VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

VÝPOČETNÍ MODEL A ANALÝZA SYSTÉMU ADAPTIVNÍCH SEMAFORŮ

COMPUTATIONAL MODEL AND ANALYSIS OF ADAPTIVE TRAFFIC LIGHTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **FILIP TERBR**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JOSEF STRNADEL, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Terbr Filip**
Program: Informační technologie
Název: **Výpočetní model a analýza systému adaptivních semaforů**
Computational Model and Analysis of Adaptive Traffic Lights
Kategorie: Modelování a simulace

Zadání:

1. Zdokumentujte klíčové pojmy a principy související s řízením provozu na pozemních komunikacích pomocí semaforů. Navrhněte vhodnou abstrakci účastníků provozu řízeného semafory, jeho prvků a okolí.
2. Proveďte detailní rešerši v oblasti prostředků výpočetního modelování systémů a analýzy jejich vlastností. Zvolte prostředky a dopravní uzel vhodné k řešení zadané práce.
3. Pomocí zvolených prostředků vytvořte výpočetní model chování systému adaptivních semaforů (tj. semaforů schopných samočinné koordinované adaptace na aktuální provoz s cílem maximalizovat dopravní propustnost). Model musí zahrnovat účastníky provozu, prvky sloužící k jeho řízení, umožnit sledovat děje klíčové pro řízení a činnost semaforů a jejich vliv na dopravu.
4. Funkčnost modelu demonstrujte v několika vhodně zvolených podmínkách.
5. Diskutujte a zhodnoťte možnosti vytvořeného modelu z hlediska sledování dějů a analýzy různých vlivů a navrhněte možné směry pokračování v projektu.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání, vytvoření základního výpočetního modelu řízení dopravy na bázi semaforů.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Strnadel Josef, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 25. října 2019

Abstrakt

Bakalářská práce se věnuje tématu adaptivních křižovatek a studiem jejich vlivu na plynulost provozu a jiné. V práci je obsažen rozbor aktuálně používaných dopravních detektorů, rozbor hlavních problémů klasických křižovatek a možnosti jejich řešení, následně návrh modelu, popis realizace a testování vytvořeného modelu. Výsledný model je vytvořen v softwaru UPPAAL SMC a jedná se o systém navzájem komunikujících časovaných automatů. Testování probíhá porovnáváním výsledků testů klasických semaforů s výsledky adaptivních semaforů.

Abstract

This thesis deals with the topic of adaptive intersections and the study of their influence on the flow of traffic and others. The work contains an analysis of currently used traffic detectors, an analysis of the main problems of classic intersections and the possibilities of their solution, followed by the design of the model, a description of the implementation and testing of the created model. The resulting model is created in the UPPAAL SMC software and is a system of timed automata communicating with each other. Testing is performed by comparing the results of tests of classical traffic lights with the results of adaptive traffic lights.

Klíčová slova

křižovatka, adaptivní křižovatka, simulace, modelování, UPPAAL, dopravní detektory, silniční provoz, časované automaty, analýza modelu, statistické ověřování modelu, verifikátor

Keywords

intersection, adaptive intersection, simulation, modelling, UPPAAL, traffic detectors, road traffic, timed automata, model analysis, statistical model verification, verifier

Citace

TERBR, Filip. *Výpočetní model a analýza systému adaptivních semaforů.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Josef Strnadel, Ph.D.

Výpočetní model a analýza systému adaptivních semaforů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Josefa Strnadela, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Filip Terbr
25. května 2020

Poděkování

Rád bych poděkoval panu Ing. Josefu Stnadelovi, Ph.D. za všechny rady, vedení a odborné připomínky, které mi poskytoval po celou doby vypracovávání mé bakalářské práce.

Obsah

1 Úvod **3**

2 Rozbor problému a možnosti realizace **4**
2.1 Světelná dopravní signalizace 4
2.1.1 Střídání semaforů 5
2.1.2 Účastnící provozu 5
2.1.3 Problémy klasických semaforů 5
2.1.4 Očekávaný přínos adaptivních semaforů 6
2.2 Detekce vozidel v křižovatce 7
2.2.1 Video detekce 7
2.2.2 Indukční smyčky 7
2.2.3 Ultrazvukové detektory 9
2.2.4 Mikrovlnné radary 9
2.3 Realizační prostředky a metody 10
2.3.1 SUMO 10
2.3.2 Jazyk C++ s knihovnou SimLib 10
2.3.3 UPPAAL 11
2.3.4 Rozšíření UPPAALu 11
2.3.5 Použité metody 12
2.3.6 Zvolený realizační prostředek 14
2.3.7 Zajímavosti z používání UPPAALu 18

3 Realizace modelu adaptivních semaforů **20**
3.1 Návrh řešení 20
3.2 Popis řešení 23
3.2.1 Automat řídicí intervaly 24
3.2.2 Automat řídící semafory 28
3.2.3 Automat řídicí frontu u semaforu 29
3.2.4 Automat ovládající výjezdové silnice 32
3.2.5 Automat ovládající statické vozidlo 33
3.2.6 Automat ovládající dynamické vozidlo 34
3.2.7 Automat ovládající provoz 35

4 Testování a zhodnocení realizace **36**
4.1 Modelované křižovatky 36
4.1.1 První modelovaná křižovatka 36
4.1.2 Druhá modelovaná křižovatka 38
4.1.3 Třetí modelovaná křižovatka 39

4.2 Aplikace scénářů na modelované křižovatky 41
4.2.1 Způsob sběru a vyhodnocení dat 41
4.2.2 Velká vytíženost křižovatky 42
4.2.3 Nízká vytíženost křižovatky 43
4.2.4 Chování křižovatky v noci 43
4.2.5 Jedna z výjezdových silnic je přehlcena 44
4.2.6 Shrnutí výsledků 44

5 Závěr **45**

Literatura **46**

A Experimenty **48**
A.1 Velká vytíženost křižovatky 48
A.2 Nízká vytíženost křižovatky 56
A.3 Chování křižovatky v noci 63
A.4 Jedna z výjezdových silnic je přehlcena 67

Kapitola 1

Úvod

S dopravními křižovatkami se v dnešní době setkáváme snad úplně všichni, buď jako řidiči dopravních vozidel nebo jako chodci a jejich vylepšení je určitě na místě. Úpravou klasických semaforů na adaptivní semafory se snažíme odstranit mínusy klasických semaforů a přidání různých plusů. Buď můžeme tento pohled na vylepšení brát z pohledu samotných řidičů, kteří vylepšeními získají například snížení času, který stráví v křižovatce, s tím související snížená spotřeba paliva, protože čím méně budou čekat na zelenou, tím méně spálí paliva. Pokud budou muset řidiči méně zastavovat, sníží se šance na vytvoření nehody při zastavování, například nepozorností řidičů.

Pomocí různých detektorů můžeme informovat řadič křižovatky o aktuální situaci v křižovatce. Například různými detektory vozidel, které mohou informovat o obsazenosti jednotlivých pruhů, detektory MHD, jenž nám mohou pomoci upravit intervaly, aby mohlo MHD pokračovat s určitou preferencí. Podobné můžeme použít při detekci vozidel záchranných složek, aby se křižovatka co nejrychleji uvolnila a vozidla záchranných složek mohla ihned pokračovat.

Tato práce popisuje návrh a realizaci modelu křižovatky s adaptivním chováním, který vychází z poznatků získaných studiem používaných metod detekce, dopravního zákonu o dopravě a osobních poznatků z pohledu řidiče vozidla. Následně je tento model křižovatky s adaptivním chováním porovnáván s modelem křižovatky bez adaptivního chování, o tomto pojednává sekce o testování (část 4). Téma jsem si vybral z důvodu zájmu o dopravu a zajímalo mě aktuální řešení adaptivních křižovatek. Toto téma jsem si také vybral z důvodu, zkusit si něco nového a toto téma mi nabídlo rovnou dvě možnosti naučit se s nějakým jiným softwarem, jako je software SUMO a software UPPAAL.

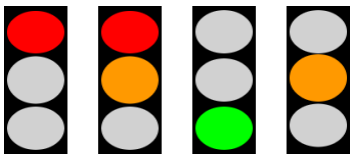
Kapitola 2

Rozbor problému a možnosti realizace

2.1 Světelná dopravní signalizace

Pro tuto sekci bylo čerpáno z daných technických podmínek pro řízení provozu na pozemních komunikacích, viz [17].

Jedná se o soustavu zařízení určených k řízení provozu na pozemních komunikacích s pomocí světelných signálů. Technické požadavky na zařízení určuje převzatá evropská norma EN 50556:2011, česká verze označena ČSN EN 50556 [12].



Obrázek 2.1: Světelné kombinace

Používá se tříbarevná soustava, kde červená symbolizuje příkaz „Stůj!“, vozidla nemohou pokračovat do křižovatky, oranžová symbolizuje příkaz „Připrav se k zastavení vozidla“ a zelená, která symbolizuje příkaz „Volno“, vozidla mohou pokračovat do křižovatky. Při přechodu ze signálu „Stůj“ do signálu „Volno“ se většinou používá rozsvícení kombinace červeného signálu a oranžového signálu.

Obrázek 2.2: Doplňková zelená šipka (převzato z https://automoc.eu/clanky/2017/zelena-sipka-vedle-semaforu/)

Obrázek 2.3: Směrový semafor (převzato z https://www.idnes.cz/auto/historie/semafor-vyroci-londyn.A181211_074820_auto_ojetiny_fdv)

Součástí tříbarevné soustavy může být i doplňková zelená šipka, většinou umístěna hned vedle červeného signálu. Tato doplňková šipka umožňuje pokračovat v jízdě příslušným směrem, řidič však musí dát přednost vozidlům ve volných směrech a nesmí ohrozit ani omezit přecházející chodce. V tříbarevné soustavě se mohou nacházet světla ve tvaru směrových signálů. V tomto případě tato tříbarevná soustava ovlivňuje jen směr daný směrovou šipkou.

2.1.1 Střídání semaforů

Hlavní komponentou světelné signalizace je řadič. V dnešní době je snaha, aby tyto řadiče byli napojeny dálkově i na centrální dispečink, odtud je možné upravit chování křižovatky, či ovládat křižovatku.

Obrázek 2.4: Řadič
(převzato z https://preference.prazsketramvaje.cz/showpage.php?name=technika)

Vždy nemusí mít řadič u křižovatky tuto podobu, ale jedná se o nejběžnější způsob uložení. Často se v blízkosti nachází rozhraní pro ovládání křižovatky, které může například policista využít pro řízení křižovatky.
Na tento řadič se připojí všechny semafory v křižovatce a pokud jsou v křižovatce nějaké detektory [16], tak jsou k řadiči připojeny. Následně už je vše na nastavení dané křižovatky.
U starších řadičů, kde nejsou použity žádné detektory, jsou nastaveny pevné intervaly, tyto intervaly musí být nastaveny tak, aby počítaly s vyklízecím časem nebo s možnou rychlostí reakce řidičů. U novějších řadičů se už většinou počítá s různými detektory a řadiče jsou softwarově i hardwarově připraveny na automatickou úpravu intervalů podle aktuální situace.

2.1.2 Účastnící provozu

Každý, kdo se přímým způsobem účastní provozu na pozemních komunikacích je účastníkem provozu. Hlavním účastníkem silničního provozu je v dnešní době řidič osobního vozidla. Osobní vozidlo má v dnešní době téměř každý. Právě v křižovatce se nejčastěji setkává s ostatními účastníky provozu. Řidič se musí v každé příležitosti chovat, tak aby neohrozil sebe a ani ostatní účastníky provozu. Účastníci provozu se musí řídit světelnými, případně i doprovodnými akustickými signály, dopravními značkami, dopravními zařízeními a zařízeními pro provozní informace, či pokyny od policisty nebo osoby oprávněné k řízení provozu na pozemních komunikacích. S tímto je počítáno i při navrhování modelu křižovatky. [25] Model zatím nerozlišuje, zda se jedná o osobní vozidlo, nákladní vozidlo nebo cyklistu.

2.1.3 Problémy klasických semaforů

Dodržování pevných intervalů

Tato skutečnost se stává problémem, když není křižovatka tolik vytížená, ale semafor stále dodržuje intervaly, takže spouští semafory, u kterých není například žádné vozidlo a vozidla u nespuštěných semaforů čekají na svůj interval.

- Možné řešení:

  Toto se dá vyřešit tak, že pokud nebude v křižovatce žádné vozidlo, intervaly se budou střídat normálně nebo bude spuštěn „noční režim“, přednost není upravovaná světelnou signalizací. Jakmile bude detekováno přijíždějící vozidlo, semafor přepne na interval, kde je i semafor, kam vozidlo přijíždí.

Pevné intervaly mohou být problém, i když je křižovatka hodně vytížená a některé semafory jsou příliš využívané, ale intervaly jsou stále stejné. Může nastat, že fronta u tohoto semaforu bude tak veliká, že bude ovlivňovat ostatní semafory, tak že fronta přesáhne do příjezdové silnice a zablokuje tak přístup k ostatním semaforům.

- Možné řešení:

  Toto se dá vyřešit pouštěním častěji intervalu s více vytíženými semafory. Například se nastaví, že pokud bude ve frontě více jak 5 vozidel, semafor požádá o prioritu a bude v dalším cyklu spuštěn.

Dodržování času intervalu

Dalším problémem je přesné dodržování času, kdy je interval aktivní, pokud má semafor nastaveno, že potrvá 60 vteřin, ukončí se až po těchto 60 vteřinách. Toto se může stát problémem, pokud už vozidla od semaforů v aktuálním intervalu odjela a ráda by jela vozidla od jiných semaforů z jiného intervalu, tyto vozidla musí počkat těchto nastavených 60 vteřin.

- Možné řešení:

  Křižovatka může kontrolovat, zda semafory z aktuálního intervalu mají prázdné fronty a podle toho ukončit aktuální interval a spustit nový interval s vozidly ve frontách.

  Dále by bylo možné globálně upravovat čas intervalu podle vytíženosti křižovatky. Nízká vytíženost, krátké intervaly. Velká vytíženost, normální nebo delší intervaly.

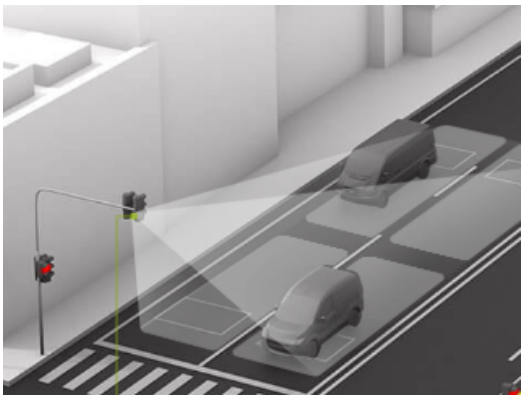
2.1.4 Očekávaný přínos adaptivních semaforů

Od adaptivních semaforů se očekává zrychlení průjezdů vozidel křižovatkou, hlavně v nízkém a středním provozu. [24] U velkého provozu je problémem, že jsou většinu času všechny proudy plné vozidel a semafor se většinu času chová jako klasický semafor s tím, že pokud bude zrovna v nějakém intervalu méně vozidel, povolení průjezdu skončí dříve než u klasických semaforů, které by čekali na čas ukončení. U ostatních dvou druhů provozů je výhoda, že se adaptivní funkce semaforů využijí většinu času na plno. Adaptivní semafory by měly dokázat uvolnit provoz ve městech. Toto by mělo být umocněno, pokud bude více křižovatek s adaptivními semafory za sebou. Z rychlejšího uvolňování vozidel z křižovatky vznikne několik výhod, jak pro řidiče jednotlivých vozidel, kdy řidiči stráví méně času v křižovatce a tím ušetří za pohonné hmoty, jelikož budou méně stát. Z toho vyplývá i méně vytvořených škodlivých látek z vozidla. [20]

2.2 Detekce vozidel v křižovatce

2.2.1 Video detekce

Obrázek 2.5: Umístění kamery (převzato z https://pid.cz/o-systemu/preference/)



Obrázek 2.6: Pokrytí kamerou (převzato z https://www.siegreen.com/traffic-control/)

Video detekce je nejspíše nejpřesnější metodou, jak detekovat vozidla v křižovatce. [22] V dnešní době je video detekční software na takové úrovni, že dokáže zjistit, zda se řidič věnuje řízení. Pro křižovatku je důležitou vlastností zjištění počtu vozidel v jednotlivých proudech. Při dodatečném nastavení detekce je velmi snadné přidat nové detekční zóny. Lze detekovat i o jaké vozidlo se jedná, například zda je přítomno osobní vozidlo nebo autobus MHD, podle této informace může křižovatka zrychlit odbavení autobusu. Tyto detekční kamery jsou velmi náročné na rychlost zpracování videa, při větším rozlišení zachytávaného obrazu bude větší i objem dat, který musí kamera zpracovat. Většina video-detekčních kamer je citlivá na povětrnostní podmínky. Při špatném umístění kamery může docházet k tomu, že větší vozidla zakryjí malá vozidla. [19]

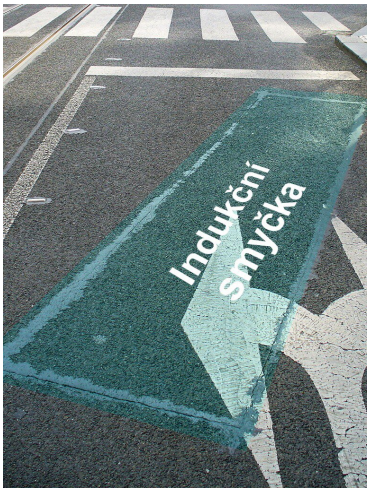
Některé kamery mohou mít problém se stíny, odrazy, nebo s vibracemi kamery a nečistotami na objektivu. Instalace a údržba kamer může vyžadovat uzavření pruhu, či celé vozovky.

Většinou se používá běžná kamera a detekce probíhá pouze rozpoznáváním vozidel v obraze, ale existují i řešení s použitím kombinace s termální kamerou, ta zvyšuje přesnost rozpoznávání, takové řešení nabízí společnost FLIR. [7]

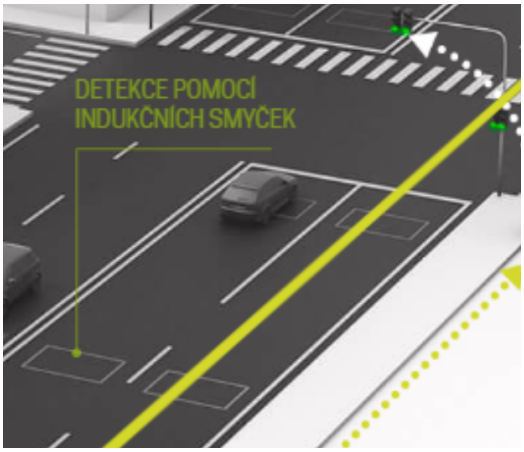
Firma Traficon je světovou jedničkou věnující se dopravní analýze a zpracováním obrazu. Jejich produkt TrafiCAM^{TM} je video detekční kamera, která slibuje přímou náhradu za indukční smyčky, dokáže v křižovatce pokrýt až 8 indukčních smyček. Další jejich produkt TrafiCAM^{TM} Collect-R slibuje video detekci až pro 4 proudy vozidel. [22]

2.2.2 Indukční smyčky

I když je video detekce na velmi výborné úrovni, nejpoužívanějším druhem detekce jsou stále indukční smyčky. Vlastní smyčka tvoří pod vozovkou indukční část oscilátoru, za přítomnosti vozidla se sníží induktance a vzniká měřitelný impuls.

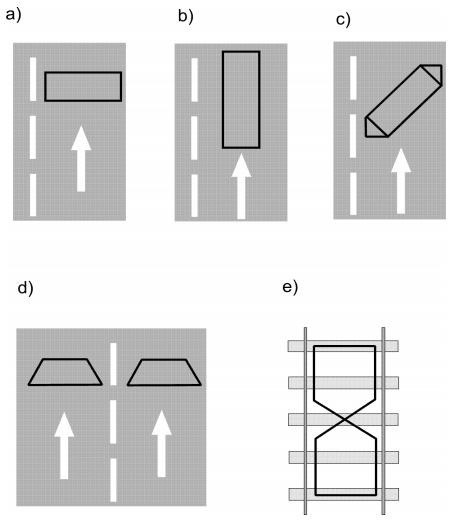


Obrázek 2.7: Smyčka v silnici
(převzato z https://pid.cz/o-systemu/preference/)



Obrázek 2.8: Rozložení smyček
(převzato z https://www.cross-traffic.com/root/download/rizeni-dopravy.pdf)

Klíčové je zde nastavení citlivosti, při špatném nastavení nebude detekce fungovat správně, například pokud bude některá smyčka nastavena velmi velkou citlivost, může detekovat vozidla i z jiných proudů a tím podávat nepravdivé informace řadiči. [18]



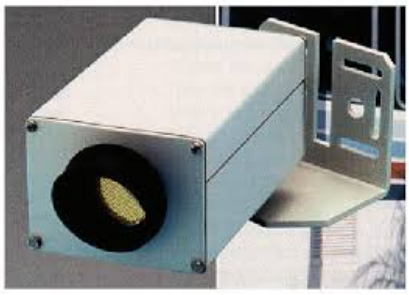
Obrázek 2.9: Možné uložení smyček
(převzato z https://zolotarev.fd.cvut.cz/mzd/ctrl.php?act=show,file,23843)

Existuje několik možností, jak smyčky uložit pod vozovkou, každá možnost má své klady.

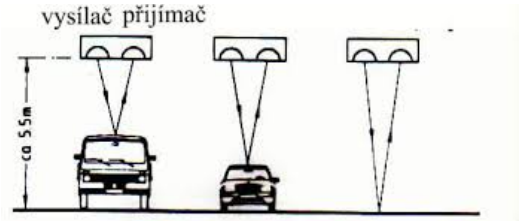
- Možnost a) je klasické uložení.
- Možnost b) slouží pro přesnější detekci kolon.
- Možnost c) dokáže detekovat i cyklisty, ale nejvíce dokáže ovlivňovat ostatní proudy.
- Možnost d) je upravení klasického uložení, aby smyčka nebyla tolik ovlivňována ostatními pruhy.
- Možnost e) je speciální uložení smyčky pro detekci kolejových vozidel.

Indukční smyčky disponují vysokou přesností počítání vozidel, některé modely dokážou i klasifikovat o jaké vozidlo se jedná. Nevýhodou indukčních smyček je, že při instalaci nebo při údržbě je vyžadováno uzavření pruhu, vozovky nebo i celé křižovatky. Obecně těžká vozidla snižují životnost, jak vozovky, tak indukčních smyček.

2.2.3 Ultrazvukové detektory



Obrázek 2.10: Vysílač/přijímač (Převzato z https://pdfs.semanticscholar.org/2fad/e3757dedd6856f02d2e9d32df62ab81f4737.pdf)



Obrázek 2.11: Detekování ultrazvukem (převzato z https://zolotarev.fd.cvut.cz/mzd/ctrl.php?act=show,file,23845)

Používají se dva detektory, jeden, který vysílá signál a druhý, který signál odchytává. Vysílají tlakové vlny zvukové energie, která je na nad slyšitelným spektrem. V pravidelných intervalech vysílač vysílá vlny a měří čas, kdy se odražená vlna vrátí do přijímacího detektoru, pokud se čas liší od času, kdy se vlny odráží od asfaltu.

Tímto principem se zjistí přítomnost vozidla, ale dokážou zjistit i rychlost, délku a výšku vozidel. Při správném nastavení je možná detekce vozidel ve více pruzích. Výhodou jim jsou malé rozměry a velmi snadná instalace. Změny teplot a extrémní poryvy větru mohou negativně ovlivnit funkčnost detektoru, ale existují i ultrazvukové detektory, které dokážou tento problém vykompenzovat. Při použití malé frekvence vysílání signálu se snižuje přesnost detekce, tedy měření obsazenosti. [19]

2.2.4 Mikrovlnné radary

Obrázek 2.12: Mikrovlnný radar (převzato z https://www.instituteofasphalt.org/requirements/papers/44_5.pdf)

Z radaru jsou vysílány velmi krátké impulzy o velkém výkonů a následně jsou přijímány odražené vlny. Vzdálenost detekovaných vozidel je určována pomocí časové korelace vyslaného a přijímaného signálu. Aby byl odraz zachycen radarem musí vyslaný signál dopadnout na vozidlo kolmo, aby se signál odrazil přímo do radaru.

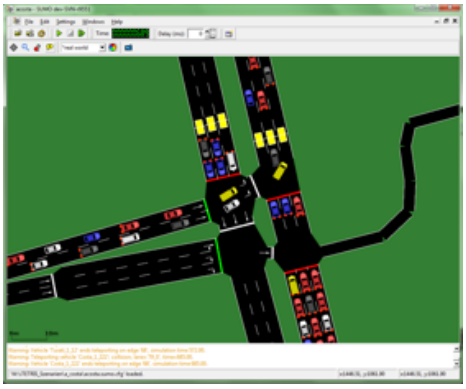
Celkově je to možná varianta detekce vozidel, ale není ideální, sice dokáže detekovat vozidla ve více pruzích, ale radar nedetekuje stojící vozidla a při instalaci na boku vozovky může docházet k blokování, což znamená, že nemusí zachytit úplně všechny vozidla.

Mikrovlnné radary jsou necitlivé ke špatnému počasí, mohou fungovat ve dne i v noci, může poskytovat měření rychlosti a při správném nastavení dokáže detekovat vozidla ve více jízdních pruzích.

Většinou se používají v kombinaci s nějakou jinou technologií, aby se spojily klady a společně odstranily mínusy. [19]

2.3 Realizační prostředky a metody

2.3.1 SUMO



Obrázek 2.13: Software SUMO (převzato z http://sumo.sourceforge.net/)

Neboli "Simulation of Urban MObility"je software pro simulaci silničního provozu. Dovoluje nám sledovat aktuální provoz v síti předem vytvořených silnic. Každé vozidlo v síti je modelováno zvlášť, má svou trasu a pohybuje se v síti samostatně. V základu se jedná o deterministické modelování, ale existují možnosti, jak do modelu přidat náhodnost.

Software dokáže zpracovávat velké sítě křižovatek, dokumentace informuje, že až 10 tisíc ulic může být zpracováváno, víceproudé ulice se změnou pruhu, různá pravidla pro průjezd, vysoká rychlost simulace, dokumentace udává až 100000 aktualizací vozidla za vteřinu. [5] Software nabízí simulaci různých typů vozidel.

Tento software se používá v rámci několika národních a mezinárodních výzkumných projektech, například pro předpověď provozu, výběr trasy a přesměrování, vyhodnocení metod sledování provozu nebo simulace automobilové komunikace.

Pro řešení tohoto projektu je software SUMO asi nejvhodnější variantou, ale také tou nejjednodušší variantou, jelikož se jedná software zaměřující se přímo na simulaci křižovatek, dopravních uzlů.

Software SUMO je založený na jazyce C++ a využívá pouze přenosné knihovny, software je volně šiřitelný.

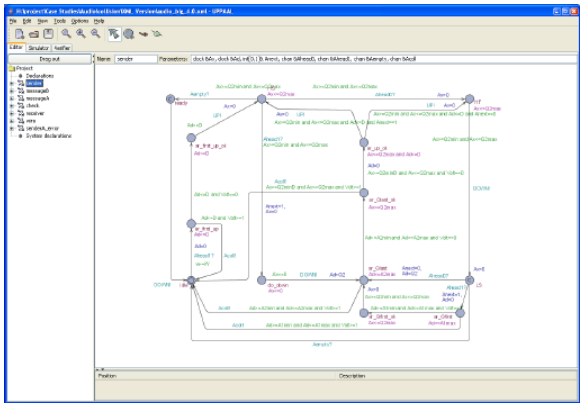
2.3.2 Jazyk C++ s knihovnou SimLib

Jazyk C++ kombinuje vysokou rychlost jazyka C a objektivní programování. Je vhodný pro aplikace vyžadující vysoký výkon. [15]

Knihovna SimLib, volně šířitelná knihovna, je objektově orientovaná knihovna vhodná pro modelování a simulaci v jazyce C++. Od roku 1990 je vyvíjena na Ústavu informatiky a výpočetní techniky FEI VUT Brno. Pomocí této knihovny lze popisovat spojité, diskrétní a kombinované modely. Přímo v jazyce C++ lze popsat simulovaný model, nepotřebuje dodatečný překladač simulačního jazyka. Díky tomuto řešení lze přímo využívat všech ostatních prostředků vytvořených v jazyce C++. Obsahuje nástroje pro sběr informací o chování modelu při simulaci, moduly pro modelování systému s fuzzy popisem. [13]

Pro řešení je tato možnost asi tou nejsložitější, jelikož se jedná o modelování bez nějaké grafické odezvy, to znamená, že model bude popsán textovou formou, následně se spustí simulace a simulace navrátí výsledky opět textovou formou.

2.3.3 UPPAAL



Obrázek 2.14: Software UPPAAL (převzato z http://www.uppaal.org/)

Je software pro modelování časovaných automatu a jejich následnou verifikaci. Slouží pro modelování systému v reálném čase, jenž jsou modelovány jako sítě časovaných automatů, rozšířené o datové typy. Software je vyvíjen ve spolupráci mezi Katedrou informačních technologií na **Upp**sale univerzitě ve Švédsku a Katedrou informatiky na **Aal**borg univerzitě v Dánsku, ze spojení názvů těchto univerzit vznikl název UPPAAL.

Existuje několik verzí tohoto softwaru, uvažovalo se nad klasickou verzí, verzí SMC nebo verzí STRATEGO, tyto verze jsou více popsány v sekci (část 2.3.4). Každá verze rozšiřuje původní software o jiné funkce. Uživatelské rozhraní je v jazyce Java, ale verifikátor je v jazyce C++.

Pro řešení projektu je software UPPAAL asi nejvhodnější variantou, po softwaru SUMO. Nabízí grafické rozhraní, jak pro samotné modelování, tak pro simulaci. Modelování probíhá metodou Drag & Drop (táhni a pusť). Stavy modelu se naklikají myší a přechody se vkládají klikem na počáteční stav a tahem na konečná stav daného přechodu.

Použití softwaru pro studijní účely je zdarma, bez povinnosti užít licenci UPPAALu. Jakékoliv jiné použití vyžaduje použití licence UPPAALu. [11]

2.3.4 Rozšíření UPPAALu

SMC

Jedná se o rozšíření, které přidává k základnímu UPPAALu sadu nástrojů pro kontrolu statistického modelu. Používá techniku, která monitoruje několik běhů systému s ohledem

na určitou vlastnost, následně z této statistiky získá celkový odhad správnosti návrhu či odhad hodnot. Dokáže vypočítat odhad pravděpodobností, odhad očekávaných hodnot, vizualizovat výsledky formou rozdělení pravděpodobností, vývojem počtu běhů s časovými mezemi. [3]

Stratego

Toto rozšíření se zaměřuje na strategie. Usnadňuje vytváření, optimalizaci, srovnávání a také zkoumání důsledků a výkonu strategií. Ve svém konceptu popisuje matematický model systému, skládající se z několika procesů (hráčů) jako hru. Tyto procesy mají nezávislé cíle a často spolu soutěží. Strategie je v tomto případě recept akcí jednoho hráče pro případné situace, vedoucí k dosažení cíle. [10]

CORA

Používá časové automaty s lineární cenou, díky tomu dokáže toto rozšíření nalézt optimální cestu, tedy cestu s nejnižšími akumulovanými náklady. Poskytuje uživateli zprostředkovat další pohled na model, díky tomu lze zvýšit výkonnost. Je možné anotovat model odhadem zbývajících nákladů na dosažení stavu splňujícího cílové podmínky, to má za důsledek, že se může výrazně zkrátit čas potřebný k nalezení dobrého nebo optimální řešení. [2]

TRON

Jedná se o testovací nástroj, vhodný pro testování shody časových systému, zaměřený hlavně na vestavěný software v různých řadičích. Provádí testování shody, důraz je kladen na testování časovaných a funkčních vlastností. [9]

TIGA

Rozšíření implementuje první efektivní on-the-fly algoritmus pro řešení her, založených na časovaných herních automatech s důrazem na dosažitelnost a bezpečnost. Jednotlivé kroky algoritmu jsou prováděny efektivně pomocí tzv. zón jako základní datové struktury. [4]

2.3.5 Použité metody

Modelování a simulace

Pro popis modelování a simulace se musíme nejprve seznámit se souvisejícími pojmy.

Systém - Lze si jej představit jako množinu prvků, které mají mezi sebou určité vazby. Běžně se systém skládá z vícero prvků, ale existují, velmi vzácně, systémy s jedním prvkem. Nejčastěji se rozdělují na reálné systémy a nereálné systémy.

Model - Je napodobenina systému jiným systémem. Platí že každému prvku z modelovaného systému je přiřazen prvek v modelujícím systému. Pokud jsou oba systémy statické, jedná se o statický model. Pro simulační modely musí být splněno, že oba systémy jsou dynamické systémy.

Modelování je následně vytváření modelů systému, ale lze modelovat pouze to, co známe a dokážeme popsat. [14]

"Podstatou modelování ve smyslu výzkumné techniky je náhrada zkoumaného systému jeho modelem (přesněji: systémem, který jej modeluje), jejímž cílem je získat pomocí pokusů s modelem informaci o původním zkoumaném systému." Ivan Křivý a Evžen Kindler, Simulace a modelování, str.15-16 [6]

Simulací získáváme nové znalosti o systému, když budeme experimentovat s modelem systému. Cílem je tedy získání nových informací v závislosti na vstupních hodnotách parametrů.

"Simulace je výzkumná technika, jejíž podstatou je náhrada zkoumaného dyn. systému jeho simulátorem s tím, že se simulátorem se experimentuje s cílem získat informace o původním zkoumaném dynamickém systému." Ivan Křivý a Evžen Kindler, Simulace a modelování, str.17 [6]

Postup při simulaci je následující:

- Nastavení vstupních hodnot parametrů a počátečního stavu modelu
- Případné zadávání dodatečných podnětů z okolí při simulaci
- Vyhodnocení informací o chování systému a výstupních dat

Provádí se více simulačních běhu, dokud nezískáme plnohodnotné informace o chování systému.

Konečné automaty

Konečný automat je výpočetní model primitivního počítače, který se skládá ze stavů, přechodů. Množina stavů je konečná. Nejčastěji se konečné automaty rozdělují na deterministické a nedeterministické. Deterministický konečný automat má pouze jeden počáteční stav a přechodová funkce vrací pouze jeden stav, oproti tomu nedeterministický konečný automat může mít více počátečních stavů a přechodová funkce vrací množinu stavů.

Formálně je konečný automat uspořádaná pětice

$$Q, \Sigma, \delta, g_0, F$$

kde Q je množina stavů, Σ je vstupní abeceda, δ je přechodová funkce, pomocí $\delta : Q \times \Sigma \to Q$ získáme další stav, g_0 je počáteční stav a F je množina koncových stavů. [23]

Časované automaty

Časovaný automat je konečný automat rozšířený o proměnné s hodinami. Všechny hodiny fungují synchronně. Jednotlivé hodiny mohou býti resetovány, ale jejich krokování nemůže být upraveno. Formálně je časovaný automat šestice

$$L, l_0, \Sigma, X, I, E$$

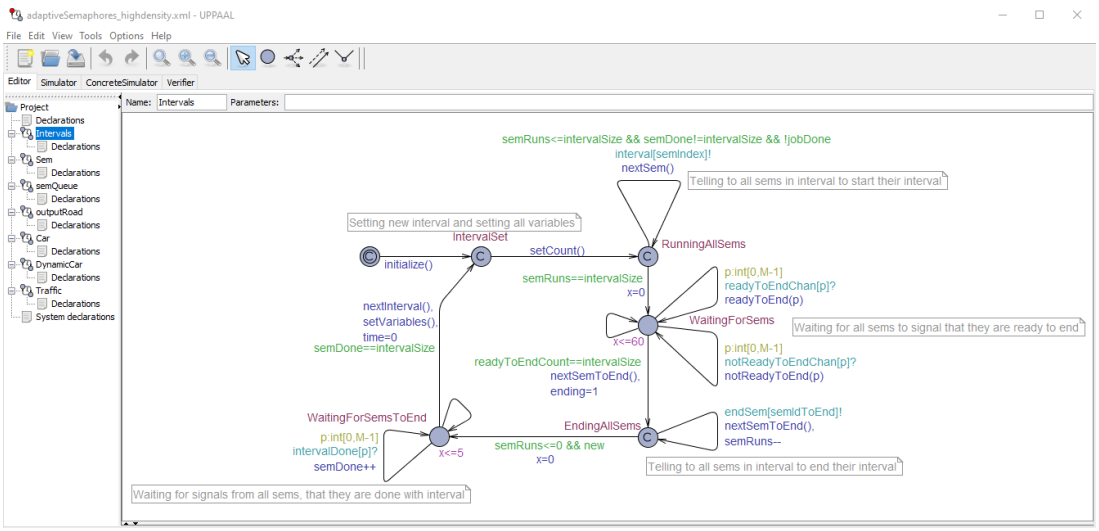
kde L je množina stavů, $l_0 \in$ L je počáteční stav, z požadavků UPPAALu pouze jeden, Σ je množina akcí, X je množina hodin. I udržuje informaci o invariantech ve stavech. E je množina přechodů mezi stavy s akcí, podmínkou přechodu a množinou hodin, které se vynulují. [21]

Invariant je, v kontextu časovaných automatů v UPPAALu, podmínka, která musí být splněna, aby běh automatu mohl v aktuálním stavu počkat nebo vstoupit do dalšího stavu.

2.3.6 Zvolený realizační prostředek

Pro realizaci byl nakonec zvolen software UPPAAL s rozšířením SMC, přesněji zvolil jsem UPPAAL nejvyšší verze 4.1, která je už rozšířena o funkce z SMC rozšíření. [11]

Editor



Obrázek 2.15: Editor v UPPAALu

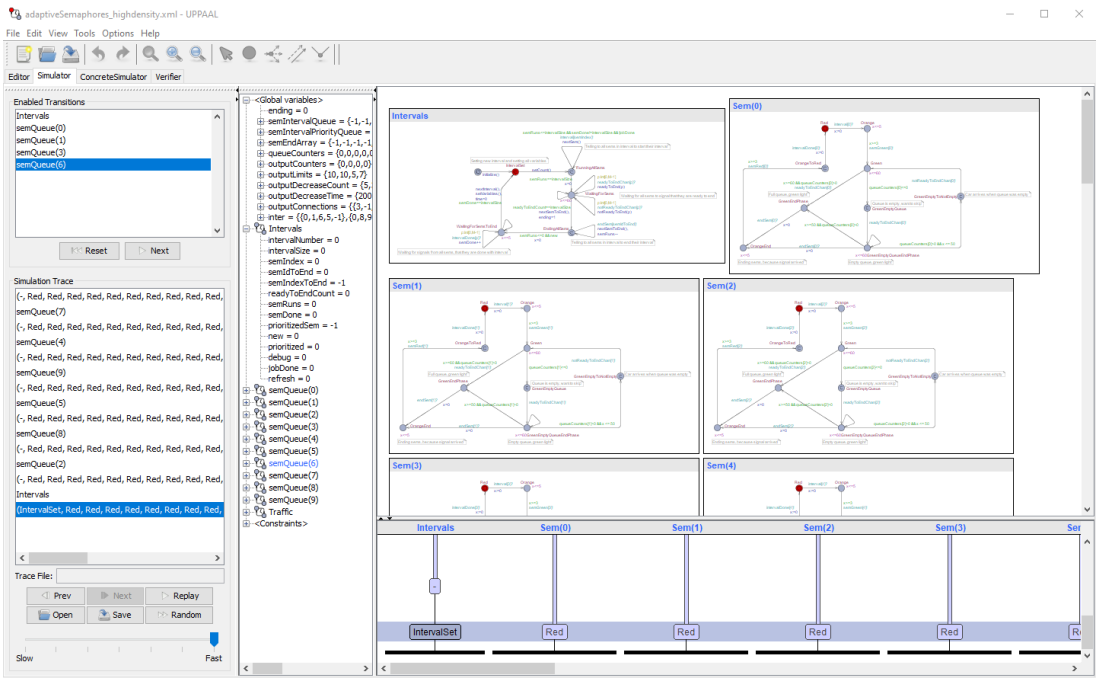
Slouží pro návrh automatů, automaty jsou uložené v šablonách, které se následně volají v systémových deklarací, aby bylo jasné, které automaty jsou součástí aktuálního systému. Při práci v editoru můžeme použít několik funkcí, můžeme vytvořit nový stav, u stavu lze definovat o jaký typ stavu se jedná nebo jak dlouho zde může automat zůstat. Dále u nich lze nastavit, zda se jedná o `urgent` lokaci nebo `committed` lokaci, `urgent` lokace nám zajistí, že jakmile bude stav dostupný, bude využit, `commited` lokace nám zajistí, že jakmile se automat dostane do tohoto stavu, udělají se přednostně přechod z tohoto stavu.

Dále dokážeme definovat přechody mezi stavy, u stavů lze nastavit `select`, `guard`, `sync`, `update`.

- `Select` nám slouží pro deterministický výběr z nějakého intervalu. Pomocí `select`, při správném nastavení kanálů, zjistit který automat zavolal kanál. V této práci je nejčastěji pomocí `select` určováno, který semafor zavolal nebo které vozidlo přijíždí.
- `Guard` udržuje informaci o podmínce, jenž musí být dodržena, aby mohl automat přes tento přechod pokračovat.
- `Sync` slouží pro definici kanálů, jeden přechod se může chovat buď jako příjemce nebo jako vysílač, nikdy jako oboje. Když je přechod nastaven jako příjemce, automat nemůže pokračovat tímto přechodem, pokud nepřišla výzva z nastaveného kanálu. Pokud je nastaven jako vysílač, musí být dodržena pouze `guard`, automat vyšle výzvu nastaveným kanálem.
- Do `update` lze definovat chování při aktivním přechodu, nejčastěji volání deklarovaných funkcí nebo jen úprava proměnné.

Dále pomocí editoru lze vytvářet místa, kde se chování rozděluje a výběr přechodu je určen pomocí pravděpodobnostních vah na přechodech vycházejících z tohoto místa. V této práci je funkce nevyužita. Ke všem těmto prvkům, které lze přidat do automatu, můžeme přidat i komentář, ten je poté vidět, jak na horním obrázku, napsán šedou barvou.

Simulátor



Obrázek 2.16: Simulátor v UPPAALu

V simulátoru lze systém testovat, jsou zde zobrazeny všechny prvky definované v systémových deklaracích. Lze definovat více stejných automatů, jen s jiným označením, toto je v této práci využito pro použití více semaforů.

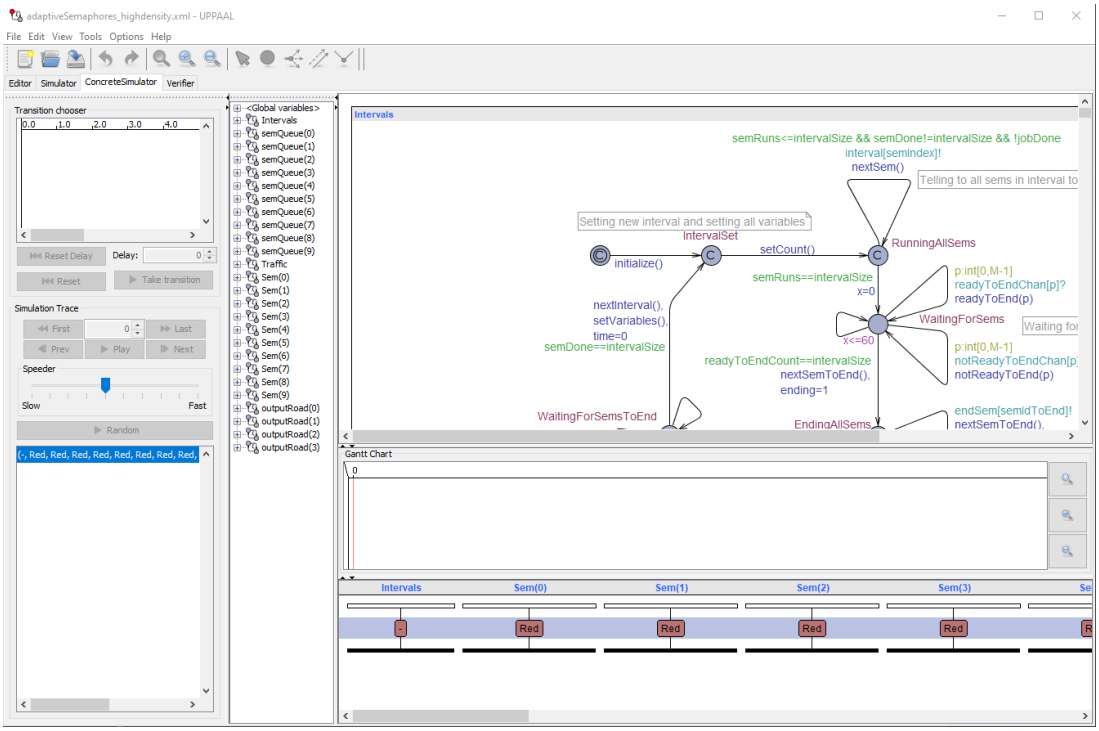
Na obrázku vlevo nahoře lze vybírat z možných přechodů, který se má provést, při označení přechodu, se modrou barvou vyznačí, které proměnné budou ovlivněny. Při kliknutí na tlačítko `Next` pod výběrem se přechod provede. Vedlejším tlačítkem `Reset` se průběh zničí a bude nastavena počáteční pozice systému.

Pod výběrem lze sledovat a přepínat v historii systému. Položky v historii jsou postupně, přechod, poté aktuální stav systému. Pod historií lze historii uložit nebo načíst, následně v historii procházet. Je zde i tlačítko `Random`, to spustí náhodné volení přechodů, posuvníkem pod tlačítky lze upravit rychlost.

Ve druhém sloupci jsou zobrazeny všechny proměnné v systému, velmi příjemná věc při hledání nějakého problému. Jsou zde zobrazeny i časy v jednotlivých automatech.

Ve pravé části lze pozorovat všechny automaty obsažené v systému. Červenou barvou je označený aktuální stav nebo označený přechod. Ve spodní části lze pozorovat aktuální pozice ve všech automatech, je to vlastně to stejné, co každá druhá položka v historii přechodů.

Simulátor s časem



Obrázek 2.17: Simulátor s časem v UPPAALu

V tomto simulátoru lze simulovat systém s náhledem na čas. Jinak se simulátor chová a ovládá úplně stejně jako simulátor bez náhledu na čas.

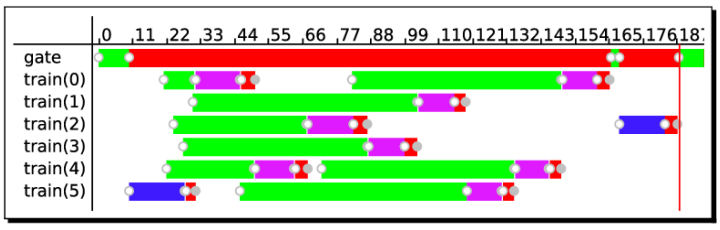
Vlevo při výběru přechodu lze nastavovat zpoždění těchto přechodů. Už při výběru přechodu je vidět, jaký bude mít zvolený přechod vliv na celkový čas. Pod výběrem je opět historie přechodů, ve které lze opět procházet. Nad historií je ještě automatický náhodný výběr přechodů s posuvníkem ovládající rychlost simulace.

Ve druhém sloupci rozložení je opět náhled na všechny systémové proměnné.

Na pravé straně je v horní části opět náhled na všechny části systému s vyznačením aktuálního stavu automatů a zvýrazněným vybraným přechodem. V prostřední části je náhled na časový graf. Pro zobrazení je nutné předchozí nastavení v systémových deklaracích.

```
gantt {
  gate:
    Gate.Occ -> 0,  // red
    Gate.Free -> 1; // green
  train(i:id_t):
    Train(i).Appr -> 2,  // blue
    Train(i).Stop -> 1,  // green
    Train(i).Start -> 3, // magenta
    Train(i).Cross -> 0; // red
}
```

(a) Definition in System declarations.

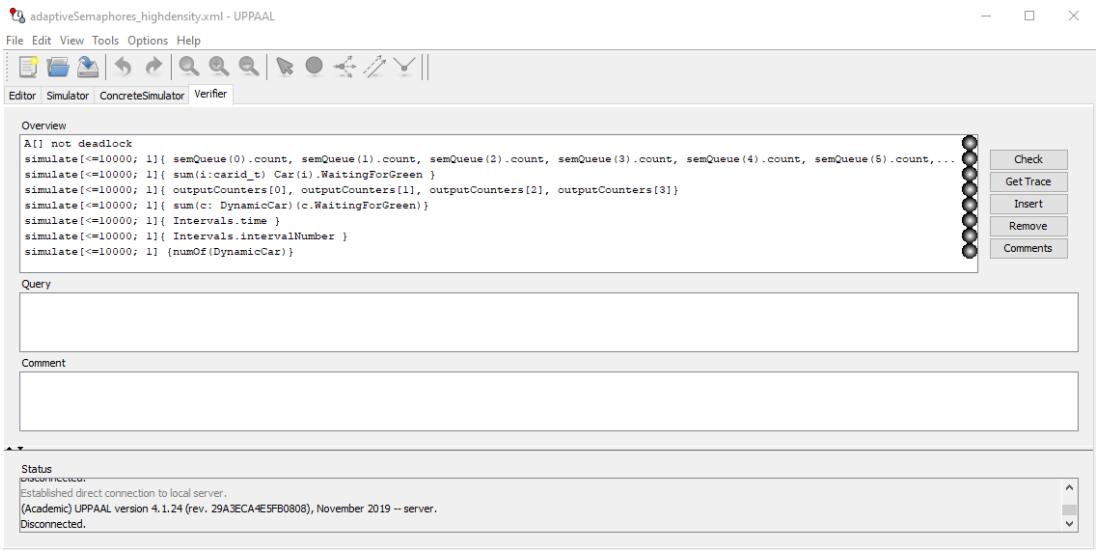


(b) Trace visualization in Concrete Simulator.

Obrázek 2.18: Časový graf (převzato z https://www.it.uu.se/research/group/darts/papers/texts/uppaal-smc-tutorial.pdf)

Ve spodní části jsou zobrazeny opět aktuální stavy všech automatů s názvy stavů.

Verifikátor



Obrázek 2.19: Verifikátor v UPPAALu

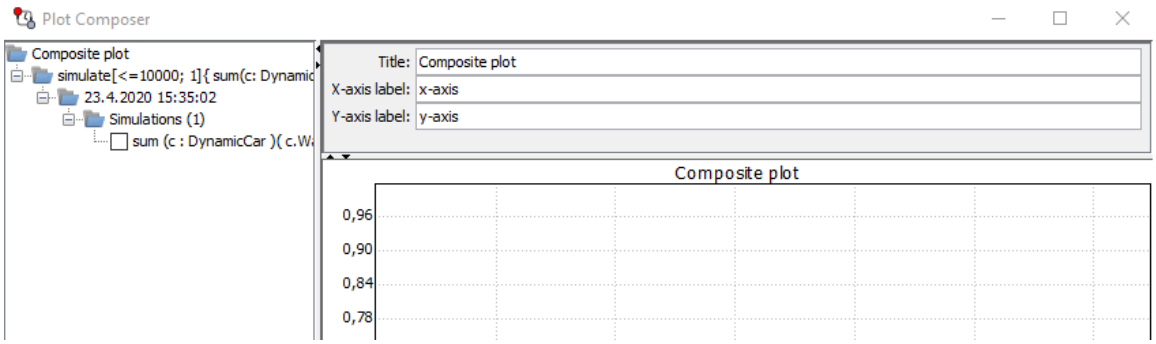
Poslední částí je verifikátor. Slouží k ověřování a testování systému za pomocí dotazu pomocí speciální syntaxe. Syntaxe je popsána v softwarové dokumentaci.

Lze ověřovat, zda systém nemůže uváznout, zda může nějaký stav nastat nebo zda stav může být proveden po jiném stavu a podobné. SMC rozšíření k těmto základním dotazům přidává i dotazy na pravděpodobnost, výpočet předpokládaných hodnot proměnných nebo vykreslení grafu ze sledovaných hodnot. [1]

Testování pravděpodobnosti může mít 3 varianty výpočtu. Přímý kalkul pravděpodobnosti nějakého jevu, vrací hodnotu. Testování hypotézy, například: je pravděpodobnost nějakého jevu větší než nějaký limit, vrací úspěšnost dotazu. A nakonec porovnání pravděpodobností, například: je pravděpodobnost nějakého jevu větší než pravděpodobnost jiného jevu.

Výpočet předpokládaných hodnot probíhá z více spuštění systému, počet spuštění se nastaví v dotazu. Z výpočtu lze získat například minimum a maximum sledovaných hodnot.

Při vykreslování grafu ze sledovaných hodnot lze také použít možnost více spuštění, graf se poté bohužel stává nečitelným. Na úpravu grafu lze využít vestavěnou funkci nazvanou `Plot composer`, ta nám dovoluje upravit graf a následně exportovat do různých formátů.



Obrázek 2.20: Plot composer

V levé části můžeme vybrat jaké hodnoty se mají zobrazit v grafu, při jejich výběru se vždy v horní části zobrazí textová pole, které lze upravit a podle toho se bude upravovat graf, na obrázku si můžeme všimnout, základního nastavení, to je titulek grafu, pojmenování os.

2.3.7 Zajímavosti z používání UPPAALu

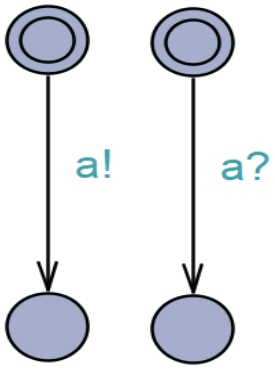
Synchronizace skrze kanály

Automaty v UPPAALu lze synchronizovat pomocí kanálů, tyto kanály se deklarují v globální deklaraci. Jsou tři typy těchto kanálů, klasický, všesměrový a urgentní. [8]

- Klasické
  Mějme příklad, kdy máme nastavenou synchronizaci kanálem označeným `a`. Na obrázku vlevo vidíme přechod s vysílačem do tohoto kanálu, označený vykřičníkem.
  - <jméno kanálu>!

  Vpravo je přechod s přijímačem toho kanálu, tentokrát označený otazníkem.
  - <jméno kanálu>?

  Pokud se provede levý přechod, informuje o tom pravý přechod, který se touto akcí provede. Při použití klasických kanálů je příjemce signálu vybrán nedeterministicky.



Obrázek 2.21: Synchronizační kanály v UPPAALu

- Všesměrové

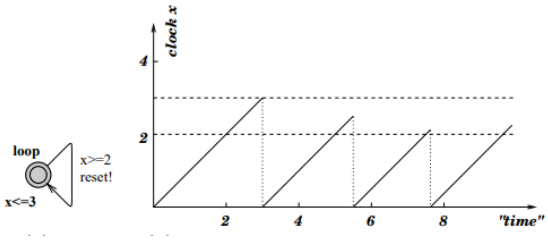
  Při deklaraci se před samotnou deklaraci přidá klíčové slovo `broadcast`. Toto provedení nám vytvoří všesměrový kanál a poté ho lze používat stejně jako klasické, jen s tím rozdílem, že se při vícero přijímačů nevybírá nedeterministicky, ale pošle se signál všem a všechny dostupné akce se provedou.

- Urgentní

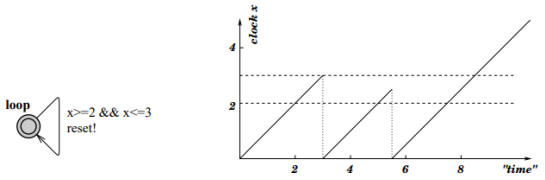
  Ty se deklarují klíčovým slovem `urgent` před samotnou deklaraci kanálu. Tato synchronizace probíhá podobně jako urgentní přechody, pokud je synchronizace dostupná, musí se provést ihned. To znamená bez jakéhokoliv zpoždění.

Čas při používání invariantů a podmínek přechodů

Čas je v těchto automatech velmi důležitý, při špatném návrhu celého systému může nastat, že celý systém uvázne.



Obrázek 2.22: Použití invariantu (převzato z https://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf)



Obrázek 2.23: Použití pouze podmínky (převzato z https://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf)

Na levém obrázku si lze povšimnout návrhu s použitím podmínky přechodu pro ohraničení ze spodní strany, to je více jak 2 časové jednotky a horní časová hranice je vytvořena za použití invariantu, to nám zaručí, že nejzazší čas, kdy bude provedena nějaká akce ze stavu jsou 4 časové jednotky. Toto je správný způsob, jak zaručit, že systém neuvázne. [8]

Na pravém obrázku je vidět použití pouze podmínky přechodu, bez použití invariantu. To nám způsobí, že když se systém někde zdrží a hodiny budou už na 5. časové jednotce, bude přechod klasifikován jako nedostupný, není splněna podmínka přechodu. Stav nemá jiný dostupný přechod, systém uvázl.

Kapitola 3

Realizace modelu adaptivních semaforů

3.1 Návrh řešení

Při navrhování výpočtového modelu a jednotlivých křižovatek, jsem se řídil silničním zákonem, aby jakákoliv část neporušovala pravidla silničního provozu.

Silniční zákon o světelné signalizaci

- (1) Při řízení provozu na křižovatce se užívá zejména světelných signálů tříbarevné soustavy s plnými signály nebo se směrovými signály.
- (2) Při řízení provozu na křižovatce znamená pro řidiče
  - a) signál s červeným světlem "Stůj!"povinnost zastavit vozidlo před dopravní značkou "Příčná čára souvislá", "Příčná čára souvislá se symbolem Dej přednost v jízdě!"a "Příčná čára souvislá s nápisem STOP", a kde taková dopravní značka není, před světelným signalizačním zařízením,
  - b) signál se současně svítícím červeným a žlutým světlem "Pozor!"povinnost připravit se k jízdě,
  - c) signál se zeleným plným kruhovým světlem "Volno"možnost pokračovat v jízdě, a dodrží-li ustanovení o odbočování, může odbočit vpravo nebo vlevo, přičemž musí dát přednost chodcům přecházejícím ve volném směru po přechodu pro chodce a cyklistům přejíždějícím ve volném směru po přejezdu pro cyklisty. Svítí-li signál "Signál pro opuštění křižovatky"umístěný v protilehlém rohu křižovatky, neplatí pro odbočování vlevo § 21 odst. 5,
  - d) signál se žlutým světlem "Pozor!"povinnost zastavit vozidlo před dopravní značkou "Příčná čára souvislá", "Příčná čára souvislá se symbolem Dej přednost v jízdě!"a "Příčná čára souvislá s nápisem STOP", a kde taková dopravní značka není, před světelným signalizačním zařízením; je-li však toto vozidlo při rozsvícení tohoto signálu již tak blízko, že by řidič nemohl vozidlo bezpečně zastavit, smí pokračovat v jízdě. Svítí-li světlo tohoto signálu přerušovaně, nejde o křižovatku s provozem řízeným světelnými signály,
  - e) signál se zelenou směrovou šipkou nebo šipkami (například "Signál pro přímý směr", "Kombinovaný signál pro přímý směr a odbočování vpravo") možnost

pokračovat v jízdě jen ve směru, kterým šipka nebo šipky ukazují. Směřuje-li zelená šipka vlevo, neplatí pro odbočování vlevo § 21 odst. 5,

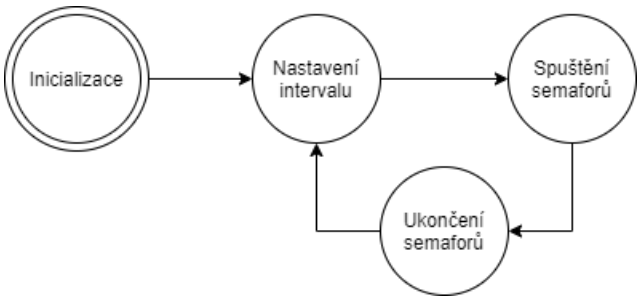
- f) signály „Signál žlutého světla ve tvaru chodce“, „Signál žlutého světla ve tvaru cyklisty“ nebo „Signál žlutého světla ve tvaru chodce a cyklisty“, jimiž je doplněn signál se zelenou šipkou směřující vpravo nebo vlevo, upozorňují řidiče, že při jízdě směrem, kterým tato šipka ukazuje, křižuje směr chůze přecházejících chodců, směr jízdy přejíždějících cyklistů nebo směr chůze přecházejících chodců a směr jízdy přejíždějících cyklistů,
- g) signál "Doplňková zelená šipka"svítící současně se signálem s červeným světlem "Stůj!"nebo se žlutým světlem "Pozor!"možnost pokračovat v jízdě jen ve směru, kterým šipka nebo šipky ukazují; přitom řidič musí dát přednost v jízdě vozidlům a jezdcům na zvířatech jedoucím ve volném směru a útvarům chodců jdoucím ve volném směru; přitom nesmí ohrozit ani omezit přecházející chodce.

- (3) Při řízení provozu mimo křižovatku, například před přechodem pro chodce nebo před nepřehledným místem, platí obdobně odstavec 2.

Převzato z https://www.mesec.cz/zakony/zakon-o-silnicnim-provozu/f2084807/

Celkový model adaptivních semaforů se skládá ze čtyřech automatů řídící samotné adaptivní semafory a dále automat pro řízení provozu. Hlavní automat řídí samotné intervaly semaforu, dává signál jednotlivým semaforům, kdy se mají spustit, čeká na signály od semaforů, že jsou připravené ukončit svou činnost, následně posílá ukončovací signály semaforům. Dalším je automat řídící samotné semafory, ten čeká na signál, aby se mohl spustit, pokud je spuštěn, dává signály automatu ovládající frontu, jestli může pouštět vozidla a posílá signály automatu řídící intervaly, že je připraven ukončit sekvenci. Automat ovládající frontu na semafory, komunikuje se semaforem jemu přiděleném a s přijíždějícími vozidly, pokud si jej vybrali. Automat ovládající vozidlo si vybere křižovatku a z tohoto výběru si následně ještě vybere výjezdovou silnici, kterou bude chtít pokračovat. Dává signál frontě, že přijíždí a čeká na signál od fronty, že může pokračovat. Nakonec automat ovládající výjezdové silnice, ten pouze upravuje počty vozidel ve výjezdových pruzích. Provoz vozidel lze spustit za pomocí dynamického přístupu nebo statického přístupu.

Automat ovládající intervaly

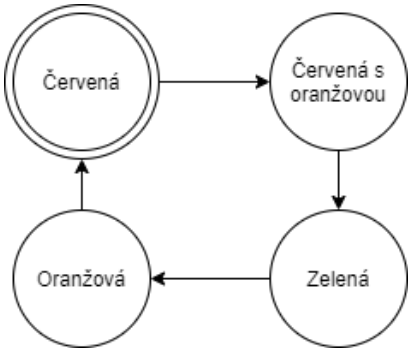


Obrázek 3.1: Základ pro automat ovládající intervaly

Jako základ pro vytváření automatu ovládající intervaly byl použit tento jednoduchý automat, který je vy výsledku rozšířen o dva stavy, ve kterých se automat může nacházet. Ale u obou stavů se dá říci, že jsou spíše pomocné pro kontrolu stavu křižovatky. Ve

stavu `Inicializace` se nastaví všechny počáteční hodnoty a konstanty, ve stavu `Nastavení intervalu` se vybere následující interval, buď jako u klasických semaforů pouze následující v seznamu intervalů nebo u adaptivních se vybere podle různých kritérií nejvíce vyhovující. Automat po nastavení může přejít do stavu `Spouštění semaforů`, ve které spustí všechny semafory z aktuálního intervalu. Nyní může automat přejít do stavu `Ukončení semaforů`, kde buď čeká na pevný časový interval nebo na předčasné ukončení všech semaforů. V samotné realizaci (část 3.2.1) je tento automat rozšířen, ale základem je právě tento automat.

Automat ovládající semafory



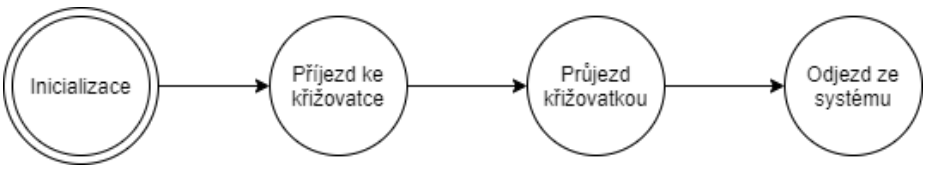
Obrázek 3.2: Základ pro automat ovládající semafory

Tento automat je ve výsledné realizace (část 3.2.2) doplněn o informační stavy pro celou křižovatku, ale jinak je tento základ součástí výsledného automatu. Jediný stav, který není obsažen ve výsledné realizaci je stav `Červená s oranžovou`, protože tento stav pro automat nic speciálního nedělá a během tohoto stavu se vozidla stejně jako ve stavu `Zelená` rozjíždí a pokračují do křižovatky, takže funkce tohoto stavu je vlastně obsažena ve stavu `Zelená`.

Automat ovládající frontu u semaforu

Tento semafor neměl původně ani existovat, bylo v plánu implementovat pouštění vozidel přímo do pozadí automatu ovládající semafor, ale nakonec se implementace nevydařila a vznikl speciální automat řídící frontu u jednotlivých semaforů. Nakonec se ukázalo, že je tahle varianta nejen jednodušší, ale nakonec i lepší.

Automat ovládající vozidlo



Obrázek 3.3: Základ pro automat ovládající vozidlo

Tento automat je základem, jak pro vozidlo generované staticky, tak i dynamicky. Při generování vozidla staticky je zde přidána možnost vrátit se zpět do křižovatky ze stavu `Odjezd`

`ze systému` zpět do stavu `Příjezd ke křižovatce`. V samotné realizaci (části 3.2.5, 3.2.6) je tento automat upraven, tak aby fungoval správně s automatem ovládajícím frontu u semaforů. Jinak je automat založen na jednoduchosti, vozidlo si ve stavu `Inicializace` vybere křižovatku, přejde do stavu `Příjezd ke křižovatce`, kde čeká než bude moct pokračovat, až mu bude průjezd povolen přejde do stavu `Průjezd křižovatkou`, kde setrvá než projede křižovatkou, poté pokračuje do stavu `Odjezd ze systému` odkud se buď vrátí do systému, ve staticky generovaném systému, nebo opouští systém.

3.2 Popis řešení

Model lze nastavit pomocí globálních deklarací. Lze nastavit počet vozidel cyklujících v okolí křižovatky nebo počet vozidel generovaných při dynamickém vytváření provozu. Jako definici křižovatky lze nastavit počet semaforů, počet intervalů a počet výjezdových silnic. Pro definici intervalů je použito dvourozměrné pole s definovanou velikostí, v poli je uložena informace, v jakém intervalu se pouští, které semafory.

```
const int N = 50;          // # cars
typedef int[0,N-1] carid_t;

const int M = 10;          // # semaphores
typedef int[0,M-1] semid_t;

const int I = 4;           // # intervals
typedef int[0,I-1] intervalid_t;

const int J = 4;           // # output roads
typedef int[0,J-1] outputid_t;
```

Obrázek 3.4: Definice počtu instancí

```
int[-1,M-1] inter[I][interval_size] = { {0,1,6,5,-1}, //pink
                                        {3,4,5,-1,-1}, //yellow
                                        {0,8,9,-1,-1}, //green
                                        {2,7,-1,-1,-1} }; //brown
```

Obrázek 3.5: Definice intervalů

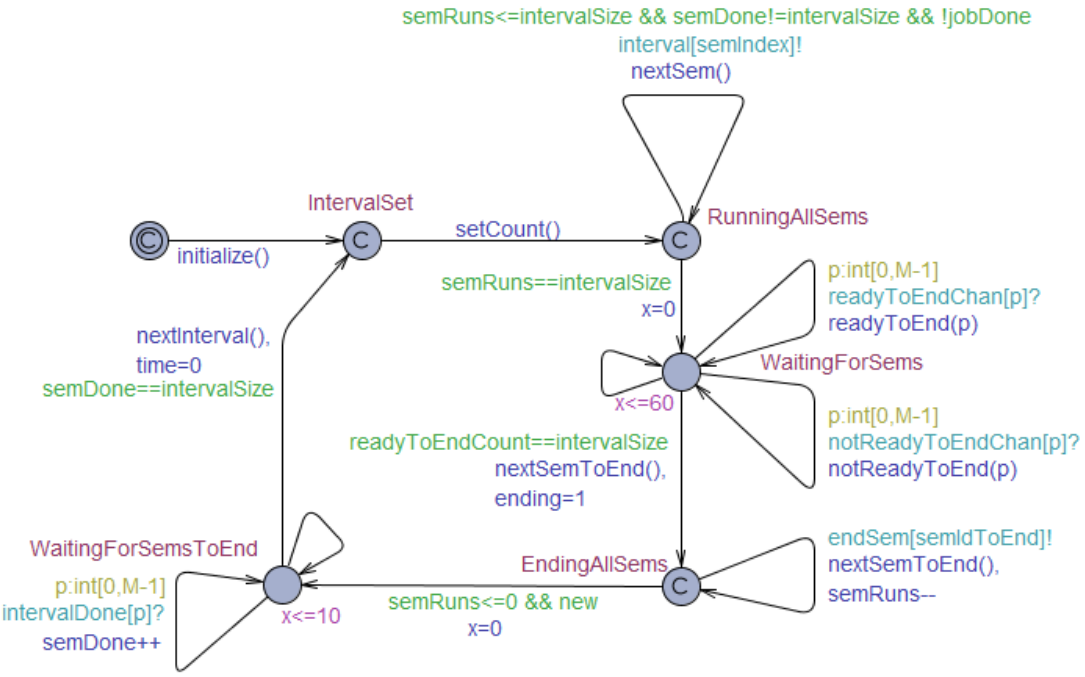
Pro definici výstupních silnic je použito dvourozměrné pole o velikosti počtu semaforů, kde pro každý semafor je definováno, kam může vozidlo pokračovat. Pro každou výjezdovou silnici je definován její limit vozidel, jak často se na ní udělá místo pro další vozidla, počet vozidel, které opustí výjezdovou silnici. Lze si to představit, tak že za nějakou dobu odjede nějaký počet vozidel.

```
int[0,10] outputCounters[J] = {0,0,0,0};
int[0,10] outputLimits[J] = {10,10,5,7};
int[1,10] outputDecreaseCount[J] = {5,3,6,2};
int[1,100] outputDecreaseTime[J] = {20,30,20,40};
int[-1,J-1] outputConnections[M][3] = {{3,-1,-1},{2,-1,-1},{1,-1,-1},{0,3,-1},{2
```

Obrázek 3.6: Definice výjezdových silnic

3.2.1 Automat řídicí intervaly

Automat začíná inicializací front na počáteční hodnoty, inicializuje se zde fronta pro ukládání semaforů, které jsou připravené ukončit svou činnost, fronta pro semafory žádající o spuštění a prioritní fronta pro semafory s vyšším počtem vozidel.



Obrázek 3.7: Automat s intervaly

Nyní se automat nachází ve stavu `IntervalSet`, který má pouze jeden výstup, takže ten se provede, na této hraně se nastaví do proměnné počet semaforů, se kterými se bude v tomto intervalu pracovat.

```
void setCount()
{
    int i;
    for(i = 0; i < interval_size; i++)
    {
        if(inter[intervalNumber][i]!=-1)
        {
            intervalSize++;
        }
    }
}
```

Obrázek 3.8: Funkce setCount()

Funkce `setCount()` vezme z globální proměnné, jaký interval je na řadě a cyklem s předem daným počtem opakování, podle možné velikosti intervalu, testuje zda prvek je

označení semaforu nebo se jedná o nedefinovaný prvek, pro jednoduchost označený -1. Nastavuje globální proměnnou `intervalSize`, která se používá v dalších funkcích.

Automat se nyní přesunul do stavu `RunningAllSems`, kde má za úkol dát signál všem semaforům z aktuálního intervalu. Automat může pokračovat pouze až dá signál všem semaforům, to je hlídáno funkcí `nextSem()`.

```
void nextSem()
{
    semRuns++;
    if(inter[intervalNumber][semRuns]!=-1)
    {
        semIndex = inter[intervalNumber][semRuns];
    }
    else
    {
        jobDone = 1;
    }
    new = 1;
}
```

Obrázek 3.9: Funkce nextSem()

Ve funkci `nextSem()` se zvýší proměnná `semRuns`, která udržuje informaci, kolik semaforů právě běží. Dále se buď nastaví index semaforu (`semIndex`) nebo proměnná `jobDone`, která symbolizuje, že byly spuštěny všechny semafory, podle toho, zda v intervalu se nachází nějaký nespuštěný semafor. Proměnná `new` je zde pouze za účelem, aby nedošlo k přeskočení semaforu, při vypínání semaforů.

Po spuštění všech semaforů přejde automat do stavu `WaitingForSems`, kdy tento automat čeká na automaty ovládající semafory, než dokončí svou činnost. Semafory se dokážou přihlásit i odhlásit z ukončování. Toto je využitelné, když semafor nahlásí, že je schopen ukončit svou činnost, ale ještě předtím, než to nahlásí všechny semafory, tak přijede vozidlo k semaforu, který má už nahlášeno, že může ukončit, proto se odhlásí z ukončování intervalu.

```
void readyToEnd(int p)
{
    int m = 0;
    if (exists(i:range_endArray) semEndArray[i]!=p) {
        readyToEndCount++;
        while (semEndArray[m]>=0){m++;};
        semEndArray[m]=p;
    }
}
```

Obrázek 3.10: Funkce readyToEnd()

Funkce `readyToEnd()` s parametrem `p`, kde tento parametr obsahuje informaci o semaforu, vezme tento parametr a porovná ho s prvky v poli `semEndArray`, pokud jej nenalezne, zvýší proměnnou `readyToEndCount`, která uchovává informaci o počtu semaforů připravených ukončit své konání. Na konec pole `semEndArray` se vloží semafor, který spustil tuto funkci.

```
void notReadyToEnd(int p)
{
    int m = 0;
    readyToEndCount--;
    while (semEndArray[m]!=p){m++;};
    semEndArray[m]=-1;
}
```

Obrázek 3.11: Funkce notReadyToEnd()

Funkce `notReadyToEnd()` s parametrem `p` je přesný opak funkce `readyToEnd()`, slouží tedy k odhlašování semaforů. Nejprve se sníží počet semaforů, připravených ukončit svou činnost. Následně se nalezne semafor v poli `semEndArray` a vymaže se odtud. Vymazání se zde řeší přepsáním na -1, jelikož semafory jsou číslovány od 0.

Až jsou všechny semafory připraveny ukončit interval, automat řídící intervaly přejde do stavu `EndingAllSems`, v tomto stavu nahlásí všem semaforům z aktuálního intervalu, aby ukončili svou činnost.

```
void nextSemToEnd()
{
    if(semIndexToEnd!=-1){
        semEndArray[semIndexToEnd]=-1;
    }
    semIndexToEnd++;
    if(semEndArray[semIndexToEnd]==-1)
    {
        semIndexToEnd = -1;
        semIdToEnd = 0;
    }
    else
    {
        semIdToEnd = semEndArray[semIndexToEnd];
    }
}
```

Obrázek 3.12: Funkce nextSemToEnd()

Funkce `nextSemToEnd()` slouží pro přepnutí na další semafor, který má být vypnut. Nejprve se najde aktuální semafor a vymaže se z pole `semEndArray`, vymazání se provede nastavením na -1. Dále se zvýší index `semIndexToEnd`, pokud aktuální index ukazuje na místo v poli `semEndArray`, kde je nastavena -1, ukončování dorazilo na konec tohoto pole. Index se nastaví na -1, pro symbolizování konce. Pokud není na konci pole, nastaví `semIdToEnd`, proměnná, ze které se bere index semaforu k ukončení.

Když mají všechny semafory nahlášeno, že mají ukončit svou činnost, automat řídící intervaly přejde do stavu, kde čeká na reakci od semaforů, že ukončily své intervaly. Až se všechny semafory ukončí, tak automat řídící intervaly přejde do stavu `IntervalSet`, při tomto přechodu se pomocí funkce `nextInterval()` nastaví další interval.

```
void nextInterval()
{
    if(!semIntervalPriorityQueueEmpty() && !prioritized){ //if sem in prioritized queue
        int i, j = 0;
        bool found = 0;
        for(i = M-1; i >= 0; i--){
            if(semIntervalPriorityQueue[i]!=-1){prioritizedSem = semIntervalPriorityQueue[i];}
        }
        while(found == 0){
            intervalNumber++;
            if(intervalNumber >= I){intervalNumber = 0;}
            for(j = 0; j < interval_size; j++){
                if(inter[intervalNumber][j]==prioritizedSem){
                    found = 1;
                }
            }
        }
        prioritized = 1;
    }
    else if(!semIntervalQueueEmpty()) //if sem in sem queue
    {
        int i, j;
        bool found = false;
        while(found==false)
        {
            intervalNumber++;
            if(intervalNumber >= I){
                intervalNumber = 0;
                prioritized = 0;
            }
            for(i = 0; i < interval_size; i++)
            {
                for(j = 0; j < M; j++)
                {
                    if(inter[intervalNumber][i]==semIntervalQueue[j] && inter[intervalNumber][i]!=-1)
                    {
                        found = true;
                    }
                }
            }
        }
    }
    else //if sem queues empty
    {
        intervalNumber++;
        if(intervalNumber >= I){
            intervalNumber = 0;
            prioritized = 0;
        }
    }
}
```

Obrázek 3.13: Funkce nextInterval()

Funkce `nextInterval()` slouží pro výběr, které semafory se mají právě spustit. Má tři rozdílné možnosti chování, první zkontroluje, jestli je nějaký prvek v poli `semIntervalPriorityQueue`, které by mělo obsahovat semafory, které jsou více žádané, to znamená, má ve frontě čtyři a více vozidel. S tímto polem zkontroluje i zda nebyl v tomto cyklu takto vybírán semafor, to je zde, aby nedocházelo k vyhladovění méně vytížených semaforů. Pokud bude zvolena tato možnost, semafor vezme z pole první semafor, a poté cyklem bez předem daným počtem prvků, s postupným zvyšováním indexu do pole s intervaly, prohledává pole s intervaly. Až nalezne interval s hledaným semaforem, ukončuje

cyklus. V proměnné `intervalNumber` je index do pole s intervaly, kde se nachází hledaný semafor, a nastaví se proměnná `prioritized`, aby už nemohla být využitelná tato možnost.

Druhá možnost volby, se řídí podle toho, jestli je nějaký semafor v `semIntervalQueue`. Možnost začíná opět cyklem bez předem daným počtem opakování, opět se zvyšujícím indexem do pole s intervaly. V každém cyklu se zkontroluje, zda v aktuálním intervalu se nenachází semafor ze `semIntervalQueue`, pokud nalezne shodu, ukončuje se cyklus a v `intervalNumber` se nachází index do pole s intervaly.

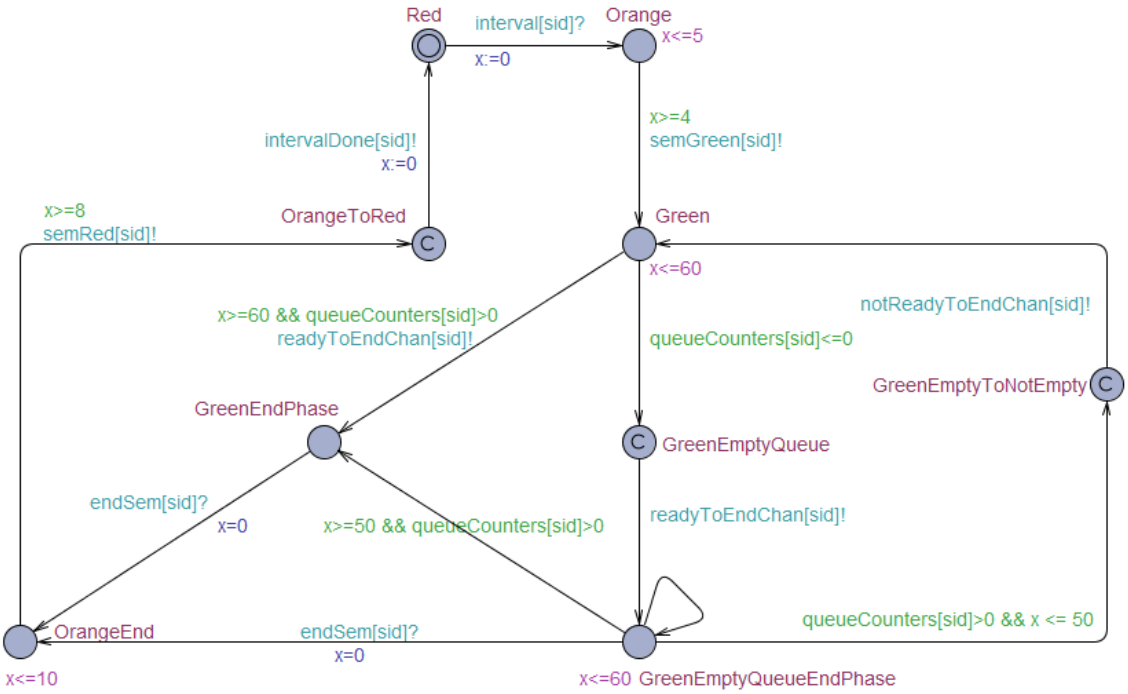
Poslední možností je jednoduché zvýšení indexu do pole s intervaly. Touto možností se křižovatka chová jako bez adaptivních funkcí.

Ve všech možnostech při zvyšování indexu do pole s intervaly, se kontroluje, zda index nepřesáhl pole, když tak nastane, index se nastaví na 0 a proměnná `prioritized` se nastaví na 0, to znamená může opět nastat první možnost.

Další interval se vybírá podle aktuální situace, pokud je v prioritní frontě nějaký semafor a nebyl v tomto cyklu ještě upřednostněn žádný semafor, tak se vybere následující interval, který obsahuje právě tento semafor. Pokud je nějaký semafor ve frontě, tak se vybere interval, který obsahuje alespoň jeden semafor, který má čekající vozidlo. Když budou obě fronty prázdné, vybere se následující interval. Tímto stavem se celý automat stále opakuje.

3.2.2 Automat řídící semafory

Tento automat se vytvoří pro každou možnost průjezdu v křižovatce, to znamená, že pokud jsou z jednoho směru 3 možnosti kam se zařadit podle odbočení, vytvoří se 3 tyto automaty.



Obrázek 3.14: Automat pro semafor

Všechny automaty, které řídí semafory, tak začínají ve stavu `Red`, takže je křižovatka v neprůjezdném stavu a čeká se na automat řídicí intervaly, aby dal signál některému nebo

některým ze semaforů. Po přijetí tohoto signálu začne semafor svou činnost, přejde do stavu `Orange`. V tomto stavu setrvá 4-5 časových jednotek, poté pokračuje do stavu `Green`, při přechodu do tohoto stavu dá signál automatu řídicí frontu na semafor, že na semaforu je zelená a fronta může být vyklízena.

Ve stavu `Green` zůstane v nejhorším případě 60 časových jednotek, tento stav nastane, pokud jsou ve frontě stále vozidla. Pokud není ve frontě žádné vozidlo, automat přejde do stavu `GreenEmptyQueue` a následně to stavu `GreenEmptyQueueEndPhase`, při přechodu dá tento semafor signál automatu řídicí intervaly, že je schopen ukončit. Nyní může automat pokračovat třemi možnostmi.

1. Do 50 časových jednotek dorazí k tomuto semaforu alespoň jedno vozidlo, automat přejde do stavu `GreenEmptyToNotEmpty` a následně do stavu `Green`, při tomto přechodu je dán signál automatu řídicí intervaly, že tento semafor není schopen nyní ukončit svou činnost.

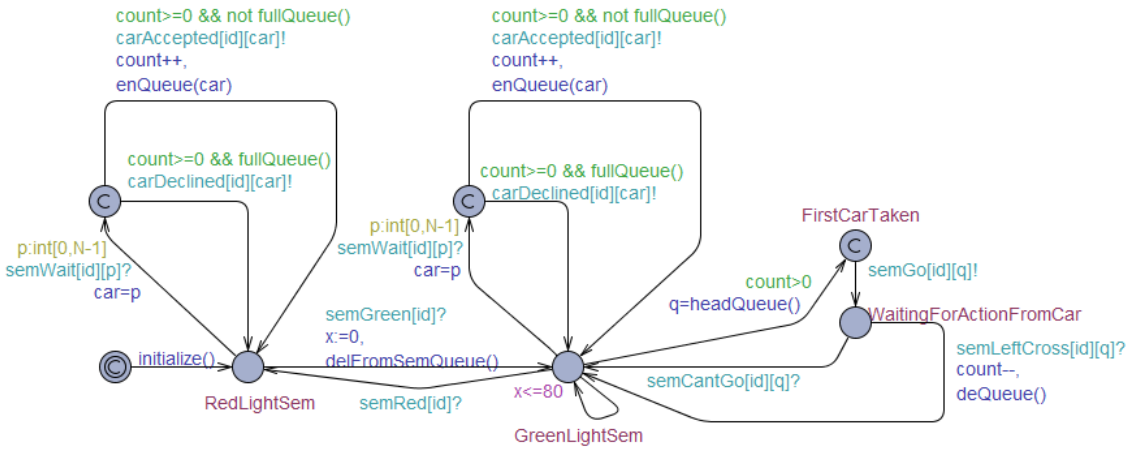
2. Pokud uběhne 50 časových jednotek a před ukončením semaforu přijede vozidlo, je vpuštěno, automat se přesune do stavu `GreenEndPhase` a automat stále čeká na signál od automatu řídicí intervaly.

3. Pokud i ostatní spuštěné semafory nahlásí, že jsou schopny ukončit se, tak automat řídicí intervaly zašle signál semaforům, aby přešly do stavu `OrangeEnd`.

Ve stavu `OrangeEnd` stráví 8-10 časových jednotek, po tomto čase nahlásí automatu řídicí frontu u semaforu, aby ukončil projíždění aut. Nyní se automat nachází ve stavu `OrangeToRed`, ze kterého pokračuje do stavu `Red`, při přechodu do tohoto stavu, dá signál automatu řídicí intervaly, že semafor ukončil svou činnost. Nyní na semaforu svítí červené světlo a čeká se na akci ze strany automatu řídicí intervaly, aby spustil další interval, s tím i semafory.

3.2.3 Automat řídicí frontu u semaforu

Pro každý automat řídicí semafor se vytvoří tento automat, který má na starost vozidla a jejich zařazování do fronty k semaforu.



Obrázek 3.15: Automat řídicí frontu

Automat začíná inicializací, kde se nastaví fronta a pokračuje do stavu `RedLightSem`, kde už přijímá vozidla do fronty, přijímá do plného stavu. Pokud v tomto stavu přijde signál od vozidla, že vozidla přijíždí k semafor, v automatu řídící frontu u semaforu se uloží informace, které vozidlo žádalo a následně se rozhodne, zda je na vozidlo ještě místo nebo musí počkat. Při přijmutí vozidla se semafor přidá do fronty semaforů, ze které se následně vybírá vhodný interval.

```
void enQueue (int p)
{
  int i = 0;
  while (queue[i]>=0){i++;};
  queue[i]=p;
  addToSemQueue();
    queueCounters[id]++;
}
```

Obrázek 3.16: funkce enQueue()

Funkce `enQueue()` s parametrem `p`, který symbolizuje přijíždějící vozidlo, slouží pro zařazení vozidla do fronty. Pomocí cyklu bez předem daného počtu opakování se najde místo ve frontě a vozidlo se tam vloží, zvýší se počítadlo vozidel ve frontě. Následně se zavolá funkce `addToSemQueue()`.

```
void addToSemQueue ()
{
  if (exists(i:range_queue) semIntervalQueue[i]==id) {
      if (count>4) {
        if (!exists(i:range_queue) semIntervalPriorityQueue[i]==id) {
          int k = 0;
          while (semIntervalPriorityQueue[k]>=0){k++;};
          semIntervalPriorityQueue[k]=id;
          k = 0;
          while (semIntervalQueue[k]!=id){k++;};
          semIntervalQueue[k]=-1;
        }
      }
  }
  else {
      int j = 0;
      while (semIntervalQueue[j]>=0){j++;};
      semIntervalQueue[j]=id;
  }
}
```

Obrázek 3.17: Funkce addToSemQueue()

Funkce `addToSemQueue()` slouží pro zařazení vozidlo do fronty semaforů. Prvně se ve funkci zkontroluje, zda už ve frontě semafor není, pokud už ve frontě je, zkontroluje se počet vozidel ve frontě, pokud je ve frontě více jak 4 vozidla, přidá se semafor, za pomocí vyhledání volného místo, do `semIntervalPriorityQueue`. Tímto semafor získá prioritu pro spuštění. Následně se semafor odstraní z neprioritní fronty semaforů (`semIntervalQueue`). Pokud semafor ještě ve frontě semaforů není, tak se přidá na konec neprioritní fronty semaforů.

Pokud přijde signál od semaforu, že na něm je zelená, spustí se pouštění vozidel křižovatkou. Automat přejde do stavu `GreenLightSem`, při tomto přechodu se semafor odstraní z fronty semaforů. V tomto stavu je automat schopen přijímat vozidla, ale hlavně pouští vozidla do křižovatky. Při přechodu do stavu `FirstCarTaken` se vybere vozidla z čela fronty, pošle se mu signál, že může pokračovat, následně se čeká na reakci vozidla, zda může pokračovat nebo mu něco brání v pokračování křižovatkou. Pokud může pokračovat, odstraní se záznam o vozidla z fronty a sníží se počet vozidel v křižovatce.

```
void delFromSemQueue ()
{
    if (exists(i:range_queue) semIntervalQueue[i]==id) {
        meta int k = 0;
        while (semIntervalQueue[k]!=id){k++;};
        semIntervalQueue[k]=-1;
    }
    if (exists(i:range_queue) semIntervalPriorityQueue[i]==id) {
        meta int j = 0;
        while (semIntervalPriorityQueue[j]!=id){j++;};
        semIntervalPriorityQueue[j]=-1;
    }
    sortSemQueue();
}
```

Obrázek 3.18: Funkce delFromSemQueue()

Funkce `delFromSemQueue()` slouží pro odstranění semaforu z fronty semaforů. Prvně se zkontroluje, zda semafor je v `semIntervalQueue`, pokud je nalezen, získá k jeho pozici index a tento index nahradí -1, to symbolizuje prázdné místo. To stejné se provede s prioritní frontou semaforů. Poté se spustí funkce `sortSemQueue()`.

```
void sortSemQueue ()
{
    int[-1,M-1] semIntervalSortingQueue[M];
    int sortIndex = 0;
    for (i : int[0,M-1]) {semIntervalSortingQueue[i] = -1;}
    for (i : int[0,M-1])
    {
        if (semIntervalQueue[i]>=0)
        {
            semIntervalSortingQueue[sortIndex]=semIntervalQueue[i];
            sortIndex++;
        }
    }
    for (i : int[0,M-1]) {semIntervalQueue[i]=semIntervalSortingQueue[i];}

    sortIndex = 0;
    for (i : int[0,M-1]) {semIntervalSortingQueue[i] = -1;}
    for (i : int[0,M-1])
    {
        if (semIntervalPriorityQueue[i]>=0)
        {
            semIntervalSortingQueue[sortIndex]=semIntervalPriorityQueue[i];
            sortIndex++;
        }
    }
    for (i : int[0,M-1]) {semIntervalPriorityQueue[i]=semIntervalSortingQueue[i];}
}
```

Obrázek 3.19: Funkce sortSemQueue()

Funkce `sortSemQueue()` obstarává seřazení a zarovnání prvků ve frontách semaforů. Tato funkce je užitečná, protože bez ní by vznikala nekonzistence. Bez ní by mohl nastat stav, kdy nově příchozí semafor přeskočí semafor, který už ve frontě byl. V intervalu se většinou spouští více semaforů najednou a není zaručeno, že se vůbec nachází v nějaké semaforové frontě nebo nejsou přímo za sebou. Tato funkce využívá pomocné pole o stejné velikosti jako jsou semaforové fronty. Pro zarovnání se tedy prvky z fronty postupně kopírují do tohoto pomocného pole, následně se toto pole zkopíruje do fronty. Toto stejné se provede obě semaforové fronty.

```
int[0,N-1] headQueue ()
{
  return queue[0];
}
```

Obrázek 3.20: Funkce headQueue()

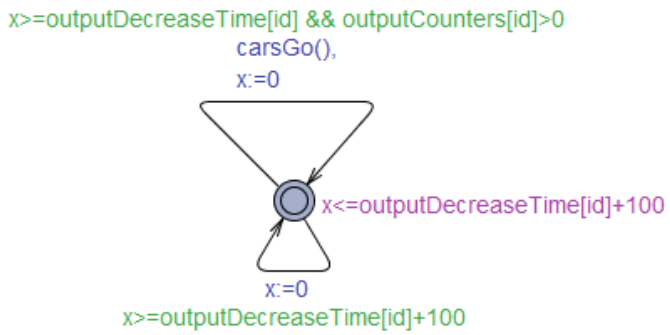
Funkce `headQueue()` vrací první vozidlo z fronty čekající na semafor. Tato funkce se využívá při vpouštění vozidel do křižovatky.

```
void deQueue ()
{
  for (i : int[1,queue_size-1])
    queue[i-1] = queue[i];
  queue[queue_size-1] = -1;
    queueCounters[id]--;
}
```

Obrázek 3.21: Funkce deQueue()

Funkce `deQueue()` slouží pro odstranění vozidla z fronty. Toto se provede poté až vozidlo dá signál, že může pokračovat, zavolá se tedy tato funkce a vozidlo vjede do křižovatky. Cyklem s předem daným počtem opakování se projde fronta od druhého prvku, postupně se všechny prvky posunou o jednu pozici dopředu. Na konec se nastaví -1, symbolizující prázdný prvek, a sníží se počítadlo vozidel.

3.2.4 Automat ovládající výjezdové silnice



Obrázek 3.22: Automat ovládající výjezdové silnice

Tento automat se generuje pro každou výjezdovou silnici. Jediným úkolem, který má, je každý definovaný čas snížit počet vozidel na dané silnici o definovaný počet vozidel.

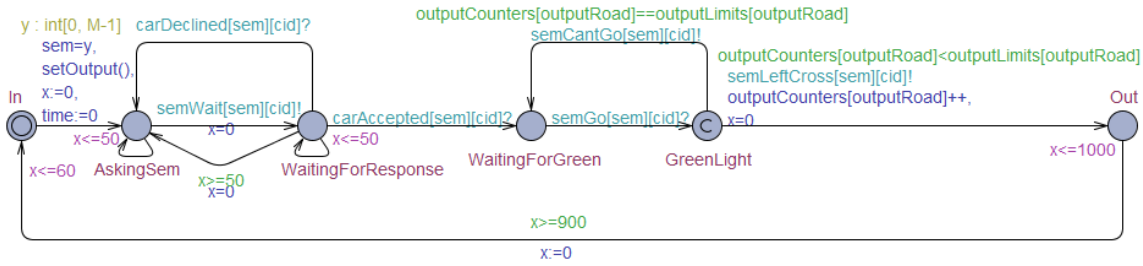
```
void carsGo()
{
    if(outputCounters[id] > outputDecreaseCount[id])
    {
        outputCounters[id] = outputCounters[id] - outputDecreaseCount[id];
    }
    else
    {
        outputCounters[id] = 0;
    }
}
```

Obrázek 3.23: Funkce carsGo()

Jedná se o jednoduchou funkci, jenž vezme aktuální počet vozidel na silnici a odečte od této hodnoty definovanou hodnotu z globálních deklarací. Pokud by snížení překročilo do záporných hodnot, nastaví se počet na nula.

3.2.5 Automat ovládající statické vozidlo

Tento automat za úkol simulovat chování vozidla, pokud bude systém nastaven na používání staticky generovaných vozidel, to znamená, že vozidla se generují před spuštěním simulace, a to kdy se připojí do křižovatky je nastaveno přímo v automatu řídící vozidlo.



Obrázek 3.24: Automat statického vozidla

Automat začíná ve stavu In, ve kterém se zdrží nějaký čas, poté pokračuje do stavu `AskingSem`, při přechodu do tohoto stavu si vybere semafor v křižovatce, ke kterému míří. Když má zvolen semafor, tak si nastaví výjezdovou cestu, kterou chce pokračovat, výběr probíhá ze speciálního pole, ve kterém je určeno, ze kterého semaforu může pokračovat, do kterých výjezdových silnic. Po tomto nastavení se automat nachází ve stavu `AskingSem`. Po určitém čase, než dorazí na hranici semaforu, pokračuje do stavu `WaitingForResponse`, při přechodu se dotáže vybraného semaforu, zda má místo, zda může pokračovat do fronty. Pokud je fronta plná, automat řídící vozidlo se vrací do stavu `AskingSem`. Pokud fronta není plná, automat řídící vozidlo pokračuje do stavu `WaitingForGreen`, automat řídicí frontu u semaforu přidá vozidlo do fronty a zvýší počet aut ve frontě.

```
void setOutput()
{
    int i;
    int rnd;
    countOutputRoads = 0;
    for(i = 0; i < 3; i++)
    {
        if(outputConnections[sem][i]!=-1)
        {
            countOutputRoads++;
        }
    }
    if(countOutputRoads>1)
    {
        rnd = fint(random(countOutputRoads));
        outputRoad = outputConnections[sem][rnd];
    }
    else
    {
        outputRoad = outputConnections[sem][0];
    }
}
```

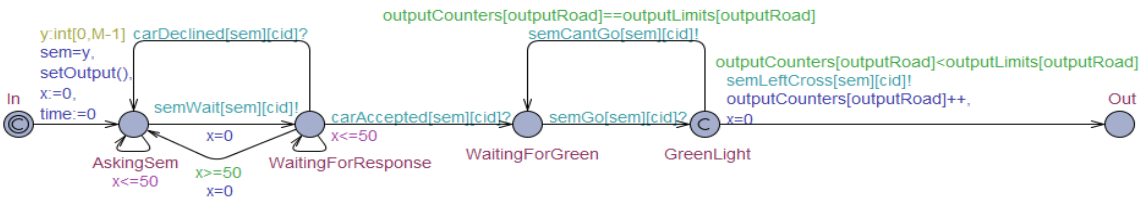
Obrázek 3.25: Funkce setOutput()

Funkce `setOutput()` má za úkol nastavit vozidlu výjezdovou silnici. Prvně se ve funkci, pomocí cyklu s předem daným počtem opakování a ověřování, zda existuje nějaká výjezdová silnice, nastaví počet výjezdových silnic. Pokud není možné pokračovat více než jednou výjezdovou silnicí nastaví se tato silnice jako výjezdová. Pokud je výjezdových silnic víc, nastaví se silnice pomocí funkce `random()`, která vygeneruje náhodné číslo z daného intervalu.

Nyní automat řídící vozidlo čeká na signál, že může pokračovat do křižovatky, automat se přesune do stavu `GreenLight`, kde auto zkontroluje, zda může pokračovat do výjezdové silnice, pokud zjistí, že je výjezdová silnice plná, tak vozidlo nemůže pokračovat a automat se vrací do stavu `WaitingForGreen` a opět čeká na signál, že může pokračovat do křižovatky. Pokud je ve výjezdové silnici místo na vozidlo, tak automat řídící vozidlo dá signál, že pokračuje do výjezdové silnice, připočte se nové vozidlo do počítadla vozidel v určité výjezdové silnici.

Jelikož se jedná o staticky tvořené vozidlo, tak automat přejde do stavu `Out`, ale po určité době se automat řídící vozidlo vrací do stavu In, aby se zabezpečil provoz v křižovatce.

3.2.6 Automat ovládající dynamické vozidlo

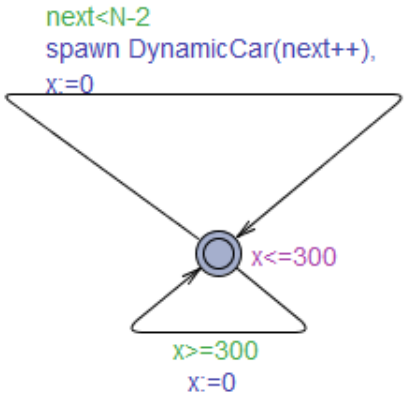


Obrázek 3.26: Automat dynamického vozidla

Tento automat simuluje vozidlo projíždějící křižovatkou v případě, kdy je systém nastaven do dynamické režimu. Vozidla se v automatu nevrací, odjíždí z křižovatky a už se nevrací. Automat je stejný jako u statického režimu, jen bez cyklující části.

3.2.7 Automat ovládající provoz

Má za úkol vytvářet dynamické instance vozidel každých ... časových jednotek, kde je tento čas podle aktuálního scénáře. Vytváří se určitý počet vozidel, který je nastaven v globální proměnné N. Spodní hrana je zde jen kvůli UPPAAL, protože pro simulaci musí být zajištěno, že se nějaká hrana provede.



Obrázek 3.27: Automat provozu

Kapitola 4

Testování a zhodnocení realizace

4.1 Modelované křižovatky

Křižovatky jsou vybírány, aby se příliš neopakovalo schéma křižovatky a jsou vybírány takové křižovatky, které znám nebo jsem u nich v minulosti byl.

4.1.1 První modelovaná křižovatka

Obrázek 4.1: První křižovatka

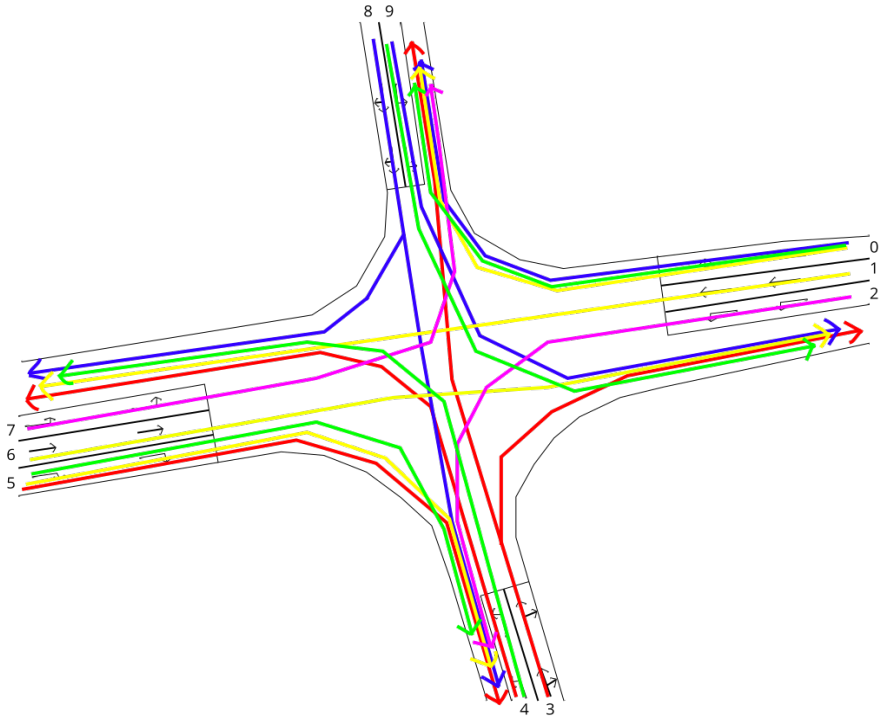
Jedná se o křižovatku ve městě Jihlava, jedná se o křížení ulic Hradební a ulice Znojemská.

Ze severu vede ulice z náměstí a ke křižovatce se příjezdová silnice rozděluje do dvou proudů, z toho, z pohledu příjezdového vozidla, je levý pruh pro odbočení doleva a pravý pruh je pro pokračování rovně nebo odbočení vpravo. To samé je i při příjezdu z jižní strany.

Z východu a ze západu se vždy proud rozděluje do tří proudů, pro odbočení vlevo, pro pokračování rovně a pro odbočení vpravo.

Intervaly

Intervaly jsou nastavené tak, aby neporušovaly pravidla silničního provozu a nijak jsem je neporovnával s realitou.



Obrázek 4.2: Intervaly první křižovatky

Je nastaveno pět intervalů. Jsou barevně označeny na předchozím obrázku. Podle těchto barev a indexů jednotlivých semaforů lze vyhledávat i v tabulce.

Intervaly označené indexy:

1.interval	žlutá	0	1	5	6
2.interval	modrá	0	8	9	-
3.interval	červená	3	4	5	-
4.interval	růžová	2	7	-	-
5.interval	zelená	0	4	5	9

Tabulka 4.1: Intervaly první křižovatky v tabulce

Intervaly se u klasických semaforů spouští přesně podle pořadí v tabulce.

V implementaci této křižovatky je nastaven počet semaforů a výjezdových silnic podle reality, tedy celkově je v křižovatce 10 semaforů, to znamená 10 jízdních pruhů, ze kterých si mohou řidiči vybrat. Počet výjezdových silnic je opět nastaven podle reality a jsou tedy čtyři.

Nastavení výjezdových silnic:

- Východní silnice - provoz nastaven na 5 vozidel za 200 časových jednotek s limitem 10 vozidel

- Jižní silnice - provoz nastaven na 3 vozidla za 300 časových jednotek s limitem 5 vozidel
- Západní silnice - provoz nastaven na 6 vozidel za 200 časových jednotek s limitem 10 vozidel
- Severní silnice - provoz nastaven na 2 vozidla za 400 časových jednotek s limitem 7 vozidel

Propojení semaforů s výjezdovými silnicemi se řídí obrázkem (obr. 4.2) a nepočítá se, že by vozidla projela z semaforu na jinou výjezdovou silnici než je definováno.

4.1.2 Druhá modelovaná křižovatka

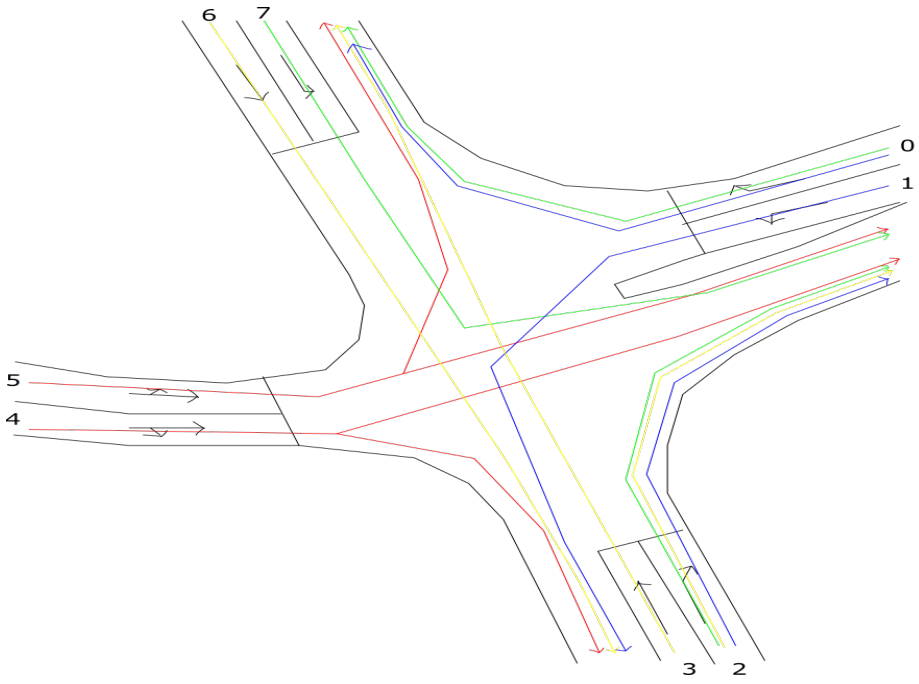
Obrázek 4.3: Druhá křižovatka

Druhou modelovanou křižovatkou je křižovatka v Brně. Setkávají se zde ulice Kollárova, Božetěchova, Křižíkova a Mojmírovo náměstí. Kollárova ulice, na obrázku levá ulice, je jednosměrná a je rozdělena do dvou proudů, oba proudy mohou pokračovat rovně. Božetěchova ulice se také rozděluje do dvou proudů, přičemž pravý proud může pokračovat pouze rovně, levý proud pouze doleva. Křižíkova ulice se rozděluje do dvou proudů, vozidla mohou pokračovat vlevo nebo vpravo. Z ulice od Mojmírova náměstí jsou 2 proudy, levý pruh může pokračovat rovně v křižovatce a pravý pruh může pokračovat pouze vpravo.

Intervaly

Intervaly jsou nastavené tak, aby neporušovaly pravidla silničního provozu a nijak jsem je neporovnával s realitou.

Barvy intervalů na obrázku odpovídají barevně intervalům v tabulce 4.2. Intervaly se u klasických semaforů spouští přesně podle pořadí z tabulky.



Obrázek 4.4: Intervaly druhé křižovatky

1.interval	žlutá	2	3	6
2.interval	modrá	0	1	2
3.interval	červená	4	5	-
4.interval	zelená	0	2	7

Tabulka 4.2: Intervaly druhé křižovatky v tabulce

V implementaci této křižovatky je nastaven počet semaforů a výjezdových silnic opět podle reality, tedy celkově je v křižovatce 8 semaforů, to znamená 8 jízdních pruhů, ze kterých si mohou řidiči vybrat. Počet výjezdových silnic je opět nastaven podle reality a jsou tedy čtyři.

Nastavení výjezdových silnic:

- Východní silnice - 6 vozidel za 200 časových jednotek s limitem 8 vozidel
- Jižní silnice - 6 vozidla za 200 časových jednotek s limitem 8 vozidel
- Západní silnice - 2 vozidel za 400 časových jednotek s limitem 6 vozidel
- Severní silnice - 3 vozidla za 500 časových jednotek s limitem 5 vozidel

Propojení semaforů s výjezdovými silnicemi se řídí obrázkem (obr. 4.4) a nepočítá se, že by vozidla projela z semaforu na jinou výjezdovou silnici než je definováno.

4.1.3 Třetí modelovaná křižovatka

Třetí modelovanou křižovatkou je křižovatka opět v Jihlavě. Tentokrát ale jednodušší křižovatka. Setkávají se zde ulice Havlíčkova, Fritzova a Úvoz.

Obrázek 4.5: Třetí křižovatka

Fritzova ulice, na obrázku levá ulice, je to vedlejší ulice a je rozdělena do dvou proudů, s tím že je pouze jeden proud ovlivňován světelnou signalizací, ovlivňovaný proud pouští vozidla ve směru doleva a rovně. Proud pouštějící vozidla doprava je ovlivňován klasickou předností vozidel jedoucí po zleva. Havlíčkova ulice, na obrázku směr zespoda, se také rozděluje do dvou proudů, tentokrát jsou oba proudy ovlivněny světelnou signalizací, pravý proud může pokračovat pouze rovně a doprava, levý proud pouze doleva. Havlíčkova ulice, na obrázku směr seshora, se rozděluje do tří proudů, pouze dva proudy jsou ovlivněny světelnou signalizací, přičemž tyto proudy umožňují pokračovat vlevo nebo rovně. Neovlivněný proud je odbočovací doprava. Z ulice Úvoz je pouze jeden proud, vozidla mohou pokračovat v křižovatce rovně nebo doleva.

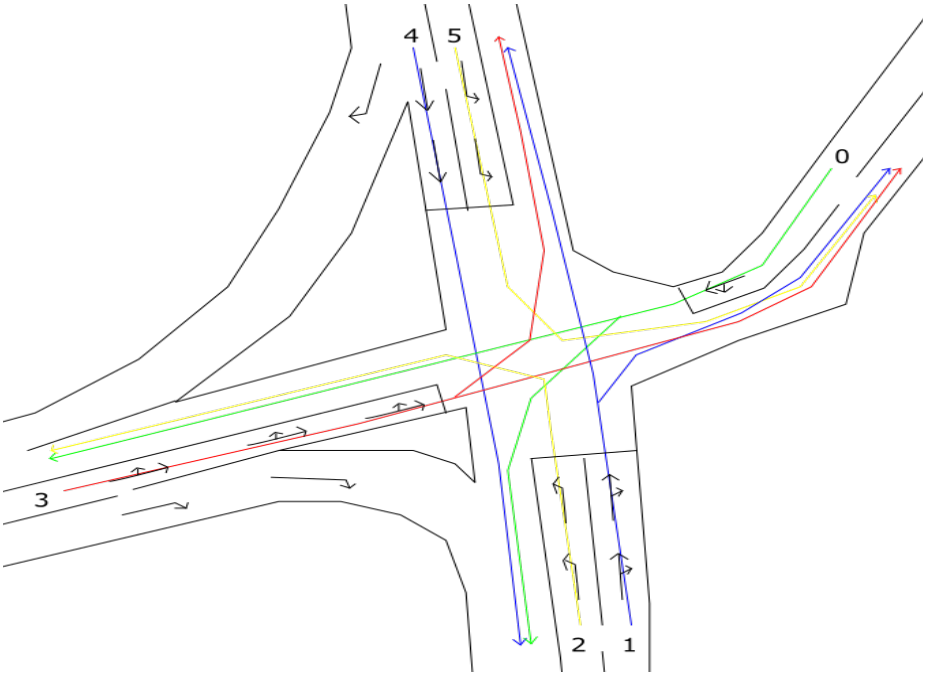
Intervaly

Intervaly jsou opět nastavené tak, aby neporušovaly pravidla silničního provozu a nijak jsem je neporovnával s realitou.

Intervaly, barevně odpovídající obrázku výše, semafory označené indexy v tabulce a spouštějí se přesně v tomto pořadí:

1.interval	žlutá	2	5
2.interval	modrá	1	4
3.interval	červená	3	-
4.interval	zelená	0	-

Tabulka 4.3: Intervaly třetí křižovatky v tabulce



Obrázek 4.6: Intervaly třetí křižovatky

V implementaci této křižovatky je nastaven počet semaforů a výjezdových silnic podle reality, tedy celkově je v křižovatce 6 semaforů, to znamená 6 jízdních pruhů, ze kterých si mohou řidiči vybrat. Počet výjezdových silnic je opět nastaven podle reality a jsou tedy čtyři.

Nastavení výjezdových silnic:

- Východní silnice - provoz nastaven na 6 vozidel za 200 časových jednotek s limitem 5 vozidel
- Jižní silnice - provoz nastaven na 4 vozidla za 400 časových jednotek s limitem 10 vozidel
- Západní silnice - provoz nastaven na 2 vozidel za 400 časových jednotek s limitem 7 vozidel
- Severní silnice - provoz nastaven na 3 vozidla za 500 časových jednotek s limitem 8 vozidel

Propojení semaforů s výjezdovými silnicemi se řídí obrázkem (obr. 4.6) a nepočítá se, že by vozidla projela z semaforu na jinou výjezdovou silnici než je definováno.

4.2 Aplikace scénářů na modelované křižovatky

4.2.1 Způsob sběru a vyhodnocení dat

Data experimentů jsou získávány z dotazu ve verifikátoru UPPAALu, kde je většina dotazů spouštěna vícekrát a konečný výsledek je získán průměrem těchto spuštění. Tyto dotazy jsou spouštěny, jak s adaptivní křižovatkou, tak i s klasickou křižovatkou bez jakéhokoli přizpůsobování intervalů. Tyto hodnoty jsou následně porovnávány v rámci scénářů.

4.2.2 Velká vytíženost křižovatky

Tímto scénářem je sledováno chování křižovatky při extrémním počtu vozidel. Testováním by se mělo zjisti, že se chová stejně jako klasická křižovatka. Pro simulaci velké vytíženosti křižovatky byl tedy zvolen maximální počet vozidel, jenž se mohou dynamicky generovat, bohužel kvůli tomu, jak je UPPAAL technicky navržen, tak se mi podařilo maximálně generovat 100 vozidel a i toto množství se občas nepodaří spustit a UPPAAL navrátí chybovou hlášku "`java.lang.OutOfMemoryError: Java heap space`", chyba upozorňující na nedostatek paměti. U dynamického spouštění se vozidla generují každých 25 až 50 časových jednotek, s tím že je zde možnost nevygenerování vozidla.

U klasických semaforů je nastavení stejné, aby testování scénářů a porovnávání výsledků dávalo smysl.

Intervaly nejsou nijak pozměněny a jsou nastaveny podle tabulek u jednotlivých křižovatek (tab. 4.1, 4.2, 4.3).

Bohužel se nepodařilo otestovat pravděpodobnosti a očekávané hodnoty adaptivní křižovatky v dynamickém režimu, jelikož se v dynamickém režimu velmi špatně odkazuje indexy na dynamicky generovány části systému. Proto jsou zde zmíněny jen výsledky ze staticky generovaného.

První testovanou vlastností je pravděpodobnost, jestli do 200 časových jednotek se objevila nějaká fronta s délkou větší než 8 vozidel. Testování se provádí na 1000 pokusech.

$$Pr[<= 200; 1000](<> exists(c : semid\_t)semQueue(c).count > 8) \tag{4.1}$$

Pro všechny modelované křižovatky vyšla velmi vysoká pravděpodobnost, ve většině testů byla pravděpodobnost větší jak 0.997009 a to, jak pro adaptivní křižovatku, tak i pro klasickou křižovatku. Pravděpodobnostní funkci lze nalézt v příloze (obr. A.1, A.8). Tímto výsledkem se potvrzuje, že při velmi hustém provozu se adaptivní křižovatka chová jako klasická.

Další testovanou vlastností je pravděpodobnost, jestli zvládne první vozidlo projet křižovatkou do 100 časových jednotek. Testování se provádí 500krát.

$$Pr[<= 100; 500](<> Car(0).Out) \tag{4.2}$$

V tomto případě má adaptivní křižovatka asi o 4% větší pravděpodobnost než klasická křižovatka. Očekávání bylo o cca 6% větší pravděpodobnost tohoto jevu, ale i tak je to dobrý výsledek. Rozdělení hustoty pravděpodobnosti lze nalézt v příloze (obr. A.2, A.9).

Další dva testy nejsou příliš dělané na staticky generovaná vozidla, kdy všechny vozidla v krátký časový úsek nahrnou na křižovatky.

$$E[<= 200; 500](max : sum(i : semid\_t)semQueue(i).count) \tag{4.3}$$

Tento test počítá vozidla ve frontách, dělá z tohoto počtu sumu a z této sumy se nalezne maximum, z 500 pokusů se vytvoří průměrný počet. Počítá se v prvních 200 časových jednotkách.

$$E[<= 60; 1000](max : sum(i : carid\_t)Car(i).WaitingForGreen) \tag{4.4}$$

V tomto testu se počítají vozidla čekající na zelenou v prvních 60 časových jednotkách. Testování probíhá tisíckrát.

V těchto dvou testech se výsledky lišili o cca 1-2 vozidla v prospěch adaptivní křižovatky, kdyby fungovalo dynamické generování vozidel v těchto testech jsou očekávány výsledky s

větším rozdílem v prospěch adaptivních. To lze sledovat při simulaci. Tyto dva testy už u ostatních scénářů nebudou zmiňovány z důvodu podávání podobných výsledků.

Výsledky testování s dynamickým generováním vozidel pro velkou vytíženost křižovatky jsou k dispozici v příloze (část A.1). Tyto simulace podávali podobné výsledky po opakovaném spuštění a jsou zde uvedené příklady výsledků.

Z počtu vozidel (graf A.3, A.10) lze pozorovat rychlejší odbavení vozidel adaptivní křižovatkou. Na grafu je vidět, jak vozidla do systému přijíždějí a kolik jich čeká na zelenou. Nejen že počet čekajících vozidel dosahuje v grafu menšího maxima, ale i rychleji dosáhne počet čekajících vozidel nuly.

Stavy front u semaforů (graf A.4, A.11) potvrzují rychlejší odbavení vozidel u adaptivní křižovatky. Na grafu jsou vidět počty vozidel v jednotlivých frontách u semaforů.

Při velké vytíženosti křižovatky se většinu času střídají intervaly pravidelně, jak u klasických semaforů. Graf A.6 ukazuje jak se intervaly střídají, ze začátku, kdy ještě nejsou tolik přeplněné fronty, je možnost nepravidelného vystřídání intervalů.

Zkracování intervalů je možné i u tohoto scénáře, graf A.7 tuto skutečnost potvrzuje.

4.2.3 Nízká vytíženost křižovatky

Bude zde sledováno chování křižovatky při nižším počtu vozidel, kde by se adaptivní vlastnosti křižovatky měly projevit nejvíce. V tomto scénáři se celkově nachází 50 generovaných vozidel a u dynamického spouštění se vozidlo generuje každých 200 až 300 časových jednotek, opět s možností nevygenerování vozidla. Nic jiného nastavení křižovatky se nezměnilo.

Prvním testem je opět testování pravděpodobnosti, zda existuje v prvních 200 časových jednotkách případ, kdy jsou ve frontě 9 a více vozidel. (test 4.1)

V tomto testu podávala adaptivní křižovatka o cca 8% nižší výsledky než klasická křižovatka. Očekávání byla o cca 7% větší rozdíl oproti klasické křižovatce, ale i tento výsledek potvrzuje funkčnost adaptivních funkcí. Pravděpodobnostní funkci lze nalézt v příloze (obr. A.15, A.22).

Dalším testem je pravděpodobnost, že první vozidlo projede křižovatkou v prvních 100 časových jednotkách. (test 4.2)

Zde se chování oproti velké vytíženosti příliš nezměnilo. Opět šance o cca 4% vetší u adaptivní křižovatky než u klasické. Očekávání, že se pravděpodobnost oproti velké vytíženosti křižovatky moc nezmění, se potvrdilo. Rozdělení hustoty pravděpodobnosti lze nalézt v příloze (obr. A.16, A.23).

Výsledky testování s dynamickým generováním vozidel pro nízkou vytíženost křižovatky jsou k dispozici v příloze (část A.2). Uvedené jsou pouze příklady výsledků.

Na grafech počtu vozidel (graf A.17, A.24) lze vidět, že se u adaptivní křižovatky tolik nestává, aby čekalo více vozidel na zelenou. Tuto skutečnost potvrzuje i graf sledující stavy front u semaforů (graf A.18, A.25), na kterých lze pozorovat, že fronty příliš neplní a vozidla jsou rychle odbavovány.

Pokud se jedná o nízko vytíženou křižovatku je nepravidelné střídání intervalů a zkracování časů intervalů skoro pořád aktivní. To potvrzují grafy sledující intervaly v adaptivní křižovatce (graf A.20, A.21). Grafy sledující intervaly pro klasickou křižovatku nejsou opakovaně zmíněny z důvodu stejných výsledků jako u vysoké vytíženosti (graf A.13, A.14).

4.2.4 Chování křižovatky v noci

V tomto scénáři bude sledováno chování křižovatky, když bude extrémně málo vozidel a vozidla budou generována po velmi dlouhé době. Je nastaveno, aby se vytvořilo celkově 10

vozidel a generovaly se po 900 až 1000 časových jednotek, znovu s možností nevygenerování vozidla. Ve většině křižovatek je spuštěn tzv. noční režim, tedy blikající oranžové světlo a přednost je není řízena světelnou signalizací, ale to by neotestovalo chování křižovatky, je tedy počítáno s variantou plně fungujících semaforů.

Při tomto scénáři a testování na frontu větší než 8 vozidel v prvních 200 časových jednotkách se výsledek u adaptivní a klasické křižovatky neliší a pravděpodobnost tohoto jevu je cca 0.3%. (test 4.1)

Při druhém testu se opět opakovaly výsledky z velké či nízké vytíženosti, tedy cca o 4% v prospěch adaptivní. (test 4.2) Rozdělení hustoty pravděpodobnosti lze nalézt v příloze (obr. A.27, A.31).

Výsledky testování s dynamickým generováním vozidel pro chování křižovatky v noci jsou k dispozici v příloze (část A.3). Uvedené jsou pouze příklady výsledků.

Grafy A.28, A.32 potvrzují, že adaptivní křižovatka tu má výhodu nepravidelného střídání intervalů, jelikož jakmile vozidlo dorazí ke křižovatce, je interval upraven tak, aby bylo vozidlo co nejdříve odbaveno. U klasických semaforů musí vozidlo počkat, než bude jeho interval na řadě. Tuto skutečnost potvrzují i grafy A.29 a A.33.

4.2.5 Jedna z výjezdových silnic je přehlcena

Jedná se o modifikaci scénáře s nízkou vytížeností křižovatky, ale s úpravou jedné výjezdové silnice, a to takovým způsobem, že se bude vyklízet extrémně pomalu, to může simulovat například nehodu na této silnici a tím vznikla kolona, která popojíždí velmi pomalu.

Pro první test se výsledky pro tento scénář liší cca o 2 až 3% v prospěch adaptivních funkcí v křižovatce. (test 4.1) Pravděpodobnostní funkci lze nalézt v příloze (obr. A.35, A.40).

Druhý test navrátil výsledky téměř podobné s maximem rozdílu jednoho procenta v prospěch adaptivní křižovatky. (test 4.2) Rozdělení hustoty pravděpodobnosti lze nalézt v příloze (obr. A.36, A.41).

Výsledky testování s dynamickým generováním vozidel pro chování křižovatky, pokud je jedna výjezdová silnice přehlcena vozidly, jsou k dispozici v příloze (část A.4). Uvedené jsou pouze příklady výsledků.

Na grafech A.39 a A.44 lze pozorovat, že výjezdová silnice č. 2 se plní výrazně více než ostatní. To má vliv i na vozidla, jenž chtějí pokračovat tímto směrem, ty se zdrží, to lze vidět na grafu A.37.

4.2.6 Shrnutí výsledků

Model adaptivní křižovatky představuje vylepšení klasické křižovatky. Rozdíl ve výsledcích mezi adaptivní a klasickou křižovatkou je sice menší než očekávaný. Potvrdilo se, že adaptivní křižovatka je více efektivní při nízké až střední vytíženosti křižovatky. Při vysoké vytíženosti se většinu času křižovatka chová jako klasická. Kdyby se podařilo generovat vozidla dynamicky a k tomu spouštět testy na pravděpodobnost a testy očekávaných hodnot předpokládá se, že by se rozdíl zvětšil. Vypovídají o tom i simulační testy a jejich výsledky (část A).

Kapitola 5

Závěr

Cílem této práce bylo vytvoření výpočetního modelu adaptivní křižovatky. Pro realizaci tohoto modelu bylo zapotřebí analyzovat aktuální principy související s řízením dopravy světelnou signalizací, například možnosti detekce vozidel v křižovatce, seznámit se s doporučeným modelovacím softwarem a vybrat si, ve kterém softwaru bude model realizován. Zjistili jsme, jak semafory fungují, jaké jsou problémy klasických semaforů. Dozvěděli jsme mnoho cenných informací o modelování a simulaci.

Model je vytvořen v softwaru UPPAAL a po této zkušenosti s tímto softwarem, bych zkusil spíše software SUMO, jenž slibuje prostředí přímo dělané pro řízení dopravy a stabilitu s vícero vozidly. Až na tyto problémy se stabilitou jsem byl se s UPPAALem spokojen a pro modely s menším počtem automatů v systému určitě velmi použitelný software.

Výpočetní model adaptivní křižovatky byl porovnán s modelem klasické křižovatky ve 4 scénářích, vysoká a nízká vytíženost křižovatky, chování křižovatky v noci a chování křižovatky, když bude jedna z výjezdových silnic špatně uvolňována. Výsledky generoval přímo software UPPAAL, zajímavé výsledky v grafem jsou umístěné v příloze této práce. V testech se model adaptivní křižovatky ukázal jako vylepšení klasické křižovatky a mohl by být použit pro orientační otestování nějaké křižovatky, aby se ověřilo, zda vylepšení křižovatky adaptivními funkcemi bude prospěšné.

Do budoucna bych si dokázal představit vytvoření pokročilejšího modelu, jenž by například dokázal využít informace o druhu vozidla, jelikož některé detektory mají tuto schopnost. Zajímavé by také bylo, zkusit model převést do simulačního software SUMO a vyzkoušet jej s větším počtem vozidel v systému.

Literatura

[1] Alexandre David, A. L. M. M. D. B. P. *Uppaal SMC Tutorial* [online]. Department of Computer Science, Aalborg University, Denmark: [b.n.], 2018 [cit. 2020-04-28]. Dostupné z: https://www.it.uu.se/research/group/darts/papers/texts/uppaal-smc-tutorial.pdf.

[2] Behrmann, G. *UPPAAL CORA* [online]. cs.aau.dk, 2014 [cit. 2020-04-28]. Dostupné z: https://people.cs.aau.dk/~adavid/cora/.

[3] David, A. *Statistical Model-Checker* [online]. cs.aau.dk, 2012 [cit. 2020-04-28]. Dostupné z: http://people.cs.aau.dk/~adavid/smc/.

[4] David, A. *UPPAAL TIGA* [online]. cs.aau.dk, 2014 [cit. 2020-04-28]. Dostupné z: http://people.cs.aau.dk/~adavid/tiga/.

[5] (DLR), G. A. C. et al. *Documentation - SUMO Documentation* [online]. 2018 [cit. 2020-04-28]. Dostupné z: https://sumo.dlr.de/docs/SUMO_User_Documentation.html.

[6] Evžen Kindler, I. K. a. *SIMULACE A MODELOVÁNÍ, Studijní skripta* [online]. Ostrava: zcu.cz, 2001 [cit. 2020-04-28]. Dostupné z: https://vendulka.zcu.cz/Download/Free/SkriptaKindlerMS.pdf.

[7] *Traffic video detection and monitoring* [online]. flirmedia.com [cit. 2020-04-28]. Dostupné z: http://www.flirmedia.com/MMC/CVS/Traffic/IT_0002_EN.pdf.

[8] Gerd Behrmann, A. D. a Larsen, K. G. *A Tutorial on Uppaal 4.0* [online]. Department of Computer Science, Aalborg University, Denmark: [b.n.], 2006 [cit. 2020-04-28]. Dostupné z: https://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf.

[9] Mikučionis, M. *UPPAAL TRON* [online]. cs.aau.dk, 2012 [cit. 2020-04-28]. Dostupné z: https://people.cs.aau.dk/~marius/tron/.

[10] Mikučionis, M. *Uppaal Stratego* [online]. cs.aau.dk, 2015 [cit. 2020-04-28]. Dostupné z: https://people.cs.aau.dk/~marius/stratego/.

[11] Mikučionis, M. *UPPAAL* [online]. 2019 [cit. 2020-04-28]. Dostupné z: http://www.uppaal.org/.

[12] *ČSN 50556 Systémy silniční dopravní signalizace*. 2011 [cit. 2020-04-28].

[13] Peringer, P. *Popis simulační knihovny SIMLIB* [online]. 1997 [cit. 2020-04-28]. Dostupné z: https://www.fit.vutbr.cz/~peringer/SIMLIB/doc/html-cz/.

[14] Peringer, P. *Modelování a simulace, Studijní skripta* [online]. Brno: vutbr.cz, 2012 [cit. 2020-04-28]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIMS-IT%2Ftexts%2Fopora-ims.pdf.

[15] Peringer, P. *SIMLIB Home Page* [online]. 2018 [cit. 2020-04-28]. Dostupné z: https://www.fit.vutbr.cz/~peringer/SIMLIB/.

[16] Plzně | Dominikánská 4 | 306 31 Plzeň 1, S. informačních technologií města. *Vozidlové detektory* [online]. 2010 [cit. 2020-04-28]. Dostupné z: http://www.svsmp.cz/svetelna-signalizace/vozidlove-detektory-typy-rozdeleni-funkce.aspx.

[17] *TP 81 NAVRHOVÁNÍ SVĚTELNÝCH SIGNALIZAČNÍCH ZAŘÍZENÍ PRO ŘÍZENÍ PROVOZU NA POZEMNÍCH KOMUNIKACÍCH* [online]. 2015 [cit. 2020-04-28]. Dostupné z: http://www.pjpk.cz/data/USR_001_2_8_TP/TP_81.pdf.

[18] Přibyl, O. *Detektory zasahující do vozovky, úvod do detekce* [online]. Praha: cvut.cz [cit. 2020-04-28]. Dostupné z: https://zolotarev.fd.cvut.cz/mzd/ctrl.php?act=show,file,23843.

[19] Přibyl, O. *Neintrusivní dopravní detektory* [online]. Praha: cvut.cz [cit. 2020-04-28]. Dostupné z: https://zolotarev.fd.cvut.cz/mzd/ctrl.php?act=show,file,23845.

[20] *Smart Traffic Systems 101* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://mobility.here.com/learn/smart-transportation/smart-traffic-systems-101-components-benefits-and-big-data-connection.

[21] Smrčka, A. *Úvod do časovaných systémů a použití při verifikaci* [online]. 2004 [cit. 2020-04-28]. Dostupné z: http://www.fit.vutbr.cz/~meduna/mti/2004_05/smrcka.pdf.

[22] *Video Detectors as Loop Replacement* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://www.aldridgetrafficcontrollers.com.au/products/video-detection.

[23] Vydavatelství Nová média, s. r. o. *Konečný automat — Matematika.cz* [online]. 2013 [cit. 2020-04-28]. Dostupné z: https://matematika.cz/konecny-automat.

[24] Wiering, J. V. J. K. A. *Intelligent Traffic Light Control* [online]. Utrecht University: Information and Computing Sciences, 2004. Dostupné z: http://dspace.library.uu.nl/bitstream/handle/1874/17996/wiering_04_intelligent_traffic.pdf?sequence=2&isAllowed=y.

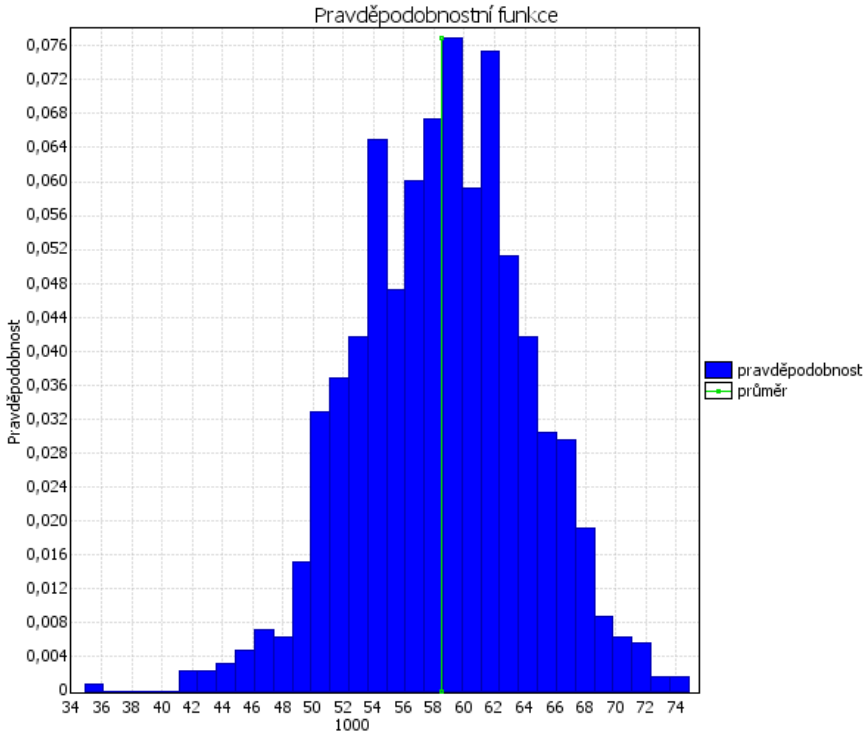
[25] *Zákon č. 361/2000 Sb. o provozu na pozemních komunikacích.* 2000 [cit. 2020-04-28].

Příloha A

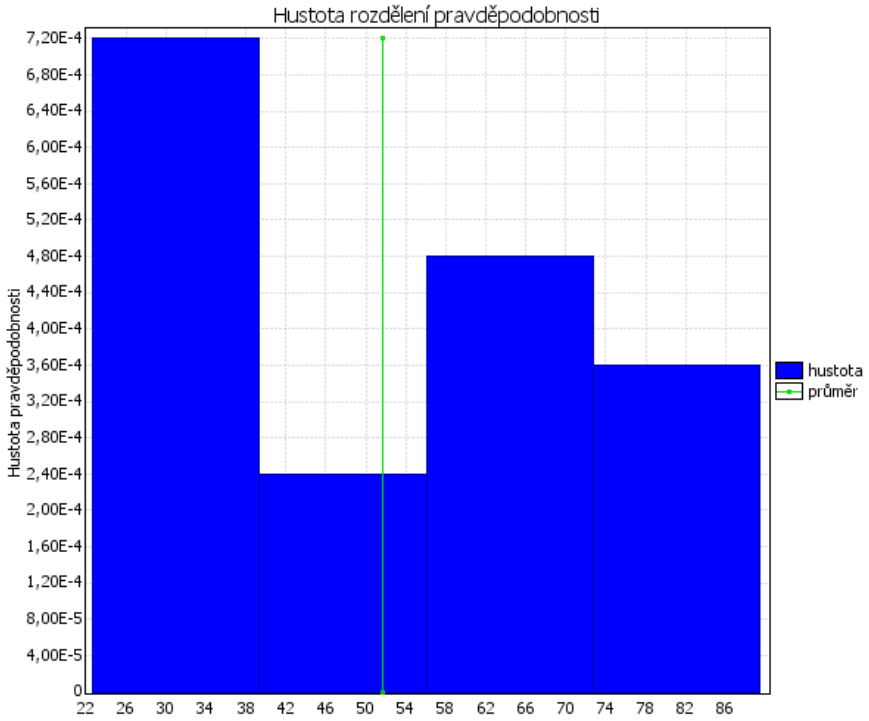
Experimenty

A.1 Velká vytíženost křižovatky

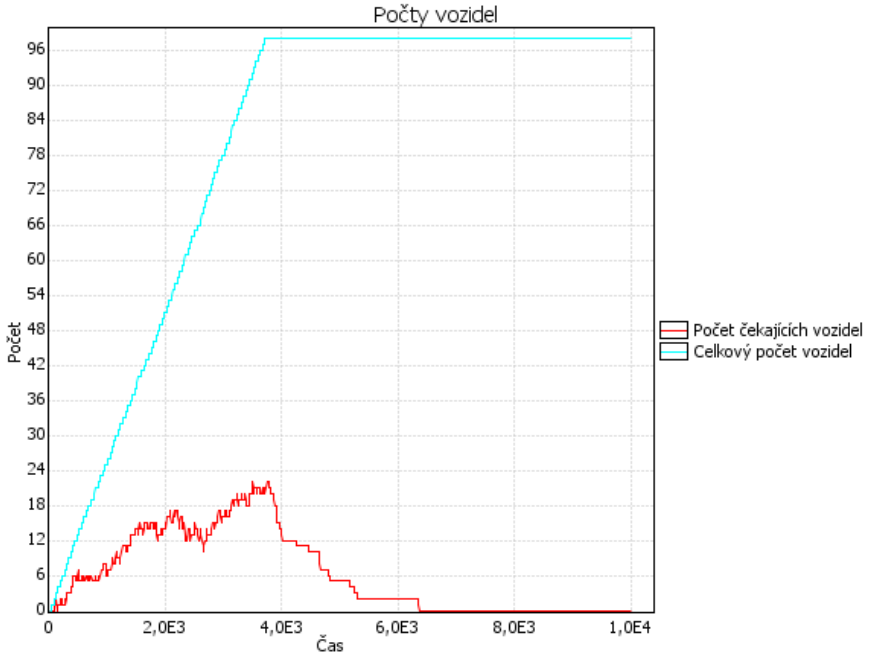
Adaptivní křižovatka



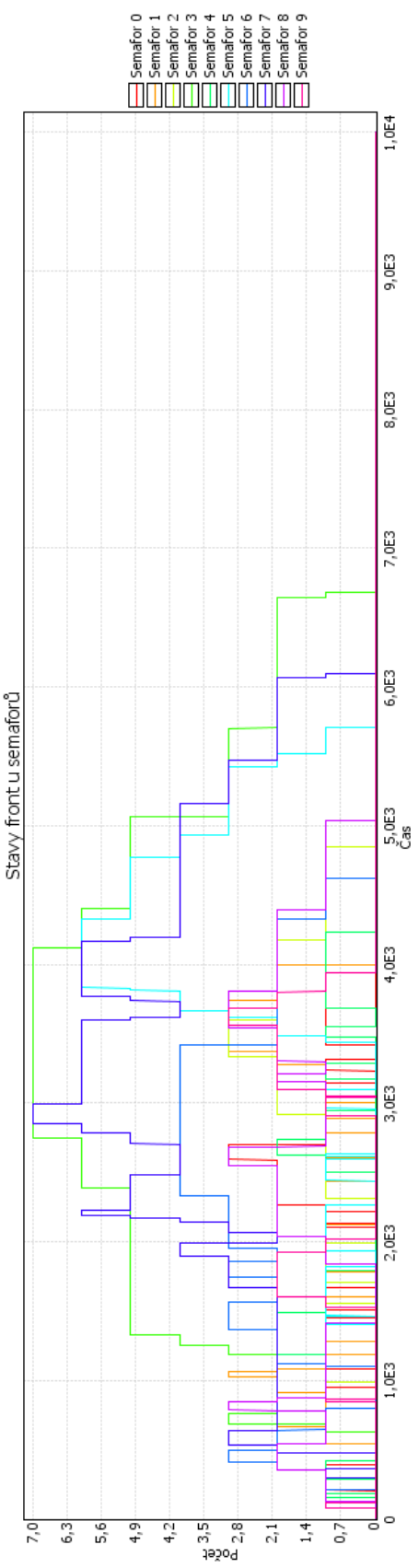
Obrázek A.1: Pravděpodobnostní funkce pro vztah 4.1



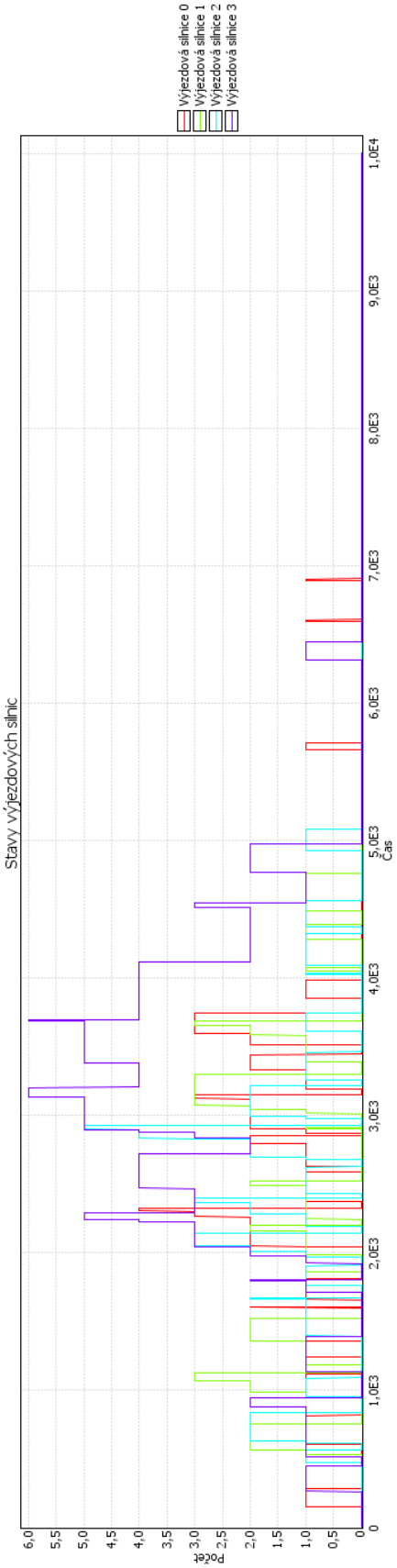
Obrázek A.2: Rozdělení hustoty pravděpodobnosti pro vztah 4.2



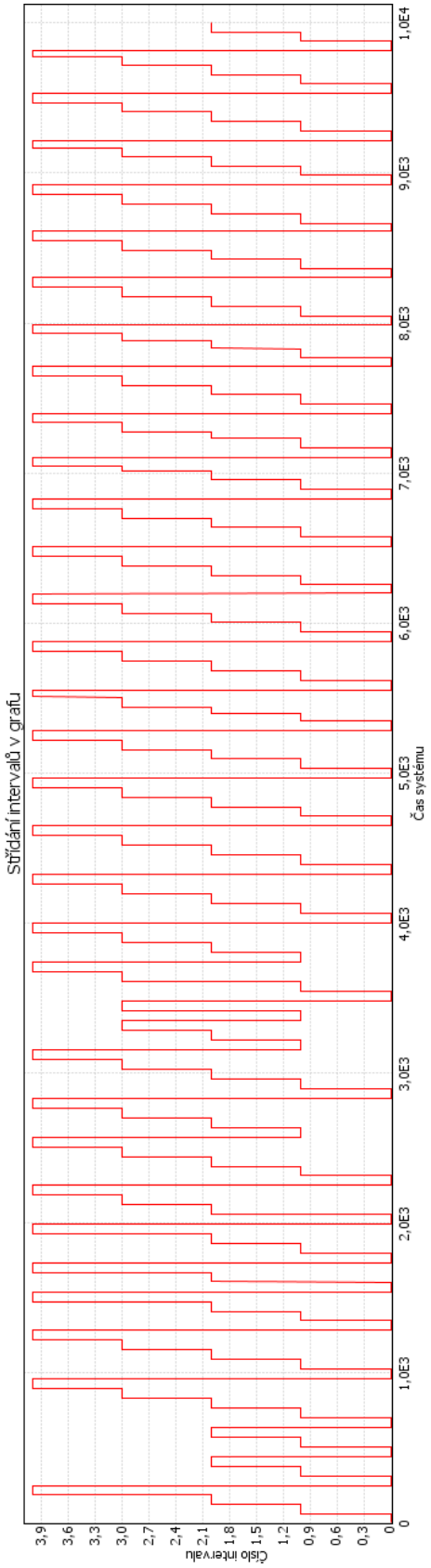
Obrázek A.3: Počet vozidel



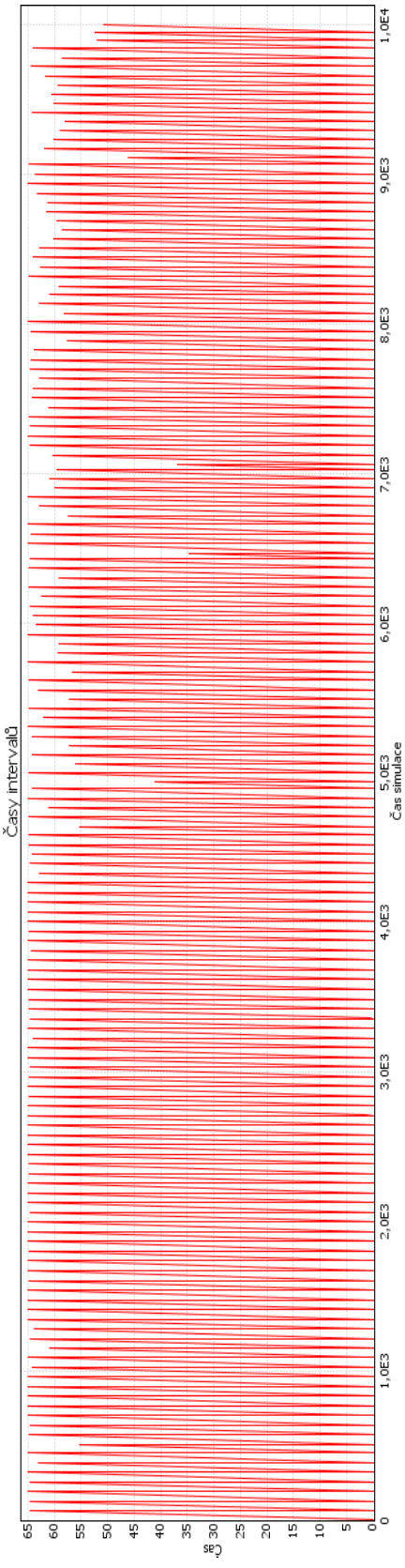
Obrázek A.4: Stavy front u semaforů



Obrázek A.5: Stavy výjezdových silnic

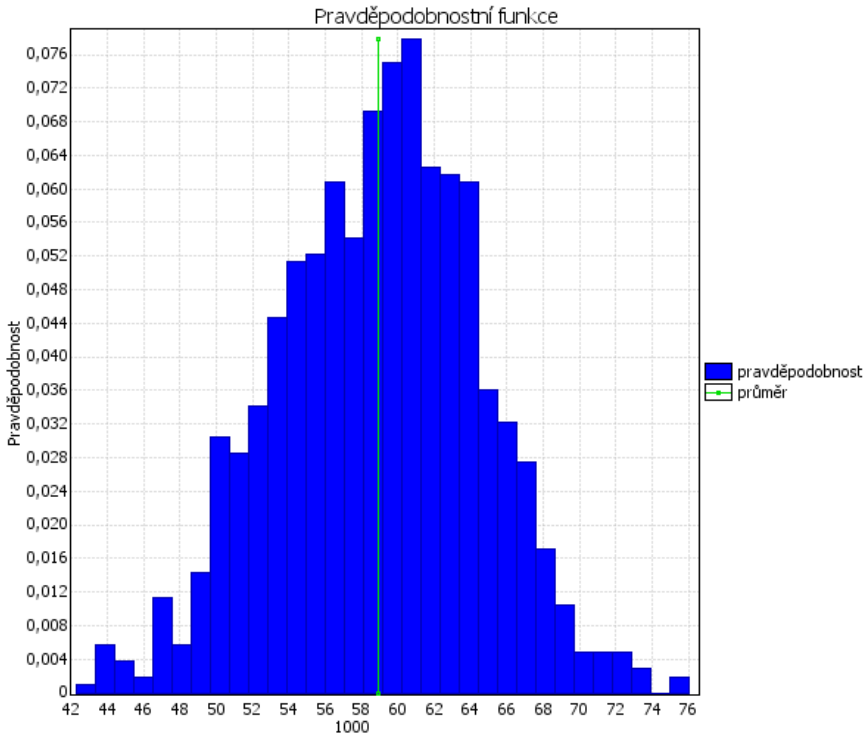


Obrázek A.6: Střídání intervalů v grafu

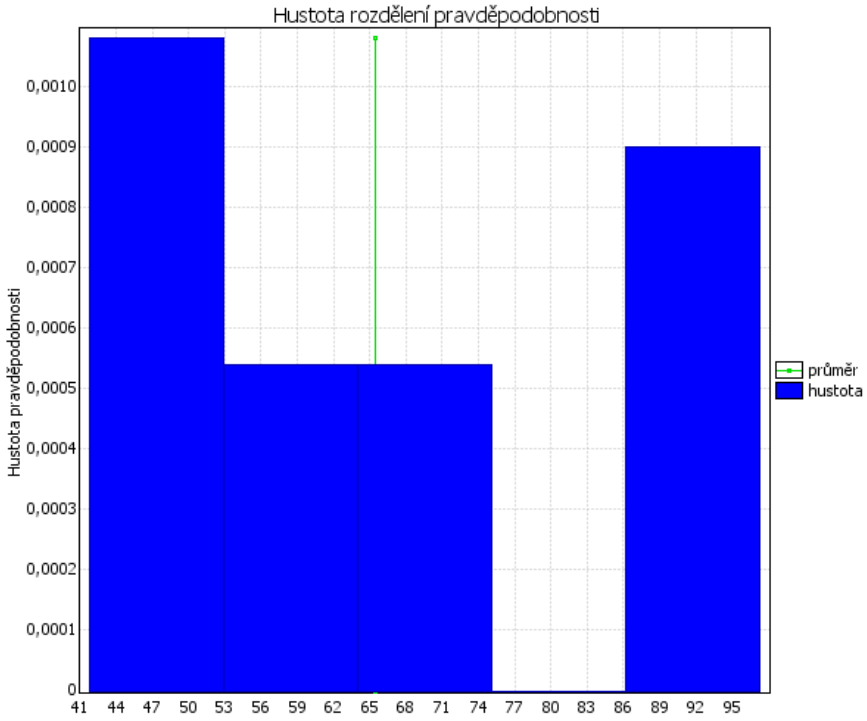


Obrázek A.7: Časy intervalů v grafu

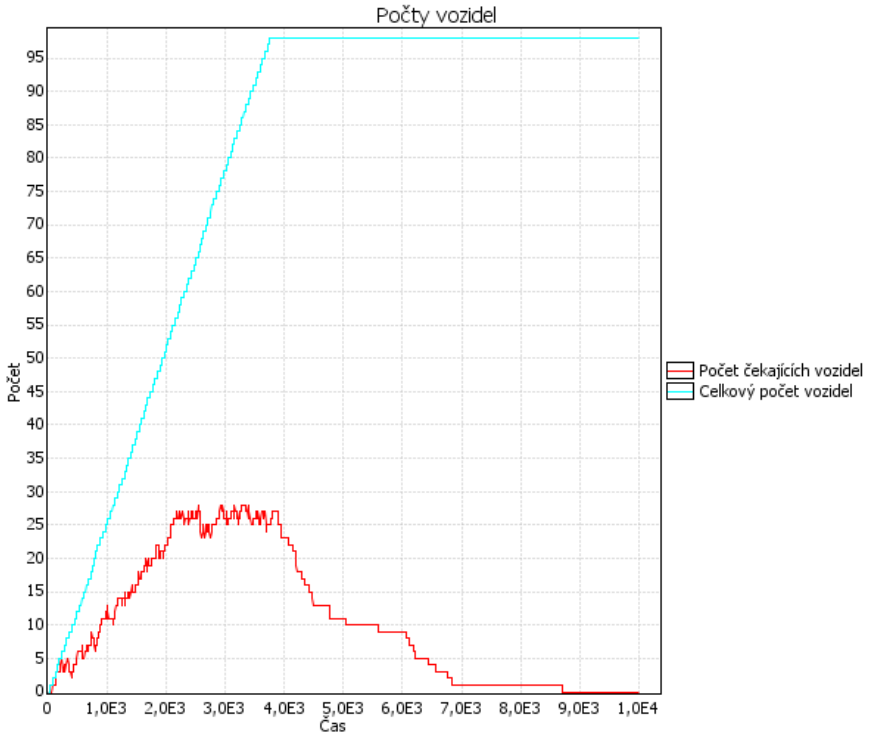
Klasická křižovatka



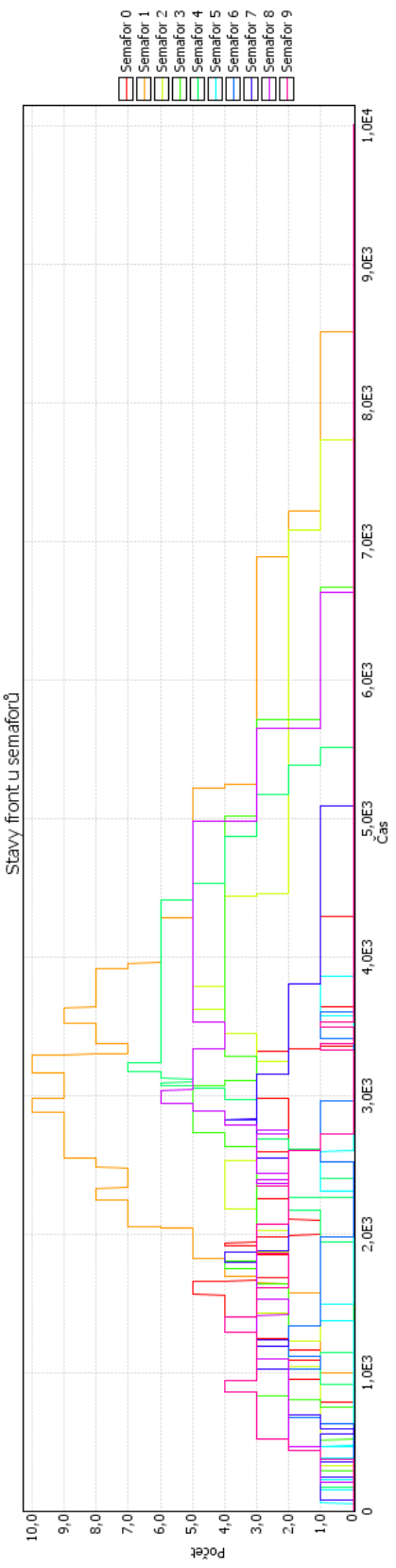
Obrázek A.8: Pravděpodobnostní funkce pro vztah 4.1



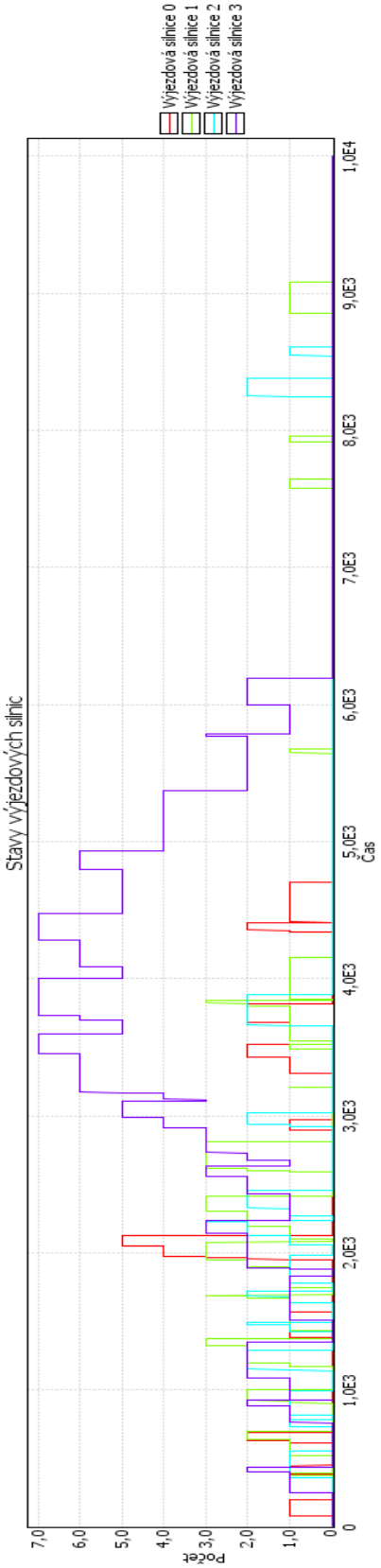
Obrázek A.9: Rozdělení hustoty pravděpodobnosti pro vztah 4.2



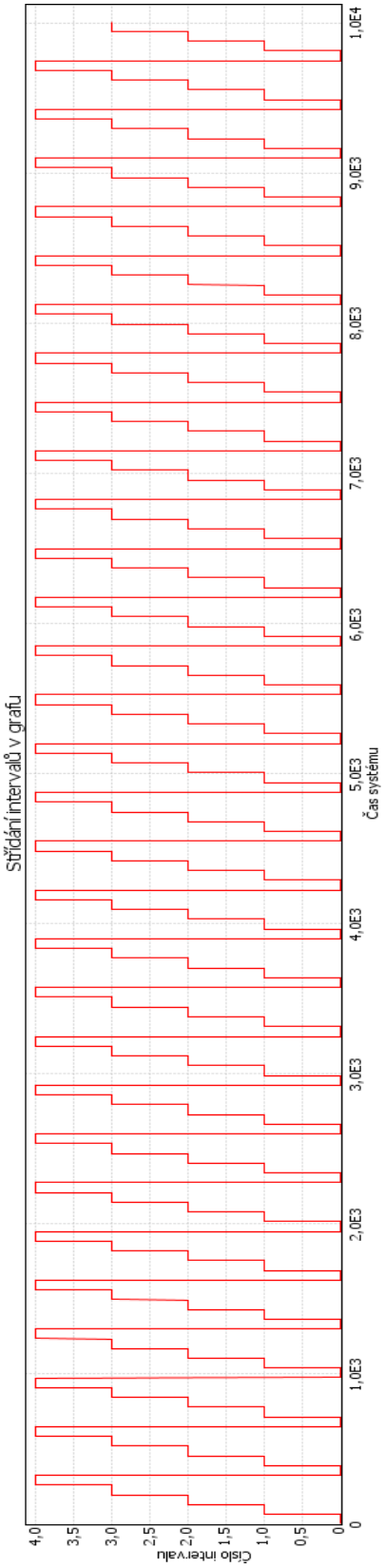
Obrázek A.10: Počet vozidel



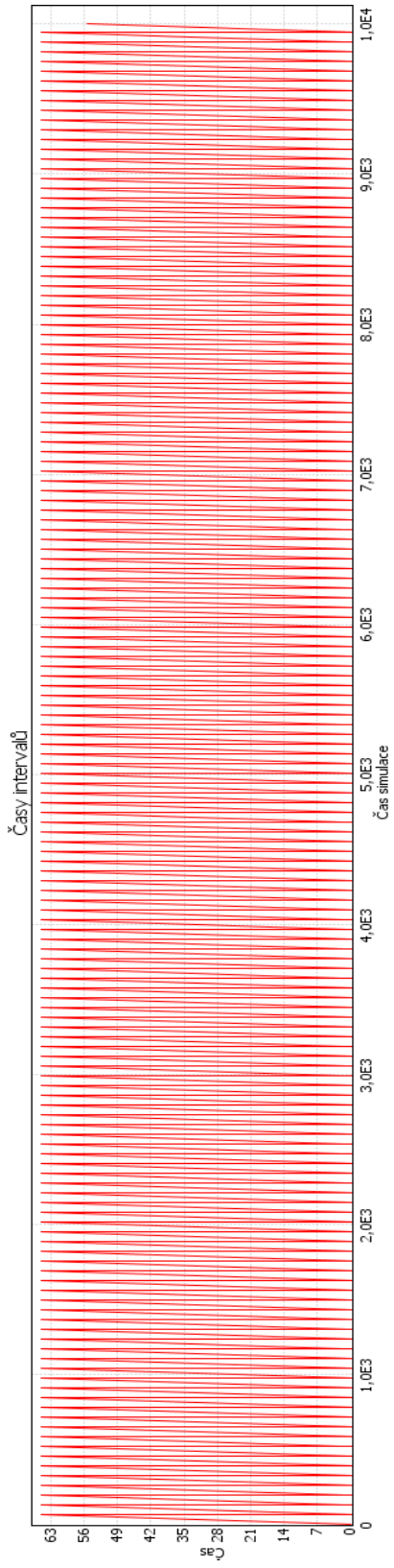
Obrázek A.11: Stavy front u semaforů



Obrázek A.12: Stavy výjezdových silnic



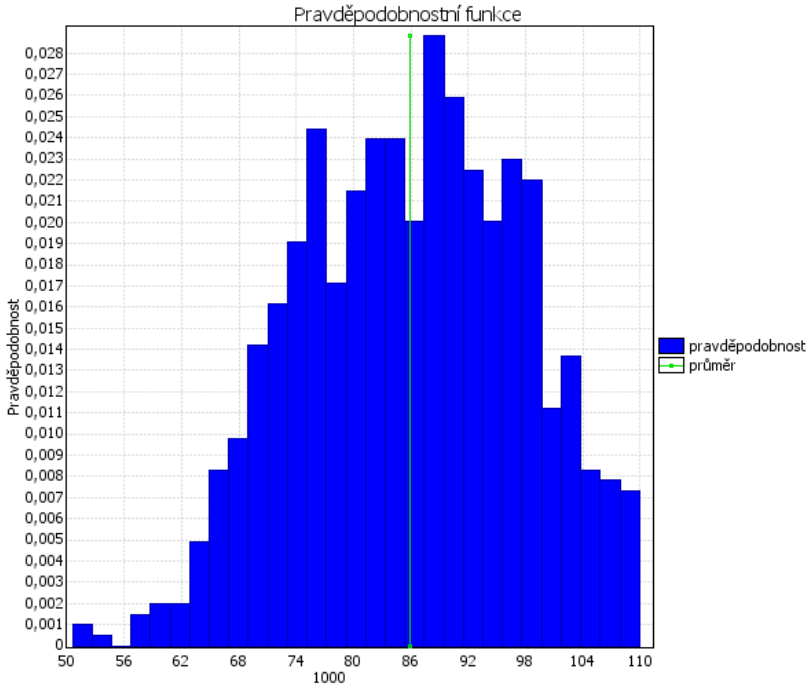
Obrázek A.13: Střídání intervalů v grafu



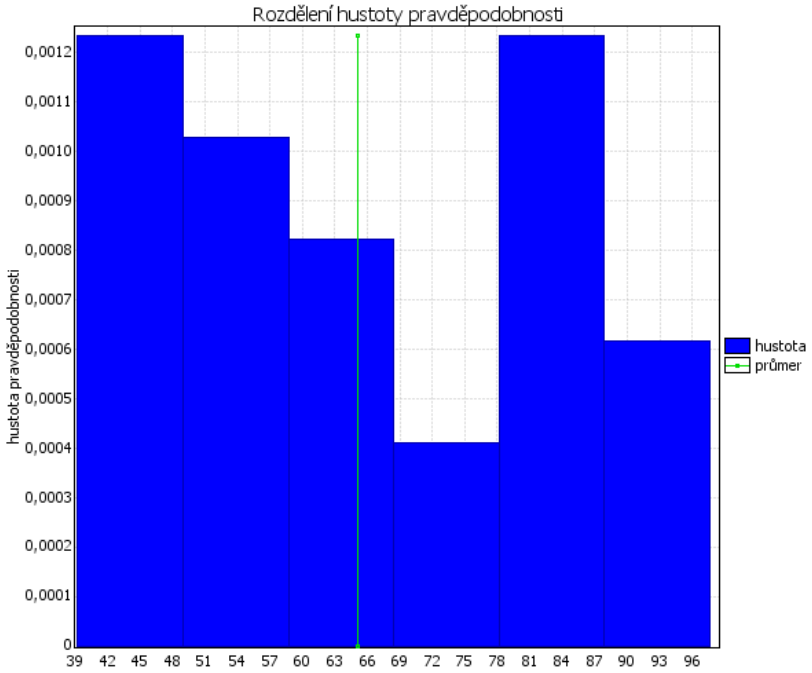
Obrázek A.14: Časy intervalů v grafu

A.2 Nízká vytíženost křižovatky

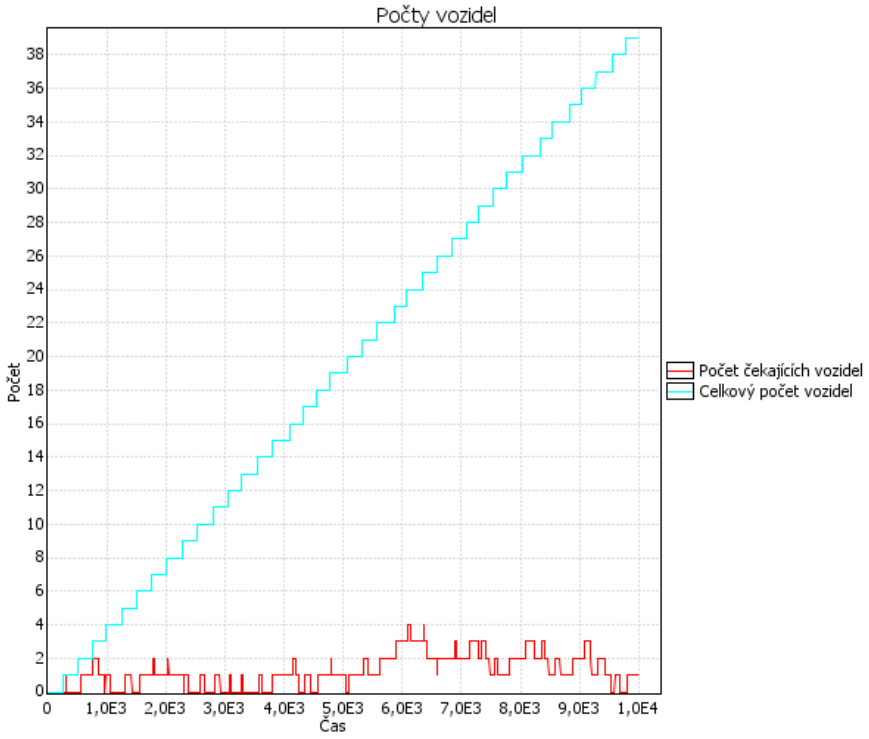
Adaptivní křižovatka



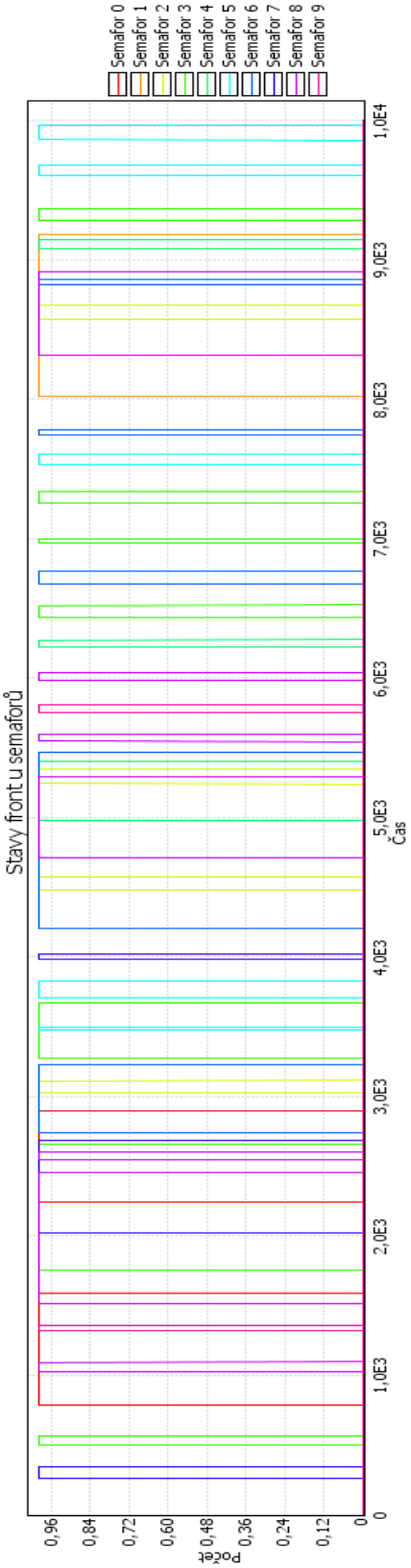
Obrázek A.15: Pravděpodobnostní funkce pro vztah 4.1



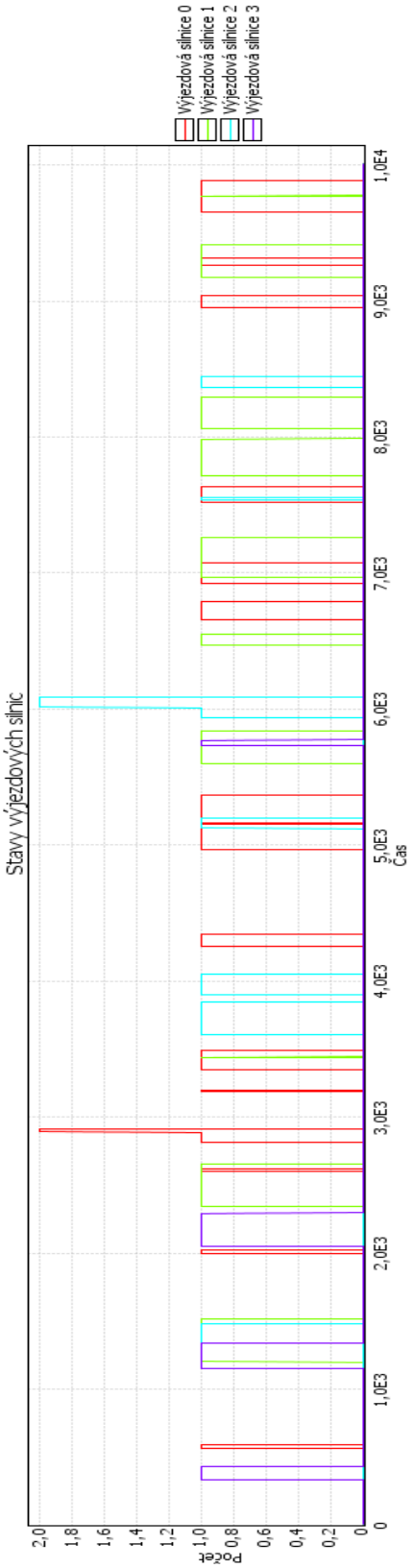
Obrázek A.16: Rozdělení hustoty pravděpodobnosti pro vztah 4.2



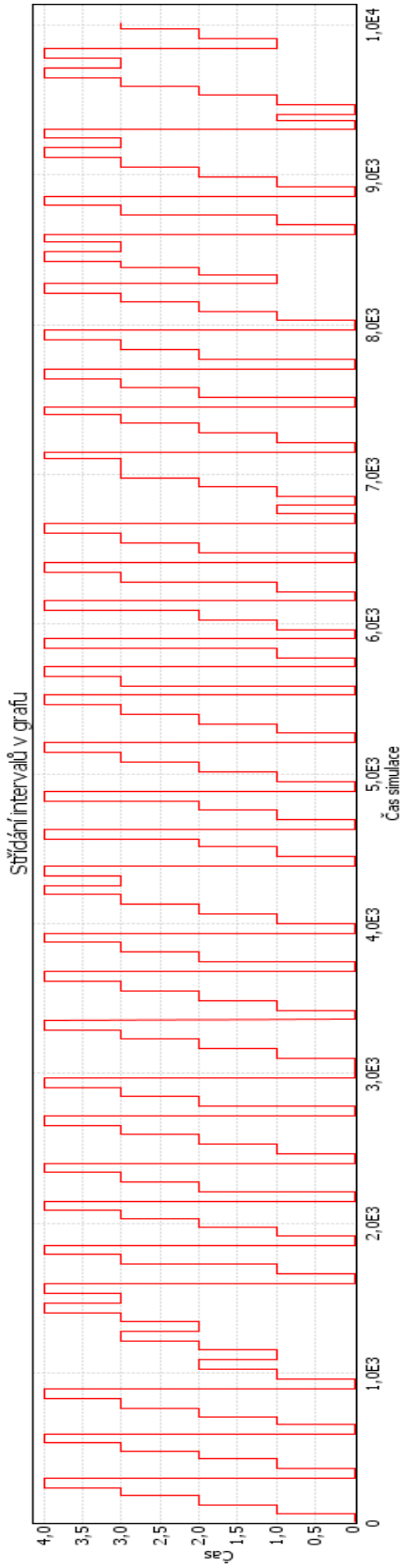
Obrázek A.17: Počet vozidel



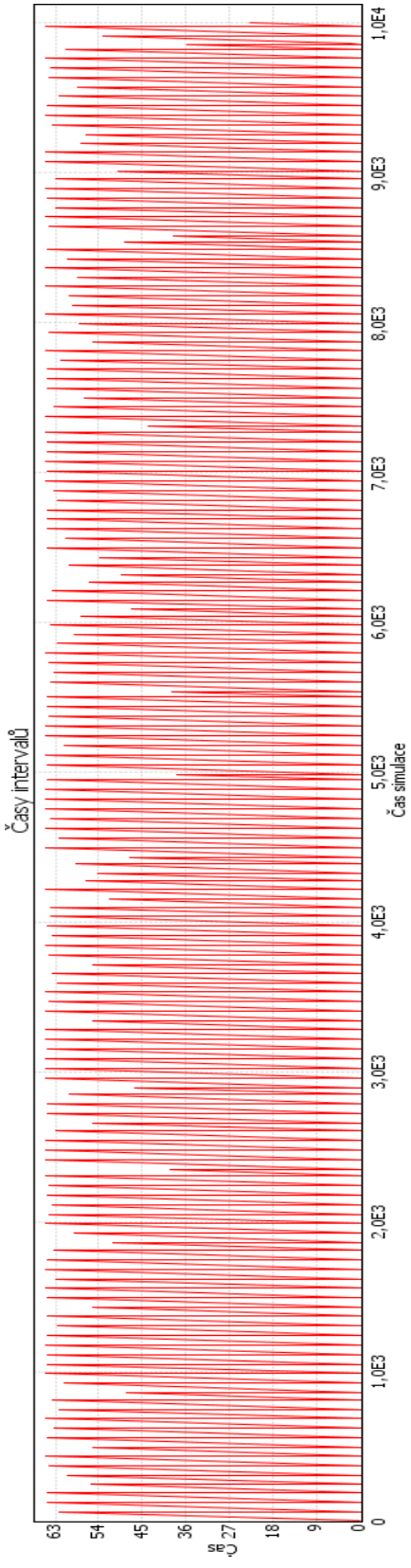
Obrázek A.18: Stavy front u semaforů



Obrázek A.19: Stavy výjezdových silnic

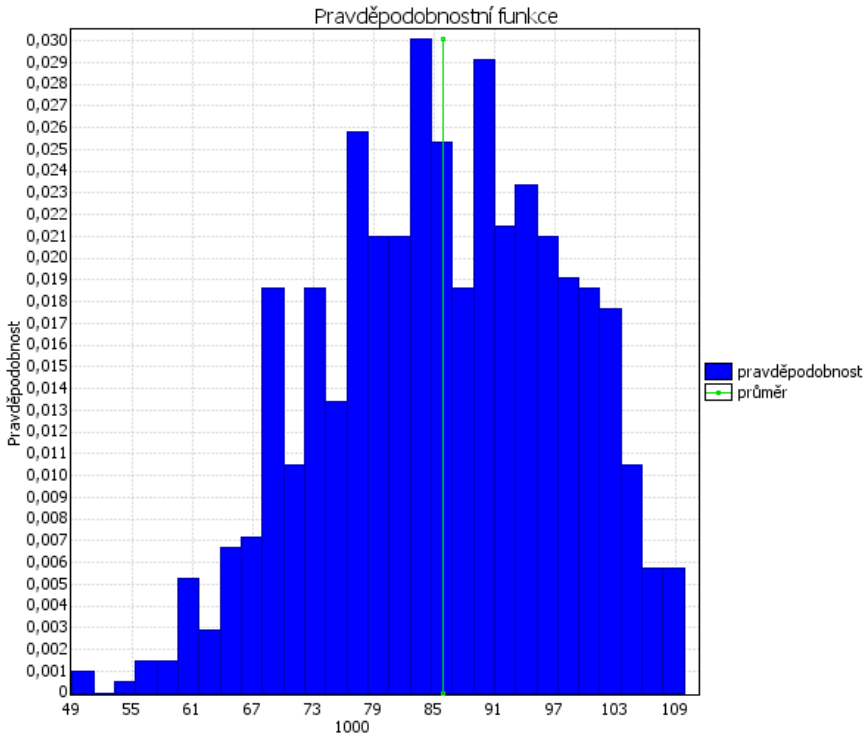


Obrázek A.20: Střídání intervalů v grafu



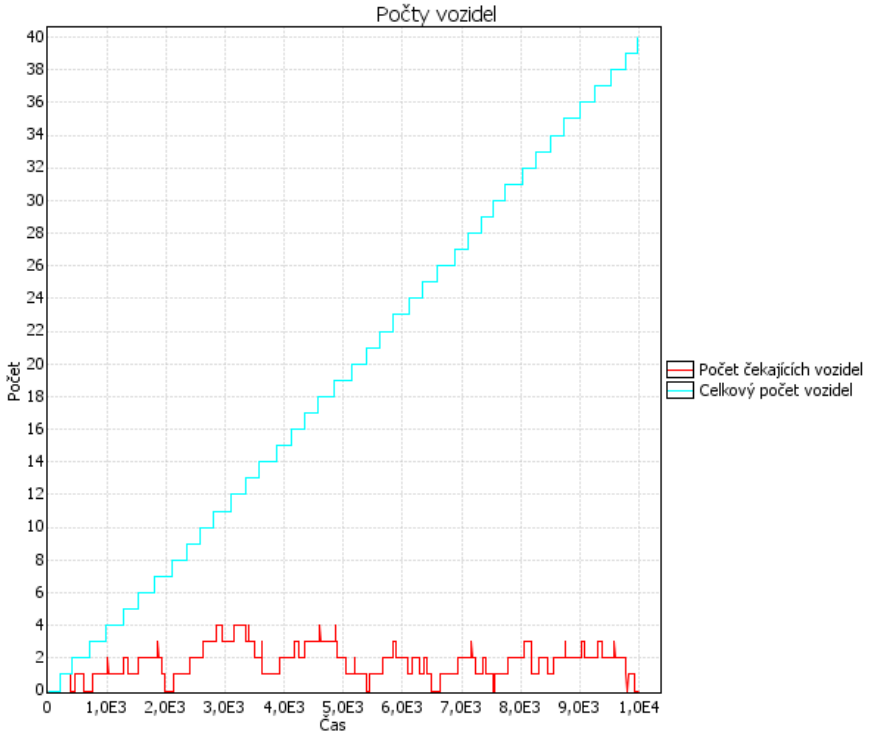
Obrázek A.21: Časy intervalů v grafu

Klasická křižovatka

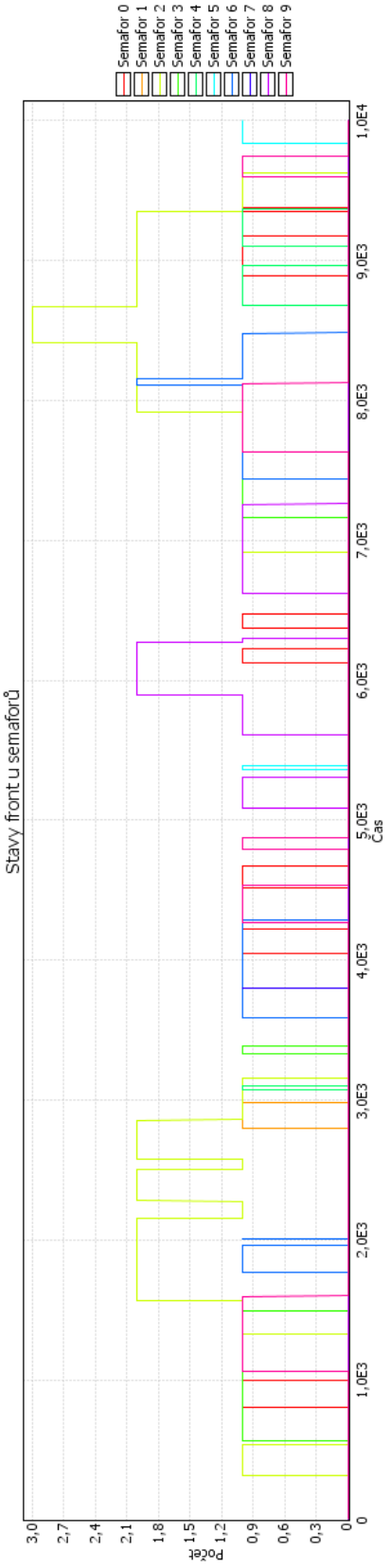


Obrázek A.22: Pravděpodobnostní funkce pro vztah 4.1

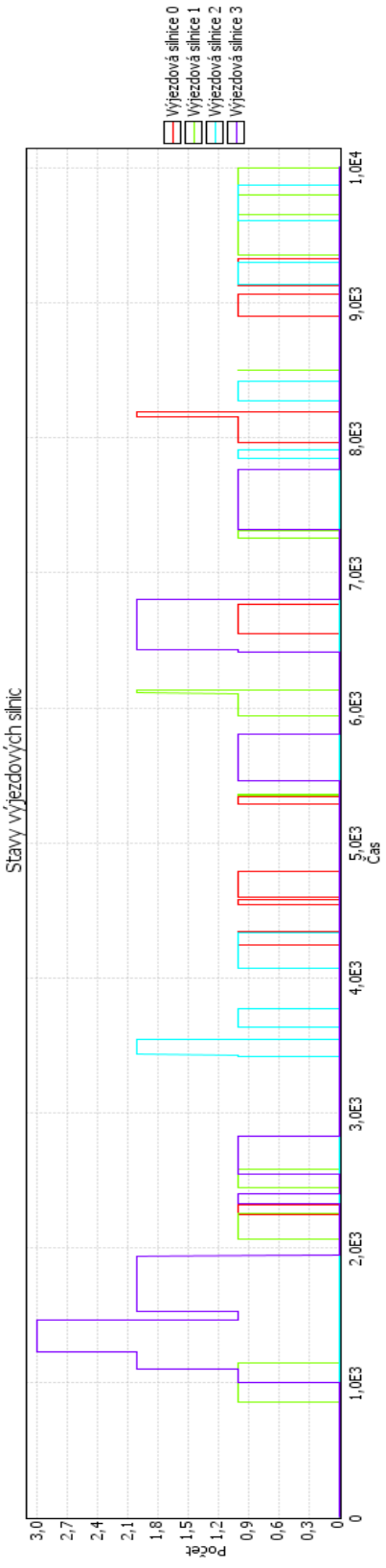
Obrázek A.23: Rozdělení hustoty pravděpodobnosti pro vztah 4.2



Obrázek A.24: Počet vozidel



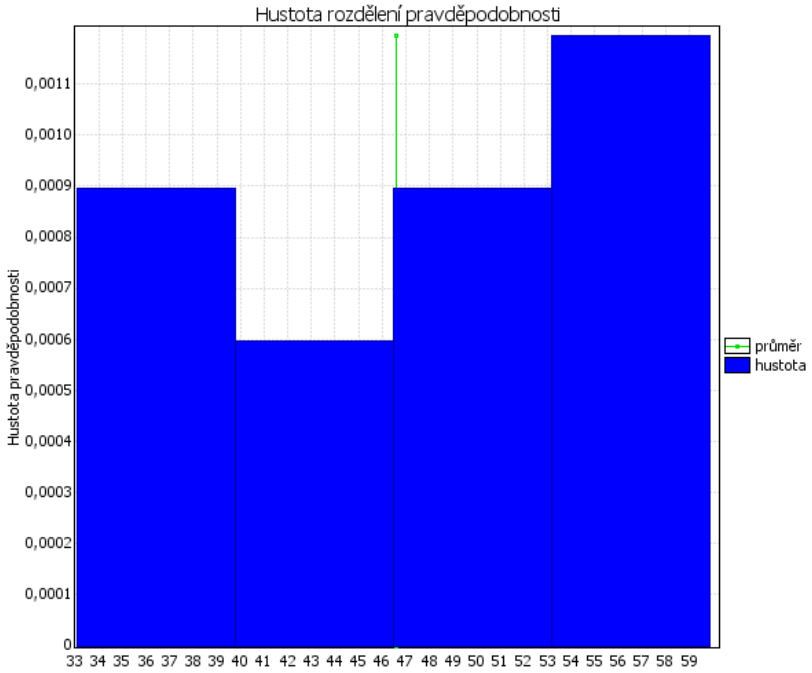
Obrázek A.25: Stavy front u semaforů



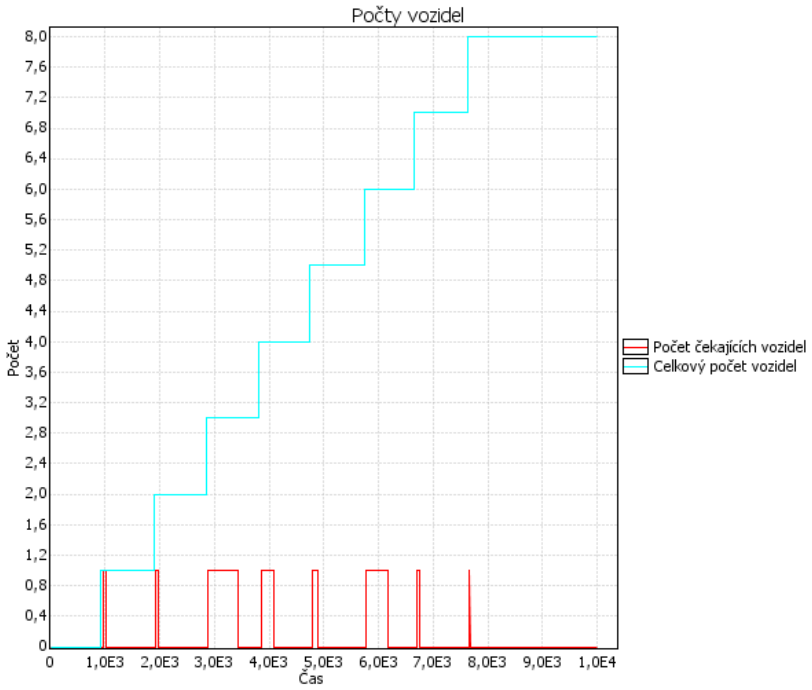
Obrázek A.26: Stavy výjezdových silnic

A.3 Chování křižovatky v noci

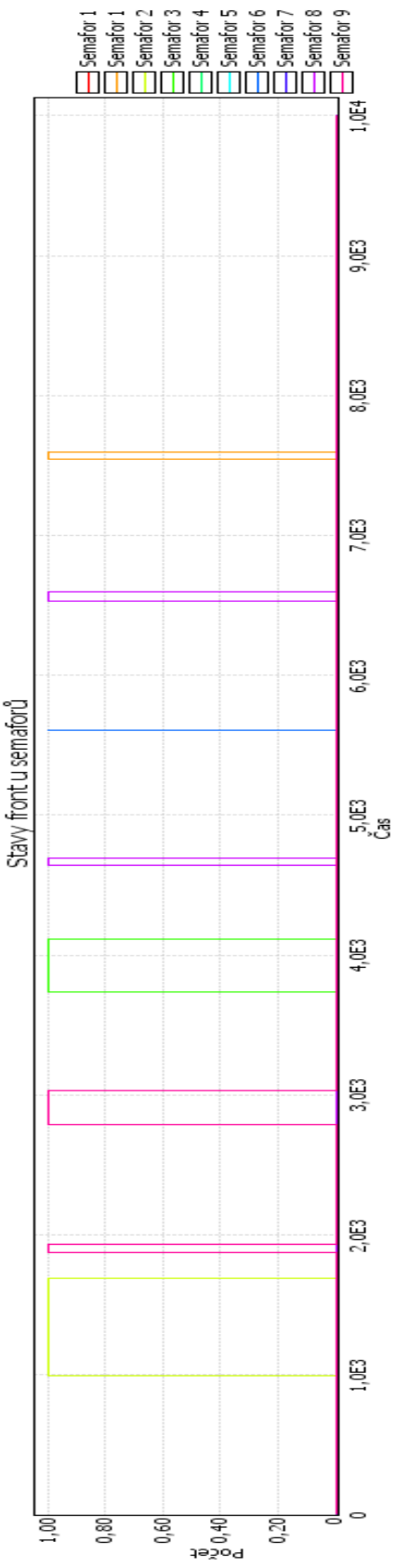
Adaptivní křižovatka



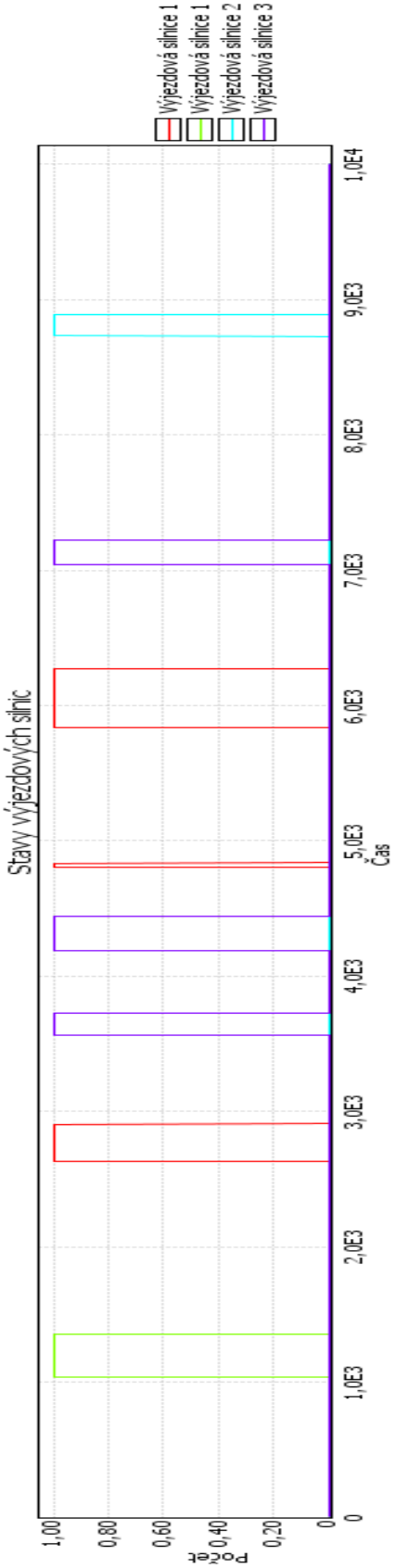
Obrázek A.27: Rozdělení hustoty pravděpodobnosti pro vztah 4.2



Obrázek A.28: Počet vozidel

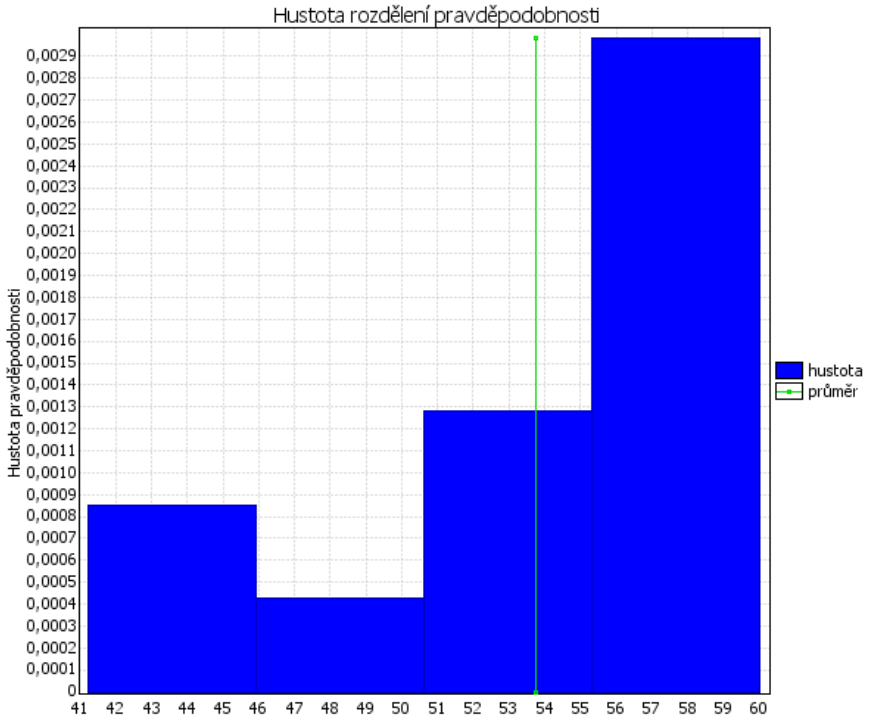


Obrázek A.29: Stavy front u semaforů

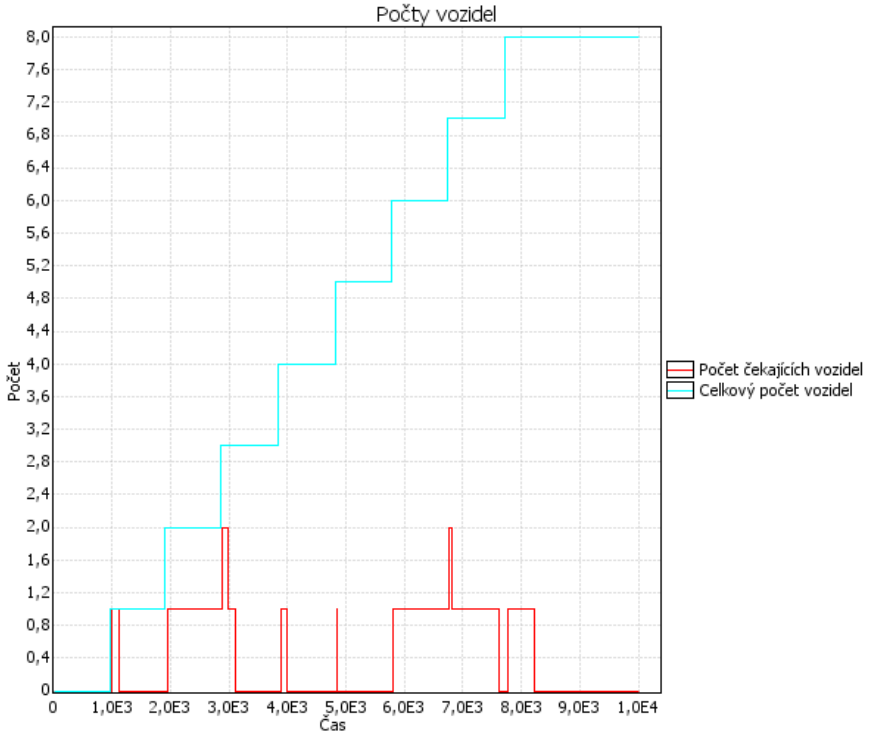


Obrázek A.30: Stavy výjezdových silnic

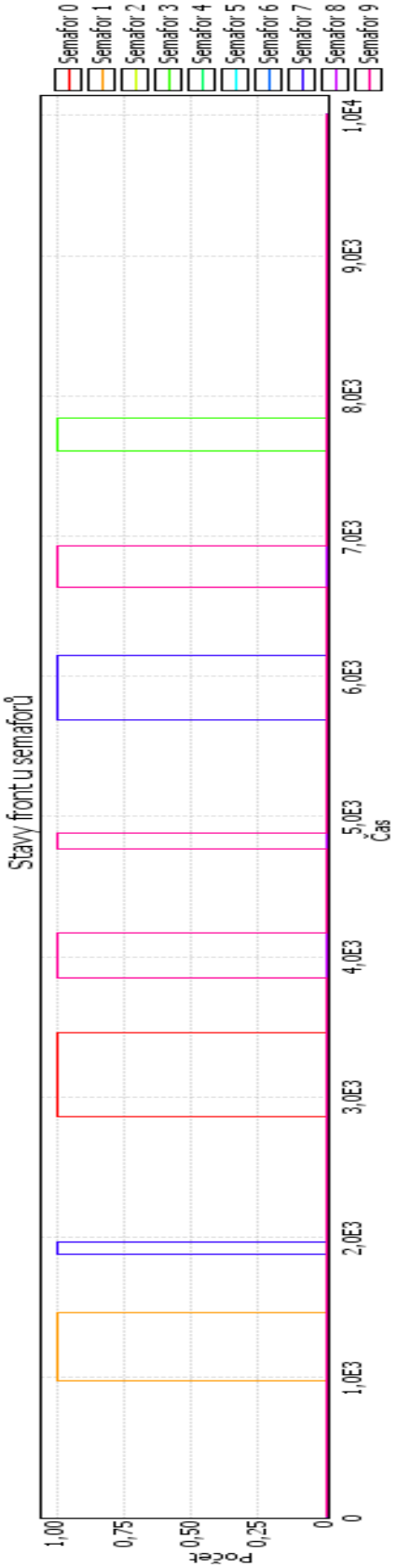
Klasická křižovatka



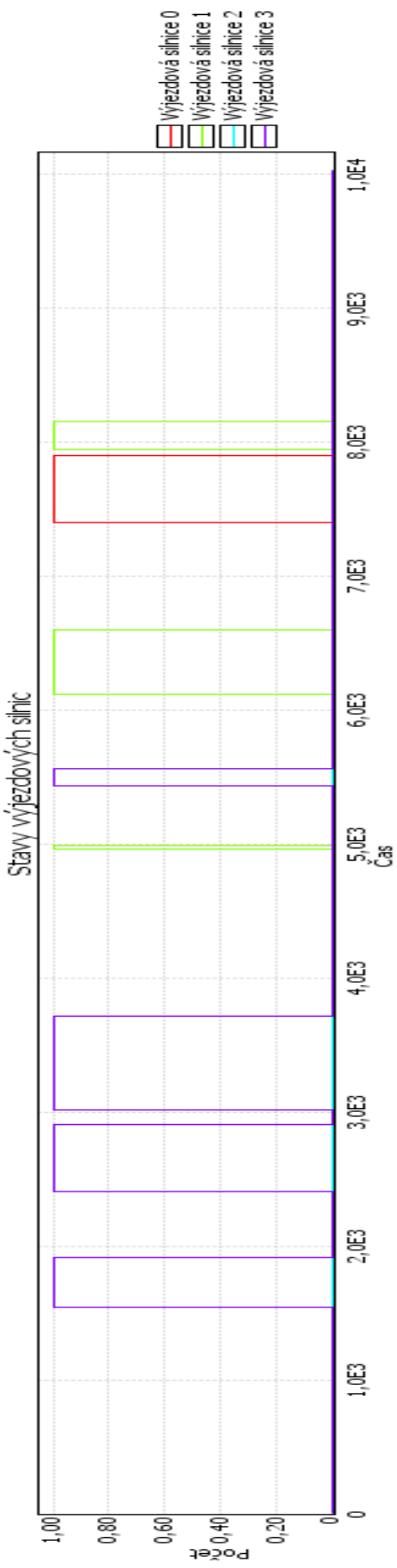
Obrázek A.31: Rozdělení hustoty pravděpodobnosti pro vztah 4.2



Obrázek A.32: Počet vozidel



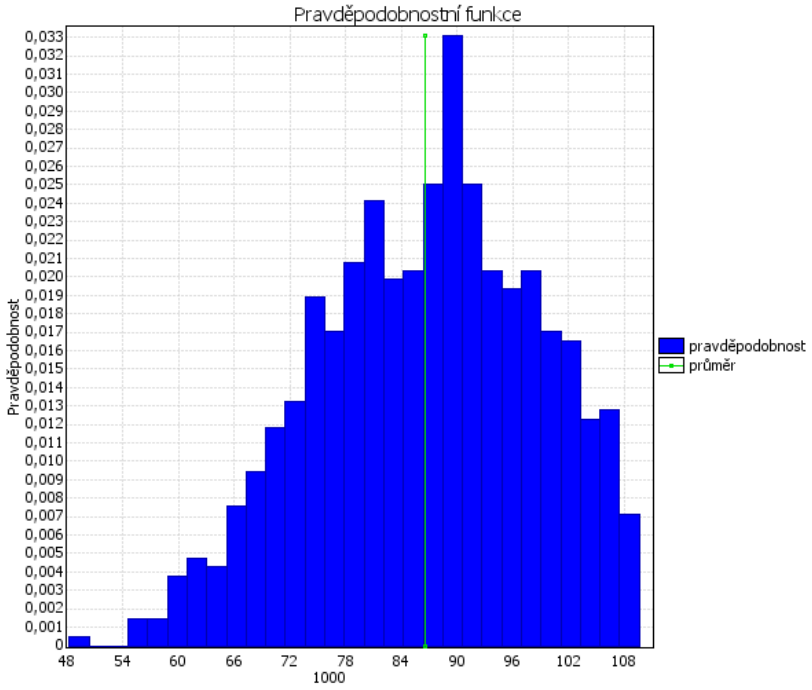
Obrázek A.33: Stavy front u semaforů



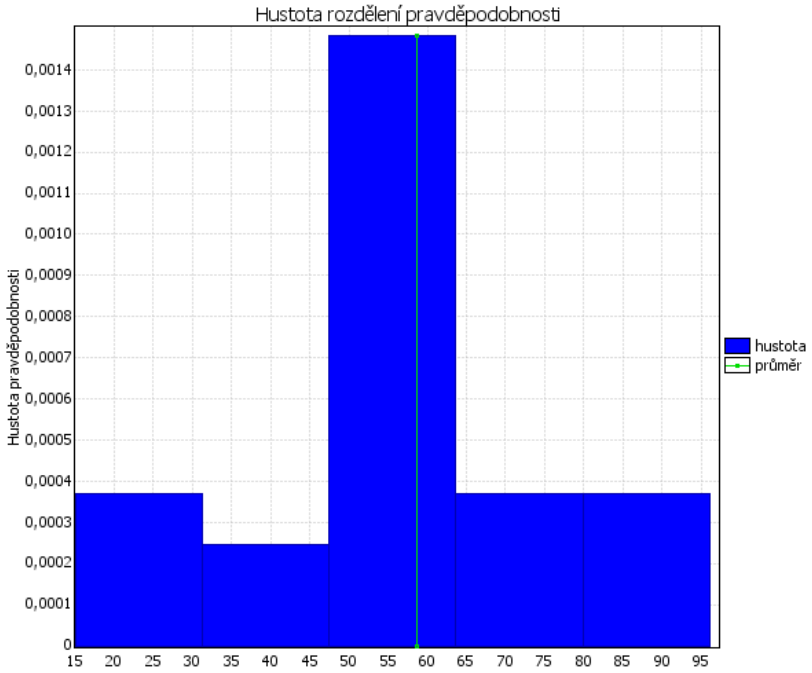
Obrázek A.34: Stavy výjezdových silnic

A.4 Jedna z výjezdových silnic je přehlcena

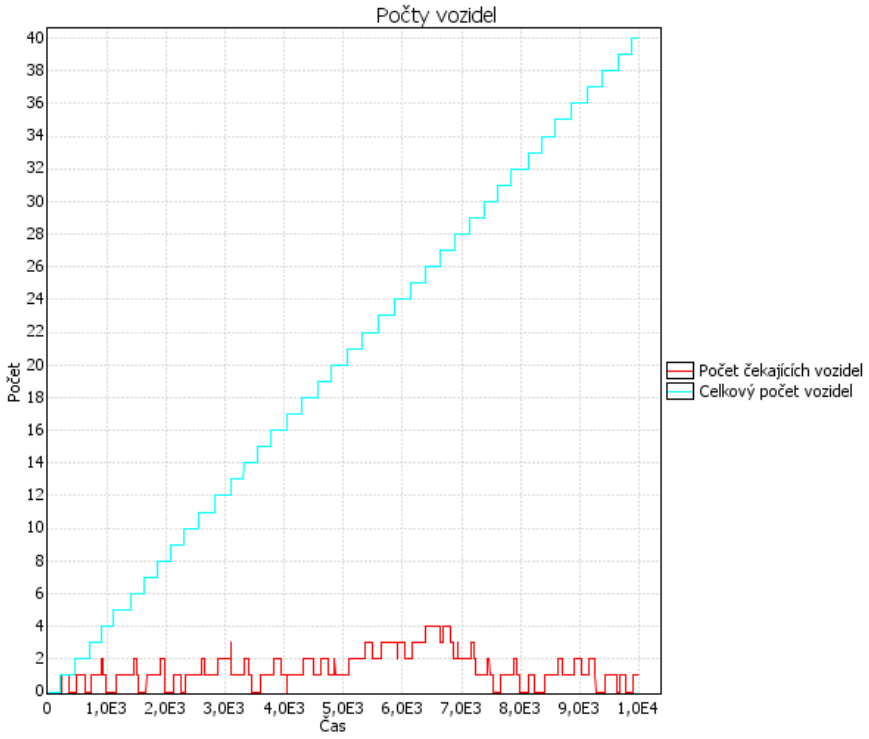
Adaptivní křižovatka



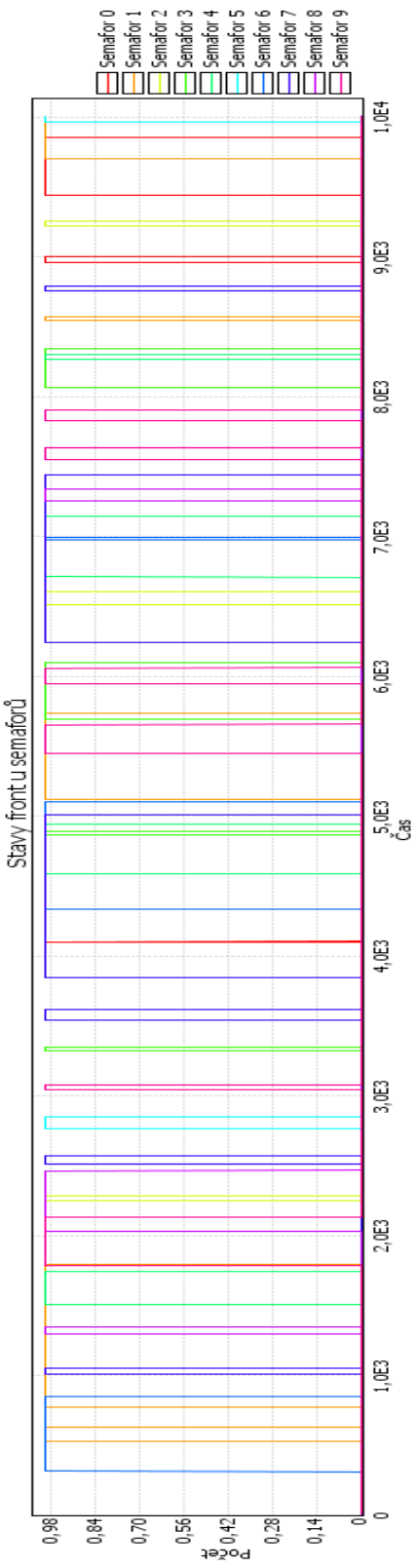
Obrázek A.35: Pravděpodobnostní funkce pro vztah 4.1



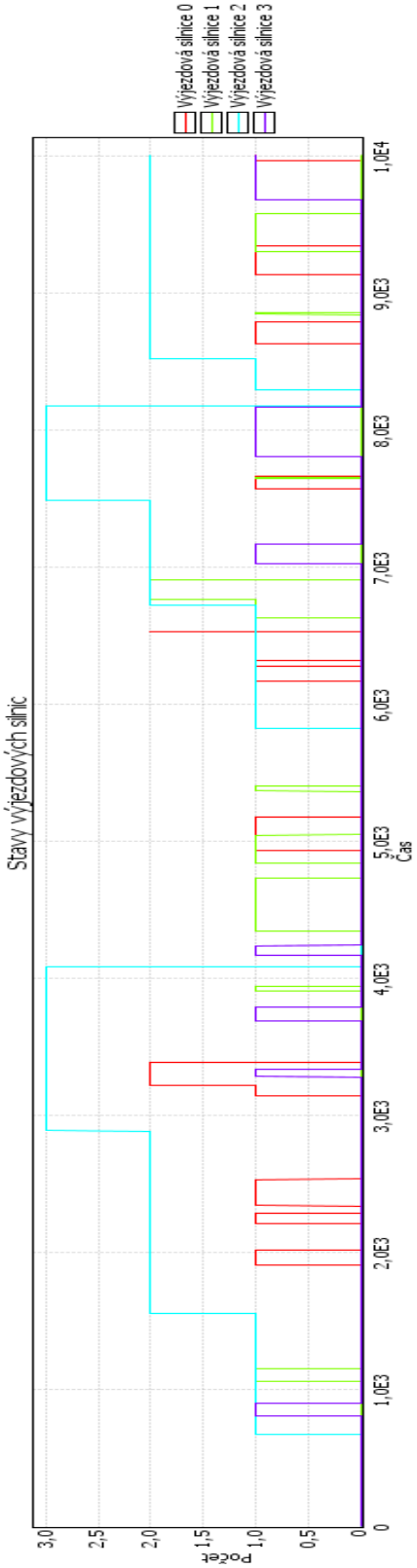
Obrázek A.36: Rozdělení hustoty pravděpodobnosti pro vztah 4.2



Obrázek A.37: Počet vozidel

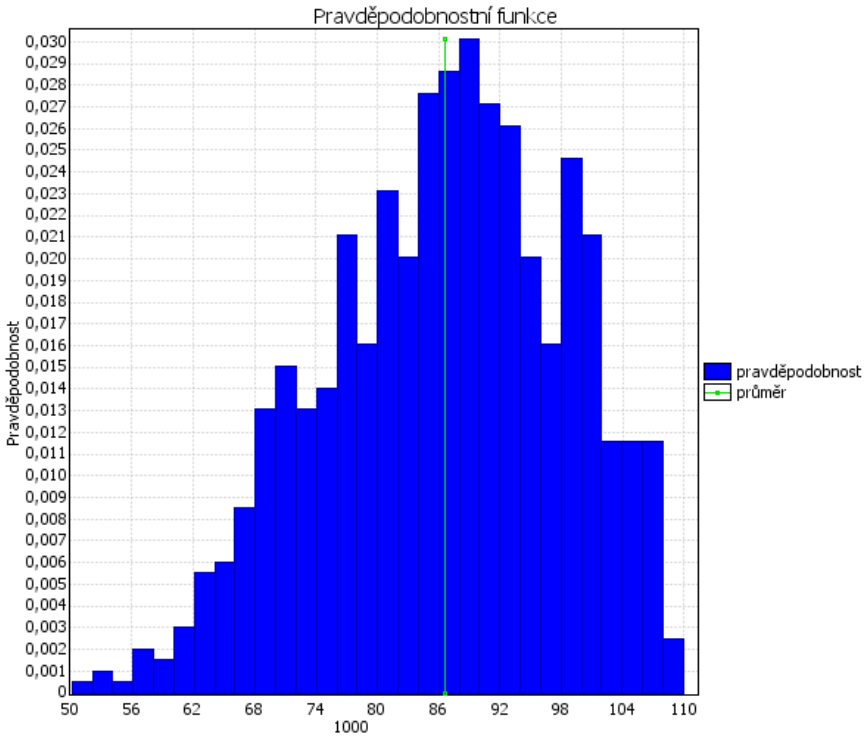


Obrázek A.38: Stavy front u semaforů

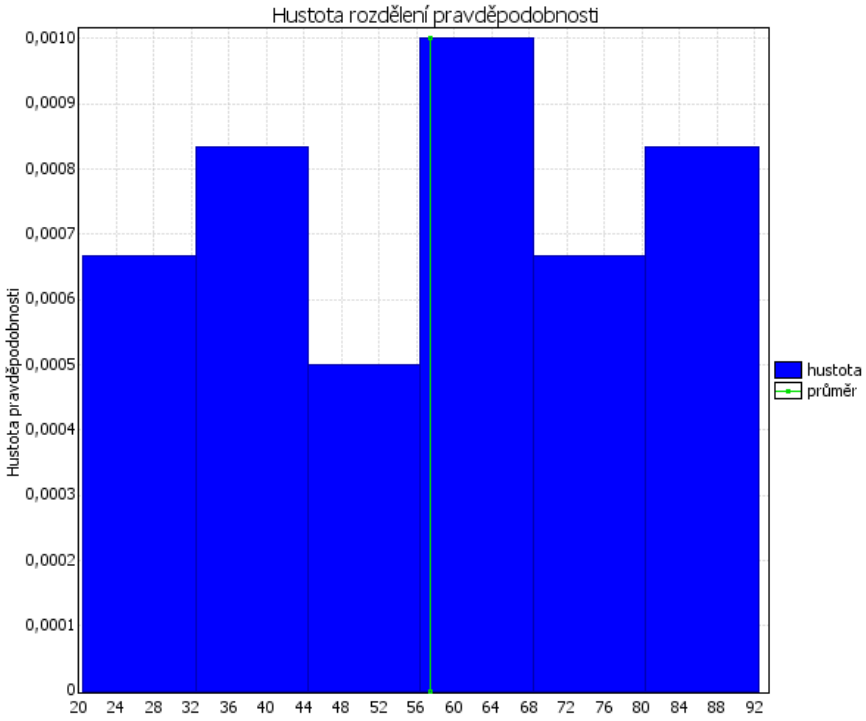


Obrázek A.39: Stavy výjezdových silnic

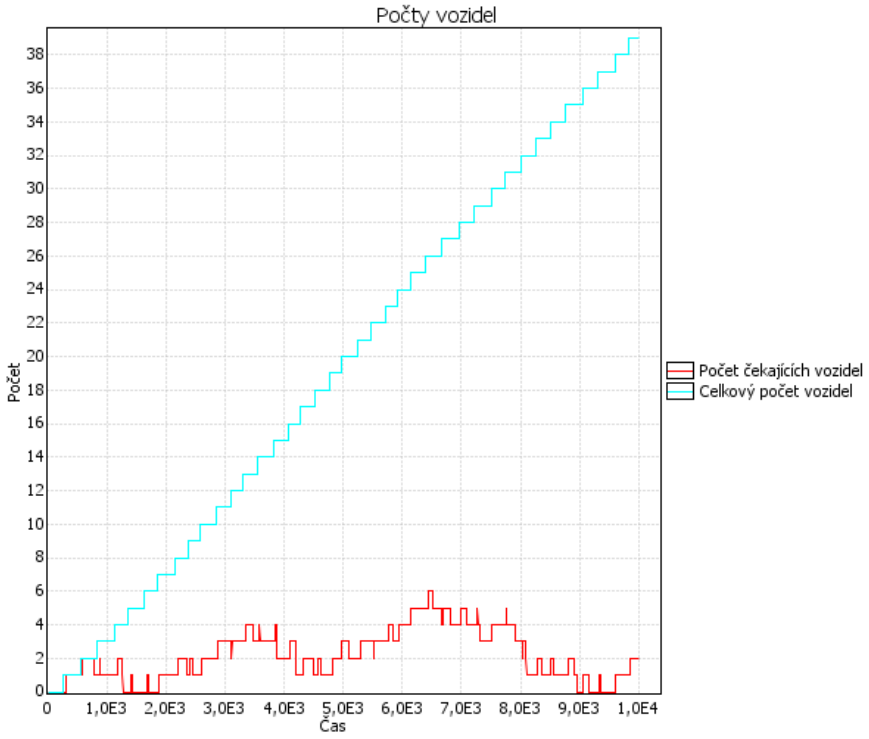
Klasická křižovatka



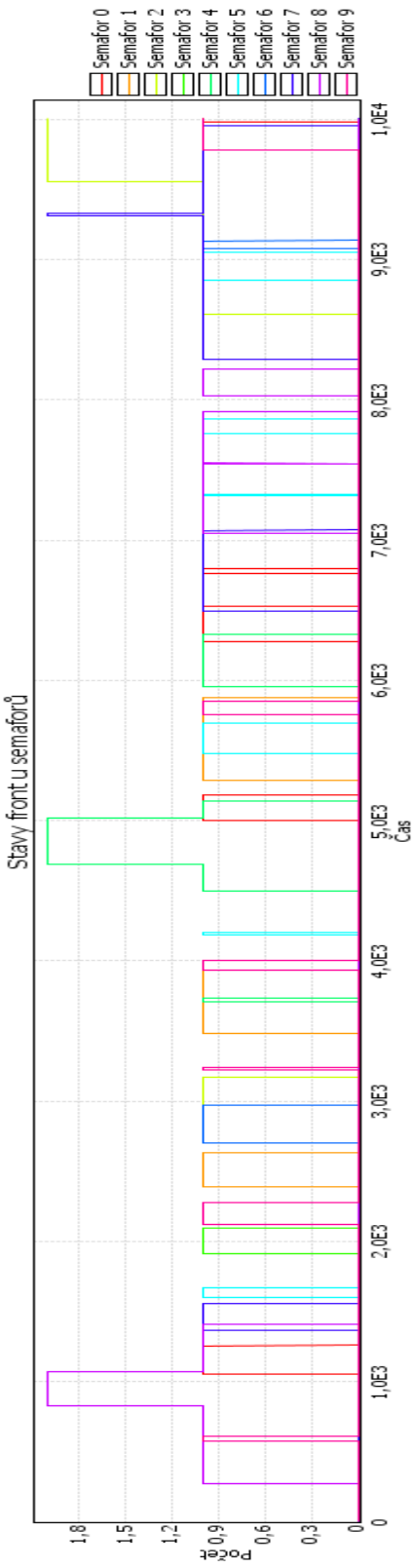
Obrázek A.40: Pravděpodobnostní funkce pro vztah 4.1



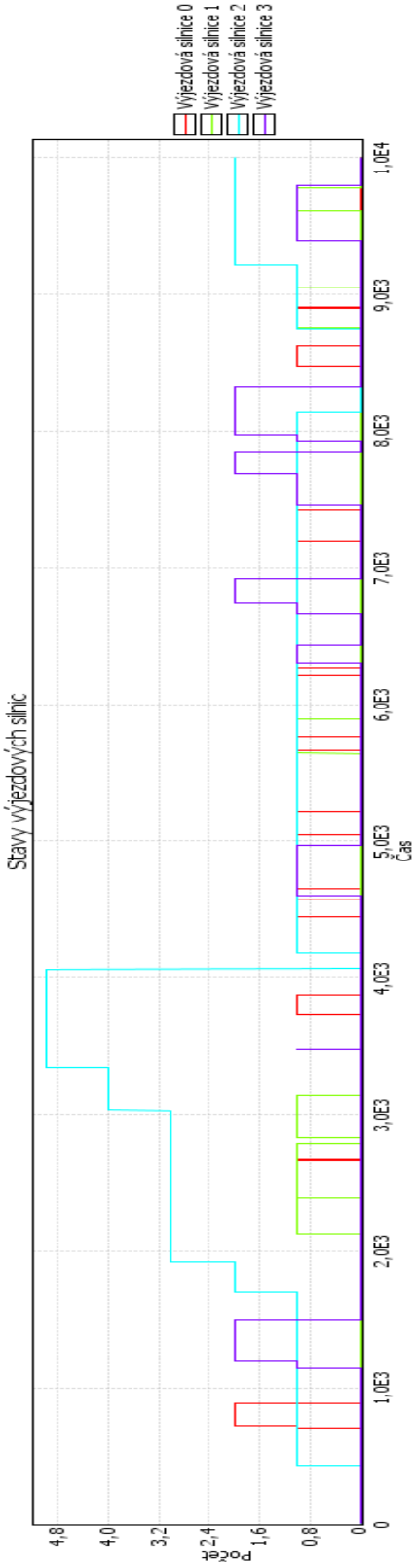
Obrázek A.41: Rozdělení hustoty pravděpodobnosti pro vztah 4.2



Obrázek A.42: Počet vozidel



Obrázek A.43: Stavy front u semaforů



Obrázek A.44: Stavy výjezdových silnic