VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH ZPŮSOBU ŘÍZENÍ DOPRAVY V CHYTRÝCH MĚSTECH

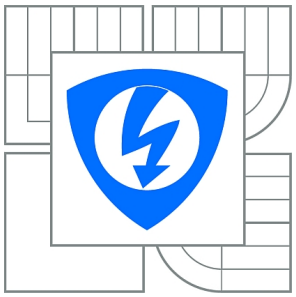
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JIŘÍ SCHIMMER
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH ZPŮSOBU ŘÍZENÍ DOPRAVY V CHYTRÝCH MĚSTECH

PROPOSAL OF TRAFFIC MANAGEMENT IN SMART CITIES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

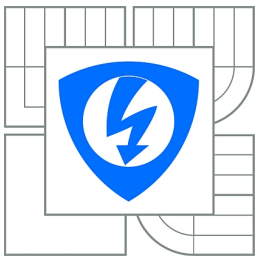
AUTOR PRÁCE — JIŘÍ SCHIMMER

AUTHOR

VEDOUCÍ PRÁCE — Ing. RADEK FUJDIAK

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Jiří Schimmer ***ID:*** 134400
Ročník: 3 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Návrh způsobu řízení dopravy v chytrých městech

POKYNY PRO VYPRACOVÁNÍ:

Úkolem studenta je zorientovat se v možnostech simulací reálných objektů a prostředích, dále pak zhodnotit dnešní použitelné algoritmy pro simulace dopravy a na základě poznatků vybrat jeden konkrétní algoritmus pro simulaci dopravy. Dále bude úkolem studenta vybraný algoritmus implementovat a vytvořit tak simulační prostředí. Na základě reálných dat bude od-simulována doprava ve vybraném městě (či dopravně vytížené městské části) a z výsledků této simulace (a poznatků z teoretických zdrojů) bude výstupem studenta návrh optimalizace pro chytré město. Tento návrh pak student následně ověří v simulačním prostředí a porovná se stávajícím řešením ve vybrané oblasti (návrh by měl být postaven na senzorických sítích, jakožto základu chytrých měst a podložen teoretickými předpoklady a podklady).

DOPORUČENÁ LITERATURA:

[1] P. PERINGER. Modelování a Simulace. Brno, 2013. VUT Brno.
[2] T. TICHÝ, V. FALTUS a M. SPĚVÁČEK. Úvod do inteligentních dopravních systémů: Řízení dopravy ve městě. Praha 1, 2011. ČVUT.
[3] I. KŘIVÝ a E. KINDLER. Simulace a Modelování. Ostrava, 2001. Ostravská Univerzita.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 2.6.2015

Vedoucí práce: Ing. Radek Fujdiak
Konzultanti bakalářské práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

ABSTRAKT

Práce se zabývá prvky telemetrie v dopravě, funkcí těchto prvků, jejich efektivitou a možnostmi vylepšení. Práce také zahrnuje vytvořené simulační prostředí čtyř křižovatek s nasazeným responzivním Dijkstrovým algoritmem pro vyhledání nejkratší trasy. V simulačním prostředí jsou nastaveny statické hodnoty intervalů jednotlivých semaforů. Následně je implementován genetický algoritmus, který v rámci optimalizace vyhledává vhodnější nastavení semaforů pro daný počet vozidel.

KLÍČOVÁ SLOVA

Telemetrie v dopravě, řízení dopravy, simulace dopravy, genetický algoritmus, Dijkstrův algoritmus.

ABSTRACT

Thesis deals with parts of telemetry in traffic, functionality of these parts and their effectiveness and improvement. Thesis also includes simulation environment of four crossroads and implemented responsive Dijkstra´s algorithm for searching the shortest path. In simulation environment are setup static values of the intervals on traffic lights. There is also implemented genetic algorithm, which is in part of optimization looking for more accurate timing solution for different amount of cars.

KEYWORDS

Telemetry in traffic, traffic control, traffic simulation, genetic algorithm, Dijkstra´s algorithm.

SCHIMMER, J. *Návrh způsobu řízení dopravy v chytrých městech.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 62 s. Vedoucí bakalářské práce Ing. Radek Fujdiak.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma Návrh způsobu řízení dopravy v chytrých městech jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a~jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce ing. Radku Fujdiakovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce. Dále bych také rád poděkoval panu doc. Ing. Radimu Burgetovi, Ph.D. za konzultaci při implementaci genetického algoritmu.

V Brně dne ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Místo ....

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ALGORITMŮ

ÚVOD

V dnešní těžké ekonomické situaci světa se společnost i samosprávy měst dostávají k otázkám řešení nákladů v různých odvětvích. Zejména z ekonomických důvodů je čím dál více aktuální téma „inteligentních“ měst a jejich řízení, jako například řízení dopravy a řešení veřejného osvětlení [1]. Vytvoření modelu inteligentního města by mělo nabídnout snížení spotřeby elektrické energie, dále pak snížit emise CO_2, zrychlit dopravu, zajistit bezpečnost občanů za pomoci kamerových systémů nebo zajistit dostatek parkovacích míst [2][3].

Řešení se nabízejí například technologická a systémová. Technologická spočívají ve zlepšení stávající technologie nebo v nahrazení zastaralé technologie vývojem nové technologie, vytvořením propracované dopravní infrastruktury za pomocí inteligentních semaforů, atd. Systémová řešení se zakládají na optimalizaci již instalovaných zařízení. Výhodou některých systémů je schopnost přejít na nová technologická či systémová řešení plynule v průběhu delšího časového horizontu a nezatížit tak rozpočet města [4].

Jako první je v této práci obsazen teoretický úvod do způsobů řízení města a jejich technologická řešení. V další části je nastíněna problematika simulací dopravy, výběr algoritmu pro nalezení nejkratší cesty v simulačním prostředí a nakonec je rozebrána problematika v oblasti optimalizace dopravy a výběr konkrétního řešení optimalizace pro vytvořené simulační prostředí.

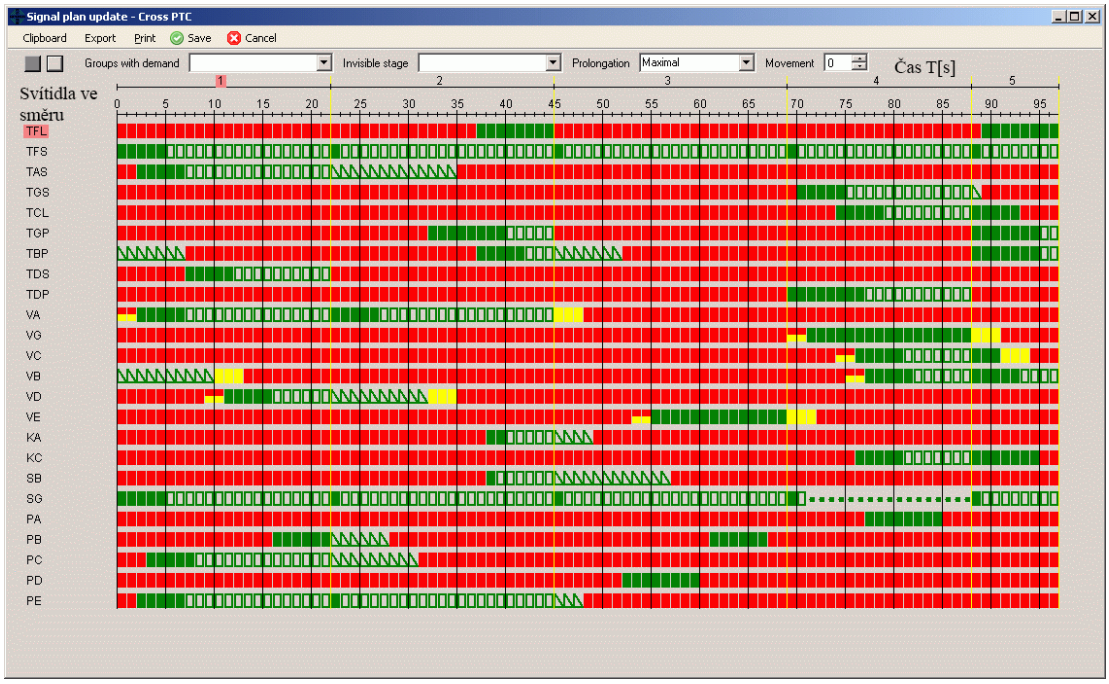
Z vytvořeného statického prostředí jsou zpracovány statistiky v závislosti na počtu vozidel projíždějících v simulačním prostředí. Dále je implementována optimalizace v podobě genetického algoritmu a z této optimalizační části jsou analyzovány výsledky optimalizace a porovnány s výsledky ze statického řešení simulované dopravní infrastruktury.

1 ŘÍZENÍ DOPRAVY

V této kapitole bude proveden rozbor oblasti řízení dopravy, kde bude vysvětlena důležitost tohoto odvětví. Budou shrnuty klíčové prvky v řízení dopravy, jejich využití v praxi a možnost jejich využití pro simulaci.

1.1 Inteligentní semafory

Inteligentní semafory se začaly instalovat kvůli složitosti dopravní situace a většímu důrazu na plynulost provozu. Díky vyšším požadavkům v dopravě došlo i k propojení jednotlivých inteligentních světelných signalizačních zařízení (dále jen „SSZ“) mezi sebou ve velíně. Tam jsou veškerá data přenášena, aby byla dále vyhodnocena operátorem a zohledněna v přehledu dopravy ve městě. Jedná se o data vztahující se k aktuálnímu stavu SSZ, signalizačních světlech v jednotlivých pruzích, informace o prodloužení časového intervalu daného pruhu pro volný průjezd vozidla a informace o případném vadném stavu sodíkové žárovky či LED diody dle použití v konkrétních řešeních SSZ [5]. Na obrázku Obr. 1 se nachází zpracovaná přijatá data do podoby signálního plánu, který vychází z dopravního plánu dané křižovatky. Vodorovná osa je čas v sekundách a na ose svislé se nachází jednotlivá návěstidla. Jsou zde zahrnuty chodecké SSZ a dopravní SSZ. První písmeno v označení je typ SSZ (přechodový či pro vozidla) a druhé písmeno je, ve kterém směru se návěstidlo nachází.



Obr. 1: Signální plán (převzato z [16])

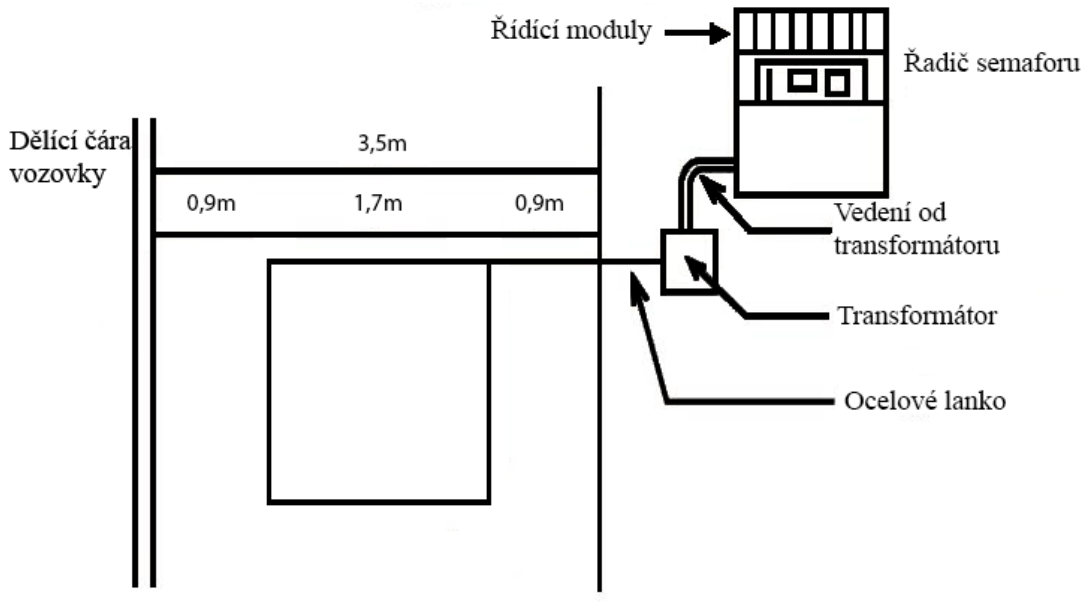
Inteligentní SSZ mají různé podoby, vždy ale obsahují základní prvky, a to řadič, vyhodnocovací jednotku obsahující program, který je navržen individuálně pro každou křižovatku dle požadavků konkrétní lokality v závislosti na dopravě. K inteligentnímu SSZ je vždy připojeno zařízení detekující pohyb vozidel či stav dopravy v určitém místě či lokalitě v okolí SSZ. Řadič vyhodnocuje získané informace z těchto detekčních prvků. Může jít například o nejčastěji využívané indukční smyčky či o méně využívané videokamery, které fungují na principu detekce pohybu v určitém místě záběru dané lokality před SSZ. Systém funguje na základě vyhodnocování dat z těchto snímačů a následnou úpravou signálního plánu v podobě prodloužení zelené v daném směru. Nebo také může vynechat zelenou pro pruhy, ve kterých není detekováno motorové vozidlo [6].

1.2 Detekční prvky

1.2.1 Indukční smyčky

Indukční smyčka využívá principu kondenzátoru k detekci motorových vozidel, motocyklů a jízdních kol, princip jejího zapojení je znázorněn na obrázku Obr. 2.

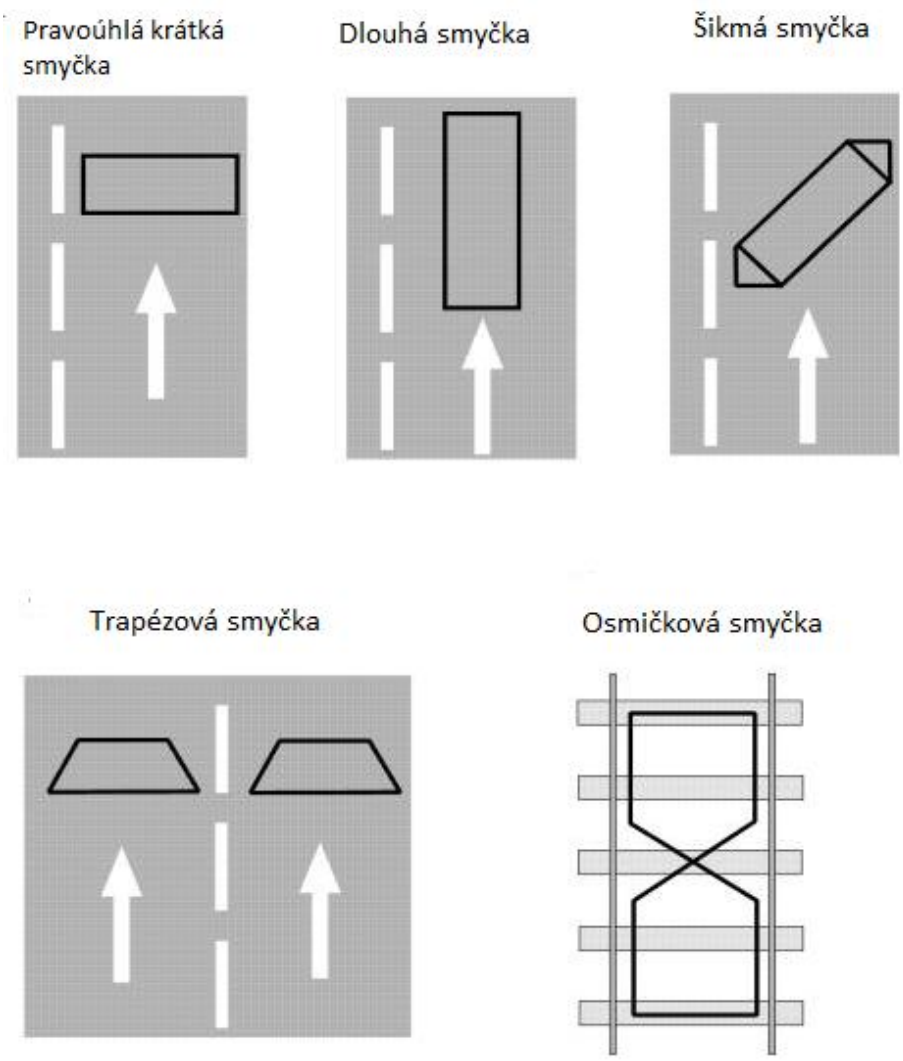
Jako materiál využívaný na výrobu indukčních smyček může být ocelové lanko s izolací odolnou natolik, aby byla schopna fungovat i po jejím zalití pružnou asfaltovou směsí a aby vydržela podmínky vnějších vlivů, jako jsou například prohlubně vzniklé působením tíhy vozidel na vozovku (vyjetí kolejí). Smyčka je připojena k transformátoru, který na základě změny magnetické indukce přivádí rozdílné napětí na primární vinutí a tím se mění i napětí na výstupu transformátoru [7].



Obr. 2: Zapojení indukční smyčky

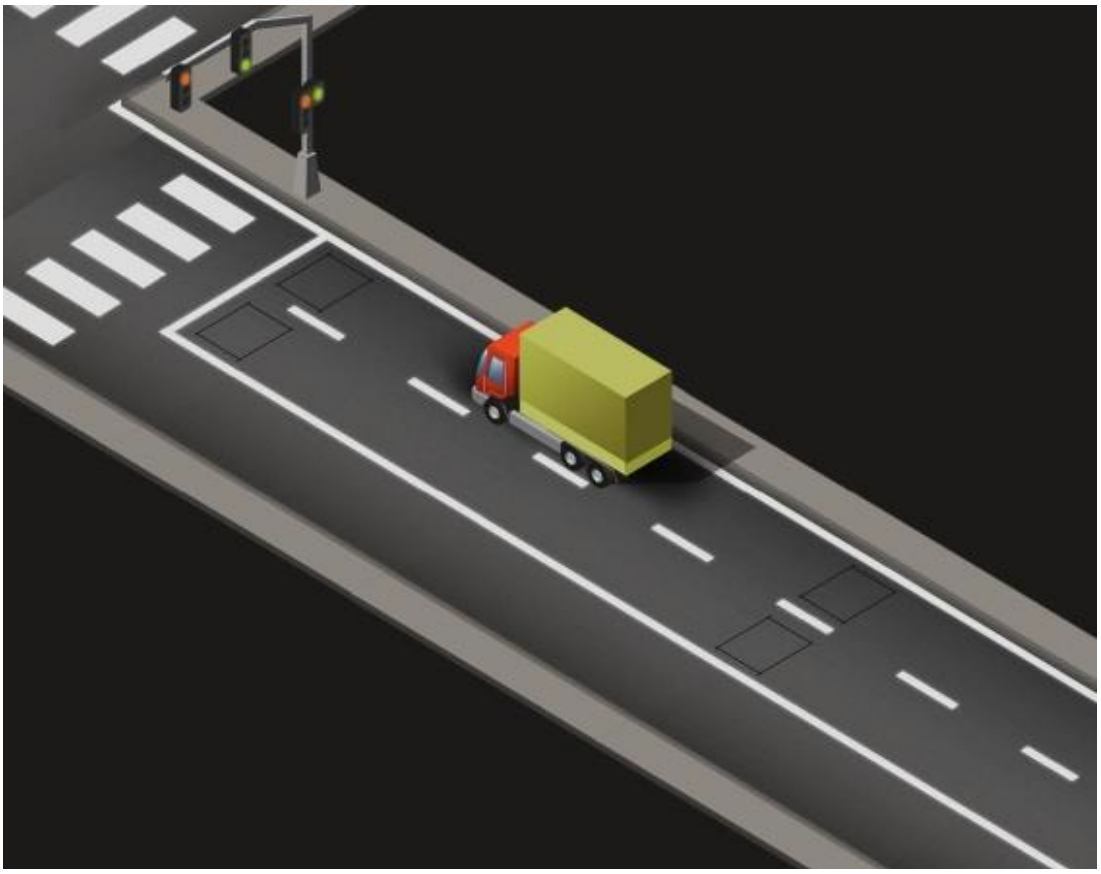
Transformátor posílá tuto změnu napětí do řadiče, který umí rozpoznat a porovnat hodnoty napětí v klidu a při detekci vozidla. Řadič musí být schopen rozpoznat chybu od neustálé detekce vozidla na dané smyčce [8].

Smyčky se používají, buď k detekci stojícího vozidla, aby tímto vygenerovaly výzvu nebo k detekování projíždějícího vozidla, aby prodloužil interval zelené k plynulému průjezdu vozidla křižovatkou. S tímto ohledem se rozlišuje umístění smyček. Smyčky, které se používají k detekci stojícího vozidla, by v ideálním případě měly být umístěny pod úhlem 45 ° z důvodu přesnější detekce motocyklů a kol viz Obr. 3.



Obr. 3: Druhy umístění indukčních smyček

Na rozdíl od toho smyčky využívané k detekci projíždějících vozidel by měly být umístěny v rozmezí 30 - 50 metrů od SSZ či hranice křižovatky viz. Obr. 4.



Obr. 4: Vzdálenost indukčních smyček (převzato z [6])

Důvod pro zvolení této vzdálenosti je prostý. Semafor musí mít dostatečný čas, aby byl schopen vyhodnotit danou situaci a nastavil prioritu pro průjezd detekovaného vozidla. Toto platí za předpokladu, že z jiného směru byla zelená a musela být změněna na červenou tak, aby byl vozidlu umožněn průjezd.

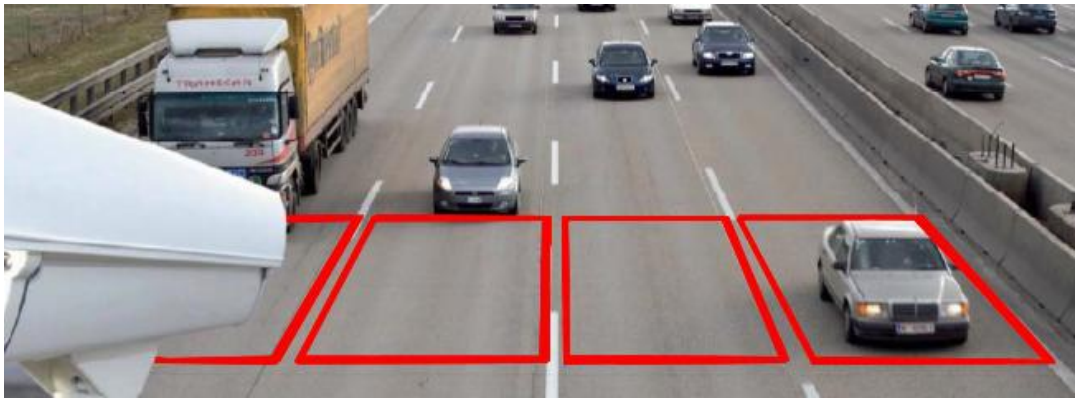
Indukční smyčky se dají využít i pro změření rychlosti vozidla mezi dvěma smyčkami. Jednoduše se dá rychlost spočítat díky známé vzdálenosti mezi indukčními smyčkami a časovému úseku vozidla na jednotlivých smyčkách. Toto měření je velice přesné. Indukční smyčky reprezentují v logické struktuře programu podmínku if. To znamená, že v simulaci můžeme považovat za přesné jakékoli najetí vozidla do oblasti před SSZ. V praxi nemusíme znát přesný počet vozidel, ale vzhledem ke vzdálenosti, jak jsou indukční smyčky umístěny a vzhledem k jejich vzdálenosti od sebe lze předpokládat, že počet vozidel při simulaci i v praxi je známý [9].

Pokud by po stránce legislativní došlo k zohlednění a využívání těchto prvků již instalovaných ve městech, dala by se tímto krokem detekovat rychlost vozidel buď z důvodu průzkumu a statistiky, či k zasílání informací pro Policii ČR. K tomuto účelu by muselo dojít k vyškolení pracovníků a certifikaci jednotlivých SSZ pro použití informací o rychlostech vozidel pro Policii ČR [10].

1.2.2 Kamery s detekcí dopravy

Kamerové systémy na detekci dopravy nejsou až tolik využívané díky dílčím nepřesnostem oproti přesným indukčním smyčkám. Detekční zóna snímání kamer je vidět na obrázku Obr. 5. Díky občasnému složitému umístění křižovatky, například na mostě či v jiné lokalitě neumožňující využití indukčních smyček, se jako vhodnější jeví použití kamer na detekci dopravy.

Kamery mají vybrané body v záběru, které při detekci změny oproti výchozímu stavu pošlou vyhodnocenou informaci ze záběru do řadiče semaforu a ten už poté pracuje s danou informací. Kamery jsou mnohem více vystavované vlivům, například počasí. Díky tomu se zvyšuje jejich opotřebení a tím se snižuje jejich spolehlivost a přesnost. Snížená přesnost je hlavně díky vlivům počasí a neschopnosti kamery fungovat tak, jako přesnější indukční smyčka. Jedná se o vlivy jako například mlha, husté sněžení či déšť. V noci s kamerami není až takový problém díky infračervenému přisvícení vozovky, které zajistí, že kamera zaznamená motorové vozidlo tak, jak má. Také zde může nastat chyba detekce v kombinaci s některým z nepříznivých vlivů počasí [11].



Obr. 5: Detekce kamer (převzato z [11])

1.3 Řadič SSZ

Řadič je zařízení, které podle programu vyhodnocuje informace z detekčních prvků popsaných výše. Řadič musí být schopen detekovat vadnou svítilnu z důvodu bezpečnosti silničního provozu. Musí fungovat dle zákonů o bezpečnosti provozu na pozemních komunikacích a norem v dopravě především z bezpečnostních důvodů. Pokud tedy nastane situace, kdy nesvítí červená barva (například z důvodu vadné žárovky, nebo LED svítilny) na návěstidle, SSZ musí přestat fungovat, křižovatka se přepne do blikající žluté a řidiči se musí řídit svislým dopravním značením. Dále musí být moderní řadič schopen dynamicky pracovat s časovými intervaly. Pokud například zaznamená vozidlo blížící se ke křižovatce z volného směru, měl by SSZ prodloužit zelenou o takový časový interval, aby vozidlo bylo schopno projet a současně bezpečně opustit křižovatku. Také časy na průjezd jednotlivých pruhů by měly být nastaveny optimálně. Čas nastavený na průjezd křižovatkou by měl být ideálně nastavený na kratší časový interval, aby projela vozidla co nejrychleji a nečekala dlouho před SSZ. Tato doba však nesmí být krátká natolik, aby časy na průjezd vozidel nebyly stejně dlouhé či o málo delší, než doba určená na vyklizení křižovatky. V tomto případě by byl SSZ

velmi neefektivní [12].

Proto je nejlepší nastavit SSZ na kratší dobu průjezdu s tím, že řadič musí být schopen potřebný časový interval prodloužit pro plynulý průjezd dalšího vozidla. Tímto dochází k šetření paliva na rozjezdy vozidel a tím i ke snížení emisí ve městě. Řadič musí být také schopen rozpoznat rozdíl mezi vadnou smyčkou a tím, když je v místě umístění smyčky vozidlo z neurčitého důvodu (havárie, porucha vozidla). Na tento fakt může řadič upozornit velín a poslat mu varování. Varování může po určitém časovém intervalu přejít až do stavu poruchy [13].

Řadič může komunikovat s velínem přes GSM modul. Odesílá aktuální informace o stavu barev v jednotlivých pruzích či informace o stavu SSZ. Řadič musí mít také záložní zdroj, který ho udrží „naživu“ po dobu nezbytně nutnou na vyslání zprávy o přerušení napájení a poté se sám automaticky vypne. Dále může být řadič vybaven GPS zařízením, které může sloužit k synchronizaci času. To je zde z důvodu automatického stmívání světel po setmění. Obvykle je intenzita snížena o 30 % svítivosti LED svítilny. Je tak činěno z důvodu zabránění oslňování řidičů jasnými LED diodovými světly v SSZ a zároveň se sníží náklady na provoz křižovatky a prodlouží se i životnost svítidla [14].

1.4 Preference v dopravě

Preference v dopravě se využívá ve dvou odvětvích. První je priorita pro městskou hromadnou dopravu (dále jen „MHD“) a druhá je priorita pro vozy hasičského záchranného sboru.

1.4.1 Preference MHD

Systém preference MHD je v dnešní době velmi oblíbený zejména v západních zemích. Jedná se o systém preference pro vozidla MHD (autobusy, trolejbusy a tramvaje). Funkce systému spočívá v detekci vozidel MHD a poté jejich prioritním odbavení na křižovatkách se SSZ. Efektivita tohoto systému je velmi debatována. Faktem je, že se ušetří jen zlomek z pořizovacích nákladů [15]. Systém pouze zefektivní časové rozložení zastávek vozidel MHD. Tím je myšleno, že doba, která je strávená na křižovatce, není využita efektivně, ale kdyby byla tato doba strávená v zastávce místo na křižovatce, byl by tento čas využitý k případnému nástupu dalších cestujících. Systém má také za úkol zajistit včasný či ideálně předčasný příjezd MHD do zastávky, kde poté MHD vyčká, dokud nemá podle jízdního řádu své linky odjet. Další z účelů tohoto systému je ukázat občanům, že systém MHD je spolehlivější, pohodlnější a levnější, než jezdit po městě vlastním motorovým vozidlem [16].

V některých německých městech dělají maximum pro to, aby lidé jezdili MHD tím, že nasazují emisní daně či navýší parkovné natolik, aby občany přinutili využívat městskou hromadnou dopravu [17]. V nizozemském Amsterdamu tento problém vyřešili pomocí jízdních kol a hromadné dopravy. Dále také Londýn zavedl městské mýto podle toho, jak moc do centra města chcete jet, o to více zaplatíte [18]. Všechna uvedená města se snaží svým občanů ukázat, že tento systém cestování po městě je bezproblémový, pohodlný, lidé se nemusí stresovat a zároveň tím, že se nejezdí tolik motorovými vozidly se šetří ovzduší ve městě a jeho okolí [19].

1.4.2 Preference vozidel hasičského záchranného sboru

Jedná se o prioritu pro vozidla složek integrovaného záchranného systému a konkrétně hasičského záchranného sboru (dále jen „HZS“). Tato vozidla mají díky své velikosti předem určené trasy, po kterých musí jezdit. Toho se dá využít a pokud zadá operátor trasu do systému dopravy, měl by systém vždy při detekci vozidla HZS sepnout zelenou barvu ve směru jízdy HZS a umožnit tak plynulý průjezd křižovatkami. U ostatních složek integrovaného záchranného systému, jako je policie a záchranná služba, je tento systém jen velmi těžko realizovatelný, neboť vozidla HZS mají na rozdíl od vozidel policie a záchranné služby předem vytyčené trasy, po kterých mohou jezdit. Díky tomu lze snadno zařídit průjezd vozidel HZS.

Na rozdíl od toho vozidla policie a zdravotnické záchranné služby mohou jet do jakékoli lokality ve městě a jediné řešení těchto situací je detektor přímo v křižovatce. Například vozidlo policie má vysílač, který funguje pouze, když má zapnuté výstražné majáky pro právo přednosti v jízdě. Křižovatka má vlastní přijímač, který detekuje, ze kterého směru vozidlo přijíždí a podle toho prioritně pustí zelenou barvu ve směru jízdy vozidla policie, které projede křižovatkou kam bude potřebovat. Toto je opět obtížnější u křižovatek, které jsou dopravně složitější a mají možnost odbočení do všech směrů. Řadič nemůže pustit všechny směry do celo zelené, aby vozidlo policie projelo. Také je třeba vzít v úvahu možnou kolizi v prioritě. V případě příjezdu sanitky z jednoho a policie z druhého směru křižovatky. Pro každou organizaci by musela být jasně stanovena priorita průjezdu [20].

1.4.3 Funkce systému priority vzhledem k zařízení a jeho instalaci

Jedná se o dvě varianty funkčnosti systému. V první variantě je každé vozidlo vybaveno zařízením s GSM a GPS modulem. GSM modul obstarává komunikaci mezi velínem a vozidlem. Výstupem této komunikace je minimálně signalizační dioda, popřípadě displej, který signalizuje řidiči, kdy má vyjet ze zastávky, aby jelo vozidlo na čas. GPS modul poté posílá informace do velína, kde vyhodnocuje data a upravuje zelenou na SSZ v daném směru podle aktuálního zpoždění a místa kde se konkrétní vozidlo nachází.

Druhý systém využívá různých dálkových detektorů, které zaznamenají vozidlo v zastávce či před SSZ a pošle přes GSM bránu informace o aktuální pozici a stavu vozidla MHD. Tyto systémy rovnou reagují na vozidlo v křižovatce tím, že mu dají přímo průjezd s vyšší prioritou. Vozidlo přijede předčasně do zastávky a tím se zefektivní funkce MHD [20].

1.5 Piezoelektrické váhy

1.5.1 Princip piezoelektrického jevu a zařízení

Piezoelektrický jev je schopnost krystalů generovat elektrické napětí při jejich deformaci, popřípadě jev opačný, kdy se v krystalu v elektrickém poli deformuje [21].

1.5.2 Využití piezoelektrických vah v dopravě

Piezoelektrické váhy se používají ke kontrole vážení nákladních vozidel za jízdy. Několik těchto vah je již v ČR instalováno [22]. Některé z nich se nacházejí i na dálnici

D1. Ředitelství silnic a dálnic dostává vyhodnocené informace z těchto vah a může je předat i Policii ČR nebo Celní správě. Zatím chybí patřičná legislativa, která by automaticky trestala přetížená vozidla. Policie na základě těchto poskytnutých informací může vytipovat vozidlo a poté ho zastavit k převážení.

Zapojení vážícího zařízení je následující. V první řadě se jedná o předřazenou indukční smyčku, která aktivuje systém detekce váhy. Dále se jedná o dva piezoelektrické články umístěné pro každý pruh nebo po celé šíři vozovky. Také mluvíme o kamerách s vysokým rozlišením, které fotí registrační značku vozidla (dále jen „RZ“). Vozidlo najede na indukční smyčku, kde je detekováno. Poté najede do oblasti váhy a je převáženo díky změně napětí v piezoelektrickém článku.

Systém by měl být vybaven korekčními koeficienty. Ty jsou zde pro případ, kdy kamion či jiné vozidlo například brzdí, zrychluje či přejíždí z pruhu do pruhu, tímto je jeho váha nerovnoměrně rozložena a musí zde být zavedena určitá matematická korekce, která uvádí přesnost změřené váhy vozidla.

Dále zde musí být software, který je schopen přepsat do elektronické podoby RZ. Kamera je sice vybavena infračerveným přisvětlením, ale v některých případech může být RZ z důvodu nečistoty či poškození nečitelná. Proto by byla potřebná i kontrola výstupní RZ či opět použit software určující přesnost přepsané RZ. U nižších procent by byla překontrolována pověřeným pracovníkem.

Tyto systémy by se díky vhodné legislativě daly využít tak, že by vozidla, která nesplňují požadavky na váhu na dané komunikaci, byla nasměrována k nejbližšímu sjezdu, či na jinou komunikaci. Za tímto účelem by byla vhodná signalizační tabule, umístěná ve vhodné vzdálenosti od piezoelektrických vah [23].

1.6 Bluetooth detekce vozidel

Systém využívá anonymního sběru MAC adres zařízení, která mají zapnutý bluetooth. V dnešní době se jedná o většinu autorádií, která mají permanentně zapnutý režim bluetooth pro připojení telefonního zařízení. Tato detekce je prováděna například v místě s větší hustotou dopravy či místě častých dopravních kolon.

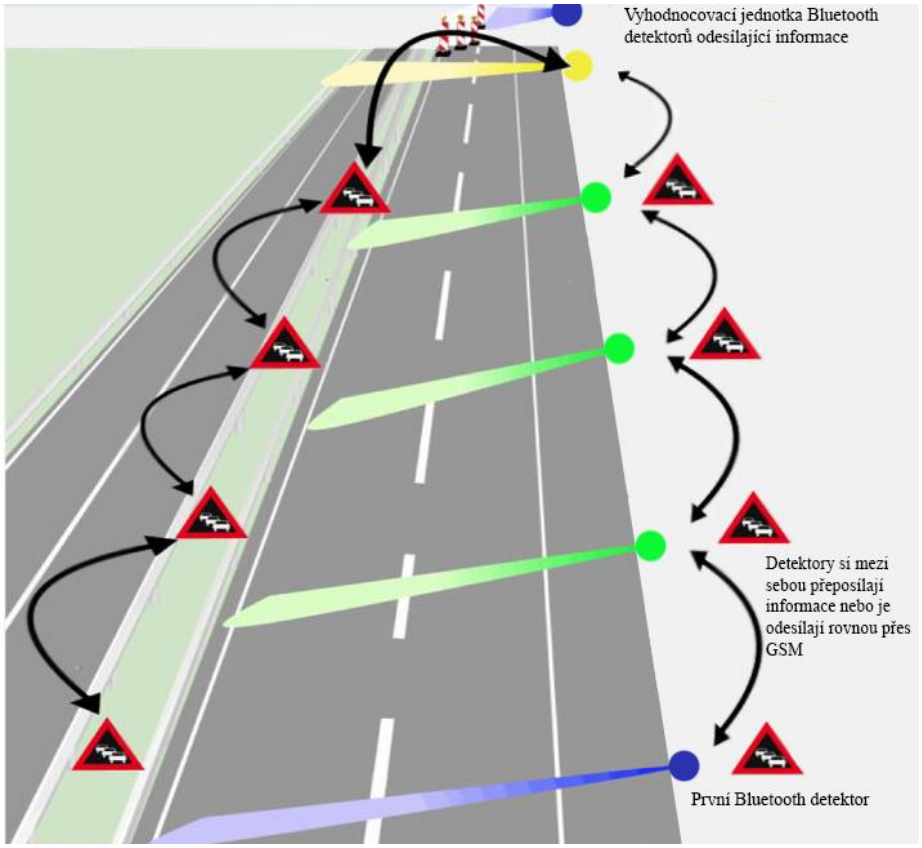
Sběr dat je rozložen do více zařízení, která jsou umístěna po celé délce rizikového úseku a rozpoznávají již detekovanou MAC adresu. Poté je vyhodnocena doba, která byla změřena mezi jednotlivými úseky. V oblasti bluetooth detekce by měly být také využívány korekce. Ty zde musí být z důvodu odbočení vozidla mimo komunikaci či zastavení vozidla například na čerpací stanici. Poté by se časy rozcházely a ovlivnilo by to zásadním způsobem měření. Tento systém je již instalován na rizikových místech stavby dálnice D1.

Jedná se převážně o orientační měření díky rozdílné rychlosti jízdy jednotlivých motorových vozidel. Některá vozidla mohou mít v měřeném úseku rozdílné omezení rychlosti například nákladní vozidla či vozidla nadměrného nákladu.

Systém je rozdělen na dva typy. První zaznamená vozidlo a poté přes GSM bránu přeposílá data k vyhodnocení do velína a tam se data zpracují s ohledem na po sobě jdoucí detektory.

Druhý systém spočívá v tom, že jsou data přeposílána mezi zařízeními až do zařízení koncového, jak je vidět na obrázku Obr. 6 a až poté odeslána do velína. Tam jsou data zpracována a předána dále například provozovatelům stránek

www.dopravniinfo.cz či www.maps.google.com, které provozují mapování úrovně zatížení dopravy ve městech a na dálnicích [24].



Obr. 6: Bluetooth detekce (převzato z [24])

1.7 Parkovací systémy

Parkování ve městech je dnes obecně dost složité, až 30% dopravy ve městě tvoří řidiči hledající parkovací místo a přitom díky neefektivnímu nakládání s parkovacími místy dochází k nevyužití až 15 % těchto míst. Většina měst se snaží tento problém vyřešit parkovišti (Park + Ride zk. P+R) v okrajových částech města s přilehlou MHD v místě parkoviště, jak již bylo uvedeno v části, která se zabývala preferencí MHD.

Obr. 7: Stav obsazenosti parkoviště (převzato z [26])

Existující systémy s parkováním vychází z koncepce na omezený vstup aut do parkoviště podle zbývajícího počtu volných míst. Na příjezdové cestě mohou být zabudovány indukční smyčky, systém kamerového sčítání dopravy či systém počítání parkovacích lístků vydávaných na vjezdu do parkoviště.

Dále se nad parkovacími stáními nachází svítící LED dioda, indikující červeně nebo zeleně stav obsazení stání podle toho, zda je místo prázdné či obsazené. V podlaze či nad parkovacím stáním jsou integrovaná čidla zaznamenávající zaparkované vozidlo. Čidla mohou být infračervená či hloubkové senzory. Firma Smart Parking Ltd poskytuje například i mobilní aplikaci zobrazující volná parkovací místa v konkrétním parkovacím domě, jak je vidět na obrázku Obr. 7 [25] [26].

1.8 Meteorologie a doprava

Meteorologie velice úzce souvisí s dopravou na silničních komunikacích. Můžeme hovořit například o meteorologických stanicích, které se nacházejí podél dálnice D1 či na některých hlavních tazích. Meteostanice neustále měří povětrnostní podmínky, vlhkost, teplotu vzduchu, teplotu povrchu vozovky, dále jsou schopny detekovat námrazu, led, mlhu, silný vítr a mokrý nebo sněhem pokrytý povrch vozovky a tyto údaje odesílat přes GSM modul a dále vyhodnocovat. Díky vyhodnocování těchto dat vědí v zimě provozovatelé služeb posypových prací kam poslat svá vozidla díky informacím o teplotě povrchu vozovky. Pokud je na povrchu vozovky teplota blížící se teplotě bodu mrazu, měl by být vyslán sypačský vůz se solí. Ten zajistí, že dojde ke smíchání sněhu či námrazy se solí a tím dojde k posunutí hranice bodu mrazu na vozovce. Hranice se posune například k -20 °C a meteostanice opět odešle tato data jako vozovku ošetřenou posypovou solí, díky teplotě naměřené na povrchu, která je hlouběji pod bodem mrazu. Tímto způsobem je možné vytipovat silnice, které je potřeba přednostně posypat solí a zajistit tak bezpečný provoz motorových vozidel [27].

1.9 Shrnutí kapitoly

Tímto bych chtěl upozornit na celkovou komplexnost, provázanost a důležitost všech shrnutých dopravních činitelů. Teoreticky byly tyto prvky nastíněny v některých případech povrchně vzhledem k jejich složitosti a obsáhlosti.

Většina aspektů řízení dopravy má dalekosáhlé důsledky vztahující se na obyvatele měst. Počínaje každodenním projížděním křižovatek, na kterých se nachází inteligentní SSZ, který nás bezprostředně ovlivňuje, přes svítidla s využitím LED technologie, až po preferenci záchranných složek, kde může jít v mnoha případech o život. Tyto pro většinu lidí všední věci velkou měrou ovlivňují náš život, pomáhají nám, aniž bychom si to uvědomovali. Proto je na tyto systémy kladen velký důraz a mají vysokou prioritu.

1.10 Telemetrie

Telemetrie je obor zabývající se měřením na dálku a dálkovým přenosem dat k centrální vyhodnocovací jednotce. Telemetrie většinou bývá zprostředkována pomocí bezdrátových sítí, jako jsou WIFI, Bluetooth, GPS, rádiové vlny a v dnešní době velmi rozšířeným GSM. Signál je možno přenášet skrze jakékoli jiné médium například metalické kabely či optická vlákna. Systémy telemetrie dnes najdeme v mnoha oborech, ve kterých se člověk snaží zefektivnit daný systém.

Základním kamenem telemetrických sítí jsou senzory. Senzory mohou sloužit k detekci, pro účely monitorování, varování nebo regulaci. Může se jednat o senzory tlakové, teplotní, indukční, detekci pomocí kamer, detekce pomocí sejmutí MAC adresy z bluetooth zařízení atd. Poté jsou informace ze senzorů přeneseny do systému pro vyhodnocení, ten data zpracuje a následně na podněty reaguje dle požadovaných vlastností [28][29].

1.11 Telemetrie v dopravě

V telemetrii v dopravě je kladen velký důraz na přesnost senzoru a rozpoznávacího systému, jelikož vyhodnocovací systém s těmito daty pracuje v reálném čase a pro ideální řízení je tedy kladen velký důraz na přesnost senzorických systémů.

Jedním z nejpoužívanějších a nejspolehlivějších detektorů je indukční smyčka. V podstatě se jedná o vodivý metalický kabel připojený na transformátor a do vyhodnocovací jednotky, která zjišťuje a sleduje změny elektromagnetické indukce v podobě změny napětí či proudu na transformátoru.

Pro zlepšení přesnosti detekce pomocí smyček využíváme změny tvaru smyčky a jejich využití vícekrát za sebou pro větší přesnost určení jízdy vozidla do různých pruhů.

Pokud nelze využít z jakéhokoli důvodu indukční smyčku, nahrazuje se kamerovým systémem s detekcí dopravy. Jedná se v podstatě o kameru připojenou na vyhodnocovací jednotku, ve které je nastaveno pole, v jehož části obrazu má docházet k detekci a následně systém po detekci vyhodnotí, že se v místě nachází vozidlo. Systém je mnohem méně přesný díky vlivům počasí (husté sněžení, mlha), v těchto situacích není schopen detekovat vozidlo, nebo dojde k chybné detekci [30].

Senzory pro zjištění hmotnosti vozidel v dopravě jsou piezoelektrické váhy, které slouží k okamžitému převážení vozidla. Před váhou bývá vždy aktivační indukční smyčka, systém je napojen na kamery snímající registrační značku vozidla. Ta je přečtena opět za pomoci rozpoznávacího softwaru. Zjištění plynulosti provozu dochází za pomoci mnoha zařízení, jedná se například o anonymní sběr dat ze systému GPS v telefonních zařízeních. Nebo také o „sejmutí“ MAC adresy Bluetooth zařízení, kterých je více v řadě za sebou a je měřen čas mezi jednotlivými průjezdy snímacími body.

Další prvek telemetrie je senzor pro výpočet hustoty provozu. Jedná se o zařízení umístěné do vyfrézované vozovky, které vychází z funkce piezoelektrického principu. Toto zařízení je dosti nepřesné díky svému principu měření, proto se tedy měření hustoty provozu provádí za pomoci lidských zdrojů. Pověřený člověk počítá vozidla jedoucí po komunikaci, čísla si zapisuje, poté vloží čísla do softwaru, který odvodí od měření zátěž na okolních komunikacích.

Všechny informace můžeme využít, díky popisovaným prvkům víme v aktuální chvíli mnoho informací o dopravě a můžeme tyto údaje přenést do simulačního prostředí. Naopak budeme schopni realizovat podmínky simulace v reálném provozu. Díky možnosti absolutní kontroly nad vozidly a všemi kontrolními prvky v simulačním prostředí není až tak jednoduché aplikovat optimalizaci na reálném zařízení.

V dnešní době dochází k optimalizacím v podobě použití inteligentních semaforů. Tyto semafory však často nejsou schopny reagovat jako jeden celek ale fungují pouze lokálně. To znamená, že optimalizace z pohledu jedné lokální křižovatky nemusí městu a dopravě přinést žádaný užitek. Zde je nutno nasadit propojení jednotlivých systémů a jejich vzájemné komunikace.

Systém více křižovatek lze převést do simulačního prostředí a optimalizovat průjezd s prioritou jednoho nebo více směrů. Vždy je vhodné vytvořit studii, která vytvoří model četnosti průjezdu vozidel na dané komunikaci. S těmito daty může již optimalizační algoritmus pracovat. Ať hovoříme o úrovni dopravy v dané křižovatce či na komunikaci vzhledem k denní době, nebo zde poznáme, že v současné chvíli je doprava zvýšená i z důvodu uzavírky nebo objížďky. Lze vytvořit plán nastavení systému semaforů podle denní doby. Tento plán by počítal s různou úrovní dopravy ze statistických dat. Vytvoření systému se schopností vyhodnocování a optimalizací v reálném čase, by bylo velmi komplikované. Je možno počítat tedy se statistickou predikcí, která se může vždy za určitý časový úsek aktualizovat díky detekci dopravy pomocí indukčních smyček. Optimalizace by se vždy aktualizovala z novějších dat, která by byla postupně sbírána.

1.12 Budoucnost v telemetrii

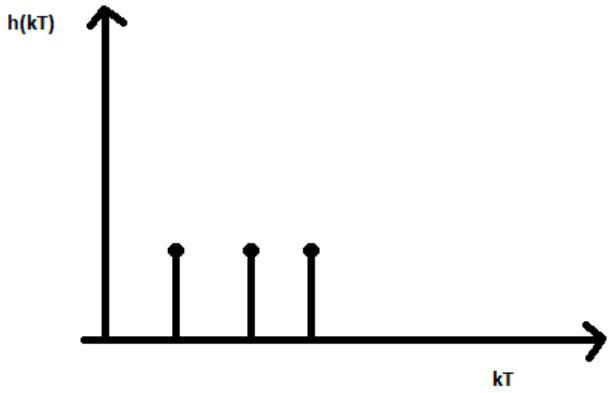
Telemetrie klade stále vyšší a vyšší požadavky na přesnost detekčních prvků. Od ní se odvíjí efektivita celého systému. Dále se telemetrické prvky a řízení budou snažit dostat k aktivnímu užívání běžnými občany. Jako příklad je možno vnést využití telemetrie v parkovacích domech. Nahlédnutím do aplikace, zjistíte, ve kterém parkovacím domě se nachází volné parkovací stání v reálném čase. Předpokládám, že v budoucnu bude kladen vyšší důraz na propojení jednotlivých systémů za účelem sběru dat a optimalizace městské dopravní infrastruktury [31].

2 SIMULACE DOPRAVY

Simulace je obecně definována jako napodobení reálné situace či reálného prostředí, ve kterém chceme vyzkoušet nový způsob řízení, který není možné z různých důvodů realizovat. Systém je převeden do virtuálního prostředí, kde dojde ke zmíněnému testování. Samotná simulace zahrnuje některé klíčové vlastnosti či chování abstraktních nebo fyzikálních systémů. Každou reálnou situaci v dopravě je možno převést do podoby matematického, fyzikálního nebo jiného modelu, který vyjadřuje její prvky. Na simulaci dopravy je možno obecně nahlížet jako na komplexní dynamicky se měnící systém.

Simulace můžeme dělit podle toho, jak probíhají v čase. Pokud čas ubíhá po malých stejně velkých časových intervalech, potom se jedná o spojitou simulaci.

Pokud ale upravujeme model jen v časech, kdy se děje nějaká událost (event) nazývá se tato simulace diskrétní. Tento model je znázorněn na obrázku Obr. 8 [32][33].



Obr. 8: Diskrétní systém

Dále můžeme provádět simulaci reálnou, která se snaží maximálně zaměřit na všechny možnosti a nalézt jejich řešení nebo s nimi počítat v programu vyhodnocovací jednotky. Tato simulace je nejvíce časově náročná z důvodu její komplexnosti z pohledu přenesení reálných podmínek do vytvořeného testovacího prostředí a jeho následné aplikace do reálných podmínek.

V dopravě jde ve velké většině případů o řadu událostí (eventů), proto se zde ve větší míře využívá diskrétních simulací. Při simulaci dopravy za účelem jejího zefektivnění například využíváme eventy jako změna barvy semaforu, čekací doba vozidla na semaforu, vozidlo vjíždějící/vyjíždějící do či ze systému. Jedná se o data, která nás zajímají nejvíce, jelikož se snažíme, aby byla doprava co nejplynulejší, s minimálními čekacími dobami vozidel na semaforech.

2.1 Problematika simulace dopravy

Problematické jsou simulace v reálném čase a ve velkém měřítku s mnoha komplexními křižovatkami. Důvodem je, že systém musí být v reálném čase schopen reagovat na mnoho různých situací. Některé situace se dají očekávat jako třeba nechat systém předvídat zatížení vzhledem k přehledu intenzity dopravy dané komunikace v určité denní hodině. Jiné situace se očekávat nedají, například nehoda v oblasti nad indukční smyčkou a její nepřetržitá detekce vozidla, porucha indukční smyčky a její neschopnost detekce.

S těmito reálnými modely situací se dá jen velmi těžko pracovat v oblasti simulace. Zmíněné situace znamenají problém s převodem simulačního modelu do reálného prostředí a samozřejmě i naopak. Také to znamená, že nelze stoprocentně spoléhat na detekční prvky. Dále musíme počítat s měnící se hustotou dopravy a náhodnými výkyvy v zátěži silnic (dopravní nehoda, objízdná trasa). Problematické prodlužování zelené v hustší dopravě a s tím související priority jednotlivých komunikací nebo dokonce priorita vozidel (IZS) v dopravě, z čehož ve výsledku musí vzniknout velmi obsáhlý přizpůsobivý systém. Převedením reálného systému na teoretický model a vytvořením simulačního prostředí poté zapracujeme optimalizaci, avšak další podstatná část problematiky je, jak přenést optimalizaci řešení do reálných podmínek [34].

2.2 Řešení problematiky v simulaci dopravy

Ve většině případů se díky složitosti velkých obecných systémů využívá rozdělení na mikrosimulace a makrosimulace. Mikrosimulace se zabývá konkrétním řešením daných problémů v omezené lokalitě, zatímco makrosimulace se soustřeďují na obecné řešení problémů řízení infrastruktury dopravy, například na úrovni města. K tomu makrosimulace využívají různé evoluční algoritmy nebo algoritmy inteligence hejna, s implementací metody predikce, aby dokázaly počítat s velkým množstvím možností [35].

2.3 Vybrané algoritmy v dopravě

V simulaci dopravy existuje obecně mnoho algoritmů vyhovující různým podmínkám. Jedná se například o algoritmy mravenčí kolonie, heuristické algoritmy, genetické algoritmy. Všechny zmíněné algoritmy slouží k optimalizaci řízení dopravní infrastruktury. Hodí se pro optimalizaci nejvíce a jsou nejpoužívanější [36][37][38].

2.4 Algoritmy teorie grafu

Algoritmy využívající teorii grafů bývají základem v plánování tras nebo v systémech pro GPS zařízení. Jedná se o souvislý graf s metrikou, díky kterému získáme „mapu“ z informace odkud kam vede silnice a jak je dlouhá. Jedná se o různé podoby interpretace informací o mapě. Mapa bývá většinou tvořena tabulkou, kde na jedné ose je výchozí bod a na ose druhé je bod cílový. Algoritmus poté porovnává jednotlivé hodnoty trasy mezi úseky, podle toho odkud kam chce vozidlo v mapě dojet [39].

2.5 Dijkstrův algoritmus

Pro tuto práci postačí algoritmus k vyhledání nejkratší tras. Jako algoritmus pro vyhledání nejvhodnější trasy postačí jeden z algoritmů teorie grafu a do práce byl zvolen algoritmus Dijkstrův. Především z důvodu jeho vlastností, díky uvedeným váhám mezi uzly je možno přičíst zátěž na úrovni jednotlivých komunikací, jedná se o algoritmus s konečným počtem průchodů. Průchodů je tedy nejvýše tolik, kolik má graf vrcholů, tudíž není tolik náročný na výpočty operací.

Graf lze zefektivnit tím, že bude obsahovat informace o seznamu sousedů jednotlivých bodů. Z mnoha obdobných algoritmů je Dijkstrův algoritmus považován za nejrychlejší. Dijkstrův algoritmus je využíván v informatice pro přenos dat v OSPF protokolu. Jako algoritmus pro nastavení optimálního časového intervalu pro jednotlivé semafory byl vybrán genetický algoritmus [40].

2.6 Evoluční algoritmy

Tyto algoritmy obecně vycházejí ze základů Darwinovy teorie o přirozeném výběru. Na schopnosti přizpůsobit se prostředí a populačního boje mezi sebou bojují jedinci o přežití, na základě největší síly (fitness) přežívají a vytváří nové generace. Tohoto modelu využívají evoluční algoritmy, aby nalezly řešení náročných a rozsáhlých úloh. Evoluční algoritmy pracují s celou skupinou řešení daného problému místo hledání jednotlivých řešení. Nově generovaná populace se postupně zlepšuje získáváním nových vhodnějších řešení a získáváním kombinací původních. Dále jsou nová řešení následována náhodnými změnami a vyřazováním nevyhovujících řešení [41].

2.7 Genetický algoritmus

Genetický algoritmus je tedy založen na mechanismu přirozeného výběru z genetiky. Jedná se o proces postupného vylepšování populace jedinců vycházejících z generace předchozí. Proces konverguje až do bodu, kdy je populace tvořena nejvhodnějšími jedinci (učící se algoritmus). Hlavním principem je výměna řetězců tedy chromozomů. Chromozom má pevnou délku a je tvořen geny. Jednotlivé geny často reprezentuje binární číslo 0 nebo 1. Každý chromozom v populaci má definovanou hodnotící (fitness) funkci, která charakterizuje míru vhodnosti chromozomu. Během procesu dochází k selekci, rekombinaci a mutaci dané možnosti. Při aplikaci je nutné stanovit omezující podmínky, podmínky konvergence, kvalifikovat proměnné, atd. Problémem bývá to, že stochastické algoritmy (algoritmus nemusí být deterministický) ne vždy zaručují nalezení optimálního řešení i pokud algoritmus v předem daném počtu kroků

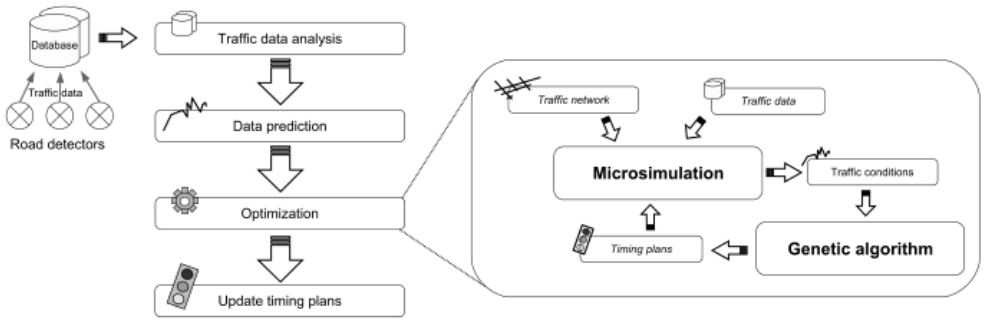
zkonverguje, není jistá vzdálenost optimálního řešení. Obvykle se problém řeší vícenásobným spuštěním výpočtu algoritmu, nebo použití hybridních metod, jako zadání počátečních podmínek, ze kterých je pak spuštěn deterministický algoritmus.

Selekce je proces, který slouží k výběru jedinců z populace, ti se mohou stát rodiči. Je mnoho metod jak provést tento krok, jedná se například o metodu Vážená ruleta. V tomto případě dostane každý jedinec takový podíl, který odpovídá jeho fitness hodnotě. Dále je zde Turnajová metoda, v této metodě je vždy náhodně vybrána skupina jedinců z populace. Vítězem se pak stává metoda s nejvhodnější fitness hodnotou.

Křížení navazuje na selekci. Rozlišujeme jednobodové a vícebodové křížení. Jednobodové křížení je metoda, kdy se náhodně zvolí bod v chromozomu, poté se chromozom rozdělí na dvě části a ty se mezi potomky vymění. Křížením vzniknou dva nové chromozomy. Je na algoritmu, zda bude pracovat s oběma nebo si jeden chromozom vybere. Vícebodové křížení je, když kód potomka vzniká z více než dvou rodičů.

V případě mutace, se jedná proces, kdy u každého jedince z nové generace dochází k procházení celého chromozomu a s malou pravděpodobností jsou měněny hodnoty genů z 0 na 1 a obráceně. Význam tohoto procesu je hlavně v tom, že se může v dané generaci objevit vlastnost, kterou doposud žádný jedinec neměl a nemohl ji tudíž ani předat svým potomkům.

Genetický algoritmus je vhodný k aplikaci a k nastavení ideálního času přepínání semaforu na základě hustoty dopravy viz. Obr. 9. Dále byl také využit algoritmus k nalezení ideální trasy. Dalo by se v něm na základě změny hustoty dopravy upravovat trasy vozidel. Ta by byla připočítávána jako váha v dané komunikaci a hledáním nejmenší váhy by algoritmus vyhledal ideální trasu. Záleželo by ovšem na preferovaném parametru hledání (nejrychlejší cesta, nejkratší cesta atd.) [42][43].



Obr. 9: Znázornění modelu microsimulace s použitím genetického algoritmu se systémem predikce (převzato z [47])

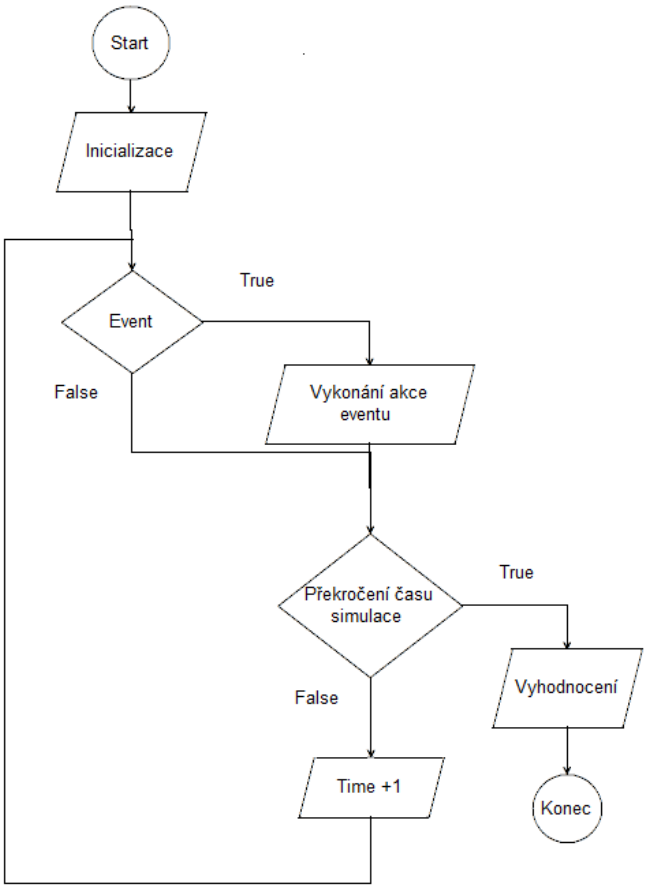
2.8 Volba programovacího jazyka

Vzhledem k nutnosti využití objektově orientovaných programovacích jazyků z důvodu jednoduchosti, byl pro práci zvolen programovací jazyk Java. Díky implementaci objektového programování, jednoduchosti, automatické správě paměti a snadné přenositelnosti se jeví programovací jazyk Java jako nejlepší možný. Další objektově orientovaný programovací jazyk, který by mohl být brán v potaz je rodina jazyku C. Jenže díky horší schopnosti podpory na různých platformách a menší podpoře knihoven

rodiny jazyka C i popularitě oproti jazyku Java zůstává rozhodnutí na tom, se kterým jazykem má programátor více zkušeností [44].

2.9 Návrh programu

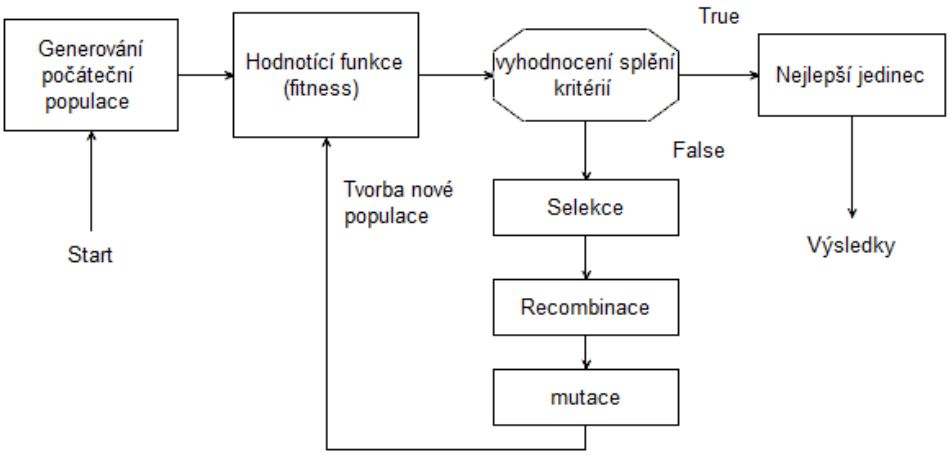
Struktura programu je vidět na vývojovém diagramu na obrázku Obr. 10. Jedná se o model diskrétní simulace. Program na začátku inicializuje všechny proměnné (vytvoření vozidel a mapy, nulování proměnných, vytváření eventů). Algoritmus zjistí, zda se jedná o event v aktuálním čase, pokud ano dojde k vykonání akce eventu (například přepnutí semaforu). Pokud ne, algoritmus přejde rovnou ke kontrole překročení času simulace. Je-li podmínka splněna, program vyhodnotí všechna data a dojde k jeho ukončení. Není podmínka splněna, algoritmus se vrátí na začátek v novém čase t+1.



Obr. 10: Vývojový diagram programu

2.10 Návrh genetického algoritmu

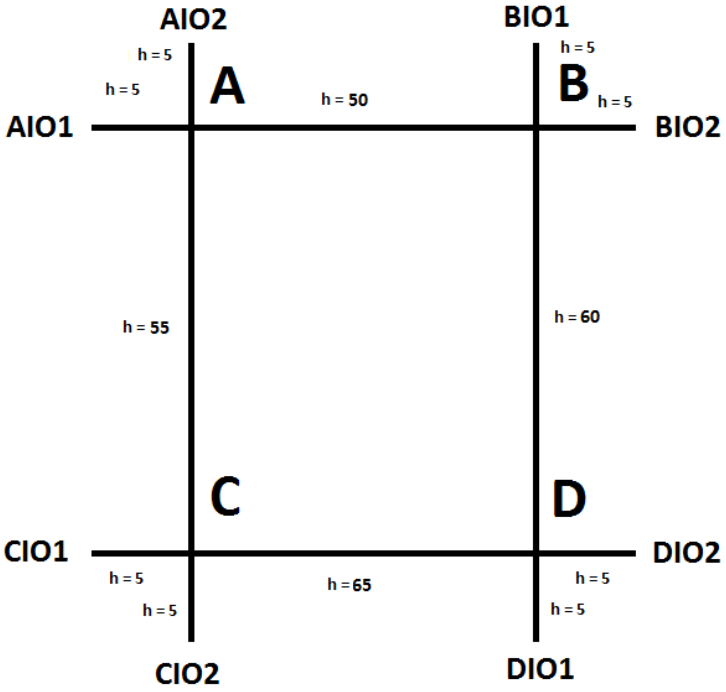
Na obrázku Obr. 11 je vidět princip funkčnosti genetického algoritmu. Algoritmus začne vygenerováním počáteční populace, vytvoří se fitness funkce, která bude porovnávat výsledky nových mutací i předcházejících generací. Dále se dostáváme k podmínce splnění požadovaných kritérií (v případě této práce čas), pokud nejsou splněny, dochází k náhodné selekci části populace, jejich rekombinaci a mutaci. Tento proces se opakuje, dokud nebude nalezen nejlepší jedinec. Na jeho základě, pak budou vyhodnoceny výsledky simulace.



Obr. 11: Znázornění průběhu genetického algoritmu

2.11 Návrh mapy prostředí

Mapa prostředí diskrétní simulace byla navržena také, aby nedocházelo ke kolizi výsledků Dijkstrova algoritmu a zároveň, aby nebyla příliš složitá z pohledu simulace a délky simulace. Jedná se o mapu 4 křižovatek A, B, C, D a pomocných vstupních a výstupních křižovatek jako hranice systému AIO1, AIO2, BIO1, BIO2, CIO1, CIO2, DIO1, DIO2. Jak je vidět na obrázku, jsou zde uvedeny i váhy jednotlivých silnic (například mezi A a B je to 50). Váhy jsou uvedeny v diskrétní situaci jako časový úsek, za jaký bude vzdálenost uražena a je uvedena v sekundách. Samotná mapa je vidět na obrázku Obr. 12.



Obr. 12: Mapa simulačního prostředí

Konfigurační údaje jsou uloženy v souboru map.txt a to ve formě kódu, který je vidět v tabulkách Tab. 1,Tab. 2 a Tab. 3.

Tab. 1: Seznam pruhů

PATH	FROM	TO	COST
PATH	A	B	50
PATH	A	C	55
PATH	B	D	60
PATH	C	D	65
PATH	A	AIO1	5
PATH	A	AIO2	5
PATH	B	BIO1	5
PATH	B	BIO2	5
PATH	C	CIO1	5
PATH	C	CIO2	5
PATH	D	DIO1	5
PATH	D	DIO2	5

Tab. 1: Seznam pruhůje zde vidět nastavení jednotlivých pruhů, odkud kam vedou a hodnoty časů pro přejezd mezi jednotlivými křižovatkami simulace.

V tabulce Tab. 2: Seznam pomocných křižovatek nalezneme seznam pomocních křižovatek, které slouží jako input output prvky systému s jejich napojením na jednotlivé křižovatky. V tabulce číslo Tab. 3 je také vidět, že se v souboru nachází

nastavení zelené a červené, které jsou rozděleny do skupin. Jedná se o pruhy směřující proti sobě, kdy oba pruhy mají zelenou a vozidla mohou pokračovat dále ve své trase.

Tab. 2: Seznam pomocných křižovatek

Lane From	Lane To
LANE_IO AIO1	A
LANE_IO AIO2	A
LANE_IO BIO1	B
LANE_IO BIO2	B
LANE_IO CIO1	C
LANE_IO CIO2	C
LANE_IO DIO1	D
LANE_IO DIO2	D

Tab. 3: Seznam pruhů a časů

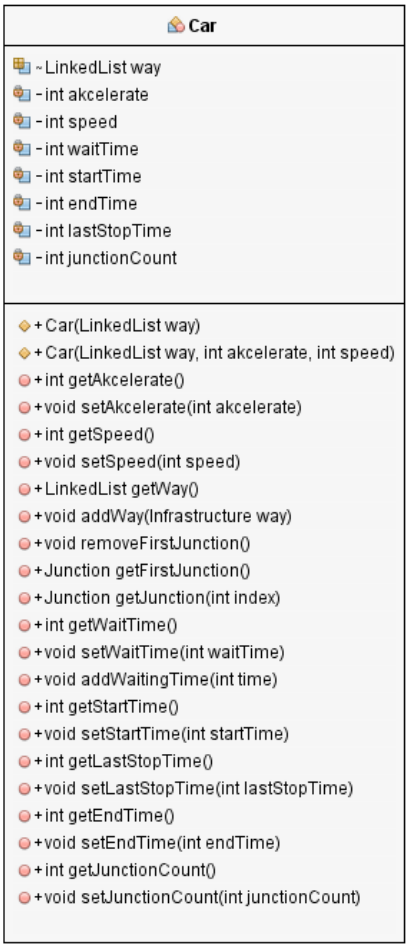
LANE	FROM	CURRENT	TO1-TO2	green	gPeriod	rPeriod	Delay	Group
LANE	AIO1	A	B-C-AIO2	true	15	15	0	1
LANE	AIO2	A	B-C-AIO1	false	15	15	-15	2
LANE	B	A	C-AIO1-AIO2	true	15	15	0	1
LANE	C	A	B-AIO1-AIO2	false	15	15	-15	2
LANE	BIO1	B	A-D-BIO2	true	15	15	0	1
LANE	BIO2	B	A-D-BIO1	false	15	15	-15	2
LANE	A	B	D-BIO1-BIO2	true	15	15	0	1
LANE	D	B	A-BIO1-BIO2	false	15	15	-15	2
LANE	CIO1	C	A-D-CIO2	true	15	15	0	1
LANE	CIO2	C	A-D-CIO1	false	15	15	-15	2
LANE	A	C	D-CIO1-CIO2	true	15	15	0	1
LANE	D	C	A-CIO1-CIO2	false	15	15	-15	2
LANE	DIO1	D	B-C-DIO2	true	15	15	0	1
LANE	DIO2	D	B-C-DIO1	false	15	15	-15	2
LANE	B	D	C-DIO1-DIO2	true	15	15	0	1
LANE	C	D	B-DIO1-DIO2	false	15	15	-15	2

3 POPIS TŘÍD

3.1 Třída Car

Tato třída slouží k interpretaci vozidla ve vybrané síti křižovatek a k jejímu adekvátnímu nastavení klíčových parametrů. Třída obsahuje 8 privátních proměnných viz. Obr. 13, popis nejdůležitějších:

- junctionCount - počet křižovatek, kterými vozidlo projede
- lastStopTime - čas posledního zastavení vozidla na semaforu
- akcelerate - čas, za který se auto stihne rozjed (delay - zpoždění)
- startTime - čas vjezdu vozidla do systému
- waitTime - celková doba čekání vozidel na semaforech
- endTime - čas výjezdu vozidla ze systému
- way - seznam křižovatek, po kterých vozidlo jede do výstupního bodu



Obr. 13: Třída Car UML Diagram

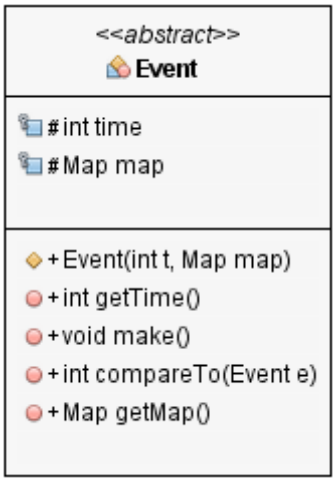
Pro proměnnou way byl zvolen objekt LinkedList z důvodu výhodných vlastností pro naplnění objektu postupně vzhledem k času. Tudíž křižovatky, kterými bude vozidlo projíždět jsou logicky v řadě za sebou, podle toho jak je trasa vygenerována.

Pro vytvoření instance tohoto objektu, slouží konstruktor Car(LinkedList way), využívá se pouze tato jednodušší podoba konstruktoru, ve kterém se vozidlu předá jeho cesta. Druhý konstruktor je vytvořen jako možnost pro využití lepší parametrizace vozidel.

3.2 Třída Event

Třída Event slouží jako abstraktní třída (nelze instancovat), ze které ostatní třídy, které budou vykonávat eventy pouze dědí a sama o sobě tato třída slouží jako společná část všech událostí v systému. Obsahuje proměnnou time, která určuje, kdy se má daný event vykonat. Proměnné viz. Obr. 14:

- time - proměnná, která určuje čas vykonání Eventu v diskrétním systému, jedná se o proměnnou typu protected
- map - obsahuje mapu simulace (obsahuje vozidla, křižovatky a Eventy)



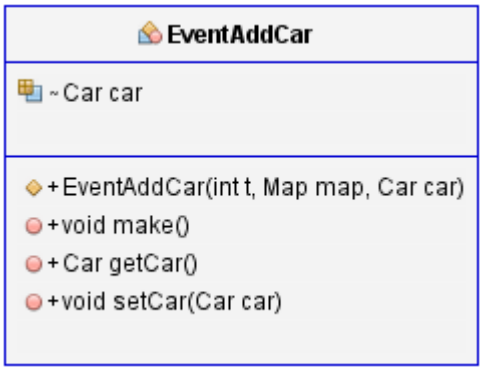
Obr. 14: Třída Event UML Diagram

Konstruktor Event nastaví čas vykonání eventu a nastaví mapu. Dále je zde abstraktní metoda make() a ta slouží k vykonání daného eventu, podle jeho účelu. Metoda compareTo slouží pro porovnání eventů na základě času vykonání pro použití objektu TreeSet.

3.3 Třída EventAddCar

Třída sloužící k přidání vozidla do systému. Obsahuje proměnné viz. Obr. 15:

- car - v této privátní proměnné je obsaženo vozidlo přidávané do systému



Obr. 15: Třída EventAddCar UML Diagram

Metoda make() přesune vozidlo z pruhu první křižovatky jeho cesty a vytvoří EventMoveCar pro další pohyb vozidla. Ukázka kódu (Kód 1), kde dochází k načtení první a následně další křižovatky.

Kód 1: Načítání křižovatky

```
1.  Junction firstJunction = this.getCar().getFirstJunction();
2.  Junction secondJunction = this.getCar().getJunction(2);
```

V další ukázce kódu je vidět hledání a přidání vozidla do pruhu následující křižovatky. Když se tak stane a vozidlo dojede do konce své cesty, vozidlo je odebráno z pruhu.

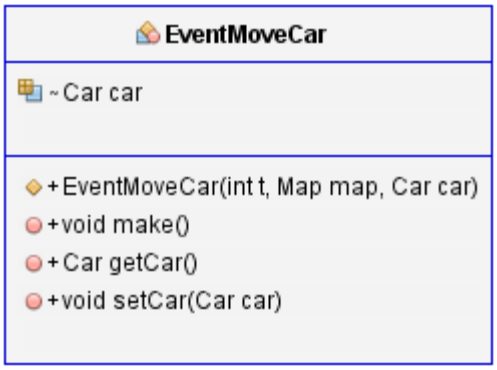
Kód 2: Přidání vozidla do pruhu

```
1.  lane = firstJunction.findLane(firstJunction, secondJunction);
2.  lane.addCar(this.getCar());
```

3.4 Třída EventMoveCar

Třída slouží k posunu vozidla z aktuálního pruhu dané křižovatky do nejvhodnějšího pruhu v další křižovatce. Obsahuje proměnné viz. Obr. 16.

- car - v této privátní proměnné je opět obsaženo vozidlo, které se bude pohybovat do dalšího bodu své cesty



Obr. 16: Třída EventMoveCar UML Diagram

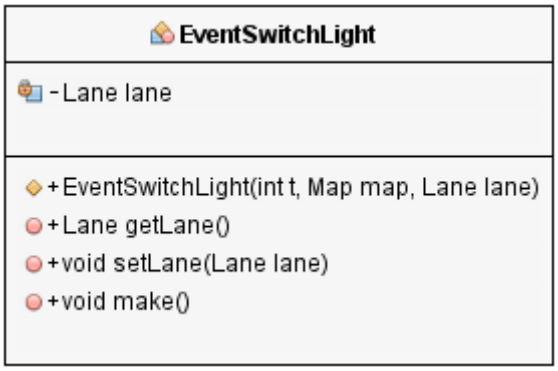
Metoda Make() odebere vozidlo z pruhu současné křižovatky a vytvoří event přesunu vozidla na další křižovatku, pokud bude v pruhu současné křižovatky zelená (Kód 3). Jestliže je v pruhu červená, vozidlo se přidá do nejvhodnějšího pruhu křižovatky a čeká na zelenou. Ve stejnou chvíli se zapíše do proměnné LastStopCar ve třídě Car aktuální čas. Jestliže vozidlo nemá v seznamu další křižovatku, dochází k dokončení jízdy a přesunu vozidla do seznamu vozidel, které opustily systém.

Kód 3: Pokud je v pruhu zelená

```
1.  if(lane.isGreen()){
2.  Junction firstJunction = this.getCar().getFirstJunction();
3.  Junction secondJunction = this.getCar().getJunction(2);
```

3.5 Třída EventSwitchLight

Třída, která se stará o přepínání světel semaforů, obsahuje proměnné viz. Obr. 17.



Obr. 17: Třída EventSwitchLight UML Diagram

Jedná se o třídu, která dědí ze třídy Event a tato třída se stará o přepínání barvy na signalizačním zařízení. Dle předcházejícího stavu semaforu (Kód 4) se nastaví barva a čas eventu pro přepnutí semaforu.

Kód 4: Nastavení semaforu

```
1.  if(this.getLane().isGreen()){
2.              this.getLane().setGreen(false);
3.              time += this.getLane().getRedPeriode();
4.          }
```

Pokud je červená, pak nastaví další barvu, čas a vytvoří další eventy pro přesun vozidel.

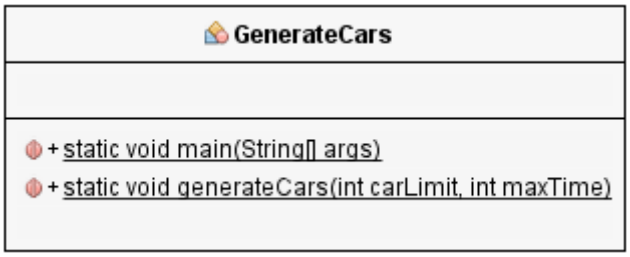
Kód 5: Načtení vozidel v pruhu

```
1.  LinkedList<Car> cars = new LinkedList<Car>(this.getLane().getCars());
2.          for(Car car : cars){
3.            if(carNumber * car.getAkcelerate()>this.getLane().getGreenPeriode()){

4.                        break;
5.                  }
```

Načtou se všechna vozidla v daném pruhu do proměnné cars, poté dojde k procházení listu vozidel v pruhu a pokud x-té vozidlo nestihne odjet z pruhu dříve, než skončí zelená, pak se procházení ukončuje (Kód 5). Pak již dochází k načtení první a druhé křižovatky, vytvoření a přidání do simulace eventu pro přesun vozidla na další pozici s parametry: čas konání dalšího eventu, aktuální čas simulace plus doba průjezdu úseku plus počet aktuálních aut krát akcelerace daného vozidla. Dále dochází k přidání mapy simulace do eventu, přidání vozidla do eventu, odebrání vozidla z pruhu křižovatky, odebrání první křižovatky ze seznamu trasy vozidel a zaznamenání čekacích časů vozidla.

3.6 Třída Generatecars

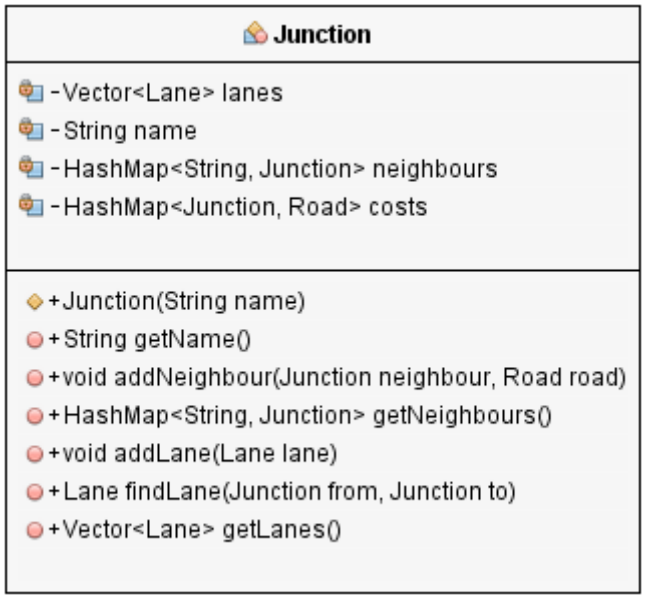
Jedná se o pomocnou třídu vytvořenou za účelem vygenerování textových souborů obsahujících požadované počty vozidel. Jak je vidět na UML Diagramu viz. Obr. 18. jedním parametrem je počet vozidel a parametrem druhým je čas. Jedná se o druhou hraniční hodnotu času. Vozidla se vytváří od času 0 do požadované hodnoty času. V tomto případě se jedná o hodnotu 7 200 sekund. Tudíž 2 hodiny času. Vygenerovaná vozidla jsou v počtu 1, 10, 100, 1 000, 2 500, 5 000, 10 000, 25 000, 35 000, 50 000 a 75 000 vozidel.



Obr. 18: Třída GenerateCars UML Diagram

3.7 Třída Junction

Třída Junction obsahuje proměnné viz.Obr. 19.



Obr. 19: Třída Junction UML Diagram

Třída Junction je připravena na vícepruhové křižovatky. Pro jednodušší systém, jako je v této práci, postačí pro jeden pruh s tím, že z něj odjíždějí vozidla do všech ostatních směrů. Časy průjezdu křižovatkou jsou v tomto případě zanedbatelné. Třída také obsahuje porovnávání prvků vektoru podle toho, kam jede vozidlo a pole počtu vozidel v pruhu s aktuální nejlepší shodou, respektive vyhledání nejkratší trasy na úrovni pruhu (Kód 6).

Kód 6: Přejezd mezi křižovatkou a pruhem

```
1.  for(Junction junctionTo : lane.getTo()){
2.      if(junctionTo == to && lane.getFrom() == from){
3.          if(result == null || lane.getCars().size() < result.getCars().size()){
                        result = lane;
4.              }
5.           break;
6.           }
```

Důvodem je možná nástavba a implementace řazení vozidel v pruzích, kdy například oba pruhy jedou rovně, tak aby se vozidla neshromažďovala v jednom pruhu, ale využila rovnoměrně pruhy oba. Následně vrací, kterým pruhem má vozidlo jet.

3.8 Třída Lane

Jedná se o třídu, která dědí ze třídy Junction a obsahuje proměnné viz .Obr. 20.

Obr. 20: Třída Lane UML Diagram

Obsahuje konstruktor, ve kterém nastavuje hodnoty pruhu - odkud vede, kam vede, stav zelené, časová perioda zelené, časová perioda červené, skupiny semaforů, nastavení a vytvoření eventu na přepínání barvy semaforu a přičtení posunu k periodě dané barvy (Kód 7).

Kód 7: Parametry konsturtoru

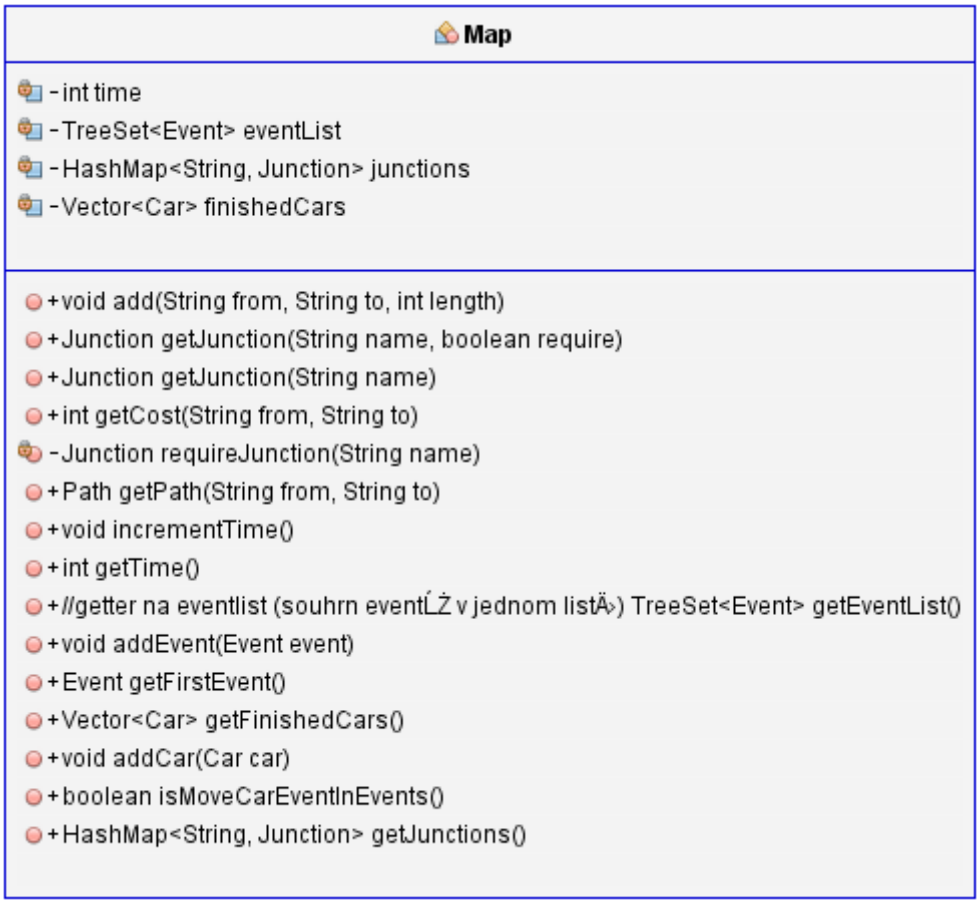
```
1.  public Lane(Map simulation, Junction from, Vector<Junction> to, boolean green,
     int greenPeriode, int redPeriode, int eventDelay, int group) {
2.
3.          this.from = from;
4.          this.to = to;
5.          this.green = green;
6.          this.greenPeriode = greenPeriode;
7.          this.redPeriode = redPeriode;
8.          this.group = group;
9.          this.waitingTime = 0;
10.         int nextTime = eventDelay;
11.         if(green){
12.             nextTime += greenPeriode;
13.         }
14.         else{
15.             nextTime += redPeriode;
16.         }
17.         simulation.addEvent(new EventSwitchLight(nextTime, simulation, this));

18.     }
```

Z toho vyplývá, že barva na semaforu je řízena na úrovni pruhů. Ve vytvořeném simulačním prostředí je tedy vytvořena křižovatka se čtyřmi vstupy a čtyřmi výstupy. Díky tomu, že každý směr má svůj jeden pruh, jedná se o křižovatku se čtyřmi semafory. Nástavba pro větší a složitější křižovatky je rozhodně možná a realizovatelná například v diplomové práci. Zde by se jednalo o rozdělení infrastruktury křižovatky a oddělení jednotlivých vozidel do pruhů pro své vlastní směry a například zvýšení množství pruhů pro jeden směr.

Dále je implementována proměnná group. Ta je zde proto, že ve zjednodušeném diskrétním prostředí simulace nevadí, když jedou současně dva pruhy v křižovatce, které leží proti sobě. Program je nastaven tak, že červená je v tomto případě opakem zelené, což ničemu nevadí, ale jedná se o jisté zjednodušení, díky dodatečné implementaci. Tyto skupiny by v navazující magisterské práci rozhodně šly přizpůsobit tak, aby byly nahrazeny skupinami tras skrze křižovatku tak, aby trasa vozidel nekolidovala. Například dva vpravo odbočovací pruhy v křižovatce v protichůdném směru mohou jet současně.

3.9 Třída Map

Třída map obsahuje proměnné, které jsou vidět na UML Diagramu viz Obr. 21.



Obr. 21: Třída Map UML Diagram

Třída map slouží převážně k načtení a distribuci informací týkající se mapy. Ta, jak již bylo zmíněno v kapitole "Návrh mapy prostředí", je uložena v rootu souboru simulace v .txt souboru. Je otevřena ve třídě main a načtena do proměnných ve třídě map.

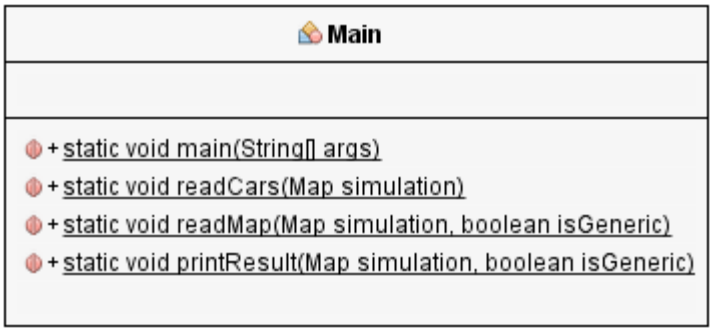
Dále se zde nachází část kódu Dijkstrova algoritmu pro nalezení nejkratší trasy. Algoritmus na základě hodnoty "cost" jednotlivých cest mezi křižovatkami generuje vozidlům seznam trasy. Dijkstrův algoritmus byl zvolen díky své schopnosti nalezení nejkratší trasy i mezi vícekřižovatkovým systémem a díky své schopnosti reagovat dynamicky na změnu mapy, nebo dokonce reagovat na přičítání parametru díky hustotě provozu v dané lokalitě. V případě tohoto simulačního prostředí postačí, že je Dijkstrův algoritmus schopen se dynamicky přizpůsobit změně "cost" hodnoty mezi jednotlivými křižovatkami. Tudíž je jakákoli editace mapy možná a vozidlům se předá pouze jiná trasa, která je pro vozidlo výhodnější. V navazující magisterské práci by se dal Dijkstrův algoritmus naplno využít i s tím, že by se vytvořila komplikovanější infrastruktura křižovatek a měnila by se hodnota "cost" s každým nově jedoucím vozidlem na trase mezi křižovatkami. Tímto by se dala eliminovat kompletně jedna neprůjezdná trasa a vozidla, která by nově vjížděla do systému, by s tímto faktem

počítala a byla by jim vygenerována alternativní trasa.

Dá se i do tohoto algoritmu jistým způsobem implementovat predikce. Z dat vozidel je známo odkud kam vozidlo jede a tím je známo i to, že vozidla se mohou shluknout v určitou dobu na určitém místě a nějakým způsobem na to reagovat přesměrováním jednotlivých vozidel na jinou trasu. Zde je diskutabilní jakým způsobem nechat zasahovat systém dopravy do toho kam chceme jet. Některá vozidla mohou preferovat čas a některá ujetou vzdálenost. Podle toho by se dala i jednotlivá vozidla odklonit na alternativní trasu.

3.10 Třída Main

Třída Main je třída, ze které se volají ostatní třídy a která vše řídí. Obsahuje tyto proměnné viz. Obr. 22.



Obr. 22: Třída Main UML Diagram

Je zde definována mapa simulace, voláno načtení mapy a načtení vozidel ze souboru. Jsou zde definovány parametry statického řešení simulace a třída main slouží i jako spustitelná třída pro získání výsledků ze statického řešení řízení semaforů s pevným nastavením časů. Doba simulace je nastavena na 7500 jednotek, hodnota je mírně zvýšena, a je i ošetřena tak, aby se simulace neukončila v případě, že je zpoždění tak velké, že všechna vozidla nedojela do cíle své cesty. Dále se zde nachází, jak je vidět v ukázce kódu procházení seznamu eventů a testování, zda je čas eventu s nejnižším časem shodný s hodnotou aktuálního času. Pokud se zde nějaký event nachází, je proveden a odstraněn ze seznamu eventů (Kód 8).

Kód 8: Práce s časem

```
1.  while(simulation.getTime() < 7500 || simulation.isMoveCarEventInEvents()){
2.            while(simulation.getFirstEvent() != null && simulation.getFirstEvent
                  ().getTime() == simulation.getTime()){
3.                  simulation.getFirstEvent().make();
4.                  simulation.getEventList().remove(simulation.getFirstEvent());
5.             }
6.                 simulation.incrementTime();
7.  }
8.  simulation.getEventList().clear();
```

Ve třídě main je také procházení vozidel, která dojela do cíle své cesty, přičítání čekacích časů vozidel a přičítání doby vozidla na cestě a tisk výsledků do souboru, jak je vidět v následující ukázce kódu (Kód 9).

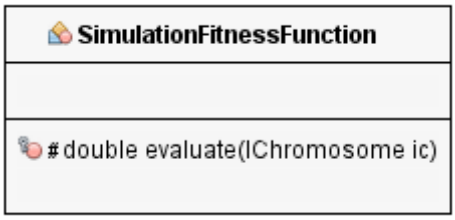
Kód 9: Zaznamenávání času vozidel

```
1.  for(Car car : simulation.getFinishedCars()){
2.                  waitingTime += car.getWaitTime();
3.                  onWayTime += car.getEndTime() -
     car.getStartTime();
4.  }
5.  Main.printResult(simulation, false);
```

Dále jsou zde velmi rozsáhlé metody načítání ze souborů a výpisu výsledků do souboru. Jedná se o dvě metody na čtení ze souboru a konkrétně metoda readCars, která načítá .TXT soubory obsahující balík vozidel pro požadovanou simulaci a metoda readMap, která načítá ze souboru mapu, jejíž struktura byla popsána v kapitole "Třída Map". Další metoda, kterou zde najdeme je pro zápis výsledků do souboru. Ten se provede po každé jedné úspěšné simulaci.

3.11 Třída SimulationFitnessFunction

Třída SimulationFitnessFunction obsahuje pouze proměnnou evaluate typu double viz. Obr. 23



Obr. 23: Třída SimulationFitnessFunction UML Diagram

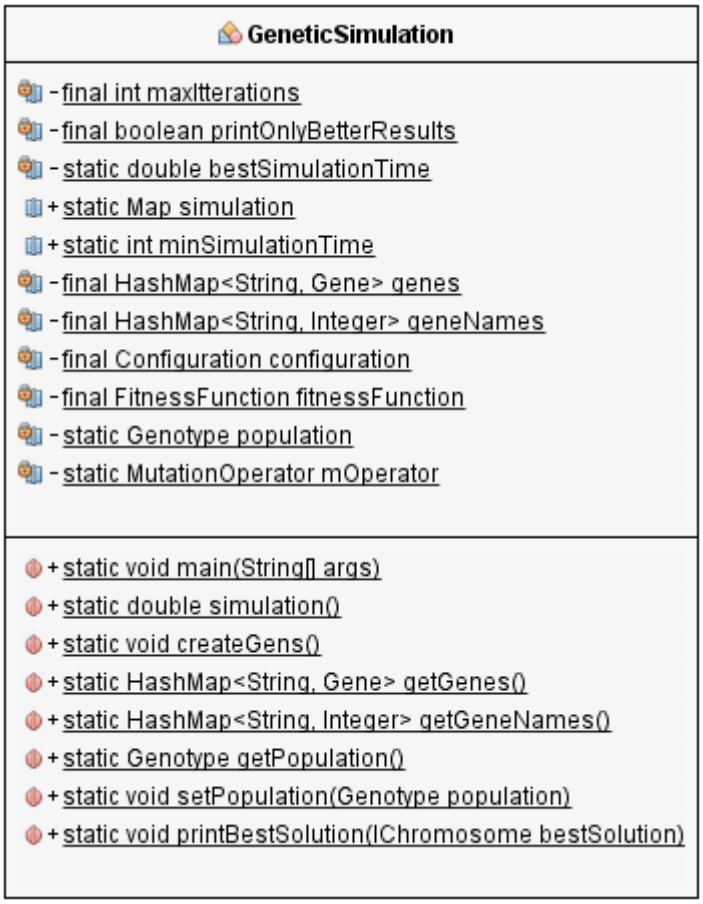
Jedná se pouze o pomocnou třídu, která dědí ze třídy FitnessFunction a podle ní je posuzována vhodnost nalezeného výsledku (Kód 10).

Kód 10: Fitness funkce

```
1.  public class SimulationFitnessFunction extends FitnessFunction
2.  {
3.      @Override
4.      protected double evaluate(IChromosome ic) {
5.          return 2147483647 - GeneticSimulation.simulation();
6.      }
7.  }
```

3.12 Třída GeneticSimulation

Třída GeneticSimulation obsahuje proměnné zobrazené v UML Diagramu viz. Obr. 24.



Obr. 24: Třída GeneticSimulation UML Diagram

Ve třídě je implementován optimalizační genetický algoritmus. Jedná se o aplikaci knihovny JGAP, která je volně dostupná. Nachází se zde deklarace klíčových proměnných pro chod algoritmu. Konkrétně jsou zde proměnné maxItterations, což je proměnná, kterou nastavíme počet iterací algoritmu, dále proměnná populationSize, to je proměnná, kde je určena velikost populace, ze které se budou vybírat vzorky pro genetický algoritmus a které budou selektovány, rekombinovány a mutovány s ostatními vzorky. Zbylé proměnné jsou hraniční hodnoty, které si může genetický algoritmus stanovit na semaforu (Kód 11).

Kód 11: Nastavení genetického algoritmu

```
1.  private static final int maxItterations = 50, populationSize = 100, minGreenPe
    riod = 1, maxGreenPeriod = 30, minRedPeriod = 1, maxRedPeriod = 30,
2.  minDelay = 0, maxDelay = 0;
3.  public static int minSimulationTime = 7500;
```

Je potřeba si také definovat "překladové tabulky", díky těm je poznat, jaká hodnota časového intervalu byla na kterém semaforu nastavena viz. ukázka kódu (Kód 12).

Kód 12: Překladové tabulky

```
1.  private static final HashMap<String, Gene> genes=new HashMap<String, Gene>();
2.  private static final HashMap<String,Integer>geneNames=new HashMap<String,Integ
    er>()
```

Pro vytvořené řešení postačí výchozí nastavení genetického algoritmu knihovny JGAP. Konkrétně se jedná o selektor, který vybírá 90% nejlepší populace a klonuje ji z celkové velikosti populace, křížení je náhodné s poměrem k velikosti populace*0.35. Mutace je náhodná a použita na jednom ze dvanácti genů v celé populaci (Kód 13). Nastavení fitness funkce, která určí kvalitu současného řešení. V tomto případě není jednoznačný výsledek, pouze nejnižší průměrná doba čekání vozidla. Musíme porovnávat čas jednotlivých simulací a nelézt nejnižší průměrný čas čekání ze všech řešení.

Kód 13: Nastavení JGAP knihovny

```
1.  private static final Configuration configuration = new DefaultConfiguration();

2.  private static final FitnessFunction fitnessFunction=new SimulationFitnessFunc
    tion()
```

V následující ukázce kódu je vidět nastavení fitness funkce do konfigurace, vytvoření genu podle mapy, vytvoření pole pro geny, procházení mapy genů načtených metodou GeneticSimulation.createGens(). Dále je zde uložení genů do pomocné mapy pole, vložení indexu genu v poli do druhé pomocné mapy podle jména, vytvoření chromosomu, nastavení chromosomu do konfigurace, nastavení velikosti populace, náhodné nastavení hodnot genu populace a vytvoření MutationOperator dle konfigurace (Kód 14).

Kód 14: Rozšířené nastavení JGAP knihovny

```
1.  configuration.setFitnessFunction(fitnessFunction);
2.  GeneticSimulation.createGens();
3.  Gene[] genesArray = new Gene[genes.size()];
4.  for (Entry<String, Gene> entry : GeneticSimulation.getGenes().entrySet()) {
5.                  genesArray[i] = entry.getValue();
6.                  GeneticSimulation.getGeneNames().put(entry.getKey(), i);
7.                  i++;
8.  }
9.  Chromosome sampleChromosome = new Chromosome(configuration, genesArray);
10. configuration.setSampleChromosome(sampleChromosome);
11. configuration.setPopulationSize(populationSize);
12. GeneticSimulation.setPopulation(Genotype.randomInitialGenotype(configuration)
13. mOperator = new MutationOperator(configuration);
14. System.out.println("Zacatek simulace");
15. population.evolve(maxItterations);
16. IChromosome bestSolution = population.getFittestChromosome();
17. System.out.println("Nejmensi cekaci doba je "+(2147483647 -
                        bestSolution.getFitnessValue()));
18. int k = 0;
19. for(Gene bestGene: bestSolution.getGenes()){
20.         System.out.println("nejlepsi gen: " + k + ", hodnota: " + bestGene.getA
    llele( ));
21.         k++;  }
22. GeneticSimulation.printBestSolution(bestSolution);
```

4 SIMULACE

4.1 Nastavení simulací

Díky dlouhé době simulace byly změřeny výsledky pouze pro hodnoty populace 20, 50 a 100. Ideální nastavení populace by se měla pohybovat okolo 500 [48]. Tato hodnota je doporučována i v dokumentaci knihovny JGAP.

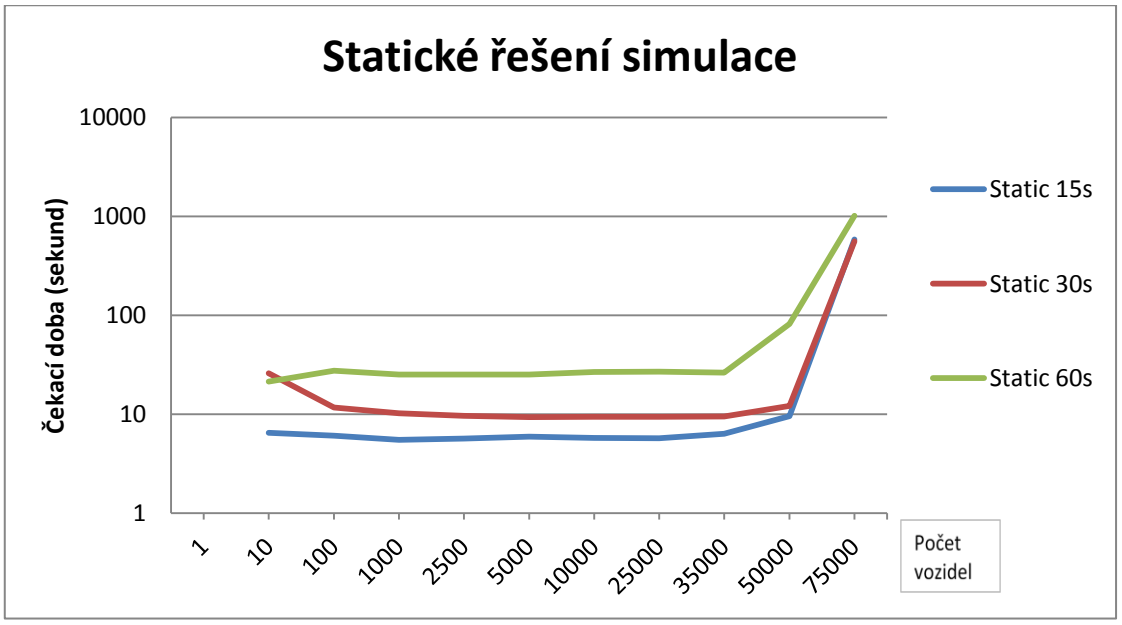
Bohužel simulace běžely po dobu více než čtyř týdnů a výsledků nebylo naměřeno tak velké množství. Pro nastavení populace 20 a iterací 20 je za celou dobu hledání optimálního řešení vygenerováno 732 řešení. Pro nastavení populace 50 a iterací 20 je za celou dobu hledání optimálního řešení vygenerováno 2 315 řešení . A pro nastavení populace 100 a iterací 50 je za celou dobu hledání optimálního řešení vygenerováno 10 670 řešení [49].

4.2 Výsledky statického řešení

Simulačním prostředím postupně projížděly různé počty vozidel za určitého nastavení semaforů. Semafory byly všechny nastaveny na hodnoty 15, 30 a 60 sekund. Jak je vidět v tabulce číslo Tab. 4 a v grafu viz.Obr. 25. V nižším zatížení simulačního prostředí je výhodnější nastavení menších časových intervalů. Se vzrůstající hustotou dopravy dochází k postupnému splynutí časů mezi patnácti a třiceti sekundami. Při maximálním zatížení vychází třiceti sekundová čekací doba dokonce o necelé procento.

Tab. 4: Statické řešení simulace

Vozidel	Static 15 sekund	Static 30 sekund	Static 60 sekund
1	0	0	0
10	6,5	26	21,4
100	6,07	11,69	27,59
1000	5,541	10,232	25,183
2500	5,6824	9,68	25,2096
5000	5,9278	9,3676	25,2426
10000	5,7726	9,4662	26,7124
25000	5,72348	9,41892	26,91668
35000	6,3600855	9,532257	26,433771
50000	9,54788	12,14634	81,51822
75000	584,59106	560,4432	1014,67413



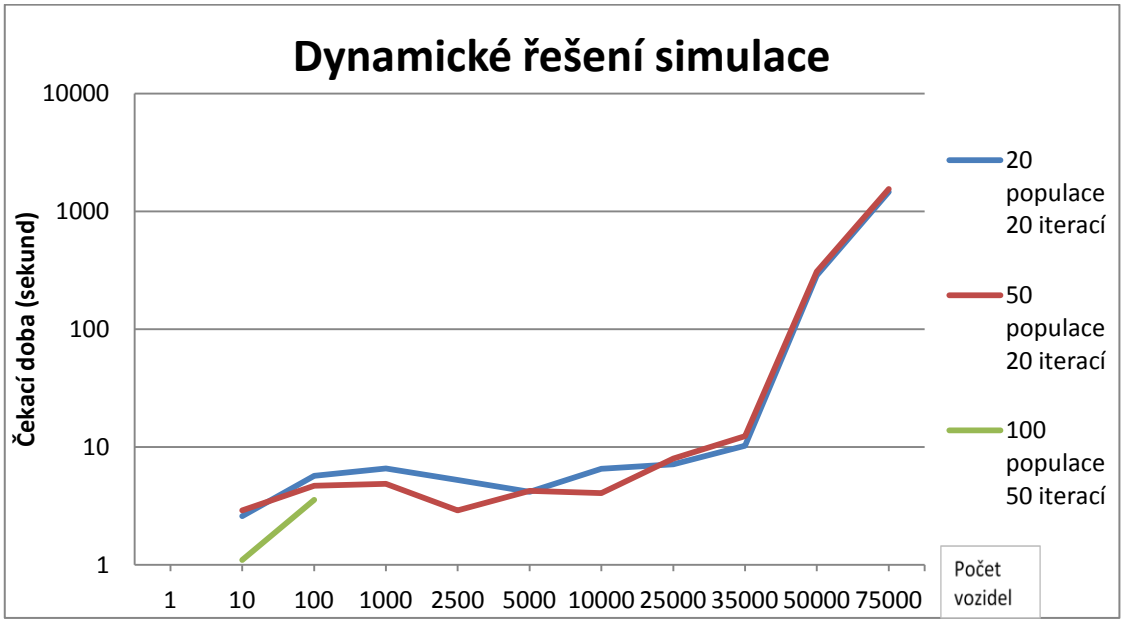
Obr. 25: Graf statického řešení simulace

4.3 Výsledky dynamického řešení

Výsledky simulací ve srovnání se statickým řešením s nastavením semaforů na 30 sekund. V tabulce číslo Tab. 5 a v grafu číslo viz. Obr. 26 je vidět, že dynamické nastavení semaforů s optimalizací v podobě genetického algoritmu je mnohonásobně efektivnější. Při zatížení simulačního prostředí enormním množstvím vozidel se snižuje šance na nalezení nejvhodnějšího řešení díky hledání extrému. Hodnoty vycházejí v různém rozsahu a hledaná nejnižší čekací doba vozidel je v tomto případě extrém. Veškeré výpočty bohužel nebyly provedeny vzhledem k časové náročnosti simulace.

Tab. 5: Dynamické řešení simulace

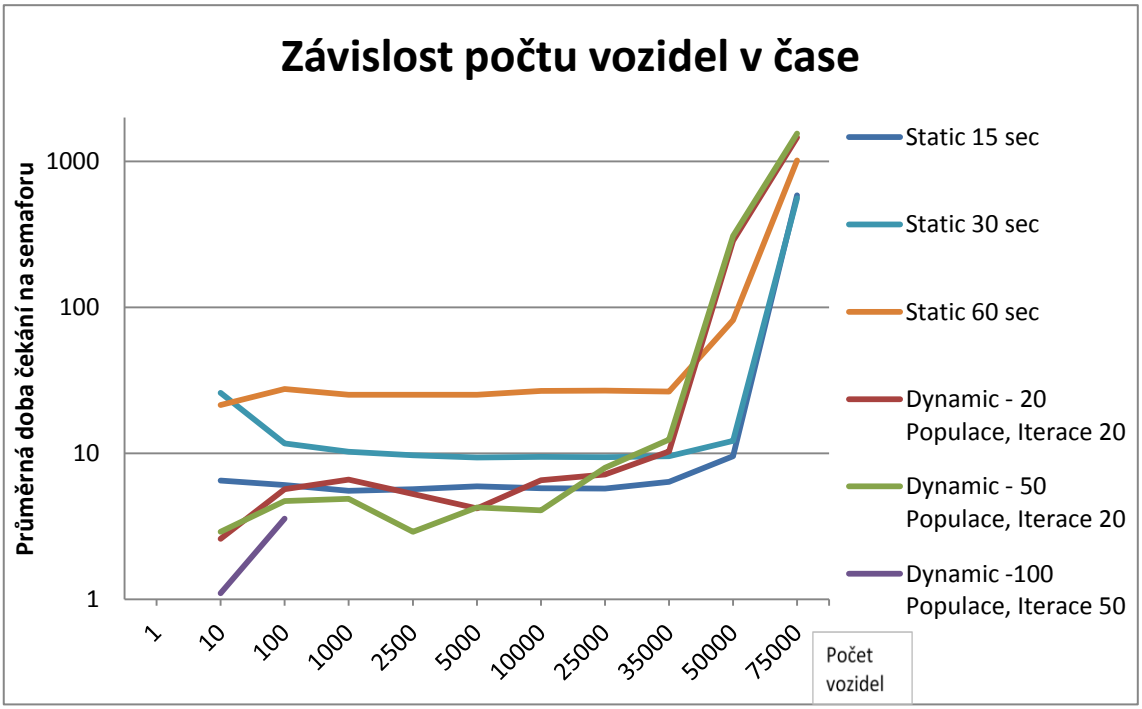
		Iterací 20, populace 20		Iterací 20, populace 50		Iterací 50, populace 100	
Vozidel	Static 30s	Dynamic (s)	Rozdíl (%)	Dynamic (s)	Rozdíl (%)	Dynamic (s)	Rozdíl (%)
1	0	0	0	0	0	0	0
10	26	2,6	900,00	2,9	796,55	1,1	2263,64
100	11,69	5,69	105,45	4,7	148,72	3,57	227,45
1000	10,232	6,59	55,27	4,87	110,10		
2500	9,68	5,25	84,38	2,9036	233,38		
5000	9,3676	4,1826	123,97	4,2602	119,89		
10000	9,4662	6,5338	44,88	4,064	132,93	4,0715	132,50
25000	9,41892	7,159	31,57	7,97408	18,12		
35000	9,532257	10,288743	-7,35	12,415714	-23,22		
50000	12,14634	286,14926	-95,76	308,5694	-96,06		
75000	560,4432	1462,9521	-61,69	1548,3116	-63,80		



Obr. 26: Graf dynamického řešení simulace

4.4 Výsledky simulací

Výsledky zcela neodpovídají teoretickému předpokladu. Podle nalezených nejlepších řešení z hodnot, jak je vidět na grafu viz. Obr. 27, vychází, že dynamické řešení je výrazně lepší v nižších počtech vozidel v některých případech až o stovky procent. Naměřené výsledky v případě simulací s vyšším počtem vozidel vycházejí hůře než simulace statické také o stovky procent. Horší výsledky jsou způsobeny tím, že hledáme extrém funkce, hledáme hraniční hodnotu nastavení. Se vzrůstajícím počtem vozidel klesá pravděpodobnost nalezení nižšího času. Genetický algoritmus nastavuje pro jednotlivé křižovatky časy samostatně, zatímco statický nastaví všechny křižovatky na stejnou hodnotu. Forma výsledků exportovaných do .TXT souboru je v příloze A.



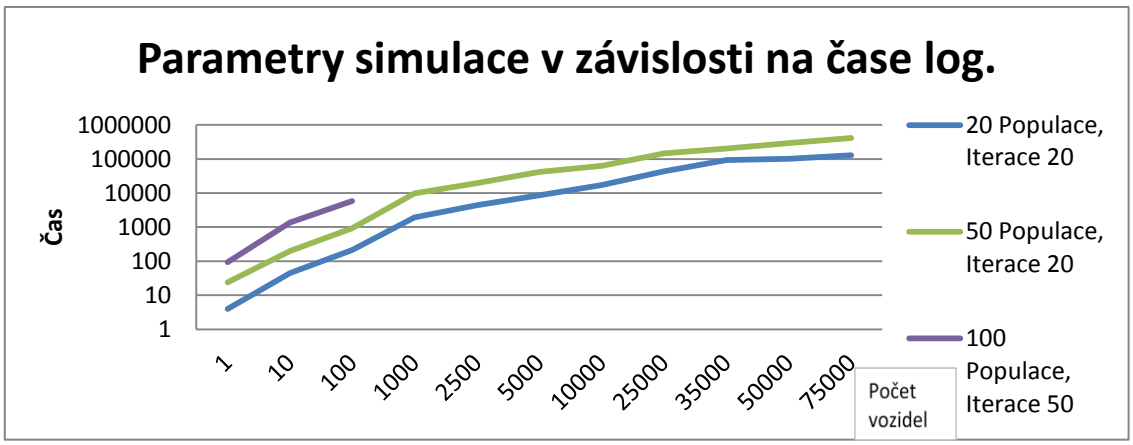
Obr. 27: Závislost počtu vozidel a průměrné doby čekání

4.5 Čas simulace

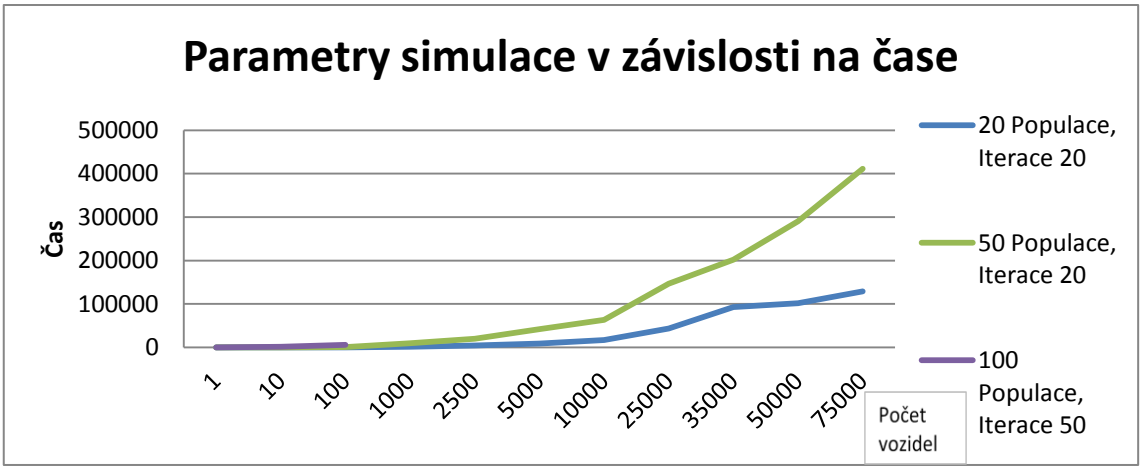
Simulace, respektive optimalizace a optimalizační algoritmus, není napsán z důvodu složitosti jako vícevlákná aplikace, není tedy schopna plně využít vícejádrové procesory. Při vyšších hodnotách parametrů populace například 500 dochází v neúměrnému navýšení času simulace, je tomu tak z důvodu jednovlákného fungování programu. Ten se poté chová tak, že se zaplní RAM do zhruba 3700MB a dochází k ukládání dat na SSD disk, který nedosahuje takových hodnot zápisu a čtení jako paměť RAM a simulace se prodlužuje. Nejdelší simulace trvala více než 114 hodin. Jako rozšíření pro diplomovou práci by rozhodně měl být program napsán jako vícevlákný. Časové závislosti počtu vozidel za jednotku času jsou vidět na grafech Obr. 28 a Obr. 29[45][46][47]. Čas simulace statického řešení se pohyboval v řádu jednotek minut a doba simulace dynamických řešení je vidět v tabulce Tab. 6.

Tab. 6: Doba simulace vzhledem k nastavení genetického algoritmu

Počet vozidel	Doba simulace: Iterací 20, populace 20 (s)	Doba simulace: Iterací 20, populace 50 (s)	Doba simulace: Iterací 50, populace 100 (s)
1	4	24	94
10	45	200	1362
100	214	935	5839
1000	1924	9817	
2500	4401	19808	
5000	8681	42103	
10000	17191	63716	321835
25000	43767	146562	
35000	92944	202271	
50000	101966	290526	
75000	129159	411554	



Obr. 28: Parametry simulace v závislosti na čase logaritmické



Obr. 29: Parametry simulace v závislosti na čase

5 ZÁVĚR

Cílem této práce bylo vytvoření simulačního prostředí, ve kterém půjde nastavit statické řešení řízení v dopravní infrastruktuře. Následně vytvořit optimalizační řešení, které nalezne vhodnější nastavení pro vytvořenou infrastrukturu, poté bude optimalizace porovnána se základním nastavení. Simulační prostředí bylo vytvořeno v podobě čtyř propojených křižovatek.

Na úrovni této infrastruktury byl implementován responzivní algoritmus na nalezení nejkratší trasy. Byly zvoleny statistické hodnoty pro nastavení semaforů ve statickém nastavení světelného signalizačního značení a byly změřeny průměrné čekací doby v závislosti na počtu vozidel v systému.

Poté byl implementován genetický algoritmus, který systematicky mění nastavení jednotlivých semaforů a optimalizuje infrastrukturu za účelem snížení čekací doby vozidel v systému. Genetický algoritmus byl mnohokrát spuštěn a proměřen s různým nastavení jeho klíčových parametrů, výsledky optimalizace byly zaznamenány a vyhodnoceny.

Optimalizační algoritmus byl nastaven podle různých parametrů, a to s nastavením 20 populace a 20 iterací, 50 populace a 20 iterací a nakonec s parametry populace 100 a iterací 50. Z výsledků měření lze vyčíst, že s vyšší populací se zlepšuje nastavení průjezdu vozidel a průměrná doba čekání klesá se vzrůstajícím počtem populace a iterací.

Výsledky optimalizace genetickým algoritmem neodpovídají zcela teoretickému předpokladu, je to dáno tím, že se vzrůstajícím počtem vozidel v systému se snižuje šance na nalezení nejnižšího čekacího času vozidla, protože tato hodnota je v tomto případě extrém ze všech výsledků, které genetický algoritmus nalezne během simulace. Proto je tedy průměrná čekací doba vozidel v simulačním prostředí lepší při vyšším počtu vozidel ve statickém nastavení semaforů, než v optimalizovaném řešení.

Tento problém by vyřešilo nastavení větší populace a více iterací, avšak díky tomu, že program, ani genetický algoritmus není optimalizován pro práci s více výpočetními vlákny, doba této simulace by byla v řádu týdnů.

LITERATURA

[1] Kolik dlužíte kvůli radnici? Podívejte se na žebříček zadlužených měst. TRAMBA, David. *Lidovky.cz* [online]. [cit. 2015-05-16]. Dostupné z: http://byznys.lidovky.cz/jak-hospodari-vase-radnice-podivejte-se-na-zebricek-zadluzenych-ceskych-mest-1s9-/statni-pokladna.aspx?c=A121101_173940_statni-pokladna_mev

[2] Inteligentní město. *http://www.inteligentnimesto.cz/* [online]. 2014 [cit. 2015-05-24]. Dostupné z: http://www.inteligentnimesto.cz/index.php/cz/

[3] EVERS, Marco. Living Lab: Urban Planning Goes Digital in Spanish 'Smart City'. *Living Lab: Urban Planning Goes Digital in Spanish 'Smart City'* [online]. 2013 [cit. 2015-05-24]. Dostupné z: http://www.spiegel.de/international/world/santander-a-digital-smart-city-prototype-in-spain-a-888480.html

[4] Future Age. *Klademe důraz na ekologičnost dopravy* [online]. 2013 [cit. 2015-05-24]. Dostupné z: http://www.futureage.eu/rozhovory/detail/546-klademe-duraz-na-ekologicnost-dopravy

[5] ČTK. Ve strahovském velíně technici připravují připojení Blanky. *Ve strahovském velíně technici připravují připojení Blanky* [online]. 2014 [cit. 2015-05-24]. Dostupné z: http://www.financninoviny.cz/zpravy/ve-strahovskem-veline-technici-pripravuji-pripojeni-blanky/1115956

[6] CROSS S.R.O., Řízení dopravy. *http://www.cross.cz/* [online]. [cit. 2015-05-24]. Dostupné z: http://www.cross.cz/download/brochure/CROSS_BRO_TrafficControl_1405_CZ.pdf

[7] POUZAR, Vladimír. Vozidlové detektory - typy, rozdělení, funkce. *Vozidlové detektory - typy, rozdělení, funkce* [online]. 2010 [cit. 2015-05-24]. Dostupné z: http://www.svsmp.cz/svetelna-signalizace/vozidlove-detektory-typy-rozdeleni-funkce.aspx

[8] MLČÁK, Tomáš a Václav VRÁNA. TRANSFORMÁTORY. In: *TRANSFORMÁTORY* [online]. Ostrava, 2006 [cit. 2015-05-24]. Dostupné z: http://fei1.vsb.cz/kat420/vyuka/Bakalarske_FMMI/Prednasky/9_trafa_sylab_bc_06.pdf

[9] BUREŠ, Petr; PŘIBYL, Ondřej. *Detektory zasahující do vozovky, úvod do detekce* [online]. Praha: Dopravní fakulta ČVUT v Praze, [cit. 2015-05-24]. Dostupné z: http://zolotarev.fd.cvut.cz/ma/ctrl.php?act=show,file,21577

[10] CAMEA. *Smyčkový rychloměr* [online]. 2008. [cit. 2015-05-24]. Dostupné z: http://camea.cz/underwood/download/files/cam_speed_web.pdf

[11] Smart Eye Traffic Monitoring System. *Europa Environmental* [online]. 2015 [cit. 2015-05-24]. Dostupné z: http://europaenvironmental.com/traffic

[12] ZÁKON O PROVOZU NA POZEMNÍCH KOMUNIKACÍCH. *361/2000 Sb*. 2000. Dostupné také z: http://www.zakonyprolidi.cz/cs/2000-361

[13] MCKEEGAN, Noel. Audi travolution: vehicle to traffic light communication system reduces fuel consumption. *Audi travolution: vehicle to traffic light communication system reduces fuel consumption* [online]. 2010 [cit. 2015-05-24]. Dostupné z: http://www.gizmag.com/audi-travolution-vehicle-to-traffic-light-communication/15340/

[14] ČTK,. Jihlava upravuje světelné křižovatky, zrychlí dopravu. *Jihlava upravuje světelné křižovatky, zrychlí dopravu* [online]. 2012 [cit. 2015-05-24]. Dostupné z: http://zpravy.aktualne.cz/regiony/vysocina/jihlava-upravuje-svetelne-krizovatky-zrychli-dopravu/r~i:article:745380/

[15] MĚSTO ZLÍN. *Preference a plošná koordinace MHD ve Zlíně* [online]. 2014. 2014 [cit. 2015-05-24]. Dostupné z: http://www.zlin.eu/preference-a-plosna-koordinace-mhd-ve-zline-cl-850.html

[16] CROSS s.r.o. *Signální plán* [online]. [cit. 2015-05-24]. Dostupné z: http://www.cross.cz/images/products/traffic-cross-ptc.gif

[17] ŽELEZNÝ, Richard. PREFERENCE PROVOZU VEŘEJNÉ DOPRAVY JE VÝZNAMNOU FUNKCÍ ZDRAVÉHO ROZVOJE MĚST. *PREFERENCE PROVOZU VEŘEJNÉ DOPRAVY JE VÝZNAMNOU FUNKCÍ ZDRAVÉHO ROZVOJE MĚST* [online]. 2007 [cit. 2015-05-24]. Dostupné z: http://www.railway2007.fd.cvut.cz/proceedings/Zelezny.pdf

[18] AUTONOVINKY,. Londýn snižuje limit emisí pro vjezd do centra. *Londýn snižuje limit emisí pro vjezd do centra* [online]. 2013 [cit. 2015-05-24]. Dostupné z: http://www.autoweb.cz/londyn-snizuje-limit-emisi-pro-vjezd-do-centra/

[19] FOLPRECHT, Jan. *PREFERENCE MĚSTSKÉ HROMADNÉ DOPRAVY* [online]. 2005 [cit. 2015-05-24]. Dostupné z: http://kds.vsb.cz/mhd/preference-priklady.htm

[20] Preference vozidel IZS. *Preference vozidel IZS* [online]. 2013 [cit. 2015-05-24]. Dostupné z: http://www.jtsczsro.cz/sluzby/preference-vozidel-izs

[21] Piezoelektrické jevy. *Piezoelektrické jevy* [online]. 2014 [cit. 2015-05-24]. Dostupné z: http://fyzika.fs.cvut.cz/subjects/fzmt/lectures/FZMT_11.pdf

[22] Vážení vozidel za jízdy v České republice. *Vážení vozidel za jízdy v České republice* [online]. 2012 [cit. 2015-05-24]. Dostupné z: http://www.cross.cz/cs/reference/vazeni-vozidel-ceska-republika.html

[23] RENFREW, David, Xiao-Hua YU, Ozgur BASKAN a Soner HALDENBILE. Traffic signal optimization using Ant Colony Algorithm. *The 2012 International Joint Conference on Neural Networks (IJCNN)* [online]. 2012 [cit. 2015-05-17]. DOI: 10.5772/13665. Dostupné z: http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1343&context=eeng_fac

[24] APELTAUER, Jiří. Zvýšení plynulosti dopravy a průjezdní kapacity vozovky v místech s dočasným omezeným průjezdem vozidel na D a R pomocí mobilních kooperativních ITS systémů - Mobilní liniové řízení provozu. *Zvýšení plynulosti dopravy a průjezdní kapacity vozovky v místech s dočasným omezeným průjezdem vozidel na D a R pomocí mobilních kooperativních ITS systémů - Mobilní liniové řízení provozu* [online]. 2014 [cit. 2015-05-24]. Dostupné z: http://www.modo.cdvinfo.cz/file/3-seminar-projektu-modo-prezentace-apeltauer/

[25] BARTER, Paul. Is 30% of traffic actually searching for parking? *Is 30% of traffic actually searching for parking?* [online]. 2013 [cit. 2015-05-24]. Dostupné z: http://www.reinventingparking.org/2013/10/is-30-of-traffic-actually-searching-for.html

[26] Smart thinking. *Smart thinking* [online]. 2013 [cit. 2015-05-24]. Dostupné z: http://www.smartparking.com/media/1016/smartthinking-issue-1.pdf

[27] GIETL, Johanna K., Otto KLEMM, Constantini SAMARA, Ingrid ALFHEIM a Mona MØLLER. Analysis of Traffic and Meteorology on Airborne Particulate Matter in Münster, Northwest Germany. *Journal of the Air* [online]. 2012, 59(7): 85-99 [cit. 2015-05-24]. DOI: 10.1007/978-1-4684-4121-5_6. Dostupné z: http://www.tandfonline.com/doi/pdf/10.3155/1047-3289.59.7.809

[28] OLŠANSKÝ, Milan. Správa vozidel – Inteligentně – telemetricky. *Správa vozidel – Inteligentně – telemetricky* [online]. 2013 [cit. 2015-05-24]. Dostupné z: http://www.automobilrevue.cz/rubriky/truck-bus/praxe/sprava-vozidel-inteligentne-telemetricky_42566.html

[29] *FleetSignal - Remote Traffic Control Systems Monitoring & Telemetry* [online]. [cit. 2015-05-24]. Dostupné z: http://www.ayantra.com/traffic-control-monitoring.html

[30] The Basics of Loop Vehicle Detection. *The Basics of Loop Vehicle Detection* [online]. 2000 [cit. 2015-05-24]. Dostupné z: http://www.marshproducts.com/pdf/Inductive%20Loop%20Write%20up.pdf

[31] City Systems Integration. *City Systems Integration* [online]. 2011 [cit. 2015-05-24]. Dostupné z: https://connect.innovateuk.org/documents/3130726/3794125/Feasibility+Study+-+Stoke-on-Trent+City+Council.pdf/da8c81dd-bd72-4288-95fd-2a22db78a7c9

[32] KUBA,. DISKRÉTNÍ SIMULACE. In: *DISKRÉTNÍ SIMULACE* [online]. Univerzita Karlova, 2013 [cit. 2015-05-24]. Dostupné z: http://kam.mff.cuni.cz/~kuba/vyuka/programovani/xaver/diskretni_simulace.html

[33] KŘIVÝ, Ivan a Evžen KINDLER. *SIMULACE A MODELOVÁNÍ* [online]. OSTRAVSKÁ UNIVERZITA, 2001 [cit. 2015-05-24]. Dostupné z: http://vendulka.zcu.cz/Download/Free/SkriptaKindlerMS.pdf. UČEBNÍ TEXTY OSTRAVSKÉ UNIVERZITY.

[34] FAKULTA STROJNÍ, VŠB-TU OSTRAVA,. Modelování dopravy na pozemních komunikacích. *Modelování dopravy na pozemních komunikacích* [online]. 2009 [cit. 2015-05-24]. Dostupné z: http://projekt150.ha-vel.cz/node/94

[35] Modelling and Simulation of Transportation Systems a Scenario Planning Approach. *Modelling and Simulation of Transportation Systems a Scenario Planning Approach* [online]. 2009 [cit. 2015-05-24]. ISSN 0005-1144. Dostupné z: http://www.academia.edu/601256/Modelling_and_Simulation_of_Transportation_Systems_a_Scenario_Planning_Approach

[36] Traffic Optimization System: Using a Heuristic Algorithm to Optimize Traffic Flow and Reduce Net Energy Consumption.
Traffic Optimization System: Using a Heuristic Algorithm to Optimize Traffic Flow and Reduce Net Energy Consumption [online]. 2009 [cit. 2015-05-24]. Dostupné z: https://keshavsaharia.files.wordpress.com/2011/01/traffic-optimization-system.pdf

[37] HALDENBILEN, Soner, Ozgur BASKAN a Cenk OZ. An Ant Colony Optimization Algorithm for Area Traffic Control. *Ant Colony Optimization - Techniques and Applications* [online]. InTech, 2013 [cit. 2015-05-24]. DOI: 10.5772/51695. ISBN 978-953-51-1001-9. Dostupné z: http://www.intechopen.com/books/ant-colony-optimization-techniques-and-applications/an-ant-colony-optimization-algorithm-for-area-traffic-control

[38] Time Optimization for Traffic Signal Control Using Genetic Algorithm. *Time Optimization for Traffic Signal Control Using Genetic Algorithm* [online]. 2009 [cit. 2015-05-24]. Dostupné z: http://www.researchgate.net/profile/Himakshi_Arora/publication/229029442_Time_Optimization_for_Traffic_Signal_Control_Using_Genetic_Algorithm/links/0c960520f498c9fa63000000.pdf

[39] JIROVSKÝ, Lukáš. MATEMATICKO-FYZIKÁLNÍ FAKULTA. *Http://teorie-grafu.cz/* [online]. 2010 [cit. 2015-05-24]. Dostupné z: http://teorie-grafu.cz/

[40] Algoritmy.net: Dijkstrův algoritmus. *Algoritmy.net* [online]. 2015 [cit. 2015-05-24]. Dostupné z: http://www.algoritmy.net/article/5108/Dijkstruv-algoritmus

[41] *http://www.kiv.zcu.cz/* [online]. KALÁTOVÁ, Eva a Jaroslav DOBIÁŠ. 2000 [cit. 2015-05-24]. Dostupné z: http://www.kiv.zcu.cz/studies/predmety/uir/gen_alg2/E_alg.htm

[42] Multiple Intersections Traffic Signal Timing Optimization with Genetic Algorithm. *Multiple Intersections Traffic Signal Timing Optimization with Genetic Algorithm* [online]. 2011 [cit. 2015-05-24]. Dostupné z: http://www.academia.edu/1767671/Multiple_Intersections_Traffic_Signal_Timing_Optimization_with_Genetic_Algorithm

[43] PETRÁK, Tomáš. *Inteligentní řízení města za pomocí telemetrických sítí*. Brno, 2014. Dostupné také z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=86147. Diplomová práce.

[44] C# and Java: Comparing Programming Languages. *C# and Java: Comparing Programming Languages* [online]. 2013 [cit. 2015-05-24]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms836794.aspx

[45] GREGORY, Keith. Understanding OutOfMemoryError. *Understanding OutOfMemoryError* [online]. 2014 [cit. 2015-05-24]. Dostupné z: http://www.kdgregory.com/index.php?page=java.outOfMemory

[46] Chapter 17. Threads and Locks. *Chapter 17. Threads and Locks* [online]. 2015 [cit. 2015-05-24]. Dostupné z: https://docs.oracle.com/javase/specs/jls/se7/html/jls-17.html

[47] STANEK, Michal. Genetic Approach to Optimize Traffic Flow by Timing Plan Manipulation. *Genetic Approach to Optimize Traffic Flow by Timing Plan Manipulation* [online]. 2006 [cit. 2015-05-24]. Dostupné z: http://www.researchgate.net/publication/224674628_Genetic_Approach_to_Optimize_Traffic_Flow_by_Timing_Plan_Manipulation

[48] MEFFERT, Dr. Klaus. Getting Started With JGAP. *Getting Started With JGAP* [online]. 2015 [cit. 2015-05-24]. Dostupné z: http://jgap.sourceforge.net/doc/tutorial.html

[49] MEFFERT, Dr. Klaus. JGAP. *JGAP* [online]. 2015 [cit. 2015-05-24]. Dostupné z: http://jgap.sourceforge.net/index.html

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

h	Parametr zátěže cesty v mapě, je využíván algoritmem pro nelezení nejkratší trasy
MAC adresa	Media Access Control jedinečný identifikátor, který má jakékoliv komunikační rozhraní či zařízení.
Event	jedná se o pojem využívaný v diskrétních simulacích. Event je námi požadovaná událost, kterou simulujeme.
Fitness funkce	jedná se o funkci v genetických algoritmech porovnávající výsledky nových mutací s předcházející generací.
Selekce	výběr
Rekombinace	vyjmutí jednotlivých částí chromozomu genetického algoritmu a skombinování s jiným chromozomem.
Mutace	s malou pravděpodobností jednoduchým způsobem náhodně mění hodnotu jednotlivých genů.
Konvergence	charakteristiky vyvíjející se populace genetického algoritmu se s časem blíží k požadovaným vlastnostem.
Stochastické algoritmy	algoritmy využívající náhodných jevů a možností.
Deterministické algor.	vždy ze stejných výchozích (vstupních) podmínek svým během vytvoří stejné výsledky.

SEZNAM PŘÍLOH

A VÝSLEDKY SIMULACE

Auto Pocet semaforu: 2 Cas cekani: 0 Doba prujezdu: 75 Cas vjezdu do systemu 807 Cas vyjezdu ze systemu: 882
Auto Pocet semaforu: 2 Cas cekani: 20Doba prujezdu: 97 Cas vjezdu do systemu 1315 Cas vyjezdu ze systemu: 1412
Auto Pocet semaforu: 3 Cas cekani: 18Doba prujezdu: 140 Cas vjezdu do systemu 1649 Cas vyjezdu ze systemu: 1789
Auto Pocet semaforu: 3 Cas cekani: 0 Doba prujezdu: 120 Cas vjezdu do systemu 3986 Cas vyjezdu ze systemu: 4106
Auto Pocet semaforu: 3 Cas cekani: 17Doba prujezdu: 140 Cas vjezdu do systemu 4529 Cas vyjezdu ze systemu: 4669
Auto Pocet semaforu: 3 Cas cekani: 0 Doba prujezdu: 116 Cas vjezdu do systemu 4608 Cas vyjezdu ze systemu: 4724
Auto Pocet semaforu: 3 Cas cekani: 7 Doba prujezdu: 128 Cas vjezdu do systemu 5780 Cas vyjezdu ze systemu: 5908
Auto Pocet semaforu: 2 Cas cekani: 0 Doba prujezdu: 60 Cas vjezdu do systemu 6186 Cas vyjezdu ze systemu: 6246
Auto Pocet semaforu: 3 Cas cekani: 8 Doba prujezdu: 124 Cas vjezdu do systemu 6396 Cas vyjezdu ze systemu: 6520
Auto Pocet semaforu: 2 Cas cekani: 0 Doba prujezdu: 60 Cas vjezdu do systemu 6996 Cas vyjezdu ze systemu: 7056

Pruh v A z AIO1 do B, C, AIO2 Cas cekani: 4
Pruh v A z AIO2 do B, C, AIO1 Cas cekani: 0
Pruh v A z B do C, AIO1, AIO2 Cas cekani: 0
Pruh v A z C do B, AIO1, AIO2 Cas cekani: 0
Pruh v B z BIO1 do A, D, BIO2 Cas cekani: 0
Pruh v B z BIO2 do A, D, BIO1 Cas cekani: 8
Pruh v B z A do D, BIO1, BIO2 Cas cekani: 1
Pruh v B z D do A, BIO1, BIO2 Cas cekani: 7
Pruh v C z CIO1 do A, D, CIO2 Cas cekani: 20
Pruh v C z CIO2 do A, D, CIO1 Cas cekani: 0
Pruh v C z A do D, CIO1, CIO2 Cas cekani: 0
Pruh v C z D do A, CIO1, CIO2 Cas cekani: 0
Pruh v D z DIO1 do B, C, DIO2 Cas cekani: 0
Pruh v D z DIO2 do B, C, DIO1 Cas cekani: 0
Pruh v D z B do C, DIO1, DIO2 Cas cekani: 30
Pruh v D z C do B, DIO1, DIO2 Cas cekani: 0
Pruh v AIO1 z AIO1 do A Cas cekani: 0
Pruh v AIO2 z AIO2 do A Cas cekani: 0
Pruh v BIO2 z BIO2 do B Cas cekani: 0
Pruh v BIO1 z BIO1 do B Cas cekani: 0
Pruh v CIO2 z CIO2 do C Cas cekani: 0
Pruh v CIO1 z CIO1 do C Cas cekani: 0
Pruh v DIO1 z DIO1 do D Cas cekani: 0
Pruh v DIO2 z DIO2 do D Cas cekani: 0

Prumerny pocet semaforu: 2.6 Prumerny cas cekani auta: 7.0
Prumerna doba prujezdu: 106.0 Prumerny cas cekani na pruhu: 2.9166667

B ZDROJOVÝ KÓD

V příloze se nachází zdrojový kód v jazyce Java. Program je okomentován pro jasnou čitelnost kódu. Na DVD, které je k práci přiloženo se nachází dvě verze programu, jedna je okomentovaná, druhá není, dále se zde nachází .pdf verze bakalářské práce a vzorek výsledků jednotlivých měření.