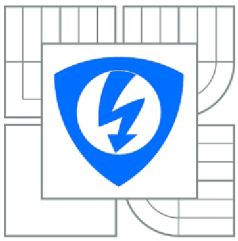
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

MULTIAGENTNÍ SYSTÉM PRO SIMULACI A ANALÝZU DOPRAVNÍHO PROVOZU

MULTIAGENT SYSTEM FOR TRAFFIC FLOW SIMULATION AND ANALYSIS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

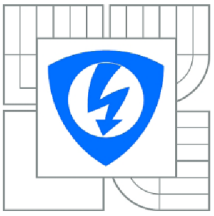
AUTOR PRÁCE
AUTHOR

Bc. VLADIMÍR KOUTNÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR HONZÍK, Ph.D.

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřící techniky

Diplomová práce

magisterský navazující studijní obor

Kybernetika, automatizace a měření

Student:	Bc. Vladimír Koutný	***ID:***	83609
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Multiagentní systém pro simulaci a analýzu dopravního provozu

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou autonomních multiagentních systémů. Navrhnete koncepci mutliagentního systému pro simulaci dopravy s využitím zvolené architektury. Zaměřte se na problém distribuce provozu při nestandardních situacích (nehoda, krátkodobá uzavírka, ...) za předpokladu možnosti ovlivnit trasu jednotlivých vozidel.

DOPORUČENÁ LITERATURA:

Dle pokynu vedoucího práce a vlastního výběru.

Termín zadání:	8.2.2010	***Termín odevzdání:***	24.5.2010

Vedoucí práce: Ing. Petr Honzík, Ph.D.

prof. Ing. Pavel Jura, CSc.

Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Tato diplomová práce se zabývá zpracováním orientovaného grafu pomocí multiagentního systému určeného pro simulaci dopravy. Práce byla vytvořena jako rešeršní studie. Z dané studie bylo po té vytvořeno simulační prostředí schopné reagovat na nejrůznější podněty. Jako agenti jsou zde myšlena jednotlivá vozidla a jejich řidiči, kteří mají různé vlastnosti a podle nich na dané podněty reagují. Komunikace je prováděna přes tzv. superagenta, který ví o všem, co se na dané mapě děje a tyto informace předává dílčím agentům. Agenti jsou předně schopni reagovat na ucpání silnice (uzavírka, nehoda) nebo na kolonu. U těchto situací se provede algoritmus určený pro nalezení nové trasy. Mimo řízených agentů můžou na mapě figurovat také agenti simulující běžný provoz.

Klíčová slova

Průchod orientovaným grafem, Eulerův graf , Dijkstra algoritmus, A* algoritmus, Multiagentní systém, model IRMA, Simulace dopravy, trojúhelníkový model, Greenshield model.

Abstract

This diploma thesis deals with an oriented graph processing applying a multi-agent system designated for traffic simulation. The thesis was written as a research study. Based on the study, a simulation environment was created able to respond to various stimuli. Meant as the agents, there are vehicles and their drivers that have various features, based on which they respond to the given stimuli. Communication is conducted via a so-called super-agent that monitors all action on the map and passes this information on particular agents. The agents are able to respond in advance to traffic jams (closures, accidents). In such situations, an algorithm designated for a new route finding is conducted. Besides the controlled ones, there can operate on the map also agents simulating common traffic.

Keywords

Passage through oriented graph, Euler graph, Dijkstra algorithm, A* algorithm, Multi-Agent systems, IRMA model, Transport simulation, Triangular model, Greenshield model.

Bibliografická citace

KOUTNÝ, V. Multiagentní systém pro simulaci a analýzu dopravního systému. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 64 s. Vedoucí diplomové práce Ing. Petr Honzík, Ph. D.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Multiagentní systém pro simulaci a analýzu dopravního systému jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne: **24.5.2010** .....

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Petru Honzíkovi, Ph.D. za konzultační pomoc při vypracování zadané práce, Ing. Jakubu Řezáči za konzultační pomoc v otázce programování v prostředí NetBeans.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1. MOTIVACE

Problematika simulace dopravy je problém, který museli řešit při stavbě měst již staří Římané. Ti ovšem na rozdíl od nás stavěli tak říkajíc „Na zelené louce“ a k plynulému průjezdu měst jim stačilo vytvořit vhodný systém na sebe kolmých ulic. Římané nemuseli počítat s tak velkým počtem vozů na počet obyvatel. A tak při ucpání jedné silnice prostě objeli blok a pokračovali dále ve vytyčeném směru.

V dnešní době je nárůst automobilové dopravy opravdu velmi strmý. Počet osobních vozů se od revoluce v roce 1989 zdvojnásobil. V polovině roku 2009 bylo v České republice registrováno 4,4 miliónu vozů, zatímco před 20 lety jich bylo o 2 milióny méně [1]. Počet vozidel na jednotlivé regiony je samozřejmě odlišný. Například v Praze je na každého občana včetně nemluvňat přihlášeno jedno vozidlo. Průměrný počet vozidel na celou Českou republiku je pak přibližně 2,5 obyvatele na jeden vůz. Pokud vezmeme v potaz tyto hodnoty a dále také to, že většina městské zástavby byla plánována v druhé polovině minulého století, nelze se proto divit, že v období špičky strávíme v automobilu více času než ve veřejné dopravě.

Proto, aby bylo možno tento problém vyřešit, vznikají v dnešní době stále nové simulátory dopravy, které jsou vhodným pomocníkem při řešení nejrůznějších problému v dopravě. Tyto simulátory se pak zabývají nejrůznějšími podotázkami dopravy a to především v odvětví logistiky.

Tato práce je rozdělena do dvou obsáhlejších částí. První část je zaměřena na teorii potřebnou k navázání na samostatnou práci. Tvoří ji 3 větší samostatné celky, kde se čtenář dozví o tvorbě orientovaného grafu a jeho vlastnostech. V části o průchodu orientovaným grafem jsou pak popsány algoritmy, které jsou vhodné pro průchod navržené mapy. V posledním celku teoretické části se čtenář seznámí s některými teoriemi určených k simulaci dopravy. V části určené pro řešení problematiky je popsána samotná realizace včetně manuálu určeného k práci s programem.

1.1 CÍL PRÁCE

Tato práce si dává za úkol vytvořit flexibilní simulátor dopravy. Tento simulátor následně dokáže pomocí daných vstupů vytvořit simulaci libovolného města.

Druhým cílem je pomocí proměnných parametrů simulovat nejrůznější změny v daném dopravním systému. K těmto změnám patří nejjednodušší úpravy jako zakomponování nákladní dopravy nebo změna nastavení průjezdnosti křižovatek a vytvoření uzavírek, až po přestavbu jednotlivých částí města a opětovné simulace.

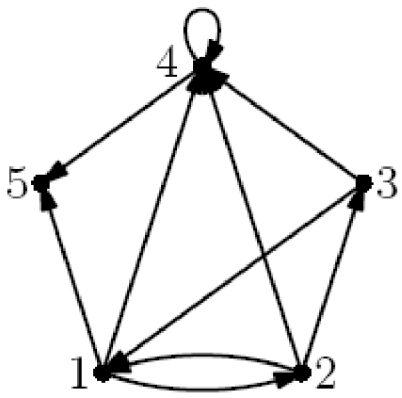
Třetím vytyčeným cílem je vyhodnotit plynulost provozu za předpokladu, že jsou některá vozidla vybavena nejen navigací, ale také informací o aktuální hustotě provozu na plánované trase.

2. ÚVOD DO PROBLEMATIKY

V této části jsou uvedeny základní informace, potřebné k dalšímu rozvoji teorie dopravní simulace použité v diplomové práci.

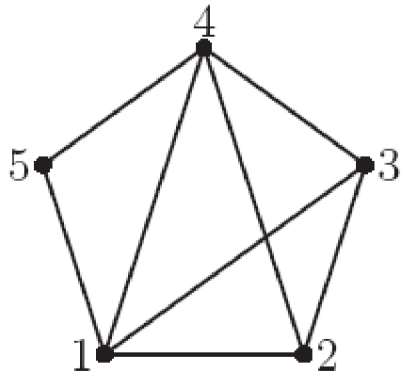
2.1 TEORIE ORIENTOVANÝCH GRAFŮ

Tato část se zabývá teorií orientovaných grafů. Snahou je přiblížit danou problematiku v širším pohledu, než je použita v praktické části této práce, z důvodu navazujících částí o průchodu grafem.



Obr. 2.1.1 Př. Orientovaného grafu [3]

Graf je tvořen z uzlů (nebo také vrcholů) a hran. Daná hrana vždy spojuje dva uzly. Pokud se jedná o dvojici neuspořádaných uzlů, pak se jedná o graf neorientovaný (symetrický). Pokud však řekneme, že jeden uzel je uzlem počátečním a druhý uzel je uzlem koncovým, mluvíme o grafu orientovaném (digraf). Z orientovaného grafu lze přejít pomocí symetrizace zpět na graf neorientovaný. Násobné hrany poté zredukujeme na jednu a uzly zcela odstraníme. Poté se orientovaný graf. na obr. 2.1.1 změní na neorientovaný graf na obr. 2.1.2.



Obr. 2.1.2 Symetrizace grafu z obr. č. 1 [3]

Orientovaný graf je pak formálně trojice $G=(V,H,\sigma)$, kde V tvoří konečnou množinu vrcholů, H tvoří konečnou množinu hran a σ je vztahem incidence.

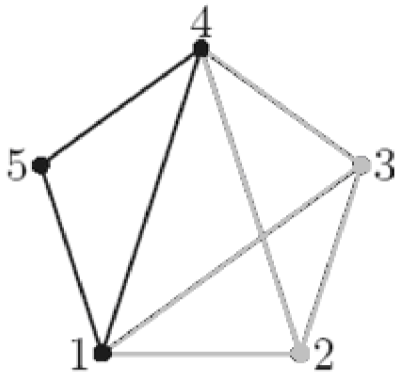
$$\sigma : H \rightarrow V \times V \qquad (2.1.1)$$

Toto uspořádání pak přiřazuje k jedné hraně H uspořádanou dvojici vrcholů (x,y). První z dvojice (x) je uzel počáteční$(PU(h))$, druhý (y) je uzel koncový$(KU(h))$. Můžeme také říci, že vrchol x sousedí z vrcholem y. Orientovaný graf může obsahovat smyčky a izolované vrcholy. **Smyčka** vznikne pokud orientovaná hrana začíná a končí ve stejném vrcholu. O izolovaném vrcholu hovoříme pokud vrchol není incidentní s žádnou hranou. Pokud graf neobsahuje žádnou smyčku, nazýváme takový graf, **grafem Acyklickým**. [2,4]

Díky vlastnostem Symetrizace lze některé vlastnosti neorientovaných grafů využít i na grafy orientované. Tyto vlastnosti jsou níže popsané.

Podgraf – Graf G´ je podgrafem $G \subseteq G'$ jestliže $V(G') \subseteq V(G)$ a současně $H(G') \subseteq H(G)$. Podgraf vznikne vyjmutím některých vrcholů V a všech hran H, které jsou na tyto vrcholy navázané. Na obr. 2.1.3 je tmavou barvou znázorněn podgraf. Většina dalších vlastností se u dopravního simulátoru objevuje pouze v části podgrafů, proto je tato vlastnost uvedena na prvním místě. Podgraf se využívá např. při řešení mikroskopických problémů, jako průjezd křižovatkou nebo u makroskopických problémů ucpání silnice. Navíc jej lze využít při částečném přibližování úseků mapy.

Obdobně jako Podgraf může vzniknout i **Faktor grafu**. Faktor grafu však vzniká tak, že se vynechá jedna nebo více hran, ale všechny vrcholy zůstanou zachovány a žádný z těchto vrcholů nebude izolován. Pokud odstraněním jedné hrany dojde k rozpadnutí grafu na dva podgrafy, mluvíme o **Mostu**. [3]



Obr. 2.1.3 Vytvoření podgrafu [3]

Stupeň vrcholu je vlastnost odlišná pro neorientované i orientované grafy. U orientovaných grafů je to počet hran spojených s jedním určitým vrcholem. Orientovaný graf pak obsahuje **polostupně vrcholu**. Polostupně vrcholu obsahují počet výstupních a vstupních hran. V našem případě počet polostupňů vrcholu není maximalizován. Avšak při vyšším stupni jak (5,5) nebo-li pět vstupů a pět výstupů na jeden vrchol dochází ke vzniku nepřehledného grafu. [3,4,6]

Ohodnocený graf je graf, na jehož hranách je nesena určitá informace, důležitá pro průchod daným grafem. V našem případě se například jedná o hodnotu určující počet pruhů v daném směru. [3,4,6]

Cesta v grafu $G = (V, H)$ je rovna posloupnosti vrcholů P = (V_1,V_2,...V_k). V_1 je vrchol počáteční, V_k je vrcholem koncovým. Vrcholy mezi těmito uzly jsou tedy vrcholy průchodu. Základními podmínkami je to, aby existovala hrana mezi po sobě následujícími vrcholy. Je také potřebné, aby nedošlo k tomu, že by se dva po sobě následující vrcholy a tedy i jejich hrana opakovaly. Pokud máme orientovaný graf, je také i cesta orientovaná. Cesta v grafu se vyhledává

prostřednictvím algoritmů. Obdobně jako cesta grafem, která je tvořena posloupností vrcholů.

Existuje i posloupnost hran Q = (H_1,H_2,...H_k). Taková posloupnost se nazývá **Tah**. Obdobně jako u cesty ani zde nesmí dojít k tomu, aby se některá z hran zapsala vícekrát. [3,4,6]

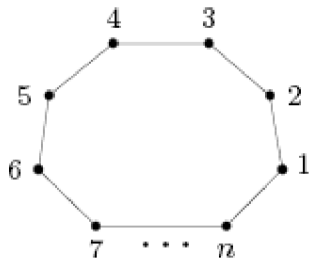
Obecně se dá pak říci, že každá posloupnost hran nebo vrcholů je nazývána **sledem**. Z tohoto tvrzení vyplývá, že Tah i Cesta je jen speciálním případem sledu. A také to, že pokud dojde k opakování hrany nebo vrcholu, nejedná se již o Tah resp. Cestu, ale pouze o sled. Speciálním případem je také **uzavřený sled**. Jedná se o sled, který začíná a končí v daném vrcholu. Pokud mluvíme o sledu tvořeném jen z jednoho vrcholu a neobsahujícího žádnou hranu, jedná se pak o tzv. **Triviální sled.**

Dostupnost vrcholu y je *orientovaně dostupná* z vrcholu x, jestliže existuje orientovaný sled, vedoucí z vrcholu x do vrcholu y. Pokud lze pro každou takovou dvojici v grafu nalézt cestu, jedná se o **Silně souvislý graf.** [3,4,6]

Algoritmy pro získání cesty jsou popsány v kapitole **2.2 Průchod orientovaným grafem**. Obecně lze však říci, že každý řízený automobil musí splnit hlavní podmínku této vlastnosti. Vůz by neměl najet na danou hranu dvakrát ve stejném směru, pokud nedošlo k nepředpokládané situaci a nedošlo k přepočítání cesty.

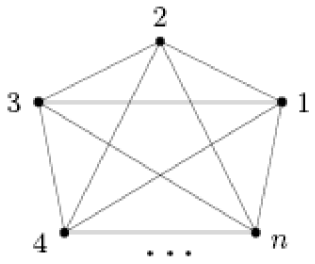
Některé grafy mají své speciální vlastnosti. Tyto grafy jsou často vyhledávány v grafech pomocí algoritmů. Tvoří tedy určité podgrafy a jsou v určité míře obsaženy ve všech grafech. Takovými grafy mohou být např. kruhový, úplný bipartitní graf atd. Na tomto místě si několik z nich popíšeme.

O **kruhovém grafu** hovoříme v případě, pokud je graf tvořen tak, že jeho vrcholy jsou spojeny vždy dvěma hranami k nejbližším sousedům a neobsahují žádné jiné spoje. Uzavřený sled průchodu takovým grafem je pak roven výčtu vrcholů se začátkem a koncem v prvém vrcholu. Kruhový graf může být neorientovaný i orientovaný a to jak jednosměrný tak obousměrný. [6]



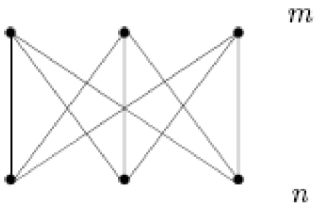
Obr. 2.1.4 Kruhový graf [6]

Úplný graf je tvořen *N* vrcholy, které jsou mezi sebou pospojovány tak, že se z každého vrcholu lze dostat do jiného libovolného vrcholu pomocí použití jediné hrany. Z toho plyne, že takový graf obsahuje $\binom{n}{2}$ hran. Příklad takového grafu je zobrazen na obr.2.1.5.



Obr. 2.1.5 Úplný graf [6]

Bipartitní graf je tvořen pomocí dvou skupin vrcholů. Každá z těchto skupin musí být neprázdná. Z každého vrcholu *m* jsou pak vytvořeny hrany spojující daný vrchol s vrcholy v druhé skupině *n*. Tyto grafy mají své uplatnění např. v umělé inteligenci.



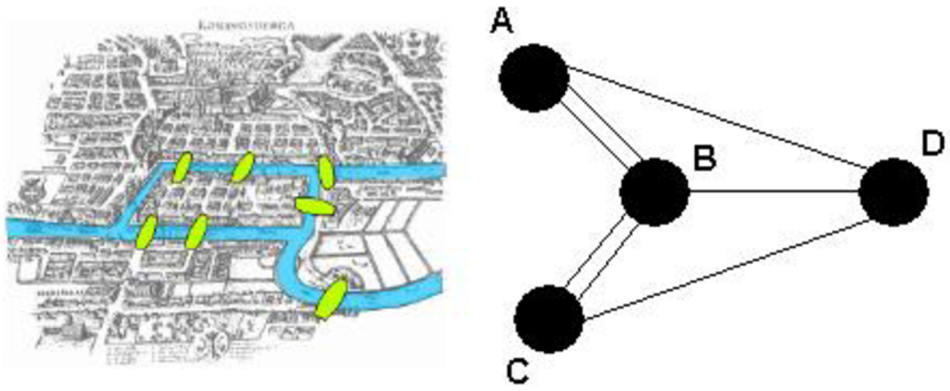
Obr. 2.1.6 Bipratitní graf [6]

Další zajímavé získávané vlastnosti z grafů sice nejsou příliš využity v této práci, dovolují nám však více pochopit složitost průchodu grafem. Z těchto důvodů zde popíšu teorii Hamiltonových a Eulerůvých cyklů.

2.1.1 Eulerův cyklus

Eulerův cyklus je nazýván také Eulerův tah. Je pojmenován po švýcarském matematikovi Leonhardu Eulerovi, který jako první vyřešil problém Königsbergských mostů.

Jednalo se o matematickou otázku, kdy v Königsbergu na řece Pregle byly dva ostrovy spojeny s oběma stranami nábřeží a mezi sebou celkem 7 mosty. Otázka zněla, zda je možno naplánovat cestu z jednoho libovolného bodu a do tohoto bodu se vrátit, přitom však neprojít přes žádný most dvakrát.



Obr. 2.1.7 Řešení Königsbergských mostů [6]

Euler postupoval tak, že si každou část označil písmenem a vytvořil z nich vrcholy, mosty pak použil jako hrany a vytvořil tak graf. Vznikla tak první realizace neorientovaného grafu. Po získání tohoto grafu se dotazoval na to, existuje-li uzavřený tah (dnes se mluví o **Eulerovském tahu**) . Euler zjistil, že takový tah nelze realizovat z toho důvodu, že všechny vrcholy grafu jsou lichého stupně. Uzavřený tah se musí realizovat tak, že se do každého uzlu vrátíme tolikrát, podle toho, jaký má stupeň (kolik hran je s ním spojených). Pokud bychom alespoň dostavěli dva mosty, byla by otázka řešitelná.

Eulerův cyklus jako takový je tvořen dvěma navazujícími kroky. Nejprve si vybereme libovolný uzel v daném grafu a zvolíme jej za startovací uzel. Druhým krokem je pak procházení vlastního grafu tak, že si pečlivě vybereme hranu, po které půjdeme a tuto hranu následně vyškrtneme z grafu. Hrany se vybírají tak, aby vnikaly izolované uzly, které můžeme vyškrtnout a aby při výběru hrany nedošlo (pokud je to možné) k rozdělení grafu na dvě části.

Z výše popsaného příkladu lze odvodit definici pro eulerův neorientovaný graf. Graf který obsahuje uzavřený tah (v_0, e_1, v_1, ..., e_k, v_0), ve kterém se každá hrana objevuje pouze jednou, se nazývá **eulerovský graf** a daný tah je potom **uzavřený eulerovský tah.** Eulerovský graf je tedy speciálním případem grafu, který lze nakreslit jedním tahem.

Z této definice lze odvodit vlastnosti orientovaného grafu. Orientovaný graf je uspořádaná dvojice (V,E), kde V je množina vrcholů a E je množina hran, přičemž hrana e z E je uspořádaná dvojice dvou navzájem různých vrcholů.

U orientovaných grafů se zkoumá tzv. **Vstupní stupeň vrcholu**. Tento Vstupní stupeň je číslo určující počet šipek do daného uzlu vstupujících. Obdobným způsobem lze získat **výstupní stupeň vrcholu**. Pokud se obě tyto hodnoty rovnají, můžeme o vrcholu říct, že se jedná o **vrchol vyvážený**. Pokud je graf tvořen pouze z vyvážených vrcholů, pak i graf je **grafem vyváženým.**

Z těchto vlastností lze vyvodit také vlastnosti orientovaného eulerovského grafu. **Eulerovský orientovaný graf** je orientovaný souvislý vyvážený graf. [7]

2.1.2 Hamiltonův cyklus

Hamiltonův cyklus je vrcholovou obdobou cyklu Eulerovského. Změna cyklu z hran na vrcholy však sebou přináší neúměrné zvýšení složitosti celkového cyklu. Hamiltonův cyklus tedy prochází grafem G tak, že každým uzlem může projít pouze jednou. Graf, který splňuje tuto podmínku, se nazývá **grafem hamiltonovským.** Na rozdíl od Eulerovského grafu není u těchto grafů známá jejich přesná charakteristika. Hamiltonův cyklus patří do úloh s řešením úplným. Z toho vyplývá, že jeho řešení je vždy možné nalézt v reálném čase. [7]

2.2 PRŮCHOD ORIENTOVANÝM GRAFEM

Průchody orientovaným grafem jsou tvořeny pomocí algoritmů speciálně navržených pro tuto činnost. Každý z těchto algoritmů řeší určitý specifický problém a lze říci, že každá činnost se dá nasimulovat určitým algoritmem. Algoritmy průchodu grafem, stejně jako graf samotný, se dají použít na libovolnou lidskou činnost.

V této části reprezentované algoritmy lze rozdělit do několika skupin. Nejpoužívanějšími algoritmy jsou obecně algoritmy používané na prohledávání délky průchodu grafem. Tyto algoritmy jsou obecně psány pro neorientované grafy, ale většina z nich se dá obdobně použít i pro grafy orientované. Kromě vzdálenosti nás však může zajímat také čas strávený na silnici, nebo typ využívané silnice, popř. i počet projetých křižovatek. Některé algoritmy se pak dají použít na několik výstupních informací současně.

Algoritmy jako Floid-Wharsallův nebo prohledávání do šířky lze sice použít obecně na grafy, ale jejich hlavní nevýhodou je to, že při počítání průchodu musí být velikost hrany rovna jedné. V reálném světě však žádná silnice nemá velikost rovnu jedné. Teoreticky by se tyto algoritmy mohly použít v Mikroskopické simulaci.

Pro vyhledávání nejkratší cesty ve váženém algoritmu se používá nejčastěji Dijkstra algoritmus nebo jeho vylepšení A*. Oba algoritmy jsou vytvořeny na principu prohledávání do šířky.

Ohodnocené algoritmy jsou algoritmy určené pro vyhledávání cesty v ohodnoceném grafu a je vcelku nepodstatné, zda se jedná o graf orientovaný nebo neorientovaný.

Pro představu o celkovém využívání algoritmů kromě zde algoritmů použitých v této práci (Dijkstra a A*), budou zde popsány také algoritmy Moorův, Prohledávání do šířky, Prohledávání do hloubky a další.

Prohledávání do šířky i prohledávání do hloubky patří mezi tzv. neinformované nebo také slepé algoritmy. Do této skupiny slepých algoritmů patří také např. prohledávání uspořádaným výběrem a nebo obousměrné prohledávání. Výhodou neinformovaných algoritmů je úplnost, pokud má daná úloha řešení, toto

bude vždy nalezeno. Z těchto algoritmů se následně vyvinuly algoritmy informované, využívající prořezávání stromu řešení.

Pro dopravu existují i speciální algoritmy pracující na principu toku. V této práci s nimi není počítáno, protože je lze využít pouze u makroskopických modelů a nikoliv u mikroskopických. V této části tak je popsán pouze jeden z těchto algortimů a to Ford-Fulkersonův algoritmus.

2.2.1 Prohledávání do šířky (Breath-First search)

Jedná se o algoritmus nejvíce využívaný pro prohledávání stromového grafu. Průchod do šířky se provádí následovně. Nejprve se vezme uzel s nejnižší hloubkou *k,* neboli ze startovacího uzlu *u* se vyhledají všechny uzly, které jsou s tímto uzlem spojeny na vzdálenost k+1. To znamená, že se do těchto uzlů lze dostat pomocí jedné jediné hrany (jsou jeho sousedé). Všechny nalezené uzly jsou označeny. Pokud jsou nalezeny všechny tyto uzly, je uzel *u* označen jako „projitý“ a algoritmus pokračuje na prvý nalezený uzel ve vzdálenosti k+1. Algoritmus se opakuje, dokud není splněn cíl. Tímto cílem může být nalezení daného uzlu, nebo vytvoření stromového grafu z grafu obecného.

Algoritmus takto dokáže najít všechny dosažitelné vrcholy ze startovacího uzlu *u*. Jsme schopni získat počet všech hran v grafu. Jak je popsáno výše, vytvoří tento algoritmus graf. Nejkratší cesta mezi startovacím uzlem *u* a konečným uzlem *z* je pak rovna hloubce nově vytvořeného stromu. Jedinou nevýhodou je čas potřebný pro vyhledání cesty. Tento algoritmus lze využít na libovolný graf a to jak orientovaný tak neorientovaný. Prohledávání do šířky je navíc základem pro Dijkstrův algoritmus minimální cesty.

2.2.2 Prohledávání do hloubky (Depth-First Search)

Jedná se o velmi podobný typ průchodu jako u prohledávání do šířky. Ovšem při prohledávání do hloubky se nejprve vezme první nalezený soused začínajícího uzlu *u*. Tento soused se označí jako „průchozí“ a následovně se hledá další soused tohoto nalezeného uzlu. Toto probíhá, dokud je možno nalézt nějaký další uzel, který

ještě není označen. Pokud se z tohoto uzlu už nikam jinam nelze dostat, označí se jako „ukončený“ a vrátí se na předchozí uzel. Z tohoto uzlu se opět hledá následník (nejbližší soused). Pokud takový uzel nelze nalézt, vrátí se o úroveň výše. Toto se opakuje tak dlouho, dokud se jako „projitý“ neoznačí startovací uzel u.

U tohoto algoritmu lze také zavést dvě časové proměnné a to čas nalezení uzlu u_n a čas dokončení uzlu u_d. Mimo těchto dvou časových proměnných je nutné si také pamatovat předchůdce daného uzlu. Obdobně jako prohledávání do šířky i zde se při průchodu tvoří strom. Co se týká časových proměnných, musí platit $u_n < u_d$.

Tento algoritmus nalezne, obdobně jako prohledávání do šířky, všechny dosažitelné vrcholy ze startovacího vrcholu u. Pokud po vzniku grafu zjistíme, že zůstaly některé neobjevené uzly, algoritmus spustíme znovu se startovacím uzlem náhodně vybraným z neobjevených uzlů. Díky tomu vzniká les stromů. Obdobně jako u prohledávání do šířky jej lze použít jak na orientované tak neorientované grafy a je také základem některých dalších algoritmů. Časová náročnost závisí na místě, kde se koncový uzel nachází. Pokud bude koncový uzel ve značné hloubce, bude objeven dřív než při algoritmu prohledávajícím do šířky. Pokud však bude blízko startovacího uzlu, může se stát, že se při vyhledávání budou muset projít všechny uzly grafu. Z tohoto důvodu se někdy u tohoto typu prohledávání používá omezení maximální prohledávané hloubky.

Mimo těchto vlastností je také možno vytvořit určitou klasifikaci hran.

Zpětná hrana – Je hrana mezi vrcholy p a q, kde vrchol q je předkem vrcholu p. Neboli vrchol q leží mezi vrcholem startu (kořene) u a vrcholem p. Takováto hrana je tedy hranou zpětnou.

Dopředná hrana – Je hrana mezi vrcholi p a q, kde vrchol p je předkem vrcholu q. Dopředná hrana je tedy opakem hrany Zpětné.

Křižující hrana – Jedná se o libovolnou hranu, spojující vrcholy stromu, pokud jeden z nich není předkem druhého.

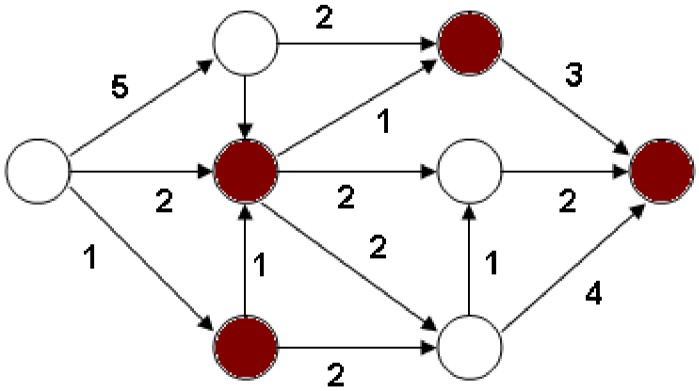
Klasifikace hran hraje v tomto algoritmu podstatnou roly. Například pomocí zpětných hran se dají nalézt cykly v grafu. Existuje-li při prohledávání grafu taková hrana (zpětná hrana), je jisté, že v grafu existuje cyklus.

2.2.3 Moorův algoritmus

Moorův algoritmus nalezne v grafu cestu s nejmenším počtem hran. Tento algoritmus pracuje s neohodnoceným grafem, nebo grafem, který má všechny hrany ohodnoceny stejně. Lze jej tedy také použít v případě, že nás ohodnocení hran vůbec nezajímá. Algoritmus pracuje tak, že startovacímu uzlu *u* přiřadí hodnotu 0. Poté cyklicky vybere další prvek, který je jeho následníkem a přiřadí mu hodnotu rovnou n+1, kde *n* je počet hran „projitých“ od uzlu startovacího k uzlu právě zkoumanému. Výsledkem je pak strom, kde jsou všechny jeho uzly ohodnoceny čísly, které udávají počet hran „projitých“ od uzlu startovacího. [17]

2.2.4 Algoritmus Dijkstra

Jedná se o variantu procházení grafu (obdobně jako do šířky), kdy při každém nalezení následujícího vrcholu využijeme hodnoty získané z hrany pro určení relativní vzdálenosti. Pomocí toho pak získáme délku nejkratšího sledu, kterým se dostaneme do potřebného cíle. Z listu nalezených vrcholů se pak vybere vrchol s nejkratší vzdáleností. Na konci zpracování pak tedy dostáváme posloupnost uzlů, která udává nejkratší cestu mezi daným startovacím a koncovým uzlem. [6]



Obr. 2.2.1 Průchod grafem pomocí Dijkstrova algoritmu

Průchod grafem je pak řízen algoritmem rozděleným do dvou kroků. V prvém kroku se vezme vrchol *u* a podle jeho sousedů se upraví (vypočítají jejich upravené vzdálenosti od prvého uzlu podle délek hran). V každém dalším kroku je vybrán ke zpracování ten uzel, který má ze všech nezpracovaných hran nejkratší vzdálenost

od startovacího uzlu. Dle indukčního předpokladu pak nemůže existovat žádná jiná kratší cesta, která by byla vedena nějakou oklikou. [6]

Z obr. 2.2.1 je patrné, že průchod grafem se bude orientovat nejnižšími hodnotami, proto je tento průchod na obrázku zvýrazněn. Pokud se na tento obrázek ale pozorně podíváme, zjistíme, že průchod grafem lze provést se stejným výsledkem i jinou cestou a to dokonce s menším počtem průchodů uzly. Tento příklad pak ukazuje problém Dijkstrova algoritmu. První nalezená nejkratší cesta nemusí být nutně ta skutečně nejkratší. Proto se Dijkstrův algoritmus musí volat znova a průchod grafem opakovat.

Počet potřebných kroků pro průchod algoritmem ze startovacího bodu do bodu konečného přes N uzlů je přibližně N^2. Pokud se však uzly určitým způsobem předzpracují, může být počet průchodů algoritmu menší. Pokud naopak necháme projet celý graf tímto algoritmem, získáme nejkratší cestu do všech uzlů v celé orientované mapě. Cesta průchodu je poté uložena do seznamu pozpátku, to znamená, že poslední nalezený uzel je zapsán jako první. [6,17]

2.2.5 Algoritmus A*

Algoritmus A*, jak již bylo napsáno v úvodu této kapitoly, vychází z algoritmu Dijkstra. Algoritmus A* předzpracovává informace o vzdálenosti tak, že nejprve vytvoří tzv. potenciál $p_v(x)$, jenž udává libovolný dolní odhad vzdálenosti ze startovacího vrcholu do vrcholu konečného. Pokud se jedná o neorientovaný graf je zorientován. „Pokud máme orientovaný graf pokračujeme v provádění grafu tak, že každá hrana h dostane nové ohodnocení určené rovnicí:

$$w'(h_S h_K) = w(h_S h_K) + p_V(h_K) - p_V(h_S) \qquad (1)$$

Kde h_S je začínající uzel hrany a h_K je koncový uzel hrany. Hodnota potenciálu musí splňovat podmínku:

$$w(h_S h_K) \geq p_V(h_S) - p_V(h_K) \qquad (2)$$

Z toho plyne podmínka, že hodnota ohodnocení se nesmí dostat do záporných hodnot. Upravená délka libovolného sledu S z počátečního uzlu do koncového je pak:

$$d_G^{w'}(S) = d_G^{w}(S) + p_V(h_K) - p_V(h_S) \qquad (3)$$

Což je konstantní rozdíl oproti původní délce S. Takže S je optimální pro původní délkové ohodnocení *w*, právě když je optimální pro nové *w´*.“[1] [6]

2.2.6 Fordův algoritmus

Tento algoritmus dokáže, na rozdíl od Dijkstrova nebo A* algoritmu pracovat i se záporně ohodnocenými grafy. V takto ohodnocených grafech se však nesmějí objevit záporně ohodnocené kružnice. Díky takové kružnici, by mohla cesta nabývat záporného nekonečna.

„Každý uzel grafu je označen dvojicí čísel (c,d): první číslo udává počet cyklů chodu algoritmu, ve kterém byla k danému uzlu nalezena zatím nejkratší cesta, druhé pak udává vlastní velikost doposud nalezené cesty. Na začátku je startovací uzel označen (0,0), ostatní uzly (0,∞). Algoritmus cyklicky vybere všechny uzly, jejichž hodnota odpovídá pořadovému číslu právě probíhajícího cyklu.“[2] Tyto cykly jsou takové, pro něž byla v předchozím cyklu nalezena lepší cesta. Vrchol, u kterého by se tak zlepšila cesta, by logicky ovlivnil i jeho následovníky, proto je potřeba zkontrolovat i ty. Pokud se toto tvrzení, při průchodu algoritmu, potvrdí, přiřadí se následovníkovy hodnota (c_k+1, d_k+1), kde c_k a d_k je ohodnocení předchůdce. Po ukončení algoritmu je nejkratší cesta k danému uzlu v druhé souřadnici. Tato souřadnice opět přiřazuje pouze počet projitých uzlů. [17]

2.2.7 Ford – Fulkersonův algoritmus

Tento algoritmus může pracovat pouze na orientovaném neohodnoceném grafu, respektive na takovém grafu, na kterém je možno využít Moorův algoritmus. Ford- Fulkersonův algoritmus využívá Moorův algoritmus pro předzpracování grafu tak, že hodnotící veličinu tok nastaví na celém grafu rovnou nule. Následně se na tento graf zavolá Moorův algoritmus, který nalezne nejkratší cestu mezi startovním

---

[1] Hliněný, P. : Základy teorie grafů, Brno, 2010, str. 33

[2] Juránek, V. : Algoritmy v teorii grafů, Brno, 2009, str. 10

uzlem *u* a konečným uzlem *z*. Na této cestě zjistí rezervu toku a o tuto hodnotu jej zvýší. Moorův algoritmus pak s touto cestou již nepracuje, hrany které mají jinou hodnotu než všechny ostatní nevidí. Algoritmus tak končí, až je celý graf ohodnocen maximálním tokem.

2.3 MULTIAGENTNÍ SYSTÉMI V DOPRAVĚ

Jiný pohled na dopravní systémy nabízí Multiagentní systémy. Multiagentní systémy využívají pro svou práci informace z nejrůznějších oborů lidského vědění. V multiagentních systémech lze nalézt informace z oborů Biologie, Chemie, Fyziky, Sociologie atd.. Multiagentní systémy se s velmi dobrými výsledky užívají při simulaci chování živočichů, sociálních skupin nebo dopravy.

Každý agent v multiagentním systému představuje určitou entitu, která má blíže nespecifikovaný úkol. V dopravním světě tak například může být agentem jeden automobil směřující do nějakého cíle (pokud se jedná o makroskopický model), ale i jedno kolo vozidla, blinkr nebo jiná funkční část mající nějakou předem danou úlohu. Agent v takovém systému může být navíc řízený a to jak svými okolními partnerskými agenty (v tomto případě mluvíme spíše o ovlivňování), tak agentem stojícím výše v žebříčku rozhodování tzv. dominantním agentem. Agenti mezi sebou využívají základních principů komunikace, spolupráce i soupeření. Agenty můžeme rozdělit na hardwarové a softwarové. Představitelem hardwarových agentů jsou roboti, softwarovými pak formy umělé inteligence.

2.3.1 Reaktivní agent

Jedná se o nejjednodušší typ agenta. Tito agenti často tvoří jednotlivé vrstvy složitějších agentů. Jsou tvořeni několika paralelními procesy, spouštějícími se na základě podnětů z prostředí, nebo z agentova vnitřního stavu popř. z obojího. Reaktivní agenti nedokážou plánovat následující akce (z toho vyplývá, že nemají mít zadán cíl) a neobsahují paměť. Získané informace můžou díky komunikaci předávat centralizovanému systému. Z toho plyne, že tento agent je prakticky vždy využíván v centralizovaných systémech. Navíc si nedokáže uvědomovat přítomnost ostatních

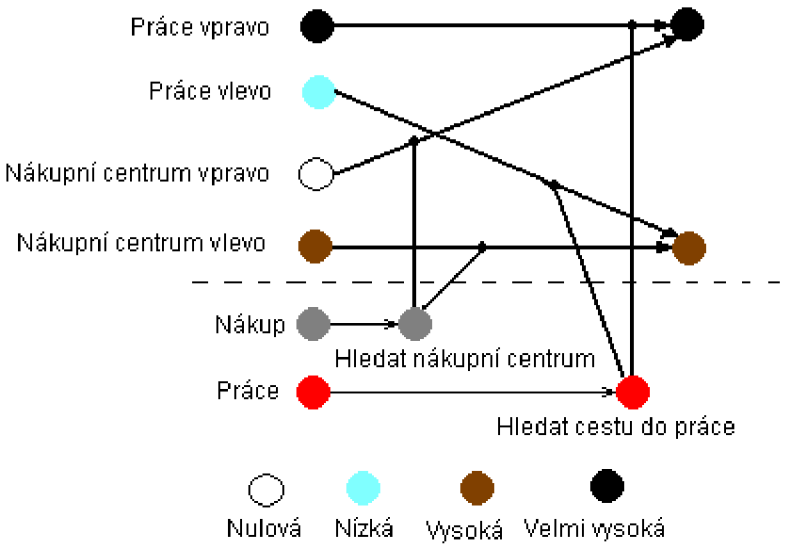
agentů ve svém okolí. Vnímá je pouze jako dynamické změny prostředí. Níže jsou popsány některé typy reaktivních agentů. [8,9]

Architektura s výběrem akce

Tento agent je tvořen z několika paralelních modulů chování. Tyto moduly mezi sebou postupně bojují (soutěží) o ovládnutí vlastního agenta. Vítěze vyhodnocuje modul výběru akcí. Při deaktivaci některého z modulů v důsledku vítězství jiného, jej vítězný modul úplně nevypne. V každém okamžiku může být aktivní pouze jeden modul.

Wernerův reaktivní model

Na rozdíl od předchozí architektury umožňuje Wernerův model vykonávat několik akcí nezávisle na sobě. Jedná se o kombinaci neuronové sítě s architekturou výběru akce. Mimo informací z okolí je důležitá i „interní motivace“. Interní motivace jsou informace získávané z ostatních částí agenta. Tyto informace nejlépe vystihuje obrázek 2.3.1



Obr. 2.3.1 Schéma Wernerova modelu

Jednotlivé barevné kolečka představují přibližnou potřebu agenta pro zisk daného cílu. Do pravé strany grafu se převedou nejvyšší hodnoty z levé strany pro daný směr. Tyto hodnoty se pak porovnají a výsledkem je směr s celkovou vyšší hodnotou. Pokud jsou obě nejvyšší hodnoty stejné, porovnávají se druhé nejvyšší hodnoty. Tímto principem se postupuje, dokud se nedojde k výsledku.

2.3.2 Deliberativní agent

Jedná se o agenta, jenž dokáže stejně jako reaktivní agent vyhodnocovat informace z okolí, ale na rozdíl od něj dokáže tyto informace využít k tzv. plánování. V praxi to znamená, že tento typ agenta si uvědomuje své okolí a může díky mapě okolí (nebo jiným principem) získávat informace o okolí. Jedná se o druhý vývojový stupeň agentů. Stále však nejsou schopni přímo komunikovat s ostatními agenty a berou je spíše jako cizí objekty v jejich prostředí. Díky tomu jsou tyto systémy závislé na centralizovaném řízení. Dle zadaného cíle a těchto informací si agent sám dokáže navrhnout strategii k vyřešení daného problému. [8,9]

Uvažující agent na bázi reaktivity

Tento typ agenta je na rozhraní mezi Reaktivním a Deliberativním agentem. Takový agent sice nedokáže plánovat svoje další kroky, ale je si vědom okolí a dokáže pomocí značek vytvářet mapu prostředí. Mapa tvořená pomocí značek není příliš dokonalá, v případě že dojde k posunu (změně okolí). Agent je schopen tuto změnu zaznamenat a upravit ji ve své mapě. [8]

IRMA

IRMA (angl. Inteligent ressource-bounded machina architecture – architektura inteligentního zařízení s omezenými zdroji). Tato architektura „obsahuje datové entity, které symbolizují agentovy poznatky o světě, tužby, záměry a knihovnu plánů. Agentovy tužby jednoduše nemusí být shodné s jeho schopnostmi ani zdroji. Některé z tužeb se stanou cíli na základě procesu výběru dle aktuálního

stavu prostředí.“[3] Podle tohoto procesu deliberace je pojmenovaná celá tato skupina agentů. Architektura IRMA dokonale kopíruje rozhodování člověka.

V knihovně plánů jsou uloženy základní poznatky o dílčích úlohách, plánech a podmínkách opakovatelnosti. Agent si vybere jednu z úloh a dle mechanizmu cílů a prostředků se rozhodne, zda může daný úkol splnit v závislosti na svých schopnostech a zdrojích. Tento typ agenta se samozřejmě dokáže vypořádat se změnami prostředí. Nevýhodou modelu je pak nedokonalost jeho modelů chování a čas potřebný k výpočtu. Systém by měl napodobovat složité mentální pochody, které díky svým jednoduchým operátorům není schopen simulovat. [8,9]

2.3.3 Sociální Agent

Jedná se o agenty komunikující s ostatními agenty ve svém okolí pomocí vyšších jazyků, což je základním rozlišovacím prvkem sociálního od deliberativního agenta. Díky tomu se mohou vytvářet sociální struktury bez vnějšího přičinění. Sociální agenti se prakticky vždy používají jako decentralizované systémy. Agenti si navzájem vypomáhají z důvodu minimalizace jejich omezeních. Jejich schopnosti jsou více strukturalizované (jeden umí nakládat, druhý vozit), atd. [8,9]

Grate*

Tato architektura se velmi podobá architektuře IRMA a to zejména v důsledku vybudování podle teorie BDI. Rozdílem mezi IRMA a GRATE* je rozšíření o sociální model. Tento model při zjištění problému kontaktuje možné partnery zainteresované ve společném řešení problému. Dle odpovědí si agent vybere nejlepšího partnera, se kterým za určitou cenu začne spolupracovat. [8]

2.3.4 Hybridní agent

Hybridní agent je takový agent, který má vnitřní strukturu tvořenou komponenty reaktivního, deliberativního i sociálního agenta. Takový agent komunikuje ve vrstvách. Díky tomu lze hybridní agenty rozdělit na agenty

[3] Kubík, A.: Inteligenti agenty, Brno,2004, str.55

s vertikální strukturou a agenty s horizontální strukturou. Oba systémy potřebují určitý druh řízení vnitřní komunikace mezi vrstvami, což sebou přináší určité problémy. Vertikální vrstvení se vyznačuje tím, že se senzory a s akčními částmi systému je spojena pouze jedna vrstva (nejnižší). Díky tomu dochází k nárůstu komunikace mezi jednotlivými vrstvami. Nižší vrstva vždy vysílá dotaz na vyšší a ta pak pomocí komunikace organizuje vrstvy nižší.

Horizontální architektura musí být řízena řídícím mechanizmem, aby bylo zajištěno racionální chování a agent nebyl ochromen bojem jednotlivých vrstev o vstupy a akční výstupy. [8,9]

2.3.5 Interakce agenta a systému

Komunikace mezi agentem a systémem nebo mezi několika agenty funguje na bázi sociálních interakcí. Tyto interakce lze rozdělit na Koordinaci, Reaktivní komunikaci, Emergenci, Kooperaci a Komunikaci. Faktem je to, že se tyto pojmy mezi sebou často velmi prolínají a nalézt tenké hranice je velmi problematické.

Koordinace

Jedná se o velmi důležitý prvek chování v multiagentních systémech, kdy pomocí komunikace dochází k plnění úkolů. Koordinace se dělí na centralizovanou (kdy agent s vyšším postavením určuje role agentů s nižším postavením) a decentralizovanou (určování rolí agentů se děje prostřednictvím protokolů, podobně jako dopravní pravidla, nebo vzájemnou dohodou). Systémy napodobují ekonomické a biologické pochody. Koordinace se dělí na tři základní typy.

Koordinace vzájemnou dohodou – Dochází k ní když se minimálně dva agenti pomocí komunikace dohodnou o vzájemném využívání svých zdrojů za účelem dosažení společného cíle. Při tomto typu koordinace nemá žádný z agentů podřízené nebo nadřízené postavení a ani žádný z agentů nemá vůči ostatním centrální postavení.

Koordinace přímým dozorem - Tento systém je nazván podle jednoho agenta, který centrálně řídí celou společnost. Tento nadřízený agent zadává podřízeným jejich jednotlivé dílčí úkoly. Za nesplnění těchto úkolů je může penalizovat. Systém může být navržen s jednou vrstvou (je pouze jeden nadřazený a pod ním jsou všichni na stejné úrovni) nebo vypracovaný na principu stromu.

Koordinace standardizace - Jedná se o systémy řízené nějakým předpisem nebo nařízením. Porušování těchto předpisů je pro celý systém nevýhodné, nemusí to ovšem platit pro samotného agenta, který tento předpis poruší. Výborným příkladem mohou být pravidla silničního provozu. Za porušení může, ale nemusí být agent penalizován.

Reaktivní komunikace (stigmergie)

Tato komunikace „je inspirovaná komunikací některých druhů společenství hmyzu prostřednictvím chemických stop zanechávaných v prostředí.“[4] Agenti pak využívají jednotlivých typů značek. Tento typ komunikace využívají především reaktivní agenti. Agent projde prostředím, zanese do mapy prostředí určité body a tuto mapu pak rozešle ostatním agentům. Ti se při průchodu orientují dle mapy a případně ji opravují pokud by došlo ke změně prostředí.

Tato komunikace definuje čtyři základní definice. Dvě z nich se vztahují k samotným agentům. Jedná se o funkci vytvářet značky a funkci interpretovat značky. Dále pak definuje značky s časovou platností a značky s trvalou platností.

Výhoda této komunikace se skrývá v její jednoduchosti. Agenti o sobě většinou nemusejí ani vědět a i bez prvotní vstupní informace reprezentující mapu prostředí. Agenti jsou schopni poměrně přesně tuto mapu vytvořit. Velkou výhodou je vysoká robustnost systému (vyřazení jednoho agenta nemusí nutně ohrozit celou skupinu a prakticky nemusí ani snížit výkon skupiny). Díky tomu lze vytvářet otevřené systémy.

---

[4] Kubík, A.: Inteligenti agenty, Brno,2004, str.80

Nevýhodou je jejich počáteční ztracenost, kdy se pohybují v prostředí zcela náhodně a snaží se zachytit významných objektů. Další nevýhodou může být značný počet agentů. Díky decentralizaci může docházet k špatné organizovanosti. [8]

Aukce

Jedná se o „druhý základní typ koordinačního mechanismu, který není považován za kooperaci.“[5] Myšlenka aukce vychází především z myšlenky omezených zdrojů. Aukce je přebrána z ekonomiky. Každý agent při svém vzniku získá určitý finanční obnos. Aukce pak má za úkol najít rovnovážnou cenu za dané zdroje. Existuje několik druhů aukcí. Každá z nich je specifickým rozhodovacím algoritmem. Zde jsou popsány některé z daných typů aukcí. [8]

Anglická aukce – Tato aukce je jednostrannou[6] aukcí. Jedná se o finančně nejrozšířenější typ aukce. Je stanovena minimální cena za kterou lze zdroj odkoupit. Každý agent může přihazovat libovolnou sumu. Přihazovaná suma nesmí logicky přesáhnout jeho majetek. Každý agent účastnící se aukce zná nabídku svých protivníků. Aukci vyhrává ten agent, který nabídne nejvyšší sumu za daný zdroj.

Holandská aukce – Jedná se také o jednostrannou aukci i když v opačném směru. V tuto chvíli neurčují danou cenu agenti, ale řídící aukcionář. Ten nastaví cenu za daný zdroj na umělé hodnotě. Tuto hodnotu postupně snižuje. Aukci vyhrává ten agent, který jako první nabídne aukcionářem stanovenou cenu.

Aukce „první nabídka s nejvyšší cenou“ – Jedná se o jednostrannou aukci, kdy jednotliví agenti účastnící se aukce nevědí o nabídkách svých protivníků a každý z nich může říct cenu pouze jednou.

---

[5] Kubík, A.: Inteligenti agenty, Brno,2004, str.81

[6] Jednostranná aukce je taková aukce, kdy s danou cenou za zdroj manipuluje pouze jedna strana účastnící se aukce.

Vickreyova aukce – Opět se jedná o jednostrannou aukci. Princip je velmi podobný jako u předchozí aukce. Žádný z agentů, který se této aukce účastní, nemá informace o částkách nabídnutých soupeři. Každý z agentů může svoji cenu nabídnout pouze jednou. Vítězí agent nabízející nejvyšší částku, avšak zaplatí cenu nabízenou agentem na druhém místě.

Burzovní aukce – Jedná se o oboustrannou aukci. V jednu chvíli nabízí své zdroje více agentů čímž logicky klesá jejich cena. Nakupující agenti se snaží nalézt druh zdrojů, kde nabídka převyšuje poptávku. Prodávající agenti se logicky snaží prodávat takové zdroje, kterých je nedostatek.

Emergence

Jedná se o chování, které se projevuje zejména u hybridních agentů. Komunikace a reakce vnitřních vrstev agenta nejsou v celkovém obrazu agenta patrné. Obecně se dá říct, že tento pojem v umělé inteligenci není stále ještě přesně definován.

Kooperace

„Kooperace je řízenou formu koordinace s účelovým uspořádáním agentů za účelem dosažení společného řešení daného problému nebo konfliktu.“[7] Kooperaci lze spatřit převážně u decentralizovaných systémů. Dochází k tomu, že se v takovém systému rokuje a vyjednává o společném řešení daného problému nebo konfliktu. Každý agent má zadanou svou roli a vztahy s ostatními agenty. Díky těmto informacím lze dosáhnout zadaného globálního cíle. Tyto informace jsou předávány v tzv.protokolu o kooperaci. V tomto protokolu jsou také zapsány informace o tom, co přesně mají obsahovat dotazové a odpovídající zprávy. „Kooperace je vyšší formou koordinace, protože jí nelze bez ní dosáhnout.“[8] Tento druh interakce se

[7] Kubík, A.: Inteligenti agenty, Brno,2004, str.77

[8] Kubík, A.: Inteligenti agenty, Brno,2004, str.77

v menším měřítku používá i v centralizovaných systémech, je patrný zejména v tabulkové architektuře a kontrakční síti. [8]

Komunikace

Sociální agenty vyžadují vyšší komunikační jazyk. Snahou je, aby tyto jazyky byly založeny na teorii komunikačních aktů. Obecně lze říct, že každá informace přenášena pomocí tohoto jazyka v sobě nese i předpoklad změny informačního stavu přijímacího subjektu.

2.4 TEORIE SIMULACE DOPRAVY

Teorie zabývající se dopravou je složitým vědním odvětvím. Teorie dopravy se dělí na dvě hlavní části a to makroskopickou simulaci a mikroskopickou simulaci. V makroskopické simulaci se na dopravu nahlíží jako na tok informací. S těmito informacemi se dále pracuje a získávají se z nich potřebná data jako plynulost dopravy, počet vozidel, čas strávený na silnici atd. Makroskopické modely se například používají pro zkoumání funkčnosti městských okruhů.

Při práci na vytváření mikroskopického modelu musíme pracovat s fyzikálními modely automobilů a vytvářet vlastního řidiče. Každý takový řidič má své unikátní vlastnosti jako reakční dobu atd.

2.4.1 Makroskopický model dopravy

Při práci s tímto modelem jsou používány vlastnosti známé z teorie toků. Obdobně se tak při této teorii používají tři základní veličiny a to:

Počet automobilů na jednotkovém úseku cesty neboli hustota toku (density of the flow) označená také písmenem $\rho(x,t)$. Počet automobilů, které projedou měřeným bodem za jednotku času, průtoková rychlost $q(x,t)$ (rate of flow) a rychlost toku neboli rychlost jednotlivých automobilů $v(x,t)$ (speed of the flow). Pro sepsání vzorců, které jsou potřeba k práci s jednotlivými veličinami je potřeba nejprve určit další potřebné veličiny. Nejprve si určíme $\Delta x(x,t)$ což je vzdálenost mezi bodem x a bodem $x_2 = x + \Delta x$. Počet vozidel v tomto prostoru je pak dán $\Delta N(x,t)$.

Průtoková rychlost *q(x,t)* je pak rovna:

$$q(x,t) = \frac{\Delta N(x,t)}{\Delta t} \quad [s^{-1}] \qquad (4)$$

Z jednotek je patrné, že se ve skutečnosti o rychlost nejedná, proto budeme tuto jednotku dále označovat pouze za průtok.

Počet vozidel *ΔN(x,t)* je také možno získat rozdílem automobilů do měřeného místa vjíždějících ku počtu aut z místa vyjíždějícího neboli:

$$\Delta N(x,t) = N(x,t) - \Delta N(x+\Delta x,t) \quad [-] \qquad (5)$$

Rychlost vozidel neboli rychlost toku *v(x,t)* lze získat vzorcem:

$$v(x,t) = \frac{\Delta x}{\Delta t} \qquad [ms^{-1}] \qquad (6)$$

Hustota toku je pak logicky rovna:

$$\rho(x,t) = \frac{\Delta N}{\Delta x} \qquad [m^{-1}] \qquad (7)$$

V některých literaturách se hustota označuje také jako *k*, proto bude toto druhé označení použito zejména u grafů získaných z literatury.

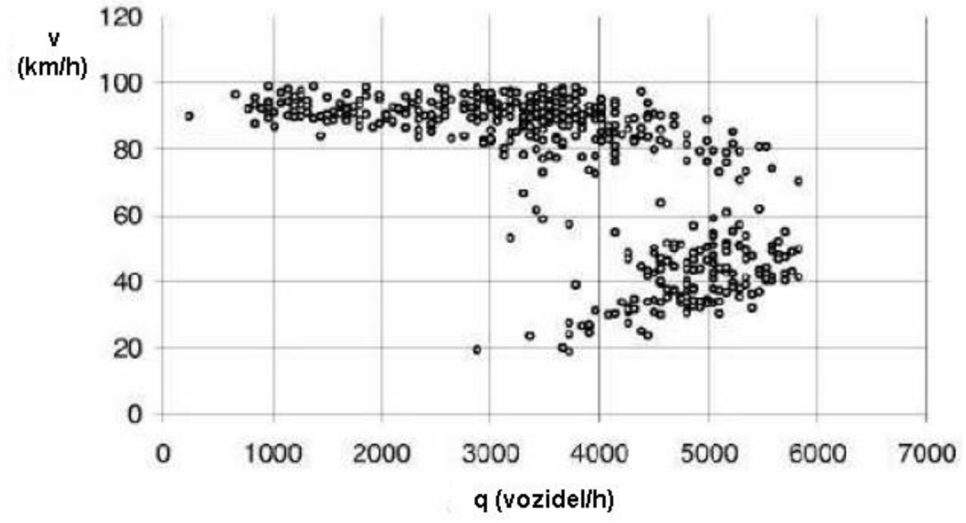
Ze vzorců 4, 6 a 7 je patrná vzájemná závislost těchto veličin. Výpočet průtoku *q(x,t)* je tedy možno získat také:

$$q(x,t) = v(x,t) * \rho(x,t) \qquad (8)$$

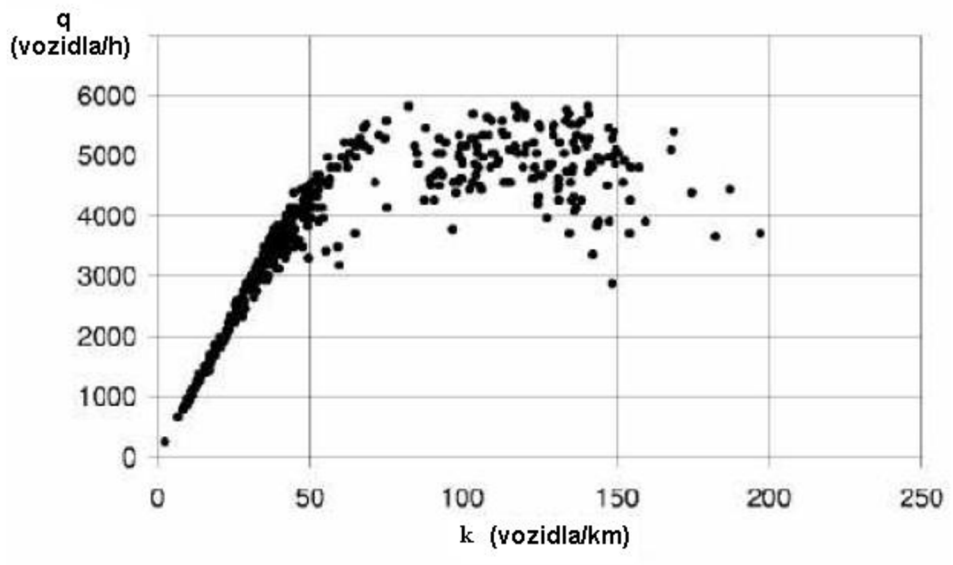
Při měření hustoty toku *ρ(x,t)* je nutno volit *Δx(x,t)*. Volba této veličiny je velmi důležitá a záleží na ní obraznost toku. Pokud bude příliš velká bude docházet k průměrování hustoty. Toto by se mohlo například stát v případě, že by se velikost zvolila nevhodně a byla by v ní například hrana společně s několika křižovatkovými uzly. Pak by se v měřeném směru vozidla „ztrácela“ na křižovatkách a naopak by se na nich některá jiná znovu „objevovala“. Výsledkem by pak mohla být hustota, která se ve skutečnosti na dané hraně vůbec neobjevuje. Pokud bude měřená vzdálenost příliš malá, nebude mít měření žádný význam. Respektive pokud budeme na dálnici měřit hustotu na vzdálenosti 5m, nebudeme mít v jednom směru nikdy více vozidel než jedno v každém pruhu. [18,19]

Máme-li tedy tři základní veličiny hustotu, rychlost a průtok, můžeme s jejich pomocí vytvořit tři základní grafy rozložení závislostí: hustota ku rychlosti (*ρ(x,t)*,

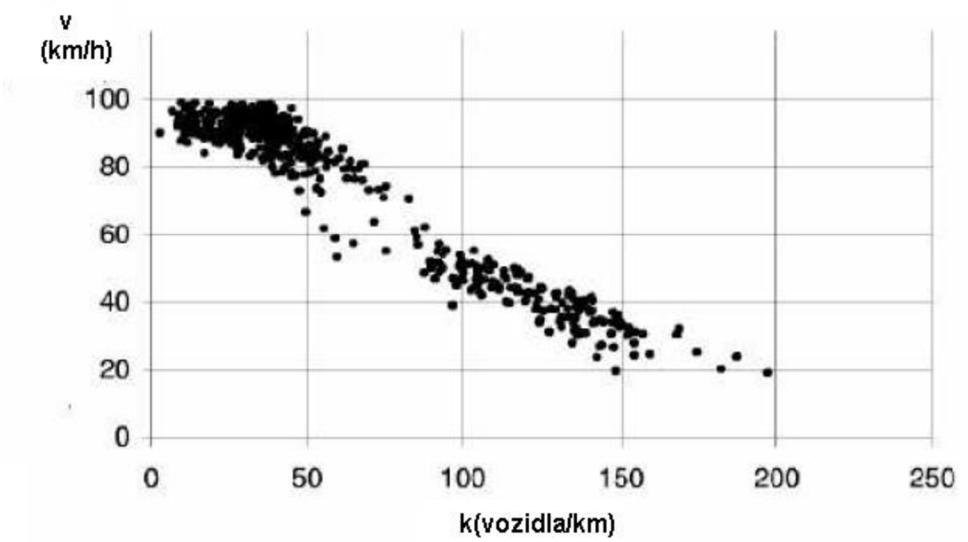
v(x,t)), hustota ku průtoku (*ρ(x,t), q(x,t)*) a průtoku ku rychlosti (*q(x,t), v(x,t)*). Všechny tyto grafy jsou znázorněny níže.



Obr. 2.4.1 Závislost rychlosti na počtu vozidel [20]

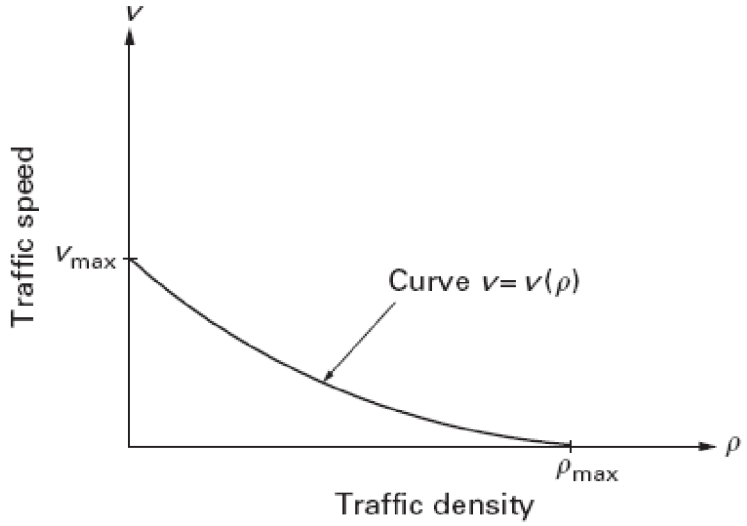


Obr. 2.4.2. Závislost průtoku na hustotě [20]



Obr. 2.4.3 Závislost rychlosti na hustotě [20]

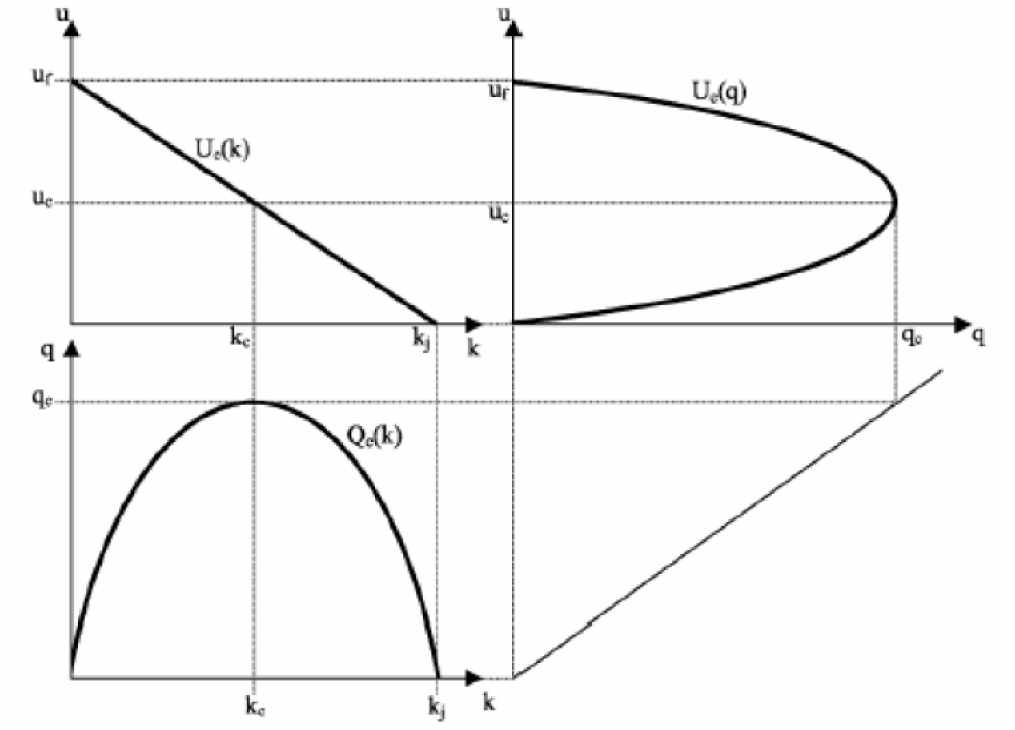
Tyto výsledky byl získány při testu popsaném v Traffic Flow Theory, *Basics of Traffic Engineering*. Při porovnání výsledků testů s teorii popsanou jak ve výše napsaném díle, tak v publikaci Traffic Flow Models, *Principles of mathematical modeling*, můžeme pomocí proložení daných grafů získat obdobné grafy jako na obr. 2.4.4.



Obr. 2.4.4 Závislost rychlosti na hustotě [19]

Díky takto získaným grafům můžeme následně model porovnávat s některým z matematických modelů. Nejčastěji využívané modely dopravní simulace jsou Greenshield a Trojúhelníkový. Matematické modely se využívají jako zjednodušení skutečné situace.

Greenshieldův matematický model



Obr. 2.4.5 Matematický model Greenshield [20]

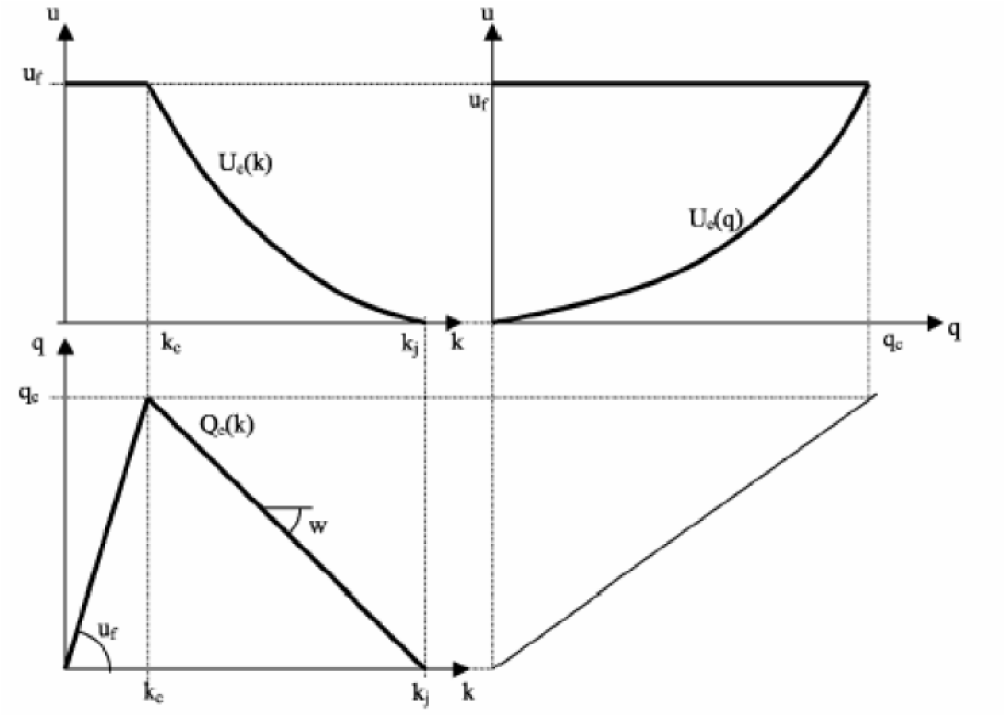
Každý z modelů zjednodušuje některý ze tří průběhů tak, aby jejich průběh byl triviální. U modelu Greenshield je tímto průběhem závislost hustoty na rychlosti, kde k_j je maximální hustota a u_f maximální rychlost. Obě ostatní závislosti mají poté parabolický průběh. Mimo u_f a k_j jsou zde popsány ještě u_c a k_c. Obě sou závislé na toku q a při těchto hodnotách je tok maximální. U_c a k_c jsou rovny polovině maximálních rozsahů daných veličin. Tento model byl vytvořen roku 1934 a je pro svou jednoduchost dodnes velmi využíván. Zbývající dva grafy lze pak popsat pomocí rovnic:

$$v = V_e\left(\frac{u_f}{k_j} * \left(k_j - k\right)\right) \qquad (9)$$

$$q = Q_e(k) = \frac{u_f}{k_j} * k(k_j - k) \tag{10}$$

[19,20]

Trojúhelníkový matematický model



Obr. 2.4.6 Trojúhelníkový matematický model [20]

Trojúhelníkový model vychází z předpokladu lineárního zvyšování (a snižování) hustoty na toku. Hlavní výhodou tohoto modelu je reálná závislost hustoty na rychlosti, kdy má maximální rychlost vozidel konstantní hodnotu do té doby, dokud se nepřesáhne kritická hustota vozidel k_c. Tato vlastnost trojúhelníkového modelu má výrazné výhody při dynamickém modelování dopravy a byl použit při vytváření simulačního prostředí jako výchozí matematický model.

2.4.2 Mikroskopické modely

Mikroskopické modely se dají rozdělit do tří základních typů. Prvním typem jsou modely zabývající se reakcí vozidla na překážku před sebou. Mimo stojících překážek se jedná také o vozidla jedoucí ve stejném pruhu (anglicky se tento typ

nazývá Car-following model).Tento model jako jediný je implementován do našeho systému, proto mu bude věnována většina této kapitoly. [19, 20]

Mimo Car-following modelu existují i další, jako model určený pro přejíždění mezi pruhy ve stejném směru (Lane-Change model) a model hledající nejkratší cestu v dopravním systému (Route-Choise model). Model hledající nejkratší cestu je, v našem případě tvořen algoritmem v této práci již popsaným, proto se zde o něm nezmíníme.

Při testování dopravních modelů je nejčastěji využíváno Car-following modelu. Pro možnost zkoumání dvou automobilů jedoucích za sebou je třeba nejprve popsat několik veličin.

x_2 – Okamžitá pozice zkoumaného vozidla

x_1 – Okamžitá pozice vozidla jedoucího před námi zkoumaným

L – Délka daného automobilu

t – Čas za který urazí měřené auto jedna vzdálenost k autu dvě.

d – Vzdálenost mezi x_2 a x_1 (nejedná se o volný prostor mezi automobily, ale vzdálenost přední „masky“ automobilu k přední „masce“ automobilu druhého. Pokud chceme získat volný prostor mezi těmito auty, musíme od vzdálenosti d odečíst délku automobilu L_1. Tato veličina se může potom označit jako s.

Rychlost automobilu: $$v(t) = \frac{\Delta x}{\Delta t} \tag{11}$$

Zrychlení se vypočítá obdobně jako: $$v(t) = \frac{\Delta^2 x}{\Delta t^2} \tag{12}$$

Tento model je většinou funkcí zrychlení na daných podmětech řidiče, jako jsou vzdálenost od překážky, reakční doba řidiče, aktuální stav vozovky atd.

3. ŘEŠENÍ PROBLEMATIKY

K vytvoření simulačního prostředí bylo užito programovacího prostředí NetBeans pracující s jazykem Java . Samotný program je pak společně s popisky uložen v příloze práce. Simulační prostředí pracuje pouze v příkazovém řádku.

Při práci bylo nutno řešit problém jak mikroskopické konstrukce samostatného pohybu automobilů, tak řešení toků z makroskopického hlediska. Teoreticky by bylo možno obě části práce od sebe oddělit, vznikly by tak dva samostatné simulátory. Jeden by pak mohl řešit pouze průjezd křižovatkou a druhý by řešil samostatný pohyb automobilu po mapě. Tento simulátor je určitým kompromisem mezi oběma možnostmi. Kompromisem jsou vlastnosti popsané v úvodních částech 3.1 a 3.3, které nebyli implementovány v důsledku zjednodušení.

3.1 MIKROSKOPICKÉ ŘEŠENÍ POHYBU AUTOMOBILŮ

Mikroskopický pohyb automobilů je postaven na silničních předpisech. Z tohoto důvodu nelze v simulátoru mimo maximální povolenou rychlost porušovat žádné jiné dopravní předpisy. Toto omezení pomáhá snížit náročnost vytvoření simulátoru a tím se např. při zařazování aut do jednotlivých pruhů daný automobil zařadí při příjezdu ke křižovatce tak, aby nepřejel přes plnou čáru. Překročení maximální rychlosti je stabilně nastaveno na 10%. Další omezení simulátoru spočívá v nemožnosti předjíždění automobilů. Toto omezení pak vychází z vlastností toků a samotné konstrukce simulačního prostředí. Další zjednodušení je v důsledku průjezdu křižovatkou. Křižovatková hrana (*crossRoadEdge)* má totiž nulovou velikost. Toto zjednodušení bylo vytvořeno z důvodu rychlejší simulace. Obecně však i tyto hrany mohou mít libovolnou velikost. Při takové velikosti však můžou v křižovatkách zůstávat auta, která se na navazující hranu již nevejdou a tak mohou zapříčinit kolaps celého dopravního systému.

3.1.1 Průjezd křižovatkou

Křižovatky je nutno dělit do čtyř základních skupin. Světelnou křižovatku, křižovatku s jednou hlavní silnicí, kruhovou křižovatku a křižovatku s předností zprava. Každá křižovatka je tvořena z několika poduzlů. Každý poduzel má pak svoji specifickou činnost (viz. popis světelné křižovatky). Nejdůležitějším uzlem je výstupní uzel, který představuje konec křižovatky. Tento uzel je vytvořen z důvodu lepší kontroly průjezdnosti dané křižovatky (pomocí součtu vozidel, které projedou všemi výstupními uzly dané křižovatky lze určit zatížení dané křižovatky) a slouží také pro specifikaci tzv. křižovatkových hran (*crossRoadEdge*). Tyto hrany tvoří společně s poduzly vnitřní strukturu křižovatky. Obecně lze říci, že pomocí křižovatkových hran auto zjišťuje, zda může pokračovat plynule v jízdě, nebo musí zastavit a čeká na možnost dalšího pokračování v jízdě. U křižovatkových hran je také možno měnit maximální rychlost, kterou lze danou hranu projet. Díky tomu lze následně docílit snížení rychlosti před některými křižovatkami a simulovat tak, např. nepřehlednou křižovatku.

Světelná křižovatka

Světelná křižovatka se od ostatních odlišuje zejména tím, že každá taková křižovatka mimo svého vnitřního navržení obsahuje i informaci o semaforu implementovanou do uzlu představujícího samotnou křižovatku. Úskalí tohoto typu křižovatky je v tom, že světelná křižovatka může mít libovolný počet přívodních pruhů, které se mohou v libovolném pořadí pomocí světelné signalizace pouštět. Díky zjednodušení dodržování pravidel silničního provozu lze každý takový pruh (respektive směr) simulovat jednou samostatnou hranou.

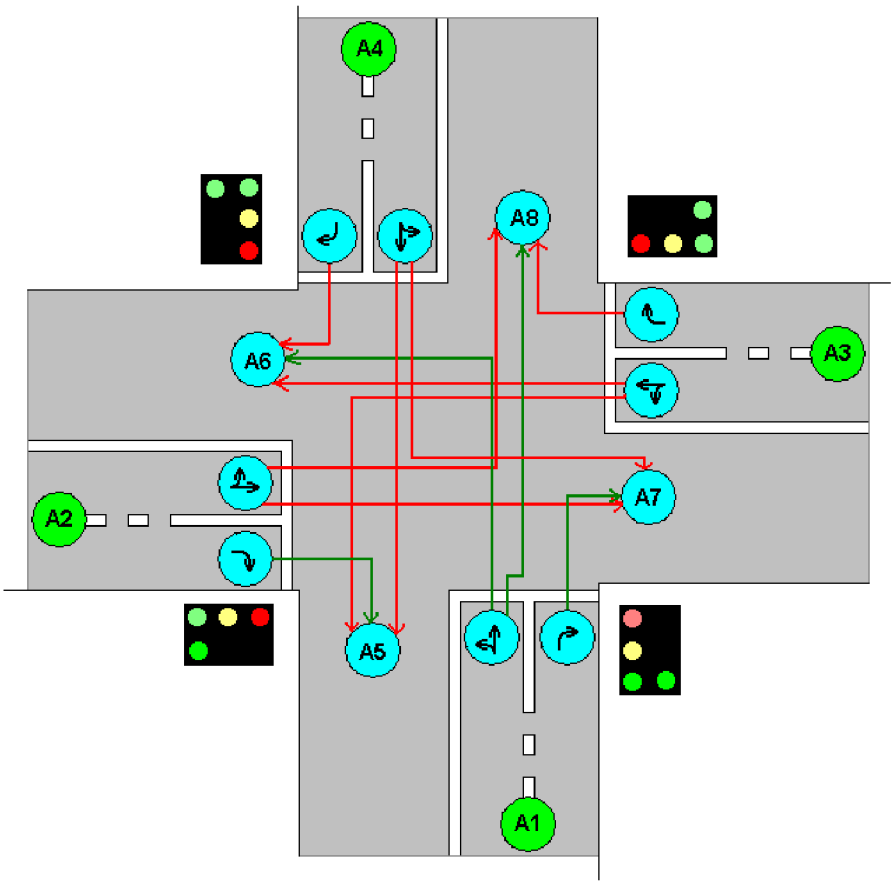
Počet pruhů se pak přenáší pomocí proměnné *RoadCapacity*. Tato vlastnost může nabývat pouze celých čísel. A každá hodnota této proměnné prezentuje v celkovém schématu jak poduzly tak i hrany tvořící jednotlivé části silnice.

Semafory jsou tvořeny pomocí dvou časovačů, kde první ovládá zelenou a druhý ovládá červenou. Další důležitou nastavovanou vlastností semaforu je čas určený k prvému překlopení semaforu. Nebýt této hodnoty, nemohlo by dojít k nastavení semaforu.

Nastavení semaforů a jeho hlídání se věnují speciální programy, které by svojí složitostí vydaly na novou diplomovou práci. Z tohoto důvodu jsem nechal kontrolu nad semafory na rozvaze uživatele. Nastavení semaforu jsem pak při testech provedl nejjednodušeji, jak jen to bylo možno, což je patrno v tabulce 3.1.1, kde Zelená představuje puštěný směr, Červená pak směr stojící. Žlutá se nikde nevyskytuje z toho důvodu, že není implementována a při měření reálné křižovatky byl čas určen pro žlutou započítáván rovnoměrně do obou barev světel. Časové sekvence 10s jsou v tabulce určeny jen pro přehlednost a lze je nastavit libovolně.

Tab. 3.1.1 Standardní nastavení semaforu

	Čas [s]						
	0	10	20	30	40	50	60
Hlavní 1	Zelená	Červená	Červená	Červená	Červená	Zelená	Zelená
Hlavní 2	Červená	Červená	Zelená	Zelená	Červená	Červená	Červená
Vedlejši 1	Červená	Zelená	Červená	Červená	Červená	Červená	Červená
Vedlejší 2	Červená	Červená	Červená	Červená	Zelená	Červená	Červená



Obr. 3.1.1 Světelná křižovatka

Na obr. 3.1.1 lze pak vidět mikroskopické řešení světelné křižovatky. Zelené uzly A1,A2,A3 a A4 tvoří vstupní část křižovatky, odtud se každé auto zařadí do správného směru. Modré uzly A5,A6,A7 a A8 představují koncové uzly křižovatky. Mezi směrovými uzly a koncovými uzly jsou znázorněny křižovatkové hrany. Červené křižovatkové hrany jsou ty, v jejichž směrech je průjezd zakázán (mají červenou). Zelené křižovatkové hrany mají v daném okamžiku zelenou a jsou plynule průjezdné. U jednotlivých hran pak rychlost v pruhu odbočujícím doprava bude mít menší rychlost než pruh jedoucí rovně, obdobně se bude chovat také hrana odbočující doleva. Semafory jsou, stejně jako v reálném světě, považovány pouze za nástavbu křižovatky s jednou hlavní silnicí popř. křižovatky s předností zprava. Proto se i průjezd světelnou křižovatkou při zpuštění dvou pruhů najednou (např. puštění protilehlých hlavních nebo vedlejších silnic) řídí stejnými pravidly.

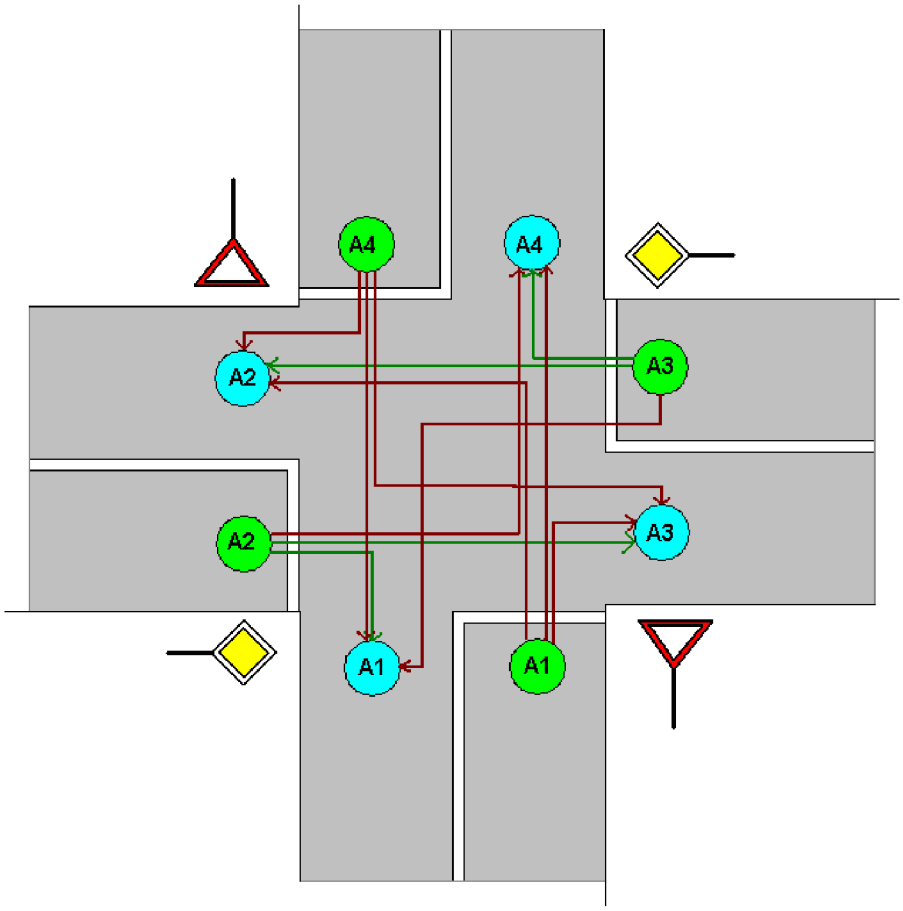
Křižovatka nesvětelná s jednou hlavní silnicí

Tato křižovatka je řešena pomocí priority (jedná se o vlastnost hran *Priority*). Teoreticky by nám postačilo k řešení nesvětelné křižovatky pouze hodnot priorit 0 a 1, kde by hodnota 0 označovala hlavní a hodnota 1 označovala silnici vedlejší. Vzhledem ke složitosti některých křižovatek však došlo k navýšení typů priorit. Toto navýšení bylo potřeba z důvodu křižovatek, které mají více jak dvě vedlejší hrany. Díky tomu, že na dané křižovatce neexistuje světelná signalizace, je nájezd z vedlejší hrany poněkud složitější.

V tuto chvíli je řešení takové, že auto jedoucí po hlavní, jede jako by před ním nebyla žádná křižovatka (tzn. pokud chce jet stále po hlavní), nebrzdí a plynule pokračuje. Pokud chce odbočit na vedlejší pak zpomaluje. Auto na vedlejší zkontroluje, zda se v bezpečné vzdálenosti, respektive v 1,5 násobku bezpečné vzdálenosti pro maximální rychlost dané hrany (tzn. pokud máme silnici o max. rychlosti 50 km/h a za volantem sedí řidič s reakční dobou 2s pak bude kontrolovat vzdálenost rovnou přibližně 42m), zda se od něj na hlavní silnici v obou směrech neobjevuje jiný vůz. Pokud tomu tak není a nemusí dávat přednost protijedoucímu vozidlu, vjede do křižovatky a pokračuje zvoleným směrem. Jakmile vjede libovolné vozidlo do křižovatky s vedlejší silnice musí vozidla na hlavní začít zpomalovat.

To se provede pomocí označení daného směru za obsazený (pomocí vlastnosti křižovatkové hrany). Po vyjetí vozidla z vedlejšího směru se opět stav křižovatky nastaví do předchozího stavu, kdy byly pro průjezd standardně nastaveny hrany vedoucí po hlavní silnici.

Může dojít k tomu, že se hlavní silnice ucpe, nebo auta pojedou malou rychlostí, ale v poměrně velkých rozestupech nedosahujících k dané hodnotě vzdálenosti. Pokud tento případ nastane, nastaví se vnitřní hodnota (*HavyTraffic)* na true. Tato vlastnost je nastavena na true pokud je hrana obsazena více jak z 80%. Testuje se aktuální rychlost vozidla a pokud tato rychlost bude pod polovinou max. rychlosti, sníží se i snímaná vzdálenost od křižovatky. To se provede změnou maximální povolené rychlosti za rychlost automobilu přibližujícího se po hlavní silnici.



Obr. 3.1.2 Schéma křižovatky s hlavní silnic

Při rozebírání obr. 3.1.2 lze opět spatřit dvě skupiny barevně od sebe oddělených uzlů. Zelené uzly tvoří opět vjezd do křižovatky, modré pak výjezd z křižovatky. Čísla jednotlivých uzlů znamenají jednotlivé směry. Pomocí koncových uzlů je možno oddělit vnitřní nastavení křižovatky od samotné hrany. Jak je výše popsáno, každá taková křižovatka je tvořena pomocí priorit. Bez koncového uzlu by však musela vzniknout víceúrovňová priorita, kde by každá křižovatka měla vždy vyšší číslo priority. Takto by mohlo dojít k cyklení a konečný systém by nakonec nemohl fungovat. Při použití tohoto uzlu i při nejjednodušších typech křižovatek se může priorita hrany libovolně měnit.

Kruhová křižovatka

Kruhová křižovatka je tvořena z poduzlů, kde každý uzel tvoří jeden z vjezdů nebo výjezdů na nebo z dané křižovatky. Stejně jako u křižovatek bez světel s jednou hlavní i zde se při příjezdu z libovolného směru vždy auta chovají jako by byla na vedlejší, auta, která jedou na okruhu, jsou pak považována za auta jedoucí po hlavní. Výhoda kruhové křižovatky spočívá v tom, že při vjezdu na takovou křižovatku je potřeba hlídat pouze jeden směr provozu. Nevýhodou je to, že pokud na takovou křižovatku vjede nákladní automobil, může na určitou dobu zablokovat několik vjezdů/výjezdů současně.

Kruhová křižovatka může být i víceúrovňová, kdy se z nitřního kruhu můžete dostat na libovolný směr, ale vnější kruh je určen pouze pro průjezdnost k nejbližšímu vjezdu/výjezdu. Toto umožňuje většinou také normální kruhová křižovatka. Víceúrovňová ovšem obě části křižovatky od sebe odděluje tak, aby z přímých průjezdů mezi jednotlivými vjezdy/výjezdy nemohlo dojít k najetí na vnitřní okruh. Při takovémto typu pak musí kruhová křižovatka obsahovat také směrovací poduzly.

Křižovatka s předností zprava

Tento případ může nastat pokud se kříží několik komunikací stejné priority. Využívá se zde opět bezpečné vzdálenosti a to tak, že auto vždy před takovou křižovatkou zastaví nebo alespoň zpomalí. Tyto křižovatky jsou totiž většinou umístěny v nepřehledných místech jako panelové zástavby, parkoviště atd. Pokud se stane, že ze všech směrů současně přijedou na takovou křižovatku automobily, pak se náhodně vybere jedno z aut a to se pustí jako první. Ve chvíli, kdy projede křižovatkou, začnou se podle pravidel pouštět další auta. Pokud by však došlo k tomu, že by za daným autem stálo další, pak se toto další auto pustí jako poslední až ve chvíli, kdy z každého daného směru odjede jedno auto.

3.1.2 Velikost vozidel a bezpečná vzdálenost, změna rychlosti

V části 3.2.1 již o bezpečné vzdálenosti respektive o reakční době řidiče bylo psáno. V této kapitole je tato veličina podrobně vysvětlena. Dále je zde popsána proměnlivá velikost vozidel.

Velikost vozidel

Velikost auta je důležitou informací z důvodu zaplnění silnic v případě, že dané auto stojí v zácpě. Jednou z možností je tuto informaci nastavit jako fixní hodnotu na velikosti 4,5 m. Druhou možností je tuto informaci dodávat náhodně v rozmezí od 4 do 15m.

První varianta umožňuje lepší testování a je zajímavá v případě, že bereme do úvahy pouze osobní dopravu. Druhá varianta je náročnější jak pro testování, tak pro zadání vstupních informací. Pokud se totiž zadá do systému příliš mnoho nákladních automobilů, logicky se velmi rychle ztíží průjezdnost danou částí simulovaného města a může dojít až k totální kolizi určeného dopravního systému.

Bezpečná vzdálenost

Bezpečná vzdálenost je taková vzdálenost, při níž je řidič schopen zabrzdit dříve než by se mohl srazit s automobilem před ním jedoucím. Je reprezentována

vzdáleností, kterou auto urazí za daný časový úsek, rovnající se reakční době řidiče. Dle BESIPU je reakční doba průměrného řidiče přibližně 2s. [5]

Tato doba se nevztahuje na profesionální řidiče ani na řidiče, u nichž jejich povolání vyžaduje okamžité rozhodování. Navíc tato hodnota v sobě zahrnuje některé rušivé vlivy jako monotónnost řízení na dálnici nebo špatný stav komunikace. Takovéto stavy však na silnici ve městě můžeme částečně potlačit. Proto jsem se rozhodl tuto hodnotu snížit na velikost 1,5s pro standardního „čerstvého“ řidiče. Hodnota 2s bude určena pro řidiče, kteří jsou již na trati delší dobu. Také budou do této skupiny spadávat řidiči „středně pokročilí“ tzn. řidiči, kteří mají řidičský průkaz 1 až pět let dle odježděných kilometrů. Do skupiny s rychlostí reakce kolem 3s budou pak patřit sváteční řidiči a řidiči začátečníci. Řidiči profesionálové nebo ti, u kterých jejich povolání vyžaduje okamžité rozhodování (např. stíhací piloti, závodníci, vrcholní sportovci), dosahují dle testů v průměru o 0,5s lepší časy než řidiči “čerstvého typu“.

Pokud budeme brát v potaz i nákladní dopravu pak čas tří sekund bude patřit také nákladním autům do délky 10m a čtyř sekundová hranice bude určena pro nákladní automobil, který přesahuje vzdálenost 10m. Vypočítaná vzdálenost je pak v tabulce 3.1.2.

Změna rychlosti auta

Jak bylo výše uvedeno každý automobil má určenou svojí maximální rychlost, která může dosáhnout až +10% max. rychlosti na dané komunikaci. Této rychlosti však auto nemusí vůbec dosáhnout. Každé z vozidel totiž postupně zrychluje nebo zpomaluje v závislosti na stavu silnice před ním.

Prakticky to znamená, že pokud se ve vzdálenosti 1,5 násobku bezpečné vzdálenosti před automobilem neobjevuje žádné cizí auto nebo jiná překážka, pak se jeho rychlost zvyšuje. Pokud se v této vzdálenosti cizí automobil objeví, udržuje námi řízené vozidlo konstantní rychlost a pokud se tato překážka dostane na hranu bezpečné vzdálenosti, začne zpomalovat. Mimo pohyblivých vozidel dokáže auto obdobným způsobem reagovat na kolonu, případně jinou stojící překážku a také příjezd ke křižovatce.

Tab. 3.1.2 Tabulka bezpečných vzdáleností

Bezpečná vzdálenost					
	Čas [s]				
Rychlost [km/h]	1	1,5	2	3	4
0	0,5	0,5	0,5	0,5	0,5
5	1,39	2,08	2,78	4,17	5,56
10	2,78	4,17	5,56	3,47	4,63
15	4,17	6,25	8,33	12,50	16,67
20	5,56	8,33	11,11	16,67	22,22
25	6,94	10,42	13,89	20,83	27,78
30	8,33	12,50	16,67	25,00	33,33
35	9,72	14,58	19,44	29,17	38,89
40	11,11	16,67	22,22	33,33	44,44
45	12,50	18,75	25,00	37,50	50,00
50	13,89	20,83	27,78	41,67	55,56
55	15,28	22,92	30,56	45,83	61,11
60	16,67	25,00	33,33	50,00	66,67
65	18,06	27,08	36,11	54,17	72,22
70	19,44	29,17	38,89	58,33	77,78
75	20,83	31,25	41,67	62,50	83,33
80	22,22	33,33	44,44	66,67	88,89
85	23,61	35,42	47,22	70,83	94,44
90	25,00	37,50	50,00	75,00	100,00
95	26,39	39,58	52,78	79,17	105,56
100	27,78	41,67	55,56	83,33	111,11
105	29,17	43,75	58,33	87,50	116,67
110	30,56	45,83	61,11	91,67	122,22
115	31,94	47,92	63,89	95,83	127,78
120	33,33	50,00	66,67	100,00	133,33
125	34,72	52,08	69,44	104,17	138,89
130	36,11	54,17	72,22	108,33	144,44
135	37,50	56,25	75,00	112,50	150,00
140	38,89	58,33	77,78	116,67	155,56
145	40,28	60,42	80,56	120,83	161,11
150	41,67	62,50	83,33	125,00	166,67

Všechny rychlosti v programu jsou udávány v m/s. Toto je z důvodu rychlejšího výpočtu bezpečných vzdáleností při přepočítávání ujeté vzdálenosti.

3.1.3 Časový krok simulace, dynamická reakce vozidel

Časový krok simulace

Je logické, že samotná simulace se neprovádí spojitě. Spojitá simulace není možná už ze samotné podstaty simulování na PC. Obzvláště pokud musí PC provádět souběžně několik posunů automobilů současně. Časový krok tedy představuje velmi důležitou vlastnost jakékoliv simulace. Počítač navíc musí být schopen v daném časovém kroku propočítat všechny výpočty. Pokud by totiž časový krok byl např. pro velký systém 1ms, došlo by k zahlcení procesoru a k havárii simulačního prostředí.

Další důležitou veličinou spojenou se simulačním krokem je poměr reálného času k času simulace. Poměr mezi těmito kroky může být 1/1 v případě, že se simuluje rychlá reakce.

V případě simulátoru dopravy jsem pak zvolil krok simulace 1s ku reálnému času také 2, 5, 10, 20, 50 a 100, neboli každá sekunda simulace představuj 2, 5, 10, 20, 50 nebo 100 sekund reálného času.

Dynamická reakce vozidel

V části 3.1.2 jsou popsány základní vlastnosti potřebné pro posun vozidel po mapě. Každé vozidlo má svoje ojedinělé identifikační číslo (pod tímto číslem se nalézá v listu vozidel). Obdobný list vozidel má i každá hrana Listu hrany (*CarsList*). V tomto listu jsou vypsána všechna vozidla jedoucí na dané hraně. Posun vozidel se pak provádí tak, že se vybere první vozidlo z listu. Zjistí se, zda ve vzdálenosti 1,5 násobku bezpečné vzdálenosti vidí překážku. Pokud překážku nevidí a současně vozidlo nedosáhlo maximální rychlosti, zvýší se jeho aktuální rychlost.

Změna rychlosti společně s posunem vozidla lze provést dvěma způsoby. Každý z nich má určité výhody i nevýhody. Jedná se nejprve o posunutí vozidla a poté vypočítání nové rychlosti a posunutí vozidla s novou rychlostí.

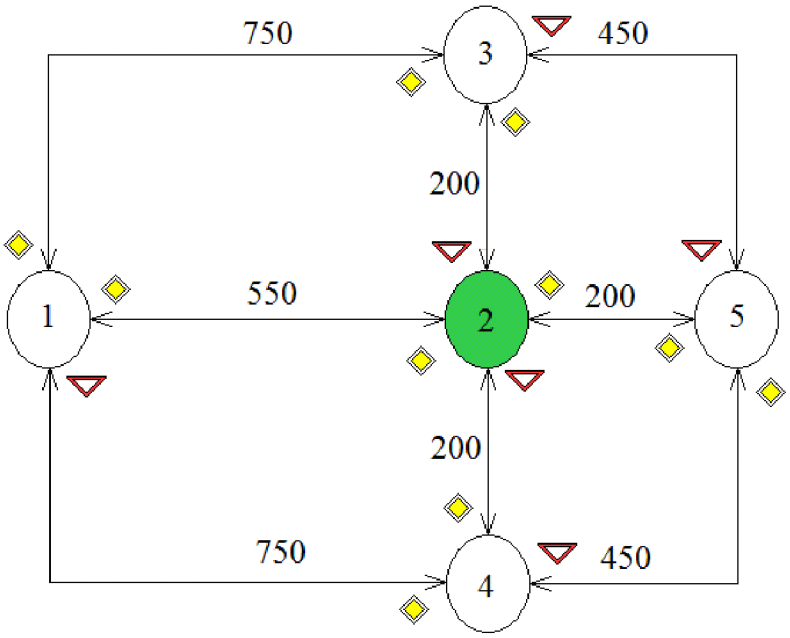
Prvním případem je nejprve posunutí vozidla a poté vypočítání nové rychlosti. Výhodou je méně přepočítávání při zrychlování, nebezpečí však hrozí při zpomalování. Může zde dojít k srážce díky tomu, že se druhé auto posune do míst,

kde by být nemělo. Při simulaci by mohlo navíc docházet k výrazným chybám v důsledku časového odstupu jednoho kroku při velkém kroku.

Druhým případem, který je v této práci použit, je nejprve vypočítání nové rychlosti a poté posun vozidla. V tomto případě se přepočítá vzdálenost ujetá danou rychlostí a dle pozice vozidla se určí, zda zrychlovat či zpomalovat. Tento princip je sice nepatrně složitější, ale vozidla se pohybují dle skutečného provozu. Zvyšování rychlosti se provádí pomocí funkce Random v rozmezí od 0,1 po 3m/s. Znovu se přepočítá možná ujetá vzdálenost a pokud by se stalo, že by novou rychlostí mohlo vozidlo narazit do překážky, sníží se tato rychlost o 0,5 m/s a znovu se testuje zda při nové rychlosti nedojde ke srážce.

3.2 VYTVOŘENÍ TESTOVACÍ MAPY Č.1

Testovací mapa číslo jedna byla vytvořena pro první testování a odladění drobných chyb. Hodnoty nad jednotlivými hranami udává velikost hrany v metrech. Zelený uzel č. 2 je uzel, který je řízen pomocí semaforu. Ostatní uzly jsou pak křižovatky s jednou hlavní silnicí.



Obr. 3.2.1 Vytvořená testovací mapa č.1

Značky u každého uzlu představují označení hlavních a vedlejších silnic na dané testovací mapě.

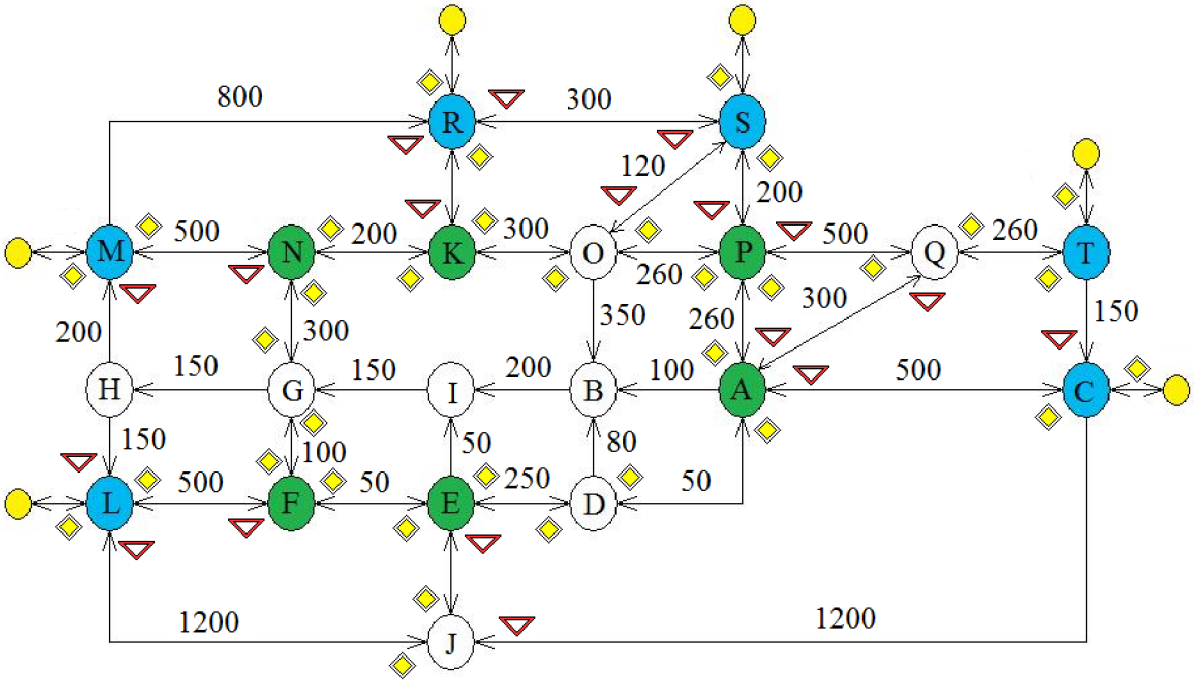
I když se to nezdá a na schématu testovací mapy č.1 je pouze 5 uzlů, v důsledku tvorby poduzlů obsahuje tato mapa celkově 57 uzlů a 14 pojmenovaných hran. Pojmenované hrany jsou hrany, které nepatří do vnitřní struktury uzlu.

K testování bylo použito celkově 30 řízených vozidel s proměnnou délkou a reakční dobou.

3.3 VYTVOŘENÍ TESTOVACÍ MAPY Č.2

Testovací mapa č.2 tvoří určitou představu o vnitřní infrastruktuře zkoumaného města. Jednotlivé uzly orientovaného grafu tvořícího tuto mapu se dělí na tři podskupiny. Světelné křižovatky (ve schématu označeny zelenou barvou), vstupní a výstupní uzly (světle modrá) a obyčejné křižovatky (bílé). Výstupní uzel je zavádějící pojem, žádné auto totiž nezastaví v křižovatce jako v konečném bodě. Při představě dojetí auta jsem tedy předpokládal, že auto ukončí svojí trasu na některé z hran. Proto je v mapě vytvořen další pomocný uzel (žlutá barva), mezi tímto žlutým uzlem a Konečným uzlem je vytvořena obousměrná hrana. Pokud některé auto najede na tuto hranu, ukončí zde svou cestu. Obdobně se na této hraně ve směru do města objevují šumové auta.

Město je děleno na vnitřní okruh řízený převážně světelnými křižovatkami (ve schématu níže označen jako A, D, E, F, G, N, K, O, P), spojnicovými uzly (B, I, J, H, Q) a samotné vstup/výstupní uzly. Každý uzel (node) je tvořen několika poduzly v závislosti na složitosti křižovatky viz. kapitola 3.1.1 Průjezd křižovatkou.



Obr. 3.2.2 Testovací mapa č. 2

Tato testovací mapa je hlavním testovacím prostředím simulátoru. Veškeré vlastnosti hran i uzlů jsou přenášeny pomocí jejich vlastností a díky tomu lze jednotlivé informace snadno měnit. Obdobně jako u testovací mapy č.1 (Obr. 3.2.1) jsou u každého vrcholu, do kterého směřují hrany vyznačeny pomocí silničních značek, hlavní a vedlejší silnice. Vrcholy, které nemají silniční značení, využívají přednosti zprava. Mimo kruhových křižovatek tak jsou zde zastoupeny všechny testované typy.

Testovací mapa č.2 byla inspirována mapou města Prostějova, respektive mapou Prostějova byl inspirován hlavně vnitřní okruh města. Pro simulaci jsem některé skutečné křižovatky spojil dohromady. Velikosti jednotlivých hran je odvozena od skutečných vzdáleností. Tyto vzdálenosti byly odvozeny z mapového měřítka na stránkách www.mapy.cz. Město Prostějov bylo vybráno z důvodu vnitřního i venkovního okruhu viz. Příloha.

3.4 MAKROSKOPICKÉ ŘEŠENÍ POHYBU AUTOMOBILŮ

3.4.1 Kapacita silnice a její obsazenost

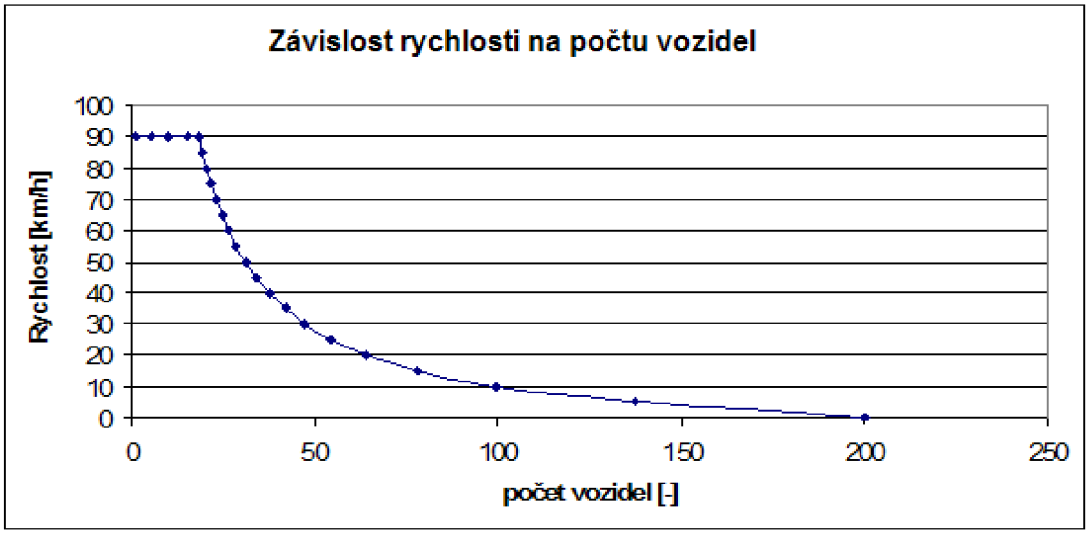
Kapacita silnice stejně jako její časové ohodnocení jsou informace, u kterých se stírá rozdíl mezi makroskopickým a mikroskopickým pohledem na řešení simulátoru. Tyto vlastnosti jsou totiž velmi důležité pro zjišťování toků vozidel potřebných pro makrosvět a přitom se bez těchto informací nedokážeme obejít ani v mikrosvětě.

Kapacita silnice a její obsazenost

Z výše uvedených vlastností je patrné, že kapacita silnice je poměrně relativní hodnota. Tato hodnota je závislá jednak na počtu autíček, tak na jejich rychlosti.

Pokud budeme mít hranu reprezentující úsek 1 km s maximální rychlostí autíček 50km/h, s jedním pruhem a za předpokladu, že nebudeme mít na konci silnice žádné čekající auto na průjezd křižovatkou a nebudeme počítat s počátečním a koncovým zrychlením respektive zpomalením, pak při maximální rychlosti a reakční době 2s bude každý automobil relativně zabírat velikost 27,78 m +4,50 m samotného auta (viz tab 3.1.2 řádek č.11). V takovém případě se na tuto hranu vejde 30 vozidel. Pokud bychom si však vypočítali, kolik stojících vozidel se vleze na danou hranu, dostali bychom se na hodnotu 200 vozidel (4,5m vozidla + 0,5 metru).

Tato závislost je patrná na grafu (obr.3.4.1). Obdobný graf je i na obr. č. 2.4.6. Obdobným způsobem byly získány i ostatní průchody trojúhelníkového modelu dopravy. Tento výsledek byl poté použit jako kontrola správné implementace daného matematického modelu do programové části této práce.



Obr.3.4.1 Graf Závislosti rychlosti na počtu vozidel

3.4.2 Ohodnocení hrany

Každá hrana má svou cenu. Cena hrany je čas, který trvá průjezd danou hranou. Obecně však lze říci, že cennou může být více informací zejména proto, že v dnešní době lze mimo rychlosti dostupnosti některého místa brát v potaz i ujeté kilometry. Proto bude vždy záležet na daném algoritmu, který se použije pro prohledávání cesty. Graf se začne vždy procházet, pokud auto dostane zprávu, že daná hrana, kterou má navrženou ve své paměti, je zablokována.

Blokování cesty je možné buď ručně, jedná se o simulaci dopravní nehody nebo jiné nepředvídatelné situace, nebo ve chvíli, kdy se na dané hraně objeví stojící kolona, přesahující buď 80% (*HeavyTraffic*) dané hrany (v případě auta závislého na časovém průjezdu), nebo ucpání hrany (*Full*) .

3.5 OVLÁDÁNÍ SIMULAČNÍHO PROSTŘEDÍ

Ovládat simulačního prostředí lze zatím pouze pomocí úprav ve vlastním zdrojovém textu.

Krok simulace

Krok simulátoru je nastaven na jednu sekundu a lze zvolit poměr mezi skutečným a reálným časem pomocí Enumerátor *SimulationSpeed*. Tuto hodnotu lze nastavit v SimulationTimer/

```
private int acceleration = SimulationSpeed.SPEED_10X.getSimulationSpeed();
```

kde SPEED_10x je momentálně nastavený poměr.

Automobil

Nastavení vozidel je možno měnit díky vlastností v samotném konstruktoru. Konstruktor, který je zde uveden jako příklad, patří k simulaci testovací mapy č.1.

```
car = new Car(this.generateNewCarID(), edge12, edge10, 0.1, 5,
  ReactTime.TIME_2S, 2, AlgorithmType.DIJSKRA__DISTANCE, false
```

ID každého vozu se vypočítá automaticky, první hrana (edge) je hranou startovací, druhá hrana je poté hranou koncovou. První hodnota (double v příkladu níže 0,1) je startovací pozice na první hraně. Druhá hodnota (double v příkladu velikost 5) je délka vozidla. Dále se zadává Reakční doba pomocí enumerátoru, maximální dovolené zrychlení vozu a pomocí enumerátoru opět typ algoritmu. Poslední zadanou (boolovská v příkladu false) je hodnota, zda je automobil řízený nebo šumový. Pokud je vůz řízený, je tato hodnota nastavena na false.

Hrana

Konstruktor hrany obsahuje celkově sedm vlastností, které jsou pro nadefinování hrany potřebné.

```
public Edge( Node pStartNode,  Node pEndNode, double pRoadLenght,
             int pRoadCapacity, int pMaxSpeed,
             boolean pCrossRoadEdge, int pPriority)
```

Nejdůležitějšími prvky každé hrany je nadefinováni startovacího a konečného vrcholu (*pStartNode, pEndNode*) a její velikost (*pRoadLenght*). Mimo těchto nejdůležitějších prvků je třeba nadefinovat také kapacitu vozovky (*pRoadCapacity*), maximální dovolenou rychlost (*pMaxSpeed*), označení, zda se jedná o křižovatkovou hranu nebo o normální hranu (*pCrossRoadEdge*) a konečně také prioritu vozovky (*pPriority*). Maximální rychlost se udává v m/s. Označení hrany je false pokud je hrana normální a true pokud se jedná o hranu křižovatkovou.

Uzly

Nastavení uzlů je opěk vyřešeno pomocí konstruktorů. Konstruktor pro uzel však obsahuje pouze dvě vlastnosti. První vlastností je semafor (pokud je tato hodnota nastavena na null, semafor v uzlu není, jinak se nastaví název semaforu). Druhou vlastností je samotné identifikační číslo uzlu. Při vkládání uzlu do grafu je možno k uzlu připsat také jméno. Pomocí jména lze následně lépe graf sestavovat.

Semafor

Poslední nastavovanou entitou je Semafor. Konstruktor semaforu vypadá následovně:

```
public Semaphore(int pRedLength, int pGreenLength,
                 int initialDelay, boolean pIsRed)
```

Při tvorbě semaforu se nastavují čtyři hodnoty - doba červené (pRedLenght), doba zelené (pGreenLength), čas po který je při startu rozsvícená první barva (initialDelay) a dotaz na barvu, která po inicializace svítí jako první (pIsRed). Pokud je tento dotaz true, svítí jako první barva červená, pokud je false svítí zelená.

Chod simulace

Po spuštění simulace se v adresáři Log vytvoří tolik souborů, kolik je v dané simulaci aut. Každý soubor nese ve svém názvu identifikační číslo auta a zda je auto řízené či šumivé.

Po otevření každého z daných souborů je možno pozorovat následující informace viz. obr.3.5.1.

```
Time: -1
PlanedRoads:
   A14k --> A41
Speed: 0.0
Position: 0.2
---------------------

Time: 0
PlanedRoads:
   A14k --> A41
   A41b --> A42k
   A42k --> A24
   A24 --> A24b
   A24b --> A21k
   A21k --> A12
   A12 --> A12b
   A12b --> A13k
   A13k --> A31
   A31 --> A31b
   A31b --> A35k
   A35k --> A53
Speed: 2.5308628138697014
Position: 0.2
---------------------
```

Obr.3.5.1 Výpis programu

První vypsaná část, která začíná řádkem Time:-1 je inicializací. V prvním kroku se vozidlu vypočítá trasa z hrany na které stojí do hrany na kterou má dojet (*PalnedRoads*). Vozidlo získá také novou aktuální rychlost (*Speed*) a pokud je čas vyšší jak 0, začne se měnit i jeho aktuální pozice (*Position*). Aktuální pozice je v procentech.

Simulace se sama neukončí po dojetí všech autíček je proto nutné simulaci ukončit ručně. Toto ukončení je možno provést stlačením klávesy Enter.

4. ZÁVĚR

Při tvorbě diplomové práce bylo potřeba nejprve vytvořit rešerši podkladů ze tří samostatných odvětví. Tyto tři části jsou simulace dopravy, multiagentní systémy a průchod orientovaným grafem. Nastudované materiály byly následně zkompletovány do návrhu multiagentního systému využívajícího průchodu orientovaným grafem k simulaci dopravního prostředí.

Tvorba programové části práce se následně „opírá“ o využití teorie trojúhelníkového modelu makroskopického pohledu na dopravní systém, mikroskopického model Car-following a také modelu Route-Choise. Model použitý pro vyhledání nekratší cesty v simulaci (Route-Choise) byl implementován pomocí algoritmu Dijkstra. Jednotliví agenti (vozidla) pracují na principu deliberativního multiagentního systému a nejvíce se blíží systému IRMA.

Každý agent je schopen se samostatně pohybovat v simulačním prostředí po předem vytvořené mapě. V systému je implementována dynamická změna rychlosti. Mapu prostředí lze libovolně měnit pomocí návrhu flexibilního simulačního prostředí.

Omezení simulátoru vychází zejména ze zjednodušení simulace. Z tohoto důvodu pak nebyl implementován mikroskopický model předjíždění (Lane-Change model). Ostatní omezení lze pomocí proměnných přednastavit tak, aby se simulace více blížila reálnému prostředí (viz. Kapitola 3.5 Ovládání simulačního prostředí). Při takovém nastavení však může dojít k zablokování některých částí mapy. Mikroskopická část simulátoru je tak tvořena pouze modelem kontroly překážky nebo také reakcí modelu na překážku (Car-following).

Řešení nastíněné v této práci lze dále rozvést pomocí velkého množství rozšíření, díky nimž se může tento systém stát přitažlivějším pro uživatele nebo se může rozšířit průkaznost simulace dopravy ve městě. Možným rozšířením pak může být například zapouzdření nejpoužívanějších typů křižovatek do jednoho uzlu. Toto zapouzdření by mohlo vytvořit daleko příjemnější prostředí pro uživatele, obdobně jako vytvoření grafického rozhraní.

Implementace více algoritmů by mohla zvýšit rozmanitost vyhledaných cest. Vytvoření náhodného objevování a mizení šumivých autíček na možných hranách by pak mohlo zvýšit vypovídací hodnotu simulace. Další možností by bylo zakreslení do mapy uzlů a jim odpovídajících hran, které by byli označeny jako nákupní centra, domov nebo práce. Každý agent by si pak vybíral dle vnitřního algoritmu, kam má zrovna jet a podle jeho vnitřního časovače uvažoval, jak dlouho v daném místě zůstane. Takto by pak vypadalo komplexní řešení multiagentního systému postaveného na základě principu IRMA.

POUŽITÁ LITERATURA

[1] <http://auto.idnes.cz/pocet-aut-v-cesku-se-od-revoluce-zdvojnasobil-stara-jsou-porad-stejne-stara/auto_ojetiny.asp?c=A091016_110258_auto_ojetiny_fdv>
Dne: 27.3.2010

[2] <http://civ.cvut.cz/info/download_prednasky.php?name=ti_03.ppt>
Dne: 12.12.2009

[3] <http://www.cam.zcu.cz/~ryjacek/students/DMA/skripta/8.pdf>
Dne:12.12.2009

[4] <http://docs.cirkva.net/CVUT/FSV/Modelov%E1%ED_12/Definice.doc>
Dne: 14.12.2009

[5] <http://www.ibesip.cz/Rychlost/Bezpecna-vzdalenost>
Dne:14.12.2009

[6] HLIŇĚNÝ, P. Základy Teorie Grafů pro (nejen) informatiky: Ústav Informatiky, Masarykova Univerzita, Brno, 2010

[7] < sofe-files.pepiino.cz/rek/document4.pdf>
Dne 5.5.2010

[8] KUBÍK,A. Inteligentní agenty. Brno: Computer Press, a.s., 2004.
ISBN 80-254-0323-4.

[9] KUBÍK,A. Agentově-orientované inženýrství: nové paradigma pro tvorbu softwaru?. Ústav informatiky, Slezská univerzita, Opava., 2003

[10] KOUTNÝ, V. Semestrální projekt II. Ústav kybernetiky, Vysoké učení technické, Brno, 2009

[11] BURIAN, P. Multiagentní systémy a řízení výroby ve společnosti. Ústav počítačové a řídící techniky, VŠCHT Praha, 2004

[12] ZBOŘIL, F. Agentní a Multiagentní systémy, (Studijní opora). Fakulta Informačních technologií, Vysoké učení technické, Brno, 2006

[13] NI, D. 2DSIM: A Prototype of Nanoscopic Traffic Simulation. School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta, 2003

[14] KLEIN, U.; SCHULZE, T.; STRAβBURGER, S. Traffic Simulation Based on the High Level Architecture, Department for Computer Science, Otto von Guericke University, Magdeburg, 1998

[15] ESTLIN, T.; GAINES, D.; FISHER, F.; CASTANO, R. Coordinating Multiple Rovers whit Interdependent Science Objectives, Jet Propulsion Laboratory, California Institute of Technology, Pasadena 2005

[16] www.mapy.cz
Dne: 14.4.2010

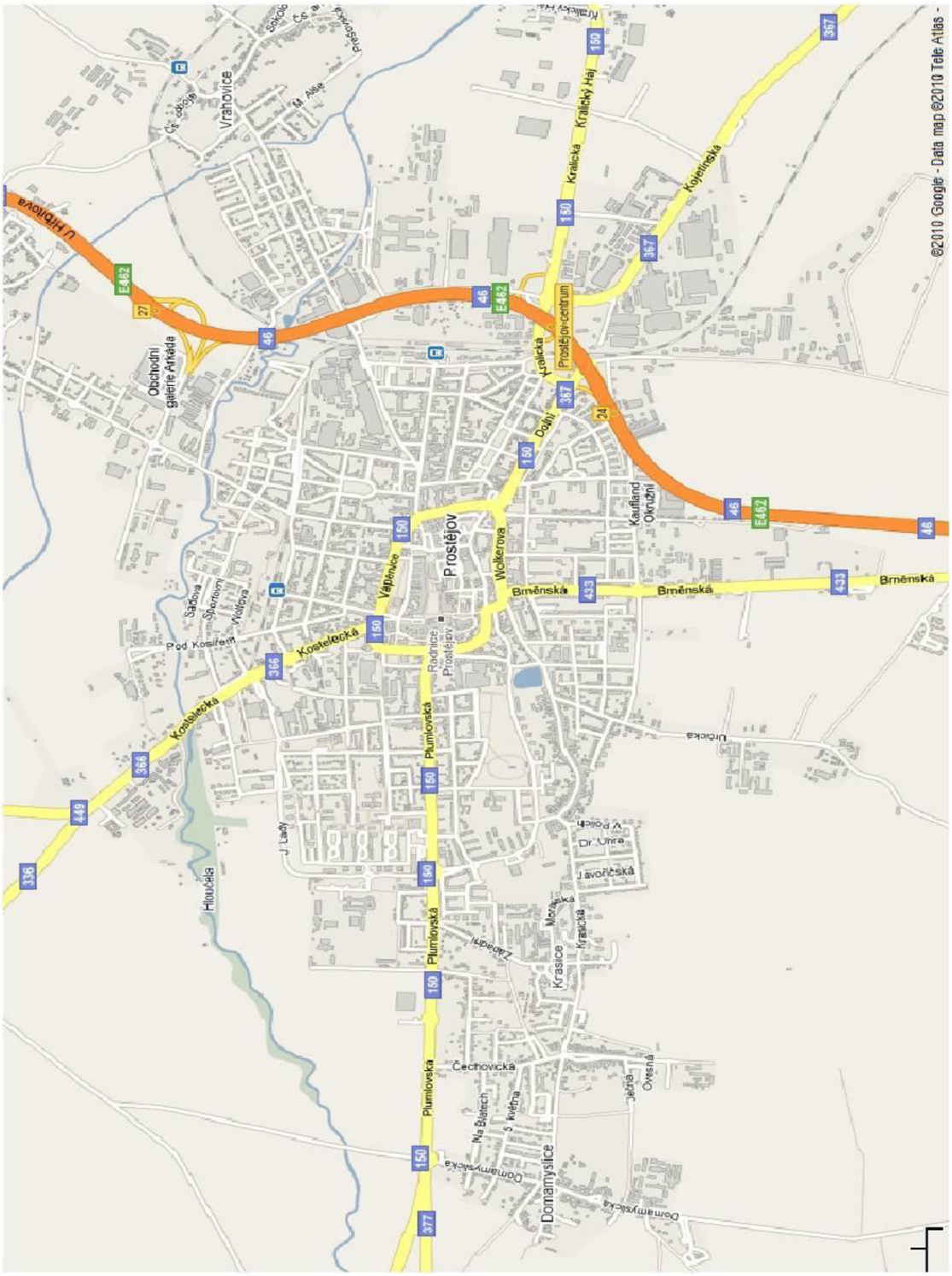
[17] JURÁNEK, V. Algoritmy v teorii grafů: Přírodovědecká fakulta, Masarykova Univerzita, Brno 2009

[18] BILAN, M. Simulace silniční infrastruktůry: Ústav Automatizace a Měřící techniky, Vysoké učení technické, Brno 2009

[19] DYM C., Traffic Flow Models, Principles of mathematical modeling, Academic Press., 6/2004, ISBN-13:978-0-12-226551-8

[20] IMMERS L.H., LOGGHE S., Traffic Flow Theory, Basics of Traffic Engineering, 5/2003

PŘÍLOHY



Obr. Mapa města Prostějova