České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská

Výzkumný úkol

Teorie a aplikace multiagentního řízení dopravy

Vypracoval: Jakub Novotný
Vedoucí práce: Dr. Ing. Jan Přikryl, Ph.D.
Konzultant: Ing. Václav Šmídl, Ph.D.

Praha, 2012

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

Praha, 8. července 2010 Jakub Novotný

Rád bych poděkoval vedoucímu práce Ing. Janu Přikrylovi, Ph.D. a odbornému konzultantovi Ing. Václavu Šmídlovi, Ph.D. za odborné vedení a úsilí vynaložené při tvorbě této práce.

Název práce:

Teorie a aplikace multiagentního řízení dopravy

Autor: Jakub Novotný

Obor: Inženýrská informatika
Zaměření: Tvorba softwaru
Druh práce: Výzkumný úkol

Vedoucí práce: Dr. Ing. Jan Přikryl, Ph.D.

Abstrakt:

V této práci je ...

Klíčová slova:

multiagentní systémy, decentralizované řízení, řízení dopravy

Title:

Theory of multiagent traffic control

Author: Jakub Novotný

Abstract:

This work represents ...

Key words:

multiagent systems, decentralized control, control of traffic lights

Zadání práce s podpisem děkana

Vypracujte přehled literatury zabývající se použitím multi-agentních systémů v řízení dopravy. Zvláštní pozornost věnujte řízení v městském provozu s množstvím křižovatek. Z použitých přístupů sestavte přehled používaných metod autonomního řízení a komunikovaných informací mezi agenty. Z dostupných metod zvolte takovou, která je v vhodná pro použití v oblasti Zličína a implementujte ji v prostředí řízení mikrosimulátoru AIMSUN. Výsledné řízení zvolenou metodou srovnejte se současným řídícím algoritmem navrženým dopravními experty.

Obsah

Kapitola 1

Úvod

Účelem této práce je prozkoumat možné způsoby decentralizovaného řízení dopravy, či metody s ústředním řídícim prvkem a jejich možnost decentralizace, vybrat jednu vhodnou metodu pro řízení dvou křižovate v oplasti Praha - Zličín, tuto metodu naimplementovat a otrestovat na simulátoru AIMSUN.

K decentralizovanému řízení bude použit tzv. multiagentní systém. Multiagentní systémy jsou dnes rychle se rozvijející formou decentralizované umělé inteligence a s úspěchem se aplikují na celou řadu problémů, kde není možné použít centrální řízení a je zapotřebí komunikace, koordinace a spolupráce.

V této práci si nejdříve definujeme základní pojmy, a poté se hlouběji podíváme na problém interakce agentů. Popíšeme zde různé způsoby výběru vhodných strategií. Prozkoumáme některé způsoby učení agentů, které pomáhají zlepšovat jejich reakce na neustále se měníci prostředí a na chování ostatních.

Po výběru nejvhodnější metody ji modifikujeme do vhodné formy pro naše účely, kde agent představuje řídící jednotku jedné signální skupiny a ovládá nastavení délky cyklu, tedy periodu, za kterou se vystřídají všechny fáze nastavení senaforů. Implementace bude provedena v jazyce C++ a s pomocí knihoven BDM, VGS API a IT++.

Metoda bude testována pomocí mikrosimulátoru dopravy AIMSUN a porovnávána se dvěma konstantními scénáři a se záznamem skutečné dopravní situace modelované oblasti. Výsledky simulace, tedy výstupní data ze simulátoru, budou zpracovány v prostředí MATLAB a pomocí jeho nadstavby VGS Toolbox, kde porovnáme několik zkoumaných dopravních parametrů, jako je doba průjezdu sítí, průměrná rychlost, počet zastavení a jiné.

Kapitola 2

Matematické metody rozhodování

2.1 Multiagentní systémy

Multiagentní systém je druh distribuované umělé inteligence, jehož základem je agent - výpočetní prvek schopný autonomní reakce, komunikace a koordinace. Každý agent je schopen vyhodnotit optimální chování v dané situaci, které závisí i na akoci ostatních agentů. Ke sploupráci jednotlivých agentů slouží různe metody vejednávání a predikce chování.

2.1.1 Historie

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

2.1.2 Agent

Obecně uznávaná definice agenta neexistuje. Jedna z možných je uvedena například v publikaci [16]:

Definice 2.1 (Agent): Agent je počítačový systém umístěný do nějakého prostředí, který je schopen autonomní akce k přiblížení se navrženým cílům. ▶

2.1.3 Druhy prostředí

Způsob práce agentů se liší podle druhu prostředí, ve kterém pracují. Podle [16] se prostředí dají klasifikovat následovně:

- Deterministické vs. nedeterministické
- Dostupné vs. nedostupné
- Statické vs. dynamické

Deterministické prostředí je takové, ve kterém má každá jednotlivá akce předem daný efekt. Prostředí je dostupné, pokud agent může zjistit jeho úplný stav v kteroukoliv dobu. Statické prostředí se na rozdíl od dynamického mění pouze vlivem akcí vyvolanými agenty. V diskrétním prostředí existuje pevné konečné číslo možných vjemů a akcí.

2.1.4 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 2.2 (Uspořádání na množině všech stavů): Mějme 2 stavy prostředí ω_1, ω_2. Řekněme, že stav ω_1 je preferován agentem i nad stavem ω_2, pokud platí $u_i(\omega_1) \geqq u_i(\omega_2)$. 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:

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

Tranzitivitu:

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

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.

2.2 Výběr strategie podle teorie her

Nyní, jak je rozebíráno v [16], popíšeme, jak mají agenti možnost ovlivňovat prostředí. Opět předpokládejme existenci dvou agentů i a j. Obecně mají různí agenti různou oblast působnosti. Množina

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

znázorňuje množinu 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

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

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 2.3 (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 2.4 (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 2.5 (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 zúžení výběru zbývajících strategií slouží Nashova rovnost. Pro zjednodušení uvažujme 2 agenty, i a j

Definice 2.6 (Nashova rovnost): 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 agenta i nejvýhodnější strategií a_1. ▶

2.3 Zpětnovazebné učení

Zde popíšeme metodu řízení, využívající zpětnovazebného učení. Nejprve zadefinujeme základní pojmy a nastíníme formalismus, na kterém je tato metoda postavena a jejíž použití v decentraizovaném řízení dopravy je popsáno v kapitole 3.1.

2.3.1 Markovův rozhodvací proces

Markovův rozhodvací proces je alternativní metoda sloužící k volbě strategií odhadem zisků z nich plynoucích do budoucna. Modeluje prostředí jako množinu stavů a akcí, které je možné provést a mění stav prostředí s určitou pravděpodobností. Čas je zde vnímán v diskrétní formě kroků a prostředí jako částečně známé, částečně ovlivnitelné a do jisté míry měnící se náhodně.

Účelem tohoto procesu je najít funkci $\pi(s) : S \rightarrow A$ která každému stavu prostředí jednoznačně přiřadí akci maximalizující zisky do určitého bodu v budoucnosti, teoreticky do nekonečna. V [15] je Markovův rozhodovací proces definován takto:

Definice 2.7 (Markovův rozhodovací proces): Markovův rozhodovací proces je definován jako uspořádaná čtveřice

$$(S, A, P, R)$$

kde jsou

- S - množina stavů prostředí
- A - množina možných akcí
- $P(i, a, j) := p(s_{t+1} = j | s_t = i \wedge a = a_t)$ - pravděpodobnost, že při aplikování akce a za stavu i přejde systém do stavu j
- $R(i, a, j)$ - okamžitý užitek přechodu systému ze stavu i do stavu j při akci a ▶

2.3.2 Dynamické programování

Dynamické programování je proces sloužící k nalezení optimální strategie dané $\pi(s)$. K tomu slouží funkce očekávaných kumulativních diskontních zisků $V^{\pi}(i) :\rightarrow \mathbb{R}$ reprezentující předpokládané dlouhodobé zisky při určité $\pi(s)$. Agent se tedy snaží zvolit strategii tak, aby funkci V^{π} při daném počátečním stavu $s_0 = i$ maximalizoval. V [15] je tato funkce definována násedovně:

Definice 2.8 (V-funkce):

$$V^{\pi}(i) = \sum_{k=0}^{\infty} E(\gamma^k R(s_k, \pi(s_k), s_{k+1}))$$

při $s_0 = i$ kde jsou

- $\gamma \in < 0, 1 >$ diskontní faktor určující snižování významnosti odhadu s narůstajícím časem
- E operator odhadu ▶

Tento zápis je spíše formální, proto se tato funkce v [15] definuje pomocí pomocí Q-funkce $Q^{\pi}(i, a)$, funkce odhadující zisk při aplikování akce a při stavu a.

Definice 2.9 (Q-function):

$$Q(i,a)^{\pi} = \sum_{j \in S} P(i,a,j)(R(i,a,j) + \gamma V^{\pi}(j))$$

kde jsou

$$V^{\pi}(i) = \max_{a} Q^{\pi}(i,a)$$

,

$$\pi(i) = \arg\max_{a} Q^{\pi}(i,a)$$

Postupným iterativním přepočítáváním V, Q a π dostaneme Bellmanovu rovnici optimality definovanou v [3] jako:

Definice 2.10 (Bellmanova rovnice optimality):

$$Q(i,a)^{*} = \sum_{j \in S} P(i,a,j)(R(i,a,j) + \gamma V^{*}(j))$$

kde funkce Q a V nezávisí na strategii π. Bellmanova rovnice optimality říká, že pokud jsou Q a V optimální, nemění se s dalšími kroky iterace. V praxi se stanoví nějaká dostatečně malá konstanta ϵ, a pokud se v dalším kroku Q a V nezmění o vícejak ϵ, považujeme je za optimální. Navíc bylo dokázáno, že vždy existuje právě jedna dvojice dvojice těchto funkcí, která je optimální a vždy ji lze tedy iterací určit.

Algoritmus pro určení funkce π tedy iterativně přes všechny stavy, akce a časové kroky upravuje funkce V, Q a π podle rovnic 2.9, a v [15] je v krocích popsán takto:

1. Inicializace počátečních hodnot funkcí V a Q
2. Opakovat kroky 3-5 do dosažení podmínek optimality
3. Upravit hodnotu Q-funkce podle rovnice

$$Q(i,a) = \sum_{j \in S} P(i,a,j)(R(i,a,j) + \gamma V(j))$$

4. dopočítat hodnotu funkce V jako

$$V(i) = \max_{a} Q(i,a)$$

5. nastavit hodnotu funkce *pi* na akci maximálního zisku jako

$$\pi(i) = \arg\max_{a} Q(i,a)$$

2.3.3 Zpětnovazebné učení (Reinforcement learning)

V praxi jsou z počátku hodnoty funkcí P a R popsané v 2.7 neznámé a je potřeba je určit pozorováním. Prakticky se tedy agent musí naučit jak prostředí reaguje na provedení každé akce v určitém stavu. Podle [15] se tedy na začátku procesu nastaví všechny hodnoty na nulu a po každé provedené akci se pozoruje jak prostředí reaguje, tedy hodnota funkce $P(i, a, j)$, a jaký okamžitý zisk z toho plyne, tedy $R(i, a, j)$. K určení těchto hodnot se používají různé algoritmy.

2.3.3.1 Q-učení (Q-learning)

Q-učení je bezmodelová metoda popsaná v [14] a [10]. K určení požadovaných hodnot, jak název napovídá, používá úpravu funkce Q definované v 2.9 v každém časovém kroce t podle rovnice:

$$Q_t(i,a) = \begin{cases} (1-\alpha_t)Q_{t-1}(i,a) + \alpha_t(r_t + \gamma V_{t-1}(i)) & : i = i_t \wedge a = a_t \\ Q_{t-1}(i,a) & : jinak \end{cases}$$

kde $\alpha_t \in < 0, 1 >$ je učící faktor v časovém kroce t a r_t je okamžitý zisk akce a. Funkce P a R popisující prostředí se zde tedy nevyskytují a Q-funkce se zde upravuje přímo z naměřených hodnot za použití snižujícího se učícího koeficientu α.

2.3.3.2 Učení na základě modelu (Model-based learning)

V této metodě, popsané v [15], se modeluje prostředí funkcemi $P(i, a, j)$ a $R(i, a, j)$, které jsou definované v 2.7. Ty jsou z počátku neznámé a jejich hodnoty se určují metodou maximální věrohodnosti podle četnosti naměřených údajů. Agent popsaný v [15] používal následující veličiny:

- $C_i(a)$ - počet, kolikrát agent zvolil akci a ve stavu i
- $C_{i,j}(a)$ - počet, kolikrát prostředí přešlo ze stavu i do stavu j při aplikace akce a
- $R_{i,j}(a)$ - součet okamžitých zisků při použití akce a ve stavu i a následném přechodu do stavu j

Model maximální věrohodnosti (MLM) je v [15] definován jako:

Definice 2.11 (MLM):

$$\hat{P}(i,a,j) = \frac{C_{i,j}(a)}{C_i(a)}$$

$$\hat{R}(i,a,j) = \frac{R_{i,j}(a)}{C_{i,j}(a)}$$

V každém časovém kroce je po aplikaci akce a model přepočítán. Poté je znovu použito dynamické programování popsané v sekci 2.3.2.

2.4 RMM - Rekurzivní modelové metody

Rekurzivní modelová metoda, která byla použita v článku [5] k modelování chování agentů ovládajících ostatní dopravní uzly, slouží k odhadu chování ostatních a gentů. Akce každého agenta z pravidlo ovlivňuje do určité míry celý systém, tudíž výběr strategie každého agenta závisí na předpokládaném chování ostatních agentů. Tato metoda minimalizuje nutnost komunikace a vyjednávání o provedení jisté akce tím, že každý agent je schopen modelovat rozhodnutí ostatních a podle známých parametrů prostředí s určitou pravděpodobností stanovit jejich volbu.

2.4.1 Formální definice

Základním stavebním kamenem RMM je matice zisků agenta, Definovaná v souladu s teorií her v [5]

Definice 2.12 (Matice zisků): Matice zisků P_{R_i} agenta R_i je definována trojicí

$$(R, A, U)$$

kde R je množina všech agentů v systému, A je množina množin $A_j = \{a_1^j, a_2^j, ...\}$ alternativních akcí agenta R_j. A_j budeme nazývat rozhodovací prostor agenta R_j. U je funkce

$$U : A_1 \times A_2 \times ... \times A_n \to \mathbb{R}$$

přiřazující hodnoty zisku všem kombinacím akcí všech agentů. ▶

Každý agent provádí danou akci z nějakého důvodu. Zisky agenta R_i jsou spojeny s provedením jeho určité akce $a_m^i \in A_i$ za předpokladu, že ostatní agenti $R_j, j \in \{1..n\}/\{i\}$ provedou akci $a_k^j \in A_j$. Matice je tedy n-dimenzionální, kde n je počet agentů v systému, a sestává se z prvků $u^{R_i}_{a_k^1 ... a_l^i ... a_m^n}$, reprezentující zisky v dané situaci.
K určení pravděpodobnosti provedení strategií ostatních agentů se definuje rekursivní modelová struktura. [5]

Definice 2.13 (Rekursivní modelová struktura): Rekursivní modelová struktura RMS_{R_i} agenta R_i je definována jako dvojice

$$(P_{R_i}, RM_{R_i})$$

, kde P_{R_i} je matice zisků definovaná v 2.12 a RM_{R_i} je rekurzivní model 2.14, který je použit k modelování rozhodování ostatních agentů. ▶

Rekursivní model je definován v [5] takto:

Definice 2.14 (Rekursivní model): Rekursivní model MR_{R_i} agenta R_i je definován jako m-tice dvojic [1]

$$MR_{R_i} = ((p_1^{R_i}, M_{\{-R_i\}}^{(R_i,1)}), ..., (p_m^{R_i}, M_{\{-R_i\}}^{(R_i,m)}))$$

kde

$$M_{\{-R_i\}}^{(R_i,k)} = (M_{R_1}^{(R_i,k)}, ..., M_{R_{i-1}}^{(R_i,k)}, M_{R_{i+1}}^{(R_i,k)}, ..., M_{R_n}^{(R_i,k)})$$

představuje jednu z m $(n-1)$-tic rozhodovacích modelů ostatních agentů a p_i^k jeho subjektivní předpokládanou pravděpodobnost. ▶

$M_{R_j}^{(R_i,k)}$ je tedy jeden z možných modelů agenta R_j, který předpokládá agent R_i s pravděpodobností $p_k^{R_i}$. Platí samozřejmě podmínka $\sum_{k=1}^{m} p_k^{R_i} = 1$.
Model $M_{R_j}^{(R_i,k)}$ se dá podle [5] rozdělit do tří forem:

- $IM_{R_j}^{(R_i,k)}$ - Recionální model
- $NM_{R_j}^{(R_i,k)}$ - Neinformovaný model
- $SM_{R_j}^{(R_i,k)}$ - Neracionální model

Recionální model odpovídá tomu, že agent R_i předpokládá o agentovi R_j, že se chová racionálně. V [5] je definován jako

$$IM_{R_j}^{(R_i,k)} = RMS_{R_j}^{(R_i,k)} \ ,$$

s parametry $p_{a_l^j}^{(R_i,k)}$ a $P_{R_j}^{(R_i,k)}$. $P_{R_j}^{(R_i,k)}$ je matice zisků, kterou podle agenta R_i v modelu k agent R_j použije. což je Rekurzivní modelové struktura s k-tými daty, o kterých agent R_i předpokládá, že je agent R_j použije k rozhodování.

[1] $\{-R_i\}$ je zkrácený zápis množiny ostatních agentů $\{R_1, ..., R_n\}/\{R_i\}$

Neinformovaný model vychází z toho, že agent R_i nemá o agentovi R_j žádné informace, tudíž přiřadí každé akci $a_l^{R_j}$ pravděpodobnost $p_{a_l^j}^{(R_i,k)} = \frac{1}{|A_j|}$, kde $|A_j|$ je počet možných akcí agenta R_j, což odpovídá rovnoměrnému rozdělení. $p_{a_l^j}^{(R_i,k)}$ značí pravďěpodobnost, že podle agenta R_i nastane v k-tém modelu agenta R_j akce a_l^j.

Neracionální model odpovídá tomu, že se agent A_j chová iracionálně. Chování se v tompto případě modeluje podle situace pokaždé jinak.

Systém se tedy rekurzivně rozvijí, dokud jsou dostupné informace. Pokud nejsou, rekurze skončí neinformovaným modelem s rovnoměrným rozdělením pravděpodobnosti všech akcí. Rekurze také může skončit iracionálním modelem, což ovšem není v našem případě příliš časté.

2.4.2 Rozhodovací algoritmus

Po zkonstruování rekurzivní modelové struktury se prochází tento systém od konce rekurze, kde jsou pravděpodobnosti známi, z důvodu ukončení rekurzivního rozvijení neinformovaným modelem. K výběru nejpravděpodobnější situace se používá tzv. užitečnosti. [5]

Definice 2.15 (Užitečnost): Užitečnost akce a_m^i podle agenta R_i je definována jako

$$u_{a_m^i}^{R_i} = \sum_{a_q^1 \in A_1} \dots \sum_{a_v^{i-1} \in A_{i-1}} \sum_{a_w^{i+1} \in A_{i+1}} \dots \sum_{a_x^n \in A_n} \left(p_{a_q^1}^{R_i} \dots p_{a_v^{i-1}}^{R_i} p_{a_w^{i-1}}^{R_i} \dots p_{a_x^n}^{R_i} u_{a_q^1 \dots a_m^i \dots a_x^n}^{R_i}\right),$$

kde

$$u_{a_q^1 \dots a_x^n}^{R_i}$$

je prvek matice zisků P_{R_i} a pravděpodobnost $p_{a_o^j}^{R_i}$ je definována jako

$$p_{a_k^j}^{R_i} = \sum_o p_o^{R_i} p_{a_k^j}^{(R_i,o)}$$

.

kde $p_o^{R_i}$ je pravděpodobnost modelu z definice 2.14 a $p_{a_k^j}^{(R_i,o)}$ značí pravďěpodobnost, že podle agenta R_i nastane v o-tém modelu agenta R_j akce a_k^j. V případě, že je algoritmus v bodě racionálního modelu, určí se tato hodnota rekurzivně, pokud je model neinformovaný, je rovna $\frac{1}{|A_j|}$. $p_{a_k^j}^{R_i}$ je tedy součet pravděpodobností dané akce v modelu vyvážený pravděpodobností modelu a $u_{a_m^i}^{R_i}$ se definuje jako součet všech prvků matice zisků, kromě těch, které zahrnují jiné akce agenta R_i, než je a_k^j, vyvažený touto pravděpodobností.

2.5 Bayesovské učení

V této kapitole je naznačena metoda bayesovského učení, což je bodový odhadu parametru rozdělení náhodné veličiny založený na Bayesově větě. Tato metoda byla použita v článku [5] ke zlepšení odhadu pravděpodobnosti uskutečnění se modelového chování ostatních agentů, což bude popsáno v 3.2. Jeho výhodou je využití experimentální i expertní znalosti. V praxi se této metody hojně využívá tam, kde jsou data získávána postupně a zpočátku je nutné parametr odhadnout na základě zkušenosti a nadále ho podle výsledků měření zpřesňovat.

Definice 2.16 (Podmíněná pravděpodobnost): Nechť A a B jsou náhodné jevy. Podmíněná pravděpodobnost jevu A na jevu B, tedy pravděpodobnost že nastane jev A pokud nastane jev B, je definována jako

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Věta 2.1 (Bayesova věta): *Nechť A a B jsou náhodné jevy a platí $P(B) > 0$. Potom*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

V souladu s [6] zavedeme následující značení:

- $y(t) = [y_1, ..., y_T]$ - vetor $T \in \mathbb{N}$ naměřených hodnot, kde $y_t \in \{0, 1\}$ a $y_t = 1$ značí, že výsledek experimentu v čase $t \in \{1, ..., T\}$ dopadlu v souladu s naší hypotézou
- Θ - parametr rozdělení
- $\hat{\Theta}_T$ - odhad parametru při T naměřených hodnotách
- $f(y_t|\Theta)$ - hustota pravděpodobnosti náhodné veličiny v čase t při parametru Θ

2.5.1 Věrohodnostní funkce

K odhadu parametru se v této metodě využívá věrohodnostní funkce, která vyjadřuje pravděpodobnost, že při daném parametru se bude vektor naměřených výsledků bude rovnat teoretické hodnotě. V [6] se tato funkce definuje jako: [2]

[2] Tato definice je poněkud zjednodušena a předpokládá, že výsledek t-tého pokusu nezávisí na předchozích pokusech, ani na počátečních podmínkách před t-tým pokusem.

Definice 2.17 (Věrohodnostní funkce):

$$f(y(T)|\Theta) = \prod_{t=1}^{T} f(y_t|\Theta)$$

Při použití těchtu veličin nabývá Bayesova věta podle [6] tvaru

$$f(\Theta|y(T)) = \frac{f(y(T)|\Theta)f(\Theta)}{f(y(T))}$$

kde jsou

- $f(\Theta|y(T))$ aposteriorní hustota pravděpodobnosti
- $f(y(T)|\Theta)$ Věrohodnostní funkce
- $f(\Theta) = f(\Theta|y(0))$ je tzv. apriorní hustota pravděpodobnosti
- $f(y(T))$ normalizační konstanta ($f(\Theta|y(T))$ se bere jako funkce Θ a parametrů $y(t)$)

Apriorní hustota pravděpodobnosti zde vyjadřuje odhad parametru při nenaměřených datech. Jedná se tedy o expertní odhad, předpokládané chování náhodné veličiny založené na známých faktech a zkušenosti. Aposteriorní hustota pravděpodobnosti představuje pravděpodobnost parametru Θ závislou na naměřených hodnotách náhodné veličiny a její střední hodnota je odhadem tohoto parametru. Tedy platí:

$$\hat{\Theta}_T = \int_{\mathbb{R}} f(\Theta|y(T))d\Theta$$

2.6 LQ řízení

LQ je zkratka slovního spojení linaer-quadratic, tedy lineárně kvadratický. Jedná se obecně o metodu, kdy je systém v diskrétním časovém kroku t popsán vektorem proměnných $x(t) = (x_1(t), ..., x_n(t))$ a my můžeme nastavovat vektor parametrů $u(t) = (u_1(t), ..., u_n(t))$. Metoda je popsána a použita v článku [12] pro nastavování časů zelených na křižovatce. Princip přechodu z času t do $t+1$ je popsán lineárním vztahem

$$x(t+1) = Ax(t) + Bu(t) \ , \tag{2.1}$$

kde matice A a B jsou matice stavů a vstupů vyjadřující odezvu systému vzhledem k $x(t)$ a $u(t)$. Účelem LQ řízení je najít optimální hodnoty $u(t)$ v závislosti na $x(t)$ dané zpětnovazebnou maticí L vztahem

$$u(t) = -Lx(t) \ . \tag{2.2}$$

Optimalita je definována pomocí kvadratického kritéria

$$J = \sum_{t=1}^{\infty} x(t)^T Q x(t) + u(t)^T R u(t) \ , \tag{2.3}$$

kde Q a R jsou diagonální pozitivně semidefinitní matice vah určující významnost jednotlivých členů kritéria. Zpětnovazebná matice L se podle [12] dostane minimalizací kritéria J jako

$$L = (R + B^T P B)^{-1} B^T P A \ , \tag{2.4}$$

kde P je jednoznačné řešení Riccatiho rovnice pro diskrétní časový krok

$$P = Q + A^T (P - PB(R + B^T P B) B^T P) A \ . \tag{2.5}$$

Rovnici a její řešení popisuje například [2].

2.6.1 Minimalizace kritéria na horizontu

Minimalizaci kvadratického kritéria J můžeme omezit da dostatečně dlouhý časový horizont do budoucna. Podobný postup je popsán například v [9]. Kvadratické kritérium ve tvaru

$$J = \sum_{t=0}^{h} x(t)^T Q x(t) + u(t)^T R u(t) \ , \tag{2.6}$$

kde Q a R jsou diagonální pozitivně definitní matice vah, rozepíšeme pomocí vztahu 4.11 a budeme minimalizovat pro jednotlivá t podle $u(t)$. Proměnná $u(h)$ se v kritériu vyskytuje pouze ve členu

$$u(h)^T R u(h) \tag{2.7}$$

a tedy musí být $u(h) = 0$. $u(h-1)$ se vyskytuje ve členech

$$u(h-1)^T R u(h-1) + (x(h-1)^T A^T + u(h-1)^T B^T) Q (Ax(h-1) + Bu(h-1) + I_0) \ , \tag{2.8}$$

což lze pomocí složených vektorů a matic přepsat do tvaru

$$(u(h-1)^T, x(h-1)^T) \begin{pmatrix} B^T \sqrt{Q}^T & \sqrt{R}^T \\ A^T \sqrt{Q}^T & 0 \end{pmatrix} \begin{pmatrix} \sqrt{Q} B & \sqrt{Q} A \\ \sqrt{R} & 0 \end{pmatrix} \begin{pmatrix} u(h-1) \\ x(h-1) \end{pmatrix} \ , \tag{2.9}$$

kde symbolem $\sqrt{Q}$ myslíme matici, pro níž platí $\sqrt{Q}^T\sqrt{Q} = Q$. Matice Q a R jsou pozitivně definitní diagonální matice, takže $\sqrt{Q}$ bude také pozitivně definitní diagonální a její diagonální prvky budou rovny odmocnině příslušných prvků původní matice Q. Složenou matici

$$M_0 = \begin{pmatrix} \sqrt{Q}B & \sqrt{Q}A \\ \sqrt{R} & 0 \end{pmatrix} \tag{2.10}$$

můžeme pomocí QR dekompozice rozložit na součin

$$M_0 = M_R M_Q, \tag{2.11}$$

kde M_Q je horní trojůhelníková matice a M_R je matice ortonormální. Pro ortonormální matici M_R platí

$$M_R^T M_R = I \ , \tag{2.12}$$

neboť její zloupce jsou vzájemně ortogonální a normované na jednotku. Z toho plyne, že součin matic $M_0^T M_0$ vyskytující se v členu 2.9 můžeme pomocí QR rozlkladu převést na tvar

$$M_0^T M_0 = (M_R M_Q)^T M_R M_Q = M_Q^T M_R^T M_R M_Q = M_Q^T M_Q \ , \tag{2.13}$$

kde

$$M_Q = \begin{pmatrix} L_u & L \\ 0 & L_q \end{pmatrix} \ , \tag{2.14}$$

je horní trojůhelníková matice. Člen 2.9 poté přejde na tvar

$$(\Delta C(h-1)^T, x(h-1)^T) \begin{pmatrix} L_q^T & 0 \\ L^T & L_u^T \end{pmatrix} \begin{pmatrix} L_u & L \\ 0 & L_q \end{pmatrix} \begin{pmatrix} u(h-1) \\ x(h-1) \end{pmatrix} \ , \tag{2.15}$$

který můžeme dále upravit na

$$(L_u u(h-1) + L_q x(h-1))^T (L_u u(h-1) + L_q x(h-1)) + x(h-1)^T L_q^T L_q x(h-1) \ . \tag{2.16}$$

Pokud tento člen chceme minimalizovat v proměnné $x(h-1)$, musí nutně platit

$$u(h-1) = -L_u^{-1} L x(h-1) \ , \tag{2.17}$$

tím dostáváme vztah, podle kterého nastavujeme řídící proměnnou u. Nyní stačí získat matice L a L_x pro hodnotu v čase 0. Zbývající nenulová část $x(h-1)^T L_q^T L_q x(h-1)$ lze přepsat pomocí rovnice 2.1 do tvaru

$$x(h-1)^T L_q^T L_q x(h-1) = (Ax(h-2) + Bu(h-2))^T L_q^T L_q (Ax(h-2) + Bu(h-2)) \ , \tag{2.18}$$

kde se vyskytuje $u(t)$ pouze pro $t = h - 2$. Při minimalizaci podle $u(h-2)$ musíme zahrnout i tuto část, člen ve složeném maticovém zápisu tudíž nabyde tvaru

$$(u(h-2)^T, x(h-2)^T) \begin{pmatrix} B^T\sqrt{Q}^T & \sqrt{R}^T & L_xA \\ A^T\sqrt{Q}^T & 0 & L_xB \end{pmatrix} \begin{pmatrix} \sqrt{Q}B & \sqrt{Q}A \\ \sqrt{R} & 0 \\ L_xA & L_xB \end{pmatrix} \begin{pmatrix} u(h-2) \\ x(h-2) \end{pmatrix}. \tag{2.19}$$

Matici

$$M = \begin{pmatrix} \sqrt{Q}B & \sqrt{Q}A \\ \sqrt{R} & 0 \\ L_xA & L_xB \end{pmatrix} = \begin{pmatrix} M_0 \\ L_xA \; L_xB \end{pmatrix} \tag{2.20}$$

opět převedeme QR dekompozicí do horního trojůhelníkového tvaru a dostaneme vztah pro $u(h-2)$. Tento postup se analogicky aplikuje na každé $u(t)$. Matice M_0 je stále stjná, L_x použijeme z předchozího kroku minimalizace.

Kapitola 3

Použití rozhodovacích metod v řízení dopravy

3.1 Použití zpětnovazebného učení

V [15] je popsána simulace používající zpětnovazebné učení na základě modelu popsané v 2.3.3.2. Na rozdíl od ezperimentu popsaného v předchozí kapitole zde autoři zvolili variantu, kde existují 2 druhy agentů: agent-vozidlo a agent-křižovatka. Agent-vozidlo má v každý časový krok následující parametry definující jeho stav, které se ukládají v průběhu celé simulace:

- *node* - dopravní uzel, kde se právě nachází
- *dir* - směr vůči tomuto uzlu
- *place* - pozice ve frontě vozidel
- *des* - místo, kam chce v systému dojet

Algoritmus se snaží minimalizovat celkový čas od vyjetí až po dosažení cíle. Každý agent má ve vyjednávání právo hlasovat o nastavení dopravního uzlu, tedy jestli bude v průjezdném nebo uzavřeném stavu. Pro každý uzel *node* a směr *dir* je to tedy akce g - na uzlu *node*, kde se vozidlo nachází ve směru *dir*, svítí červená - uzel je v tomto směru neprůjezdný, a akce g - uzel je v tomto směru průjezdný. Funkce definované v 2.3.2 se tedy zapisují ve tvarech

$$Q([node, dir, place, des], a)$$

$$V([node, dir, place, des])$$

kde $a \in \{r, g\}$. Optimální akce se v [15] volí podle maxima sumy z

$$Q([node, dir, place, des], r) - Q([node, dir, place, des], g)$$

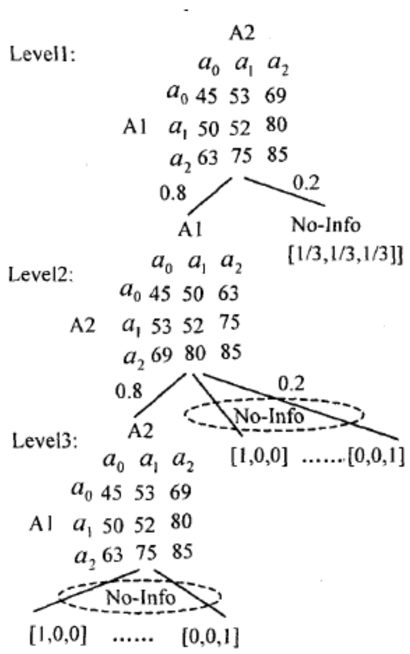
přes všechny vozidla ovlivněná frontou u daného uzlu. Komunikace je zde tedy omezena pouze na výměnu hodnot Q.

3.2 Použití RMM a Bayesova učení v decentralizovaném řízení dopravy

V následujícím textu se budeme zabývat konkrétním využitím rekursivních modelovacích metod v kombinaci s bayesovským učením v decentralizovaném řízení dopravy pomocí multiagentních systémů. Tento koncept byl testován a výsledky zveřejněny v článku [7].

Testy byly prováděny na modelu dvou křižovatek klasického typu, kde každý agent R_1, R_2 ovládal jednu z nich. K modelování chování druhého agenta bylo použito třísyupňové RMM zakončené neinformovaným modelem. Obrázek 3.1 z článku [7] znázorňuje RMM agenta R_1, kde se na každé úrovni agent rozhoduje mezi dvěma modely, informovaným a neinformovaným. Z počátku simulace jsou těmto modelům přiřazeny pravděpodobnosti po řadě 0,8 a 0,2. Tyto pravděpodobnosti jsou upravovány metodou bayesovskeho učení podle akcí provedených druhým agentem. Pro řešení tohoto modelu je použit algoritmus z [5] popsaný v předchozí kapitole.

Určujícím faktorem pro sestavení matice zisků je délka fronty čekajících aut. Tento údaj byl také použit jako klíčový pro srovnání s ostatními metodami při testování. Metoda byla testována za použití simulace psané v jazyce VC++ a porovnávána s dvěma dalšími, a to pevně dané sekvence stanovené dopravními experty a stochastická metoda "Hill-climbing process". Z tohoto srovnání vychází metoda využívající RMM a bayesovského učenís podle výsledků prezentovaných v [7] nejlépe, jak můžeme vidět na obrázku 3.2.



Obrázek 3.1: Tříúrovňová rekurzivní modelová struktura agenta R_1

Traffic flow entered Our Experimental Network (vehicle/hour)	Average waiting car's queue during each cycle		
	M1	M2	M3
<1008	<0.05	<0.05	<0.05
1008	0.1	0.12	0.19
1152	0.15	0.17	0.24
1296	0.26	0.29	0.37
1368	0.34	0.48	0.75
1512	0.47	0.80	1.12
1656	0.77	1.08	1.95
1728	0.91	1.67	2.54
1944	1.23	2.10	3.38
2088	1.94	2.45	5.80
2186	2.58	3.87	6.78
2320	3.56	4.72	8.92
2548	4.03	6.43	11.2
2670	4.85	8.77	16.56
2858	5.60	10.27	*
3016	6.48	13.89	*
3120	7.86	15.73	*
3368	9.22	17.23	*
3420	10.26	*	*
3608	11.22	*	*

Obrázek 3.2: Výsledky měření. M1 - RMM, M2 - pevné cykly, M3 - "Hill-climbing"

3.2.1 Použití LQ řízení ve strategii TUC

LQ řízení bylo použito v [12] k nalezení optimální délky zelených v systému 13-ti signálních skupin. Proměnné $x_i(t)$ zde představují obsazenost ramene i spojující křižovatky M a N. Účelem strategie je nalezení optimální délky zelených g, $g_{N,i}$ značí délku zelené na signální skupiny křižovatky N zprůjezdňující směr do ramene i. Předpokládaný vztah pro přechod systému z času t do času $t+1$ je

$$x_i(t+1) = x_i(t) + T[q_i(t) + s_i(t) + d_i(t) + u_i(t)] \ , \tag{3.1}$$

kde proměnné značí:

- T - časový krok
- $q_i(t)$ - přírůstek vozidel z křižovatek
- $u_i(t)$ - úbytek vozidel do ostatních křižovatek
- $s_i(t)$ - přírůstek vozidel z okolí sítě

- $d_i(t)$ - úbytek vozidel mimo síť

Přírůstek vozidel z křižovatek je dán vztahem

$$q_i(t) = \sum_{k \in I_m} t_{k,i} u_k(t) \ , \tag{3.2}$$

je to tedy součet úbytků vozidel z ramen ústících do křižovatky N vynásobených koeficinety $t_{k,i}$, což jsou odbočovací poměry z ramene k do ramene i. V podovném tvaru se předpokládá $s_i(t)$

$$s_i(t) = t_{i,0} q_i(t) \ , \tag{3.3}$$

kde $t_{i,0}$ je odbočovací koeficient ramene i mimo sledovanou síť. Při délce cyklu C, saturovaném toku S_i a délce zelených $g_{N,i}(t)$ ramene i platí

$$u_i(t) = \frac{S_i \sum g_{N,i}(t)}{C} \ . \tag{3.4}$$

Rovnice 3.1 tedy přechází do tvaru

$$x_i(t+1) = x_i(t) + T \left[(1 - t_{i,0}) \sum_{k \in I_m} t_{k,i} \frac{S_k \sum g_{M,k}(t)}{C} - \frac{S_i \sum g_{N,i}(t)}{C} + d_i(t) \right] \ . \tag{3.5}$$

Uvažujeme-li nominální hodnoty d^n a g^n vedoucí vždy na stav x^n, platí podle rovnice 3.5

$$0 = T \left[(1 - t_{i,0}) \sum_{k \in I_m} t_{k,i} \frac{S_k \sum g^n_{M,k}}{C} - \frac{S_i \sum g^n_{N,i}}{C} + d^n_i \right] \ . \tag{3.6}$$

Označíme-li

$$\Delta g(t) = g(t) - g^n \ , \tag{3.7}$$

můžeme psát rovnici 3.5 jako

$$x_i(t+1) = x_i(t) + T \left[(1 - t_{i,0}) \sum_{k \in I_m} t_{k,i} \frac{S_k \sum \Delta g_{M,k}(t)}{C} - \frac{S_i \sum \Delta g_{N,i}(t)}{C} \right] \ , \tag{3.8}$$

což dovoluje tuto rovnici zapsat pomocí matic v požadovaném tvaru

$$x(t+1) = Ax(t) + B\Delta g(t) \ , \tag{3.9}$$

kde A je jednotková matice.

3.2.1.1 Kvadratické kritérium

Účelem lagoritmu je minimalizovat obsazenost ramen, tedy vektor $x(t)$ a penalizovat změnu délky trvání zelené oproti nominálním hodnotám. Kvadratické kritérium optimálního řízení 2.3 jetedy v [12] definováno vztahem

$$J = \sum_{t=0}^{\infty} x(t)^T Q x(t) + \Delta g(t)^T R \Delta g(t) \; . \tag{3.10}$$

Diagonální matice Q je zde zodpovědná za vyvažování počtu vozidel jednotlivých úseků. V [12] je každý diagonální prvek $Q_{i,i}$ matice Q položen převrácené hodnotě maximálního povoleného počtu vozidel daného úseku i. $R = rI$ penalizuje změnu časů zelených. Parametr r ovlivňuje míru reakce systému a ja volen metodou pokus-oprava. Minimalizací tohoto kritéria pomcí 2.4 získáme zpětnovazebnou matici L, která určuje $g(t)$. Z rovnic 2.2 a 3.7 dostaneme výsledný vztah

$$g(t) = g^n - Lx(t) \; . \tag{3.11}$$

Toto řešení předpokládá, že známe hodnotu g^n, při které systém zůstává ve stavu x^n. Tak tomu ale většinou není. Při absenci znalosti g^n podle [12] odečteme $g(t) - g(t-1)$ a rovnice 3.11 nabývá tvaru

$$g(t) = g(t-1) - L(x(t) - x(t-1)) \; . \tag{3.12}$$

3.3 Zhodnocení

3.3.1 Zpětnovazebného učení

Metode popsaná v článku [15] používá ohodnocovací funkci založenou na parametrech jednotlivých vozidel. Výhodou oproti pojetí, kdy agent představuje pouze signální skupinu, jsou například v tom, že není potřeba odhadovat délku fronty a úloha se celá zjednoduší. Například v publikaci [11] se musí používat k odhadu funkcí V a Q neuronová síť. Navíc tento systém umožňuje i výběr optimální cesty vozidla pro průjezd dopravní sítí. Nevýhodou tohoto pojetí je ovšem značná neuniverzálnost. Už pro počítačové testování tato metode znesnadnuje či úplně znemožňuje použít celou řadu dopravních simulátorů, které jsou pro simulaci po dlouhou dobu optimalizovány a jejichž nasazení značně zjednodušuje práci a urychluje vývoj. Navíc pokud je použit řadič, který obstarává logiku přepínání průjezdnosti a lze nastavovat pouze vnější parametry jako jsou délka cyklu a offset, je metoda, která potřebuje okamžitou změnu signalizace naprosto nevhodná, proto je toto řešení pro reálné nasazení v dnešní době obtížně použitelné. Zapojení některých myšlenek z článku [15] nebo použití zpětnovazevného učení k řešení dílčích problémů by však mohlo přinést zlepšení i do způsobu žešení popsaných v dalších kapitolách.

3.3.2 RMM a Bayesova učení

V článku [5] nenjsou podrobně popsány akce agentů ani způsob, jak hodnotit jejich užitečnost. Proto je tato metoda jen obtížně reprodukovatelná, modifikovatelná či dále rozvinutelná. V naší situaci popsanné v dalších kapitolách také není nutné modelovat chování agentů, neboť je možné ho vykomunikovat pomocí posílaných zpráv. V případě reálného nasazení by však bylo možné vylepšení zapojením RMM pro odhad chování agenta pokud by nastal výpadek spojení nebo podobná situace.

3.3.3 LQ řízení

Kapitola 4

Implementace

V textu	V programu	Název	Popis
$T \in \mathbb{N}$	`T`	Vzorkovací perioda	
$C \in \mathbb{N}$	`Tc`	Délka cyklu	
$S = 0,5$	`ss`	Saturovaný tok	Maximální počet vozidel, která projedou křižovatkou za sekundu
$L \in \mathbb{R}^+$	`L`	Ztrátový čas	Čas potřebný k vyklizení křižovatky mezi dvěma fázemi
$J = \{j_1, ..., j_n\}$		Množina signálních skupin	
$J_A \subset J$		Množina signálních skupin agenta A	Skupiny náležící jednomu agentovi
$I_j \subset J$		Množina vstupů signální skupiny	Signální skupiny ostatních agentů ústících do příslušné skupiny g
$g_j^N \in< 0,1 >$		Nominální poměr zelené	Defi novaný poměr fází
$g_j(t) \in< 0,1 >$		Efektivní poměr zelené	Část z délky cyklu, kdy je signální skupina průjezdná
$i_j(t) \in \mathbb{R}^+$	`i_g`	Hustota vstupů	Počet přijíždějících vozidel do pruhu signální skupiny za jednotku času
$o_j(t) \in \mathbb{R}^+$		Hustota výstupů	Počet vyjíždějících vozidel z pruhu signální skupiny za jednotku času
$q_j(t) \in \mathbb{N}_0^+$		Fronta	Počet vozidel jedoucí menší rychlostí než $3,6km/h$
$\alpha_{j,k} \in< 0,1 >$		Odbočovací poměr z j do k	

4.1 Použitá metoda

Účelem této práce je řídit provoz pomocí nastavení optimální délky cyklu za použití multiagentního systému, kde každý agent ovládá signální skupiny jedné křižovatky a jsou mu dostupné příslušné údaje. Jako vhodná metoda bylo zvoleno LQ řízení, které se snaží minimalizovat fronty a penalizovat délku cyklu.

4.1.1 Přechodové vztahy

Jako údaj popisující stav systému byla zvolena délka fronty $q_j(t)$, což je počet aut v jízdním pruhu jedoucích méně než 3,6 km/h. Pro danou frontu v čase $t+1$ platí zřejmě vztah

$$q_j(t+1) = q_j(t) + T(i_j(t) - o_j(t)), \tag{4.1}$$

kde hustota vstupu $i_j(t)$ je součtem výstupů sousedů pronásobený odbočovacími poměry, případně vstupu do systému $i_{j,0}(t)$, pokud se jedná o koncové rameno řízené oblasti, tedy

$$i_j(t) = i_{j,0}(t) + \sum_{k \in I_j} \alpha_{k,j} o_k(t). \tag{4.2}$$

Křižovtkou z daného jízdního pruhu v průjezdné fázi projede S vozidel za sekundu, kde délka průjezdné fáze závisí na nastaveném poměru a délce cyklu. Před přechodem do nové fáze signálího plánu je potřeba vklidit křižovatku, což vede ke ztrátovému času L, který se projeví vždy jednou za cyklus. Efektivní délka zelené je tedy

$$g_j(t) = (C(t) - L) g_j^N \tag{4.3}$$

a pro hustotu výstupu platí

$$o_j(t) = \frac{S_j g_j(t)}{C(t)} = S_j g_j^N \left(1 - LC(t)^{-1}\right) . \tag{4.4}$$

Rovnice 4.1 tedy přechází do tvaru

$$q_j(t+1) = q_j(t) + T \left(i_{j,0}(t) + \sum_{k \in I_j} \alpha_{k,j} S_k g_k^N \left(1 - LC(t)^{-1}\right) - S_j g_j^N \left(1 - LC(t)^{-1}\right) \right) . \tag{4.5}$$

Za řídící proměnnou tedy vezmeme

$$y(t) = 1 - LC(t)^{-1} . \tag{4.6}$$

takže můžeme rovnici 4.5 přepsat do maticové formy

$$q(t+1) = A_0 q(t) + B_0 y(t) + I_0(t) \ , \tag{4.7}$$

kde A je jednotková matice, B je diagonální matice s prvky

$$b_{jj} = T \left(\sum_{k \in I_j} \alpha_{k,j} S_k g_k^N - S_j g_j^N \right) \ , \tag{4.8}$$

a I je vektor s prvky $I_{0_j} = T i_{j,0}(t)$.

4.1.2 Minimalizace kritéria

Podobně jako v předchozí kapitole budeme minimalizovat kvaratické kritérium J. Minimalizaci budeme provádět v proměnných $u(t)$ pro časový horizont h, tedy pro vektory $\Delta C(t_0), \Delta C(t_0+1), ..., u(t_0+h)$. Pro zpřehlednění zápisu položíme bez újmy na obecnosti $t_0 = 0$. Matici $I_0(t)$, která vyjadřuje příjezd vozidel z okolí do sledované sítě, budeme v rámci minimalizačního horizontu považovat za konstantu značenou I_0 a vztah 4.7 se dá tedy přepsat do tvaru

$$\begin{pmatrix} q(t+1) \\ 1 \end{pmatrix} = \begin{pmatrix} A_0 & I_0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} q(t) \\ 1 \end{pmatrix} + \begin{pmatrix} B_0 \\ 0 \end{pmatrix} u(t) \ . \tag{4.9}$$

Po provedení substituce

$$x(t) = \begin{pmatrix} q(t) \\ 1 \end{pmatrix} \ , \quad A = \begin{pmatrix} A_0 & I_0 \\ 0 & 1 \end{pmatrix} \ , \quad B = \begin{pmatrix} B_0 \\ 0 \end{pmatrix} \ , \tag{4.10}$$

se rovnice 4.9 zjednoduší na

$$x(t+1) = Ax(t) + Bu(t) \ , \tag{4.11}$$

kde chybí konstantní člen. Poté již můžeme použít metodu popsanou v kapitole 2.6.1

4.1.2.1 Implementace minimalizace

sec:minim

Minimaizace kvadratického kritéria je v simulaci prováděna metoduo `mat L( const int horizont )` ve třídě `QuadraticMinimalizator` , které se jako parametry konstruktoru předávají matice `A`, `B`, `L` a `Q`. Pro numerické výpočty je použite knihovna IT++, která umožňuje jednoduché operace s maticemi a vektory syntakticky podobně jako v jazyku MATLAB. Jsou zde také implementovány jednoduché algoritmy, jako například pro naše účely potřebný QR rozklad. Níže je vložen zdrojový kód metody pro minimalizaci kritéria a získání hodnot řídících parametrů. Proměnné jsou v kódu značeny stejně jako v textu výše. Ty, které jsou v kódu použity navíc, jsou popsány v komentářích.

```
mat L( const int horizont ) {
        int xdim = A.cols(); // Dimenze vektoru x
        int udim = B.cols(); // Dimenze vektoru u
        mat sqQ = sqrt(Q);   // Odmocninove matice
        mat sqR = sqrt(R);
        mat M0, M;
        mat qrQ;
        mat L, Ls, Lx, Lu;
        // sestaveni matice M0
        M0 = concat_vertical(
                  concat_horizontal( sqQ * B,      sqQ * A ),
                  concat_horizontal( sqR,          zeros( udim, xdim ) )
             );
        M = M0;
        // hlavni cyklus
        for (int h = horizont; h >= 0; h--) {
             qr(M, L);  // QR rozklad matice M
             // rozklad matice L na do slozek
             Lu = L(0,     udim-1,        0,     udim-1);
             Lx = L(udim, udim+xdim-1,   udim, udim+xdim-1);
             Ls = L(0,     udim-1,        udim, udim+xdim-1);
             // kompozice matice M
             M = concat_vertical(
                       M0,
                       concat_horizontal(Lx * B, Lx * A )
                  );
        }
        // navratova hodnota - vysledna matice L : u(t+1) = Lx(t)
        return - inv(Lu) * Ls;
```

```
}
```

4.2 Simulace

Pro simulaci byl použit mikorsimulátor dopravy AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks). Podstatou mikroskopické simulace (mikrosimulace) je modelování jízdy jednotlivých vozidel po dané komunikační síti, přičemž se zohledňují všechny parametry infrastruktury i dopravních prostředků. V této kapitole je popsána základní charakteristika mikrosimulátoru AIMSUN. Podrobnější informace lze nalézt například v [1].

AIMSUN potřebuje pro svůj běh simulační scénář skládající se z popisu dopravní sítě, plánů řízení dopravy, požadovaných dopravních data, a plánů hromadné dopravy a množinu simulačních parametrů definujících experiment. Popis dopravní sítě obsahuje geometrii sítě, popis křižovatek a rozmístění detektorů, Dopravní data se dají zadat dvěma způsoby: Pomocí matice opbsahující informace kolik jízd se uskuteční z uzlu i do uzlu j. Každému vozidlu je tedy přiřazena trasa. V fruhém případě se pro určitá místa zadají hustoty provozu a vozidla se rozmístí stochasticky podle požadovaných počtů a poměrů odbočení do dopravní sítě.

4.2.0.2 VGS API

VGS API je rozhraní napsané v jazyce C++, které rozšiřuje funkčnost mikrosimulátoru AIMSUN. Zjednodušuje simulaci reálné situace, kdy podle tabulek naměřených ručně nebo zaznamenaných údajů z dopravních detektorů generuje vozidla v průběhu simulace.

Druhým důležitým úkolem VGS API je shromažďovat data potřebná pro běh experimentu a jeho. 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. Zprostředkovává tak důležitá data o počtu zastavení vozidla, o jeho zpoždění, průměrné rychlosti, době jízdy atd. Tato data jsou dostupná v průběhu simulace i po skončení k následujícímu zpracování.

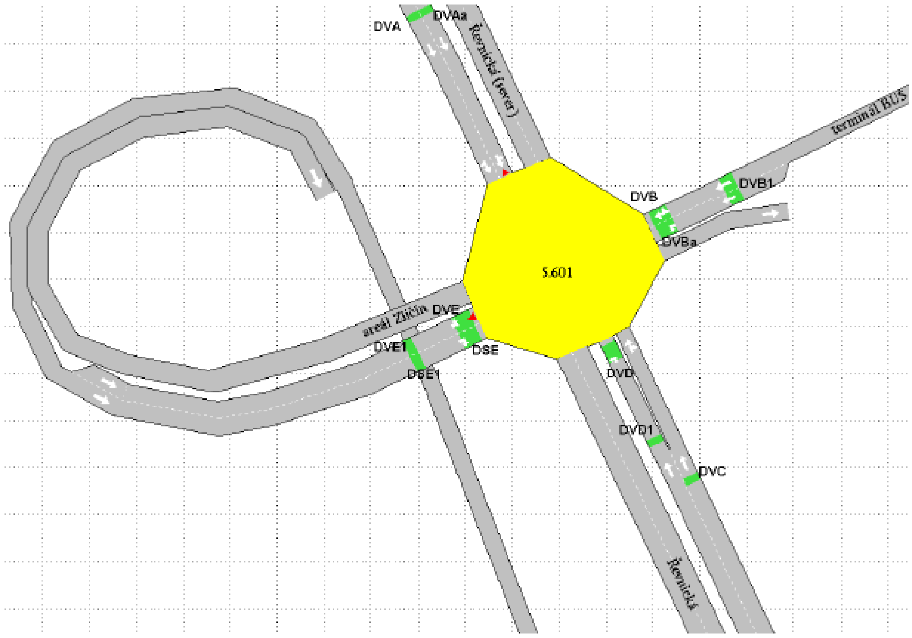
Důležitý údaj, které nám VGS API poskytuje, je délka fronty pro daný jízdní pruh. Fronta se podle dat z detektorů odhaduje velice obtížně, neboť ty vykazují zančnou chybovost, která u sledování průjezdů neni až tak zásadní, ale při použití na výpočet front by docházelo ke kumulaci této chyby a výsledky by byly prakticky nepoužitelné. Hlavní chyby detektorů spočívají v nerozpoznání mezery mezi vozidly a zpočtení dvou vozidel jako jedno, nebo zaznamenání jednoho vozidla dvěma detektory ve dvou jízdních pruzích zároveň.

4.2.1 Řadiče

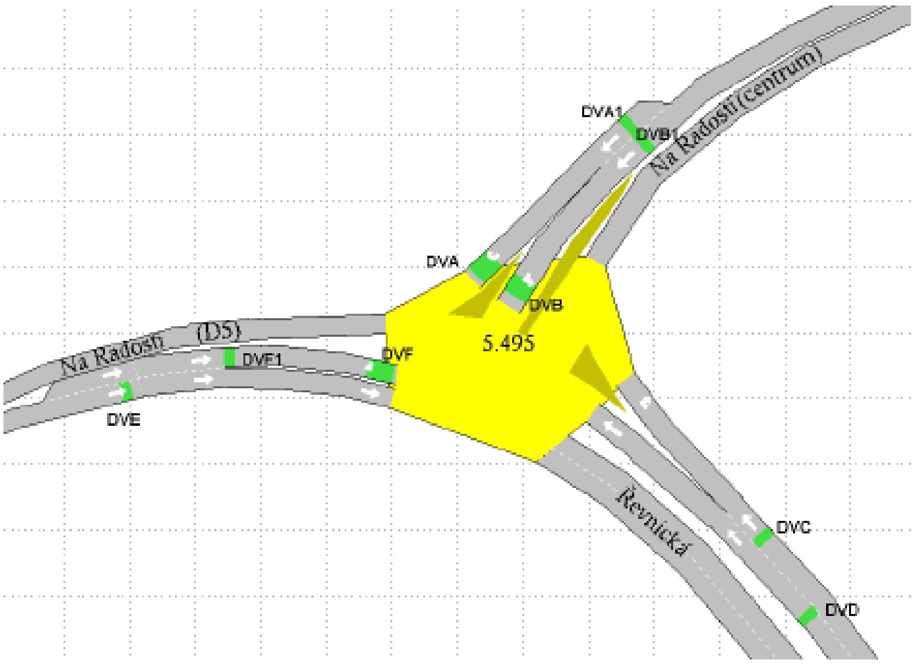
Kvůli věrnějšímu napodobení skutečnosti, jsou použity k ovládání signálních skupin použity emulátory řadičů ELS3 firmy ELTODO. Ty to řadiče, strejně jako v reálné situaci kvůli bezpečnosti, mají pevně dané fáze a v nich definované průjezdnosti daných pruhů. Ovladatelné jsou pouze vnějí parametry, jako je délka cyklu, poměry časů fází a offset. V našenm případně budeme nastavovat pouze délku cyklu, po jakou se vystřídají všechny fáze. Offset a ani poměr fází, tedy i poměr doby zelených, se nemění.

4.2.2 Oblast simulace

Pro simulaci bylo použito schéma dvou křižovatek na ulici Řevnické. Následující schémata znázorňují křižovatky s označením 495 - Na Radosti a 601 - terminál BUS, jejich pruhy (VA, VB, VC, VD, VE, VF a VA, VAa, VB, VBa, VC, VD, VE, Se) a detektory, znázorněné zelenými obdélníky.



Obrázek 4.1: Křižovatka 601



Obrázek 4.2: Křižovatka 495

4.3 Popis algoritmu

Základ hlavní smyčky programu je postaven na knihovně BDM (Bayesian Decision Making), vyvýjena na ústavu teorie informace a automatizace. Knihovna obsahuje třídu `Participant`, od které je odvozena základní třída agenta `BaseTrafficAgent`. Ta má předdefinované metody, které jsou volány v každém cyklu simulace. Jsou to metody `Adapt`, sloužící k získání dat, `Broadcast`, ve které se posílají data sousedním agentům, `Receive`, kde se zprávy příjímají a zpracovávají a `Act`, metoda určená pro nastavení řídících parametrů, v našem případě délky cyklu.

Finální agent určený pro LQ řízení je odvozen od této ťřídy. V každém cyklu posílá údaje o délce front, zatížení detektorů a odbočovacích poměrec. Z těchto dat si poté podle rovnic 4.5 a 4.7 sestaví matici B. Matice A je jednotková a penalizační matice kvadratického kritéria Q a R jsou konstantně nastaveny podle významnosti dopravních pruhů. Tím jsou sestaveny vstupní parametry pro minimalizátor, který vypočte řídící matici L potřebnou k získání optimální hodnoty délky cyklu.

4.4 Možné vylepšení do budoucna

4.4.0.1 Model toku

V článku [12] se úbytek vozidel modeluje podle rovnice 3.4, kde je uvažován tok vozidel křižovatkou za jednotku času jako konstanta S, což je saturovaný tok. Tento vztah platí pouze když je příslušný pruh naplněn vozidly do té míry, že křižovatkou projede maximální možný počet vozidel. To však neplatí, pokud není pruh dostatečně vytížen, například pokud se zmenší délka fronty v nějaký časový okamžik na nulu. Pokud je vytížení menší je tok lineárně závislý na součtu fronty $q(t)/T$ a počtu vstupujících vozidel $i(t)$. Pro teoretickou hodnotu toku S tedy dostáváme vztah

$$S_{teor}(q(t)+i(t)) = \begin{cases} \frac{q(t)}{T} + i(t) & \frac{q(t)}{T} + i(t) <= S_{max} \\ S_{max} & \frac{q(t)}{T} + i(t) > S_{max} \end{cases} \tag{4.12}$$

. tento vztah vyjadřuje to, že pokud je fronta plus přírůstrek vozidel ve vzorkovací periodě $(q(t), i(t)T)$ menší než maximální počet vozidel, který je křižovatka za periodu T schopna propustit (TS_{max}), projedou všechna vozidla. Abychom se vyhnuli podmínce podle S_{max}, aproximuje se tento vztah funkcí

$$S_{exp}(q(t), i(t)) = S_{max}\left(1 - e^{-\frac{1}{S_{max}}\left(\frac{q(t)}{T} + i(t)\right)}\right) \tag{4.13}$$

, která splňuje podmínku

$$\frac{dS_{exp}}{d(q/T + i)}(0, 0) = 1 \tag{4.14}$$

, tedy při malé frontě a hustotě provozu je přírůstek toku roven $q/T + i$, a podmínku

$$\lim_{q/T+i\to\infty} S_{exp} = S_{max} \tag{4.15}$$

, tedy při velké frontě a hustotě provozu se tok blíží konstantě S_{max}. Pro malý časový interval můžeme tuto funkci dále zjednodušit na lineární vztah

$$S(q(t)) = \sigma(q(t)/T + i_0) \tag{4.16}$$

, kde je $i_0 = i(0)$ a

$$\sigma = \frac{dS_{exp}}{d(q/T + i)}(q(0), i(0)) \tag{4.17}$$

4.4.1 Odhdad odbočovacích poměrů

V ninější verzi se pracuje s konstantními odbočovacími poměry, které byly naměřeny v reálné siutaci a zadávají se pomocí konfiguračního souboru. Pro standartní simulace je tento způsob dostačující, do budoucna by však mohlo přinést zlepšení tyto koeficienty odhadovat v průběhu simulace.

Vhodnou metodou k odhadu odbočovacích poměrů by mohl být například Kalmanův filtr, nebo jiná metoda používající bayesovské učení popsané v kapitole 2.5 , kde by se nyní používaná konstantní hodnota zavedla jako apriorní pravděpodobnost.

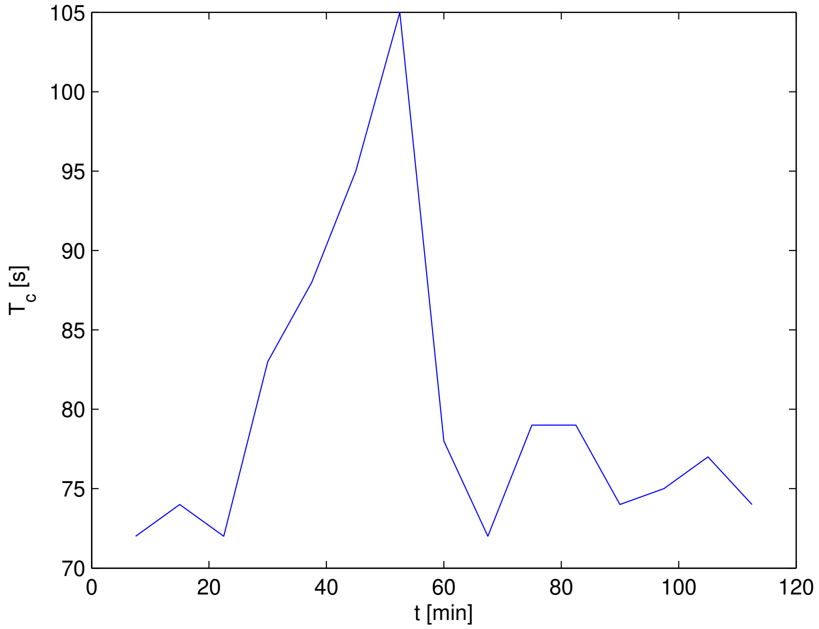
Kapitola 5

Výsledky

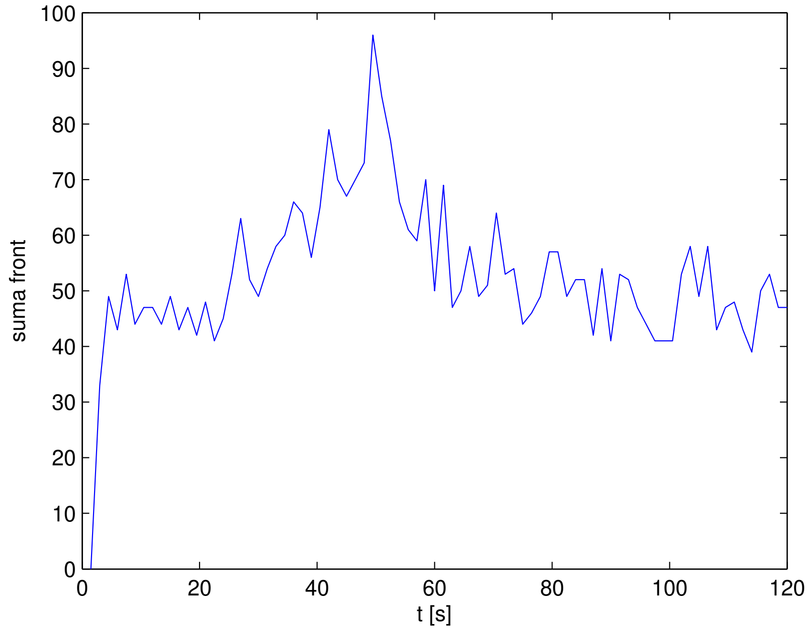
Použitá metoda byla testována na mikrosimulátoru dopravy AIMSUN v oblasti Praha - Zličín na dvou scénářích s konstantní hustotou provozu a jednom scénáři reálným. Referenční výsledky byly získány testováním těchto scénářů na konstantní délce cyklu o hodnotě 80s, která je v modelové oblasti skutečně používána.

5.1 Scénář 1

První testovaný konstantní scénář má menší intenzitu provozu a při měření s různými konstantními délkami cyklu se ukazuje, že referenční hodnota 80s je více méně optimální. Jak ukazuje následující graf, algoritmus osciluje okolo těchto hodnot až na moment kolem padesáté minuty, kdy reaguje na prudké zvýšení front, jak ukazuje další graf.

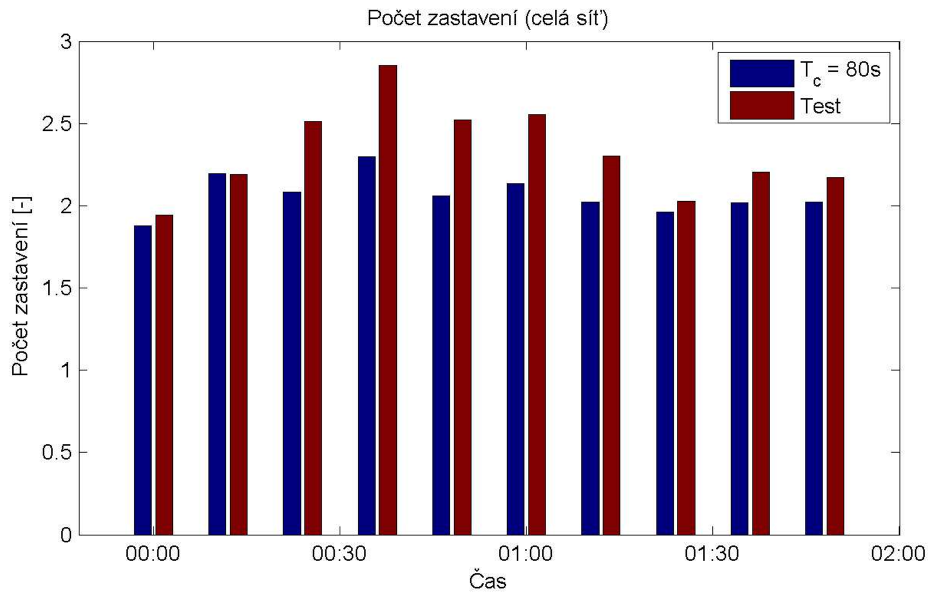


Obrázek 5.1: Průběh délky cyklu

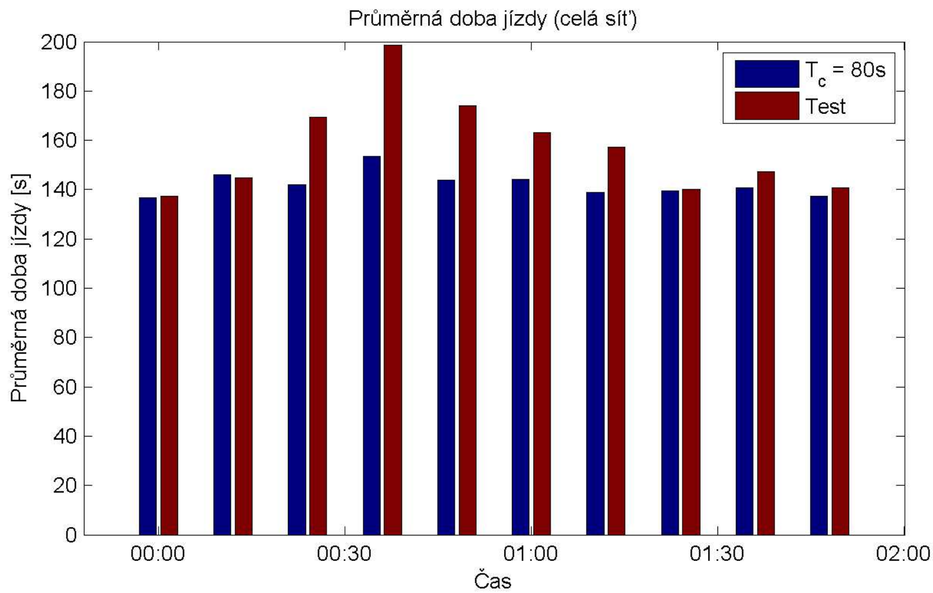


Obrázek 5.2: Průběh součtu délek front

Tento nárůst front kolem padesáté minuty se projevil negativně na vývoji dopravy, jak ukazují následující dva grafy porovnávající počty zastavení a průměrnou dobu jízdy s referenční délkou cyklu 80s.



Obrázek 5.3: Počet zastavení



Obrázek 5.4: Průměrná doba jízdy

5.2 Scénář 2

Kapitola 6

Závěr

Literatura

[1] *AIMSUN Getram v4.2 getting started - User's manual*. 2003.

[2] Anderson, M. J., B.D.O.: Optimal Control - Linear Quadratic Methods. *Prentice Hall, Englewood Cliffs, NJ*, 1990.

[3] Bellman, R.: Dynamic programming. *Princeton University Press*, 1957.

[4] Ferreira, E.; Subrahmanian, E.; Manstetten, D.: Intelligent agents in decentralized traffic control. *Intelligent Transportation Systems*, 2001.

[5] Gmytrasiewicz, P. J.; Durfee, E. H.: A rigorous, operational formalization of recursive modeling. *First International Conference on Multiagent Systems*, 1995.

[6] Nagy, I.; Nedoma, P.; Ettler, P.; aj.: O bayesovském učení. *Automa*, 2002.

[7] Ou, H.; Zhang, W.; Xu, X.: Urban traffic multi-agent system based on RMM and Bayesian learning. *American Control Conference*, 2002.

[8] Pecherková, P.; Duník, J.; Flídr, M.: Modelling and Simultaneous Estimation of State and Parameters of Traffic System. *Robotics, Automation and Control*, 2008.

[9] Schier, J.: Parallel algorithms for Robust Adaptive Identification and Square-Root LQG Controll Synthesis. 1994.

[10] Sutton, R. S.: Learning to predict by the methods of temporal didffrences. *Machine Learning*, 1988.

[11] Thorpe, T.: Vehicle traffic light controlusing sarsa. *Master's thesis, Department of Computer Science, Colorado State University*, 1997.

[12] Vaya Dinopoulou, M. P., Christina Diakaki: Applications of the urban traffic control strategy TUC. *European Journal of Operational Research*, 2005.

[13] Watkins, C. J. C. H.: Leraning from Delayed Rewards. *PhD thesis, King's College, Cambridge, England*, 1989.

[14] Watkins, C. J. C. H.; Dayan, P.: Q-leraning. *Machine Learning*, 1992.

[15] Wiering, M.; Van Veenen, J.; Vreeken, J.; aj.: Intelligent traffic light control. *European Research Consortium for Informatics and Mathematics*, 2003.

[16] Wooldridge, M.: *Multi Agent Systems*. MIT Press, Březen 2005.

Příloha A

Příloha 1