

MATEMATICKO-FYZIKÁLNÍ FAKULTA

Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Filip Čižmář

Analýza real-time dat vozidel městské hromadné dopravy

Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. Mgr. Martin Nečaský, Ph.D.

Studijní program: Informatika

Studijní obor: SW a datové inženýrství

Praha 2020

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

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

V dne
Podpis autora

Především děkuji svému vedoucímu, který mi pomohl najít zajímavé zaměření mé práce, přístup k otevřeným datům a pomoc při vypracování.

Stejně tak děkuji i Janu Vlasatému, který mi poskytl odbornou pomoc při získávání dat z datové platformy Golemio a inspiraci pro obsah mé práce.

A také děkuji všem přátelům, kteří mi pomohli se stylistikou psaného textu.

Název práce: Analýza real-time dat vozidel městské hromadné dopravy

Autor: Filip Čižmář

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. Mgr. Martin Nečaský, Ph.D., Department of Software Engineering

Abstrakt: Tato práce se zaměřuje na analýzu dostupných otevřených real-time dat z vozidel hromadné dopravy v Praze a okolí. Jejím cílem je poskytnout základní statistické informace a na základě historických dat zlepšit odhad zpoždění spoje na trase mezi dvěma referenčními body. Jako vedlejší produkt vytvoří aplikaci pro webové rozhraní, kde zobrazí aktuální polohy spojů do mapového podkladu a rozšiřující infomace o nich. Aplikace bude aktivně interagovat s uživatelem.

Klíčová slova: zpoždění MHD otevřená data veřejná doprava

Title: Analysis of real-time data of public transport vehicles

Author: Filip Čižmář

Department: Department of Software Engineering

Supervisor: doc. Mgr. Martin Nečaský, Ph.D., Katedra softwarového inženýrství

Abstract: Abstract. Tato práce se zaměřuje na analýzu dostupných otevřených real-time dat z vozidel hromadné dopravy v Praze a okolí. Jejím cílem je poskytnout základní statistické informace a na základě historických dat zlepšit odhad zpoždění spoje na trase mezi dvěma referenčními body. Jako vedlejší produkt vytvoří aplikaci pro webové rozhraní, kde zobrazí aktuální polohy spojů do mapového podkladu a rozšiřující infomace o nich. Aplikace bude aktivně interagovat s uživatelem.

Keywords: delay open data public transport

Obsah

Úvod	2
1 Analýza problému a jeho řešení	3
1.1 Popis problému odhadu zpoždění	3
1.1.1 Současná řešení	3
1.2 Analýza požadavků na uživatelskou aplikaci	3
1.2.1 Poskytovatelé mapových podkladů	4
1.2.2 Současná řešení	5
2 Analýza zdroje dat	8
2.1 Přístup k datům	8
2.1.1 Dokumentace	8
3 Zpracování dat	11
3.1 Databáze	11
3.1.1 Obsluha databáze	12
4 Vizualizace dat	15
4.1 Klientská část	15
4.2 Serverová část	16
5 Algoritmus odhadu zpoždění	17
5.1 Základní předpoklady	17
5.1.1 Analýza dat	17
5.2 Návrh modelu	18
5.2.1 Lineární model	18
5.2.2 Polynomiální model	19
5.2.3 Model pomocí konkávního obalu	19
Závěr	20
Seznam použité literatury	21
Seznam obrázků	22
Seznam tabulek	23
Seznam použitých zkratek	24
A Přílohy	26
A.1 První příloha	26

Úvod

Městská hromadná doprava v Praze a Středočeském kraji je jeden z hlavním pilířů přepravy osob v této oblasti. Svým rozsahem a důležitostí se přímo dotýká každého z nás a její fungování do značné míry ovlivňuje naše konání v krátkém i dlouhém časovém horizontu.

Každého cestujícího v přepravě jistě někdy trápilo zpoždění svého spoje. To člověka přivádí k myšlenkám jestli by nebylo možné určit s jakou pravidelností, pokud s nějakou, takové zpoždění vznikají. A jestli by nemohl být informován za včasu o vzniklé anomálii.

Ve vymezené oblasti operuje spousta soukromých i městských přepravců. Ti kteří spadají do naší zájmové oblasti zastřešuje organizace ROPID, která objednává jednotlivé spoje. Pro tuto práci je však důležité, že tato organizace zadala jednotlivým dopravcům vysílat aktuálním polohy jejich vozů. Tato data jsou přes zprostředkovatele zveřejněny na pražské datové platformě zvané Golemio, jež je ve správě společnosti Operátor ICT, a. s., která je vlastněna hlavním městem Praha.

Nicméně v době návrhu práce, kvůli právním komplikacím, nebyly k dispozici real-time data z majoritního přepravce na území Prahy Dopravní podnik hl. m. Prahy. Vzhledem k povaze této práce avšak tyto data nenabývají takové důležitosti jako data od přepravců operujících mimo Prahu. Vzhledem k tomu, že zbylí dopravci využívají převážně autobusy k přepravě cestujících, jinými způsoby dopravy se tedy zabívat nebudeme.

Práce se tedy pokusí využít dostupná otevřená data k získání infomarcí o zpoždění spojů na trase. Řešení ovšem není pouze založeno na real-time datach, ale využívá také statická data o jízdních rádech nebo zastávkách hromadné dopravy, ale také mapové podklady.

Uživatelská aplikace

Při dispozici dat o aktuálních polohách vozidel MHD se nabízí jejich využití tak, že budou vynesena do mapy a tím vznikne vizuálně přívětivé uživatelské prostředí pro prohlížení aktuálního stavu sítě vozidel. Proto práce navrhuje a implementuje uživatelskou aplikaci, která tyto vozidla zobrazí a bude interagovat s uživatelem tak, že po na uživatelskou žádost zobrazí additivní infomace o daném spoji nebo vybrané zastávce.

TODO popis aplikace
TODO testování
TODO design

Podle získaných dat se v pracovní den vypravý přibližně (TODO srovnat počet autobusu denne) autobusových spojů.

1. Analýza problému a jeho řešení

V této kapitole je detailně popsán problém a způsoby jeho navrženého a současného řešení.

1.1 Popis problému odhadu zpoždění

Spoje které zajišťují hromadnou dopravu se řídí jízdními řády, které určují jejich trasu a udávají časy příjezdu a odjezdu do daných zastávek. Toto jsou zpravidla jediné referenční body u kterých jsme schopní zjistit skutečné zpoždění, nebo předjetí (dále uvažováno jako zpoždění se zápornou hodnotou).

Vzhledem k tomu, že délka trasy mezi dvěma referenčními body nezříká dosahuje i několika desítek kilometrů (TODO spocitat prumer a median), kde mohou vznikat mimořádné události, ale zpravidla je ovlivněna obvyklým provozem, je potřeba navrhnout systém na odhat zpoždění v půběhu jízdy.

Toto celé je potřeba počítat v reálném čase, tak aby uživatelé byli dobře informováni o stavu jejich spoje. Proto je potřeba zpracovávat data okamžitě po jejich vydání, spočítat odhad zpoždění a vystavit tyto data veřejně. Vzhledem k tomu, že tyto data velmi rychle zastarávají je nutné provést tento proces co možná nejrychleji.

Pro vyjasnění je potřeba uvést, že se systém nesnaží předpovědět zpoždění, které spoj může nabrat vzhledem k dosavadnímu průběhu trasy. Ale snaží se odhadnou zpoždění v daný bod na trase vzhledem k obvyklému profilu jízdy.

TODO obrazek nelinearní trasy

1.1.1 Současná řešení

Takový algoritmus na odhat aktuálního zpoždění mezi dvěma referenčními body již existuje a je zakomponován v systému, ze kterého se čerpají data pro tuto práci. (Detailní popis dat uveden v kapitole ??.) Nicméně nezohledňuje základní parametry průběhu trasy. Tento algoritmus nahlíží na postup vozidla na trase jako na lineární funkci vůči času. Jak ale z praxe víme (TODO doplnit zdroj), rychlosť vozidel není konstantní, neboli doba jízdy není linárně závislá na ujeté vzdálenosti.

1.2 Analýza požadavků na uživatelskou aplikaci

Součástí práce je i vizualizace spočítaných dat. Jinými slovy nástroj umožňuje přístup uživatelů ke spočítaným datům.

Funkční požadavky

- Aplikace vykreslí interaktivní mapu Prahy a širšího okolí, kterou bude možné posunovat či oddalovat. V této mapě budou zobrazeny jednotlivé vozidla na aktuálních pozicích a budou se automaticky posouvat po mapě.
- Po kliknutí na vozidlo se zobrazí jeho celá trasa včetně zastávek a jeho dopočítané zpoždění.
- Celá aplikace bude postavena na principu server – client. Tedy serverová strana se postará o přístup k otevřeným datům o vozidlech a jejich uložení a také obsluhu požadavků klienta. Klientská část bude webová stránka poskytující služby popsané výše. Měla by být schopná zobrazit řádově tisíce vozidel.

Nefunkční požadavky

- Serverová část bude napsaná v jazyce Python 3.
- Webová část bude napsaná pomocí jazyků pro webové technologie, převážně v JavaScriptu.
- Pro vykreslení mapy bude využita služba Mapbox.
- Ukládání dat na serverové straně bude řešeno MySQL databází.
- Pro různé odhady zpoždění na základě historických dat bude využita knihovna scikit-learn.

Proces běhu aplikace

Jak je již zmíněno aplikace bude využívat historická data, tedy bude nutné nechat aplikaci tato data nějakou dobu sbírat. Pro efektivní odhad by bylo vhodné mít uložené historické polohy vozidel alespoň z uplynulých několika týdnů.

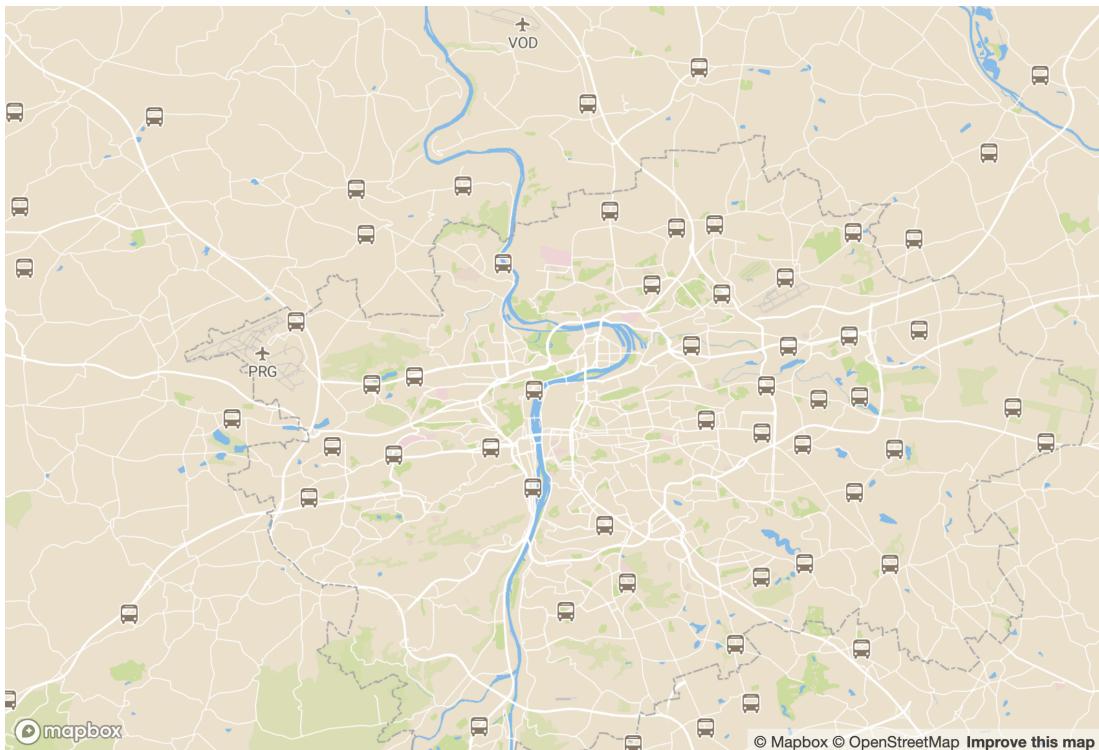
Avšak již v průběhu sběru dat může aplikace poskytovat základní službu a to vizualizování vozidel v mapě.

1.2.1 Poskytovatelé mapových podkladů

K takovému účelu nejlépe poslouží vykreslení aktuálních poloh vozidel do mapy, kde se po vyžádání uživatelem tyto data zobrazí.

Za účelem vytvoření dostatečně přívětivé uživatelské aplikace je nezbytné využít některého z poskytovatelů mapového API, neboli využít již existující mapový podklad a zanést do něj získané informace.

Jedním z těchto poskytovatelů je společnost Google, která má propracované mapové podklady a prostřednictvím služby Google Maps poskytuje pro tuto práci požadovanou službu. Další platformou je Mapbox, který poskytuje velmi podobné služby jako Google Maps. Nicméně natozdíl od Googlu využívá jako mapový podklad OSM otevřená geografické data. Protože smyslem práce je v co největší míře využít otevřená data je žádoucí využít právě Mapbox.



Obrázek 1.1: Mapa z golemio.cz.

TODO dokumentace mapbox, zeptat se jestli je to vubec nutne rozebirat

1.2.2 Současná řešení

Vizualizaci vozidel VHD do mapy již nabízí několik portálů. Všechny jsou však poměrně strohé.

Golemio

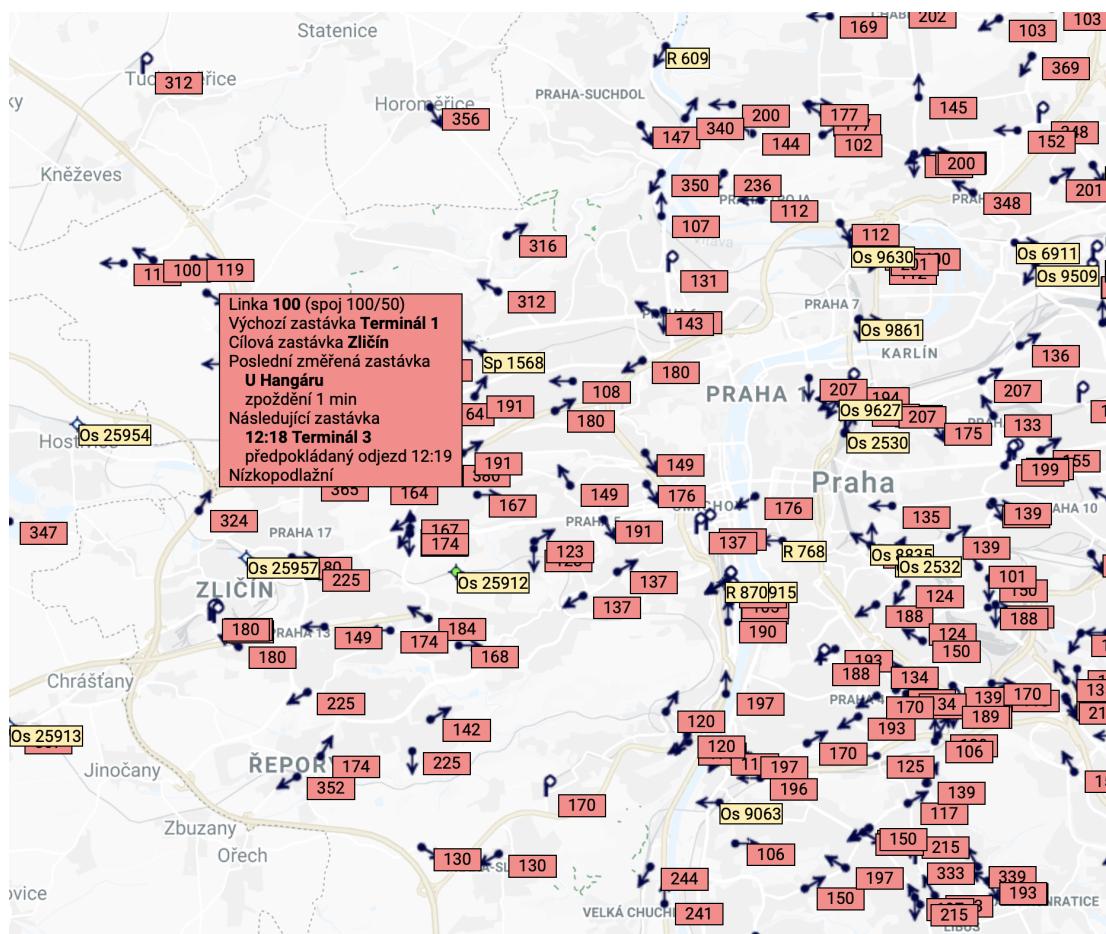
Takovou mapu zobrazuje i samotný provozavatel datové platformy. Nicméně nejsou zde vidět ani čísla linek zobrazených autobusů, natož pak nějaké další informace.

Tram-bus

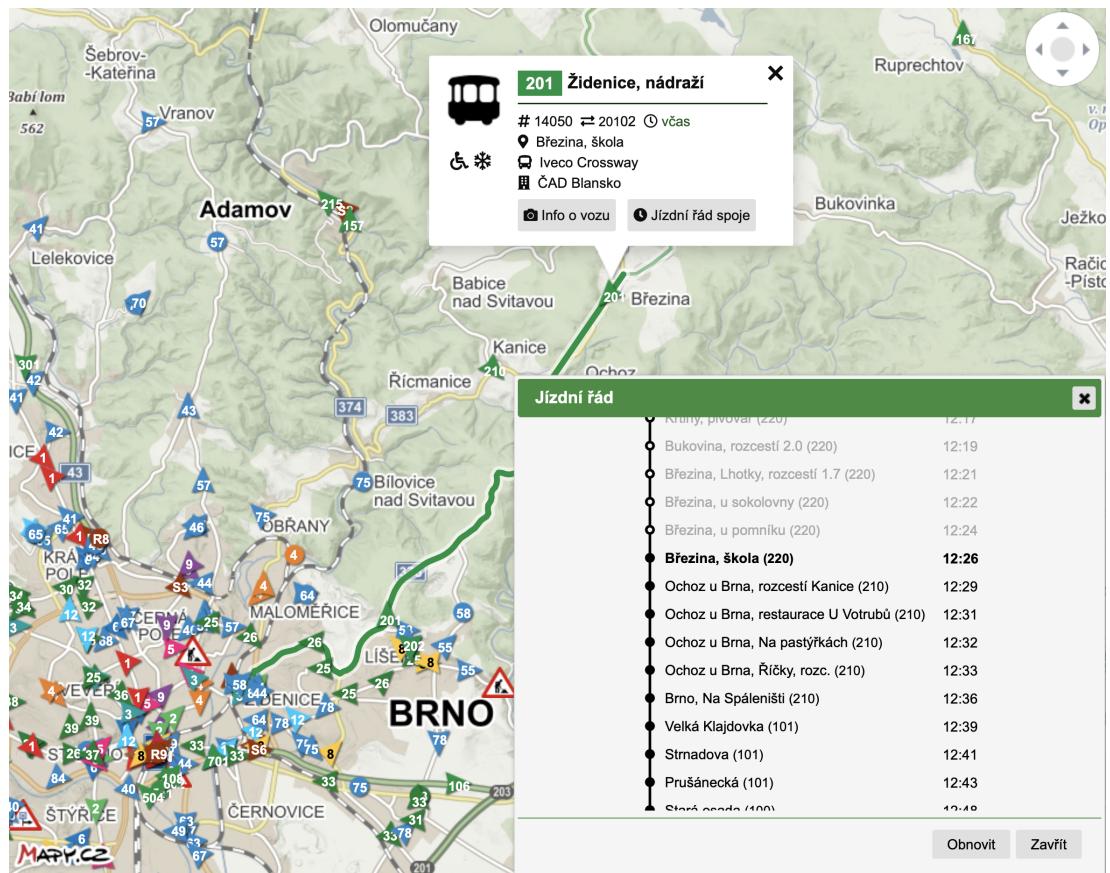
Dalším poskytovatelem je portál tram-bus, který si vede o něco lépe. Ukazuje směr jízdy vozidel, čísla linek a po kliknutí informace o zpoždění a nejbližší zastávky. Pozn.: na mapě již jsou vidět spoje DPP, protože v době psaní této práce již byly data veřejné.

IDSJMK

Mimo Prahu je velice pěkně udělaná aplikace pro zobrazení vozidel IDSJMK (Integrovaný dopravní systém Jihomoravského kraje). Ten ihned po načtení stránky zobrazuje všechny dobravní prostředky, tedy tramvaje, autobusy a vlaky vše s čísly linek. Dále pak umožňuje po kliknutí na vybraný spoj zobrazit více informací včetně jízdního řádu.



Obrázek 1.2: Mapa z www.tram-bus.cz.



Obrázek 1.3: Mapa z mapa.idsjmk.cz.

Tato aplikace je po vizuální i funkční stránce dobrou inspirací pro tvorbu aplikace v této práci.

TODO popis geojson???

2. Analýza zdroje dat

V této kapitole je popsán zdroj real-timových dat o polohách vozidel využívané v této práci.

2.1 Přístup k datům

Na mnohých jednáních s kolegy ze společnosti Operator ICT bylo řečeno, že využívané vozidla vysílají data o své poleze při různých událostech. Zejména pak při brzdění, rozjezdu, ale také, pro účely této práce nejdůležitější, při vyhlášení zastávky, nebo jinak každých 20 sekund.

Taková data pak přímo putují k provozovateli systému na monitorování vozidel, kterým je společnost CHAPS jakožto partner DPP. Ten však tato data zpracovává a posílá ke zveřejnění na platformě Golemio. Bohužel při tomto procesu zpracování se vytratí informace o události v jaké byly data pořízeny. Tedy informace o příjezdu nebo odjezdu ze zastávky jsou zjistitelné pouze z GPS souřadnic.

Po té co jsou tyto data přeneseny do společnosti Operátor ICT by měla být zveřejněna, nicméně data ve výše popsané podobě jsou poměrně chudá, proto je k nim přidáno více atributů. Z pohledu této práce je nejzajímavější informace o vzdálenosti, kterou vozidlo urazdilo od jeho výchozí zastávky. Dále jsou přidána data o jízdních rádech a zastávkách jejichž původcem je ROPID.

2.1.1 Dokumentace

Na úvod je nutné poznamenat, že datová platforma je stále ve vývoji a formát dat se může měnit. S tím mohou přicházet určité výpadky a problémy. K jednomu takovému výpadku došlo i při vývoji této práce, kdy po dobu 14 dnů plafomarma vůbec neodpovídala na dotazy nebo vracela prázdné datasety.

Současně s využívaným datovým formátem, je nasazený pokročilý formát který obsahuje více informací a je přehledněji opraven. Nicméně při zahájení vývoje této práce nebyl k dispozici, proto jsou využívána data pouze ze starší verze.

Oficiální dokumentace datové platformy je poměrně zastaralá sama o sobě, tak že aktuální sada parametrů jí neodpovídá a neobsahuje žádné popisy dat. Proto vysvětlení jednotlivých atributů se zakládá na intuitivním pochopení nebo vyplynulo z jednání se správci platformy. V následujících kapitolách bude popsán formát dat, tak jak přichází od zdroje, a proto se může od oficiálně vystaené dokumentace lišit. A také budou popsány pouze atributy využívané v této práci nebo zajímavé pro její budoucí rozvoj.

TODO reference na dokumentaci

Každá datová sada je exportována ve formátu GEOJSON pokud se jedná o geografická data, nebo jinak ve formátu JSON. A přistupuje se k nim přes jednotné API pomocí HTTP požadavku daného URL adresou a jeho hlavičkou.

TODO reference na specifikace geojson <https://tools.ietf.org/html/rfc7946>

Ačkoli se dokumentace tváří tak, že data jsou exportována ve formátech JSON nebo GEOJSON, většinou formát dat není přesně podle specifikace těchto formátů. Například může být uveden atribut `wheelchair_accessible`, který je typu `bool` a je nastaven na hodnotu `True`, nicméně podle specifikace se tyto hodnoty píší s malým písmenem. Pro tuto práci to sice nepředstavuje komplikaci, protože tento atribut není potřeba, ale mohlo by se stát, že některé parsery JSONu vyhodnotí řetězec jako nevalidní a skončí chybou.

TODO Celá datová platforma Golemio je pojatá jako Open Source projekt.

Pozice vozidel

Jsou nejdůležitější datovou sadou pro tuto práci. Jelikož se jedná o real-time data, data rychle zastarávají a je nutné je velmi často aktualizaovat.

- coordinates aktuální GPS souřadnice vozidla
- origin_timestamp čas zachycení pozice vozidla, v časovém pásmu UTC
- gtfs_trip_id unikátní identifikátor tripu pro spárování s jízdním řádem
- gtfs_shape_dist_traveled vzdálenost vozidla uražená od začátku tripu v metrech
- delay_stop_departure zpoždění zachycené při odjezdu z poslední projeté zastávky v sekundách

Jednotlivé tripy

TODO jak se rekne trip cesky

Dále jsou k dispozici data o každém tripu. To je popis trasy vozidla, včetně zastávek a času příjezdů a odjezdů do/z nich. Také může být vyžádáno k informacím o tripu připojit celý shape trasy, tj. lomená čára kopírující celou trasu daného tripu po povrchu Země.

Míra unikátnosti těchto tripu je předmětem dohadů a zřejmě jsou pod správou plánovačů MHD, nicméně můžeme s určitou mírou spolehlivosti tvrdit, že každý trip se jede nejvýše jednou za den.

- trip_headsign nápis na čele vozidla, typicky cílová stanice
- route_id číslo linky
- trip_id unikátní identifikátor tripu pro spárování s real-time daty, pravděpodobně odpovídá atributu `gtfs_trip_id`

Navíc s každým tripem může být vyžádáno zaslání seznamu zastávek, kterýma projíždí. Po té se obdrží tento seznam s kompletními informacemi o zastávkách, tedy má stejnou informační hodnotu jako samostatný dotaz na zastávky. Navíc je každá zastávka doplněna o informace vázající se k danému tripu.

Zastávky

- arrival_time čas příjezdu spoje do zastávky
- departure_time čas odjezdu spoje do zastávky
- shape_dist_traveled vzdálenost zastávky na trase od výchozího bodu daného tripu v metrech
- stop_id unikátní identifikátor zastávky
- coordinates GPS souřadnice zastávky, často None, je třeba využít atributy `stop_lat` a `stop_lon`
- stop_name název zastávky

TODO Vyzkoušení zjištěno, že shape traveled je po celých 100 metrech.

3. Zpracování dat

Na datové platformě jsou real-time data o vozidlech dostupná do historie řádově jednotek minut, což je naprosto nedostatečné pro jakékoliv pozdější využití. Hlavně pro počítání statistik nad daty je potřeba zřídit lokální databázi, která bude držet historická data, tak jak proudila ze zdroje. Navíc data jsou poskytována ve formátu JSON, který svou povahou není zrovna úsporný co se do velikosti souboru týče. Proto je vhodné zvolit ukládání dat v jiném formátu.

3.1 Databáze

Tato práce využívá relační databázi obsluhovanou dotazovacím jazykem SQL. Tato databáze se skládá z 5 tabulek. Jsou jimi:

- **trips** všechny objevené tripy
 - **id_trip** unikátní identifikátor používaný v databázi
 - **trip_source_id** identifikátor tripu převzatý ze zdroje dat
 - **id_headsign** identifikátor nápisu pro daný trip
 - **current_delay** aktuální zpoždění tripu
 - **shape_dist_traveled** aktuální vzdálenost ujetá od výchozí stanice
 - **last_updated** čas poslední aktualizace, převzatý ze zdroje dat
 - **trip_no** číslo dané linky
- **headsigns** náписy nad vozidlem, cílová stanice
 - **id_headsign** unikátní identifikátor nápisu
 - **headsign** text nápisu
- **trip_coordinates** všechna historická real-time data
 - **id_trip** identifikátor tripu, ke kterému se záznam váže
 - **lat** zeměpisná šířka polohy vozidla
 - **lon** zeměpisná délka polohy vozidla
 - **inserted** čas vložení záznamu
 - **delay** zpoždění zachycené v poslední projeté stanici před pořízením záznamu
 - **shape_dist_traveled** vzdálenost ujetá od výchozí stanice tripu
- **stops** všechny zastávky
 - **id_stop** unikátní identifikátor zastávky
 - **trip_source_id** identifikátor zastávky převzatý ze zdroje dat
 - **parent_id_stop** identifikátor rodičovské zastávky, pokud existuje

- `stop_name` název zastávky
- `lat` zeměpisná šířka polohy zastávky
- `lon` zeměpisná délka polohy zastávky
- `rides` trasa každého tripu, seznam zastávek s časy odjezdů a příjezd tvořící jízdní řád
 - `id_stop` identifikátor tripu
 - `id_stop` identifikátor zastávky
 - `arrival_time` čas příjezdu tripu do zastávky
 - `departure_time` čas odjezdu tripu ze zastávky
 - `shape_dist_traveled` vzdálenost zastávky od výchozí zastávky tripu

Atributy se jménem `*source_id` jsou pravděpodobně unikátní identifikátor entity ve zdroji dat, nicméně z dokumentace zdroje to nevyplývá. Také je toto id textový řetězec a ačkoli je tvořen pouze číslicemi a podtržítky, není nikde zaručeno, že jej lze převést na číselný ko'd. Takže pro lepsí výkon databáze je použito automaticky generované id typu INT.

Každá tabulka má několik indexů, které zlepšují výkon databáze při vkládání a hledání dat. Obzvláště pokud je atribut označen jako unikátní, kde se při každém vložení ověřuje unikátnost.

TODO obrazek na bilem pozadí

Databáza je nastavená tak, aby umožňovala získat všechny potřebné informace o vozidlech, ale hlavně přístup k historickým real-time datům a to separovaně pro dvojici referenčních bodů.

TODO SQL dotazy na sestavení celé databáze jsou definovana v priloze, napsat to jako odstavec nebo jak se toto uvádí?

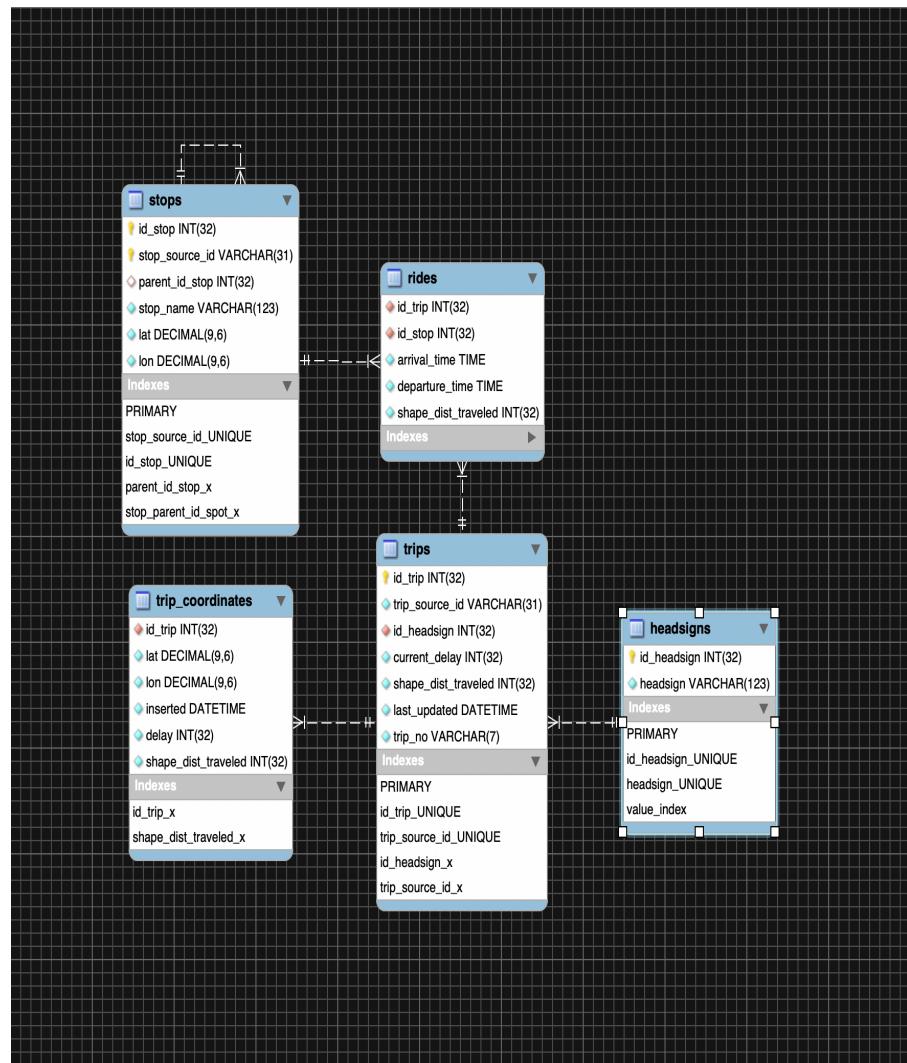
3.1.1 Obsluha databáze

Tato databáze je plněna skriptem jehož algoritmus postupně využívá všechna dostupná data z datové platformy následovně.

Algoritmus:

```

načti všechny dostupné zastávky
dokud skrip běží
    načti aktuální polohy vozidel
    pro každé nalezené vozidlo
        pokud trip vozidla je známí
            aktualizuj data
        jinak
            stáhni informace o tripu
            vlož trip
            zpracuj a vlož jízdu tripu
  
```



Obrázek 3.1: EER diagram databáze.

Protože všechny infomace ukládané do databáze jsou důležité pro hlavní cíl této práce, tak pokud se vyskytne trip, který neobsahuje některou z požadovaných infomarcí je pak automaticky zahozen.

Nejčastěji chybějící atribut je zpoždění v poslední zastávce, toto je nutné vědět pro počítání zpoždění mezi refenrečními body (zastávkami). Absence této informace může být způsobena tím, že vozidlo vůbec nevysílá data potřebná k jejím dopočtení, pak nemá smysl jej do databáze zahrnovat. Nebo vozidlo už vysílá, ale ještě nezahájilo jízdu, tedy nemá žádnou poslední projetou zastávku, v takovém případě budou data ignorována až do doby příchodu první relevantní informace.

Vkládání dat je řešeno pomocí databázových transakcí tak, aby stav databáze byl vždy konzistentní. Tedy pokud nejsou poskytnuta data ve formátu, který skript akceptuje, nebo nějaké povinné atributy chybí. Vložení celého tripu nebude provedeno.

Mimo popsanou databázi se do určeného adresáře ukládají trasy jednotlivých tripů. Tyto data jsou používány jen pro vykreslení na mapu, proto není nutné je držet v hlavní databázi.

Stejně tak i aktuální polohy vozidel jsou mimo databázi zapisovány do souboru, který je určen a formátován pro čtení webovou aplikací. Aktualizace tohoto souboru je provedena přednostně ihned po načtení real-timových dat.

4. Vizualizace dat

Pro vizualizaci dat do webového rozhraní na mapový podklad je využito prostředí Mapbox. K tomu je dále potřeba vytvořit serverovou aplikaci pro komunikaci s webovou stránkou.

4.1 Klientská část

Tato webová stránka je napsána ve standardním HTML s tím, že pro stylování objektů je použit jazyk CSS. Hlavní vlastnosti stránky, jako je zobrazení entit do mapy je použitý jazyk JS, zejména pak jeho možností pro zacházání s DOM elementy. Pro připojení a načítání dat ze serveru se používá technologie AJAXových dotazů.

Koncepce klientské aplikace je taková, že žádná data nezpracovává ani nepře-počítává a zobrazuje jen data taková, která obdržela od serverové strany typicky ve formátu GEOJSON.

Mapbox API

Prostředí Mapbox je široce využívaný multiplatformový nástroj pro zobrazení mapového podkladu a umožňuje do něj zanést širokou škálu různých geometrických útvarů. Takže mapové prostředí intuitivně interaguje s uživatelem a vývojáři mohou využít jednoduchého API pro zobrazení žádoucích dat do mapy.

Webová aplikace této práce využívá naprostoto základní funkcionality, které mapbox umožňuje. Jejich popis včetně načtení do webové stránky za předpokladu, že jsou splněny základní parametry webové stránky je následující.

Rozhraní se do webové stránky importuje pomocí:

```
<script src='https://api.tiles.mapbox.com/
    mapbox-gl-js/v1.4.0/mapbox-gl.js'></script>
<link href='https://api.tiles.mapbox.com/
    mapbox-gl-js/v1.4.0/mapbox-gl.css' rel='stylesheet' />
```

Dále je potřeba vytvořit element s identifikátorem webové stránky, kde bude mapa zobrazena.

Po naimportování je v JavaScriptu k dispozici knihovna jménem `mapboxgl` pomocí, které se ovládá celé mapové prostředí. Pomocí ní je tedy možné vytvořit mapu.

```
var map = new mapboxgl.Map({
    container: 'map', // identifikátor HTML elementu
    style: 'mapbox://styles/mapbox/streets-v11',
    center: [14.42, 50.08], // střed mapy při
        inicializaci [lng, lat]
    zoom: 10 // zoom při inicializaci
});
```

Nyní stačí jen vytvořit HTML element za pomocí JS a po té může být přidám do mapy funkcí:

```
new mapboxgl.Marker(element)
    .setLngLat([Lng, Lat]) // zeměpisná výška a šířka
        umítnění elementu
    .addTo(map);
```

Pro vykreslení složitějších objektů, jako je třeba lomená čára se využívá funkce addLayer.

```
map.addLayer({
    "id": id, // identifikátor vrstvy
    "type": "line", // geometrický útvar k zobrazení
    "source": {
        "type": "geojson", // formát zdrojových dat
        "data": data // zdroj dat
    },
    "paint": {
        "line-color": "#BF93E4", // barva
        "line-width": 5 // šířka
    }
});
```

K manipulaci s objekty typu Layer se používá

```
map.getLayer(id);
map.removeLayer(id);
```

Funkce a design aplikace

4.2 Serverová část

Příchozí požadavky od klineta jsou řešeny skriptem na serverové straně. Který je napojený na databázi a z ní extrahuje potřebná data.

Data jsou posílána v textové podobě ve formátu GEOJSON, který skrip konstruuje z dat získaných z databáze.

Server reaguje na 4 typy požadavků:

- `get_vehicle_positions` vrátí aktuální polohy všech vozidel,
- `get_tail` vrátí lomenou čáru popisující pohyb vozidla v uplynulých n minutách podle id tripu,
- `get_shape` vrátí lomenou čáru popisující trasu spoje podle id spoje,
- `get_stops` vrátí seznam zastávek pro spoj podle jeho id.

Server je naprogramován pomocí Pythoní knihovny `simple_server`, která slouží pouze k debugování, jak se píše v její dokumentaci. (TODO odkaz na dokumentaci). Protože se nepočítá s reálným nasezením této aplikace, není potřeba programovat robustní server. Pro takové užití by bylo třeba přejít na více vláknové řešení.

5. Algoritmus odhadu zpoždění

Tato kapitola popisuje hledání optimálního modelu pro popis pohybu vozidel na trase.

Z toho jak je problém formulován vyplývá, že se má odhadovat zpoždění mezi dvěma referenčními body a jediné takové jsou zastávky na trase daného spoje. Proto cíl algoritmu může být formulován jako vytvořit popis průběh trasy mezi každou dvojcí zastávek, které alespoň jeden spoj obsluhuje a jsou bezprostředně sousedící ve sledu zastávek ve směru jízdy tohoto spoje. Nechť se dvojce bodů a spoj je obsluhující splňující předcházející předpoklad označuje jako A, B a S.

5.1 Základní předpoklady

Hned na začátek je potřeba ustanovit základní předpoklady, ze kterých bude vycházet sestrojený algoritmus.

Zastávky je potřeba rozlišovat na jednotlivá nástupiště. Toto výrazně nezvýší počet dvojic zastávek A a B. – Naprostá většina zastávek má pouze dvě nástupiště. Pro každý směr jedno. Pokud má více nástupišť, pak tak bývá v případech, kdy ze zastávky odjíždí spoje do více směrů a tudíž pro každé nástupiště je jiná následující zastávka.

Všechny spoje S bez ohledu na linku nebo dopravce jedou ze zastávky A do zastávky B po stejně trase a tedy vzdálenost je konstatní. – Předpokládá se, že žádný dopravce neužívá jinou komunikaci a pro všechny platí pravidla silničního provozu stejně. V případě mimořádností se trasa může měnit, nicméně detekce mimořádností a jejich řešení je nad rámec této práce.

Čas jízdy ze zastávky A do B závisí pouze na denní době. – Vysvětleno výše navíc platí žádný z dopravců nedisponeje právem přednosti v jízdě před jiným dopravcem nebo výrazně výkonějším vozidlem. Dojezdové časy mohou být ovlivněny jen charakterem řidiče, avšak toto není zjistitelné z poskytnutých dat a zároveň se předpokládá, že charaktere řidičů jsou rovnoměrně rozloženy mezi všechny dopravce a linky. Podle jízdních rádů některé linky jedou rychleji než jiné, avšak skutečné doba jízdenky je stejná. To že některý spoj zastávku projíždí a tím je rychlejší než jiný spoj není porušení tohoto předpokladu protože se jedná o dvě různé dvojce zastávek.

Během jízdy mohou nastat mimořádnosti, které porušují výše uvedené předpoklady, nicméně detekce mimořádností a jejich řešení je nad rámec této práce a jejich počet je zanedbatelný, proto na statistické modely nebudou mít vliv.

5.1.1 Analýza dat

Z dat popsaných v kapitole (TODO link kap 2???) je potřeba získat všechny dvojce zastávek A, B a všechny označené polohy vozidel mezi nimi. Toto je možné z databáze popsané v kapitole (TOTO link pak 3) zjistit pomocí jízdních

řádů a dále pomocí atributu `dist_shape_traveled` uvedého u každé zastávky pro každý spoj získat i jednotlivé polohy vozidel.

Toto je možné realizovat pomocí následujícího SQL dotazu.

TODO upravit a vylepsit tento dotaz

```
select inn.id_trip,
       inn.id_stop,
       inn.lead_stop,
       departure_time,
       inn.lead_stop_departure_time,
       (inn.lead_stop_shape_dist_traveled - inn.shape_dist_traveled) as diff_shape_
trip_coordinates.inserted,
       (trip_coordinates.shape_dist_traveled - inn.shape_dist_traveled) as shifted_
trip_coordinates.delay
FROM (
    SELECT id_trip, id_stop, shape_dist_traveled, departure_time,
           LEAD(id_stop, 1) OVER (PARTITION BY id_trip ORDER BY shape_dist_trav
           LEAD(shape_dist_traveled, 1) OVER (PARTITION BY id_trip ORDER BY sha
           LEAD(departure_time, 1) OVER (PARTITION BY id_trip ORDER BY shape_di
    FROM rides) as inn
    JOIN trip_coordinates
    ON trip_coordinates.id_trip = inn.id_trip and inn.lead_stop_shape_dist_trave
```

Nyní je pro představení si situace uvedeno několik vizualizací těchto dat. Zejména pak s důrazem na rozlišnost mezi průběhy tras mezi různými dvojicemi zastávek.

Všechny diagramy níže a posléze algoritmy pracují s těmito daty jako body třírozměrného prostoru. Každý tento bod reprezentuje jedno hlášení o poloze vozidla. První dimenze bodu je denní doba (v grafech znázorněno jako počet sekund od půlnoci), druhá dimenze je vzdálenost od výchozí zastávky (znázorněno jako počet metrů) a třetí dimenze reprezentuje čas od vyjetí z výchozí stanice (počet sekund).

TODO obrazky krátka trasa s moc nekonzistentními daty, dlouha trasa hladka, dlouha trasa s anomaliemi, trasy s nedostatkem dat, trasy kde jsou pekne videt krizovatky (cca 5 prikaldů)

5.2 Návrh modelu

Z výše uvedených vizualizací dat vyplývá, že hledání univerzálního modelu popisující všechny možné průběhy tras je nereálné.

Lepší je různé průběhy tras rozdělit podle jejich vlastností do kategorií a pro každou použít jiný model. Použité dílčí modely jsou:

5.2.1 Lineární model

Ačkoli je snaha tento model nahradit lepším, v některých situacích může jeho použití i na dálce dávat smysl. Zejména pak v případech kdy není k dispozici

dostatek dat a nebo je vzdálenost dvou zastávek natolik malá, že nemá smysl ani jakýkoliv odhad zpoždění dělat. (TODO do problemu: Lepší by bylo volit trasy s nejmenší dobou jízdy, ale čas jízdy je promněnlivý a težko se získá skutečná doba jízdy z dat. Navíc v praxi jsou vzdálenost a doba jízdy dostatečně závislé)

(TODO obrázek lineárního modelu)

5.2.2 Polynomiální model

Polynomiální model se hodí pro situace kdy je průběh trasy nějak ovlivněn vždy ve stejném úseku a má vliv na každý projíždějící spoj. Nebo se v průběhu dne pozvolna mění v závislosti na dopravním vytížení projížděných úseků.

K tomuto dochází například v případech kdy spoj zastavuje ve městě a v následujících několika málo kilometrech jede pomaleji, poté zrychlý a dále opět vjede do města. Takový model se hodí spíše na delší trasy s plynulou jízdou.

Pro spočítání polynomiálního modelu se využívá knihovna sklearn konkrétně algoritmus zvaný Rigde, který sám o sobě hledá linární závislosti, nicméně vstupní hodnoty jsou mezi sebou náležitě pronásobeny tak, aby simulovali polynomiální funkci. Toho se dosáhne pomocí funkce PolynomialFeatures. Optimální stupeň polynomu se zjistí spočítáním modelu pro každý stupeň v rozumných mezích a nakonec se zvolí ten s nejmenší chybou. (TODO jak moc detailně se má toto popisovat?)

(TODO obrazek poly modelu)

5.2.3 Model pomocí konkávního obalu

Posledním a nejkomplikovanějším modelem, zasluhující nejvíce pozornosti, je popsání trasy za pomocí konkávního obalu bodů.

Tato metoda vznikla jako řešení situace, kdy na trase exituje bod, který určité procento projíždějících spojů zdrží o netriviální dobu. Něco takového nastane pokud spoje projízdí světelnou křížovatkou nebo místem přek kterým se tvoří kolona vozidel. Zde dochází ke skové změně průběhu bodové funkce a spojité modely by s okolím tohoto kritického místa měly problém. Pro jeden takový bod by možná šlo navrhnout jednoduší řešení, nicméně je potřeba algoritmus, který umí pracovat s více kritickými body na trase. Například je nutné poradit si se situací zobrazené na diagramu (TODO link na obrazek), kde jsou na trase takové kritické body dva. V této modelové situaci je pro jednoduchost uvažováno, že se většina projíždějících spojů zdrží v prvním kritickém bodě, nebo v druhém, nebo se lehce zdrží v obou. S takovým zdržením je počítáno v jízdním rádu a tedy vozidla, která projedou první kritický bod bez zdržení jedou na čas stejně tak, jako vozidla v něm zdržená. O snížení nebo zvýšení případného zpoždění spoje je možno rozhodnout až po projetí druhého bodu. Jinými slovy na ose uplynulého času od vyjetí ze zastávky vzniká jakýsi podprostor v němž se zpoždění nemění. Pro jeho popis se využívá právě konkávní obal všech spojů, které přijely včas.

Problémy

Ačkoli je myšlenka jednoduchá, při implementaci vyvstává několik technických problémů.

Nejprve k samotnému konkávnímu obalu. Je potřeba říct, že na množině bodů není definován jednoznačně (TODO obrázek nejednoznačnosti konkavnoho obalu). A jedná se o početně složitý algoritmus. V tomto díle je zapotřebí spočítat obal ve třídimenzionálním prostoru, což je velmi komplikovaný úkol, a proto je potřeba přijít s zjednodušením úlohy. Tedy počítat obal pouze pro dvoudimenzionální prostor. Toho se nedá dosáhnout jinak než diskretizací úlohy a počítání obalu pro každou hodinu zvláště, tedy ze všech bodů, které byly zaznamenány v průběhu jedné hodiny. Tím může dojít k větší granularitě obalu, než by bylo vhodné, nicméně předpokládá se, že hodina je dostatečně dlouhý časový interval na to, aby zde byly zachyceny všechny druhy průběhu jízdy (započítávají se všechny spoje jedoucí ve stejnou hodiny nejméně týden zpět) a zároveň je to dostatečně krátký interval na nezkreslování denních výkyvů v čase jízdy. Navíc všechny body tvořící obal pro danou hodinu jsou poté přidány do celkového obalu i s přesným časem zaznamenání a pro počítání konečných výsledků je použit třídimenzionální obal.

Dále se jako netriviální ukazuje detekce spojů, které přijely včas a tedy všechny jejich body mají být předány k výpočtu obalu. Nabízí se použít data o všech spojích, které přijely do cílové zastávky s co nejmenším zpozděním, ale je nutné mít na paměti, že příjezdy podle jízdního řádu nemusí vůbec odpovídat realitě. Proto se zdá být nejlepší použít data od spojů, které přijely ve stejnou dobu jako je průměr všech příjezdů do cílové zastávky. Toho se docílí tak, že se poslední body podle vzdálenosti všech spojů použijí pro odhad času příjezdu. Zobrazeno na diagramu (TODO link na obrázek odhad příjezdu). Dále se pro každou hodinu seřadí všechny spoje podle vzdálenosti odhadnutého příjezdu od skutečného a použije se určité procento nejbližších spojů k tomuto odhadu.

Z předchozího popisu řešení vyplývá ovšem, že pro výpočet obalu jsou použity spoje, které ani zdaleka nemusely přijet včas jak je požadováno, ale předpokládá se, že se nepříliš vzdalují od průměrného času příjezdu. To že střední zpozdění pro celý obal není nulové se vyřeší spočtením odchylky průměrného příjezdu od příjezdu podle jízdního řádu a následně přičtení této konstanty k odhadnutnému zpozdění. Každopádně to, že rozptyl příjezdů spojů zahrnutých ve výpočtu obalu může být netriviální, vyžaduje nahlížet na tento obal jako na lineární prostor pohybu zpozdění. Tedy že odhad zpozdění pro bod nacházející se v obalu je lineárně závislý na vzdálenosti od hranice obalu, avšak protože je známo časové rozpětí příjezdu spojů použitých pro výpočet obalu, je možné tuto vzdálenost snadno přenést na skutečné pozdění.

Popis algoritmu

Po popsání a vyřešení všech komplikací je možné nastínit průběh algoritmu.

Nejprve konstrukce konkávního obalu:

poslední_čas = vyber všechny poslední body podle vzdálenosti od každého spoje
odhad_příjezdu = odhadni čas příjezdu v průběhu celého dne podle bodů v poslní_čas
spoje_včas = prázdné pole

pro každou hodinu h:

spoje_včas += vyber x \% spojů, které přijely nejblíže odhadu v hodině h

konkávní_obal = prázdné pole

pro každou hodinu h:

body = vyber všechny body zaznamenané v hodině h a náležící kterémukoli spoju
konkávní obal += spočítej konkávní obal z body

Vrací: konkávní_obal

Dále odhad zpoždění z konkávního obalu:

Vstup: bod v prostoru vzdálenosti, průběhu dne a času na trase

pokud je bod v konkávní_obal:

velikost_okna = rozdíl horní hranice obalu od spodní v čase příjezdu do zastávky
spodek_okna = spodní hranice okna v čase příjezdu do zastávky
poměr = vzdálenost bodu od spodní hranice obalu / vzdálenost bodu od horní hranice obalu
odhad_příjezdu = velikost_okna * poměr + spodek_okna

jinak:

pokud je bod pod obalem:

spodek_okna = spodní hranice obalu v čase příjezdu do zastávky
odhad_příjezdu = spodek_okna - vzdálenost bodu od obalu

jinak:

vrch_okna = horní hranice obalu v čase příjezdu do zastávky
odhad_příjezdu = vrch_okna + vzdálenost bodu od obalu

Vrací: odhad_příjezdu - pravidelný příjezd

f

Závěr

Seznam použité literatury

Seznam obrázků

1.1	Mapa z golemio.cz.	5
1.2	Mapa z www.tram-bus.cz.	6
1.3	Mapa z mapa.idsjmk.cz.	7
3.1	EER diagram databáze.	13

Seznam tabulek

Seznam použitých zkratek

Slovník

AJAX Asynchronous JavaScript and XML. 15

API rozhraní pro programování aplikací. 4, 9, 15

CHAPS CHAPS s. r. o.. 8

CSS Cascading Style Sheets. 15

DOM Document Object Model. 15

DPP Dopravní podnik hlavního města Prahy, a.s. 5, 8

GEOJSON standardní formát navržený pro reprezentaci jednoduchých prostorových geografických dat. 9, 15, 16

GPS Global Position System. 8, 9, 10

HTML Hypertext Markup Language. 15, 16

HTTP HyperText Transfer Protocol. 9

IDSJMK Integrovaný dopravní systém Jihomoravského kraje. 5

INT Celé číslo. 12

JS JavaScript. 15, 16

JSON JavaScript Object Notation. 9, 11

MHD městská hromadná doprava. 2, 9

OSM OpenStreetMap. 4

ROPID Regionální organizátor pražské integrované dopravy, p. o.. 2, 8

SQL Structured Query Language. 11, 18

URL Unique Resource Link. 9

UTC Koordinovaný světový čas. 9

VHD Veřejná hromadná doprava. 5

A. Přílohy

A.1 První příloha