České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

BAKALÁŘSKÁ PRÁCE

Decentralizované řízení dopravní signalizace, nastavení délky cyklu

Praha, 2010 **Autor: Jakub Novotný**

Kapitola 1

AAAAA

Seznamte se se současným stavem řízení dopravní signalizace v Praze v oblasti Zličína, zvláště pak s rozvržením křižovatek, sensorů a signálními plány jednotlivých řadičů. Mikrosimulátor dopravy AIMSUN se běžně používá k srovnávání různých variant dopravního řešení. Seznamte se s tímto nástrojem a se způsobem předávání dat mezi ním a externími aplikacemi pomocí toolboxu vgsapi. Decentralizace řízení dopravy spočívá v zavedení komunikace mezi řadiči sousedících křižovatek, posílání zpráv o současném stavu a vyjednávání o společném postupu pro změny řídícího plánu. Teoretický přístup je též znám jako agentní přístup (multi-agent system). Navrhněte vhodnou komunikační strategii pro alespoň dvě křižovatky jejímž cílem je změna délky cyklů signalizačních plánů jednotlivých křižovatek za účelem zvýšení průjezdnosti oblastí. Tuto strategii implementujte jako rozšíření stávajícího řešení v C. Navrhněte vhodný komunikační protokol a pokud to bude výhodné použijte napojení na knihovnu rozhodovacích algoritmů BDM. Chování navrženého řízení srovnejte s expertně navrženým lokálním řešením pomocí Monte Carlo simulace a mikrosimulátoru AIMSUN.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 2

Úvod

Stav dopravní situace ve městě bezesporu ovlivňuje život každého z nás a pro jeho zlepšení, ať už máme na mnysli výstavbu a zlepšování dopravních komunikací, nebo teoretický výzkum, jsou v dnešní době investovány nemalé finanční prostředky. Optimalizované řízení provozu je jedním z možností, jak odlehčit dopravě bez nutnosti zasahování do fyzického stavu silnic.

V součané době se spoléhá hlavně na ruční nastavení parametrů křižovatky z centrály na základě dat z detektorů. Účelem této práce je prozkoumat možnosti řízení novým způsobem, a to decentralizovaným řízením multiagentím systémem. Pomocí programu v C++, kde jsou agenti ovládající křižovatky reprezentováni objektem, a komunikací přes knihovny AimsunDs s mikrosimulátorem dopravy Aimsun, budeme testovat možnosti výměny údajů, nalezení optimální strategie a změny dopravní situace na křižovatkách.

Nejprve popíšeme mmikrosimulátor Aimsun, použité knihovny a používané parametry. Poté si přiblížíme některé důležité prvky z teorie multiagentních systémů.

V druhé části práce popíšeme běh cel0ho programu, způsob implementace třídy agenta, zpracování údajů a volbu nejbhodnější strategie. Na závěr vyhodnotíme získané výsledky.

Kapitola 3

AIMSUN

AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks), je mikrosimulátor dopravy. Podstatou mikroskopické simulace (mikrosimulace) je modelování jízdy jednotlivých vozidel po dané komunikační síti, přičemž se zohledňují všechny parametry infrastruktury i dopravních prostředků, a to včetně chování řidiče.

3.1 Vstupní data pro AIMSUN

AIMSUN potřebuje pro svůj běh simulační scénář a množinu simulačních parametrů, které definují experiment. Scénář se skládá ze čtyř druhů dat: popis dopravní sítě, plány řízení dopravy, požadovaná dopravní data, a plány hromadné dopravy.

3.1.1 Scénář

Popis dopravní sítě zahrnuje geometrii sítě, popis křižovatek a rozmístění detektorů, které jsou rozmístěny podál dopravní sítě.

Plány řízení doprahy obsahují fáze a jejich délky pro křižovatky, které jsou řízeny dopravními světly. V každé fázi je definováno která signální skupina je průjezdná.

dopravní data se dají zadat dvěma způsoby:

- Pomocí objemů dopravy v určitých místech dopravní sítě, poměrů odbočení a počátečního stavu

- Pomocí matice, kde prvek na i-tém řádku a v j-sloupci udává kolik jízd se uskuteční z místa i do místa j

V prvním případě se vozidla rozmístí stochasticky podle pořžadovaných počtů a poměrů odbočení do dopravní sítě, v případě druhém je každému vozidlu přiřazena trasa z místa i do místa j.

Plány hromadné dopravy obsahují linky autobusů, jejich zastávky a jízdní řády.

3.1.2 Výstupní data AIMSUNu

AIMSUN poskytuje jak statistické výstupy v podobě hodnot doby průjezdu vozidla sítí, počtu zastavení, zpoždění a průměrné rychlosti, tak výstupy z detektorů - počet vozidel, rychlost a doba stání na detektoru, tak i plynule modelovaný grafický výstup.

Obrázek 3.1: Grafický výstup ze simulátoru AIMSUN

Obrázek 3.2: Grafický výstup ze simulátoru AIMSUN - detail

3.1.3 VGS API

VGS API je rozhraní napsané v jazyce, které rozšiřuje funkčnost mikrosimulátoru AIMSUN. Jeho dva nejdůležitější úkoly jsou zjednodušení reálné simulace a statistické zpracování výstupních dat.

3.1.3.1 Reálná simulace

Aimsun umožňuje simulaci reálné dopravní sítě, kdy hodnoty hustoty dopravy odpovídají skutečně naměřeným hodnotám v reálném prostření simulované sítě. Běžně se používají ručně sečtené hodoty v intervalech jedné hodiny. Údaje se potom vkládají opět ručně do simulátoru.

Pro přesnější simulaci se používají data z detektorů v simulované oblasti. Objem dat je ale pro ruční vkládání neúnosně velký. Proto bylo vyvinuto VGS API, které umožňuje generování vozidel v průběhu simulace. VGS API se předá soubor s uloženými údaji z detektorů, to automaticky rozběhne AIMSUN, do kterého generuje v průběhu simulace vozidla podle reálných dat.

3.1.3.2 Zpracování dat

Druhým důležitám úkolem VGS API je zhromažďovat data potřebná pro vyhodnocení experimentu. Aimsun sice disponuje jednoduchým rozhraním pro vizualizaci dat a jejich export do textových souborů, není ale možné například porovnávat jednotlivé scénáře simulací. VGS API proto periodicky ukládá všechny klíčové ukazatele jak pro jednotlivé segmenty dopravní sítě, tak i pro celý simulovaný systém. Uživatel má po ukončení simulace k dispozici údaje o počtu zastavení vozidla, o jeho zpoždění, průměrné rychlosti, době jízdy a době stání, o dopravím toku a hustotě dopravy na jednotlivých segmentech.

Jedním z nejpodstatnějších údajů, které nám VGS API poskytuje, je délka fronty pro daný jízdní pruh. Tento údaj je prakticky nemožný určovat podle dat z detektorů, podstatně zkreslených chybami dvojího typu. Jednak vzniká chyba při přejezdu vozidla z pruhu do pruhu v oblasti detektorů, kdy jedno vozidlo zaznamenají oba detektory. Vozidlo by se potom přičetlo do obou front. K druhému případu chyby dochází, pokud je mezi dvěma vozidly tak malý rozestup, že je detektor považuje za jedno. V tomto případě naopak dochází k samovolnému vytrácení vozidel. Pokud bychom chtěli počítat délku fronty jako rozdíl počtů vozidel, která do křižovatky přijela a která ji opustila, docházelo by ke kumulaci této chyby v průběhu simulace. Hodnota délky fronty se ve VGS API získá vysčítáním vozidel s menší rychlostí než 3,6 km/h po segmentech jízdního pruhu.

3.1.4 Řadiče

K řízení signálních skupin křižovatky se používá tzv. řadič. V reálném případě se jedná o počítač napojený na dopravní ústřednu, signální skupiny křižovatky a její detektory. v případě simulace je použit emulátor řadiče ELS3 firmy ELTODO. Ten má implementován pouze algoritmus řízení. O simulaci detektorů a přepínání signálních skupin se stará AIMSUN.

3.1.5 Oblast simulace

Kapitola 4

Multiagentní systémy

4.1 Úvod

Multiagentí systém je druh distribuované umělé inteligence. Tento systém se skládá z jednotlivých výpočetních prvků, tzv. agentů, které musí mít dvě základní schopnosti. Zaprvé musí být schopni autonomní akce rozhodnutí - zjistit jak nejlépe dosáhnout požadovaných cílů a zadruhé je to schopnost interakce s ostaními agenty. V druhém případě nejde jen o pouhou výměnu dat, ale o typ kolektivní aktivity - návrh, potvrzení, odmítnutí.

4.1.1 Historie

Multiagentí systémy jsou na poli počítačové vědy poměrně novinkou. Studium tohoto tématu probíhá od začátku osmdesátých let devatenáctého století. Větší pozornosti se jim dostalo v polovině let devadesátých s rozvojem internetu.

4.1.2 Agent

Neexistuje obecně uznávaná definice agenta. Přikloníme se k definici použité v publikaci Wooldridge a Jennings(1995).

Definice 4.1 (Agent): Agent je počítačový systém umístěný do nějakého prostředí, který je schopen autonomní akce k přiblížení se navrženým cílům. ▶

Agent v naprosté většině případů nemá celkovou kontrolu nad prostředím. Prostředí může ovlivňovat jen částečně. Obecně, stejně jako v našem případě, je prostředí nedeterministické. Stejná akce provedená dvakrát za sebou nemusí vést ke stejnému výsledku.

4.2 Prostředí

Russel a Norvig navrhli klasifikaci prostředí podle jednotlivých parametrů následovně.

4.2.1 Dostupné vs. nedostupné

Prostředí je dostupné, pokud agent může zjistit jeho úplný stav v kteroukoliv dobu.

4.2.2 Deterministické vs. Nedetrministické

Deterministické prostředí je takové, ve kterém má každá jednotlivá akce předem daný efekt.

4.2.3 Statické vs. dynamické

Statické prostředí se na rozdíl od dynamického mění pouze vlivem akcí vyvolanými agenty.

4.2.4 Diskrétní vs. spojité

V diskrétním prostředí existuje pevné konečné číslo možných vjemů a akcí.

4.3 Interakce agentů

4.3.1 Stavy prostředí a preference agentů

Mějme pro jednoduchost 2 agenty. Označme si je i a j. Předpokládejme, že máme množinu

$$\Omega = \{\omega_1, \omega_2, ...\}$$

obsahující všechny možné stavy prostředí, v kterém agenti operují. Aby byl agent schopen efektivně ovlivňovat prostření, musí být schopen ohodnotit, jak je pro něj daný stav příznivý. Hodnocení daného stavu agenta i a j formálně definujeme jako funkce

$$u_i : \Omega \to \mathbb{R},$$

$$u_j : \Omega \to \mathbb{R}.$$

Čím je stav ω příznivější pro agenta i, tím je větší hodnota funkce u_i.

Definice 4.2 (Uspořádání na množině všech stavů): Mějme 2 stavy prostředí ω_1, ω_2. Řekněmě, že stav ω_1 je preferován agentem i nad stavem ω_2, pokud platí $u_i(\omega_1) = u_i(\omega_2)$. Značíme

$$\omega_1 \succeq_i \omega_2.$$

Stav ω_1 je silně preferován agentem i nad stavem ω_2, pokud platí $u_i(\omega_1)\, u_i(\omega_2)$. Značíme

$$\omega_1 \succ_i \omega_2$$

Relace $\succeq_i$ je zřejmě uspořádání, protože má všechny potřebné vlastnosti.
Reflexivitu:

$$\forall \omega \in \Omega : \omega \succeq_i \omega$$

Transitivitu:

$$\forall \omega_1, \omega_2, \omega_3 \in \Omega : \omega_1 \succeq_i \omega_2 \wedge \omega_2 \succeq_i \omega_3 \Rightarrow \omega_1 \succeq_i \omega_3$$

Porovnatelnost:

$$\forall \omega_1, \omega_2 \in \Omega : \omega_1 \succeq_i \omega_2 \vee \omega_2 \succeq_i \omega_1$$

Relace $\succ_i$ zjevně nesplňuje podmínky reflexivity.

4.3.2 Akce agentů

Nyní popíšeme, jak mají agenti možnost ovlivňovat prostředí. Opět předpokládejme existenci dvou agentů i a j. Obecně mají různí agenti různou oblast působnosti. Množiny

$$A = \{a_1, a_2, ...\}$$

znázorňují množiny všech akcí, které jsou agenti schopni provézt. Na tyto akce reaguje prostředí přechodem do nějakého stavu $\omega \in \Omega$. Formálně můžeme tento přechod zapsat jako funkci

$$\tau : A \times A \to \Omega.$$

4.4 Strategie

Popišme zde způsob, jak se agent rozhodne pro realizaci určité akce. Agent je nyní schopen ohodnotit, který stav prostředí je pro něj příznivější než jiný, neví však jak budou

reagovat ostatní agenti, není schopen určit tudíž, i za předpokladu, že by systém byl deterministický, do jakého stavu systém přejde. K výběru optimální akce se používají prvky z teorie her. Zadefinujme nejprve v souladu s touto teorií základní pojmy.

Definice 4.3 (Dominance množiny): Mějme 2 podmnožiny $\Omega_1, \Omega_2 \subset \Omega$. Řekneme, že Ω_1 je pro agenta i dominantní nad množinou Ω_2, pokud platí

$$\forall \omega \in \Omega_1, \forall \omega' \in \Omega_2 : \omega \succeq_i \omega'.$$

Řekneme že Ω_1 je pro agenta i silně dominantní nad množinou Ω_2, pokud platí

$$\forall \omega \in \Omega_1, \forall \omega' \in \Omega_2 : \omega \succ_i \omega'.$$

Abychom používali terminologii teorie her, budeme nyní akce $a_i \in A$ jednotlivých agentů nazývat strategiemi.

Definice 4.4 (Množina výsledků): Nazvěme množinu všech stavů, do kterých může prostředí přejít při hraní strategie $a_i \in A$, množinou možných výsledků. Označme ji

$$a_i^* \subset \Omega.$$

Definice 4.5 (Dominance strategie): Řekneme, že strategie a_i je dominantní nad strategií a_j, pokud je množina a_i^* dominantní nad množinou a_j^*. Strategie a_i je silně dominantní nad strategií a_j, pokud je množina a_i^* silně dominantní nad množinou a_j^*. ▶

Racionálně uvažující agent tedy vyloučí všechny strategie a_i, jestliže existuje strategie a_j, která nad strategií a_i silně dominuje. K zůžení výběru zbývajících strategií slouží Nashova Rovnost. Pro zjednodušení uvažujme 2 agnenty, i a j. Dvě strategie, a_1 a a_2 jsou v Nashově rovnosti, pokud za předpokladu že agent i zvolí strategii a_1, je nejvýhodnější strategií pro agenta j je strategie a_2 a zároveň pokud agent j zvolí strategii a_2, je pro agnta i nejvýhodnější strategií a_1.

4.4.1 Použití pro výběr délky cyklu

Délka cyklu řadiče křižovatky je parametr, který je pro všechny křižovatky ve skupině společný. Nesmí tedy dojít k situaci, kdy by každý agent nastavil jinou délku cyklu. Množina strategií $A = \{a_1, a_2, ...\}$ je tedy v našem případě množinou všech nastavitelných délek cyklu Tc_k v dané situaci. Mějme funkci $u_{i_{Tc}} : A \to \mathbb{R}$, jejíž funkční hodnota

v bodě Tc_j udává předpokládaný zisk pro agenta i při délce cyklu Tc_j. Funkci $u_i(\omega) = u_i(\tau(Tc_k, Tc_l))$ můžeme zadefinovat následovně (pro jednoduchost předpokládejme existenci dvou agentů i a j)

$$u_i(\tau(Tc_k, Tc_l) = \begin{cases} u_{i_{Tc}}(Tc_k), & k = l \\ -\infty, & k \neq l \end{cases},$$

kde hodnota $-\infty$ vyjadřuje jakýsi kolabs systému při nastavení různých délek cyklu. To však znamená, že žádná strategie není sinlně dominantní nad jinou. Zároveň za předpokladu, že agent i zvolí strategii Tc_l, agent j nemůže udělat lépe, než že zvolí stejnou strategii. To znamená že pro všechny strategie $Tc_l \in A$ platí, že jsou v Nashově rovnosti samy se sebou. Takto definovaná funkce nám v souladu s teorií zaručí, že agenti nenastaví různé délky cyklu, potřebujeme ale dodatečné kritérium kterou délku cyklu vybrat.

4.4.2 Globálně nějlepší řešení

Komunikace agentům dovoluje vzájemně si předat předpokládané zisky pro určitou délku cyklu. Nejprve musíme určit množinu možných délek cyklu tak, aby v každém kroku simulace byla pro všechny agenty společná. Určíme tedy přirozená čísla n a dTc. Množina možných strategií A bude

$$A = \{Tc_{-n}, Tc_{-n+1}, ..., Tc_{n-1}, Tc_n\},$$

$$Tc_i = Tc + i \cdot dTc,$$

kde Tc je délka cyklu v minulém kroku simulace. Každý agent je pak schopen sestavit množinu součtů zisků

$$U = \{u_{Tc_{-n}}, u_{Tc_{-n+1}}, ..., u_{Tc_{n-1}, u_{Tc_n}}\},$$

kde

$$u_{Tc_i} = \sum_k u_{k_{Tc}}(Tc_i).$$

Agent poté zvolí takovou délku cyklu Tc_i, pro kterou platí

$$u_{Tc_i} = \max(U).$$

Toto je výběr globálně nejlepšího řešení, kde agent upřednostní takový čas délky cyklu, u kterého se předpokládá největší součet zisků od všech agentů nad časem, u kterého předpokládá největší zisk pro sebe.

Kapitola 5

Popis implementace

5.1 Třída Lane

Každá instance třídy Lane reprezentuje jeden dopravní pruh. Po inicializaci se do příslušných proměnných uloží textové řetězce, podle kterých se přiřazují k pruhu data z detektorů a AIMSUNu, jak je, pro naše účely důležitá, maximální délka fronty na dopravním pruhu za jednu simulační periodu.

5.2 Třída LaneHandler

Tato třída zprostředkovává přístup agentovi k údajům z jednotlivých pruhů. Stará se o zaznamenávání a statistické spracování dat z AIMSUNu. Také je zde implementován odhad čekací doby automobilu do průjezdu křižovatkou, což je údaj používaný v hodnotící délky cyklu funkci agenta.

5.2.1 Fronta

Agent předává třídě LaneHandler údaj o maximální délce fronty za vzorkovací periodu. Jedná se o největší počet automobilů v příslušném jízdním pruhu, jejichž rychlost reřesahuje určitou hodnotu. V našem konkrétním případě je tato mezní rychlost nastavena na 3,6 km/h. Tento údaj je však pro naše potřeby zkreslený, a to hlavně tím, že ve většině případů se délka cyklu neshoduje se vzorkovací periodou. Uvažujme jízdní pruh s poměrem délky zelené 0,5. Pokud bude například délka cyklu 180 sekund, a budeme

li předpokládat, že začátek jedné periody se bude shodovat s časem, kdy na příslušném semaforu naskočí zelená, bude v první vzorkovací periodě pruh průjezdný po dobu 45 sekund, ale v další nebude průjezdný vůbec a fronta se za stejné hustoty provozu zvětší.

5.2.2 Odhad fronty

K odhadu fronty je použito metody tzv. exponencionálního zapomínání. Metoda vychází z povoucího průměru Který je vhodný pro odhad nekonstantní hodnoty. Označíme si veličinu $\overline{q_i}$ jako plovoucí průměr fronty v kroce i, kam budeme ukládat posledních n měření. Označme si velikost fronty naměřenou v i-tém kroce jakoq_i. V kroce $i+1$ je hodnota hodnota plovoucího průměru přepřičytením poslední naměřené hodnoty rovna

$$\overline{q_i} = \frac{1}{n} \sum_{j=0}^{n-1} q_{i-j}.$$

Obsahuje tedy všech posledních n naměřených hodnot. $\overline{q_{i+1}}$ se tedy rovná

$$\overline{q_{i+1}} = \frac{1}{n} \left(n\overline{q_i} - q_{i-n+1} + q_{i+1}\right) = \overline{q_i} - \frac{1}{n} q_{i-n+1} + \frac{1}{n} q_{i+1}.$$

Abychom nemuseli ukládat posledních n měření. Zjednodušíme algoritmus tím, že místo kroku $i-n+1$ odečteme od průměru jeho hodnotu podělenou délkou průměrovacího okna n. Výpočet přejde na tvar

$$\overline{q_{i+1}} = \frac{1}{n} \left(n\overline{q_i} - \overline{q_i} + q_{i+1}\right) = \overline{q_i} + \frac{1}{n} \left(q_{i+1} - \overline{q_i}\right).$$

$\frac{1}{n} \left(q_{i+1} - \overline{q_i}\right.$ je vlastně přírustek v kroku $i+1$, označme si ho jako Δq_{i+1}, a faktor $1/n$ můžeme chápat jako jeho váhu w. Po dosazení dostáváme jednoduchý tvar

$$\overline{q_{i+1}} = \overline{q_i} + w\Delta q_{i+1}.$$

Průměr s exponenciálním zapomínáním se nazývá proto, že je v něm v i-tém kroce uložena i první hodnota, má však zanedbatelnou váhu $\left(\frac{n-1}{n}\right)^i$. Jedná se o jaksi zjednodušenou verzi Kalmanova filtru. Grafy 5.1 a 5.2 znázorňují rozdíl mezi okamžitou a filtrovanou délkou fronty při váze $w = 0.2$. Graf 5.1 pro konstantní délku cyklu 80 sekund, graf 5.2 pro délku cyklu peridicky se měnící od 40 do 120 sekund.

Obrázek 5.1: Porovnání okamžité a filtrované délky fronty
délka cyklu 80 s

Obrázek 5.2: Porovnání okamžité a filtrované délky fronty
délka cyklu 40-120 s

5.2.3 Odhad čekací doby

Představme si, že ke přižovatce přijíždí auto a_n a mezi tímto autem a křižovatkou je n dalšíchch aut. Budeme chtít odhadnout střední hodnotu času do průjezdu auta a_n křižovatkou $< t_{wn} >$. Nejprve odhadněmě $< t_{wn} >$ v řípadě, že počet aut před ním je roven 0. Pokud přijede v době, kdy je signální skupina ve stavu zelená, projede okamžitě. Pokud ne, čeká do konce fáze. Označme si délku cyklu křižovatky T_c a poměr doby, kdy pro příslušný pruh svítí zelená jako r_g. Předpopkládejme rovnoměrné rozdělení, hustota pravděpodobnosti příjezdu auta v čase t je tedy

$$\omega(t) = \begin{cases} \frac{1}{T_c}, & 0 < t < T_c \\ 0, & jinak \end{cases} .$$

Střední hodnotu t_{w0} tedy spočteme jako

$$< t_{w0} >= \int_0^{T_c} \omega(t) t_{w0}(t) dt = \int_0^{T_c(1-r_g)} \frac{1}{T_c}\, tdt + \int_{T_c(1-r_g)}^{T_c} \frac{1}{T_c}\, 0dt = \frac{1}{T_c} \int_0^{T_c(1-r_g)} tdt = \frac{1}{2}\, T_c(1-r_g)^2$$

Výraz $r_g T_c$ představuje jakousi délku časového okna, kdy auto projede bez čekání. Pokud chceme zjistit střední hodnotu času průjezdu libovolného auta $< t_{wn} >$, musíme toto okno zmenšit o dobu, kterou trvá průjezd křižovatkou autům před ním. Každý pruh křižovatky je schopen propustit maximální počet aut za sekundu. Tento údaj se nazývá saturovaný tok, budeme ho značit s_s, a bere se jako konstanta rovnající se $0.5 auto/s$. Pro n-té auto se tedy okno nulového čekání zkrátí na $r_g T_c - \frac{n}{s_s}$ a střední hodnota doby průjezdu křižovatkou se změní na

$$< t_{wn} >= \frac{1}{T_c} \int_0^{T_c(1-r_g)+\frac{n}{s_s}} tdt = \frac{1}{2T_c} \left(T_c(1-r_g) + \frac{n}{s_s} \right)^2 .$$

To však platí pouze když stihnou všechna auta křižovatkou projet, tedy pokud

$$r_g T_c < \frac{n}{s_s} .$$

jestliže tato nerovnost není splněna, vozidlo nestihne projet na první zelenou a dobu čekání si prodlouží o další cyklus, přičemž počet aut stojících před ním se sníží o $(r_g T_c) s_s$. Počet cyklů, které musíme připočítat k čekací době se rovná dolní celé části z podílu čekajících aut a počtu aut, které projedou na zelenou za jeden cyklus, tedy

$$\left\lfloor \frac{n}{r_g T_c s_s} \right\rfloor ,$$

a do horní meze intrgrálu potom přispěje pouze počet aut, který zbyl po odečtení těch, které zcela pokryla fáze se zeleným světlem. Po zahrnutí tohoto předpokladu se výraz pro $< t_{wn} >$ změní na

$$< t_{wn} >= \left\lfloor \frac{n}{r_g T_c s_s} \right\rfloor T_c + \frac{1}{2T_c} \left(T_c(1 - r_g) + \frac{n - \left\lfloor \frac{n}{r_g T_c s_s} \right\rfloor r_g T_c s_s}{s_s} \right)^2 .$$

Počet vozidel, která projedou na zelenou, však není přesně přímo úměrný délce zelené. Musíme počítat, že jich projede o něco méně vlivem pomalých reakcí řidiče, časových ztrát při rozjezsu kolony a podobně. Tyto vlivy započteme tak, že si zavedeme malý kladný parametr δ, o který snížíme dobu trvání průjezdnosti křižovatky. Výraz pro dobu průjezdnosti se potom změní na

$$(r_g T_c - \delta)$$

a povzorec počet cyklů, po jejichž celý čas stráví vozidlo čekáním je roven

$$\left\lfloor \frac{n}{(r_g T_c - \delta) s_s} \right\rfloor .$$

Konečná podoba výrazu pro střední hodnotu t_{wn} je tedy

$$< t_{wn} >= \left\lfloor \frac{n}{(r_g T_c - \delta) s_s} \right\rfloor T_c + \frac{1}{2T_c} \left(T_c(1 - r_g) + \frac{n - \left\lfloor \frac{n}{(r_g T_c - \delta) s_s} \right\rfloor r_g T_c s_s}{s_s} \right)^2 .$$

Tento výpočet je implementován jako metoda *getEcarWT*, pijímající dva paramentry. Délku cyklu *tc* a pořadí auta *n*.

```
double getEcarWT ( const double tc, const int n ) {
        //gr - pomer casu zelene - atribut tridy
        //ss - saturovany tok - konstanta
        // pocet aut, ktera projedou za zeleneou
        double cpg = tc * gr * ss;
        // number of waiting cycles
        double wc = floor( n / cpg );
        double ET_last_cycle = ( 1/ (2*tc) )
                * ( tc*(1-gr) + (n-wc*cpg)/ss )
                * ( tc*(1-gr) + (n-wc*cpg)/ss );
        double ET = wc*tc + ET_last_cycle;
        return ET;
}
```

Pro hodnocení výhodnosti se používá funkce *getWT*, která sčítá všechny čekací časy pro auta ve frontě

```
double getWT ( const double tc ) {
        double sumEWT = 0;
        int n = getAverageQueueLength();
        for ( int i = 0; i <= n; i ++ ) {
                sumEWT += getEcarWT( tc, i );
        }
        return sumEWT;
}
```

5.2.4 Odhad parametru δ

Delta vyjadřuje korekční parametr doby průjezdnosti křižovatky. Jak jsme již ukázali v kapitole 5.2.3, počet vozidel, které projedou křižovatkou za jeden cyklus řadiče je

$$(r_g T_c - \delta) s_s.$$

To znamená že pokud je fronta větší, než $(r_g T_c - \delta) s_s$, mměla by se její velikost za dobu T_c změnit o

$$\Delta q_{T_c} = n_{Tc} - (r_g T_c - \delta) s_s,$$

kde n_{Tc} je počet vozidel, přijíždějících na křižovatku za dobu T_c. Tento údaj je vlastně výstup z detektorů, který je nám však dostupný pouze jednou za simulační cyklus, trvající 90 sekund. Protože pracujeme s filtrovanými hodnotami front, nedojde k velké chybě, pokud výraz přepočítáme na 90 sekund do podoby

$$\Delta q_{90} = n_{90} - \frac{90}{T_c} (r_g T_c - \delta) s_s = n_{90} - 90 r_g s_s - \frac{90 s_s}{T_c} \delta,$$

z čehož si parametr δ vyjádříme jako

$$\delta = \frac{T_c}{90 s_s} (n_{90} - \Delta q_{90}) - r_g.$$

Jak údaje o projíždejících vozidlech za simulační cyklus n_{90}, tak parametry δ, jsou filtrovány způsobem, použitým na odhad délky fronty a popsaným v kapitole 5.2.2.

5.3 Třída agenta

5.4 Hlavní smyčka

Hlavní smyčka programu se stará o synchronizaci všech potřebných parametrů a provádění iterací. Před započetím opakování kroku simulace nastaví podle konfiguračního souboru počáteční parametry mikrosimulátoru dopravy AIMSUN, jako jsou délka cyklu a fází řadičů, inicializuje jednotlivé agenty a zavolá jejich metody fromSettings a Validate. Také inicializuje instanci třídy Logger pro logování a AIMSUNDs potřebnou pro výměnu dat s AIMSUNem.

5.4.1 Krok simulace

Délka kroku simulace je pevně nastavena na 90 sekund shodně s periodou sběru dat z AIMSUNu. V jednom kroku se nejprve načtou data z AIMSUNu do vektoru glob_dt, který se následně předá každému z agentů jako paramentr jejich metody adapt. Následně zprostředkuje komunikaci mezi agenty tak, že naplní frontu zpráv voláním metod broadcast jednotlivých agentů. Z této fronty postupně poté odebírá zprávy, které předává agentům parametrem jejich metody receive, dokud není fronta prázdná. Tento cyklus se opakuje, dokud agenti komunikují, nebo dokud se nedosáhne předem stanoveného počtu opakování. Po ukončení komunikace naplní vektor výstupů glob_ut voláním metod act hodotami parametrů pro AIMSUN, vše zaloguje a vyčká do dalšího kroku.

Kapitola 6

Výsledky

Měření byla prováděna v oblasti popsané v kapitole 3.1.5 při použití dvou scénářů s konstantními hustotami provozu a jedním scénářem se záznamem reálné situace o délce 24 hodin. Pro porovnání sloužily simulace s pevně nastavenou délkou cyklu, a to hlavně na 80 sekund, neboť se tato hodnota používá na obou křižovatkách při absenci zásahů z dopravní centrály. Porovnávají se hodnoty počtů zastavení, doby průjezdu a doby zastavení vozidel zprůměrované přes krátký časový úsek, ktrý je násobkem kroku simulace. Pro zpracování výsledků byly použity výstupy z VGS API popsané v kapitole 3.1.3.

6.1 Konstantí scénář 1

6.2 Konstantí scénář 2

6.3 Reálný scénář

Literatura