Univerzita Hradec Králové

Fakulta informatiky a Managementu

Katedra informačních technologií

Simulace dopravy v Repast Simphony

Bakalářská práce

Autor: Tomáš Baier

Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. RNDr. Kamila Štekerová, Ph.D.

Hradec Králové Květen 2021

Prohlášení

Prohlašuji, že jsem bakalářskou práci Simulace dopravy v Repast Simphony vypracoval samostatně za použití v práci uvedených pramenů a literatury.

Poděkování:

Děkuji své vedoucí bakalářské práce doc. RNDr. Kamile Štekerové, Ph.D za její ochotu a pomoc

Anotace

Výstupem práce je návrh simulace semaforově řízené dopravy v Repast Simphony. V první části jsou vysvětleny důležité pojmy a postupy při tvoření agentových modelů. V souvislosti s tím jsou také popsány typy agentů a typy prostředí. Druhá část obsahuje návrh simulace dopravy a její následná implementace za pomoci Repast Simphony. Po implementaci jsou pak provedeny na simulaci různé experimenty.

Annotation

Title: Traffic simulation in Repast Simphony

The output of this theses is a design of light-controlled traffic simulation. The first part explains important concepts and creation of agent-based models. Different types of agents and environment are described. The second part presents a design of the traffic simulation and its implementation in Repast Simphony. Several experiments are run.

Obsah

Obrázky

Výpisy

Grafy

1. Úvod

Agentové modelování je rychle se rozšiřující způsob tvorby modelů obsahující autonomní a interagující agenty. Proč je nutnost pro takovýto způsob modelování? Jelikož lidská představivost dokáže dosáhnout pouze určitých mezí, je nutno komplexnější problémy namodelovat. Tyto modely následně umožňují rozsáhlé testování, které by v reálných podmínkách bylo velice obtížné, nebo přímo nemožné.

Agentové modely jsou využity v řadě vysoce důležitých odvětví, jako je například epidemiologie, doprava, ekonomie a spousta dalších.

Cílem této bakalářské práce je v teoretické části přiblížit principy a základy agentového modelování. Poté budou vysvětleny rozdíly mezi typy agentů, kteří jsou děleni na základě jejich složitosti, způsobu řešení problémů a funkcí. Dále popíšeme prostředí, v němž agenti působí, z hledisek a potřeb dané simulace.

Dále se více zaměříme na simulace dopravního provozu, a vysvětlíme problematiku tohoto odvětví. Bude popsána také problematika řízení křižovatek za pomocí semaforového řízení, a dále vysvětleny způsoby, jakými může být taková situace řešena. Představené způsoby řízení semaforů jsou poté modelovány v praktické části. Ke konci této kapitoly je pak ukázána řada modelů, které simulují dopravu.

Poslední kapitolou teoretické části je popis sady nástrojů pro agentové modelování Repast Simphony. V této kapitole je představena tato sada nástrojů, její historie, vývoj a její možnosti. Následně je porovnána s jinými modelovacími programy.

Zkoumaný problém praktické části práce bude tvorba agentových modelů v Repast Simphony. V rámci této části je navržen a implementován model semaforově řízené křižovatky, která využívá řady způsobů pro řízení intervalů.

Na vytvořeném modelu je následně prováděna řada experimentů, které měří efektivitu implementovaných způsobů řízení intervalů na předem definovaných situacích.

1. 1. Motivace

Řízení dopravy je jedním z vysoce sofistikovaných problémů dnešní doby. Kvůli neustálému nárustu obyvatelstva a množství používaných automobilů, je nutné správně a efektivně upravovat a rozšiřovat dopravní infrastrukturu. Využitím inteligentních způsobů řízení dopravy lze dosáhnout efektivnější propustnosti stávajících dopravních systémů. Pro tvorbu dopravních simulací je nutné brát v potaz velkou řadu faktorů, které ji ovlivňují. Těchto faktorů je nespočet, v rámci této práce mohu ale vyzkoušet implementaci alespoň nějakých z nich.

1. 2. Cíl

Cílem práce je vytvoření simulace dopravy na semaforově řízené křižovatce

a. Implementace statického řízení semaforové křižovatky
b. Implementace ovládání semaforů za pomoci senzorů a částečného/plného řízení signálů

Dále budou provedeny a vyhodnoceny experimenty

1. 3. Metody

Metody pro dosažení této bakalářské práce jsou

1. **Literární rešerše agentového modelování**

Informace o principech agentového modelování, agentů a jejich typů dle složitosti a také prostředí, ve kterém se pohybují.

2. **Analýza reálné semaforové dopravy**

Analýza reálné semaforové dopravy je důležitá pro dosažení realistické implementace chování semaforů v simulaci, umožňující testování, které přibližně odpovídá reálným situacím.

3. **Pozorování a experimentování na vytvořeném modelu**

Pozorováním a experimentováním na modelu je dosažena odpověď na hypotézu, že částečné nebo plné řízení signálů je efektivnější způsob řízení dopravy než nastavení statických intervalů.

2. Teoretická část

2. 1. Principy modelování a simulací

Agentové modelování a simulace je přístup pro tvoření komplexních modelů, které se skládají z interagujících, autonomních robotů (agentů). Agenti mají určité chování, většinou definované řadou jednoduchých pravidel a interakcemi s ostatními agenty, což nadále jejich chování ovlivňuje [1]. Existuje řada přístupů k modelování agentových modelů. V případě modelování individuálních agentů se plný efekt rozmanitosti jejich atributů a chování dá pozorovat v průběhu tvoření chování systému jako celku [1]. V případě modelování systému od základů (agent za agentem a interakce za interakcí), se v takovýchto modelech dá často pozorovat sama organizace systému [1]. Vzory, struktury a chování systému, které nebyly přímo naprogramovány, vznikají skrze interakce mezi agenty.

Důraz na modelování heterogenity (rozmanitosti) agentů a vznik samo-organizace systému jsou dva faktory, které dělí agentové modelování od ostatních [1]. Agentové modelování lze využít pro modelování sociálních systému, které se skládají z agentů, kteří mezi sebou interagují, navzájem se ovlivňují, učí se ze svých zkušeností a adaptují své chování, aby do prostředí lépe zapadli.

Samotný princip agentového modelování spočívá ve využití agentů (softwarových, heterogenních entit), kteří představují reálné jednotky systému, který je sledován. Tyto jednotky jsou zařazeny do definovaného kontextu prostředí, ve kterém konají a reagují na nastalé situace.

Agent

Agent je hlavní součástí agentových systémů. Agent pomocí senzorů vnímá své prostředí a na základě vjemů v tomto prostředí provádí akce.

I když neexistuje jedna přesná definice agenta, řada definic obsahuje body, které se shodují.

Chování nezávislé komponenty (agenta) se může pohybovat od primitivního reaktivního chování po vysoce komplexní adaptivní inteligenci [3].

Agent musí být komponenta, která je schopná se nějakým způsobem učit ze svého prostředí a změnit své chování reakcí na data tímto způsobem získána [3].

Obrázek 1 Model agenta, přeloženo z [2]

Z praktického hlediska na modelování mají agenti řadu vlastností:

„*Agent je rozpoznatelný diskrétní jedinec s řadou vlastností a pravidel řídících jeho chování a rozhodovací schopnosti. Agenti jsou soběstační. Diskrétní požadavek agenta naznačuje že má hranice, které lehce určí, jestli je něco součástí agenta, jestli není, nebo jestli je to sdílená vlastnost*“ [2]

„*Agent je umístěný v prostředí, ve kterém interaguje s ostatními agenty. Agenti obsahují protokoly o interakci s ostatními agenty (komunikační protokoly) a schopnost odpovídat prostředí. Agenti jsou schopni rozpoznat a rozlišit vlastnosti ostatních agentů*“ [2]

„*Agent je řízený cílem. Má cíle, kterých se snaží dosáhnout s ohledem na jeho chování.*“ [2]

„*Agent je autonomní. Je schopen fungovat samostatně ve svém prostředí a interagovat s ostatními agenty alespoň skrze omezený rozsah situací.*“ [2]

„*Agent je flexibilní a disponuje schopností se učit a přizpůsobit své chování na základě jeho zkušeností. Toto vyžaduje paměť. Agent může obsahovat pravidla, která upravují jeho pravidla chování.*“ [2]

Agenti jsou rozmanití a různorodí ve svých atributech a pravidlech chování. Pravidla chování se liší jejich složitostí, a tím jaké množství informací agent zvažuje v průběhu

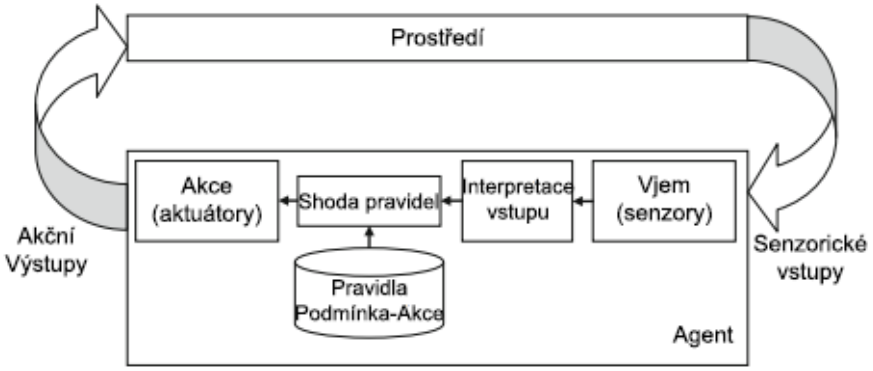
svého rozhodování, agentově vnitřním modelu okolního světa (a ostatních agentů) a rozsahem paměti předcházejících událostí, které si agent uchovává pro použití při rozhodování.

Typy agentů

Agenti jsou rozdělování do speciálních skupin podle jejich složitosti. Je dobré si pro řešený problém vybrat architektonicky co nejjednoduššího agenta, musí však být dostatečný na splnění modelovaného úkolu.

Jednoduchý reflexní agent

Nejjednodušší typ agenta. Výběr akce agenta spočívá pouze na jeho aktuálním vjemu, a nebere v potaz žádné předchozí akce, které v minulosti provedl. Tento typ agenta je vybaven bází vědomostí zvaných pravidla podmínka-akce, které pomáhají agentovi vybrat akci na základě stavu okolního prostředí [4].



Obrázek 2 Jednoduchý reflexní agent, přeloženo z [4]

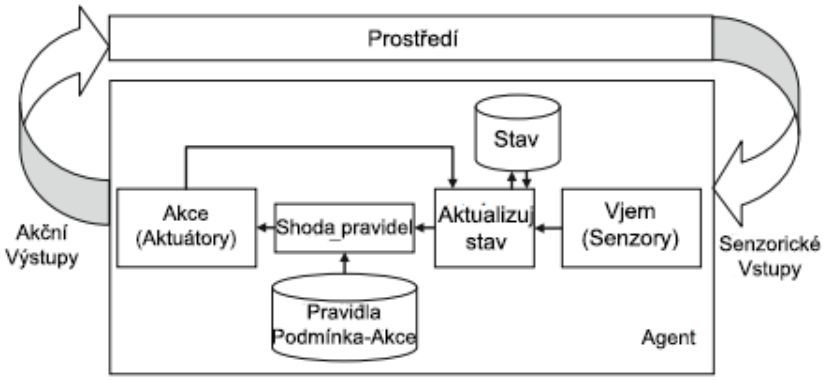
Obrázek ukazuje, že agent může úspěšně splnit svou akci za předpokladu, že správného rozhodnutí je možné dosáhnout pouze na základě stávajícího vjemu. Toho je však možné dosáhnout pouze v plně pozorovatelných prostředích [4], je tedy nutné zvolit jiný typ agenta pro prostředí s pouze částečnou pozorovatelností.

Tento typ agenta za pomocí senzorů získá vjem, který nějakým způsobem interpretuje. Interpretovaný vjem následně porovná s pravidly, která má nastavené v bázi vědomí podmínka-akce, a v případě shody, na základě tohoto vjemu provádí odpovídající akci v prostředí.

Modelově založený reflexní agent

Tento typ agenta je používán v případech, kdy jsou stavy prostředí pouze částečně pozorovatelné. Nepozorovatelné části prostředí sleduje pomocí vnitřního stavu. Vnitřní stav je tvořen na základě historie vjemů [4]. Tímto způsobem alespoň nepřímo zachytí nějaké části nepozorovatelných stavů.

Aktualizace vnitřního stavu potřebuje dva druhy informací. Prvním typem je informace o evoluci prostředí nezávislé na akcích agenta, zatímco druhý typ je informace o dopadu akcí agenta na dané prostředí [4]. Vědomosti o fungování prostředí se říká „model prostředí“.

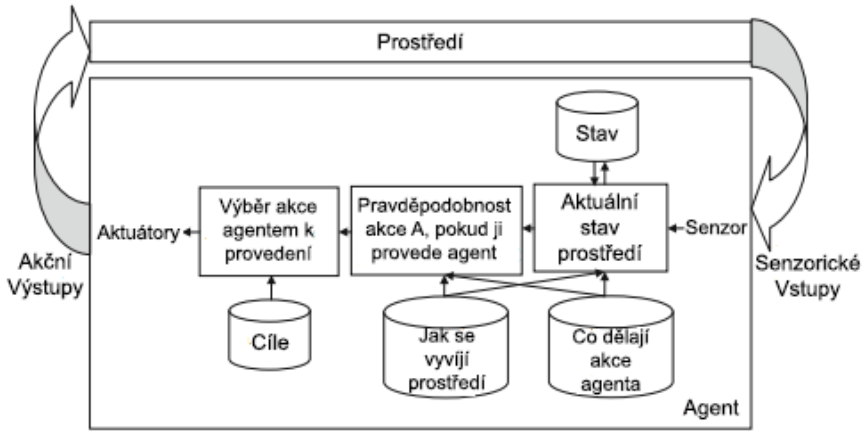


Obrázek 3 Modelově založený reflexní agent, přeloženo z [4]

Agent s reprezentací cíle

Tento typ agenta má společně s reprezentací stávajícího okolí také určenou cílovou informaci, která agentovi stanovuje žádoucí situaci, které by měl svými akcemi dosáhnout.

Agent tohoto typu využívá dvou druhů informací. Prvním z nich je „cílová informace“, která mu napomáhá při identifikaci akce pomocí, které dosáhne stanoveného cíle [4]. Druhý typ informací jsou stavy nepozorovaného okolí, které jsou zachyceny vnitřním stavem, a napomáhají agentovi konat uspokojivé akce v částečně pozorovaném okolí [4].



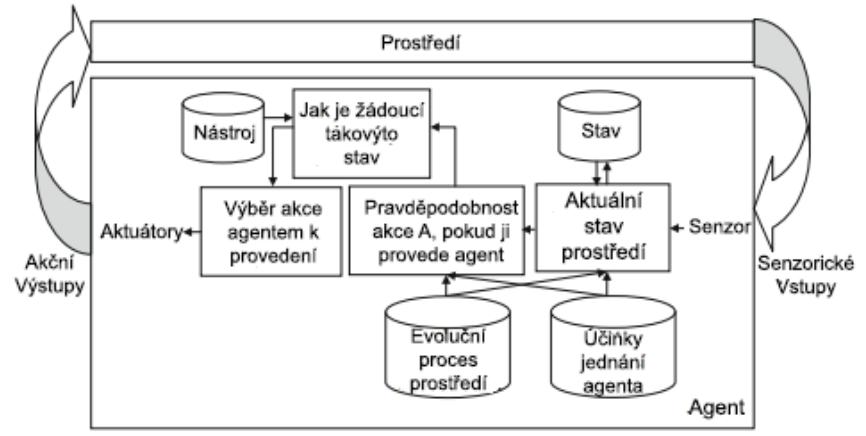
Obrázek 4 Agent s reprezentací cíle, přeloženo z [4]

Řešení agenta s reprezentací cíle může být velice jednoduché, například v případě, kdy jedna akce přímo dosáhne cíle. Avšak ve většině případů z reálného života takto lehké řešení nenastane. Agent musí brát v potaz velice dlouhou sekvenci možných akcí a z nich musí vybrat právě ty, které dosáhnou žádoucího cíle.

Agent s užitkovou funkcí

Tento typ agenta je obohacený agent s reprezentací cíle. Agent navíc obsahuje užitkovou funkci, která ohodnocuje vhodnost stavu nebo sekvence stavů [4].

Racionální agent vybere pomocí užitkové funkce akci s cílem maximalizovat předpokládaný užitek [4]. Užitková funkce nejvíce napomáhá v případě dvou či více konfliktních cílů. V případě nejistoty při výběru určitých cílů pak užitková funkce každému z nich přiřadí váhu na základě jejich důležitosti [4].



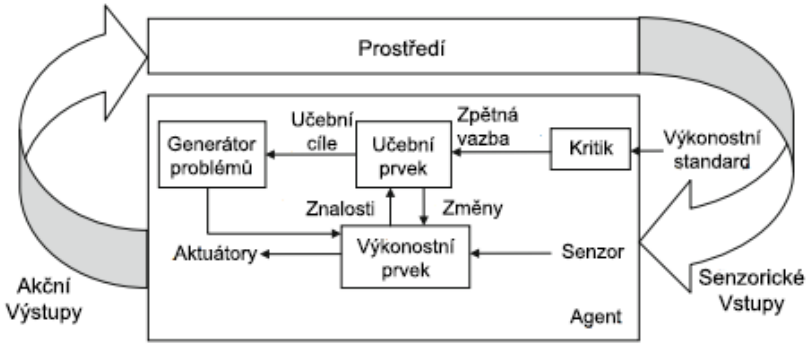
Obrázek 5 Agent založený na utilitách, přeloženo z [4]

Učící se agent

Učící se agent se skládá ze čtyř částí. Učícího elementu, výkonnostního elementu, kritika a generátoru problémů [4].

Výkonnostní element má za úkol výběr vnější akce na základě vstupního vjemu. Kritik hodnotí výkon agenta za pomoci stanoveného výkonnostního standardu a zajišťuje, že učící element je aktualizován potřebnými informacemi. Učící element vezme informace od kritika, a pomocí nich se snaží zlepšit výkonnostní prvek agenta. [4]

Výkonnostní prvek vždy navrhne nejlepší akci na základě jeho stávajících znalostí. Avšak zkoušení neoptimálních stavů agenta v prvních fázích úkolu může ke konci vézt k jeho zlepšení. Na návrh takovýchto akcí slouží právě generátor problémů.



Obrázek 6 Učící se agent, přeloženo z [4]

Prostředí agenta

Dalším důležitým prvkem agentového modelování je prostředí, ve kterém se model odehrává. Agent v prostředí vykonává své akce a interaguje s ostatními agenty. Tato prostředí se dělí podle těchto šesti charakteristik.

Plně pozorovatelné a částečně pozorovatelné prostředí

V případě plně pozorovatelného prostředí je jeho kompletní stav přístupný agentům skrze jejich senzory. V tomto případě pak agent nemusí sledovat prostředí pomocí vnitřního stavu, což snižuje náklady na údržbu [4]. Toto prostředí je jedno z nejjednodušších pro agentovo rozhodování.

Částečně pozorovatelné prostředí je způsobeno dvěma důvody. Prvním z nich je špatná charakteristika senzorů agenta, což způsobí nepřesné měření prostředí [4]. Druhým z nich jsou nedostatečné senzory pro zachycení celého stavu prostředí [4].

Deterministické a stochastické prostředí

Deterministický stav prostředí je plně stanovený jeho stávajícím stavem a akcemi, které agent v prostředí provádí [4]. V jiném případě je prostředí považováno za stochastické.

Agent, který se nachází v plně pozorovatelném, deterministickém prostředí se nemusí zabývat problémy nejistoty. Částečně pozorovatelné komplexní prostředí je většinou stochastické, což zvyšuje obtížnost agenta zvládnout nejistotu spojenou se všemi nepozorovatelnými stavy. [4]

Epizodické a sekvenční prostředí

V epizodickém prostředí jsou zkušenosti agenta rozděleny do epizod. Každá z epizod obsahuje dva moduly, z nichž první je vnímání a druhý je provedení jedné akce agentem. Hlavní charakteristikou epizodického prostředí je, že všechny akce agenta, které provedl v předchozích epizodách, nijak neovlivňují volbu akce v epizodách dalších [4]. Agent se v určité epizodě rozhoduje pouze na základě vjemu získaném v té samé epizodě.

V případě sekvenčního prostředí předchozí akce agenta ovlivňují jeho budoucí rozhodování. [4]

Oddělené a nepřetržité prostředí

Tento typ prostředí je založen na oddělené/nepřetržité povaze stavu prostředí, způsobu ovládání času a povaze vjemů a akcí agenta [4].

Jedno-agentové a více agentové prostředí

V tomto případě prostředí, se bere v potaz, zda se agent nachází v prostředí sám, nebo jestli je v prostředí více interagujících agentů. Na základě kolaborační strategie mezi agenty se tento typ prostředí dá rozdělit na kompetitivní, kooperativní nebo částečně kooperativní [4]. V kompetitivním prostředí agenti mezi sebou soupeří, naopak v kooperativním se snaží dosáhnout společného cíle.

Agentové modelování a komplexnost

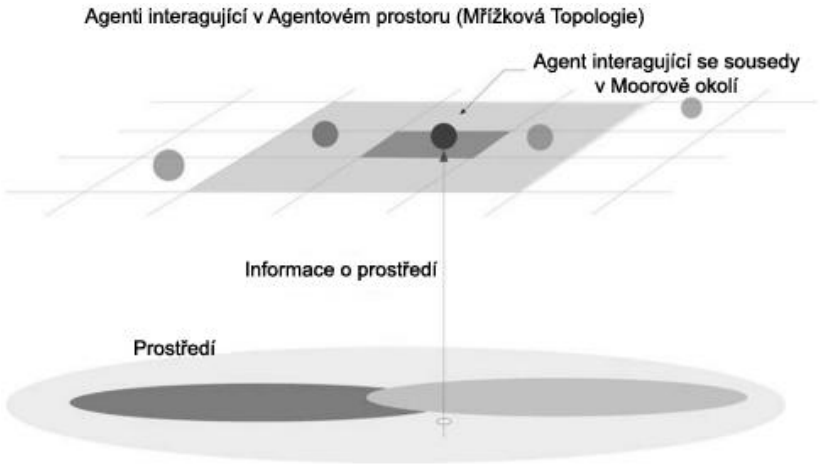
Komplexní systémy se skládají z interagujících, autonomních komponent. Komplexní adaptivní systémy dávají navíc agentům schopnost se přizpůsobit na individuální nebo populační úrovni.

Struktura agentového modelu

Typický agentový model obsahuje 3 elementy.

1. sadu agentů s řadou atributů a chování
2. sadu vztahů a interakcí mezi agenty (topologie propojenosti definuje jak a s kým agenti interagují)
3. prostředí agentů (agenti interagují se svým okolím i s ostatními agenty)

Při modelování je nutno tyto tři elementy identifikovat, namodelovat a naprogramovat pro vytvoření typického agentového modelu z obrázku 7.



Obrázek 7 Struktura typického agentového modelu, přeloženo z [6]

2. 2. Simulace dopravního provozu ve městě

Simulace dopravního provozu je velice důležitý krok při rozvoji infrastruktury. Kvůli růstu populace a zvyšujícímu se počtu automobilů, je nutné se umět adaptovat na takovéto situace rozšiřováním infrastruktury, která však musí být efektivně navržena.

Jelikož odhadovat, jakým způsobem rozšířit je buďto velice složité nebo přímo nereálné, je k řešení tohoto problému využito právě simulace dopravního provozu. Simulováním

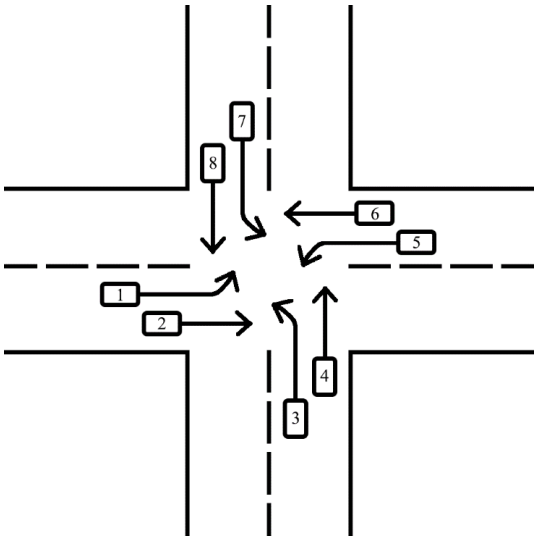
reálných situací je možné zjistit, jakým nejefektivnějším způsobem expandovat nebo upravit infrastrukturu, s cílem zlepšit efektivitu dopravy.

V případě dopravního provozu ve městě je často využíváno semaforů, jelikož z důvodu zvýšené frekventovanosti silnic není možné dopravu bezpečně uřídit pouze značkami.

Řada křižovatek využívá statických intervalů na semaforu, kde jsou přímo v systému nastaveny délky intervalů. Délky těchto intervalů jsou založeny na předchozích pozorováních. Frekventovanější silnice mají vždy delší dobu otevření než ty méně frekventované. Toto řešení však není efektivní, jelikož nebere v potaz možné změny v množství dopravy.

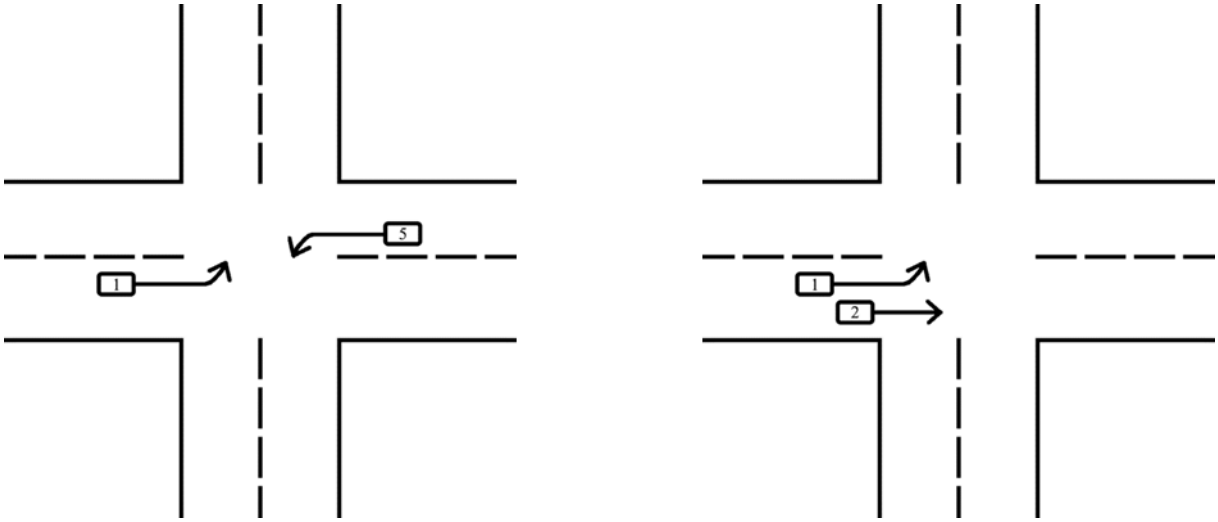
Vstupní podmínky křižovatky

Na křižovatce je nutno stanovit, jakými směry se řidiči mohou pohybovat zároveň. Všechny směry, kterými se mohou pohybovat vypadají takto:

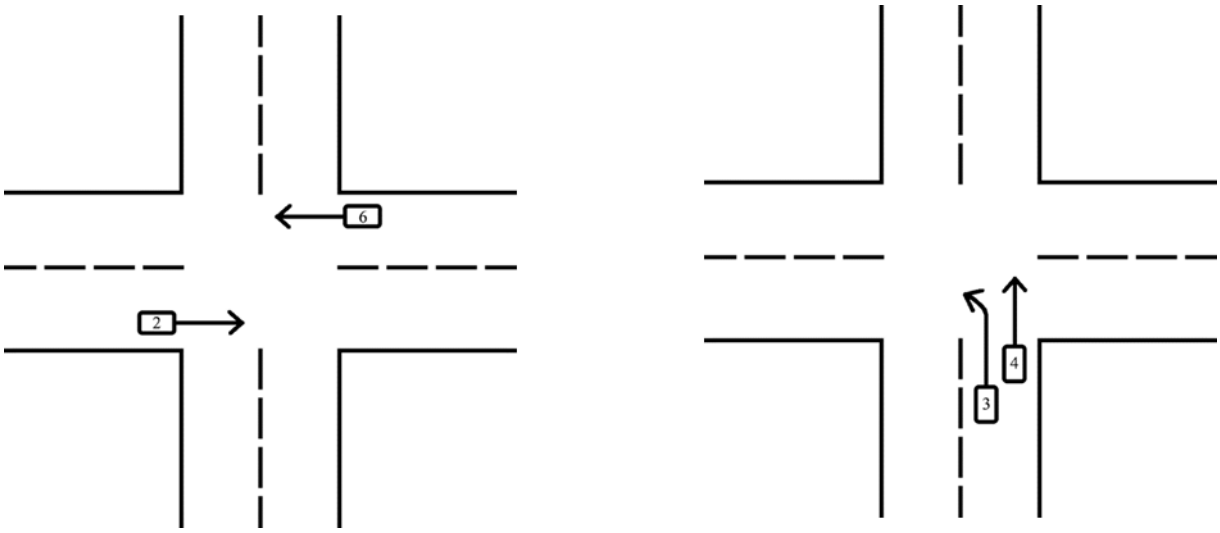


Obrázek 8 Směry pohybů na čtyřstranné křižovatce, předěláno z [7]

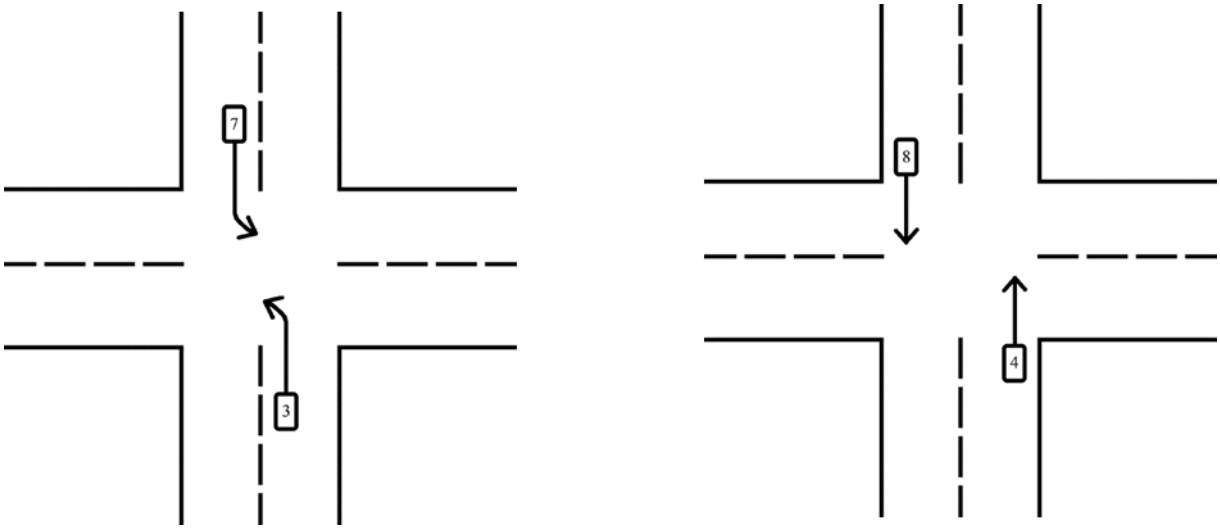
V případě, že řidič jede křižovatkou rovně, může zároveň odbočovat doprava, jelikož tím nenaruší cestu nikoho jiného, který v tento moment může také projíždět. Toto však platí pouze v případě řešení semaforovým řízením, nikoliv na běžných křižovatkách.



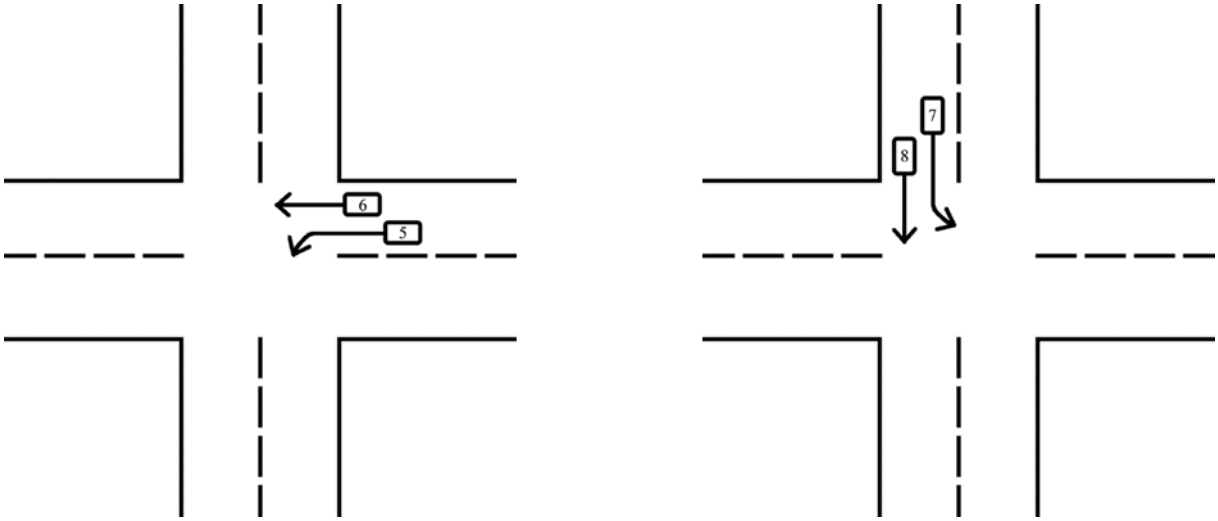
Obrázek 9 Povolené směry <1,5> a <1,2>, předěláno z [7]



Obrázek 10 Povolené směry <2,6> a <3,4>, předěláno z [7]



Obrázek 11 Povolené směry <3,7> a <4,8>, předěláno z [7]



Obrázek 12 Povolené směry <5,6> a <7,8>, předěláno z [7]

I když existuje řada směrů, které se mohou pohybovat synchronně, často nejsou vhodným řešením pro semaforovou dopravu. Semafory využívají čtyř zelených fází, které jsou proloženy mezi-fázemi (oranžovými), které se snaží řidiče upozornit na nastávající změnu stavů.

Tyto fáze jsou:

1. Rovná fáze. Dvě protilehlé cesty jsou otevřené a povolují řidičům jet rovně nebo zabočit doprava.
2. Fáze odbočení doleva. Dvě protilehlé cesty povolují řidičům odbočení doleva (tato fáze existuje pouze v případě existence levého odbočovacího pruhu).

3. Druhá rovná fáze. Opět otevřené protilehlé cesty ortogonální k první fázi povolující řidičům projetí rovně nebo odbočení doprava.
4. Druhá fáze odbočení doleva. Povolené odbočení doleva na cestách ortogonálních k cestám ve fázi 2.

Ovládání dopravního signálu

Ovládání dopravního signálu je jedna z nejrozšířenějších metod řízení semaforové dopravy. Funguje na principu detekce automobilů na semaforu. Toho je většinou docíleno senzorem umístěným blízko semaforu [8]. Senzory mohou být buďto kamery nebo radary, ale většinou se využívá induktivních cívek umístěných uvnitř silnice. Tyto senzory následně posílají informace řídícímu počítači, který je umístěn blízko křižovatky. Tento počítač řídí délky intervalů reakcí na výstupy ze senzorů.

Částečné ovládaní dopravního signálu

V částečném ovládání dopravního signálu jsou silnice na křižovatce definovány jako hlavní a vedlejší. V tomto řešení je hlavní silnice vždy zelená, a mění svůj stav pouze v případě, že byl detekován automobil na jedné z vedlejších cest [8]. Toto řešení je vhodné na hlavních silnicích městem, kde je velice dobře předvídatelné, jaká silnice bude frekventovanější.

Plné ovládání dopravního signálu

V plném ovládání dopravního signálu jsou naopak od částečného řešení pozorovány všechny silnice křižovatky a intervaly jsou založeny na přítomnosti automobilů na těchto silnicích [8]. V tomto řešení není žádná ze silnic považována jako hlavní ani vedlejší.

2. 3. Repast Simphony

Repast Simphony je volně šiřitelná (open source) sada nástrojů pro tvorbu agentových modelů. Repast (REcursive Porous Agent Simulation Toolkit) byl původně vyvinut v roce 2000 na Univerzitě Chicago a následně rozvinut Argonne Národní Laboratoří na znovupoužitelnou softwarovou infrastrukturu [5].

Další vydání Repastu pouze přidala na funkcionalitě a škále této sady nástrojů pro tvorbu agentových modelů.

Repast je spravován neziskovou organizací ROAD (Repast Organization for Architecture and Design), která je vedena správní radou obsahující členy z odvětví vlády, akademie a industrializačních organizací [5]. Zároveň díky tomu, že je Repast otevřený software, řada dalších jednotlivých skupin vývojářů přispěla k vývoji Repastu.

Zdrojový kód je dostupný přímo z oficiálních webových stránek:

http://repast.sourceforge.net/

Repast Simphony využívá vývojového prostředí Eclipse, a nabízí velkou řadu knihoven pro agentové modelování.

Implementační jazyk

Repast Simphony nabízí řadu možných implementačních jazyků. Tvorba modelů je možná v jazyce Logo, statecharts, Groovy nebo objektově orientovaném programování v Javě. Také je možné provázaní všech typů v jednom modelu.

ReLogo

Implementace modelů v Repast Simphony je možná v jazyce ReLogo, který je založený na jazyce Logo. Využívá prvků želvy (turtles), které jsou pohybující se agenti v modelu a záplaty (patches), což jsou stacionární agenti, kteří jsou přiřazení na jeden bod v prostředí [9].

Implementace těchto prvků je v Repast Simphony dosažena jejich vložením do kontextu a projekcí. Přidávání agentů do kontextu a projekcí se využívá i v řešení objektově orientovaným programováním.

Objektově orientované programování

Repast Simphony podporuje tvorbu agentových modelů za pomocí objektově orientovaného programování. Tvorba modelu objektovým programováním je velice intuitivní. Agentovy vnitřní stavy (věk, rychlost, hlad atd.) jsou reprezentovány proměnnými nebo za pomocí polí [9]. Agentovo chování (stárnutí, zrychlování, jezení atd.) je reprezentováno metodami daného objektu [9]. Repast Simphony umožňuje implementaci modelů za pomocí Javy, což znamená, že agenti jsou reprezentováni Java třídami. Právě implementací modelu za pomocí tohoto řešení se zabývá praktická část této práce.

Pojetí času

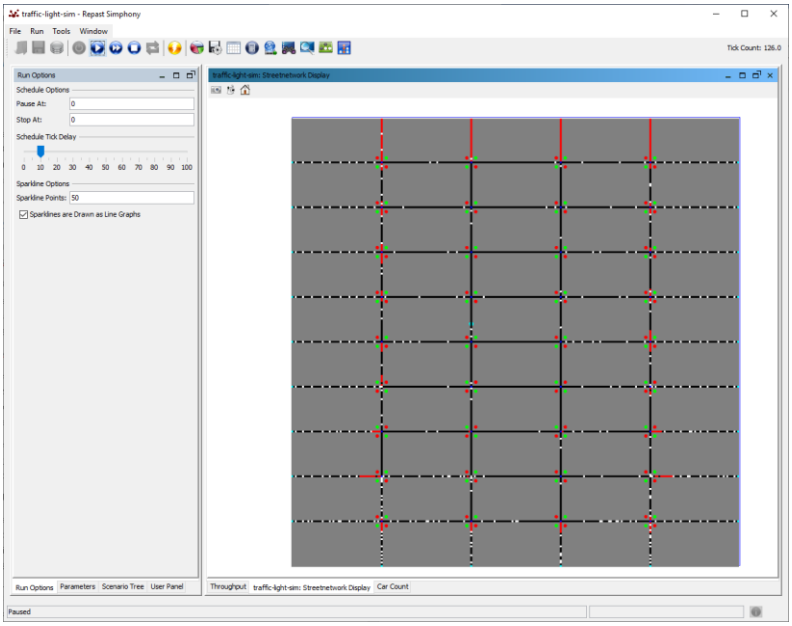
Repast Simphony využívá krokový model běhu simulace, kde pravidla chování jsou prováděna každý krok simulace. Tato rozhodnutí však nejsou prováděna zároveň, avšak v sekvenci za sebou. Na toto je nutné si při modelování v Repast Simphony dávat pozor, jelikož občas je nutné provést jednu akci před druhou pro správný průchod simulace.

Grafické rozhraní

Repast Simphony nabízí vlastní grafické simulační rozhraní. V rozhraní lze nastavit jaké projekce se v něm mají zobrazovat, kteří agenti se mají v těchto projekcích zobrazit a také vzhled těchto agentů. Následně je možné nastavit vrstvení agentů, které umožňuje přímo nastavit jací agenti se mají zobrazovat nad jakými, a naopak kteří agenti mají být překryti ostatními, v případě že se nachází na stejném poli. Dále umožňuje tvorbu grafů z výstupů simulací za pomocí vestavěných funkcí.

Příklad dopravního modelu v Repast Simphony

Jedním příkladem dopravního modelu, vytvořeným v Repast Simphony je model Semaforová simulace od Alexandra Wöstmanna. Tento model simuluje semaforově řízenou dopravu na mřížkově rozložených silnicích. Automobily zčervenají, v případě že čekají (za automobilem nebo na semaforu) a zbělají, v případě že jsou v pohybu. Barvy na všech semaforech se mění stejně a staticky (jsou pevně nastaveny intervaly).



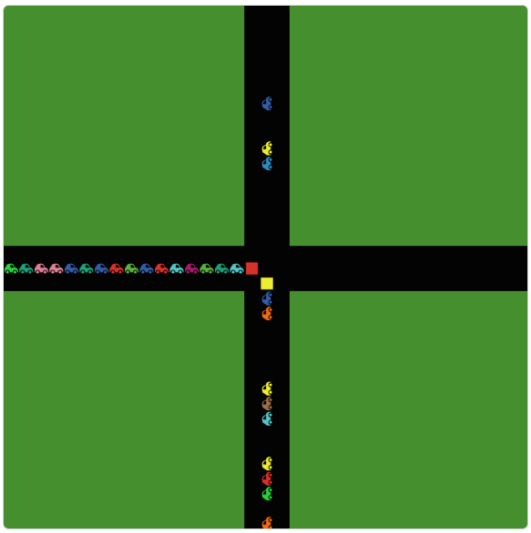
Obrázek 13 Model semaforově řízené dopravy [15]

NetLogo

NetLogo je multi-agentní programovatelné modelovací prostředí vhodné pro modelování přírodních a společenských jevů. NetLogo bylo vytvořeno v roce 1999 Uri Wilenskym a dále vyvíjeno v Centru propojeného učení a počítačového modelování. NetLogo je vytvořeno v Javě, avšak pro tvorbu modelů je využito multi-agentních modelovacích jazyků StarLogo nebo StarLogoT.

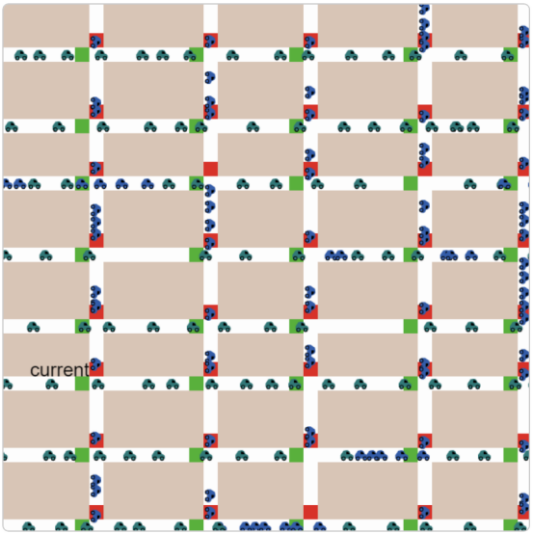
Příklady dopravních modelů v NetLogo

Jedním z příkladů nabízených u modelovacího programu NetLogo je Dopravní Křižovatka. V modelu je simulována jednosměrná křižovatka, kde se agenti pohybují pouze jedním směrem, a v případě „červené“ čekají na křižovatce. Zároveň na sebe reagují a čekají v případě detekce dalšího automobilu před nimi.



Obrázek 14 Netlogo – Traffic Intersection model [12]

Dalším modelem založeným na dopravě je Dopravní Mřížka. Je to simulace velice podobná předchozímu příkladu Dopravní Křižovatka, tento model je však rozšířen o velké množství křižovatek, jeho princip je ale velice podobný.



Obrázek 15 NetLogo – Traffic Grid model [13]

Tvorba modelů v prostředí NetLogo je efektivní a uživatelsky přívětivá, právě díky využití lehce pochopitelných modelovacích jazyků založených na jazyce Logo.

PTV Vissim

PTV Vissim je mikroskopický multimodální program pro tvorbu simulací dopravních toků. Je vyvíjen firmou PTV Planung Transport Verkehr AG v Karlsruhe, Německo. PTV Vissim byl vyvinut v roce 1992 a je momentálním lídrem v simulaci dopravních situací.

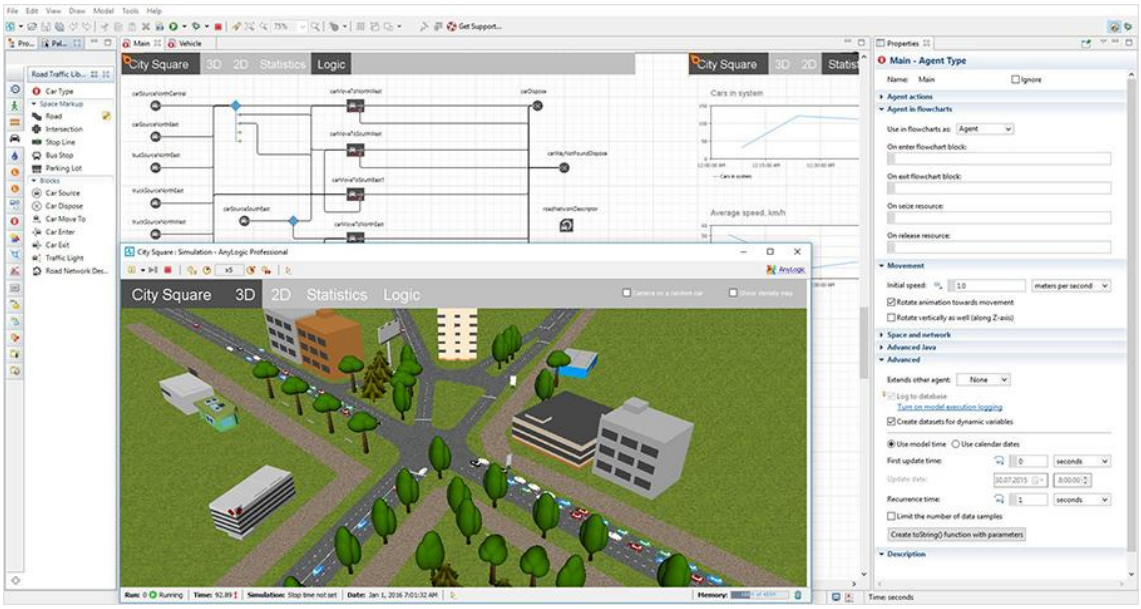
Rozsah použití programu se pohybuje od dopravního inženýrství, veřejné dopravy, územního plánování nad požární ochranou (simulace evakuací) po 3D vizualizaci.

Základní dopravní model určující pohyb vozidel byl vytvořen panem Rainerem Wiedemannem v roce 1972 na Karlsruhe Univerzitě. Je to model následování automobilů, který bere v potaz fyzické a psychologické aspekty řidičů. Také využívá model řešící dynamiku chodců Model Soiální Síly, který vyvinul Dirk Helbing v roce 1995.

Obrázek 16 Modelovací prostředí PTV Vissim

Anylogic

Anylogic je sada nástrojů pro modelování a simulaci velké řady různých odvětví, obsahující simulování řetězců dodavatelů, výrobu, přepravu, provozování skladů, železniční logistiku, těžbu po simulace portů a terminálů.



Obrázek 17 Modelovací prostředí Anylogic

Již v základu obsahuje velkou řadu knihoven pro tvorbu velice komplexních modelů. Obsahuje například knihovnu pěšců a knihovnu semaforů, pomocí které lze namodelovat realistické dopravní situace.

3. Praktická část

V předchozí části byly vysvětleny principy agentového modelování, typy agentů a prostředí, ve kterých se pohybuje. Dále byl popsán modelovací jazyk Repast Simphony, ve kterém je řešen následující model.

3. 1. Specifikace modelu

Specifikace modelu je sepsána pomocí ODD protokolu, který slouží k velice efektivnímu popsání agentových modelů a všech jeho funkcí, stavových proměnných a pravidel chování.

Zadání

Vytvoření modelu dvouproudové křižovatky se semaforovým řízením a následné aplikování různých způsobů ovládání semaforů.

Provedení experimentů na ovládání semaforů a zjištění nejefektivnějších řešení pro namodelované situace.

ODD protokol

1. Účel

Účel modelu je měření efektivity způsobů řízení semaforové dopravy na křižovatce implementací statického, částečného a plného ovládání signálů s autonomně se chovajícími automobilovými agenty.

2. Entity, stavové proměnné a měřítka

Automobil

Název	Typ	Popis
currentPosition	GridPoint	Aktuální souřadnice automobilu
nextPosition	GridPoint	Souřadnice, na které se chce agent pohnout
initialDir	GridPoint	Úvodní směr agenta
instructions	List<GridPoint>	Sada instrukcí pro pohyb agenta

Silnice

Název	Typ	Popis
currentPosition	GridPoint	Aktuální lokace silnice

Semaforový řidič

Název	Typ	Popis
sensorLocations	List<GridPoint>	Seznam souřadnic senzorů
tlLocations	List<GridPoint>	Seznam souřadnic semaforů
minMainInterval	Integer	Minimální doba otevření hlavní silnice
maxMinorInterval	Integer	Maximální doba otevření vedlejší silnice

Semaforový senzor

Název	Typ	Popis
distance	Integer	Vzdálenost senzorů od křižovatky
carAmount	Integer	Počet detekovaných automobilů

Semafor

Název	Typ	Popis
position	GridPoint	Souřadnice semaforu
color	Stav	Aktuální barva na semaforu

Startovní bod/Koncový bod

Název	Typ	Popis
position	GridPoint	Souřadnice startovního/koncového bodu

3. Přehled procesů a plánování

Automobil – ‘najdiUvodniSmer’, ‘zabocDoleva’, ‘jeAuto’, ‘jeKrizovatka’, ‘vyberSmer’, ‘jeZelena’, ‘jed’, ‘jeKonec’

Semaforový řidič – ‘generujSenzory’, ‘generujSemafory’, ‘inicializuj’, ‘zmenStav’, ‘pouzijStatickeOvladani’, ‘pouzijCastecneOvladani’, ‘pouzijPlneOvladani’

Semaforový senzor – ‘jeAuto’, ‘prictiAuto’, ‘vyresetujPocetAut’

Semafor – ‘zelena’, ‘oranzova’, ‘cervena’

4. Designové koncepty

Základní principy – základní princip modelu je představit řadu způsobů řešení semaforové dopravy implementací různých algoritmů pro tento cíl a následné testování efektivity těchto algoritmů

Vznik – díky schopnosti automobilových agentů si vybírat náhodně směry jízdy, je zajištěno testování algoritmů pro řízení na nepředpokládatelných situacích.

Přizpůsobování – semaforový systém disponuje schopností přizpůsobit se na dané množství automobilů změnou délky intervalů, čímž se snaží co nejefektivněji řídit nastalou dopravní situaci.

Cíle – cíl automobilového agenta je projet křižovatkou směrem, který si náhodně zvolí. Cíl agenta semaforového řízení je co nejefektivněji propustit automobilové agenty touto křižovatkou.

Učení – agenti se neučí, semaforový systém vždy reaguje na nastalou situaci bez toho, aniž by bral v potaz předchozí rozhodnutí.

Předpověď – automobilový agent vždy předpovídá, že je možnost existence řady entit na polích, na která se chce posunout, a je připraven na ně reagovat.

Snímání – každý automobilový agent snímá své okolí z důvodu prevence kolize s jinými automobily. Dále sleduje, zda se nenachází na křižovatce, a v případě že jí chce projet, také stav semaforu na silnici, na které se právě nachází. Semaforový řidič za pomocí senzorů neustále sleduje přítomnost automobilů na silnicích.

Interakce – interakce mezi automobilovými agenty jsou nepřímé. Automobiloví agenti spolu nikdy přímo nekomunikují, avšak reagují na sebe. Semaforový řidič interaguje s automobilovými agenty za pomocí senzorů, čímž je mu umožněna úprava intervalů dle nastalé situace.

Stochasticita – automobiloví agenti si náhodně vybírají, jakým směrem chtějí projet křižovatkou.

Kolektivy – existuje semaforový kolektiv. Je to kolektiv agentů, kteří spolu spolupracují s cílem měnit barvy na semaforech v reakcí na detekci automobilů.

Pozorování – na modelu je pozorována efektivita řady způsobů pro řízení semaforové dopravy. Výsledky z těchto pozorování jsou – průměrná doba čekání automobilů na semaforu, maximální doba čekání na semaforu, propustnost křižovatky, schopnost adaptace na nastalou situaci.

5. Inicializace

V počátku simulace existuje křižovatka sestavena ze silnicových entit. Na této křižovatce se nachází sada semaforů, které mají nastavený první stav (otevření některého směru). Na počátku každé silnice existuje jeden startovní bod a dva koncové body. Každá příjezdová cesta ke křižovatce obsahuje semaforové senzory.

6. Vstupní data

Model nepoužívá vstupní data k reprezentaci časově proměnlivých procesů.

7. Pod-modely

Automobil

najdiUvodniSmer – po vzniknutí automobilu a přemístění do startovního bodu, si automobil nalezne nejbližší silnici, na kterou se může posunout, a nastaví ji jako svůj úvodní směr.

jetDoleva – agent se po nalezení úvodního směru náhodně rozhodne, zda chce pokračovat rovně, nebo zabočit do levého odbočovacího pruhu. V případě že chce pokračovat rovně, drží úvodní směr, dokud se nespustí jedno z jeho pravidel chování. V případě že chce odbočovat doleva, nastaví si sadu instrukcí na odbočení do levého odbočovacího pruhu, a po jejich vykonání pokračuje směrem ke křižovatce do doby, než se spustí jedno z jeho pravidel chování.

jeAuto – agent neustále kontroluje, jestli místo, na které se chce posunout, již neobsahuje jiného automobilového agenta. V případě že ano, čeká agent do doby, než se pole uvolní.

jeKrizovatka – na každý bod, na který se agent chce posunout je před pohybem volána funkce, která zjistí, zda se na tomto bodě nachází křižovatka. V případě že ano, je zavolána funkce jeZelena a vyberSmer.

vyberSmer – v případě, že se funkce jeKrizovatka vrátila jako pravdivá, rozhodne se automobil, jakým směrem chce projet křižovatkou. V případě, že se automobil ve funkci jetDoleva rozhodl jet doleva, nastaví si sadu instrukcí pro průjezd křižovatkou doleva. V případě, že se ve funkci jetDoleva rozhodl pokračovat rovně, vybere agent náhodně jestli chce na křižovatce dále pokračovat rovně, nebo jestli chce zabočit doprava, a na základě tohoto rozhodnutí si nastaví buďto instrukce na projetí rovně nebo instrukce na projetí doprava.

jeZelena – v případě že automobil detekoval křižovatku na bodě, na který se chce posunout, zjistí, zda na bodě, na kterém se právě nachází existuje semafor zelené barvy. Pokud je barva červená nebo oranžová, automobil čeká do doby, než se změní na zelenou. V případě že nastane zelená, má povoleno se automobil posunout dál.

jed – v případě že funkce jeAuto, jeKrizovatka a jeZelena vrátily všechny nepravdu, a sada instrukcí neobsahuje žádné pokyny pro projetí křižovatkou, jede automobil směrem, který má právě nastavený. Jestliže na základě detekce křižovatky vznikly instrukce pro projetí křižovatkou, automobilový agent čeká, dokud není jeZelena pravdivá. V případě že má zelenou projíždí křižovatkou podle instrukcí a po vyjetí z křižovatky pokračuje směrem ke koncovému bodu.

jeKonec – v případě, že agent detekuje koncový bod na bodě, na kterém se nachází, je odstraněn z modelu.

Semaforový řidič

generujSenzory – semaforový řidič vygeneruje semaforové senzory na příjezdových silnicích ve vzdálenosti získané ze vstupních parametrů modelu při spuštění simulace.

generujSemafory – semaforový řidič vygeneruje semafory na příjezdových silnicích ke křižovatce při spuštění simulace.

inicializuj – semaforový řidič inicializuje úvodní stav (nastavením jednoho ze stavů) pro semafory při spuštění simulace.

pouzijStatickeOvladani – nastaví řízení semaforů pevně stanovenými délkami intervalů, které jsou definovány v parametrech modelu

pouzijCastecneOvladani – nastaví řízení semaforu za pomocí částečného ovládání

pouzijPlneOvladani – nastaví řízení semaforu za pomocí plného ovládání

zmenStav – provede změnu stavu na semaforech podle podmínek aktuálně využívaného ovladače dopravy.

Semaforový senzor

jeAutomobil – funkce využívána funkcí prictiAutomobil pro detekci projíždějících automobilů

prictiAutomobil – v případě že senzor detekuje projíždějící automobil na stejném bodě kde se on nachází, přičte ke své vnitřní proměnné „pocetAut“ plus jedna

vyresetujPocetAut – funkce volána semaforovým řidičem po využití informace ze senzoru. Vynuluje hodnotu na daném senzoru

Semafor

Oranzova – funkce využívána semaforovým řidičem pro změnu barvy semaforu na oranžovou

Cervena – funkce využívána semaforovým řidičem pro změnu barvy semaforu na červenou

Zelena – funkce využívána semaforovým řidičem pro změnu barvy semaforu na zelenou

3. 2. Implementace modelu

Implementace modelu je řešena za pomocí objektově orientovaného programování v Javě v Repast Simphony.

Tvorba kontextu

Kontext nastaví počáteční stav simulace a je nutné přidávat entity do něho, pokud se mají zobrazit v simulaci. Dalo by se říct, že kontext je v Repast Simphony kontejner, který

zabalí všechny agenty modelu. Tvorba kontextu je v Repast Simphony dělána za pomoci vytvoření Java třídy, která implementuje ContextBuilder.

```
public Context build(Context<Object> context)
```

Výpis 1 Deklarace kontextové třídy

V případě implementace projekcí nebo sítí je nutno je nejdříve za-definovat v „context.xml“, zde je nutné jim nastavit typ a id, na které je poté odkazováno v kódu.

```
1  <projection type="continuous space" id="space"/>
2  <projection type="grid" id="grid"/>
3  <projection type="network" id="traffic network"/>
```

Výpis 2 Nastavení projekcí v context.xml

Po deklaraci v „context.xml“ je možné vytvořit projekce ve třídě, která implementuje ContextBuilder.

```
1  ContinuousSpaceFactory spaceFactory =
   ContinuousSpaceFactoryFinder.createContinuousSpaceFactory(null);

2  ContinuousSpace<Object> space =
   spaceFactory.createContinuousSpace("space", context, new
   SimpleCartesianAdder<Object>(), new
   repast.simphony.space.continuous.StrictBorders(), sirka, vyska);

3  GridFactory gridFactory =
   GridFactoryFinder.createGridFactory(null);

4  Grid<Object> grid = gridFactory.createGrid("grid", context, new
   GridBuilderParameters<Object>(new WrapAroundBorders(), new
   SimpleGridAdder<Object>(), true, sirka , vyska));
```

Výpis 3 Vytvoření projekce ContinuousSpace a Grid

Výpis 3 vytvoří ContinuousSpace za pomocí ContinuousSpaceFactory a Grid za pomoci GridFactory. V ContinuousSpace se agenti mohou pohybovat volně, zatímco v Grid mají nastavenou mřížku, a mohou se pohybovat pouze mezi body této mřížky.

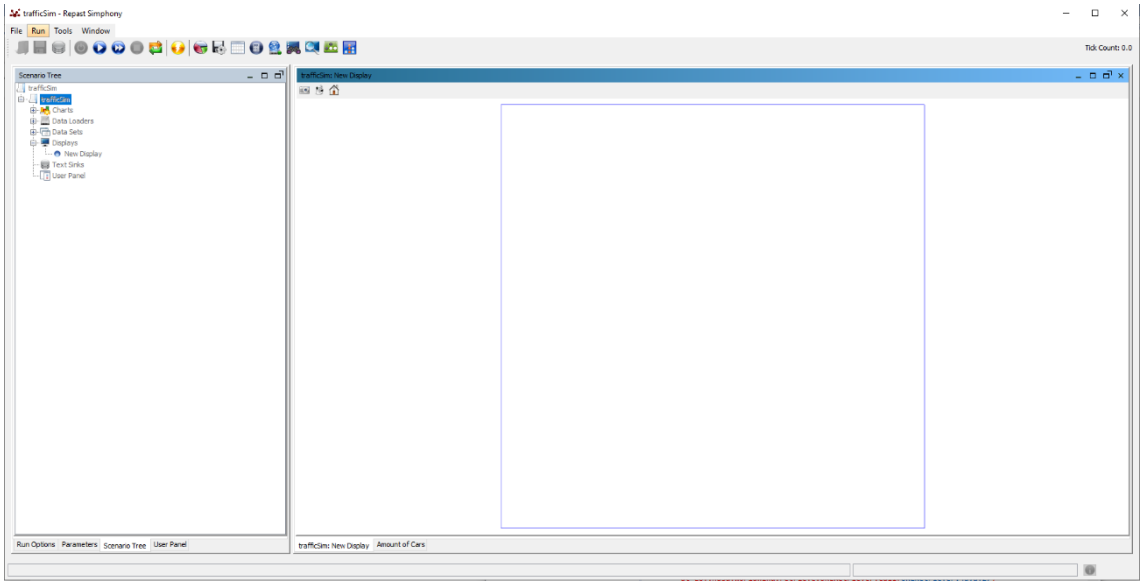
Obě funkce přijímají stejné parametry, a to název Gridu/ContinuousSpacu, kontext, a další parametry, které samotné projekce definují.

Prvním parametrem je typ okrajů, kde nezáleží na použitém typu, jelikož se v našem modelu k okrajům agenti nikdy nepřiblíží (v případě WrapAroundBorders se souřadnice agentů kteří opustili grid přepočítají, aby se objevili na druhé straně, v případě StrictBorders tomu tak není, a když opustí grid, spadne program do SpatialException).

Druhým parametrem je typ „přidávače“, pomocí kterého se do kontextu přidávají agenti. Jedním z „přidávačů“ může být náhodný přidávač. Ten agenty do prostoru rozmístí náhodně. Tento způsob přidávání agentů v našem případě však nevyhovuje, proto je lepší využít SimpleGridAdder/SimpleCartesianAdder, a po přidání s agenty pohnout manuálně na potřebné pozice.

Poslední dva parametry jsou celočíselné rozměry pro šířku a výšku mřížky/prostoru.

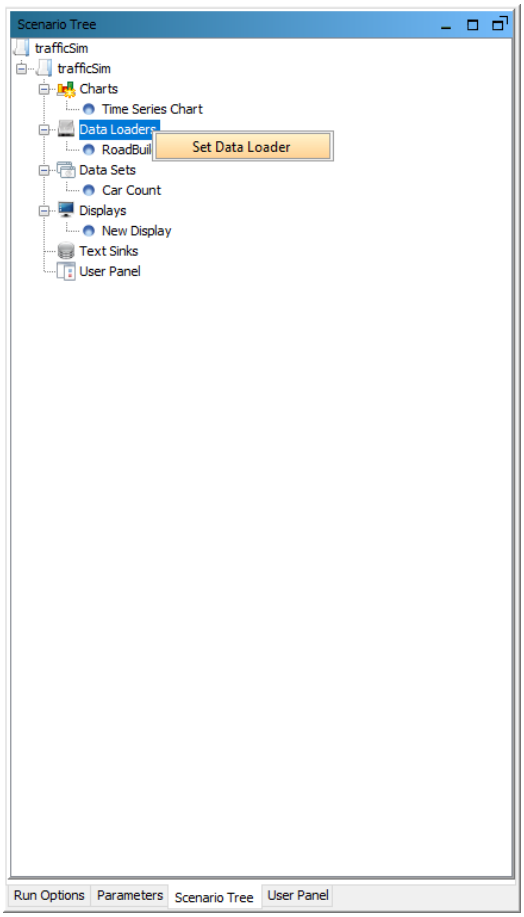
V tomto stavu model vypadá takto:



Obrázek 18 Model po vytvoření prostoru a mřížky

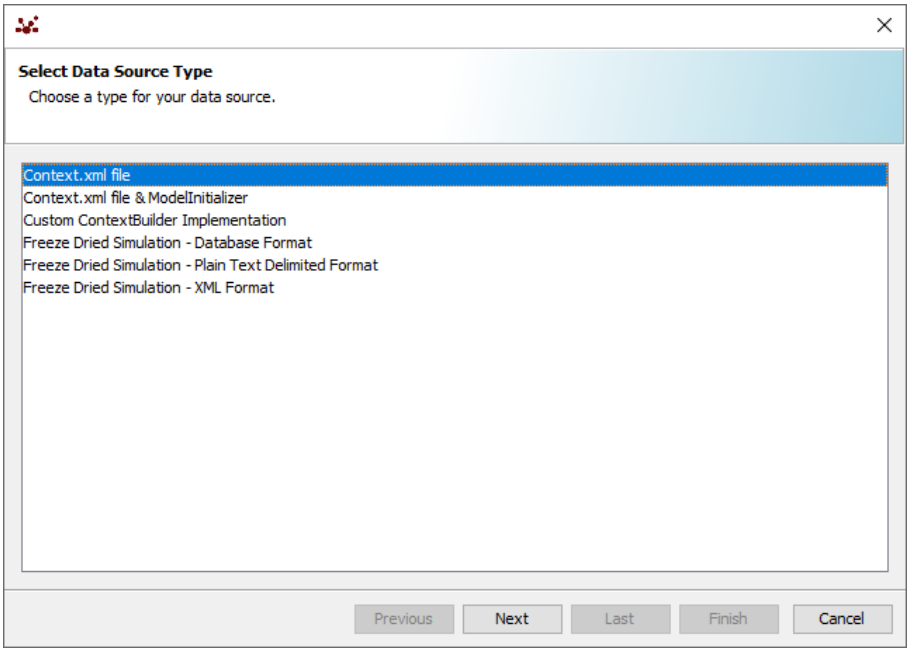
Přidání kontextu do modelu

Po vytvoření kódu pro tvorbu kontextu modelu a projekcí, je nutné ho přidat v grafickém rozhraní. To se provádí skrze zavaděč dat:



Obrázek 19 Přidání tvůrce kontextu v grafickém rozhraní

Po přidání zdroje dat je nutné nastavit jaký typ zdroje chceme využít.

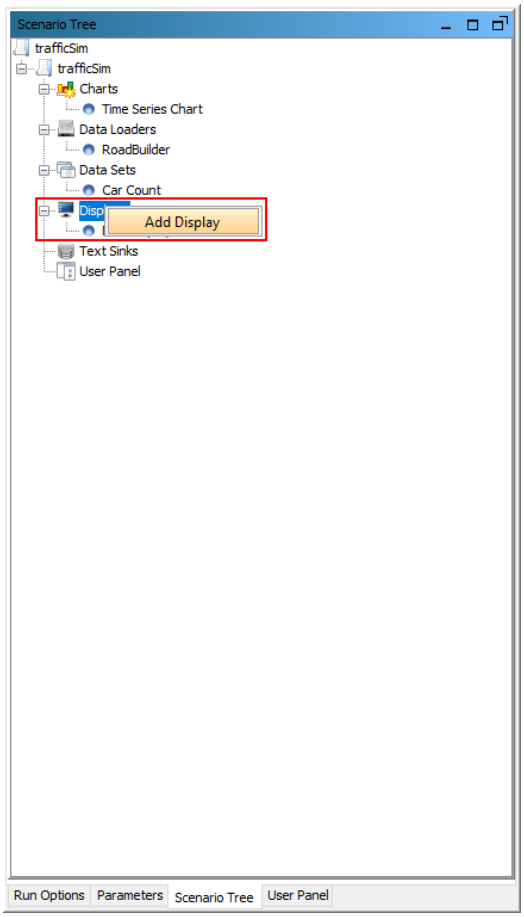


Obrázek 20 Výběr typu zdroje dat

V případě že chceme použít vlastní datový zdroj (třídu která implementuje ContextBuilder), je nutné vybrat možnost „Custom ContextBuilder Implementaion“ a vybrat třídu, která ContextBuilder implementuje.

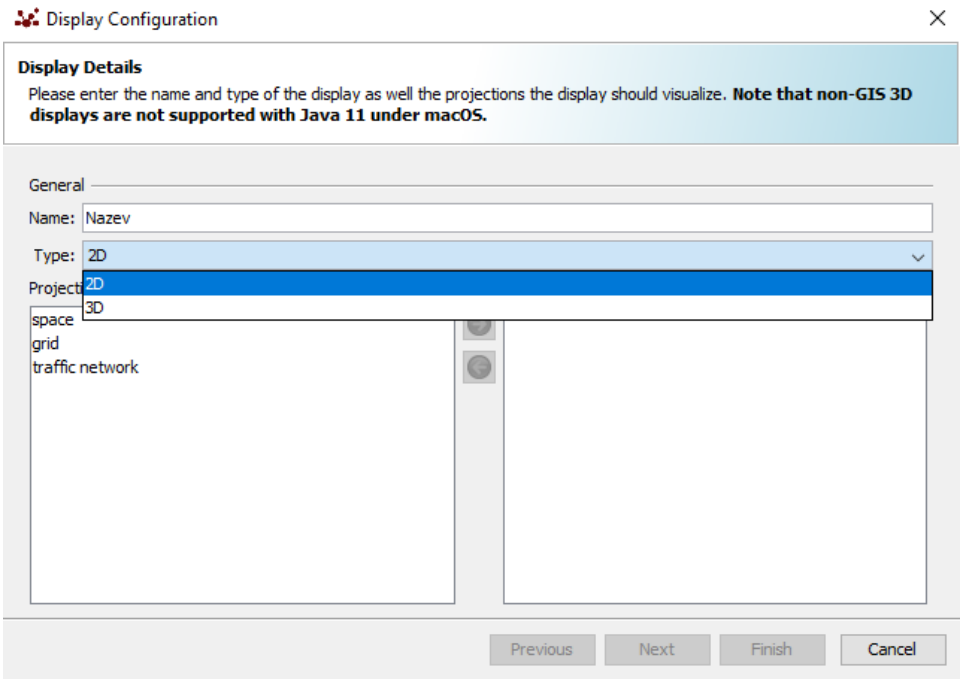
Přidávání agentů do modelu

I když jsou agenti přidání do kontextu, v simulaci samotné se zatím nijak nezobrazí. Je nutné je do simulace přidat v grafickém rozhraní v části Displeje.



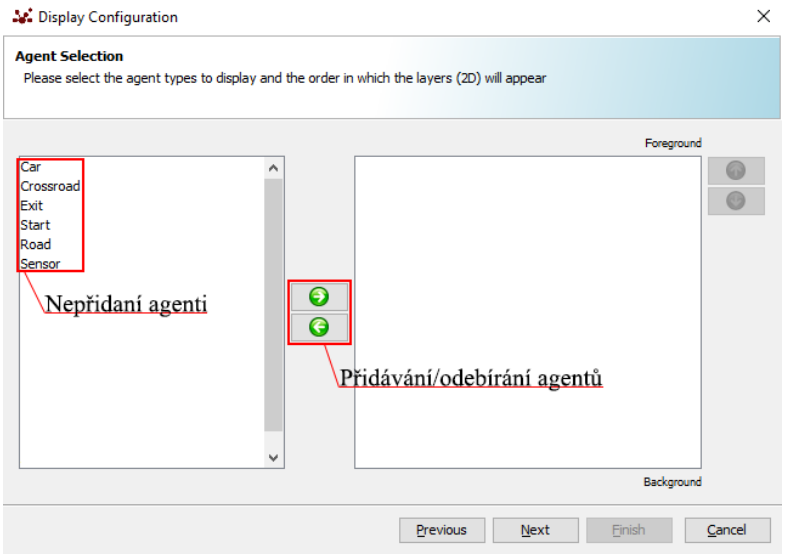
Obrázek 21 Vytvoření obrazu v grafickém rozhraní

Po přidání obrazu je nutné ho patřičně nastavit.



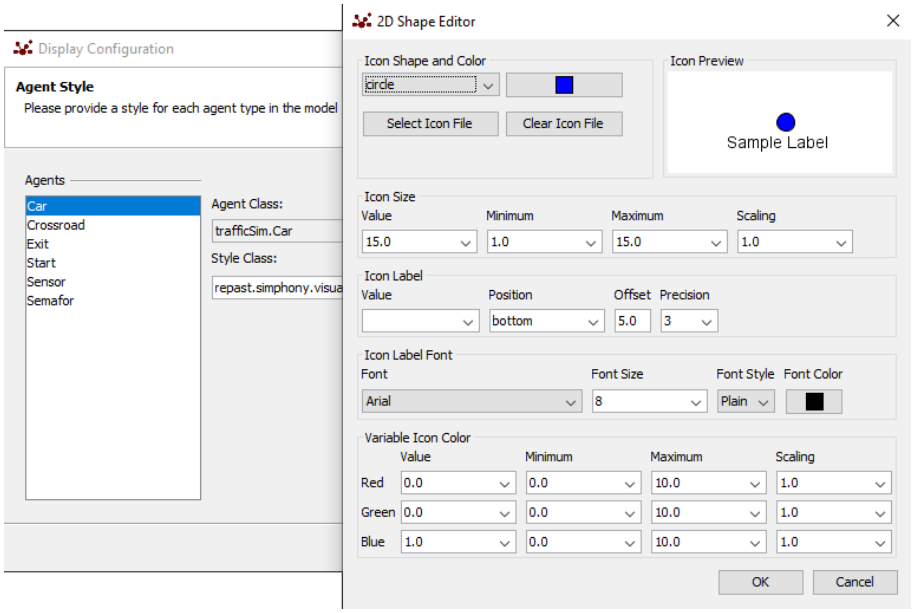
Obrázek 22 Úvodní nastavení obrazu

Na obrázku 22 lze vidět, že v první fázi nastavování obrazu je nastaven jeho název, typ zobrazení (2D/3D) a výběr projekce, kterou má obraz zobrazovat. Repast Simphony umožňuje zobrazování projekcí souvislého prostoru, mřížky, zeměpisu, sítí nebo fyzikálních prostor. Po úvodním nastavení se vybírají agenti, kteří se na obraze mají vizuálně zobrazovat.



Obrázek 23 Přidávání agentů do obrazu

Na levé straně obrázku 23 se nacházejí agenti, kteří nejsou ještě přidání. Pro přidání/odebrání agentů se využívají dvě zelené šipky uprostřed. Šipka doprava agenta přidá, šipka doleva ho naopak odstraní. Agenti, kteří jsou přidáni, se zobrazují v pravé části. Přidaní agenti využívají posloupnosti zobrazení v obraze, kdy agenti výše se budou zobrazovat v simulaci před agenty, kteří se nacházejí v seznamu pod nimi, tudíž když chceme, aby automobil byl viditelný, a nebyl schovaný pod silnicí, musí být v seznamu výše než silnice.



Obrázek 24 Nastavení vzhledu agentů

Jedním z posledních kroků přidávání agentů je nastavení jejich vzhledu. Repast Simphony sám nabízí řadu tvarů a možnost výběru barvy, nebo výběr vlastní ikony. Dále lze nastavit jeho velikost a textový popis agenta.

Dalším krokem je nastavení samotného vzhledu obrazu (nastavení velikosti obrazu, možnost ukázání ohraničujícího rámečku a nastavení jeho barvy).

Posledním krokem je pak nastavení plánování zobrazení. Zde se nastavuje, kdy má zobrazování začít (jaký krok simulace), priorita tohoto obrazu (první/poslední/náhodně), frekvence zobrazení (jednou/opakovat) a interval, po jakém se má zobrazení provést znovu.

Plánování v Repast Simphony

Je velice nutné nastavení plánování v Repast Simphony. Jelikož Repast funguje na krokové exekuci simulací, je nutné využití metod pro volání funkcí agentů v těchto krocích. Pro tento důvod existuje anotace ScheduledMethod. Jakákoliv funkce, která má nastavenou tuto anotaci, je stanovena jako funkce, která má být spouštěna plánovaně (v určitých krocích simulace). Je možné k ní nastavit řadu atributů. Hlavními jsou start (v jakém kroku simulace se má spustit poprvé) a interval (v jakých krokových intervalech se má funkce spouštět znovu).

Tvorba kontextu

Tvorbu kontextu lze chápat jako zabalování všech agentů modelu. V kontextu se nachází všichni agenti a projekce, které jsou v simulaci využity. Vstupní stav kontextu stanoví, jak simulace začne.

Tvorbou kontextu zároveň stanovíme prostředí, jelikož jsou do něho vloženi i agenti, kteří se nijak nepohybují (například silnice). Typ prostředí je v našem případě plně pozorovatelné, deterministické, sekvenční, nepřetržité, více agentové prostředí (Principy modelování a simulací – Prostředí agenta, str. 15).

Tvorba silnic

Jelikož se simulace odehrává na jedné křižovatce, je nutné vygenerovat sadu silnic v pruzích. Pro vygenerování dvouproudové silnice tedy stačí použít for cyklu, který vygeneruje 4 silnice nad sebou z jednoho konce mřížky na druhý.

Pro každý bod silnice je vytvořena nová entita Silnice (Road) a po přidání do kontextu je posunuta v Grid a ContinuousSpace na určenou pozici.

```
1  for(int i = start.getY(); i != konec.getY() + 1; i ++) {
2  Road road = new Road();
3  context.add(road);
4  grid.moveTo(road, X, i);
5  space.moveTo(road, X, i);
6  }
```

Výpis 4 Generace silnice za pomocí for cyklu [Vlastní zpracování]

Ve výpisu 4 je uveden pouze příklad generace jednoho pruhu na ose X. Bod „start“ reprezentuje startovní bod a bod „konec“ reprezentuje koncový bod pro generaci cesty.

Tato generace je provedena vždy čtyřikrát nad sebou na ose Y a poté na ose X, čímž vznikne dvouproudová obousměrná křižovatka ze silnicových entit.

Tvorba agentů

Typům agentů v Repast Simphony vždy odpovídá jedna Java třída. Instance této třídy pak mohou reprezentovat řadu agentů stejného typu. V této třídě mají agenti deklarovány všechny své funkce, chování, atributy, stavové proměnné atd. První sada agentů, které budeme vytvářet je semaforový systém.

Semaforový systém

Je kolektiv agentů, kteří mezi sebou interagují a spolupracují s cílem upravit intervaly na semaforech a zobrazit tyto informace automobilovému agentovi, který reakcemi na tyto vjemy upravuje své chování.

Obsahuje semaforový řidič, který řídí chod semaforů a mění barvy na semaforech, dále semafor, který reprezentuje otevřené směry (červená/oranžová/zelená) a semaforový senzor, který je používán pro detekci přibližujících se automobilů ke křižovatce.

Semaforový senzor

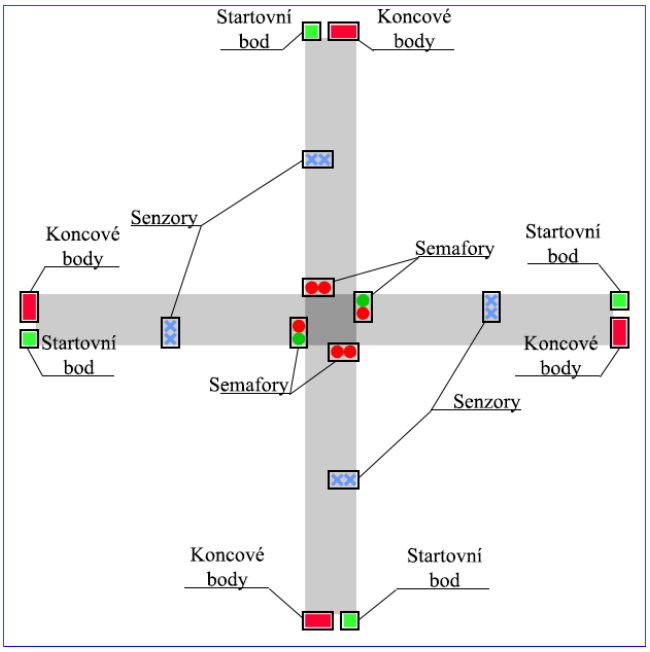
Semaforový senzor je jednoduchý reflexní agent (Principy modelování a simulací - Typy agentů, str. 12), umístěn určitou vzdálenost od semaforu (vzdálenost je definována parametrem, který je vkládán uživatelem při inicializaci modelu). Senzor slouží k detekování projíždějících automobilů. K detekci automobilů je využita funkce isCar (výpis 5).

```
1  private boolean isCar() {
2  Iterable<Object> objects = grid.getObjectsAt(currentPos.getX(),
   currentPos.getY());
3  for (Object object : objects) {
4  if(object.getClass() == Car.class) {
5  return true;
6  }
7  }
8  return false;
9  }
```

Výpis 5 Funkce detekce

Tato funkce získá všechny objekty v bodě „currentPos“ (což je aktuální bod senzoru) a uloží je do Iterable (seznam všech objektů), což nám následně umožní prohledat všechny objekty za pomocí „for each“ cyklu, čímž zjistíme, zda se na daném bodě nenachází automobil. V případě že ano, vrací funkce „pravda“ a ukončuje se prohledávání. V případě že automobil nedetekovala a dosáhla konce Iterable, vrací „nepravda“.

Pokaždé, když senzor detekuje automobil, zvýší svoji vnitřní proměnnou carAmount (množství detekovaných automobilů) o jedna. Vypočítaná hodnota je následně používána v semaforovém řidiči pro úpravu délky intervalů. V modelu je senzor reprezentován modrým křížem.



Obrázek 25 Vysvětlení reprezentace agentů/entit v modelu [Vlastní zpracování]

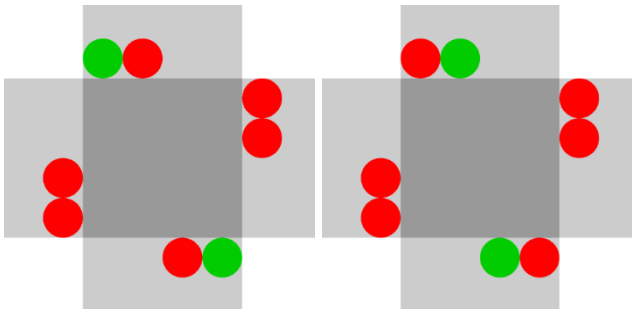
Semafor

Semafor samotný je jednoduchý reflexní agent, který mění otevřené směry dle informací od semaforového řidiče. Obsahuje vnitřní stavy (isGreen, isRed, isOrange), pomocí kterých automobily zjišťují, zda mohou projet křižovatkou. Zároveň jsou tyto stavy využity pro nastavení barvy semaforu za pomocí třídy implementující DefaultStyleOGL2D, to však nemá žádný dopad na simulaci a je využita ryze pro lepší sledovanost modelu.

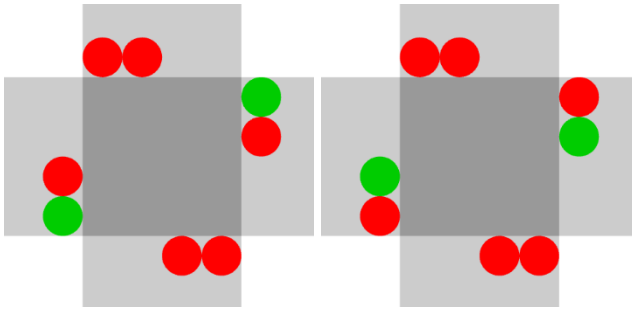
Semaforový řidič

Semaforový řidič je mozek semaforu. Je to modelově založený reflexní agent (Principy modelování a simulací – Typy agentů, str. 12), který má na starost změnu barev semaforů a změnu délky intervalů otevření.

Semaforový řidič má předdefinované čtyři stavy semaforů, mezi kterými prochází. Tyto stavy jsou:



Obrázek 26 Otevřené směry při stavu 1 a 2 [Vlastní zpracování]



Obrázek 27 Otevřené směry při stavu 3 a 4 [Vlastní zpracování]

Při každé změně z jednoho stavu na druhý nastává mezi-stav, který aktuálně zelené semafory změní na oranžové. Mezi-stav je držen tak dlouho, aby umožnil automobilům bezpečně opustit křižovatku.

```
1  public void setState1() {
2  state1 = true; state2 = false; state3 = false; state4 = false;
3  topLT.setRed();
4  topST.setGreen();
5  rightLT.setRed();
6  rightST.setRed();
7  bottomLT.setRed();
8  bottomST.setGreen();
9  leftLT.setRed();
10 leftST.setRed();
11 }
```

Výpis 6 Funkce pro změnu stavu [Vlastní zpracování]

Výpis 6 ukazuje princip změny barvy semaforu (v tomto příkladu na stav 1). Stav 1 odpovídá povolení průjezdu křižovatkou z vrchu rovně/doprava a zdola rovně/doprava (Obrázek 26). Funkce tedy nastavuje všechny semafory na červenou, až na topST (semafor pro vrchní stranu, pohyb rovně) a bottomST (semafor pro spodní stranu, pohyb rovně), které nastaví na zelenou.

Automobilový agent

Automobilový agent je jediným typem agenta, který se v modelu pohybuje. Je to modelově založený reflexní agent (Principy modelování a simulací – Typy agentů, str. 12). Agent bere v potaz jeho předchozí rozhodnutí, pro určení jeho dalších kroků. Například, když se v prvním kroku rozhodne, že bude odbočovat na křižovatce doleva, využije této informace pro nastavení správných instrukcí pro její projetí.

Pohyb

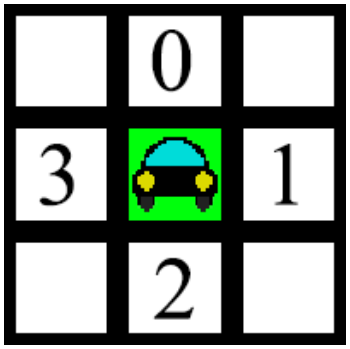
Automobilový agent má ve svých atributech pevně deklarované pole směrů pro pohyb. Tento seznam v kódu vypadá takto:

```
1  private static final GridPoint[] directions = {new GridPoint(0, 1),
   new GridPoint(1, 0), new GridPoint(0, -1), new GridPoint(-1, 0)};
```

Výpis 7 Deklarace instrukcí pohybu agenta [Vlastní zpracování]

Pole obsahuje čtyři GridPoint (body mřížky), které odpovídají čtyřem vektorům pohybu. První bod je pohyb nahoru, druhý bod je pohyb doprava, třetí bod je pohyb dolů, a čtvrtý bod je pohyb doleva.

Instrukce tedy korespondují těmto směrům (čísla odpovídají indexům v seznamu z výpisu 7):



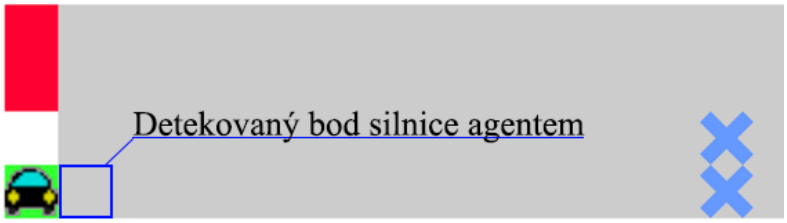
Obrázek 28 Směry ze seznamu instrukcí [Vlastní zpracování]

Následný pohyb je řešen přičtením směrového vektoru ke stávajícímu bodu agenta, čímž je vytvořen bod, na který se agent v dalším kroku bude pohybovat.

Samotný pohyb agenta, je poté volán za pomocí naplánované metody ScheduledMethod (Implementace modelu – Plánování v Repast Simphony, str. 41).

První pohyb

První krok, který automobilový agent po spuštění simulace udělá, je nalezení nejbližší silnice, na kterou se může posunout.



Obrázek 29 Jedna z úvodních pozic agenta po přidání do modelu [Vlastní zpracování]

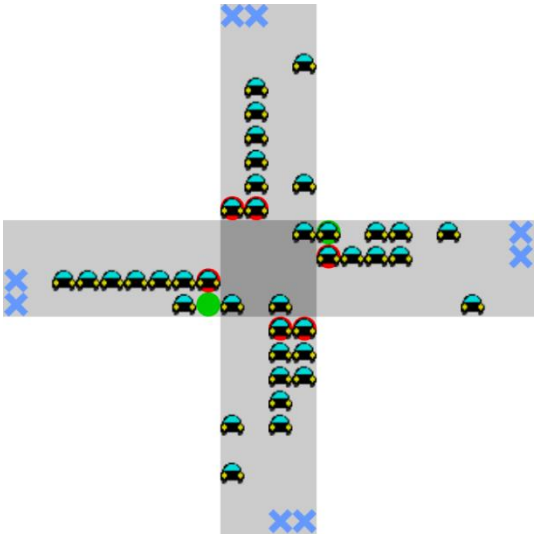
V případě uvedeném na obrázku 29 najde agent bod silnice, který má stejnou Y souřadnici jako on, zároveň je X souřadnice větší než jeho. Na základě těchto informací pak funkce úvodního pohybu vybere vektor pohybu na pozici jedna (vektor (1, 0) viz Automobilový agent – Pohyb, str. 45), přičte ho k aktuální pozici agenta, čímž vytvoří bod, na který se má agent pohnout. Zároveň si agent uloží tento směr do stavové proměnné.

Tímto směrem se agent pohybuje, dokud nenastane jedna z podmínek: detekuje další automobil/detekuje křižovatku/detekuje semafor/dojede do koncového bodu.

Detekce dalších automobilů

Jelikož se automobil musí zastavit a počkat na uvolnění pozice, na kterou směřuje v případě, že se na ní nachází jiný automobil, potřebuje funkci, pomocí které tuto situaci bude schopen detekovat.

Proto je vždy před pohybem nejdříve vypočítán bod, na který se má agent pohnout, a poté je testován na přítomnost jiných automobilů. Detekce automobilů probíhá stejně jako detekce automobilů semaforového senzoru (viz Výpis 5).



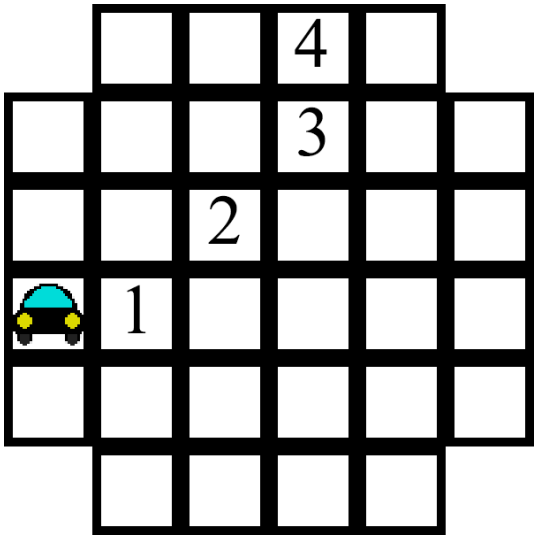
Obrázek 30 Příklad čekajících automobilových agentů na semaforu [Vlastní zpracování]

Detekce a pohyb křižovatkami

Agent se při každém pokusu o pohyb ptá, jestli bod, na který se chce posunout není entita křižovatky.

V případě že křižovatku detekuje, si na základě předchozích rozhodnutí vybere, jakým směrem se na křižovatce vydá. V případě že na začátku odbočil do levého odbočovacího pruhu, nastaví si do sady instrukcí pro pohyb instrukce pro odbočení doleva. V případě že se na začátku rozhodl jet rovně, náhodně si vybere, jestli chce na křižovatce pokračovat rovně, nebo jestli chce odbočit doprava. V případě že chce pokračovat rovně, má nastavený správný směr. V případě že chce odbočit doprava, nastaví si do sady instrukcí pro pohyb směr na odbočení doprava.

Náhodností výběru je zajištěna určitá stochasticita simulace, která se snaží lépe reprezentovat rozhodování řidičů v reálném světě.



Obrázek 31 Příklad sady instrukcí pro odbočení doleva [Vlastní zpracování]

Po vytvoření sady instrukcí se podívá, jaká barva semaforu je momentálně na bodě, na kterém se nachází. V případě že je červená, čeká na zelenou. V případě že má zelenou, využívá sadu instrukcí pro průjezd křižovatkou, a následně pokračuje do koncového bodu.

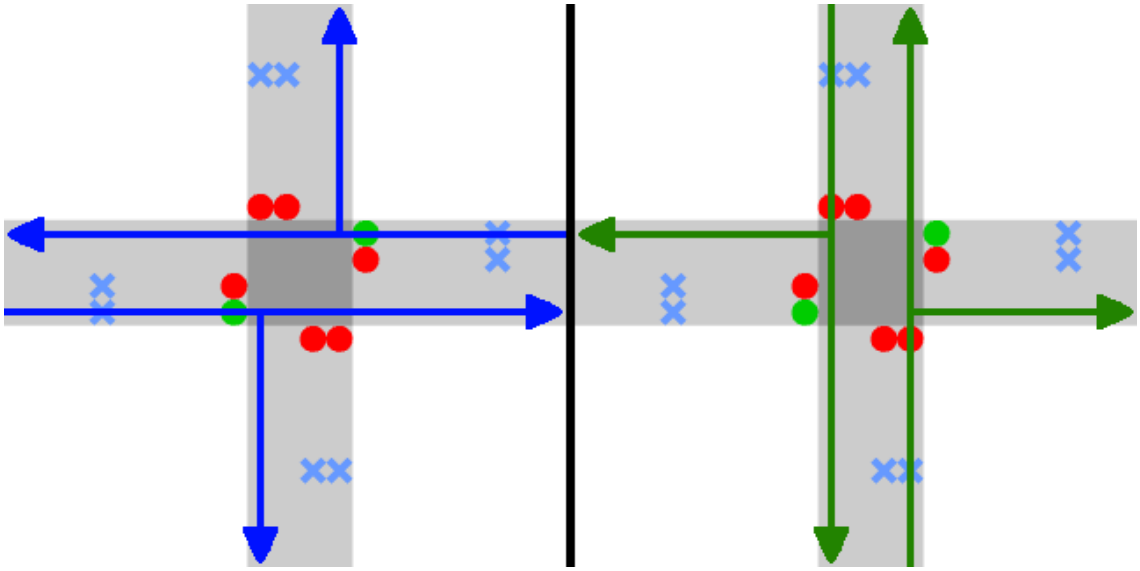
Způsoby řízení dopravy

Staticky nastavená délka intervalů

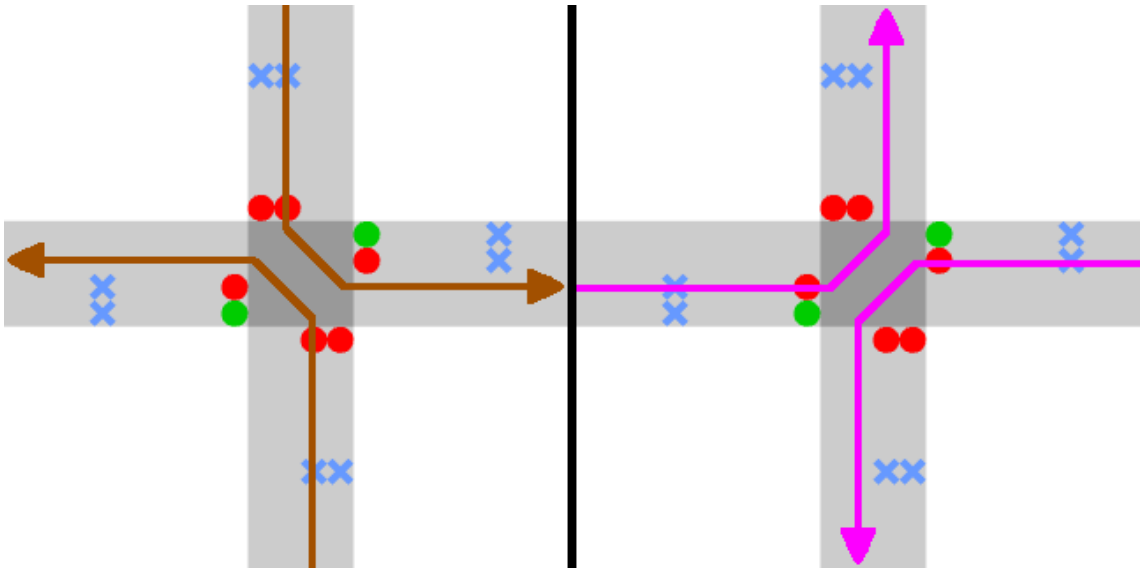
V tomto řešení není simulace nijak adaptována množstvím automobilů. V modelu jsou předdefinovány délky intervalů, a změny semaforů jsou prováděny v nadefinované sekvenci.

Částečné ovládání dopravního signálu

V řešení částečným ovládáním dopravního signálu je vždy zvolena jedna z hlavních (nejfrekventovanější) silnice. Tato silnice má vždy zelenou a mění se pouze v případě, že na vedlejších silnicích je pomocí senzorů detekován automobil.

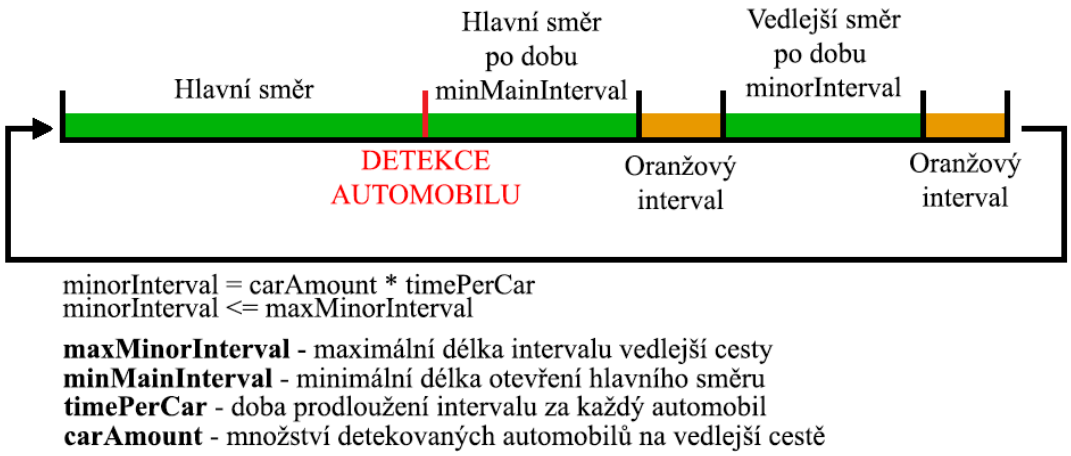


Obrázek 32 Povolené synchronní směry na křižovatce 1 [Vlastní zpracování]



Obrázek 33 Povolené synchronní směry na křižovatce 2 [Vlastní zpracování]

Z obrázků 32 a 33 lze vidět, jaké směry mohou zároveň projíždět skrze křižovatku. V případě, že například směr modrý je hlavním (nejfrekventovanějším) směrem, všechny ostatní směry jsou pak brány jako vedlejší. V tomto případě zůstává modrý směr otevřený tak dlouho, dokud není detekován automobil na silnici vedlejší. V případě že je automobil detekován, zůstává ještě hlavní směr otevřený po dobu minMainInterval (Obrázek 34), a poté se mění otevřený směr na ten, na kterém byl detekován automobil.



Obrázek 34 Princip částečného ovládání dopravního signálu [Vlastní zpracování]

Na obrázku 34 je popsán princip částečného řízení signálů.

V případě detekce více automobilů na více vedlejších cestách je stanovena prioritní řada, kde ten, kdo přijel ke křižovatce první, první odjíždí. Po uplynutí intervalu vedlejší cesty se vrací zpět na interval cesty hlavní a je držen po dobu minMainInterval (minimální povolená délka hlavního intervalu) a poté se pokračuje v prioritní řadě, dokud není prázdná. V případě že je prázdná, je opět nastaven hlavní směr, který je držen do doby detekce automobilu na vedlejší cestě.

Plné ovládání dopravního signálu

Tento způsob ovládání funguje podobně jako částečné ovládání dopravního signálu. Rozdíl je však ten, že detekce automobilů je prováděna na všech silnicích a žádná není nastavena jako hlavní. Toto vytvoří „adaptivní“ křižovatku, která mění své směry dle potřeby toku provozu.

Délka každého intervalu odpovídá počtu detekovaných automobilů v tomto směru. Každý detekovaný automobil prodlužuje délku intervalu o určenou hodnotu. Zároveň délka intervalu nemůže přesáhnout předdefinovanou maximální délku intervalu (což zajistí že není otevřený pouze jeden směr po příliš dlouhou dobu).

3. 3. Experimenty

Experimenty testují efektivitu způsobů ovládání dopravy v porovnání se staticky nastavenými délkami intervalů.

1. Experiment – Předdefinované ovládání proti částečnému ovládání

Ovládání statickými intervaly má v systému předdefinovanou délku intervalů každého stavu semaforu. Nijak se neadaptuje na nastalé situace.

Naopak částečným ovládáním signálů je semaforový systém schopen se adaptovat na nastalé situace, avšak částečné ovládání semaforů je vhodné využít pouze v případě, že se na křižovatce dá identifikovat ta hlavní (nejfrekventovanější). Částečné ovládání je vhodné například na hlavních silnicích ve městě, kde boční silnice jsou cesty k obytným komplexům, a nevjíždí k nim ani od nich velké množství automobilů, a naopak hlavní silnice je velice frekventovaná.

První experiment se snaží porovnat dva způsoby ovládání, což jsou řízení statickými intervaly a částečné ovládání signálů, na jednu předdefinovanou situaci.

Parametry:

Počet automobilů

Levá strana	**Pravá strana**	**Horní strana**	**Dolní strana**
1 za 2 kroky	1 za 2 kroky	1 za 100 kroků	1 za 100 kroků

Délky statických intervalů

Stav 1	**Stav 2**	**Stav 3**	**Stav 4**
8	8	50	8

Délka simulace: 5000 kroků

Šance automobilových agentů na odbočení doleva: 20%

Výsledky statickými intervaly

Výsledky	Počet automobilů	Průměrná doba čekání	Maximální doba čekání	Počet automobilů, které nečekaly	Procento automobilů, které nečekaly
Sim 1	2106	71,49	535	833	39,5 %
Sim 2	2303	67,3	488	786	34,1 %
Sim 3	2188	69,64	560	796	36,4 %
Sim 4	2362	55,06	491	803	33,9 %
Celkový průměr	2239,75	65,89	518,5	804,5	36 %

Výsledky částečným ovládáním signálu

Výsledky	Počet automobilů	Průměrná doba čekání	Maximální doba čekání	Počet automobilů, které nečekaly	Procento automobilů, které nečekaly
Sim 1	2494	40,1	428	930	37,3 %
Sim 2	2548	31,3	438	1008	39,6 %
Sim 3	2464	34,8	483	929	37,7 %
Sim 4	2491	34	440	957	38,4 %
Celkový průměr	2499,25	35,1	447,25	956	38,2 %

Z výsledků simulace, lze vypozorovat, že řešení částečným ovládáním signálů dokázalo za průběh simulace propustit o **10 %** více automobilů než v řešení statickými intervaly. Průměrná doba čekání je také razantně nižší v řešení částečným ovládáním, a to o **47 %**. V případě, že by nastala dlouhá vlna, kdy žádné automobily nezatáčejí nebo nepřijíždějí do/z vedlejší cesty, všechny automobily na hlavní cestě by v řešení částečným ovládáním

měly neustále zelenou a byla by nulová doba čekání. V tomto případě by ve statickém řešení byly signály měněny zbytečně.

2. Experiment – plné řízení dopravy proti statickým intervalům

Plné řízení dopravy je využíváno na křižovatkách, na kterých se ve všech směrech pohybuje průměrně stejný počet automobilů. Proto pro tento experiment bude simulace nastavena dle těchto podmínek.

Parametry:

Počet automobilů

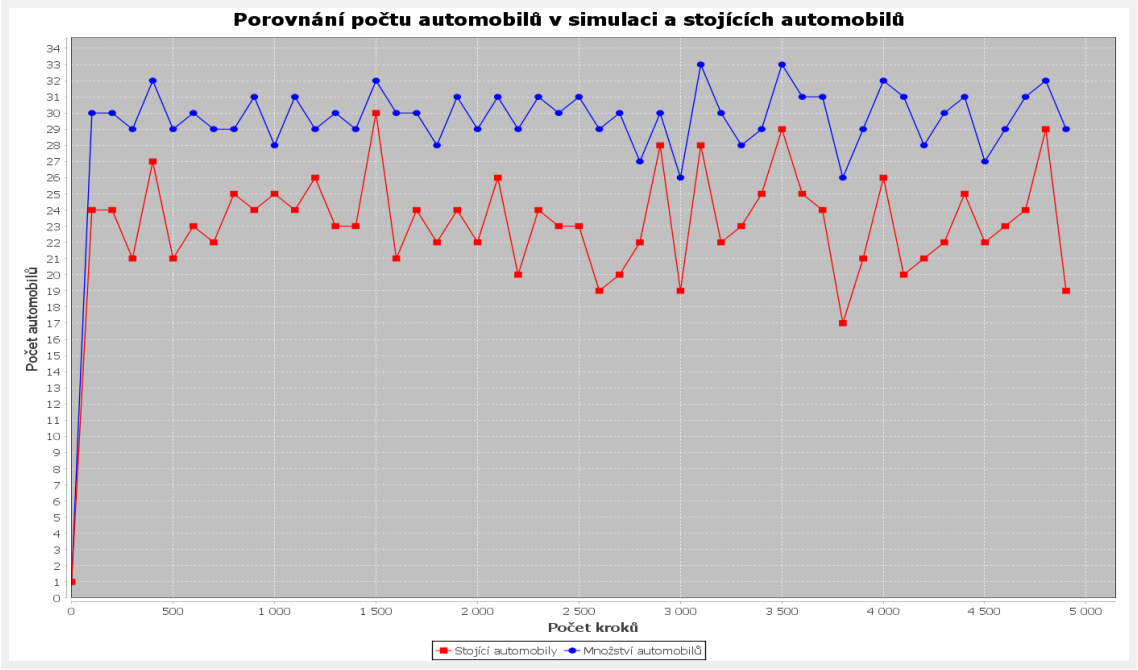
Levá strana	Pravá strana	Horní strana	Dolní strana
1 za 10 kroků	1 za 10 kroků	1 za 10 kroků	1 za 10 kroků

Délky statický intervalů

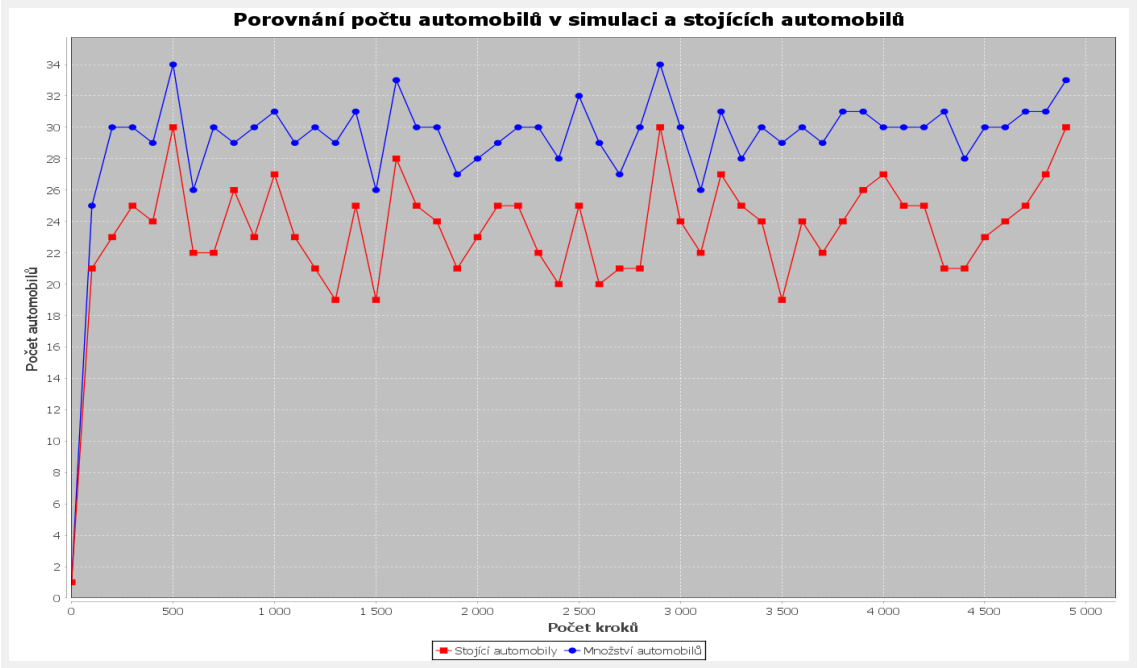
Stav 1	Stav 2	Stav 3	Stav 4
10 kroků	10 kroků	10 kroků	10 kroků

Délka simulace: 5000 kroků

Šance automobilových agentů na odbočení doleva: 50 %



Graf 1 Porovnání počtu automobilů s počtem stojících automobilů v řešení statickými signálem



Graf 2 Porovnání počtu automobilů s počtem stojících automobilů v řešení plným řízením signálu

Z grafů 1 a 2 se zdá, že i při naprosto ideálních podmínkách pro statické řešení, se mu řešení plným řízením signálů vyrovná. Není tomu však tak. Když se podíváme na výsledky, které lépe zachycují délky čekání na semaforu, je velice lehce viditelné, že řešení plným ovládáním je mnohonásobně lepší.

Výsledky statickým ovládáním signálů

Výsledky	Počet automobilů	Průměrná doba čekání	Maximální doba čekání	Počet automobilů, které nečekaly	Procento automobilů, které nečekaly
Sim 1	1903	111,45	384	377	20 %
Sim 2	1882	110,12	434	398	21 %
Sim 3	1939	98,49	336	397	20 %
Celkový průměr	1908	106,69	384,7	390,67	20,3 %

Výsledky plným ovládáním signálů

Výsledky	Počet automobilů	Průměrná doba čekání	Maximální doba čekání	Počet automobilů, které nečekaly	Procento automobilů, které nečekaly
Sim 1	1971	15,92	84	652	33 %
Sim 2	1970	16,68	81	626	32 %
Sim 3	1969	16,12	94	649	33 %
Celkový průměr	1970	16,24	86,3	642,3	32,7 %

Z výsledků lze vypozorovat, že automobily čekaly průměrně o **85 %** kratší dobu v řešení plným ovládáním než při řešení statickém. Zároveň byla maximální doba čekání výrazně kratší v řešení plným ovládáním, a to o **77 %.** V řešení plným ovládáním také větší množství automobilů vůbec na semaforu nečekalo, **33 %** oproti **20,3 %** celkového počtu automobilů.

3. Experiment – simulace špičky

Cílem tohoto experimentu, je dokázat schopnost adaptace částečného ovládání signálů na situace špičky (doba dne kdy je největší míra automobilové dopravy). Délka špičky většinou bývá 2 hodiny (nejčastěji 7:30 – 9:30 nebo 16:00 – 18:00).

Situace špičky bude simulována za pomocí zvýšení počtů automobilů v rozmezí 3000–4000 kroků. Celá délka simulace bude 5000 kroků. Zvýšení automobilů ve špičce bude dvojnásobné ve všech směrech.

Parametry:

Počet automobilů – normální

Levá strana	Pravá strana	Horní strana	Dolní strana
1 za 6 kroků	1 za 6 kroků	1 za 60 kroků	1 za 60 kroků

Počet automobilů – špička

Levá strana	Pravá strana	Horní strana	Dolní strana
1 za 3 kroky	1 za 3 kroky	1 za 30 kroků	1 za 30 kroků

Délky statických intervalů

Stav 1	Stav 2	Stav 3	Stav 4
10 kroků	10 kroků	40 kroků	10 kroků

Délka simulace: 5000 kroků ~ 3,4 hodiny

Délka špičky: 3000 kroků ~ 2 hodiny, v intervalu 1000-4000

Šance automobilových agentů na odbočení doleva: 5 %

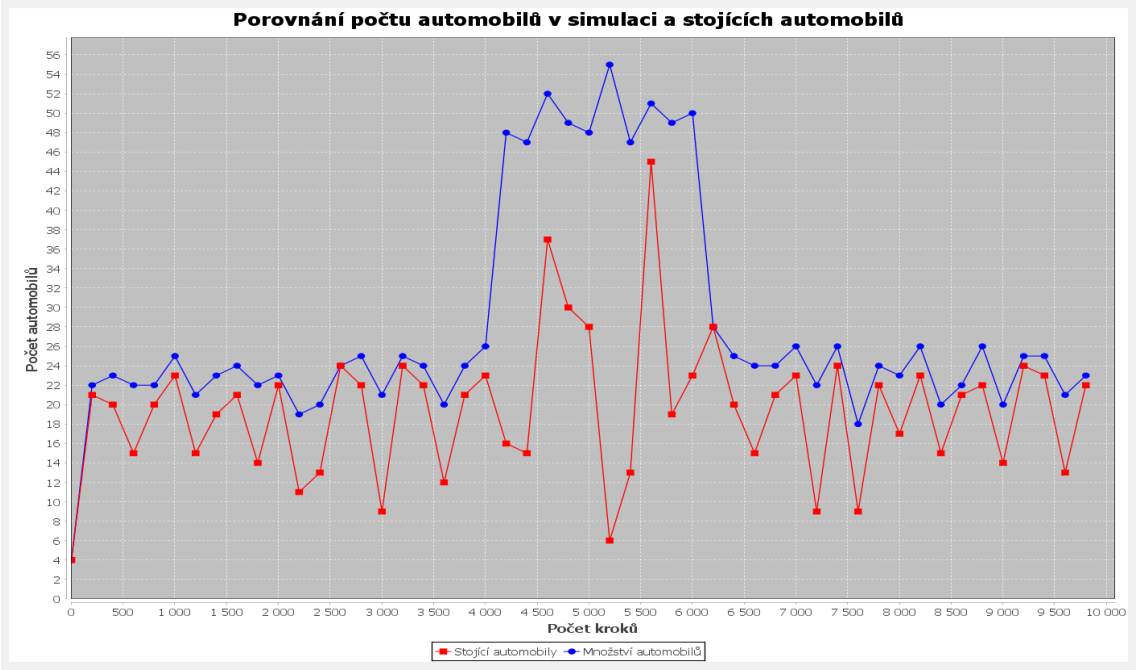
Výsledky částečným ovládáním signálů

Výsledky	Počet automobilů	Průměrná doba čekání	Maximální doba čekání	Počet automobilů, které nečekaly	Procento automobilů, které nečekaly
Sim 1	2851	16,88	188	538	18,9 %
Sim 2	2862	16,67	152	529	18,5 %
Sim 3	2862	17,1	124	541	18,9 %
Celkový průměr	**2858,333**	**16,88**	**154,67**	**913**	**18,8 %**

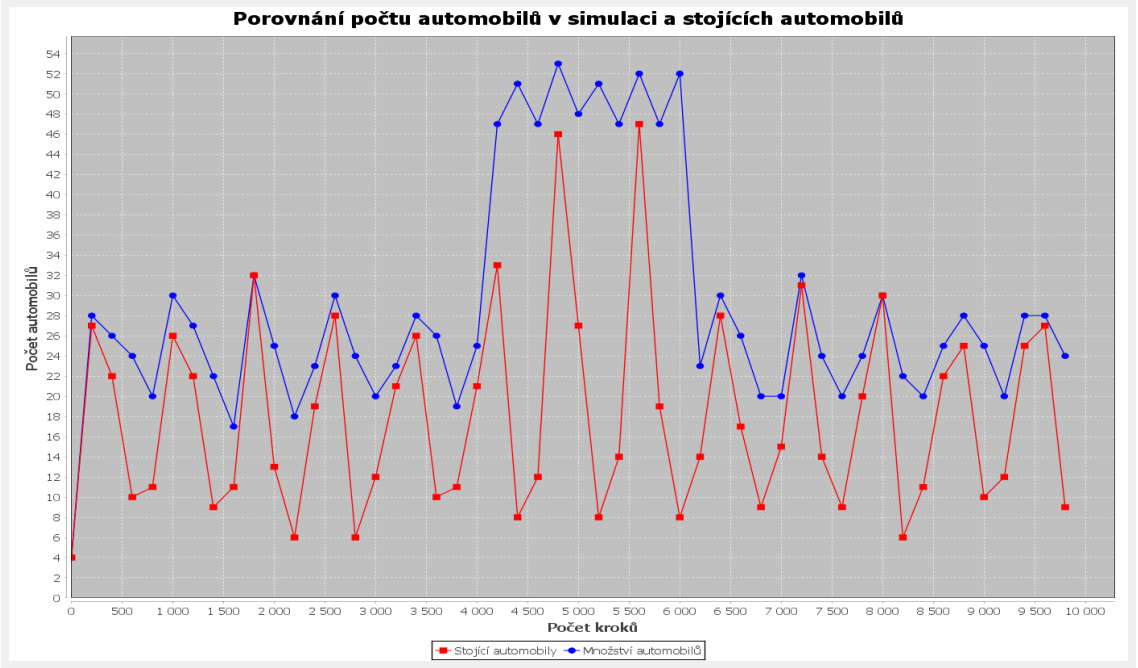
Výsledky statickým ovládáním signálů

Výsledky	Počet automobilů	Průměrná doba čekání	Maximální doba čekání	Počet automobilů, které nečekaly	Procento automobilů, které nečekaly
Sim 1	2445	28,64	82	394	16,1 %
Sim 2	2442	28,7	82	398	16,3 %
Sim 3	2435	28,9	78	406	16,7 %
Celkový průměr	**2440,67**	**28,76**	**80,67**	**399,3**	**16,4 %**

Z naměřených hodnot je opět vidět, že efektivnější řešení je částečné ovládání na silnicích, které jsou stresované špičkami dopravy. Průměrná doba čekání je v částečném řízení signálů o **41 %** kratší než v řešení statickém. Maximální doba čekání je u statického řízení signálů lepší. Avšak to se dá očekávat, jelikož při částečném řízení signálů je preferováno otevření hlavního směru, tudíž automobily na vedlejší cestě mohou čekat déle.



Graf 3 Porovnání počtu automobilů s počtem stojících automobilů v řešení částečným řízením signálů



Graf 4 Porovnání počtu automobilů s počtem stojících automobilů v řešení statickým signálem

Z grafů 3 a 4 lze vypozorovat, že propouštění automobilů je plynulejší v částečném řízení signálů. Počet automobilů (modrý) neobsahuje takové množství skoků jako v řešení statickém.

4. Výsledky

V praktické části byl implementován model semaforového řízení dopravy na křižovatce. Tento model využívá tří systémů řízení intervalů, a to statické nastavení intervalů, které je nastaveno pomocí parametrů uživatelem, částečné ovládání signálů a plné ovládání signálů.

Na konci praktické části je na modelu prováděna řada experimentů, které dokazují efektivitu využití částečného a plného ovládání signálů oproti statickým intervalům. Tyto experimenty jsou prováděny na předem definovaných situacích a snaží se co nejpřesněji zachytit reálné situace.

4. 1. Výhody

Díky možnosti využití Javy je možné tvoření velice komplexních systémů za pomocí objektově orientovaného programování.

Obsahuje velkou řadu knihoven pro tvorbu modelů.

Obsahuje podporu pro sociální sítě, genetické algoritmy, systémovou dynamiku a geografické informace (GIS).

Umožňuje tvorbu modelů v jazyce Logo, který je přívětivý pro začínající tvůrce modelů.

4. 2. Nevýhody

Repast Simphony je synchronní systém, což znamená, že chování agentů je prováděno synchronně (za sebou). Například, agent A provede akci a až poté pokračuje agent B. Jejich rozhodování není nikdy prováděno paralelně. Toto znemožňuje tvoření plně asynchronních systémů.

Nutnost znalosti objektového programování pro vývoj modelů implementací v Javě.

Dokumentace Repast Simphony není tak rozsáhlá jako například dokumentace NetLogo.

Počet příkladových modelů není velký. Naopak například NetLogo nabízí na vlastních stránkách odkaz na jejich sdílející platformu Modelling Commons, kde lze nalézt velkou řadu existujících NetLogo modelů, a tento katalog pouze stále roste.

Repast Simphony není nejvhodnější pro tvorbu modelů řešících ryze dopravní situace z důvodu existence řady programů určených právě pro tvorbu takovýchto simulací. Zároveň z důvodu využívání krokového spouštění funkcí může být složité přímo nastavit podmínky, které odpovídají realitě.

5. Závěr

V této práci byl čtenář seznámen s principy agentového modelování a jejich komponentami. Byly popsány typy agentů dle jejich složitosti a chování. Dále byly vysvětleny typy prostředí, ve kterých se agentové modely odehrávají. Poté byla přiblížena problematika dopravy a semaforového řízení a byly představeny způsoby řízení semaforových signálů. Následně byly ukázány modely, které řeší semaforovou dopravu.

V praktické části byla vytvořena simulace dopravy na městské křižovatce v Repast Simphony, ve které byly využity dva typy agentů, a to jednoduchý reflexní agent a modelově založený reflexní agent. Simulace se odehrávala v plně pozorovatelném, deterministickém, sekvenčním, nepřetržitém, více agentovém prostředí. Tato simulace využívá tří způsobů řízení dopravy, a to statických intervalů, částečného ovládání a plného ovládání signálů. Na této simulaci byla poté dokázána efektivita využití těchto způsobů ovládání semaforové dopravy, a bylo prokázáno, že využitím částečného/plného ovládání signálů je zefektivněn tok dopravy na křižovatce.

Model vytvořený v praktické části ukázal, že je možné v Repast Simphony tvořit simulace dopravy, není to však ideální. Jelikož simulace dopravy je v dnešní době velice rozšířené a důležité téma, existuje již řada simulačních programů přímo určených pro tvorbu dopravních simulací. Jedním z programů je například simulační software dopravy od firmy Anylogic, který je velice efektivní v implementaci dopravních simulací. Dalším programem je například PTV Vissim, který také nabízí řadu funkcí specificky určených pro tvorbu simulací dopravy a je momentálním lídrem v tomto odvětví. Využití Repast Simphony pro tvorbu dopravních simulací je vhodné, pouze v případě nutnosti vlastní definice modelu prostředí, vnímání agentů a rozhodování.

Jelikož byl model primárně určen na měření efektivity způsobů řízení dopravy, byla využita určitá míra abstrakce. V budoucnu by mohli být automobiloví agenti rozšířeni, aby využívali řadu dalších rozhodovacích hledisek, jako je zrychlování/zpomalování na základě vzdálenosti od dalších automobilů. Také by mohl model využívat geografických dat, pro možnost simulace v reálných prostředích, což Repast Simphony umožňuje. Dalším extrémním rozšířením by mohla být simulace fyzických a psychologických

aspektů řidičů, pro lepší reprezentaci reálných situací, to je však neskutečně složité a zdaleka by přesahovalo rámec této práce.

6. Zdroje

[1] Macal, C., North, M., 2010, *Tutorial on agent-based modelling and simulation*. J Simulation 4, 151–162 Dostupné z: https://doi.org/10.1057/jos.2010.3

[2] Macal, C., North, M., 2005, *Tutorial on agent-based modelling and simulation*. J Simulation

[3] Bonabeau, Eric., 2002, *Agent-based modeling: Methods and techniques for simulating human systems*. Proceedings of the National Academy of Sciences 99.suppl 3: 7280-7287 Dostupné z: https://doi.org/10.1073/pnas.082080899

[4] Rakshit P., Konar A., 2018, *Agents and Multi-agent Coordination. In: Principles in Noisy Optimization. Cognitive Intelligence and Robotics*. Springer, Singapore Dostupné z: https://doi.org/10.1007/978-981-10-8642-7

[5] North, M.J., Collier, N.T., Ozik, J., 2013, Complex adaptive systems modeling with Repast Simphony. *Complex Adapt Syst Model* **1,** 3. https://doi.org/10.1186/2194-3206-1-3

[6] Joshua M. Epstein & Robert L. Axtell, 1996. *Growing Artificial Societies: Social Science from the Bottom Up*. MIT Press Books, The MIT Press, edition 1, volume 1, number 0262550253, December.

[7] Younes, M.B., Boukerche, A. 2018. *An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems*. Wireless Netw 24, 2451–2463 Dostupné z: https://doi.org/10.1007/s11276-017-1482-5

[8] Hillhouse G. *How do traffic lights work?* In: YouTube [online]. Zveřejněno 14.5.2019. Dostupné z: https://www.youtube.com/watch?v=DP62ogEZgkI&ab_channel=PracticalEngineering

[9] Collier, N. and North, M., 2013, '*Parallel agent-based simulation with Repast for High Performance Computing*', SIMULATION, 89(10), pp. 1215–1235. doi: 10.1177/0037549712462620.

[10] Ozik., J. 2018 [online] '*ReLogo getting started guide*' Dostupné z: https://repast.github.io/docs/ReLogoGettingStarted.pdf

[11] Collier N. a North. M. [online] '*Repast Java getting started guide*', Dostupné z: https://repast.github.io/docs/RepastJavaGettingStarted.pdf

Modely

[12] Wilensky, U., 1998 [online]. *NetLogo Traffic Intersection model*. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. Dostupné z: http://ccl.northwestern.edu/netlogo/models/TrafficIntersection

[13] Wilensky, U., 2003 [online]. *NetLogo Traffic Grid model*. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. Dostupné z: http://ccl.northwestern.edu/netlogo/models/TrafficGrid

[14] Eric Tatara, *Příkladové modely Repast Simphony*, [online] Repast Simphony. Dostupné z: https://github.com/Repast/repast.simphony.models

[15] Alexander Wöstmann, 2018 [online]. *Traffic light simulation*. Dostupné z: https://github.com/awoestmann/traffic-light-sim