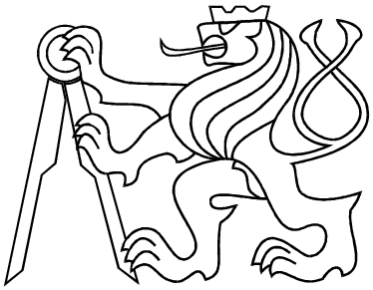
České vysoké učení technické v Praze
Fakulta elektrotechnická



BAKALÁŘSKÁ PRÁCE

AIMSUN - MATLAB Toolbox

Praha, 2007 **Autor: Lukáš Dibelka**

České vysoké učení technické v Praze, fakulta elektrotechnická

Katedra řídicí techniky Školní rok: 2006/2007

Zadání bakalářské práce

Student: Lukáš Dibelka

Obor: Kybernetika a měření

Název tématu: Aimsun-Matlab Toolbox

Zásady pro vypracování:

1. Seznamte se s prostředím dopravního simulátoru GETRAM.
2. Prostudujte si stávající dopravní Aimsun-Matlab Interface, vyvíjený na ÚTIA AV ČR, který umožňuje komunikaci Matlabu s dopravním simulátorem Aimsun.
3. Tento toolbox rozšiřte dle průběžně zadávaných požadavků. Zejména se jedná o snadnější, systematičtější a v maximální míře automatizované přidávání dalších dopravních oblastí pro simulace a dále o uživatelsky přívětivější ovládání celého toolboxu.
4. Vytvořte dokumentaci a uživatelskou příručku.

Seznam odborné literatury:

- Transport Simulation Systems: *GETRAM Extensions Version 4.2 - User manual [online]*, http://marabu.utia.cas.cz:1800/svn/doprava/Ruzne/AIMSUN_manualy/GetramExtensionsv4.2.pdf.
- Petr Gebouský: *AIMSUN-MATLAB Interface - User's Guide [online]*, http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/manualASYN/manualASYN.pdf.

Vedoucí bakalářské práce: Ing. Pavel Dohnal

Datum zadání bakalářské práce: zimní semestr 2006/07

Termín odevzdání bakalářské práce: 15. 8. 2007

Prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická -4- L.S.

Prof. Ing. Zbyněk Škvor, CSc.
děkan

V Praze, dne 28. 2. 2007

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb. , o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne _____ _____

podpis

Poděkování

Děkuji především Pavlu Dohnalovi za odoborné vedení a pomoc, kterou mi poskytl při vypracování mé bakalářské práce.
Dále bych chtěl poděkovat své rodině, přítelkyni a kamarádům za podporu při studiu.

Abstrakt

Cílem této bakalářské práce je naučit se provádět dopravní experimenty v AIMSUN - MATLAB toolboxu, podílet se na vývoji toolboxu, rozšiřovat jej podle požadavků kolegů na ÚTIA AV ČR a pro aktuální verzi toolboxu vytvořit uživatelskou příručku v anglickém jazyce. AIMSUN - MATLAB toolbox představuje rozhraní mezi mikrosimulačním nástrojem AIMSUN a prostředím pro technické výpočty MATLAB. Pomocí tohoto toolboxu je možné simulovat reálné a hypotetické dopravní situace a testovat na nich algoritmy řízení.

Abstract

The aim of this bachelor is to learn how to maintain traffic experiments in AIMSUN - MATLAB toolbox, participate in development of toolbox according to requirements from developers in ÚTIA AV ČR and create user's guide in English language for actual version of toolbox. AIMSUN - MATLAB toolbox represents interface between microsimulator tool AIMSUN and enviroment for technical computing MATLAB. This toolbox allows to simulate real and hypotetical traffic situations and test control algorithms on them.

Obsah

Seznam obrázků **vii**

1 Úvod **1**

2 Úvod do dopravní problematiky **3**

2.1 Význam dopravních simulací 3

2.1.1 Model dopravní oblasti 3

2.1.2 Základní dělení simulačních nástrojů 3

2.2 Měření dopravních dat a řízení křižovatek 4

2.3 Stručný popis softwaru GETRAM 5

2.3.1 Editor dopravní sítě TEDI 5

2.3.2 Dopravní simulátor AIMSUN 7

2.4 Rozrhraní GETRAM Extensions 9

2.5 AIMSUN - MATLAB toolbox 11

2.5.1 Popis toolboxu 11

2.5.2 Provedení experimentu 14

3 Vlastní práce **17**

3.1 Vykreslování historie délek fází 17

3.2 Zjištění typu řízení aktuální křižovatky 20

3.3 Změna aktuálního řídícího plánu a traffic result 23

3.4 Instalační funkce uiinstall 24

3.4.1 Změna vzhledu uživatelského rozhraní 26

3.4.2 Popis funkce installSetup 26

3.5 Vytvoření konfig. souboru experimentu pomocí GUI 28

3.6 Podpůrné funkce pro práci s textem 33

4 Závěr **35**

Literatura **I**

A Obsah přiloženého CD **III**

B MATLAB - AIMSUN Toolbox 2.1, user's guide **V**

Seznam obrázků

2.1 Uživatelské rozhraní programu TEDI 6
2.2 Uživatelské rozhraní programu AIMSUN 8
2.3 Proces výměny dat mezi AIMSUNem a externí aplikací 9
2.4 Schéma komunikace mezi AIMSUN a GETRAM Extensions 10

3.1 Výstup funkce `plotPhases` pro matici *phases* 18
3.2 Uživatelské rozhraní funkce `uiinstall` 25
3.3 Uživatelské rozhraní funkce `uiicreateArea` 33

Kapitola 1

Úvod

S rozvojem automobilové dopravy a výpočetní techniky vzrostl význam dopravních simulací. Realizace nové infrastruktury a jakákoliv změna ve stávající infrastruktuře je finanncně velice nákladná, proto je vhodné, jako součást plánování výstavby, provést simulaci dopravní oblasti. Díky tomu je možné zjistit veškeré dopady zásahu do existující infrastruktury ještě před realizací jakékoliv změny a tím ušetřit značné finanční prostředky.

Další oblastí nasazení dopravních simulací je testování algoritmů řízení světelných křižovatek. Tato úloha je velmi důležitá, jelikož hustota dopravy v městech dnes narostla do rozměrů saturovaných toků. V městské dopravní síti se tedy vytváří kolony a prodlužuje se doba průjezdu vozidel dopravní sítí. Online řízení dopravy předpokládá znalost aktuálních dopravních dat na konkretních místech v dopravní síti. Na základě těchto dat je možné pomocí zpětnovazebních algoritmů vypočítat optimální signální plány pro jednotlivé křižovatky.

Dopravní simulační nástroje se dělí na dvě skupiny, makrosimulátory a mikrosimulátory. Makrosimulátory vycházejí z matematických modelů popisujících chování všech vozidel. Mikrosimulátory modelují pohyb každého vozidla, které se pohybuje dopravní sítí. Ve své práci se budu zabývat pouze mikrosimulačními nástroji, konkrétně softwarovým balíkem GETRAM verze 4.2 od španělské firmy TSS.

Software GETRAM se skládá z grafického editoru TEDI pro vytvoření dopravního modelu a simulátoru AIMSUN pro provedení simulace. Program AIMSUN je pouze simulátor, který poskytuje aktuální dopravní a statistická data. Pro aplikaci algoritmu řízení je nutné použít program, který umožňuje realizovat výpočet signálních plánů křižovatek na základě použitého algoritmu. Takovýmto nástrojem může být například MATLAB. Aby bylo možné realizovat komunikaci mezi těmito programy, poskytuje prostředí

GETRAM rozhraní GETRAM Extensions, které zajišťuje výměnu dat mezi AIMSUNem a jinou aplikací. GETRAM Extensions je obecný nástroj, pro spolupráci s MATLABem je na ÚTIA AV ČR vyvíjen AIMSUN - MATLAB toolbox.

Tento toolbox implementuje rozhraní mezi AIMSUNem a MATLABem a vytváří tak silný nástroj pro provádění dopravních experimentů s možností testování nových algoritmů řízení. Umožňuje zpracovávat data ze simulace a upravené je posílat simulaci zpět (řízení křižovatek).

Tato práce je rozdělena na dvě hlavní kapitoly. V první kapitole je podrobněji vysvětlen význam dopravních simulací, detailnější popis rozdílů mezi mikro a makrosimulačním nástrojem. Dále je zde uveden popis nástrojů TEDI, AIMSUN, rozhraní GETRAM Extensions a AIMSUN - MATLAB toolbox. Poslední částí první kapitoly je zkrácený popis postupu při vytváření experimentu.

V druhé kapitole uvádím detailní popis práce, kterou jsem se podílel na vývoji toolboxu. Cílem mé práce bylo rozšířit toolbox o přívětivější uživatelské rozhraní, vytvořit systém pro jednoduché přidávání nových dopravních oblastí a dále implementovat průběžné požadavky od ostatních vývojářů a uživatelů toolboxu. Součástí mého zadání také bylo vytvoření uživatelské příručky k toolboxu v anglickém jazyce. Přikládám ji tedy jako přílohu.

Kapitola 2

Úvod do dopravní problematiky

2.1 Význam dopravních simulací

Dopravní simulace slouží ke studiu, analýze chování silniční dopravy a k předvídání budoucích situací na základě aktuálního stavu provozu nebo statistických dat. Analýza dopravní situace se provádí hlavně při plánování výstavby dopravní oblasti s ohledem na budoucí provoz. Umožňuje vyhodnocovat hypotetické situace, které se nedají sledovat v reálném provozu. Výsledky analýzy jsou přímo závislé na použitém modelu. Čím přesnější model je použit, tím kvalitněji je možné budoucí situaci analyzovat. Druhou oblastí použití dopravních simulací je testování algoritmů řízení dopravní oblasti.

2.1.1 Model dopravní oblasti

Veškeré dopravní simulace jsou založeny na matematických modelech, které představují matematický popis řešeného problému. Jejich rozdílnost spočívá v míře přiblížení se k realitě, kde dominují zejména mikrosimulační nástroje. Jelikož pro analýzu rozsáhlých dopravních sítí by tvorba modelu pomocí mikrosimulačního nástroje byla neúměrně náročná a zbytečně detailní, je vhodnější využít větší míry zobecnění, tj. makrosimulačního nástroje.

2.1.2 Základní dělení simulačních nástrojů

Simulační nástroje dělíme na makroskopické, mikroskopické a mesoskopické. Mesoskopické simulátory jsou určitým kompromisem mezi mikroskopickým a makroskopickým nástrojem. Jejich význam je ve srovnání se zbylými kategoriemi zanedbatelný.

Makroskopické simulace makroskopické simulační nástroje modelují celé dopravní proudy na základě makroskopických veličin jako např. intenzity, hustoty, rychlosti dopravního proudu a vztahů mezi nimi. Mezi základní simulační možnosti patří přidělení dopravní zátěže na simulační síť, identifikace "dopravních hrdel", tvorba šokových vln, apod. Jejich podstata je velice blízká standartním dopravně kapacitním výpočtům, avšak výhodou je automatizace tohoto procesu a rovněž možnost analýzy velmi rozsáhlé sítě.

Mikroskopické simulace základem mikroskopické simulace (mikrosimulace) je modelování jízdy jednotlivých vozidel po dopravní síti, přičemž se zohledňují všechny vlastnosti infrastruktury i dopravních prostředků. Parametry mikrosimulace jsou například rozměry, rychlost, zrychlení, hmostnost vozidla atd.

Tradiční výpočtové metody dosazují agregované údaje jako celkové intenzity, podíly typů vozidel v dopravním proudu a další údaje do matematických vzorců. Jejich nevýhodou je vysoká míra zobecnění a tudíž špatné nasazení na konkrétní situaci. Naproti tomu mikrosimulace je vysoce univerzální, automatizovaná a relativně přímočará metoda. Oproti klasickým výpočtům umožňuje podstatně lepší přiblížení se realitě, ale také podstatně jednodušší a ověřitelnější zadávání vstupních údajů.

Popis softwarového balíku GETRAM, který se zabývá mikroskopickými simulacemi, je uveden v kapitole 2.3.

2.2 Měření dopravních dat a řízení křižovatek

Dopravní data se získávají z detektorů umístěných většinou pod povrchem jízdních pruhů. Detektory jsou binární, tzn. jejich výstup je log. 1 pokud je na detektoru přítomno vozidlo, nebo log. 0 pokud vozidlo přítomno není. Podle délky výstupního impulsu je možné určit typ a rychlost vozidla. Počet impulsů na jednu periodu měření představuje **intenzitu dopravního proudu** [vozidlo/čas]. Celková délka impulsů na periodu představuje **obsazenost** [%].

Při znalosti aktuálních dopravních dat z vybraných míst dopravní oblasti (především ramena křižovatek), je možné na křižovatky aplikovat zpětnovazební řízení. Řízení křižovatky znamená generování délky zelených signálů na světelných signalizačních zařízeních SSZ. Celá posloupnost signálů (červená, žlutá a zelená) pro jednu signální skupinu se nazývá

fáze [s]. Celková délka fáze je konstantí a odpovídá **cyklu** [s] řízení, mění se tedy pouze jednotlivé signály ve fázích. Pokud je řízení statické, je nastavení fází a signálů napevno uloženo v signálních plánech.

2.3 Stručný popis softwaru GETRAM

Softwarový balík GETRAM (**G**eneric **E**nviroment for **T**raffic **A**nalysis and **M**odeling) pro platformu Win32 má tři základní části :

TEDI editor pro vytvoření dopravního modelu

Aimsun mikroskopický simulátor dopravních situací

Aimsun 3D vizualizační modul pro 3D zobrazení simulace

2.3.1 Editor dopravní sítě TEDI

Grafický editor TEDI slouží k nakreslení dopravní sítě, tj. pro vytvoření modelu dopravní oblasti, který se může skládat hlavně z :

Sections jízdní pruhy vytvářející spojení mezi uzly dopravní sítě

Junctions, joins uzly dopravní sítě (křižovatky)

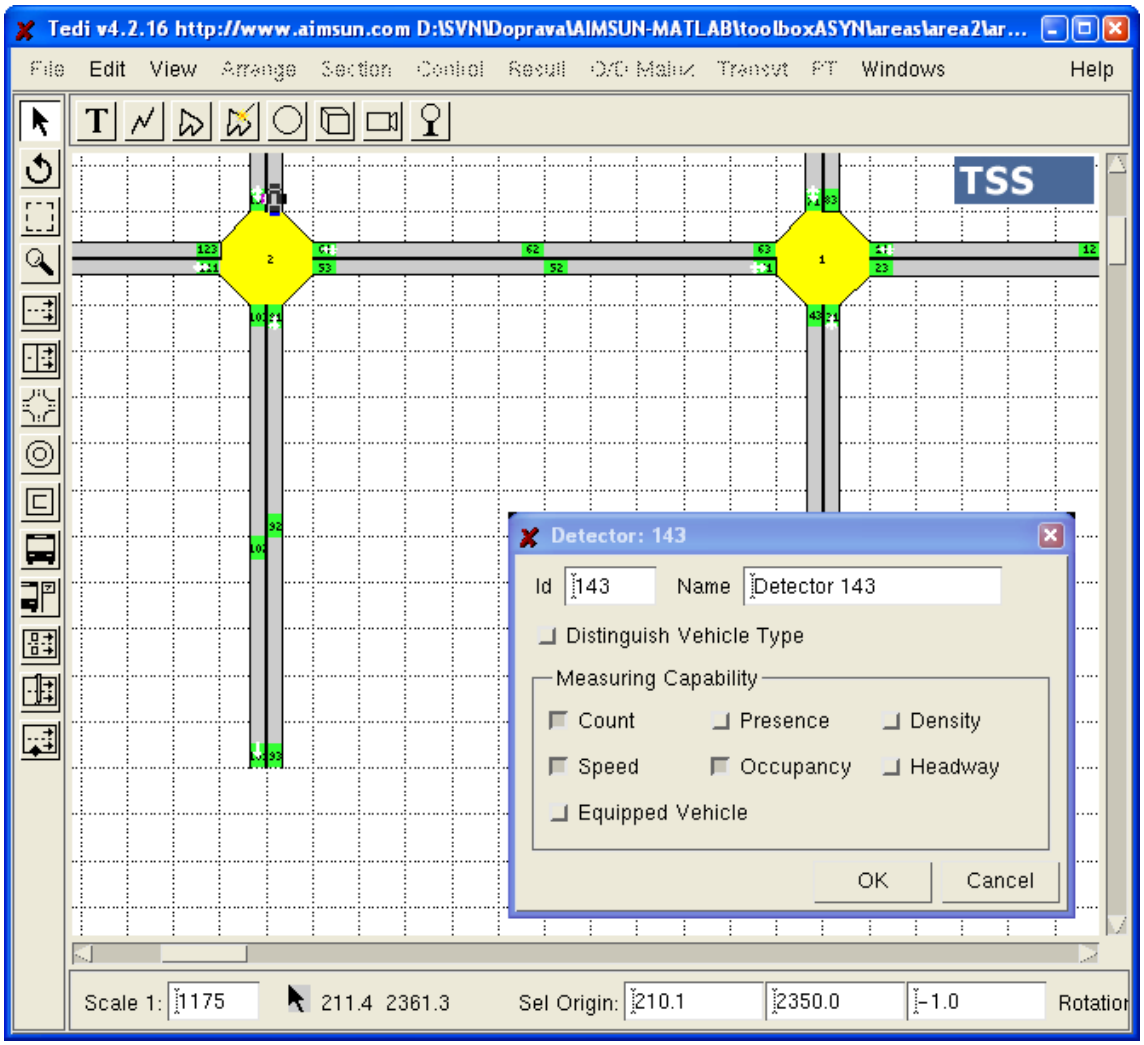
Centroids dopravní zdroje a ústí

Detectors místa sběru dat

VMS zobrazení dopravních informací

TEDI umožňuje přidat do modelu také autobusové zastávky, přechody pro chodce atd. Každý prvek modelu dopravní oblasti je charakterizován svým identifikačním číslem (dále jen ID).

Součástí modelu je také definice typů vozidel, vjezdů vozidel do jednotlivých sekcí, řídících plánů křižovatek *control plan* a objektu *result container* obsahujícího výsledky simulace. Uživatelské rozhraní programu TEDI je na obr. 2.1



Obrázek 2.1: Uživatelské rozhraní programu TEDI

2.3.2 Dopravní simulátor AIMSUN

Advanced **I**nteractive **M**icroscopic **S**imulator for **U**rban and **N**on-Urban **N**etworks slouží pro mikroskopické simulace dopravních oblastí. Dopravní oblast se v AIMSUNu nazývá scénář. Skládá se z dopravní sítě, řídícího plánu *control plan* a objektu výsledků simulace *traffic result*. Součástí scénáře je také nastavení modulu pro spojení simulátoru AIMSUN s externí aplikací viz 2.4.

Vstupními daty simulace jsou :

Initial/end time určuje, jak dlouho bude simulace probíhat

Warm up period zaplnění sítě před spuštěním simulace

Simulation step doba odezvy řidiče

Queuing up speed rychlost, kterou se pohybují vozidla v koloně

Queuing leaving speed rychlost vozidla, které opouští kolonu

Car following model maximální počet vozidel, maximální vzdálenost atd.

Lane changing model pravděpodobnost předjíždění

Accidents definice mimořádných situací

Výstupními daty jsou :

Mean flow střední tok a hustota vozidel

Mean speed střední rychlost vozidel

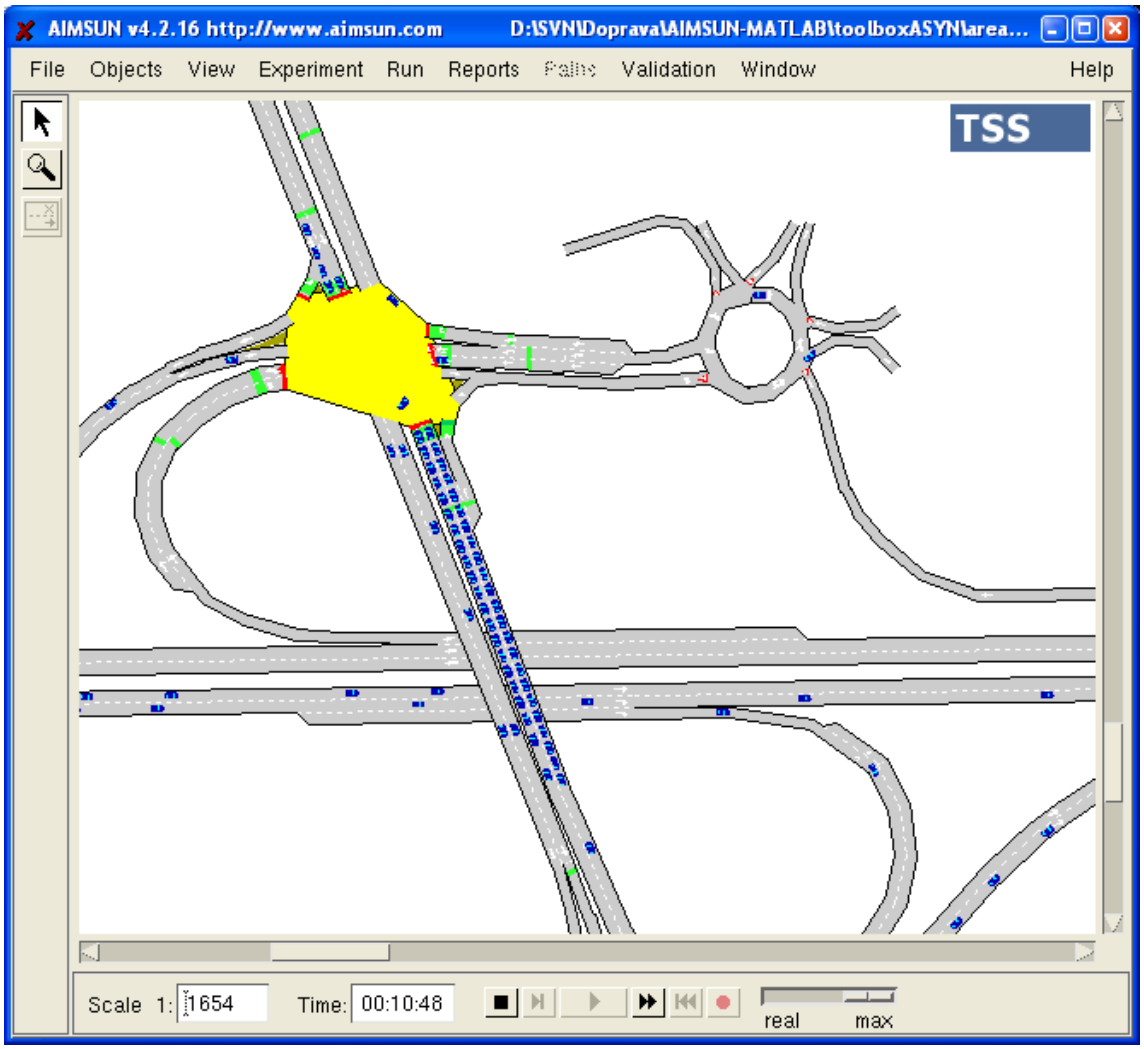
Travel time, delay time doba průjezdu dopravní sítí a zpoždění vozidel

Stop time, number of stops okamžik zastavení vozidla a jejich počet

Queue length (mean and maximum) délky dopravních kolon

Fuel consumed, pollution emitted množství spotřebovaného paliva a vyprodukovaného znečištění.

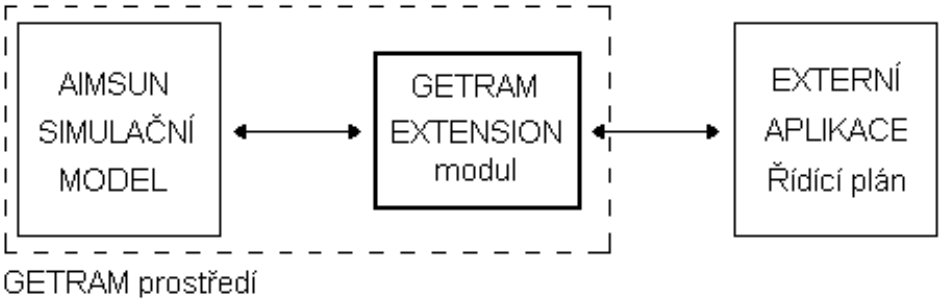
Výstupní data je možné zobrazit v podobě grafů, uložit v ASCII formátu pro další zpracování (např. Excel), nebo poskytnou externí aplikaci viz 2.4. Simulace probíhá v definovaném čase. Rychlost simulace může být reálná (s vykreslováním projíždějících vozidel) nebo zrychlená. Ukázka uživatelského rozhraní za běhu simulace je na obr. 2.2.



Obrázek 2.2: Uživatelské rozhraní programu AIMSUN

2.4 Rozrhraní GETRAM Extensions

Program AIMSUN dokáže přes modul GETRAM Extensions komunikovat s externí aplikací. Podporovaná je obousměrná komunikace, tzn. AIMSUN poskytuje externí aplikaci aktuální data ze simulace a externí aplikace zasílá AIMSUNu aktuální řízení křižovatek. Externí aplikace tedy slouží pro realizaci algoritmu řízení. Proces výměny dat mezi AIMSUNem a externí aplikací je na obr. 2.3



Obrázek 2.3: Proces výměny dat mezi AIMSUNem a externí aplikací

Modul GETRAM Extensions realizuje propojení mezi AIMSUNem a externí aplikací. Komunikace s AIMSUNem je součástí modulu. Komunikace s externí aplikací musí být vytvořena uživatelem.

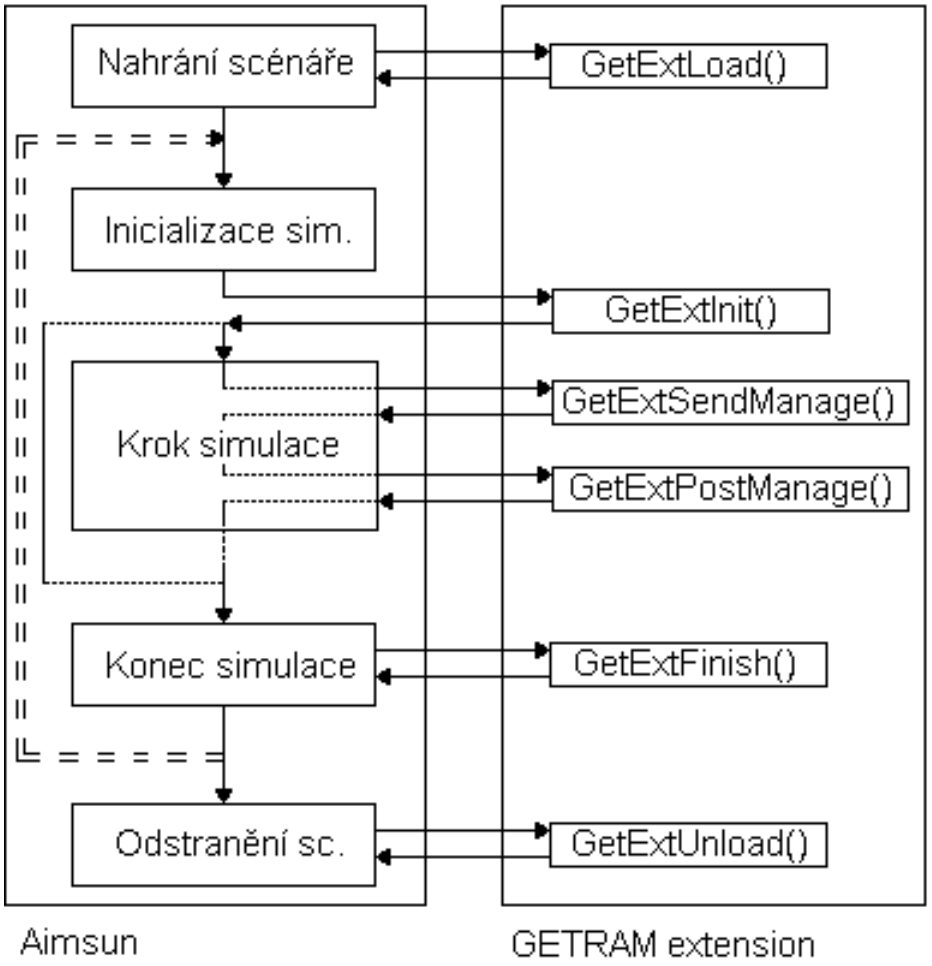
GETRAM Extensions má pro komunikaci s AIMSUN modelem definováno šet high-level funkcí, které jsou volány v těchto okamžicích :

1. GetExtLoad() při nahrání GETRAM Extensions modulu pomocí AIMSUN
2. GetExtInit() inicializace a start simulace
3. GetExtManage(float time, float timeSta, float timTrans, float acicle) na začátku každého simulačního kroku
   - time absolutní čas simulace [s]
   - timeSta perioda simulace [s]
   - timeTrans délka warm-up periody [s]
   - acicle délka každého simulačního kroku [s]
4. GetExtPostManage(float time, float timeSta, float timTrans, float acicle) na konci každého simulačního kroku

5. GetExtFinish() na konci simulace

6. GetExtUnload() při odstranění modulu pomocí AIMSUN

GETRAM Extensions modul může být vytvořen jako DLL knihovna napsaná v jazyce C++ nebo Python. Pro její vytvoření jsou poskytnuty hlavičkové soubory *GetExt_common.h*, *AKI_proxie.h*, *CIProxie.h*, knihovna *GetExt42.lib* a implementační soubor *GetExt.cxx*.



Obrázek 2.4: Schéma komunikace mezi AIMSUN a GETRAM Extensions

2.5 AIMSUN - MATLAB toolbox

2.5.1 Popis toolboxu

Úkol AIMSUN - MATLAB toolboxu je implementace rozhraní mezi programem AIMSUN a aplikací MATLAB. Spojení těchto dvou programů představuje silný nástroj pro provádění dopravních simulací (experimentů) s možností testování řídících algoritmů. Aktuální verze toolboxu je asynchronní, tzn. umožňuje zasílat a získávat data v různých periodách. Toolbox je vytvořen pro AIMSUN verze 4.2 a MATLAB verze 7.1. AIMSUN - MATLAB toolbox umožňuje :

- zpracovávat data
  - měřit intenzity, obsazenosti a rychlosti na detektorech a agregovat je.
  - měřit průměrné a maximální délky kolon a agregovat je
  - měřit periodické statistické data (celkový čas průjezdu vozidel, počet zastavení atd.)
  - měřit délky fází a mezifází použitého řízení se všemi detaily
- provádět akce
  - generovat incidenty v určitém místě a čase
  - generovat časově proměnné vjezdy vozidel do definovaných sekcí.
  - měnit délky fází signálních plánů u externě řízených křižovatek

Toolbox se skládá ze dvou částí :

- sdílená dynamická knihovna
- sada MATLAB funkcí

Jádrem toolboxu je dynamická knihovna *GetExt42R.dll*, která je za běhu simulace nalinkována jak AIMSUNem, tak MATLABem. Přes ní dochází ke komunikaci a výměně dat mezi těmito programy. Tato knihovna využívá funkce rozhraní GETRAM Extensions viz 2.4 a byla vytvořena s využitím zdrojových kódů poskytovaných v rámci GETRAM Extensions.

Data jsou reprezentována *"C"* strukturou, která je uložena ve sdílené části Windows stránkovacího souboru. Dimenze všech polí struktury jsou statické. Pokud by nevyhovovaly velikosti experimentu, je nutné provést úpravy v definicích těchto polí a poté knihovnu překompilovat. Proces synchronizace a komunikace AIMSUNu a MATLABu :

```
Cyklus života Aimsun scénáře

Start

%inicalizace
  poslání init dat
  předání aktivity MATLABu
  %čekání na AIMSUN

  aplikace obdržených akcí

WHILE NOT konec simulace

  IF (buffer plný) &
  (je třeba poslat data)
    poslání dat
    předání aktivity MATLABu

    %čekání na předání aktivity
    AIMSUNu

  END

  IF (externí generátor vozidel)
  & (nový interval)
    poslání dat
    předání aktivity MATLABu

    %čekání na předání aktivity
```

```
AimsunSimulationStep( , phaseTiming)

followSimulation = 1
firstConrolAlreadyApplied = 0
WHILE followSimulation
  simulationState = getActualState
  %čekání na předání aktivity
  MATLABu

  SWITCH simulationState,

    CASE simulace ukončena
      kontrola dat příslušících
      simulačnímu kroku
      načtení globálních statistických
      dat
      NfinishedExperiments++

    CASE inicalizace
      čtení inicalizačních dat
      poslání vjezdů vozidel
      poslání incidentů
      poslání řízení křižovatek

      předání aktivity AIMSUNu

    CASE čtení dat z detektorů
      kontrola dat příslušících
      simulačnímu kroku
```

```
    AIMSUNu

    generování vjezdů vozidel
  END

  IF (externí řízení) &
  (začátek cyklu hlavní křižovatky)
     poslání dat
     předání aktivity MATLABu

    %čekání na předání aktivity
    AIMSUNu

    aplikace řízení křižovatek
  END
 END
%Ukončovací blok
 poslání dat
 předání aktivity MATLABu
```

```
    načtení vjezdů vozidel,
    pokud je třeba, tak vrátit
    aktivitu AIMSUNu

CASE čtení periodických statistik
    nebo cyklu
    kontrola dat příslušících
    simulačnímu kroku

    předání aktivity AIMSUNu

CASE čekání na nové
    nastavení řízení křižovatek
    IF firstConrolAlreadyApplied
      kontrola dat příslušících
      simulačnímu kroku
      followSimulation = 0
    ELSE
      poslání řízení křižovatek
      firstConrolAlreadyApplied = 1
      předání aktivity AIMSUNu
    END

CASE no simulation started
    IF NfinishedExperiments <
      Nexperiments
      spuštění nové
      simulace
    ELSE
      followSimulation = 0
    END
  END
END
```

Druhou částí toolboxu je sada MATLAB funkcí, které slouží pro přípravu a provedení experimentu, sběr statistických dat, aplikaci řídícího algoritmu atd. Tyto MATLAB funkce volají knihovní funkce *GetExt42R.dll* MATLAB funkcí `calllib`. Nejdůležitější funkcí, která vykonává jeden simulační krok, je funkce :

```
[E, state, buf] = AimsunSimulationStep(E, realF)
```

Popis parametrů :

E objekt obsahující informace o aktuálním experimentu

realF matice fází pro všechny externě řízené křižovatky

state pokud *state = 1* simulace může pokračovat, pokud *state = 1* pak konec simulace

buf buffer obsahující poslední obdržené data

Objekt *E* obsahuje data prováděného experimentu (ID detektorů, sekcí, vstupních sekcí atd.). Pro jeho vytvoření slouží funkce uvedené v [4]. Ostatní MATLAB funkce jsou popsány v kapitole 2.5.2 a příloze B.

Obě části toolboxu vytváří při své činnosti log soubory (Dll log a MATLAB log). Jejich obsahem je výpis všech vnitřních proměnných, velikosti volné paměti, všech provedených příkazů atd. Log soubory slouží především pro odstraňování chyb v toolboxu. Pokud se experiment z nějakého důvodu úspěšně neprovede, je možné poslat vývojářům toolboxu tyto log soubory a ti zajistí odstranění chyb.

2.5.2 Provedení experimentu

Před provedením experimentu je nutné nakreslit dopravní síť v programu TEDI viz 2.1, a poté vytvořit scénář pomocí programu AIMSUN viz 2.2. Takto vytvořenou oblast je nutné s konfiguračním souborem *trafficArea.mat* viz 3.5 nakopírovat do příslušného adresáře v adresáři *areas* toolboxu. S takto implementovanou oblastí je možné provádět dopravní experimenty. Pro tyto účely slouží v toolboxu čtyři funkce (`initArea`, `getActualData`, `getActualStatistics` a `setActualControl`). Detailní popis těchto funkcí je uveden v uživatelské příručce viz příloha B.

Zde uvádím pouze jejich stručný popis :

initArea(AreaName, VehGenerator, Entrances, ReportPath, Description, Author) výběr oblasti a vstupních dat (tj. kolik vozidel kam jezdí a jak jsou zastoupené jednotlivé typy jako osobní auta, autobusy, atd.)

getActualData(Experiment) vrácení posledních agregovaných dat z detektorů ve formě sloupcových vektorů příslušných veličin, kde řádky tvoří data z příslušných detektorů.

getActualStatistics (Experiment) vrácení délek kolon na daném pruhu ve formě sloupcových vektorů příslušných veličin, kde řádky representují jednotlivé jízdní pruhy.

setActualControl (Experiment, PhasesLengths, EstimatedQueues, RealEntrances) nastavení délek fází na všech řízených křižovatkách a provedení jednoho kroku simulace.

Pro otestování toolboxu a dopravní oblasti slouží funkce `demoArea` :

demoArea(trafficArea) spuštění simulace s konstantním řízením (*control plan* viz 3.3). Výstupem simulace jsou vektory intenzit, rychlostí a obsazeností.

Pokud je ovšem požadavek na zasílání řízení křižovatek, je nutné upravit strukturu souboru *demoArea.m*, resp. do zpětnovazební smyčky přidat výpočet aktuálního řízení. Ukázka struktury souboru *demoArea.m* s výpočtem řízení :

```
%inicializace oblasti a vstupních dat
Experiment = initArea (AreaName, 1, Entrances, 'c:\Report',
Experiment, 'MyName');

% hlavní smyčka se opakuje, dokud není dokončen experiment
% (neproběhla celá doba simulace)

while Experiment.state,

% získání intenzit, obsazenosti, rychlostí a času, ze kterého
```

```
    data pocházejí
    [Experiment, Intensity, Occupancy, Velocity, TimeData] =
    getActualData (Experiment);

    %získání průměrné a maximální délky kolon a času, ze kterého
    statistiky pocházejí
    [Experiment, QueueAverage, QeueMax, TimeStatistics] =
    getActualStatistics (Experiment);

    %výpočet délek fází na základě algoritmu řízení
    %zaslání délek fází a provedení dalšího kroku simulace

    Experiment=setActualControl(Experiment, PhasesLengths);
  end;
```

Po posledním kroku simulace dojde k uložení výstupních dat experimentu a k vytvoření TEXového reportu s popisem oblasti a výstupními daty ve formě grafů atd. Podrobný postup při provádění experimentů v AIMSUN - MATLAB toolboxu je uveden v příloze B.

Kapitola 3

Vlastní práce

V průběhu vývoje AIMSUN - MATLAB toolboxu jsem musel řešit průběžné požadavky od uživatelů a vývojářů toolboxu z ÚTIA AV ČR. Detailní popis své práce (převážne MATLAB funkce) uvádím v kapitole 3.

3.1 Vykreslování historie délek fází

Funkce `plotPhases` slouží pro zobrazení historie délek fází řídícího plánu viz 2.2. Chová se obdobně jako funkce `plot` MATLABu, tzn. matici fází vykreslí do aktuální figury.

```
plotPhases(phases)
```

Popis parametrů :

phases matice historie délek fází v tomto tvaru ($f_1 - f_n$ fáze a na nich signály $t_1 - t_m$)

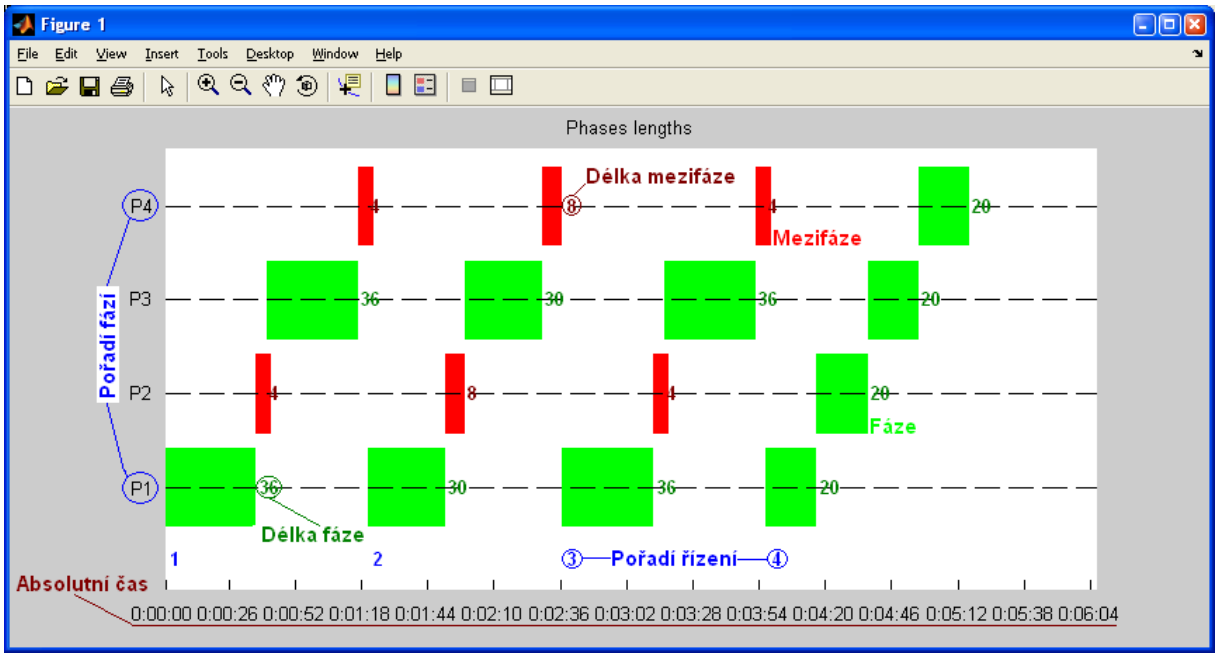
$$\boldsymbol{phases} = \begin{bmatrix} f_1t_1 & f_1t_2 & ... & f_1t_m \\ f_2t_1 & f_2t_2 & ... & f_2t_m \\ f_3t_1 & f_3t_2 & ... & f_3t_m \\ & & ... & f_nt_m \end{bmatrix}$$

Pojmem fáze se rozumí část cyklu, kdy jeden nebo více odbočovacích směrů má uděleno právo projet křižovatkou bez konfliktu s jinými účastníky provozu. Počet řádků

matice tedy odpovídá počtu fází řízené křižovatky. Počet sloupců představuje počet zobrazených signálů.

Funkce `plotPhases` zobrazuje jednotlivé signály jako zelené a červené obdélníky, jejichž délka a výška je závislá na délce signálů, počtu fází aktuálního řízení a na uživatelském nastavení. Prvek matice *phases* se vyhodnotí jako signál "červená" (červený obdélník) tehdy, je-li jeho hodnota menší než konstanta *InterPhase*. Pokud je vetší, je považován za "zelený" signál (zelený obdélník). Na ose x se zobrazuje absolutní čas s počátkem v 0:00:00 a na ose y je zobrazeno pořadové číslo každé fáze. Dále se zobrazuje podélná mřížka u každé fáze. Každý cyklus má ve spodním řádku zobrazeno své pořadové číslo. Ukázka výstupu funkce `plotPhases` pro matici *phases* je na obr. 3.1.

$$\boldsymbol{phases} = \begin{bmatrix} 36 & 30 & 36 & 20 \\ 4 & 8 & 4 & 20 \\ 36 & 30 & 36 & 20 \\ 4 & 8 & 3 & 20 \end{bmatrix}$$



Obrázek 3.1: Výstup funkce `plotPhases` pro matici *phases*

Funkce se musí chovat jako funkce plot, tzn. musí matici *phases* vykreslit do aktuální figury. Pokud žádná neexistuje, dojde k jejímu vytvoření. Algoritmus vykreslení je :

- vytvoření bílého plátna (matice tří elementů, z nichž každý představuje stupeň intenzity červené, modré a zelené barevné složky), jehož výška je závislá na počtu fází v matici *phases* a šířka na součtu délek všech fází v matici *phases*.
- podle matice *phases* se do plátna vykreslí jednotlivé fáze
- transformace plátna funkcí `image()` a jeho vykreslení do aktuální figury funkcí `plot()`
- nastavení rozměrů os x a y
- vykreslení podélné mřížky
- vykreslení popisek (délka každé fáze, pořadí každého řízení)
- výpočet prvků osy x. Aby nedocházelo k zobrazení a následnému překrytí velkého počtu prvků na ose x, je jejich počet omezen na hodnotu definovanou konstantou *XaxisScale*.

Vykreslování je tedy rozděleno na dvě části. Nejdříve se vytvoří potřebná grafika, která se zobrazí do aktuální figury. Teprve poté se do figury (ne do původního plátna) vykreslí všechny textové popisky.
Aby bylo možné ovlivnit vzhled výstupu funkce `plotPhases`, jsou v úvodní části zdrojového souboru funkce uvedeny tyto konstanty :

InterPhase maximální délka "červeného" signálu

GridColor barva podélné mřížky

PhaseColor barva "zeleného" signálu

InterPhaseColor barva "červeného" signálu

BoxColor barva obtažení obdélníků

Title titulek výstupní figury

Xlabel popisek osy x

Ylabel popisek osy y

TextColor barva textu

XaxisScale maximální počet časových značek na ose x

YScale výška mezery mezi jednotlivými fázemi

YBlock velikost obdélníků

FontSize velikost písma

MinBlockWidth minimální šířka "červeného" signálu. Tato konstanta je uvedena kvůli občasnému nevykreslení tenkých objektů MATLABem.

PhaseAxis povolení/zakázání zobrazení os

Volbu těchto konstant je nutné brát uvážlivě, protože může dojít k různým nechtěným efektům při vykreslování (překrývání, nezobrazení části dat).

3.2 Zjištění typu řízení aktuální křižovatky

Každá křižovatka v dopravní oblasti může mít definovaný jeden ze tří typů řízení :

0 - Uncontrolled neřízená křižovatka

1 - Fixed pevné řízení (z AIMSUNu)

2 - External externí řízení (např. z MATLABu)

Pro zjištění typu řízení křižovatky slouží funkce :

```
[JunctionType] = getActualJunctionType(trafficArea,JunctionID);
```

Popis parametrů :

trafficArea název dopravní oblasti

JunctionID jedinečné ID křižovatky v dopravní oblasti *trafficArea*

JunctionType typ řízení křižovatky

Informace o typu řízení každé křižovatky je uložena v souboru *nazev_ridiciho_planu.con*, který se nachází v adresáři dopravní oblasti. Každá dopravní oblast může mít více řídících plánů, které se během simulace mohou postupně přepínat. Pořadí a okamžiky jejich přepínání je možné nastavit v scénáři, nebo mohou být přepínány pomocí API funkcí rozhraní GETRAM Extensions. Rozhraní GETRAM Extensions poskytuje API funkci `ECIGetNameCurrentControl()`, která vrací název aktuálního řídícího plánu.
Pro zjištění typu řízení křižovatky je tedy nutné provést dva kroky :

1. Zjištění názvu aktuálního řídícího plánu

   Pro zjištění názvu aktuálního řídícího plánu je v GETRAM Extensions funkce :

   ```
   char * ECIGetNameCurrentControl();
   Parameters: none
   Output:
   = NULL : name of the control plan
   != NULL : Error
   ```

   Jediným problémem je zpřístupnit tuto funkci MATLAB části toolboxu proto, aby se dala funkce volat během provádění experimentu. Všechny funkce rozhraní GETRAM Extensions, které se používají při experimentu v toolboxu, jsou definovány v knihovně GetExt42R.dll. Funkce pro zjištění aktuálního řídícího plánu je v knihovně GetExt42R.dll definována jako :

   ```
   char *GetActualControlName(){
     return stepInfo.currentControlPlanName;
   }
   ```

   stepInfo je "C" struktura, která obsahuje data a informace vztahující se k aktuálnímu simulačnímu kroku. Funkce `GetActualControlName()` tedy vrací pouze element struktury *stepInfo*. Samotné zjištění názvu řídícího plánu se provádí ve funkci `GetExtManage`, která je volána na začátku každého simulačního kroku viz 2.4.
   Funkci `GetActualControlName` lze z MATLABu v průběhu provádění experimentu

volat pomocí `calllib('libAIM','GetActualControlName')`, kde *'libAIM'* je alias pro dll knihovnu GetExt42.dll, která musí být v okamžiku volání funkce `calllib` načtena ve workspace MATLABu.

2. Pokud se úspěšně podařilo zjistit název aktuálního řídícího plánu, je možné určit typ zadané křižovatky. Informace o všech křižovatkách v dané oblasti jsou uloženy v souboru *nazev_ridiciho_planu.con*, jehož struktura je následující :

```
* Junction 1------------------------------
* idnode, type, critical, offset
1  2  0  0.0 0 0.0000
* nbRings, nbBarriers, singleEntry
1 1 1
* Bus Preemption
* nbBusPreemption
0 . .
* Junction 2------------------------------
* idnode, type, critical, offset
2  2  0  0.0 0 0.0000
* nbRings, nbBarriers, singleEntry
1 1 1 15.0 36.0 90.0 1
```

Pro potřeby funkce `getActualJunctionType` jsou důležitá čtyři čísla na řádku pod řetězcem ** idnode, type, critical, offset*.

První číslo představuje *JunctionID*, druhé číslo typ křižovatky *JunctionType*. Úkolem druhého kroku funkce `getActualJunctionType` je :

- otevření souboru *nazev_ridiciho_planu.con*
- vyhledání řádku v souboru, který přísluší požadované křižovatce *JunctionID*
- přečtení *JunctionType*

Návratovou hodnotou je tedy typ křižovatky (pokud byla nalezena). Pokud nalezena nebyla, je návratovou hodnotou NaN.

3.3 Změna aktuálního řídícího plánu a traffic result

Jak již bylo naznačeno, je možné v průběhu simulace přepínat mezi řídícími plány (control plan). Možné je také přepínat objekty výstupních dat simulace (traffic result). Pro změnu řídícího plánu slouží funkce :

```
setActualControlPlan(trafficArea,controlPlanName)
```

Popis parametrů :

trafficArea název dopravní oblasti

controlPlanName název nového řídícího plánu

Pro změnu traffic result pak funkce :

```
setActualTrafficResult(trafficArea,trafficResultName)
```

Popis parametrů :

trafficArea název dopravní oblasti

trafficResultName název nového traffic result

Změnu řídícího plánu a traffic result v průběhu simulace nelze provést využitím API funkcí GETRAM Extensions. Je nutné změnit obsah souboru *název_dopravni_oblasti.sce*, jehož struktura je např. :

```
...
#TRAFFIC_RESULT
trafficResults
#STATES
NoEntrances
#CONTROL controlPlan 00:00:00
#EXTENSIONS
C:\SVN\doprava\AIMSUN~1\TOOLBO~1\GETEXT~1.DLL
```

Stačí vyhledat klíče *TRAFFIC_RESULT* a *CONTROL*, a poté změnit hodnotu na dalším řádku.

Jelikož dochází k editaci původního scénáře, pro potřeby simulace se vytváří kopie původního scénáře. Po provedení experimentu se musí všechny změny vrátit zpět. Tzn., smazat aktuální scénář a na jeho místo nakopírovat zálohovaný scénář, který neobsahuje změny způsobené funkcemi `setActualControlPlan` a `setActualTrafficResult`.

Pro vytvoření kopie scénáře slouží funkce `copyTrafficArea`. Pro smazání scénáře pak funkce `removeTrafficArea`.

3.4 Instalační funkce uiinstall

Pro snadnou instalaci AIMSUN - MATLAB toolboxu slouží instalační funkce :

```
uinstall(sourceDir, destinationDir, figureTitle, closeAfterInstall,
dictionaryFile)
```

Popis parametrů :

sourceDir povinný parametr, složka, ze které se budou kopírovat soubory

destinationDir nepovinný parametr, defaultní složka, která se zobrazí v políčku *Destination folder*. Pokud je *destinationDir* uvedený znakem '!', není povolena změna cílové cesty.

figureTitle nepovinný parametr, titulek okna instalátoru

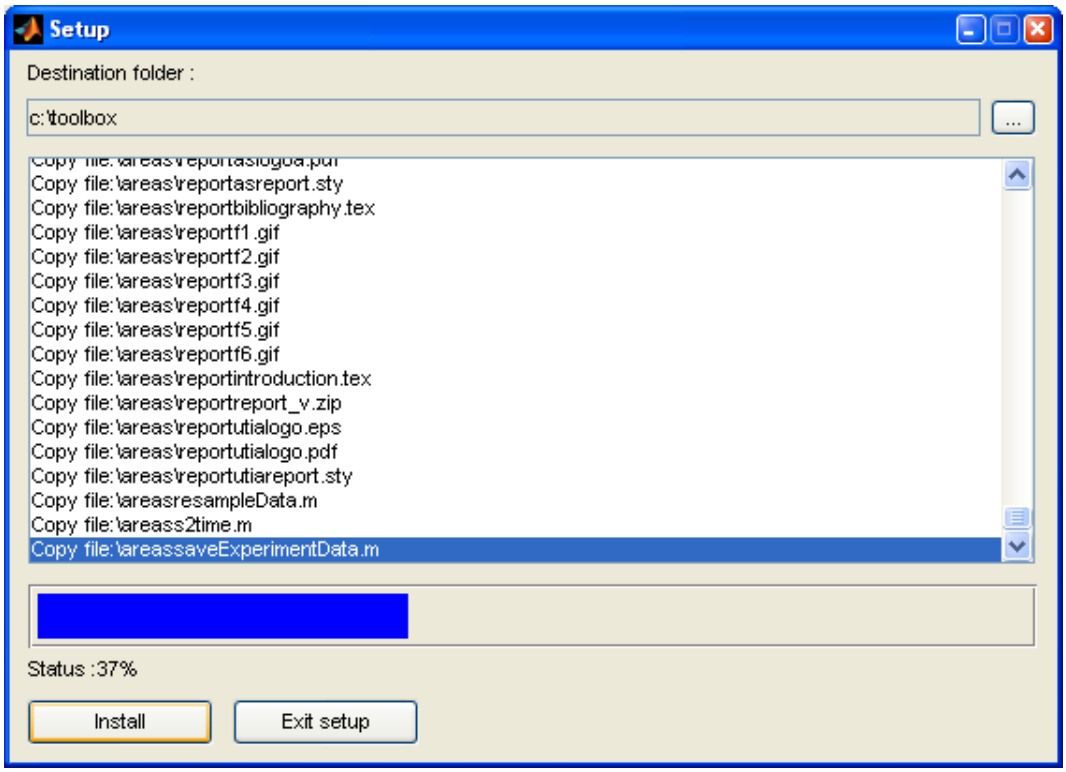
closeAfterInstall nepovinný parametr, zavření okna instalátoru po ukončení instalace

dictionaryFile nepovinný parametr, soubor s instalačním slovníkem

Funkce `uiinstall` je universální, nemusí být použita pouze pro instalaci AIMSUN - MATLAB toolboxu. Při instalaci dochází ke kopírování souborů ze zdrojového adresáře do cílového adresáře. Pokud je součástí názvu zdrojového souboru přípona **.in*, dojde k nahrazení řetezců `@var@` v tomto souboru proměnnou `var`. Například, pokud zdrojový soubor obsahuje souborové cesty ve formátu `@var@\cesta`, je možné na místo proměnné

`var` vložit cílový adresář instalace. Seznam proměnných a jejich hodnot je uložen v instalačním slovníku viz 3.4.2. Pokud není součástí názvu zdrojového souboru přípona **.in*, dojde pouze k překopírování tohoto souboru do cílového adresáře.

Pro komunikaci s uživatelem využívá funkce `uiinstall` grafického uživatelského rozhraní MATLABu. Příklad vzhledu GUI funkce `uiinstall` je na obr. 3.2.



Obrázek 3.2: Uživatelské rozhraní funkce `uiinstall`

Uživatelské rozhraní obsahuje ovládací prvky a zobrazovací prvky. Pokud je povolen výběr cílového adresáře pro instalaci, je možné tlačítkem *"..."*, pomocí klasického výběrového dialogu, tuto cestu zvolit. Pro zahájení instalace slouží tlačítko *"Setup"*. Instalace může být ukončena stiskem tlačítka *"Exit setup"*. Právě kopírovaný/instalovaný soubor se zobrazuje v seznamu souborů. Stav instalace je zobrazován v procentuálním vyjádření a zároveň v grafické podobě (bargraf).

Funkce `uiinstall` slouží pouze pro uživatele (programátora, který ji aplikuje) instalátoru a pro nastavení vzhledu grafického rozhraní viz 3.4.1 . Vytvoření GUI, jeho ovládání a instalaci/kopírování souborů zajišťuje funkce `setupInstall`. Detailní popis této funkce je v kapitole 3.4.2

3.4.1 Změna vzhledu uživatelského rozhraní

Pro ovlivnění vzhledu GUI funkce `uiinstall` jsou definovány tyto konstanty :

Status text v popisce stav instalace

Finished text, který se zobrazí v command line MATLABu po úspěšném ukonční instalace

ScanDir text zobrazující se při prohledávání zdrojového adresáře

CreateDir text zobrazující se při vytáření stromové struktury

SetupButton text tlačítka zahájení instalace

ExitButton text tlačítka ukončení instalace

DefaultDestDir výchozí cílový adresář

CopyFile text zobrazující se v seznamu souborů u každého kopírovaného souboru

InstallFile text zobrazující se v seznamu souborů u každého instalovaného souboru

DestinationFolder text v popisce cílového adresáře

Změnou těchto konstant je možné docílit například různých jazykových mutací.

3.4.2 Popis funkce installSetup

Funkce

```
installSetup(sourceDir,destinationDir,figureTitle,closeAfterInstall,
ExitButton,SetupButton,Status,ScanDir,CreateDir,CopyFile,InstallFile,
Finished,DestinationFolder,dictionaryName,command)
```

slouží pro vykreslení uživatelského rozhraní a pro instalaci/kopírování souborů.

Parametry funkce `installSetup` jsou shodné s parametry a konstantami funkce `uiinstall` stejného názvu. Parametr *command* určuje, jestli je funkce volána pro vytvoření GUI, nebo jako callback funkce (funkce, která se volá když je ovládací prvek aktivován) ovládacích prvků.

Každý ovládací prvek je vytvořen MATLAB funkcí `uicontrol` a má definován svůj jedinečný identifikátor tag. Jako callback funkce je u všech ovládacích prvků použita funkce `setupInstall`. Prvním krokem funkce `setupInstall` je rozhodnutí, zda vykreslit GUI, nebo reagovat na událost ovládacího prvku. Toto rozhodnutí se provede na základě obsahu parametru *command*. Obsahuje buď tag ovládacího prvku (callback funkce), nebo hodnotu *'command'* (vykreslení GUI).

První operací při stisku tlačítka *"Setup"* je vytvoření instalačního slovníku (pokud je zadán jeho název). Formát slovníku je **.csv* (excelovský formát, kde každý sloupec je oddělen středníkem). Pomocí funkcí `separateStrings` se vytvoří slovník jako cell matice `dictionary`.

Dalším krokem funkce `installSetup` je vytvoření seznamu všech souborů a adresářů v zdrojovém adresáři. Tento seznam se vytvoří pomocí funkce :

```
[files,paths] = recursiveDir(path,files,paths)
```

Popis parametrů :

path cesta k adresáři, který se má prohledat

files vektor cest ke všech souborům, které se nachází v zdrojovém adresáři

paths vektor všech adresářů, které se nachází v zdrojovém adresáři

Tato funkce je volána rekursivně, stačí ji tedy pouze zavolat z `installSetup` a funkce automaticky prohledá libovolně "košatou" adresářovou strukturu. Návratovou hodnotou jsou dva jednořadkové vektory. Vektor *files* obsahuje celé cesty všech souborů v *sourceDir* oddělené středníkem. Vektor *paths* obsahuje celé názvy všech adresářů v *sourceDir* oddělené středníkem. Po vytvoření seznamu souborů a adresářů dojde k vytvoření identické adresářové struktury v adresáři *destinationDir*.
Po vytvoření adresářové struktury dojde k překopírování všech souborů ze *sourceDir* do *destinationDir* dvěma zpusoby :

- soubory bez přípony *.in jsou kopírovány binárně funkcí `copyfile`.
- soubory s příponou *.in jsou kopírovány v textovém režimu. Nejdříve se načte obsah celého souboru do jednoho řetězce a poté se pomocí `regexprep` nahradí výskyt všech `@var@` v načteném řetězci. Výsledek se uloží do určeného souboru.

Funkce `uiinstall` vypíše po svém zavolání *sourceDir* do command line MATLABu a pak vypisuje průběh kopírování jednotlivých souborů. Soubory s příponou .in jsou vypisovány jako install file a soubory bez přípony .in jsou vypisovány jako copy file.

Příklad výpisu :

```
>> uiinstall('D:\SVN\doprava\AIMSUN-MATLAB\toolboxASYN\setup','d:\instal')

Scanning directory D:\SVN\doprava\AIMSUN-MATLAB\toolboxASYN\setup
for files to be install ...

Install file :  plotPhases001.m.in
Install file :  lukas01plotphases(2).m.in
Copy file    :  plotPhases.m
Install file :  plotPhases001.m.in
Copy file    :  OA.pdf
Install file :  plotPhases.m.in
. . .
```

3.5 Vytvoření konfig. souboru experimentu pomocí GUI

Před provedením experimentu na dopravní oblasti je nezbytné mít nakreslenou dopravní síť v programu TEDI viz 2.3.1, vytvořený scénář v programu AIMSUN viz 2.3.2 a konfigurační soubor *trafficArea.mat*. Obsahem tohoto souboru je konfigurační datový objekt *MyArea*, obsahující všechny informace nutné pro provedení experimentu. Struktura objektu *MyArea* je :

areaName řetězec s popisem oblasti
např. "Smichov - Prague"

detectors matice ID detektorů. Agregované detektory jsou na stejném řádku. Doplnění do plné matice konstantou *NaN*.
např. [11, 26, 4,NaN]

sections matice ID sekcí. Na sebe navazující sekce jsou na stejném řádku. Doplnění do plné matice konstantou *NaN*.
např. [5, 6, 4,NaN]

junctions řádkový vektor ID křižovatek
např. [1, 2, 3, 4]

masterJunction ID hlavní křižovatky
např. [1]

entranceSection řádkový vektor ID sekcí, do kterých jsou nastaveny vjezdy vozidel (ať už generovaných z AIMSUNu nebo z MATLABu).
např. [4, 5, 6]

defaultControl vektor (matice) délek fází, kde každý řádek popisuje jednu křižovatku. V jednotlivých sloupcích jsou délky jednotlivých fází. Doplnění do plné matice konstantou *NaN*.
např. [52 4 30 4 NaN NaN; 42; 4; 15; 4; 21; 4]

inputSection matice, kde každý řádek obsahuje pár vstupní sekce + detektor. U takto zvolených párů budou v reportu zobrazeny intenzity, obsazenosti a délky odsimulovaných a odhadnutých kolon.
např. [1, 12; 2, 22; 3, 32]

outputSection matice, kde každý řádek obsahuje pár výstupní sekce + detektor. U takto zvolených párů budou v reportu zobrazeny intenzity a obsazenosti.
např. [5, 52; 6, 62]

kappa sloupcový vektor parametrů κ pro každou sekci

beta sloupcový vektor parametrů β pro každou sekci

lambda sloupcový vektor parametrů λ pro každou sekci

MSE sloupcový vektor středních kvadratických chyb (mean square error)

kappa2 sloupcový vektor parametrů κ pro každou sekci (druhá rovnice modelu)

lambda2 sloupcový vektor parametrů λ pro každou sekci (druhá rovnice modelu)

MSE2 sloupcový vektor středních kvadratických chyb (druhá rovnice modelu)

Soubor *trafficArea.mat* s objektem *MyArea* je možné vytvořit ručně v MATLABu, nebo s využitím grafického rozhraní funkce `uicreateArea`.

```
uiCreateArea(AreaName)
```

Popis parametrů :

AreaName název dopravní oblasti implementované v toolboxu

Zavoláním této funkce se zobrazí grafické rozhraní obr. 3.3 umožňující vytvoření objektu *MyArea*. Jednotlivé ovládací prvky odpovídají položkám objektu *MyArea*.

Princip vytoření GUI a reakce na události ovládacích prvků je shodný s funkcí `uiinstall` viz 3.2. Funkce `uicreateArea` slouží pouze pro styk s uživatelem (programátorem) funkce. Veškeré akce vykonává funkce :

```
createArea(command,trafficArea)
```

Popis parametrů :

command callback parametr funkce

trafficArea název dopravní oblasti

Tuto funkci má smysl volat pouze pro dopravní oblasti implementované v toolboxu, tj. oblasti, které se nachází ve složce *areas* toolboxu. Pro neimplementovanou oblast funkce vypíše chybovou hlášku a skončí.

Prvním krokem funkce `uicreateArea` je tedy test na existenci dopravní oblasti a načtení jejich parametrů, tj. všechny sekce, detektory, křižovatky a řídící plán. Popis oblasti je uložen v souborech *sections*, *nodes* a *controlPlan.con*, které se nachází v adresáři *trafficArea* každé oblasti. Struktura souboru *sections* je :

```
* Section --------------------------------
2  @
*  level, nblanes (main), slope (type, percentage, heights)
0 1  2 0.00000 0.00000 0.00000
* contour type: 0 (segment), 1 (polysegment)
0
```

```
...
* nb. detectors
3
* detector: identifier, type, capability
@23@Detector 23 0  13
...
* Section --------------------------------
3  @
...
```

Z tohoto souboru stačí pouze načíst ID každé sekce, které se nachází na řádku pod klíčem `* Section`. ID detektorů ležící na dané sekci se nachází na řádku pod klíčem `* detector`. ID sekcí, resp. detektorů se načtou do vektorů *sections*, resp. *detectors.*

Struktura souboru *nodes* je :

```
* Node --------------------------------
1  1  @Junction J1
...
```

Do vektoru *junctions* se načtou ID všech křižovatek, což jsou první čísla na řádku klíče `@Junction`.

Struktura souboru *controlPlan.con* je :

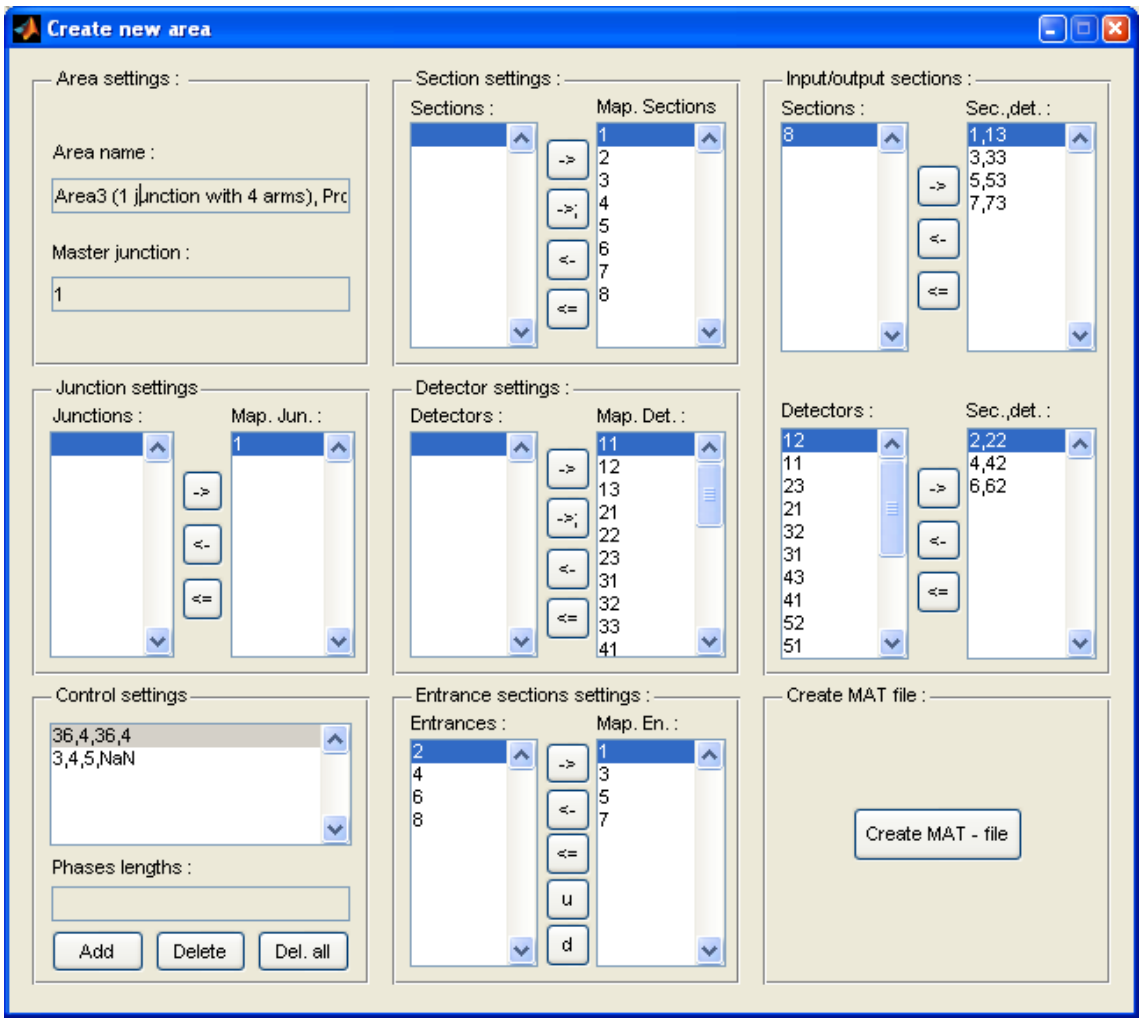
```
* Junction --------------------------------
* idnode, type, critical, offset
1  2  0  0.0 0 0.0000
...
*   nbphases
4
* phase: min, init, max, nb_signals
15.0 36.0 90.0 1
*  signal group
```

```
1
...
* phase: min, init, max, nb_signals
4.0 4.0 4.0 0
* phase: actuated, recall
...
* phase: min, init, max, nb_signals
15.0 36.0 90.0 1
...
* phase: min, init, max, nb_signals
4.0 4.0 4.0 0
...
```

Každá křižovatka má definovány fáze řízení viz 2.2. Jednotlivé signály těchto fází tvoří první čísla, která se nachází na řádku pod klíčem `* phase: min, init, max, nb_signals`. Z takto načtených fází se vytvoří matice signálního plánu *defaultControl*.

Po načtení sekcí, detektorů, křižovatek a řídícího plánu se provede test na existenci konfiguračního souboru *trafficArea.mat*. Pokud existuje, načte se jeho obsah do objektu *MyArea* a zobrazí se v namapovaných sekcích, detektorech a křižovatkách. Ostatní sekce, detektory a křižovatky je možné namapovat pomocí grafického rozhraní.

Přidání jedné položky do namapovaného seznamu se provede tlačítkem `"->"`. Pro přidání agregované položky slouží tlačítko `"->;"`, které přidá položku na vybraný řádek. Odebrání jedné, resp. všech položek seznamu se provede tlačítkem `"<-"`, resp. `"<="`. Stiskem tlačítka `"Create MAT - file"` dojde k vytvoření nebo přepsání souboru *trafficArea.mat*.



Obrázek 3.3: Uživatelské rozhraní funkce `uiicreateArea`

3.6 Podpůrné funkce pro práci s textem

Věškeré informace o scénáři jsou uloženy v textových souborech. Pro jejich načtení je třeba tyto soubory "parsovat". Universální funkce pro práci s textem jsou :

readItem(file, key, row) ze souboru *file* načte obsah klíče *key* na řádku *row*

getSubItem(item, number, separator) z řetězce *item* načte položku na pozici *number*, položky jsou oddělené znakem *separator*

V ovládacím prvku listbox jsou textové informace uloženy jako cell vektor. Pro přidání položky se musí obsah listboxu načíst, převést na řetězec, ve kterém jsou jednotlivé řádky odděleny znakem "|". Funkce pro práci s komponentou listbox jsou :

listAdd(tag, data) přidání jednoho řádku *data* do listboxu *tag*

listAgregate(tag, data) přidání položky *data* na právě vybraný řádek v listboxu *tag*

listGet(tag) vrácení vybraného řádku z listboxu *tag*

listRemove(tag) smazání vybraného řádku z listboxu *tag*

listSwitchUp(tag) přehození vybraného řádku s horním řádkem v listboxu *tag*

listSwitchDown(tag) přehození vybraného řádku se spodním řádkem v listboxu *tag*

Kapitola 4

Závěr

V okamžiku zadání bakalářské práce již byla vytvořena část toolboxu, která zajišťuje komunikaci, synchronizaci a výměnu dat mezi AIMSUNem a MATLABem. V první verzi toolboxu bylo nutné před provedením experimentu napsat skript zajišťující nastavení všech parametrů experimentu. To však vyžadovalo vytvoření zdrojových kódů a určité znalosti programování v prostředí MATLAB. Aby se předešlo programování, navrhl jsem systém, kde veškeré vstupní parametry experimentu jsou soustředěny v konfiguračním souboru *trafficArea.mat*. Tento soubor obsahuje objekt *MyArea*. V první fázi bylo nutné vytvořit tento soubor ručně, pomocí editoru MATLABu. Tento způsob již odboural nutnost psát zdrojové kódy. Dalším vylepšením tohoto systému je grafické rozhraní funkce `uicreateArea`, pomocí kterého je možné tento soubor vytvořit s prakticky nulovou znalostí prostředí MATLAB.

Další složitou operací při práci s toolboxem byla jeho instalace. Soubory toolboxu bylo nutné nakopírovat na uživatelský počítač a poté nastavit cesty s umístěním toolboxu v MATLABu. Obě operace bylo třeba udělat manuálně. Vytvořením funkce `uiinstall` jsem tyto operace eliminoval na prosté volání jedné funkce. Pomocí uživatelského rozhraní je možné vybrat cestu pro umístění toolboxu a vše ostatní se již provede automaticky. Tato funkce navíc umožňuje použít překládací slovník, takže struktura nainstalovaných souborů může vypadat jinak, než struktura zdrojových souborů. Tato funkce je tedy naprosto universální a může být použita pro instalaci jakýchkoliv souborů. Textové popisky grafického rozhraní je možné lehce změnit, tím je zaručena možnost jazykových mutací instalátoru.

Podstatou algoritmu řízení světelných křižovatek je změna délek trvání zelených a červených signálů na křižovatkách. Tyto délky jsou proměnné, jelikož jsou závislé na aktuálním stavu dopravního provozu. Pro grafické srovnání signálních plánů během delšího

časového úseku (typicky jednoho dne) jsem vytvořil funkci `plotPhases`. Tato funkce má podobné vlastnosti jako funkce `plot` MATLABu, tzn. do aktuální figury vykreslí obsah vstupního parametru, kterým je matice signálního plánu pro celý zobrazovaný časový úsek.

Určitou alternativou k řízení křižovatek pomocí řídícího algoritmu je přepínání mezi několika signálními plány podle určitých podmínek. Jelikož rozhraní GETRAM Extensions takovéto přepínání během simualace neumožňuje, musel jsem za běhu simulace měnit obsah souborů scénáře. Toto řešení však není úplně ideální. Nabízí se měnit řídící plán pomocí API funkcí GETRAM Extensions, avšak tato funkce není rozhraním GETRAM Extensions podporována.

Při návrhu algoritmu řízení je také nutné vědět, které křižovatky jsou externě řízené, tzn., o které křižovatky se nestará AIMSUN, ale aplikace realizující řídící algoritmus. V aktuální verzi toolboxu je tato specifikace prozatimně vyřešena nutností zadat tyto křižovatky tvůrcem experimentu. Pro automatické zjištění typu křižovatky jsem vytvořil funkci `getActualJunctionType`, která toto umožňuje pomocí rozhraní GETRAM Extensions a souborů scénáře.

V současné době je AIMSUN - MATLAB toolbox stále ve vývoji, ale je již plně funkční a je využíván na ÚTIA AV ČR, ČVUT FD, UK MFF k dopravním experimentům a testování algoritmů řízení. Aktuální verze toolboxu je ke stažení na adrese : http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/toolboxASYN. Uživ. jméno a heslo je *guest*.

Předpokladem do budoucnosti je rozšíření toolboxu o další uživatelská rozhraní a jejich sjednocení. AIMSUN - MATLAB toolbox je momentálně vytvořen pro program AIMSUN verze 4.2. Jelikož aktuální verzí programu AIMSUN je 5 (s lepší podporou od výrobce než u 4.2), bude nutné provést v toolboxu potřebné změny kvůli jeho kompatibilitě. Pro snadnější instalaci toolboxu se bude vytvářet jeho CD distribuce.

Literatura

[1] *AIMSUN Version 4.2 - User manual [online].*
http://marabu.utia.cas.cz:1800/svn/doprava/Ruzne/AIMSUN_manualy/aimsun_42.pdf

[2] Transport Simulation Systems: *TEDI Version 4.2 - User manual [online].*
http://marabu.utia.cas.cz:1800/svn/doprava/Ruzne/AIMSUN_manualy/tedi_42.pdf

[3] Transport Simulation Systems: *GETRAM Extension Version 4.2 - User manual [online].*
http://marabu.utia.cas.cz:1800/svn/doprava/Ruzne/AIMSUN_manualy/getram_extensions_42.pdf

[4] Gebouský P.: *AIMSUN-MATLAB Interface - User's Guide [online].*
http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/manualASYN/manualASYN.pdf

[5] Nagy I., Homolová J. , Pecherková P. : *Dopravně záavislé řízení silničního provozu ve městech.*
Academy of Sciences of the Czech Republic - Institute of Information Theory and Automation, Prague, 2007

[6] Nagy I. : *AIMSUN Praktický přehled mikro-simulačního software.*
http://www.fd.cvut.cz/department/k611/PEDAGOG/K611THO_soubory/3_AIMSUN.pdf

[7] *ELTODO - Modelování dopravy .*
http://sim.eltodo.cz/teorie.html

Příloha A

Obsah přiloženého CD

Obsah přiloženého CD :

- bp_2007_Lukas Dibelka.pdf : tato práce ve formátu pdf
- codes : zdrojové kódy
- literatura : použitá literatura v elektronické podobě
- toolboxAsyn : aktuální verze AIMSUN - MATLAB toolboxu
- dokumentace : html dokumentace AIMSUN - MATLAB toolboxu
- ukazky simulace : video ukázky simulací v prostředí AIMSUN a AIMSUN 3D

Příloha B

MATLAB - AIMSUN Toolbox 2.1, user's guide

Abstract

This text is the user's guide for installing and using toolbox for performing traffic experiments in Matlab 7.1 (or higher). The simulation is automatically ran in extern application - traffic simulator Aimsun 4.2 [1]. This is part of software package Getram Extensions 4.2, which must be installed, because is necessary for toolbox function. Getram Extensions 4.2 [3] is supported only for Win32 platform, so that toolbox can be installed only on this platform (Windows 2000, Windows XP, etc.)

In this text, there are references to documents on SVN server *UTIA*. There are two ways how to access them. Either by using internet browser, or by using SVN client (e.g. : http://tortoisesvn.net/downloads). Both ways require to log in. Account *guest* with password *guest* can be used for reading.

1 Downloading of toolbox

Actual toolbox version is asynchronous, so enables to send control (phases lengths), obtain traffic statistics (queues lengths) and obtain data from detectors (intensities, occupancies, velocities) with various time cycle. Download link is : http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/toolboxASYN

Development of previous version was suspended, so its functionality is not guaranted, however it can be downloaded from : http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/toolbox

2 Installation of toolbox

There is Matlab script `install.m` for installing toolbox.

This script is placed in root directory of toolbox, generally in directory :

```
[svn_directory] + \AIMSUN-MATLAB\tooloxASYN\
```

In Matlab directory list can be references to new, or old toolbox version. Never both versions together, because they could not work correctly.

If paths in Matlab have already been set to older version of toolbox, install script will delete them and set to new paths. If some problem occurs (during using toolbox), is necessary to check Matlab directory list and old paths remove manually. Correct functionality of installed toolbox is possible to check by running one of these script :

```
% two four-legged intersections, input flows generated internal, only cars.
demoArea ('area2');

% two four-legged intersections, input flows saved offline, 90% cars, 10% buses.
demoArea ('area2', 1, 'buses', 'c:\MyReports');
```

Upwards commands run experiments with fixed control. In first case there will be only cars in traffic area generated directly by Aimsun. In second case offline saved vehicle entrances (cars and buses) will be sent. More detailed description of function `demoArea` is in chapter 6.2. In second case experiment report will be saved in directory "c:\MyReports"

3 Performing experiments

There are four functions to support performing experiments. Functions `initArea`, `getActualData`, `getActualStatistics` and `setActualControl` are more detailed described in next chapters. More detailed description of Matlab functions can be provided by Matlab by calling help command and name of function.

3.1 Initialization - description

Selection of area and input data (i. e. how much vehicles go to various directions and how much are represented cars, buses, etc.)

```
[Experiment] =
initArea(AreaName, VehGenerator, Entrances, ReportPath, Description,Author);
```

Description of parametres :

AreaName string value of area name, which determines where experiment will be performed. List of implemented testing areas could be shown by calling `showAreaList` function (chapter 6). List of implemented testing areas is :

'area1' two four-legged intersections
http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN- MATLAB/toolboxASYN/areas/area1/area1.pdf

'area2' two four-legged intersections
http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN- MATLAB/toolboxASYN/areas/area2/area2.pdf

'area3' one four-legged controlled intersection
http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN- MATLAB/toolboxASYN/areas/area3/area3.pdf

'smichov' Praha Smichov area
http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/toolboxASYN/areas/smichov/smichov.pdf

'zlicin' Praha Zlicin area (only partial support, some attributes are not defined)

VehGenerator method how to generate input flows of vehicles. Three methods are supported :

0 internal generator (vehicles generated by Aimsun)

1 offline sent vehicle entrances (i. e. entrances measured and saved in file)

2 vehicle entrances sent real-time (in file are saved only information about types of vehicles, etc.)

List of implemented testing vehicle entrances for method 1 and 2 could be shown by calling `showAreaList` function (chapter 6). If this parameter is not used, the 0 (internal generator) method is selected.

Entrances string value of file name, in which traffic data (entrances to several traffic lanes) are saved, including representation of several types of vehicles. List of implemented testing entrances could be shown by calling `showAreaList` function (chapter 6). List of implemented testing entrances is :

'vehicles' modified real data from Smichov area, all vehicles are cars (4m long and 2m wide)

'buses' modified real data from Smichov area,
90% are cars (4m long and 2m wide)
10% are buses (12m long and 2.5m wide)

Entrance parameter is optional. If it is not set, data is automatically loaded from *'vehicles'*, i. e. all vehicles are cars.

ReportPath name of directory, where TeX report about experiment is saved. If path is empty, no report is saved.

Description soever long description of experiment. As name of experiment is used first sentence from this description (finished by dot "." or by two backslashes)

Author name and contact (preferably email address) of author of experiment

Experiment generated structure of experiment data, which contains information about experiment and traffic area.

3.2 Obtaining traffic data

This function returns last aggregated data from detectors as column vector of appropriate variables, where rows are data from appropriate detectors.

```
[Experiment, Intensity, Occupancy, Velocity, Day, Time] = getActualData (Experiment);
```

Description of parameters :

Experiment generated structure of experiment data, which contains information about experiment and traffic area.

Intensity vector of intensity

Occupancy vector of occupancy

Velocity vector of velocity

Day day, from whom data comes from

Time time in seconds, from whom data comes from (time is released at the beginning of each of next day)

Within the context of upwards described function, following structure attributes are interesting.

Experiment.experiment.detectionInterval duration of gathering data (i. e. how long time slot assigned data represents)

Experiment.detectors matrix of detectors for determination which data belongs to everyone detector. Each row represents one or more aggregated detectors. Notation of these detectors is in more detail described in area documentation.

3.3 Obtaining queue lengths

This function returns queue lengths as column vector of appropriate variables, where rows represent traffic lanes.

```
[Experiment, QueueAverage, QeueMax, Day, Time] = getActualStatistics (Experiment);
```

Description of parameters :

Experiment generated structure of experiment data, which contains information about experiment and traffic area.

QueueAverage vector of average queue lengths.

QueueMax vector of maximum queue lengths.

Day day, from whom data comes from

Time time in seconds, from whom data comes from (time is released at the beginning of each of next day)

Within the context of upwards described function, following structure attributes are interesting.

Experiment.experiment.statisticsInterval duration of gathering statistic data (i. e. how long time slot assigned statistic data represents)

Experiment.sections vector of traffic lanes (sections) to uniquely determination of pertinence statistic data. Each row represents one traffic lane. Notation of these lanes is in more detail described in area documentation.

3.4 Performing one simulation step

This function sets all phases lengths on all controlled junctions and performs one simulation step.

```
Experiment =
setActualControl (Experiment, PhasesLengths, EstimatedQueues,RealEntrances);
```

Description of parameters :

Experiment generated structure of experiment data, which contains information about experiment and traffic area.

PhasesLengths matrix of phases lengths, where each row represents one controlled junction and one column represents one phase. Each row matches to related junction in row vector *Experiment.junctions*. If junctions have various number of phases is necessary to fill in matrix by *NaN*.

EstimatedQueues column vector of user estimated queues lengths on particular sections. If this parameter is not set, there is not confrontation of estimated and real queue lengths in report.

RealEntrances column vector of real time generated entrances, whose number of rows is same as number of rows of vector (matrix) of sections *Experiment.sections*. Each row matches to related section in *Experiment.sections*. This parameter has purpose, only if type of vehicle generator 2 (chapter 3.1) has been set. If this parameter is not set (and has purpose), vector *RealEntrances* is completed by zeros entrances.

Within the context of upwards described function, following structure attributes are interesting.

Experiment.state state variable indicating phase of simulation. Simulation is finished when this variable is set to 0. For usage see below at illustration of feedback loop (chapter 4)

Experiment.junctions vector of controlled junctions. Notation of these junctions is in more detail described in area documentation.

Experiment.sections vector (matrix) of input sections. Notation of these sections is in more detail described in area documentation.

4 Feedback loop

Illustration of main feedback loop with control :

```
%initialization of area and of input data
Experiment = initArea (AreaName, 1, Entrances, 'c:\Report', 'Experiment, 'MyName');
%main loop is repeated until experiment is not finished
%experiment has not been finished yet
while Experiment.state,

  % obtaining intensity, occupancy, velocity and time from whom data comes from
  [Experiment, Intensity, Occupancy, Velocity, TimeData] =
  getActualData (Experiment);

  % obtaining average and maximum queue length and time from whom statistic data comes from
  [Experiment, QueueAverage, QeueMax, TimeStatistics] =
  getActualStatistics (Experiment);

  % by user written code, where is proposed appropriate control (phases lengths)

    calculation of PhasesLengths

  % sending phases lengths and performing next simulation step
  Experiment=setActualControl(Experiment, PhasesLengths);
end
```

In chapter 2 is described, how to run demo example, which contains this feedback loop.
Source code is mentioned at link :
http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/toolboxASYN/areas/demoArea.m

5 Generating of report

5.1 Manual startup of report generation from Matlab command line

Below described function enables to additionally make report about one or several experiments (usually max. 4) and confront results graphically and numerically.

```
[ReportFileName, ReportVersion] =
createTEXReport (Experiments, ReportPath, Version, Params);
```

Description of parameters :

Experiment - generated structure of experiment data, which contains information about experiment and traffic area. Instead of one structure is possible to set array of these structures, thereby the compare report is created. Only experiments on same areas and set with same parameters (vehicle entrances, simulation length, etc.) could be compared.

ReportPath name of directory (path), where the TeX report about experiment is saved. If this parameter is not set, function is terminated with error.

Version string value of version number. If this parameter is not set, the automatic increment is used.

Params configuration of report format, this parameter has not been implemented yet in actual version of toolbox.

All necessary basis for generating report (patterns, pictures, etc.) are placed in directory *ReportPath\report*. Final documentation in pdf format is possible to create by compiling TeX file with report. Word *"report"* is part of name of documentation (e. g. *"smichov_vehicles_report.tex"*).

Another interesting basis for creating documentation are pre-filed files with aim of experiment and with conclusion, which name is same as name of the main file of report. Word *"report"* is replaced by *"aims"* and *"conclusion"* (e. g. *"smichov_vehicles_conclusion.tex"*). These files could be used for generation of report more times.

5.2 Manual startup of report generation by using graphical interface

This function could be called instead of previous function :

```
uireport();
```

This function finds out (from user) from which experiments is report created (Fig. 1) and where report is saved (Fig. 2).

5.3 Automatic startup of report generation after the end of experiment

Data from experiment are automatically saved at the end of experiment (i. e. after last simulation step, chapter 3.4). TeX report is created by calling *createTEXReport* function (chapter 5.2). Report and data are saved in directory which is set in *initArea* function 3.1).

Illustration of created reports :
http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/toolboxASYN/areas/smichov/smichov_report.pdf
http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/toolboxASYN/areas/area2/area2_report.pdf

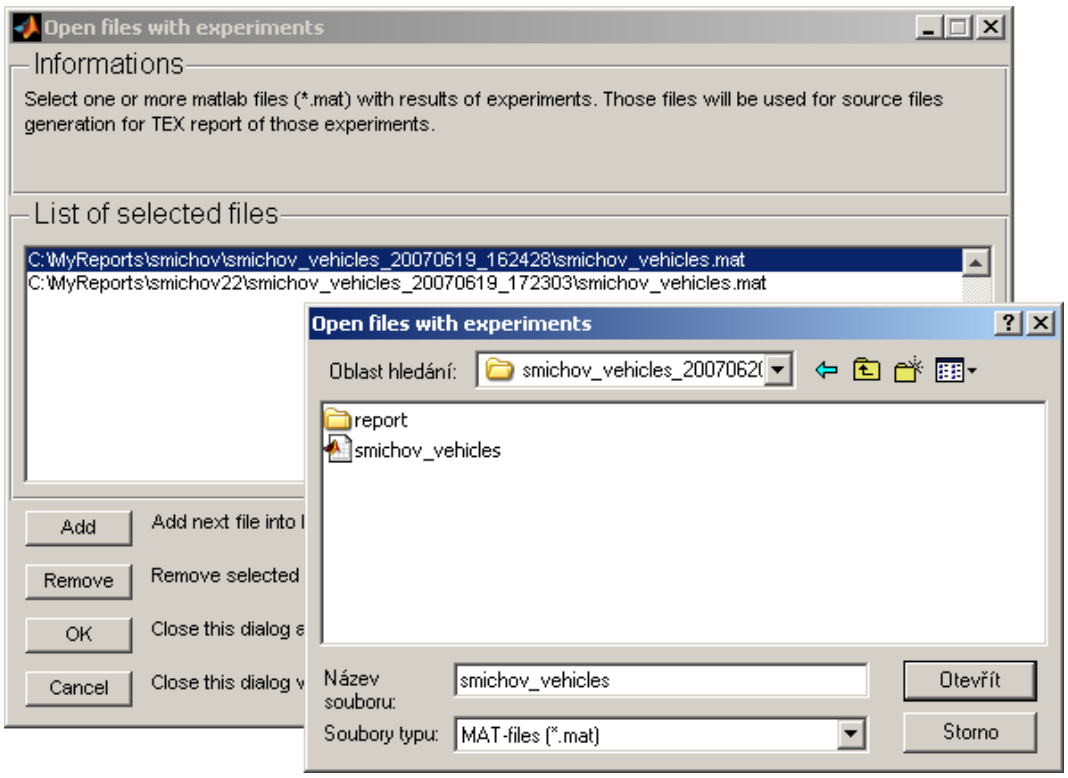


Figure 1: Selection of experiments

6 Other usefull functions

Other functions are introduced in this chapter. Details about usage and function parameters could be shown in Matlab by calling *help* function.

afterError termination of Aimsun, e. g. if experiment was terminated by $CTRL + C$ command

getKappaLambda estimation of values λ, κ

getKappaBetaLambda estimation of values λ, β, κ

showAreaList shows traffic area and entrances list which could be used for experiments.

6.1 Getting actual and total simulation time

Description of parameters :

actualSeconds actual simulation time (i. e. which time slot is simulated)

totalSeconds total simulation time (i. e. how long time slot will be simulated)

6.2 Demo example

```
demoArea  (AreaName, VehGenerator, Entrances);
```

Starts up demo example of toolbox usage. Function parameters are same as parameters of *initArea* function (chapter 3.1). Demo example is in chapter 2. Function *demoArea* starts up experiment with fixed control and at the end of experiment will generate report using *createTEXReport* function. Report will be saved in directory *"c:\experimentReports"*. Details about parameters, which could be used, could be shown by calling *showAreaList* function.

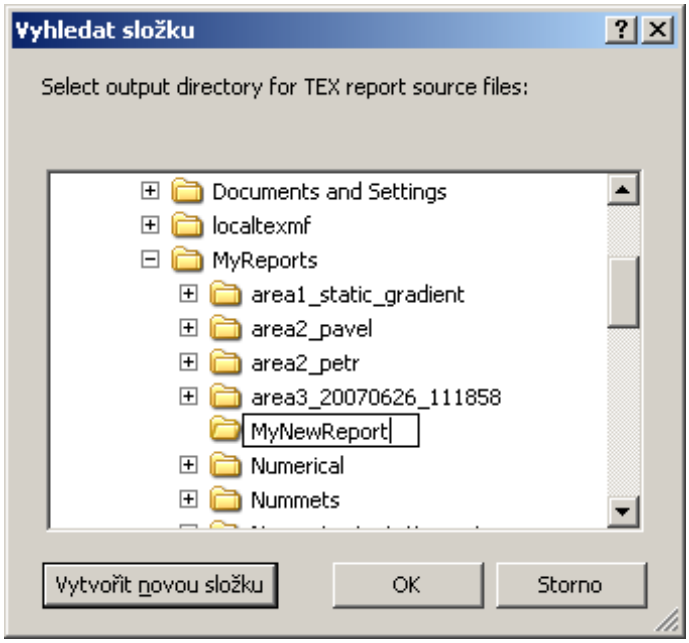


Figure 2: Selection of report folder

7 How to add new testing area

In this chapter are intructions, how to create new testing area and how to implement it to toolbox. List of implemented testing area could be shown by calling *showAreaList* function.

For adding new area is necessary to do these steps :

1. In subdirectory *"\areas"*, which is placed in toolbox, are testing areas. In this directory is necessary to create new subdirectory named as new testing area. This name should be in DOS 8.3 format (i. e. maximum length is 8 characters, without spaces and special characters).
2. In TEDI editor
   (a) create area (traffic network) including traffic lanes, junctions, etc. (dimensions, capacities,...). This area has to be saved in directory created in previous step. Area has to have identical name as previous created directory.
   (b) set up type of control of all controlled junctions to EXT (external control)
   (c) create fixed (static) control plans
   (d) configurate types of vehicles (lengths, widths, velocities,...)
   (e) set up vehicle entrances to sections (intensities, deviations rates)
3. In Aimsun simulator
   (a) create new scenario, which will be saved in directory from step 1 and will be named like this directory
   (b) is necessary to set up traffic area (control, entrances) in this scenario
4. In Matlab
   (a) create structure Area with these items (case sensitive)
   areaName string value containing area description *e. g. "Smichov - Prague"*
   detectors vector (matrix) of ID numbers of detectors. Aggregated detectors are at same row. For completion to full matrix is used *NaN* value.
   e. g. [11, 26, 4,NaN]

sections vector (matrix) of ID numbers of sections. Consequential sections are at same row. For completion to full matrix is used *NaN* value.
e. g. [5, 6, 4,NaN]

junctions row vector of ID numbers of junctions.
e. g. [1, 2, 3, 4]

masterJunction ID number of main junction.
e. g. [1]

entranceSection row vector of ID numbers of sections, where input flows are set up (generated by Matlab or by Aimsun).
e. g. [4, 5, 6]

defaultControl vector (matrix) of phases and inter phases lengths. Each row represents one junction. Columns represents phases lengths. For completion to full matrix is used *NaN* value.
e. g. [52 4 30 4 NaN NaN; 42; 4; 15; 4; 21; 4]

inputSection matrix, where each row represents pair input section + detector. For these selected pairs will be intensities, occupancies, simulated and estimated queue lengths shown in report.
e. g. [1, 12; 2, 22; 3, 32]

outputSection matrix, where each row represents pair output section + detector. For these selected pairs will be intensities and occupancies shown in report.
e. g. [5, 52; 6, 62]

Notation of detectors, sections and junctions have to be identical as in Aimsun.

(b) is necessary to save structure Area as MAT - file to traffic area directory from step 1. Name has to be identical as name of traffic area.

References

[1] Transport Simulation Systems: *AIMSUN Version 4.2 - User manual [online].* http://marabu.utia.cas.cz:1800/svn/doprava/Ruzne/AIMSUN_manualy/aimsun_42.pdf

[2] Transport Simulation Systems: *TEDI Version 4.2 - User manual [online].* http://marabu.utia.cas.cz:1800/svn/doprava/Ruzne/AIMSUN_manualy/tedi_42.pdf

[3] Transport Simulation Systems: *TEDI Version 4.2 - User manual [online].* http://marabu.utia.cas.cz:1800/svn/doprava/Ruzne/AIMSUN_manualy/getram_extensions_42.pdf

[4] Gebouský P.: *AIMSUN-MATLAB Interface - User's Guide [online].* http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/manualASYN/manualASYN.pdf

[5] Dohnal P.: *AIMSUN-MATLAB Toolbox - Nvod k pouit [online].* http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/manualMAT/manualMAT_cz.pdf

[6] Dohnal P.: *Popis experimentované testovací oblasti Area1 (2 junctions with 3 arms), Proving Ground [online].* http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/toolboxASYN/areas/area1/area1.pdf

[7] Dohnal P.: *Popis experimentované testovací oblasti Area2 (2 junctions with 4 arms), Proving Ground [online].* http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/toolboxASYN/areas/area2/area2.pdf

[8] Dohnal P.: *Popis experimentované testovací oblasti Area3 (1 junction with 4 arms), Proving Ground [online].* http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/toolboxASYN/areas/area3/area3.pdf

[9] Dohnal P.: *Popis experimentované testovací oblasti Smichov, Prague [online].* http://marabu.utia.cas.cz:1800/svn/doprava/AIMSUN-MATLAB/toolboxASYN/areas/smichov/smichov.pdf

[10] Dohnal P.: Bayes Estimation of a Queue Length. In *Proceedings of the 7th International PhD Workshop: Interplay of societal and technical decision-making, Young Generation Viewpoint.* Academy of Sciences of the Czech Republic - Institute of Information Theory and Automation, Prague, 2006, pp. 1-8.

[11] Kratochvílová J., Nagy, I.: *Lokální řízení městské dopravy.* Academy of Sciences of the Czech Republic - Institute of Information Theory and Automation, Prague, 2004.