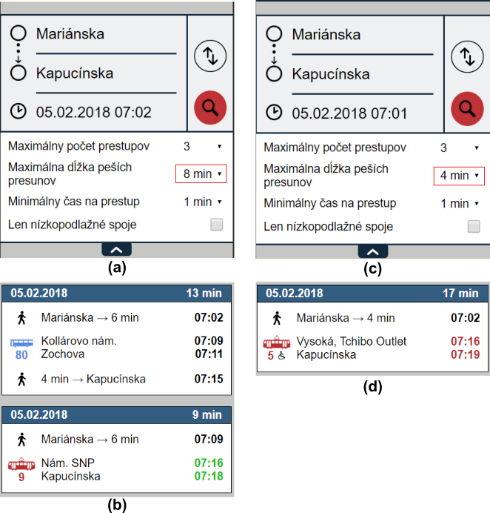
V prípade (b) bola priama cesta odfiltrovaná, pretože rozdiel v časoch príchodu do cieľa je väčší ako a rozdiel v počte prestupov \* 5 minút. (6 minút <= 1\*5minút).

V prípade (c ) je už rozdiel v časoch rovných 5 minút. Preto cesta s menším počtom prestupov nebola vyfiltrovaná.



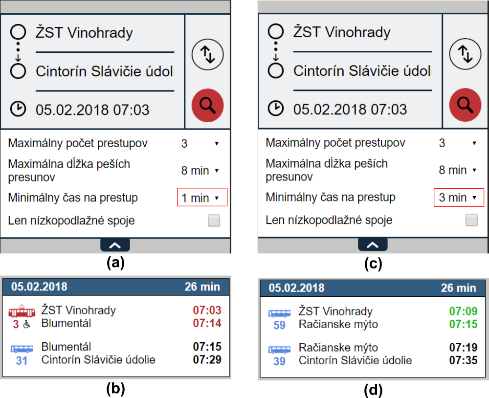
Do aplikácie sme pridali ďalší formulár, aby používateľ mohol zadávať vyhľadávacie kritériá. Zároveň sme upravili algoritmus, aby počas vyhľadávania ciest prihliadal na zadané parametre. Na obrázku (a) môžeme vidieť zadané parametre vyhľadávania. Všimnime si najmä, že maximálny počet prednastavený na hodnotu 3 a začiatočný bod nie je zastávka, ale aktuálna poloha používateľa.

Vyhľadané cesty (obrázok (b)) sú dve. Jedna cesta obsahuje 1 prestup a druhá neobsahuje žiadny. Po zmene vyhľadávacích parametrov (obrázok (c)), je vyhľadaná cesta už len jedna – tá bez prestupu (obrázok (d)).

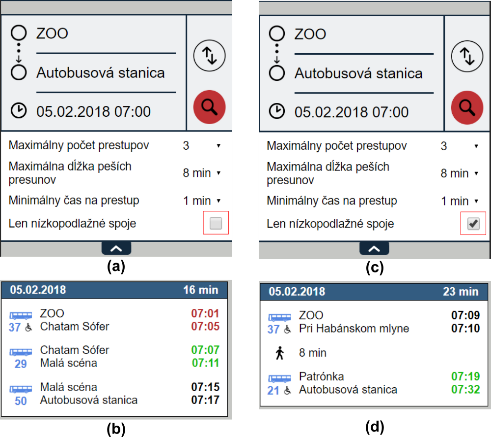


Na ďalšom obrázku ukážeme, ako sa algoritmus vysporiadava s obmedzením dĺžky pešieho presunu. Po zadaní parametrov na obrázku (a), aplikácia ponúkla cesty na obrázku (b). Tu vidíme ponuku dvoch optimálnych ciest. Prvá cesta je najrýchlejšia, ale obsahuje 2 pešie presuny. Druhá cesta obsahuje len jeden peší presun.

Následne sme zmenili vo vyhľadávacích parametroch hodnotu maximálnej dĺžky pešieho presunu na 4 minútu (obrázok ©). Na obrázku (d) môžeme vidieť vyhľadanú 1 cestu. Najrýchlejšia cesta bola vyfiltrovaná, keďže obsahovala 6 – minútový peší presun.



V ďalšej ukážke znázorníme fungovanie nastavenia minimálneho času na prestup. Na obrázku (a) je hodnota minimálneho času na prestup nastavená na hodnotu 1. Vyhľadaná cesta (obrázok (b)) vyžaduje 1 prestup a čas na prestup medzi linkami je 1 minúta. Po tom ako sme zmenili hodnotu minimálneho času na prestup na hodnotu 3 vo vyhľadávacích parametroch (obrázok (c)), dostávame inú vyhľadanú cestu s jedným prestupom (obrázok (d)). Na prestup medzi zastávkami má cestujúci 4 minúty.



V poslednej ukážke dokážeme fungovanie vyhľadávania s prihliadnutím na obmedzenie len nízkopodlažných vozidiel. Na obrázku (a) vidíme zobrazené parametre vyhľadávania a na obrázku (b) vidíme vyhľadanú cestu. Cesta obsahuje 2 prestupy a je zložená z troch jázd. Len pre jednu z jázd tejto cesty platí, že má priradené nízkopodlažné vozidlo. Po zmene vyhľadávacích parametrov (obrázok (c)) je vyhľadaná iná cesta. Táto cesta obsahuje 2 jazdy, pričom obe jazdy majú priradené nízkopodlažné vozdilo.

**ZÁVER**

Podarilo sa nám navrhnúť a naimplementovať aplikáciu na vyhľadávanie optimálnych ciest v MHD Bratislava. Dominantnou funkciou aplikácie je, že prihliada na aktuálnu dopravnú situáciu a pri vyhľadávaní ciest zohľadňuje aj meškanie vozidiel. Keďže cestujúci majú rôzne preferencie a optimálna cesta môže byť pre každého iná, aplikácia neponúka používateľovi len jednu cestu, ale viacero alternatívnych optimálnych ciest. Používateľ môže zvoliť jeho aktuálnu polohu ako začiatočný bod. Používateľ vie zadať začiatočnú a cieľovú zastávku výberom zo zoznamu zastávok v prípade, že pozná jej názov alebo si môže zvoliť zastávku z mapy. Okrem iného aplikácia ponúka aj zobrazenie všetkých liniek a ich trás.

/V prípade, že používateľ pozná názvy zastávok, môže ich vybrať zo zoznamu ako začiatočnú a cieľovú zastávku, prípadne si ich môže zvoliť z mapy./

Pred samotným návrhom a vývojom aplikácie sme študovali literatúru, ktorá sa venovala riešeniu rôznych otázok vznikajúcich pri riešení hľadania /hľadaní/ ciest vo verejnej doprave. Kľúčovým problémom bol najmä výber algoritmu na vyhľadávanie ciest. Vo fáze štúdia odbornej literatúry sme sa stretli najmä s grafovými riešeniami hľadania ciest. Našli sme aj rôzne modifikácie a optimalizácie týchto algoritmov. My sme sa však zaujímali najmä o také algoritmy, ktoré sa dokážu vysporiadať s časovo závislými dátami, rôznymi typmi dopravných prostriedkov (módmi) a alternatívnymi cestami. Pri tvorení požiadaviek na aplikáciu sme sa inšpirovali podobnými existujúcimi aplikáciami, ktoré okrem vyhľadávania spojov ponúkajú rôzne iné funkcionality. O tom sa zmieňujeme v kapitole Východiská.

Napokon sme sa rozhodli pre algoritmus, ktorý využíva fakt, že mestská doprava jazdí po vopred určených trasách. Týmto algoritmom je RAPTOR algoritmus ktorého výhodou je, že je pomerne jednoducho prispôsobiteľný. Navrhli sme zmeny v algoritme, ktoré bolo nutné urobiť, aby algoritmus vedel pracovať s dátami poskytnutými Dopravným podnikom Bratislava. Keďže algoritmus vyžadoval určitú formu dát, upravili sme dáta tak, aby ich algoritmus vedel spracovať. Bolo nutné navrhnúť ďalšie zmeny tak, aby algoritmus vedel pri vyhľadávaní zohľadniť používateľské preferencie~~. Po získaní množiny ciest začínajúcich v štartovnej zastávke a končiacich v cieľovej zastávke,~~  Navrhli sme filtre, ktoré minimalizujú množinu všetkých ciest na množinu optimálnych ciest.

Vo fáze implementácie sme natrafili na niekoľko problémov s dátami, keďže nevyhovovali formátu, ktoré RAPTOR algoritmus vyžadoval. Museli sme bojovať /sa potýkať/ aj s veľkosťou dátovej štruktúry pre algoritmus, ako aj sa vysporiadať s historickými dátami. Na klientskej strane vznikol problém so zobrazením zastávok na mape tak, aby zobrazenie bolo čo najviac používateľsky prívetivé.

Aplikáciu sme testovali v jednotlivých fázach vývoja. Algoritmus sme stále obohacovali o nové vylepšenia. Začali sme vyhľadávaním nad statickými cestovnými poriadkami, pokračovali sme prispôsobením algoritmu a dátovej štruktúry tak, aby prihliadal na meškania spojov a poskytoval cesty zodpovedajúce aktuálnej dopravnej situácii. Ďalej sme vytvorili a aplikovali filtre na cesty tak, že výsledkom sú vždy optimálne cesty. V poslednom kroku sme prispôsobili algoritmus tak, aby pri vyhľadávaní vedel zohľadniť používateľské kritériá. Pre potreby testovania, ale aj vylepšovania algoritmu sme si vytvorili testovacie dáta. Správnosť vyhľadaných ciest tak mohla byť kontrolovaná.

Aplikácia má niekoľko častí, ktoré by bolo možné vylepšiť. Počas implementácie sme objavili dáta, ktoré sa nám nepodarilo zanalyzovať. Išlo o dáta, ktoré predstavovali časy odchodu zo zastávok po 23:59 hodine. Tieto dáta sme odignorovali a z toho dôvodu nemusí správne fungovať vyhľadávanie na prelome dní.

V tomto riešení zapracovávame filtre na vyhľadané cesty po zbehnutí algoritmu. Stálo by za zváženie, či nie je efektívnejšie ich filtrovať už počas behu algoritmu.

V neposlednom rade by mohlo byť vyhľadávanie paralelizované na viacerých CPU jadrách. Linky v dátovej štruktúre sú zapísané nezávisle a pri vyhľadávaní môže každé jadro spracovávať inú podmnožinu liniek zvlášť.