

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: David Petr

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Agentní simulační model přepravní kontroly ve veřejné dopravě

Pokyny pro vypracování:

1. Seznamte se s přístupem a technikami agentního modelování a s frameworkem pro simulační modelování dopravy AgentPolis.
2. Seznamte se s problematikou přepravní kontroly v sítích veřejné dopravy.
3. Formalizujte chování cestujícího veřejné dopravy jako problém maximalizace užítku.
4. Navrhněte řídicí architekturu agenta-cestujícího implementující navržený formalizovaný model chování. Řídicí architektura by měla podporovat různou míru pozorovatelnosti a zapamatování si prostředí, různou míru adaptace a komunikace mezi cestujícími.
5. Navrženou architekturu agenta-cestujícího implementujte jako agenta v simulačním frameworku AgentPolis.
6. Vyhodnoťte chování výsledného simulačního modelu na reprezentativní sadě testovacích scénářů.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí bakalářské práce: Ing. Michal Jakob, Ph.D.

Platnost zadání: do konce zimního semestru 2013/2014


prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

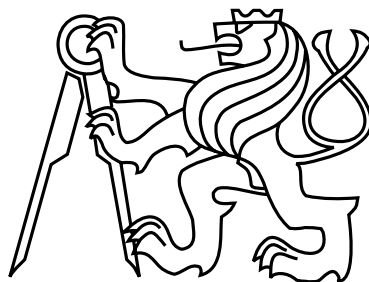



prof. Ing. Pavel Ripka, CSc.
děkan

České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra kybernetiky



Bakalářská práce

Agentní simulační model přepravní kontroly ve veřejné dopravě

David Petr

Vedoucí práce: Ing. Michal Jakob PhD.

Studijní program: Otevřená informatika, Bakalářský

Obor: Počítačové vědy

24. května 2013

Poděkování

Úvodem své bakalářské práce bych chtěl velmi poděkovat mému vedoucímu Ing. Michalu Jakobovi PhD. za jeho velkou trpělivost a pomoc při řešení problémů, které se v průběhu práce vyskytly. Velký dík patří taktéž Ing. Zbyňku Molerovi, který dokázal pomoci s implementací nad modelem AgentPolis.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Ve Včelné dne 22.5.2013


.....

Abstract

This bachelor thesis is devoted to the problematics of agent-based modelling in a public transport system. It modifies the implementation of a passenger on a software platform AgentPolis in a way, that he has a memory of all observations, which he has made. According to this memory, he can adaptively decide, whether he will buy a ticket or not. The main goal of this work was to add some parameters to experiments, so we can discuss their impact on final results.

Abstrakt

Tato bakalářská práce se zabývá agentním modelováním pasažéra v systému veřejné dopravy. Upravuje implementaci cestujícího na platformě AgentPolis tak, aby byl schopen si různou mírou pamatovat informace z prostředí a dokázal se podle nich adaptivně rozhodovat o nákupu jízdenky. Hlavním cílem bylo přidat do implementace různé parametry, a na základě nich diskutovat jejich závislost na výsledcích jednotlivých experimentů.

Obsah

1	Úvod	1
1.1	Způsob zkoumání multi-agentním přístupem	1
1.2	Význam simulace	3
2	Teoretické pozadí problému	4
2.1	Obecné teorie	4
2.2	Pojetí Colina Boyda	6
2.3	Pojetí Petera Kooremana	7
2.4	Srovnání obou matematických přístupů	7
3	Modely agentů	9
3.1	Prostředí	9
3.1.1	Faktory simulace	10
3.1.1.1	Vlastnosti prostředí	11
3.1.1.2	Parametry pasažéra	11
3.1.1.3	Parametry revizora	12
3.2	Pasažér	12
3.2.1	Paměť	13
3.2.2	Rozhodovací logika	15
3.2.3	Denní rozvrh	17
3.3	Inspektor	19
3.3.1	Rozhodovací logika	19
3.3.2	Denní rozvrh	19

4 Implementace	21
4.1 AgentPolis & FareEvasionLA	21
4.1.1 Typy nejdůležitějších tříd agentů	21
4.2 Pasažér	23
4.2.1 Zavedení faktorů	24
4.2.2 Zavedení vlastního rozhodování o jízdě	24
4.2.3 Implementace paměti	24
4.2.3.1 Datová struktura paměti	25
4.2.3.2 Funkce nad pamětí	25
4.2.3.3 Sdílení paměti	27
4.2.4 Úprava tras	27
4.2.4.1 Zavedení pasažérů s pravidelným rozvrhem	28
4.2.4.2 Zavedení zpátečních tras	28
4.2.4.3 Zavedení pravděpodobných odboček	28
4.2.4.4 Zavedení náhodných rozvrhů tras	29
4.3 Inspektor	29
4.3.1 Úprava tras	30
4.4 Automatizace experimentů	31
4.5 Zpracování výsledků	32
5 Experimenty	35
5.1 Uvažované parametry	36
5.2 Metriky	36
5.3 Jednotlivé experimenty	36
5.3.1 Srovnání typů pamětí pasažéra	37
5.3.2 Sdílení informací mezi pasažéry	39
5.3.3 Srovnání efektivity různých typů revizorů	41
6 Závěr	43
A Instrukce ke spuštění simulačního prostředí	47

Seznam obrázků

3.1	Schéma simulačního běhu v prostředí AgentPolis-FareEvasionLA	10
3.2	Schéma přístupu k paměti pasažérem	13
3.3	Schéma pasažérova denního cyklu	18
3.4	Schéma denního rozvrhu inspektora	20
4.1	Závislosti modulu FareEvasionLA	22
4.2	Schéma inicializace agentů Pasažér a Inspektor ve FareEvasionLA	22
4.3	Schéma základního životního cyklu pasažéra	23
4.4	Schéma implementace paměti pasažéra	26
5.1	Graf hodnot metriky „nekontrolován, s jízdenkou“pro všechny typy pamětí pasažéra v průběhu dní	38
5.2	Graf hodnot metriky „nekontrolován, bez jízdenky“pro tři míry sdílení infor- mací v průběhu dní (pasažéři s pravidelným rozvrhem)	40
5.3	Graf hodnot metriky „nekontrolován, bez jízdenky“pro tři míry sdílení infor- mací v průběhu dní (pasažéři s pravidelným rozvrhem)	41
5.4	Graf hodnot metriky „kontrolován, s jízdenkou“pro dva typy revizorských kon- trol.	42

Kapitola 1

Úvod

Tato práce se v následujících kapitolách bude zabývat agentním modelováním pasažérů v systému hromadné dopravy metra ve městě Los Angeles, USA. Cílem těchto kapitol je přiblížit teoretický model počítačového agenta, který je základní součástí simulace, následnou implementací úprav, které si níže popíšeme a v neposlední řadě také experimenty ovlivňované různými parametry prostředí.

Modelování chování pasažéra v systému veřejné dopravy nemá význam pouze při odhadování potřebné kapacity v konkrétních oblastech městského celku, nýbrž dokáže pomoci i s takovou problematikou, jako je optimalizace přepravní kontroly pasažérů. Proto jsem si jako téma mého zkoumání vybral právě tuto oblast, jde o oblast, s kterou je konfrontován každý člověk využívající městskou hromadnou dopravu a je to práce, ve které je vidět určitý hmatatelný výsledek.

Není zároveň potřeba zmiňovat, jak velký finanční potenciál toto zkoumání představuje pro jednotlivé dopravní podniky, ale o tom detailněji níže.

1.1 Způsob zkoumání multi-agentním přístupem

Veškeré experimenty a simulace jsou vytvářeny nad simulačním frameworkem AgentPolis¹, vyvíjeným výzkumnou skupinou Katedry počítačů na Fakultě elektrotechnické Českého vysokého učení v Praze. V rámci této skupiny byl také vytvořen modul FareEvasionLA, sloužící

¹<http://agents.fel.cvut.cz/projects/agentpolis>

právě pro účely simulace přepravní kontroly v systému metra města Los Angeles. Tento modul byl v rámci mé bakalářské práce modifikován, aby umožňoval implementaci rozhodovací logiky pasažéra o koupi resp. nekoupi jízdného, a aby dokázal odsimulovat větší množství experimentů s různými faktory.

Důležitým prvkem tohoto softwaru je tzv. multi-agentní řešení. V práci WOOLDRIDGE, M. Agent-based software engineering. *IEE Proceedings-Software*. 1997, 144, 1, s. 26–37 [10] můžeme nalézt následující definici agenta:

Agent je izolovaný počítačový systém, který se nachází v určitém prostředí a je schopen flexibilních a autonomních akcí, vedoucích ke splnění objektivního účelu pro který byl navržen.
(přeloženo z anglického jazyka)

Tato definice jasně napovídá, že jiný přístup by byl pro model veřejné dopravy velice složitý. Ten agentní má velkou výhodu v tom, že se může chovat jako jakási „populace“ [4], která mezi sebou dokáže v různé míře komunikovat a daleko lépe tak dokáže odpovídat reálnému světu.

Abychom to tedy trochu shrnuli, tak mezi nejzákladnější výhody patří: [2]

- **Zachycení emergentních² fenoménů:** to jsou nápadné jevy složitěho systému, které vznikají jednoduchými interakcemi malých, autonomních systémů, a které je velmi složité obecně odvodit.
- **Flexibilita:** systém je velmi jednoduše modifikovatelný v rámci daného problému - jednoduše přidáme daný počet agentů nebo můžeme upravovat jejich chování tím, že omezíme nebo naopak přidáme schopnost vnímání okolního prostředí.
- **Realistický model:** oproti standardnímu přístupu, agentní modelování poskytuje reálnou simulaci právě díky tomu, že se mohou stávat různé nepředvídatelné situace, které by se v centrálně řízeném systému neobjevily (pokud se samozřejmě nejedná o čistě deterministické chování, kde takové situace neuvažujeme). Budeme-li systém řídit centrálně, informace o chování agentů bude pouhým zobecněným a průměrným modelem agentního přístupu.

²emergence - výskyt vlastností nadřazeného a složitěho systému pomocí malých interakcí všech součástí uvnitř

- **Možnost adaptace pasažérů na prostředí:** nespornou a obrovskou výhodou agentního přístupu je jistě možnost postupného učení agentů, což v centrálně řízeném systému nemáme šanci nasimulovat.

1.2 Význam simulace

Pro praktický příklad z hlediska významu experimentování s pasažéry a revizory nemusíme pro příklad chodit nijak daleko. Stačí se podívat na situaci v hlavním městě Praha, která provozuje systém metra v tzv. otevřeném systému, což umožňuje komukoliv vstoupit do stanice, aniž by se u vstupu musel prokázat platným jízdním dokladem. I přes četné revizorské kontroly se odhaduje možný počet černých pasažérů dokonce až ke 20% [6]. Ovšem pro zavedení uzavřeného systému metra by bylo potřeba instalovat přes 800 nových turniketů [6], což představuje obrovský výdaj s návratností mnoha let.

Právě proto se zdá být počítačová simulace dopravního systému velmi zajímavou alternativou a doplňkem ke zmíněnému konvenčnímu řešení. Jistě, můžeme namítnout, že nikdy nelze vytvořit přesnou simulaci, že chování pasažéra ovlivňuje mnoho dalších faktorů, které nelze snadno nadefinovat. I přesto by mohl tento přístup přispět alespoň k malé optimalizaci strategie revizorů, což by i při malé eliminaci černých pasažérů přineslo nemálo finančních prostředků do rozpočtu dopravních podniků.

Tato bakalářská práce si však v žádném případě neklade za cíl, jakkoliv zkoumat optimalizaci kontroly jízdného. Má spíše experimentálně zjistit a vyzkoušet, které faktory ovlivňují agenty při jejich rozhodování nejvíce, popřípadě jak rychle se pasažéři na danou situaci adaptují.

Kapitola 2

Teoretické pozadí problému

Bohužel pro tuto problematiku neexistuje příliš materiálů. Většina se jich věnuje pouze obecnému, nikoli matematickému popisu jako takovému. Existují však dvě teoretické práce, na které se v následujících odstavcích blíže podíváme. Nejprve si však shrneme ty obecnější.

2.1 Obecné teorie

K nejzákladnějším dělením systémů veřejné dopravy patří bezesporu rozdělení na systémy *a)* otevřené; *b)* uzavřené (s turnikety na každé stanici). Uzavřený systém má oproti otevřenému obrovskou výhodu v daleko lepší eliminaci „černých pasažérů“. Například v New Yorku došlo zavedením uzavřeného systému k výraznému poklesu jízd bez koupě jízdného z 6,9% na 1,3% [9]. Z údajů je však patrné, že ani tento systém nedokáže omezit ztráty na úplné minimum - vždy se najde cesta, jak i tento systém „obcházet“. Právě na něm byl vysledován také další překvapivý fakt, a to ten, že míra vyhýbání se zaplacení jízdného se v průběhu dne a také dokonce v průběhu roku mění. Nebo také fakt, že největší počet „neplatících“ se vyskytuje na nejméně frekventovaných místech, a ne naopak, jak by se dalo očekávat. Ač by se mohlo zdát, že uzavřený systém je nejlepší ze všech možných, tak situace není tak jasná, neboť zavedení turniketů do všech stanic představuje enormní počáteční investici pro dopravní podnik a jeho návratnost se pohybuje v řádu mnoha let - konkrétní údaje bohužel zdroj neuvádí.

Zkusme se však podívat na problém z trochu jiného pohledu a rozdělme si na jednotlivé

typy také pasažéry. Ti se dělí do následujících skupin [1]: a) náhodní - to jsou ti, kteří si pravidelně jízdenku kupují, zapomenou si však peněženku doma, nebo si jízdenku neoznačí; b) tzv. calculating dodgers - tento termín by se dal volně přeložit jako „vypočítaví neplatiči jízdného“, kteří se vždy před každou cestou podle rizika kontroly rozhodují, zda-li si mají jízdenku koupit nebo ne. Je to také hlavní skupina, na kterou cílí boj proti cestujícím bez jízdenek pomocí agentního modelování; c) notoričtí neplatiči jízdného - obecně lidé z nejnižších příjmových skupin, u kterých je velmi složité vymáhat jakékoliv sankce, a proti kterým se velmi těžko bojuje.

Proti „černým pasažérům“ se dá bojovat i jinými způsoby. Tak například v Edmontonu v Kanadě probíhal přibližně po jeden rok [5] tzv. „crackdown“, což znamenalo, že se po tuto dobu navýšil počet personálu kontrolující jízdenky. Nakonec se však ukázalo, že snížení míry pasažérů jezdících „načerno“, ke kterému skutečně došlo, netrvalo příliš dlouho (cca. půl roku) a procento lidí bez jízdenky se vrátilo do původního stavu. Bohužel více revizorů natrvalo by se finančně nevyplatilo.

Ne vždy se ovšem cestující rozhodují podle počtu revizorů či podle výše pokuty. V systému veřejné dopravy v Hong Kongu se mezi pasažéry provedl dotazníkový výzkum [8] co ovlivňuje jejich koupi resp. nekoupi jízdného. A ač se zdá tento dopravní systém velmi odlišný, tak co se týče rozhodování pasažérů se lišit příliš nemusí. Například důvody, proč nekupují jízdné jsou následující. Mezi nejzmiňovanější patřil ten, že si cestující zapomněli dobít kartu nebo to nestihli, protože spěchali. Druhým důvodem bylo špatné označení jízdenky a překvapivě až třetím důvodem v pořadí byly peníze - cestující má málo peněz a tímto se snaží ušetřit. Překvapivě 12% cestujících uvedlo, že si jízdenku nekupují úmyslně ze zásady. Zajímavý byl také dotaz kladený pasažérům, jak by si mysleli, že by se měl eliminovat počet neplatících cestujících. Mezi prvními návrhy totiž figuroval návrh na rozhlasové varování ve stanicích a ve vzdělávacích programech pro děti na tuto tematiku. Minimum cestujících by navrhovalo zvýšit počet revizorů (8%), zato vyšší pokutu opakovaným „černým pasažérům“ navrhovalo celých 34%.

V následujících dvou kapitolách se nyní podíváme na matematickou teorii pasažérova rozhodování.

2.2 Pojetí Colina Boyda

V prvním článku BOYD, C. et al. Fare evasion and non-compliance: A simple model. *Journal of Transport Economics and Policy*. 1989, s. 189–197 [3] je popisován rozhodovací model cestujícího jako očekávaný peněžní užitek daný payoff-maticí¹:

π - Pravděpodobnost kontroly	Detekován $p(D) = \pi$	NeDetekován $p(D) = 1 - \pi$
Evade (nekoupit j.)	$-p$ (pokuta)	0
NonEvade (koupit j.)	$-f$ (jízdné)	$-f$ (jízdné)

Kdy cestující se bude rozhodovat podle nerovnosti:

$$EMV(E) > EMV(NE) \quad (2.1)$$

(EMV=ExpectedMonetaryValue=OčekávanýPeněžníUžitek), neboli:

$$p\pi > f \quad (2.2)$$

- tedy cestující se rozhodne pro nákup jízdenky, pokud bude splněna následující nerovnost:

$$f/p > \pi \quad (2.3)$$

kde:

f : cena jízdenky

p : cena pokuty

π : pravděpodobnost chycení revizorem na trase pro kterou se cestující rozhoduje o koupi jízdného

To v praxi znamená, že pokud je hodnota pokuty $2x$ hodnota jízdného, musí být pravděpodobnost chycení revizorem větší $\frac{1}{2}$.

Autor dále v článku počítá s procentem tzv. „pocitivých pasažérů“, což jsou cestující, kteří si jízdenku kupují vždy. Pro nás je však důležitý tento zjednodušený model, neboť přidáním této skupiny agentů by se chování výsledků v zásadě nezměnilo, jen by se výsledky mohly „posunout“ k lehce jiným číslům.

¹nejznámější analýza rozhodování, která tabulkovým přehledem shrnuje všechny faktory rozhodnutí

2.3 Pojetí Petera Kooremana

Naproti tomu Peter Kooreman ve své práci KOOREMAN, P. Fare Evasion as a Result of Expected Utility Maximisation: Some Empirical Support. *Journal of transport economics and policy*. 1993, s. 69–74 [7] neuznává čestné pasažéry jako takové, ale zavádí pojem averze pasažéra k risku. Jinak je ovšem model rozhodování velmi podobný.

Pasažér si jízdenku koupí tehdy, pokud:

$$U_k > p_x U_f + (1 - p_x) U_0 \quad (2.4)$$

nebo po úpravě:

$$\frac{(U_k - U_0)}{(U_f - U_0)} \leq p_x \quad (2.5)$$

kde,

U_k : užitek z koupě jízdenky

U_0 : užitek z nekoupě jízdenky a nepokuty

U_f : užitek z nekoupě jízdenky a pokuty

p_x : pravděpodobnost kontroly pasažéra

Kooreman se vymezuje vůči teorii C.Boyda, protože je toho názoru, že nelze vyjádřit stabilní počet „čestných pasažérů“. Je přesvědčen, že pasažérova ochota riskovat stoupá nepřímo úměrně s pravděpodobností chycení inspektorem.

2.4 Srovnání obou matematických přístupů

Přes drobné rozdíly jsou však tyto přístupy velmi podobné a opírají se o jednoduchou logiku pravděpodobnosti zisku či ztráty. I přes svou jednoduchost je ovšem nutné uznat, že podle jiných a složitějších úvah se pasažér nakonec nerozhoduje, a tak tento model pro následující simulaci přijmeme jako výchozí. Pasažér totiž opravdu pokládá na misky vah dvě kritéria –

na jedné je cena jízdenky a na druhé cena pokuty. Pasažér si nějakým způsobem přibližně pamatuje kolikrát revizora viděl a kolikrát (pro něj teoreticky zbytečně) si již zakoupil jízdenku bez toho, aniž by byl kontrolován a vytváří si tak empiricky přibližnou pravděpodobnost, která ovlivňuje jeho rozhodování. Jistě, dalo by se namítnout, že existují i jiné faktory ovlivňující pasažéra, např. jeho bohatství nebo také zmíněná ochota riskovat, nicméně zohlednit je všechny by bylo ale velmi náročné a nemuselo by ani přinést očekávaný výsledek (jak již bylo zmíněno výše).

Kapitola 3

Modely agentů

Každý agent má definován svůj obecný životní cyklus (pravidla chování), podle kterých jedná. Agenti mají k dispozici určité akce: schopnosti pozorování, učení se a samozřejmě také prostor voleb. V následující kapitole si tedy nastíníme mechanismus fungování jednotlivých agentů včetně zformalizování, abychom důkladně porozuměli fungování simulace.

3.1 Prostředí

Prostředí simulace jako takové samozřejmě není agent, nicméně vyskytují se zde také agenti v podobě vozů metra, které se autonomně pohybují a které mohou přepravovat jak revizory, tak pasažéry. Ti samozřejmě nemají přílišný význam, jezdí podle jízdního řádu, nijak se nerozhodují a nijak nemohou ovlivnit průběh experimentů. Samotné prostředí však všem agentům poskytuje důležité prostředky jako jsou akce a proměnné potřebné pro jejich chování.

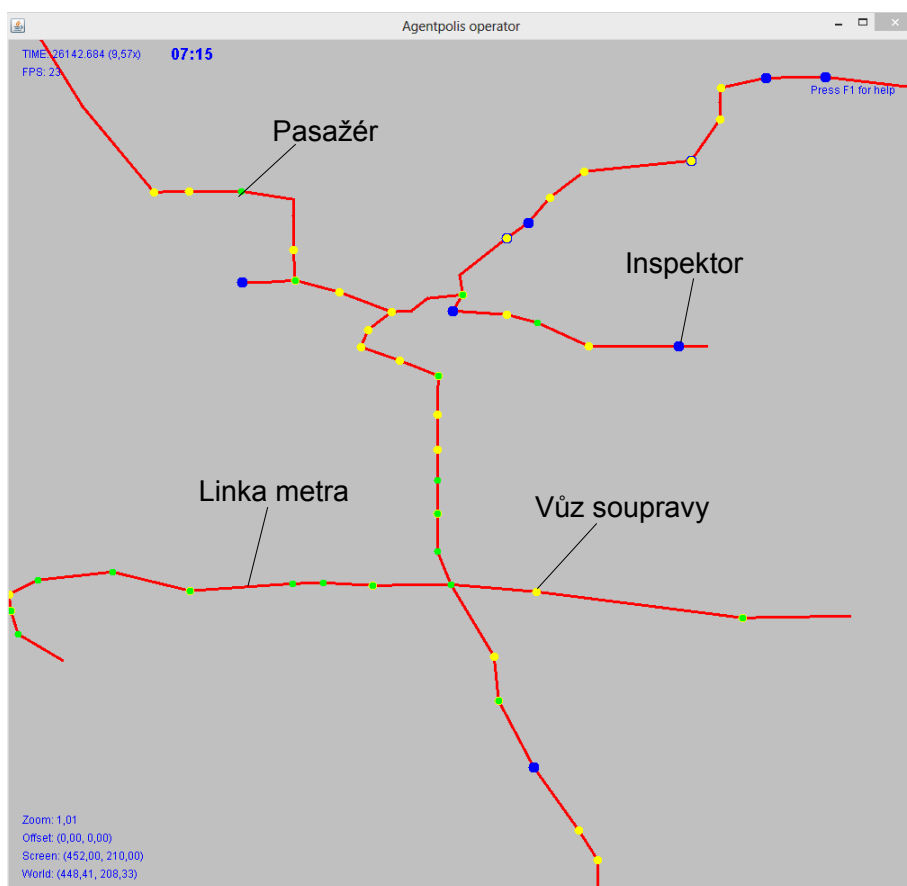
Pasažér

pasažérovi je samozřejmě umožněno vstoupit a vystoupit do stanice, následně cestovat vozem metra a má přehled o čase a své poloze. Má navíc schopnost interakce s agenty inspektorů tak, že vidí, pokud ve stanici je/není revizor, zároveň mu vnější prostředí poskytuje informace o tom, zda je/není právě kontrolován. Jeho prostor voleb je omezený, smí se rozhodovat pouze o koupi resp. nekoupi jízdného, jeho cestování systémem však určuje nastavení prostředí.

Akce pro revizora

inspektor má samozřejmě stejné přepravní možnosti jako pasažér, navíc smí kontrolovat doklady a pokutovat „černé pasažéry“. Jeho prostor volby je nulový, své trasy má definované prostředím.

Na následujícím schématu (č. 3.1) je vidět okno simulace modulu FareEvasionLA, kde lze pozorovat výskyt všech tří typů agentů v dopravním systému metra.



Obrázek 3.1: Schéma simulačního běhu v prostředí AgentPolis-FareEvasionLA

3.1.1 Faktory simulace

V rámci každého z experimentů je možné nastavit několik faktorů, které ovlivní výsledný běh, s tím související následnou statistiku a umožní tak zkoumat reakci agentů s různými parametry.

3.1.1.1 Vlastnosti prostředí

Vlastnosti prostředí agentů se většinou nemění, ač by např. cena pokuty samozřejmě mohla. Jde nám spíše o vlastnosti agentů.

Délka simulace

udává délku simulace v milisekundách resp. ve dnech

Počáteční pravděpodobnost spatření revizora

stanovuje počáteční pravděpodobnost setkání inspektora, v situaci, kdy pasažéři ještě nebudou mít dostatečnou informaci o prostředí

Cena jízdného

určuje cenu jízdného pro vstup do metra

Cena pokuty

pokuta při neprokázání se revizorovi platným jízdním dokladem

Počet agentů

počet pasažérů a inspektorů v prostředí

3.1.1.2 Parametry pasažéra

Poměr sdílení informací

udává poměr náhodně vybraných pasažérů k celé množině ($< 0, 1 >$), kteří mezi sebou budou sdílet jednu paměť

Poměr pravidelných cest

udává poměr náhodně vybraných pasažérů k celé množině ($< 0, 1 >$), kteří budou každý den opakovat stejnou trasu

Typ paměti

1. pasažéři nebudou mít žádnou paměť
2. pasažéři budou mít paměť s uloženými místy, bez ohledu na časový úsek pozorování

3. pasažéři budou mít paměť s uloženým časem události, bez ohledu na místo
4. pasažéři budou mít plnou paměť - tedy každá událost bude mít uveden čas i stanici

3.1.1.3 Parametry revizora

Poměr inspektorů na linkách (vs. stanicích)

udává poměr náhodně vybraných inspektorů ($< 0, 1 >$), kteří budou jízdenky kontrolovat na náhodně vybrané lince ve vozech metra

Poměr pravidelných cest

udává poměr náhodně vybraných inspektorů ($< 0, 1 >$), kteří budou každý den opakovat stejnou trasu (nebo stanici, pokud inspektor nejedí na lince, ale kontroluje ve stanici)

3.2 Pasažér

Základní model pasažéra spočívá v tom, že tento agent má dostupnou množinu akcí A:

$$A \equiv \{vstupDoStanice, vystupZeStanice, cestujMetrem, kupJizdne, zapamatujSiInspektory\} \quad (3.1)$$

, kde nejdůležitější jsou pro naše účely samozřejmě akce *kupJizdenku* a *zapamatujSiInspektory*.

Agent má také množinu pozorování P:

$$P \equiv \{cas, poloha, pamet\} \quad (3.2)$$

Mezi jednotlivými agenty existuje také určitá míra komunikace, a to přes sdílenou paměť pro určité procento náhodně vybraných agentů. Této skupině je následně poskytnuta identická matice paměti a pasažéři jí vyplňují společně stejným způsobem, jakoby měli každý svou vlastní.

3.2.1 Paměť

Pasažérova paměť je strukturovaná nejprve podle času a následně podle místa pozorování. Pasažér interaguje s pamětí následujícím způsobem (viz obr. č. 3.2). Agent tedy zareaguje



Obrázek 3.2: Schéma přístupu k paměti pasažérem

na každou akci, kterou mu poskytne prostředí, jako jsou: a) žádný inspektor ve stanici; b) inspektor ve stanici; c) zkontrolován inspektorem a uloží je do paměti. Tyto akce se dělí do dvou skupin podle toho, jestli pasažér revizora viděl či ne. Nezáleží však na tom, zda inspektor kontroloval právě jeho. Na základě těchto pozorování si pasažér jen vybere příslušný čas, který je diskretizován hodinou dne a pod příslušným polem stanice aktualizuje čítač podle již zmíněného druhu akce (viz obr.č. 4.4). Paměť se ovšem může ještě agregovat, a to v závislosti na zvoleném faktoru simulace (viz 3.1.1.2). O agregaci se stará vnitřní funkce třídy, která implementuje paměť.

Pokud však chceme paměť trochu teoreticky formulovat, je vhodnější přistupovat k paměti ne jako ke dvěma zřetězeným polím, nýbrž jako ke dvěma nezávislým maticím. Obě matice budou mít stejné rozměry, každá však bude nositelem jednoho typu informace, a to buď čítače pozorování akce „revizor“, nebo „žádný revizor“.

Nechť tedy existuje matice $P0$ a matice $P1$. Pro přehlednost budeme předpokládat, že matice $P1$ bude nositelem čítačů pozorování typu „revizor“. Obě matice definujeme formálně jako:

$$P0 = (p0_{s,t}), s \in \{1, 2, \dots, x\}, t \in \{1, 2, \dots, y\}; \quad (3.3)$$

$$P1 = (p1_{s,t}), s \in \{1, 2, \dots, x\}, t \in \{1, 2, \dots, y\}. \quad (3.4)$$

kde:

- $s \in \mathbb{N}$: je identifikátor stanice metra
- $t \in \mathbb{N}$: je identifikátor hodiny dne
- $p0_{s,t} | p1_{s,t} \in \mathbb{N}$: je s-t-tý prvek matice $P0/P1$

Procházení paměti pak jednoduše probíhá přes výběr pomocí souřadnice času a místa. Pokud naopak pasažér chce uložit informaci do paměti, vybere si příslušnou matici, příslušný prvek a k němu přičte číslo 1. Na formální rovině je také vhodné si definovat agregaci paměti podle času a místa, což je jeden z parametrů experimentů.

Při agregaci paměti podle místa (tj. pasažér si pamatuje pouze čas) mají hodnoty obou matic tento vzorec:

$$p0_{s,t} = \sum_{i=1}^m (p0_{i,t}); \quad (3.5)$$

$$p1_{s,t} = \sum_{i=1}^m (p1_{i,t}). \quad (3.6)$$

Agregace podle hodin (tj. paměť si pamatuje pouze místo) pak probíhá obdobně:

$$p0_{s,t} = \sum_{i=1}^{24} (p0_{s,i}); \quad (3.7)$$

$$p1_{s,t} = \sum_{i=1}^{24} (p1_{s,i}). \quad (3.8)$$

- $s \in \mathbb{N}$: je počet stanic metra
- $t \in \mathbb{N}$: je identifikátor hodiny dne
- $p0_{s,t}/p1_{s,t} \in \mathbb{N}$: je s-t-tý prvek matice P0/P1

3.2.2 Rozhodovací logika

Pasažér má vcelku úzký prostor rozhodování - dvakrát denně (viz schéma 3.3) při každém vstupu do stanice metra se rozhoduje, zda si koupí jízdenku či ne. K tomuto rozhodování mu dopomáhá paměť, která může být buď sdílená s ostatními pasažéry nebo pouze pasažérova. Jak již bylo popisováno výše, paměť může být různě omezena, nicméně operace nad pamětí, určené k rozhodování, zůstávají stejné. Pro pasažéra je důležitá jedna veličina, a to pravděpodobnost výskytu revizora na jeho plánované cestě, která se počítá nejprve pro každou stanicí zvlášť. Pravděpodobnost jedné stanice se počítá na základě četností toho, kolikrát pasažér na stanici/v čase/na stanici i v čase (záleží na typu paměti) zaznamenal revizorů. Je jasné, že zpočátku nemá pasažér příliš informací o daných místech, proto, pokud se v jeho paměti místo či čas nevyskytuje, je v prostředí zdefinovaná tzv. počáteční pravděpodobnost. Ta se využívá i v případě, že pasažér žádnou paměť nedisponuje.

Tedy shrnuto:

$$\alpha_s = \frac{inspect}{inspect + noinspect} \quad (3.9)$$

kde:

- $\alpha_s \in <0, 1>$: je pravděpodobnost kontroly revizorem v jednotlivé stanici s na trase
- $noinspect \in \mathbb{N}$: četnost pozorování stanice/času bez inspektora $\equiv$ hodnota prvku $p0_{s,t}$ v matici $P_0(s, t)$
- $inspect \in \mathbb{N}$: četnost pozorování stanice/času s inspektorem $\equiv$ hodnota prvku $p1_{s,t}$ v matici $P_1(s, t)$
- $s \in \mathbb{N}$: je identifikátor stanice metra
- $t \in \mathbb{N}$: je identifikátor hodiny začátku trasy
- P_0 : je matice paměti pasažéra s hodnotami čítače pozorování žádných revizorů v čase t a místě s (viz 3.3)
- P_1 : je matice paměti pasažéra čítače pozorování revizorů v čase t a místě s

Pokud je stanovená pravděpodobnost jedné stanice, tak se pomocí standardního pravděpodobnostního vzorce dopočítá pravděpodobnost výsledné cesty.

$$\pi = 1 - \prod_{i=1}^m (1 - \alpha_s) \quad (3.10)$$

kde:

- π : je celková pravděpodobnost kontroly revizorem na trase
- α_s : je pravděpodobnost kontroly revizorem v jednotlivé stanici s na trase
- m : je počet stanic v plánované pasažérově trase

Rozhodovací logika pasažéra, jak již bylo řečeno, bude vycházet z modelu prof. Boyda [3]. Budeme se tedy bavit o maximalizaci užitku pro pasažéra, který si jízdenku koupí za podmínky, že užitek z tohoto nákupu je větší než kdyby si jízdenku nekoupil. Pokud tento užitek

klesne, vyplatí se jízdenku si nekupovat. Cestující se tedy rozhodne koupit jízdenku, pokud bude splněná tato nerovnost:

$$U(SJizdenkou) > U(BezJizdenky) \quad (3.11)$$

, kde funkce užitku je nadefinována takto

$$U(A) = P(A)c(A) \quad (3.12)$$

, kde

- $P()$ je pravděpodobnost jevu
- $c()$ je cena tohoto jevu pro pasažéra

a tedy:

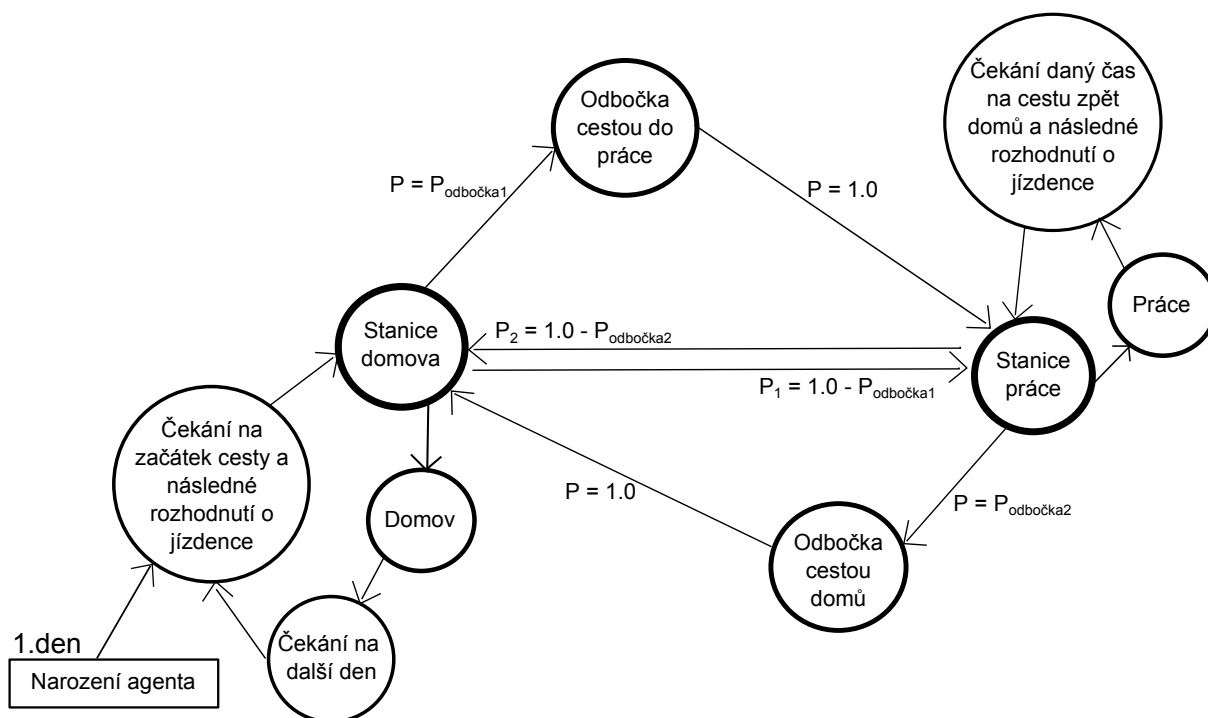
$$P(\Omega)f > \pi p \quad (3.13)$$

, kde

- $U()$ je funkce užitku, kterou chce pasažér maximalizovat - větší hodnota určuje rozhodnutí pasažéra
- f : cena jízdenky
- p : cena pokuty
- $P(\Omega)$: pravděpodobnost jistého jevu (pasažér vždy za jízdné zaplatí)
- π : pravděpodobnost kontroly revizorem

3.2.3 Denní rozvrh

Pro tuto práci jsem do modulu FareEvasionLA implementoval trochu jiné schéma denních cest pasažérů. Cílem této práce je totiž zkoumat různé trendy a tendence interakce mezi pasažéry na základě různých parametrů. Proto je nutné, aby tyto experimenty nebyly nijak zatíženy různorodě rozvrženými daty. Budeme tedy počítat s náhodným generátorem s rovnoměrným rozložením, abychom pracovali s co nejobecnějším scénářem. Původně trasy cestujících vycházely z reálných dat pozorovaných v Los Angeles, pro účely této práce jsem upravil denní rozvrh pasažéra podle schématu č. 3.3.



Obrázek 3.3: Schéma pasažérova denního cyklu

Domov, *Práce* jsou náhodně vybrané stanice z množiny stanic prostředí, které se na základě parametru počtu pravidelných cestujících mohou denně měnit nebo zůstávají po celou dobu simulace stejné

Odbočka 1 a 2 jsou náhodně vybrané stanice z množiny stanic prostředí, které se samozřejmě také mohou měnit, liší se však tím, že jsou pouze určitou měrou pravděpodobné a nemusí po cestě nastat, na rozdíl od stanic *Domov* a *Práce*, které pasažér jednou za den určitě navštíví

Znamená to tedy, že každý pasažér má náhodně vybrány čtyři stanice z množiny všech stanic systému metra v Los Angeles. První dvě představují pravidelnou cestu, která má za úkol simulovat cestu do práce a zpět. Dvě zbývající zastupují odbočky mimo hlavní trasu (návštěva lékaře, nákup potravin...) a každý den vstupují do hlavní trasy s určitou pravděpodobností.

3.3 Inspektor

V následující kapitole nebudeme již popisovat model tak podrobně jako v té předchozí. Inspektor má v simulaci vše přidělené, o ničem se nerozhoduje, ani si nic nepamatuje. Pohybuje se systémem pouze po náhodně vygenerovaných cestách.

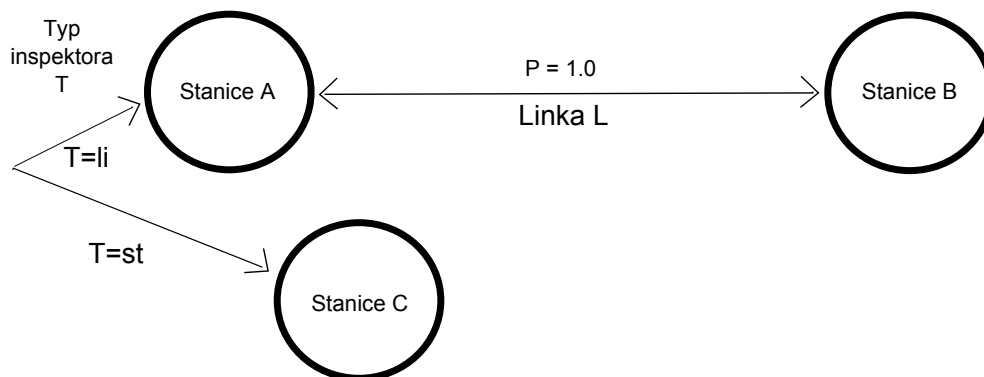
3.3.1 Rozhodovací logika

Jak již bylo zmíněno, inspektor na rozdíl od pasažéra nedisponuje žádnou rozhodovací logikou. Naším cílem bylo převážně implementovat adaptaci na prostředí pouze u pasažéra. Revizor se tedy o ničem nerozhoduje, dostane přidělen svůj denní rozvrh a tam podle něho kontroluje pasažéry a jejich jízdní doklady. Má samozřejmě také množinu akcí (viz 3.14), která je podobná pasažérově, z nichž nejdůležitější jsou akce kontroly jízdného. Oproti pasažérovi však postrádá prostor volby.

$$A \equiv \{vstupDoStanice, vystupZeStanice, cestujMetrem, kontrolujVeVoze, kontrolujVeStanici\} \quad (3.14)$$

3.3.2 Denní rozvrh

Podobně jako pasažéři, i inspektoři mají znáhodněný rozvrh cest. Ty se dělí na dvě skupiny podle typu revizora, což je jeden z faktorů simulací. Jedna skupina revizorů provádí kontrolu pouze na náhodně vybraných stanicích z celé množiny, druhá dostane přidělenou náhodně jednu linku metra a na té jezdí neustále mezi konečnými stanicemi a kontrolují pasažéry ve vagonech po určitý, předem definovaný čas (pracovní směnu).



Obrázek 3.4: Schéma denního rozvrhu inspektora

Stanice A,B jsou koncové stanice náhodně vybrané linky metra L , která se na základě parametru počtu pravidelných rozvrhů inspektorů mohou denně měnit

Stanice C je náhodně vybraná stanice z množiny všech stanic na systému metra - ta se samozřejmě opět může denně měnit

Typ T je typ inspektora, jež jej předurčuje buď ke kontrole pouze na jedné stanici metra ($T = st$), nebo ve vozech na jedné lince ($T = li$)

Kapitola 4

Implementace

V následující kapitole se budeme zevrubněji věnovat samotné implementaci teoretických modelů v rámci modulu FareEvasionLA. Bude se ovšem i tak jednat o pouhý nástin způsobu implementace, není jistě nutné uvádět kompletní pracovní popis celého programování.

4.1 AgentPolis & FareEvasionLA

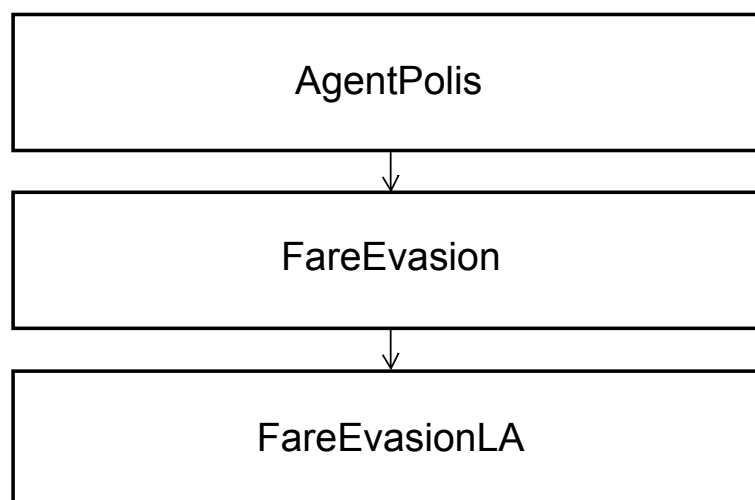
Modul FareEvasionLA je vystavěn na platformě AgentPolis a FareEvasion, z níž využívá většinu důležitých tříd, popřípadě implementuje abstraktní třídy/interface (obr. č. 4.1). Hlavní spustitelnou třídou je třída s názvem *FareevansionTicketBasedTicketHintPatrolMain*, která se stará o načítání většiny informací z konfiguračních souborů a spouští inicializační třídy obou typů agentů - tedy pasažéra a inspektora. Hlavní inicializační proces obou agentů probíhá velice podobně a dal by se shrnout do jednoduchého schématu (viz. obr č. 4.2)

4.1.1 Typy nejdůležitějších tříd agentů

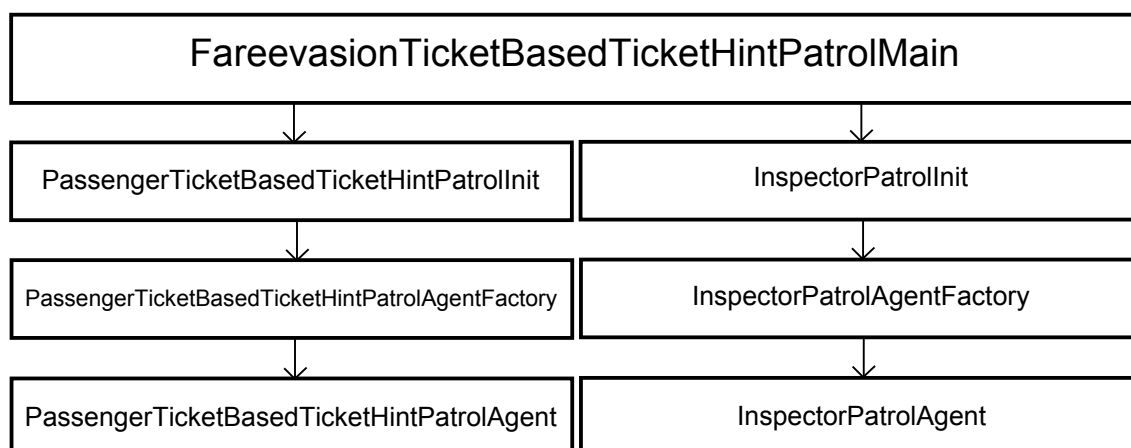
Zde si uvedeme nejdůležitější třídy z předchozího schématu (viz 4.2).

...Init

třída volaná přímo metodou *main*, která má na starosti zejména generování rozvrhů cest pro agenty



Obrázek 4.1: Závislosti modulu FareEvasionLA



Obrázek 4.2: Schéma inicializace agentů Pasažér a Inspektor ve FareEvasionLA

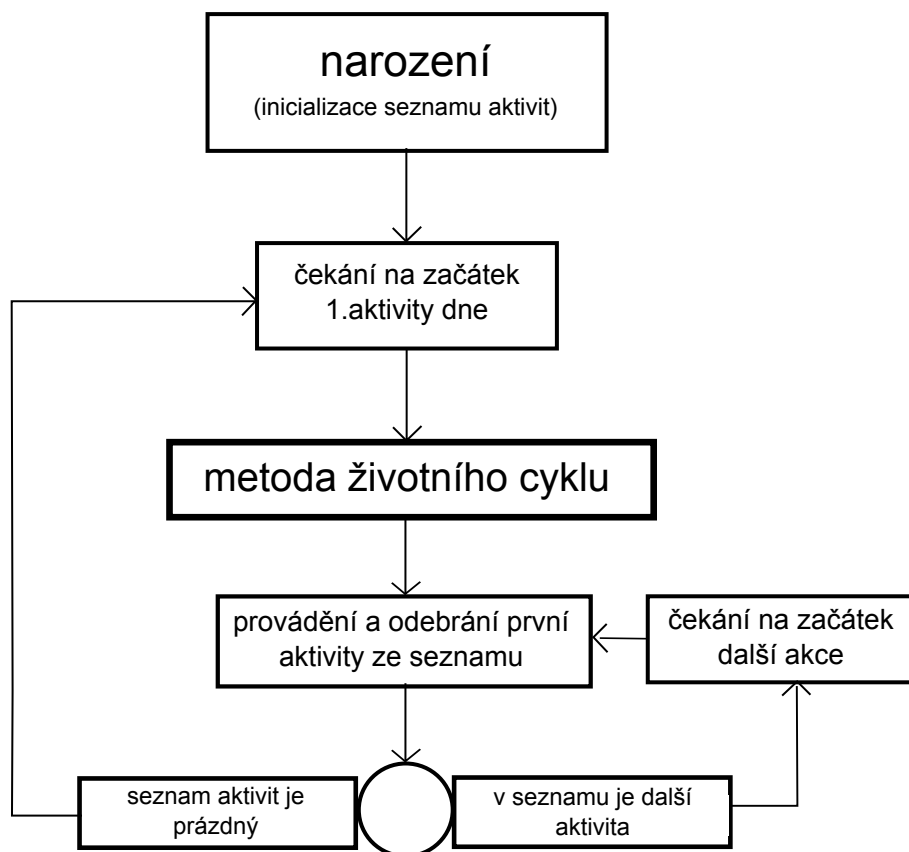
...Factory

tato třída je jakýmsi „mezistupněm“ a obstarává pomocí Google Guice Injectoru¹ závislosti z modulu AgentPolis, které předává dál agentům. Jedná se například o prostředí času, ze kterého má agent o čase přehled a může po určitý čas také čekat (např. do dalšího dne). Celá akce tak probíhá pod „kontrolou“ prostředí a nemůže dojít např. k uváznutí. Injector mimojiné obstarává plánovač cesty mezi stanicemi, ovšem v tomto případě je vyvolán o třídu výš, v třídě typu *Init*.

¹<http://google-guice.googlecode.com/git/javadoc/com/google/inject/Injector.html>

...Agent

hlavní třída upravující chování agenta. Zde se odehrává celý jeho životní cyklus a všechny reakce na podněty z prostředí. Cyklus je velmi podobný v obou případech agentů (viz obr.č.. 4.3)



Obrázek 4.3: Schéma základního životního cyklu pasažéra

4.2 Pasažér

Třídy pasažéry bylo nutné modifikovat samozřejmě nejvíce, proto se jí budeme věnovat poněkud detailněji.

4.2.1 Zavedení faktorů

Nejprve bylo nutné zajistit distribuce důležitých proměnných do třídy agenta, na základě kterých si agent upravuje své schování. Zavedení všech důležitých parametrů probíhá v obou třídách inicializace agentů, následně se předávají jako parametr.

4.2.2 Zavedení vlastního rozhodování o jízdě

První krok k řešení celého problému bylo samozřejmě odstranění přidělování pasažérům jízdenky dle daného rozložení. Informaci o tom, jestli má pasažér jízdenku, či ne, nese boolean proměnná *PassengerTicketBasedTicketHintPatrolAgent.hasTicket*. Ta se nyní nastavuje pomocí metody *PassengerTicketBasedTicketHintPatrolAgent.considerChangingDecision* při „narození“ pasažéra a dále pak při každém začátku trasy ze stanic *Domov* a *Práce* (viz schéma 3.3). Metoda nastavuje proměnnou pomocí již známého vzorce - zde ve znění

```
public void considerChangingDecision() {
    double probability =
        memory.estimateProbabilityOnTrip(estimateRoute());
    if (probability * feeCost >= fareCost) {
        return true;
    } else {
        return false;
    }
}
```

Funkce nad objektem *memory* je již záležitostí další kapitoly.

4.2.3 Implementace paměti

Narozdíl od teoretického modelu jsem se rozhodl pro implementaci zvolit přístup zřetězených polí, který se zdál být z hlediska modifikace a celkové flexibility paměti výhodnější než pamět maticová. Tato úprava představovala poměrně rozsáhlé rozšíření stávajícího kódu, proto jsem se ji rozhodl přesunout do zvláštního objektu třídy *MemoryProcessor*, ve které se nacházejí

veškeré potřebné metody a funkce pro práci s pamětí pasažéra. Třída *PassengerTicketBasedTicketHintPatrolAgent* používá poté jen metody *memory.estimateProbabilityOnTrip*, kterou již známe a *memory.saveTimeAndPosition*, která se stará o ukládání jednotlivých pozorování pasažéra do paměti. Paměť je možné u pasažérů také více způsoby agregovat (viz 3.1.1.2), o to se stará funkce uvnitř třídy *MemoryProcessor*, kterou si popíšeme níže. Bylo taktéž nutné upravit třídu samotného agenta tak, aby při každém zaznamenaném pozorování definovaném třídami, implementujícími *interface PassengerState*, ukládal tuto informaci přes již zmíněnou metodu třídy *MemoryProcessor*.

4.2.3.1 Datová struktura paměti

Struktura zcela odpovídá již zadanému modelu, zde se jedná o dva objekty *ArrayList* (kvůli třídění podle času a zároveň místa), kde ve druhém *ArrayListu* se již nachází pole o velikost 1x2, kde v prvním poli na pozici 0 se nachází počet návštěv stanice bez revizora, na druhé pozici pak s pozorováním revizora. Díky těmto čítačům se pak snadno vypočítá pravděpodobnost jedné stanice v určený čas - takže pokud nemá pasažér dostatek dat, tak použije počáteční výchozí pravděpodobnost kontroly, o které jsme se již zmínili v kapitole "Modely agentů" (viz 3.2.2).

4.2.3.2 Funkce nad paměť

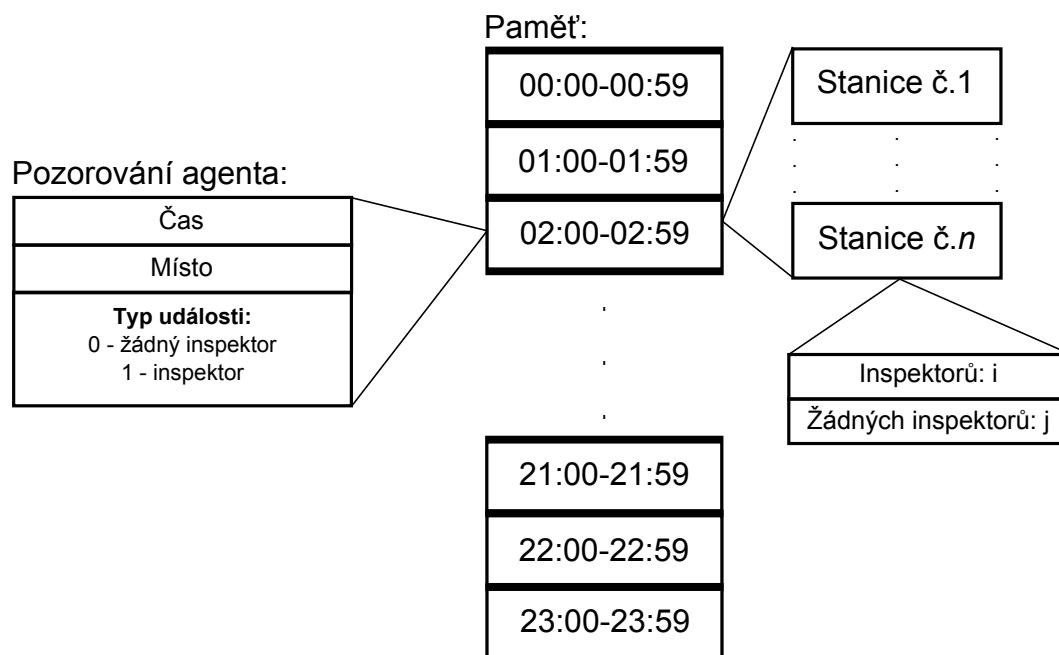
Nad pamětí byly vytvořeno mnoho pomocných funkcí, kterými se zde nebudu zabývat, jsou to většinou malé funkce na převody milisekund na hodiny, hledání id místa a jiné pomocné funkce pro práci se seznamy. Některé ostatní si zde ale krátce popíšeme:

estimateProbabilityOnStation

Již popsaná funkce, která vrací *textitdouble* s hodnotou pravděpodobnosti spatření revizora v dané stanici metra.

estimateProbabilityOnTrip

Funkce s podobným účelem, avšak ta iteruje pasažérovu trasu a pro každou stanici zavolá funkci *estimateProbabilityOnStation*. Ty si samozřejmě postupně ukládá a z



Obrázek 4.4: Schéma implementace paměti pasažéra

výsledných hodnot vypočítá následujícím způsobem (v souladu s dříve definovaným vzorcem 3.10) pravděpodobnost setkání s revizorem na trase.

```
for (int i = 1; i < trip.length; i++) {
    finalProbability = finalProbability * (1 - trip[i]);
}
finalProbability = 1 - finalProbability;
```

saveTimeAndPosition

metoda starající se o ukládání jednotlivých poznatků pasažéra - podle místa a času aktualizuje čítače revizorů resp. „nerevizorů“ v paměti

flattenMemory

Velmi důležitá funkce agregující paměť buď podle času nebo podle místa. Přijímá paměť jako takovou a vrací pouze jen jeden seznam tříděný buď podle hodin nebo podle stanic.

4.2.3.3 Sdílení paměti

Sdílení paměti se realizuje přes *boolean* parametr *isShared* a skrze přidělení sdílené paměti určité skupine pasažérů, které se realizuje v třídě s inicializací. Náhodný výběr se provádí skrze následující algoritmus.

```
List<Integer> sharedPassengers = new ArrayList<Integer>();
    double ad = passengerParameters.get("memSharingRatio")*numberOfPassengers;
    int a = (int) ad;
    for (int i = 0; i < numberOfPassengers; i++) {
        if (i<a) {
            sharedPassengers.add(1);
        }
        else {
            sharedPassengers.add(0);
        }
    }
    Collections.shuffle(sharedPassengers, generatorSharingRatio);
```

Později se při vytváření agentů jen ptáme, jestli se hodnota v seznamu nalezená pod hodnotou id aktuálně vybraného pasažéra rovná 1 - poté patří pasažér do skupiny se sdílenou pamětí a je mu později přidělena reference na objekt této paměti

```
boolean isShared = sharedPassengers.get(i)==1;
```

4.2.4 Úprava tras

Úprava tras vyžadovala trochu zevrubnější úpravu několika tříd modulu *FareEvasionLA* - především již zmíněných hlavních tříd pro inicializaci pasažéra. Stávající model totiž počítal s náhodným výběrem trasy po každém dni pasažéra. Zároveň pracoval s reálnými daty s trasami, které měli určité pravděpodobnostní rozložení a které byly uloženy v samostatném CSV souboru. Navíc bylo potřeba implementovat pozměněný model cestování pasažéra, který dosud primárně počítal pouze s jednou trasou za den - nyní bylo potřeba zavést vždy ještě cestu zpět. K novému modelu také patřily dvě „odbočky“, což jsou stanice, které pasažér s

určitou pravděpodobností navštíví cestou do práce (tak můžeme nazvat cestu tam) a zpět domů. Ale vezměme to nyní postupně.

4.2.4.1 Zavedení pasažérů s pravidelným rozvrhem

Tato modifikace spočívala v zavedení *boolean* parametru *isRegular*. Pasažéři s pravidelným rozvrhem se vybírali stejným způsobem jako pasažéři se stejnou pamětí (viz 4.2.3.3). Bylo však potřeba také upravit třídu *PassengerTicketBasedTicketHintPatrolAgent*, neboť i ta je stavěná na neustálou změnu denní trasy. Úprava však nebyla složitá, stačilo v této třídě na základě parametru *textitisRegular* ukládat prvotní trasu do globální proměnné a v metodě *doWaitToNextDayAndSetPassengerType*, což je metoda odpovídající kroku "Čekání na 1.aktivitu dne" v modelu simulačního běhu agenta (viz 3.1), podmínit změnu rozvrhu, resp. načíst a předat dále globální proměnou pravidelné trasy.

4.2.4.2 Zavedení zpátečních tras

Zpáteční trasy jsou řešeny přímo ve třídě *PassengerTicketBasedTicketHintAgent*, a to tím stylem, že pokud agent dosáhne konečné stanice na své denní trase (metoda *approachToDestinationStation*), tak se mu nastaví globální proměnná třídy *Trips trips* s otočenými stanicemi a s počátečním časem simulace. Model pasažéra totiž počítá s tím, že se pasažér v koncové stanici neotočí ihned, ale bude mimo systém cca.8 hodin (+- náhodná hodnota v intervalu $< 0; 0.5$)), a pak se po této době vydá na cestu do výchozí stanice. Poté co se nastaví nastaví *boolean* přepínač *isOnTheWayHome*, zavolá se metoda *passengerEnterToStation*, která zajišťuje základní „životní cyklus“ pasažéra a ve své podstatě se opakuje ten stejný scénář jako při první cestě (s rozdílem hodnoty jednoho *true/false* přepínače).

4.2.4.3 Zavedení pravděpodobných odboček

První věc, která byla nutná změnit, bylo přidání čtyř nových údajů ve třídě *textitPassengerTrip*, a to následující: a) id 1.odbočky; b) id 2.odbočky; c) pravděpodobnost 1.odbočky; and d) pravděpodobnost 2.odbočky. . Navíc byl zrušen údaj *arrivalTime*, který je ukázkou toho, jak rozvrhy byly koncipovány. To znamená, že existuje pro pasažéra jedna trasa, která

má počáteční a konečný čas, a to se do nového modelu samozřejmě nehodilo.

Změnit jen třídu tras samozřejmě nestačí, a tak se také muselo zasáhnout do třídy agenta.

Původní kód, který „volal“ plánovač přes již zmíněný *Injector* vypadal takto

```
long currentPosition = agentPositionSensor.getCurrentPositionById(getId());
try {trips = tripPlanner.findTrip(getId(), currentPosition, node);}
catch (TripPlannerException e) {
    e.printStackTrace();
}
doTrip();
```

V nové verzi se pomocí pravděpodobnosti s rovnoměrným rozložením testuje, zda-li má pasažér z hlavní trasy vůbec odbočovat. Pokud ano, tak je potřeba najít trasu z počáteční stanice do té nově přidané a následně trasu pokračující do koncové stanice z původní dvojice. Nyní stačí dát tyto dvě trasy za sebe do instance třídy *Trips trips* podobně jako v původní verzi. Původní implementace totiž vzhledem k přestupním stanicím umožňuje jakýsi zásobník tras (které smí být v rámci jedné linky), ze kterých pasažér postupně odebírá.

4.2.4.4 Zavedení náhodných rozvrhů tras

Zde stačilo pouze změnit hodnoty načítané z CSV souboru nahradit za náhodné hodnoty stylem:

```
stationsById.get(randomGenerator.nextInt(stationsById.size() - 1));
```

kde *randomGenerator* je instance třídy *java.util.Random* a *nextInt(n)* je generátor kladných celých čísel z intervalu $< 0, n$.

4.3 Inspektor

U inspektorů nebyly potřeba tak rozsáhlé úpravy, stačilo se zaměřit pouze na rozlišení „stanicových“ a „linkových“ typů. Ostatní úpravy jako znáhodnění a zpravidelnění tras byly prakticky totožné s předchozí sekci, a tudíž je nebudeme zmiňovat.

4.3.1 Úprava tras

Inspektoři se svým chováním na trasách dělí na dvě základní skupiny. Tyto dvě skupiny navazují na již existující implementaci frameworku FareEvasionLA, kde se návrh tras realizoval přes reálné scénáře uložených v .csv souborech. V těchto scénářích se vyskytovaly ne 2 typy revizorů, nýbrž 2 typy aktivit, s tím, že v jednom plánu se mohly vyskytovat kombinace kontroly ve vozech metra a kontroly ve stanicích. Tento model jsem tedy přenesl na revizory samotné.

revizor kontrolující ve stanicích

Revizor na každý den dostane náhodně přidělenou jednu stanicí, na které po předem definovaný čas kontroluje pasažéry a jejich jízdní doklady. Tyto stanice se mohou a nemusí měnit, záleží na parametru revizora. Implementace tohoto typu nevyžadovala příliš velké úpravy, pouze se v inicializaci (*InspectorPatrolInit*) změnilo vytváření cestovních plánů. Vytvořily se plány pouze s jednou aktivitou - a to aktivitou kontroly jízdenek ve stanicích po předem definovaný čas (v nastavení simulace). Ve třídě *InspectorPatrolAgent* pak následovaly jen drobné úpravy týkající se pravidelnosti cest.

revizor kontrolující ve vozech metra

Zde byla situace ovšem trochu složitější. Revizor tohoto typu dostane přidělenou vždy jednu linku, po kterou jezdí neustále mezi koncovými stanicemi, a to také po předem nastavený čas. Pro toto chování však implementace připravena nebyla, počítala pouze s jednou aktivitou jízdy metrem, která měla definovaný jak počáteční tak i koncový čas. Bylo tedy nutné upravit inicializaci cestovních plánů tak, aby jeden plán obsahoval počáteční a koncovou stanicí jedné náhodně vybrané linky a aby obsahoval počáteční čas. Koncový čas se dopočítal přičtením doby, po kterou mají revizori denně kontrolovat. Následovala úprava třídy samotného agenta *InspectorPatrolAgent*, kde bylo nutné zavést opakování jedné cesty, samozřejmě včetně cest zpátečních. To bylo realizováno úpravou metody *LifeCycle*, což je jakýsi ekvivalent třídy *PassengerTicketBasedTicketHintAgent.passengerEnterToStation* (viz 4.2.4.2), která se stará o hlavní životní cyklus inspektora. Zde bylo nutné přidat booleovské proměnné podobně jako u pasažéra, aby se oddělily zpáteční cesty od původních a upravilo

se tak chování podle směru cesty. Zpáteční cesty jsou řešeny velmi podobně jako u pasažérů, opět se vytváří nová cesta s opačně přiřazenými stanicemi a je nastavena místo původního plánu. Tato úprava se realizuje v metodě *finishedUsingPublicTransport()*, což je metoda, která ošetřuje konec cestovního plánu revizora. Je zde samozřejmě opět analogie s agentem pasažéra metoda *PassengerTicketBasedTicketHintAgent.approachToDestinationStation* (viz 4.2.4.2). Dále bylo nutné zavést čítač času, po který inspektor již kontoluje a který se potom mohl srovnat s předpokládaným koncem směny. To zajišťuje, že inspektor nezačne další aktivitu na lince metra, pokud již překročil stanovený stav inspekce. I zde může inspektor svou cestu pravidelně opakovat, či denně dostane přidělenou jinou.

4.4 Automatizace experimentů

Automatizace experimentů je zajišťována nově zavedenou třídou *AutomaticExperiments*, která vychází z původní třídy s „main“ metodou *FareevasionTicketBasedTicketHintPatrolMain*. Nejdůležitější změnou bylo přidání cyklu přes všechny parametry simulace a úpravy k usnadnění automatického vyhodnocení, které se provádí programem Matlab[©]². Bylo totiž nutné nějakým způsobem vždy předat parametry, s kterými se simulace spouštěla. To je nyní realizováno tak, že každý soubor s kompletním záznamem událostí celé simulace má název složený z těchto parametrů. Soubor tak vypadá takto:

```
1-seed1-numpass1000-numins10-linei0.1 -regulari0.1-regularpass0.1-sharing0.1-mtype1.0.csv
```

, kde (podobně jako v 3.1.1)

- *1* je id jednotlivého experimentu
- *seed* je id experimentu, který se liší od předchozího, pokud je k simulaci použito $n > 1$ random seedů³ a tedy n -krát více experimentů

²<http://www.mathworks.com>

³seed - číslo sloužící k zafixování stejné posloupnosti náhodných čísel pro jednu hodnotu seedu. Je nutným prvkem, pokud chceme provádět simulace s parametry, kdy simulace musí mít vždy stejný průběh. V opačném případě bychom výsledky jednotlivých simulací mohli jen obtížně srovnávat.

- *numpass* udává počet pasažérů
- *numins* udává počet revizorů
- *linei* udává počet revizorů, kteří jezdí na linkách
- *regulari* udává počet revizorů, kteří denně jezdí stejným rozvrhem
- *regularpass* udává počet pasažérů, kteří denně jezdí stejným rozvrhem
- *sharing* udává počet pasažérů, kteří sdílejí informace o revizorem mezi sebou
- *mtype* zastupuje typ paměti, kterou má stejnou celá populace agentů (1 bez paměti, 2 pamět jenom na místa, 3 paměť jen na čas, 4 kompletní paměť)

Tyto všechny parametry se dají jak definovat přímo ve zdrojovém kódu, což je výhodnější pro použití v IDE⁴, nebo přes konfigurační soubor „factors.groovy“ (v příloženém CD je používán konfigurační soubor). Vzhledem k tomu, že bylo nutné respektovat platformu AgentPolis, která vyžaduje použití konfiguračního souboru „config.groovy“, tak se při každém spuštění experimentu tento soubor pro jednotlivou simulaci vygeneruje. Abychom však mohli cokoliv iterovat, bylo nutné vytvořit metodu *experimentProcedure*, která představuje jeden běh simulace. Při experimentování s větším počtem dnů v simulaci se navíc vyskytl jeden problém - velikost jednotlivých logovacích souborů platformy AgentPolis se již při 100 denní simulaci s 1000 pasažéry a 20 inspektory pohyboval kolem 70-120 MB. To by při veliké sadě simulací mohlo způsobovat velké problémy, proto byla zavedena možnost, aby se po skončení každé simulace zkomprimovaly a přesunuly do složky „all“, kterou využívá zpracování výsledků v Matlab-u.

4.5 Zpracování výsledků

Zpracování je realizováno Matlab-skriptem *parseResults.mat*. Ovšem dříve než začneme popisovat jednotlivé implementační kroky, bude potřeba si nejprve uvědomit, podle jakých metrik budeme následně experimenty mezi sebou srovnávat. Pro účely této práce budeme

⁴IntegratedDevelopmentEnvironment - vývojové prostředí jazyka Java (Eclipse, NetBeans...)

používat metriku ve formátu, která vychází z payoff-matice v sekci 2.2. Výstupem každého experimentu bude tedy jedna nebo více takovýchto tabulek. Počet tabulek závisí na tom, jestli chceme navíc měřit závislost metriky na jednotlivých dnech resp. hodinách resp. stanicích. V tomto případě bude existovat vždy jedna tabulka pro daný den resp. hodinu resp. stanici.

###	Měl jízdenku	Neměl jízdenku
Zkontrolován	i	m
Nezkontrolován	j	n

$i, j, n, m \in \mathbb{N}$ - počty výskytů jednotlivých stavů pasažéra

Výstup zpracování máme již definovaný, zbývá dodefinovat jen vstup. Vstupní data jsou zaznamenána v souborech, kde každý řádek představuje jednu situaci s daného typu. Zpracování tedy nebude nic jiného, než agregace podle těchto typů (popřípadě i dalších veličin, viz výše). Každý řádek události je zapsán v následujícím formátu.

"čas události", "ID pasažéra", "název události", "stanice události", ..., "jízdenka ano/ne", ...

Soubor s událostmi se tedy pomocí vestavěné funkce *textscan()* převede do matice a následně se projde celá tato matice a aktualizují se čítače událostí podle názvů jednotlivých typů na základě jednoduchého schématu.

	i	j	m	n
"Checked for ticket" + Jízdenka: ano	+			
"No inspector" + Jízdenka: ano		+		
"Not checked for ticket" + Jízdenka: ano		+		
"Caught without ticket" + Jízdenka: ne			+	
"Not checked for ticket" + Jízdenka: ne				+
"No inspector" + Jízdenka: ne				+

- $i \in \mathbb{N}$ je počet zkontrolovaných pasažérů s jízdenkou
- $j \in \mathbb{N}$ je počet nezkontrolovaných pasažérů s jízdenkou

- $m \in \mathbb{N}$ je počet zkontrolovaných pasažérů bez jízdenky
- $n \in \mathbb{N}$ je počet nezkontrolovaných pasažérů bez jízdenky

Výsledkem celého zpracování je na konci .xlsx soubor s tabulkou ve formátu:

id a faktory experimentu	čítače událostí	(+popř. den, hodina, místo)
--------------------------	-----------------	-----------------------------

Tento soubor dále zpracováváme pomocí nástroje „kontingenční tabulka“ v aplikaci Microsoft Excel⁵, kde budeme v následující kapitole zkoumat vlivy jednotlivých faktorů.

⁵<http://office.microsoft.com/cs-cz/excel/>

Kapitola 5

Experimenty

V této sekci se podíváme na jednotlivé výsledky experimentů. Nebudeme se zabývat složitou analýzou všech výsledků, půjde nám jen o základní popis chování agentů v závislosti na vybraných parametrech. Tyto experimenty totiž jsou bez jakéhokoliv sofistikovanějšího hardwaru výpočetně velmi náročné - v konfiguraci s 1000 pasažéry a 100 dny simulace trvaly na mém osobním počítači vždy kolem 6-8 minut. Konfigurace mého hardwaru byla následující:

- Procesor: Intel Core 2 Duo 2.20 GHz (na obou jádrech), Paměť RAM: 2 GB

Nebudeme tedy generovat všechny kombinace faktorů¹. V každé sekci se budeme věnovat detailněji jednomu faktoru a vždy si při tom částečně zafixujeme faktory ostatní.

Nejenom experimenty jsou výpočetně náročné, také jednotlivé soubory se záznamy událostí narůstají s přibývajícím počtem pasažérů opravdu enormně. To zatěžuje následné vyhodnocení programem Matlab. Počet událostí roste totiž přímo úměrně s počtem pasažérů, a tedy velikost souboru se záznamem roste lineárně. Pokud při 1000 pasažérech má nyní záznam 100 dní (přibližně 100 MB), log pro 10 000 pasažérů bude mít 1 GB. Samozřejmě lze docílit poměrně značné úspory velikosti souborů tím, že by se zredukovala informace na jednom řádku, nicméně o samotné logování během se stará platforma AgentPolis, kterou nebylo možné upravit. Budeme se tedy většinou v následujících kapitolách omezovat menším počtem agentů, neboť i v tomto počtu je možné omezeně ukázat některé fenomény a vyhodnocení je nesrovnatelně flexibilnější.

¹což by již při 4 hodnotách 5 faktorů představovalo cca. 1024 experimentů

5.1 Uvažované parametry

Mezi parametry které budeme detailněji zkoumat, patří:

- Typ paměti
- Míra sdílení informace o kontrolách
- Poměr typů cest revizorů

Výše ceny jízdného bude po celou dobu ponechána na hodnotě 13, výše pokuty na hodnotě 1000. Hodnotu počáteční pravděpodobnosti kontroly při nulové informaci o stanici zafixujeme na hodnotě 2%, která je popravdě nadnesená. Je ovšem směrodatná pouze pro pasažéry s nulovou pamětí, ostatní budou tuto hodnotu používat pouze zpočátku, dále si již doplní svá data.

5.2 Metriky

V rámci experimentů bude hlavní veličinu představovat vždy jedna z metrik, které jsme si již definovali zde [4.5](#). Pro shrnutí si je ještě jednou uvedeme:

- Pasažér byl zkontrolován, neměl však jízdenku
- Pasažér byl zkontrolován a měl jízdenku
- Pasažér nebyl zkontrolován a neměl jízdenku
- Pasažér nebyl zkontrolován a měl jízdenku

5.3 Jednotlivé experimenty

V této sekci si uvedeme základní popis experimentů a jejich vyhodnocení.

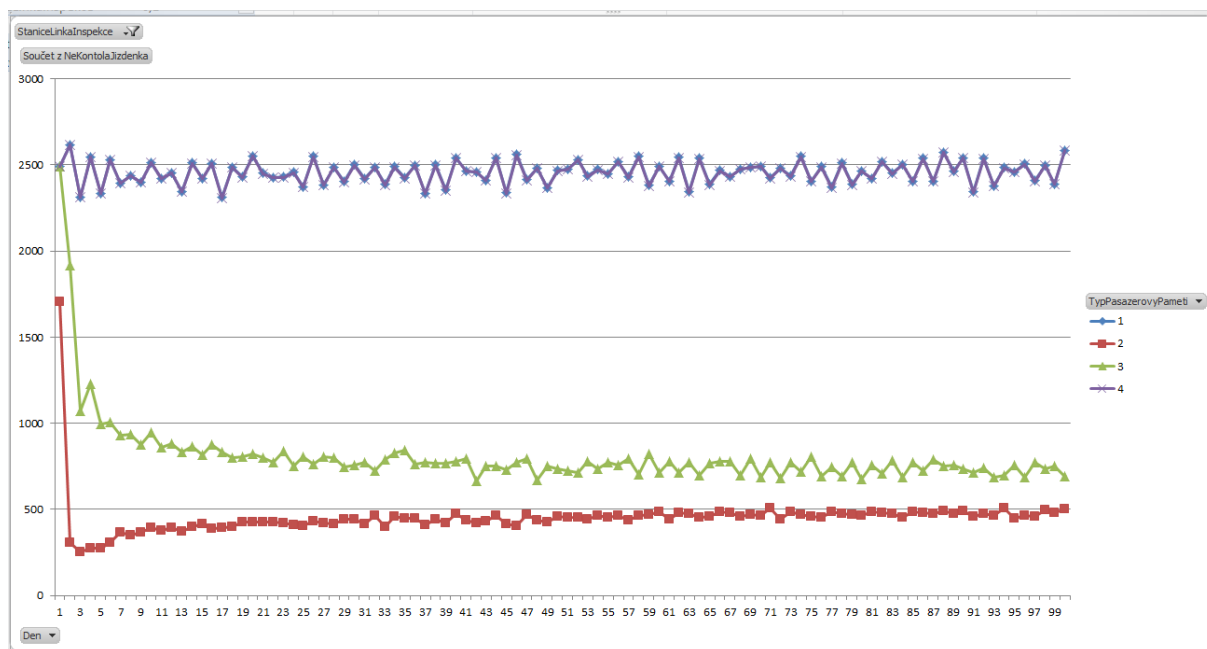
5.3.1 Srovnání typů pamětí pasažéra

Zde se budeme zabývat vlivu paměti na rozhodování pasažéra, což je v podstatě nejdůležitější hodnota. Na typu paměti závisí pasažérovo rozhodování asi nejvíce, neboť pokud paměť nebude mít paměť žádnou, bude se rozhodovat jenom v závislosti na definované počáteční pravděpodobnosti kontroly revizorem. Pokud disponuje kompletním typem paměti, dokáže si pamatovat jednotlivá pozorování podle času i místa a dokáže tak usoudit, zda je pro něj výhodnější si jízdenku koupit, či ne. Jednotlivé experimenty běžely s následujícími parametry (celkem se jednalo o 12 experimentů):

- počet dní: 100
- počet pasažérů: 500
- počet inspektorů: 10
- míra kontroly jízdného ve vozech metra: {0.1, 0.5, 0.9}
- pravidelné rozvrhy inspektorů: {0.9}
- míra sdílení informací: {0.9}
- pravidelné rozvrhy pasažérů: {1.0}
- typ paměti: {1, 2, 3, 4}

Ukážeme si nyní grafy se srovnáním jednotlivých typů paměti. Jako metriku jsem vybral například „nekontrolován, s jízdenkou“, která by měla v průběhu času klesat nebo se alespoň ustalovat. Jednotlivá čísla představují tyto typy pamětí:

- 1 - žádná paměť
- 2 - pouze paměť podle stanic
- 3 - pouze paměť podle hodin
- 4 - kombinovaná paměť podle míst a hodin



Obrázek 5.1: Graf hodnot metriky „nekontrolován, s jízdenkou“ pro všechny typy paměti pasažéra v průběhu dní

V grafu (5.1) je velmi nápadná jedna poměrně paradoxní situace. Ačkoliv by se pasažér s „plnou pamětí“ měl rozhodovat co nejoptimálněji, zde tomu tak není. Chová se totiž téměř identicky jako pasažér s pamětí žádnou. Po dlouhém zkoumání implementace a hledání jisté chyby v kódu, jsem musel uznat, že pasažér se opravdu takto chová, a že není na vině implementace, nýbrž samotný datový návrh paměti. Ten počítá s velmi striktní diskretizací časového prostoru a pokud pasažér uvažuje nad jízdou v určitém čase, která je na rozhraní dvou hodin, tak se výsledky mohou velmi razantně měnit. Navíc pasažér nemá ani po 100 dnech tolik dat, aby jimi mohl rozumně naplnit svou paměť (která může obsahovat až 2160 položek). Bohužel jsem však na toto úskalí přišel již velmi pozdě, a tak již nebyl prostor model paměti upravit. Nicméně i přesto můžeme vidět adaptaci pasažéra s pamětí podle místa a paměti podle hodiny, která se ustálila v poměrně stabilním intervalu.

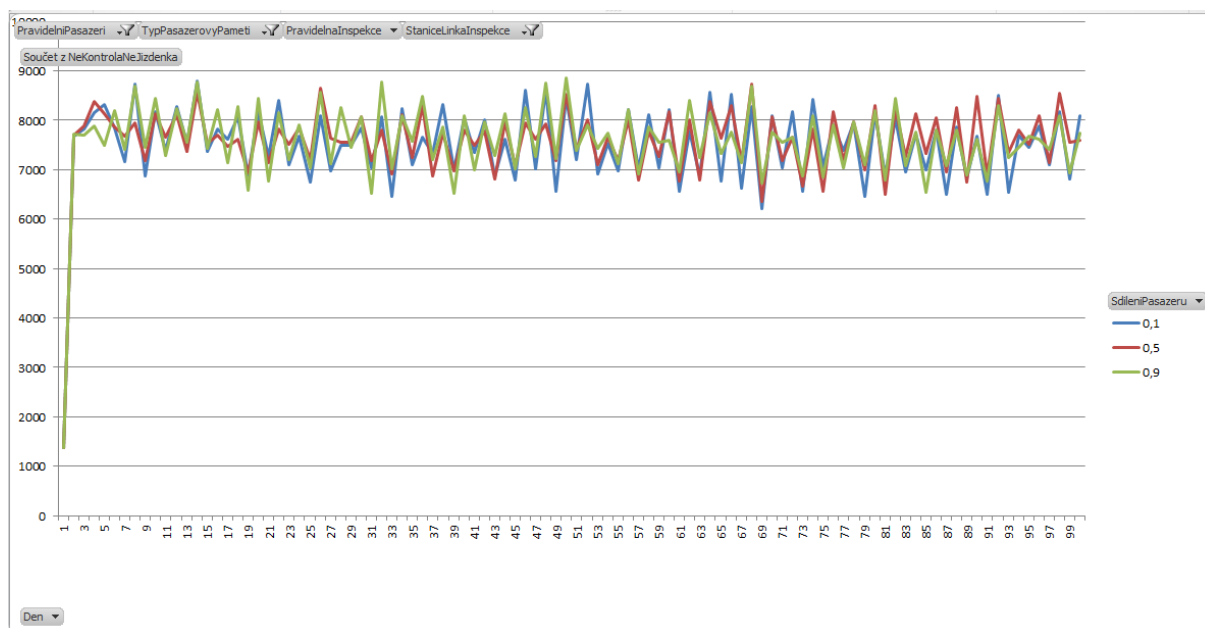
5.3.2 Sdílení informací mezi pasažéry

Faktor sdílení informací je velmi důležitý, neboť v době multifunkčních mobilních zařízení roste tendence sdílet informace o polohách revizorů pomocí internetu². V následující sekci se alespoň částečně podíváme na to, jestli má sdílení nějaký vliv na naši simulaci. Jednotlivé experimenty běžely s následujícími parametry (celkem se jednalo o 18 experimentů):

- počet dní: 100
- počet pasažérů: 1000
- počet inspektorů: 20
- míra kontroly jízdného ve vozech metra: 0.7
- pravidelné rozvrhy inspektorů: {0.1, 0.9}
- míra sdílení informací: {0.1, 0.5, 0.9}
- pravidelné rozvrhy pasažérů: {0.1, 0.9}
- typ paměti: {2, 3, 4}

Vzhledem k výsledkům „plné paměti“ budeme srovnávat pouze součet paměti stanic a hodin. Nejprve si zobrazíme graf, který zobrazuje vývoj hodnot jedné metriky „nekontrolován, bez jízdenky“ (ostatní jsou velmi podobné). Byly zároveň vybrány experimenty, kde se vyskytovali pasažéři s mírou pravidelných cest 0.9. Vidíme, že se výsledky téměř překrývají - pasažéři mají totiž poměrně rozsáhlou databázi pozorování (neboť jezdí pravidelně stejnou cestou), sdílení informací jim rozhodnutí již příliš neovlivní. Podíváme se tedy na pasažéry nepravidelné. Jak je vidět, změna není příliš markantní. Je to dáno také tím, že pasažéři jezdí podle náhodného generátoru s rovnoměrným rozložením (což ignoruje jednotlivé špičky provozu a nejfrekventovanější stanice), tudíž pro ostatní pasažéry nejsou pozorování ostatních pasažérů příliš přínosné, neboť prostor databáze je stále ještě velmi velký. Z předchozích grafů tedy nemůžeme vyvodit téměř žádné závěry. Zkusíme si tedy ještě srovnat průměrnou hodnotu všech čtyř parametrů pro míru sdílení 0,1 a 0,9 (pro pravidelné i nepravidelné cesty pasažérů), jestli nedostaneme objektivnější výsledek.

²<http://www.farebandit.net/>

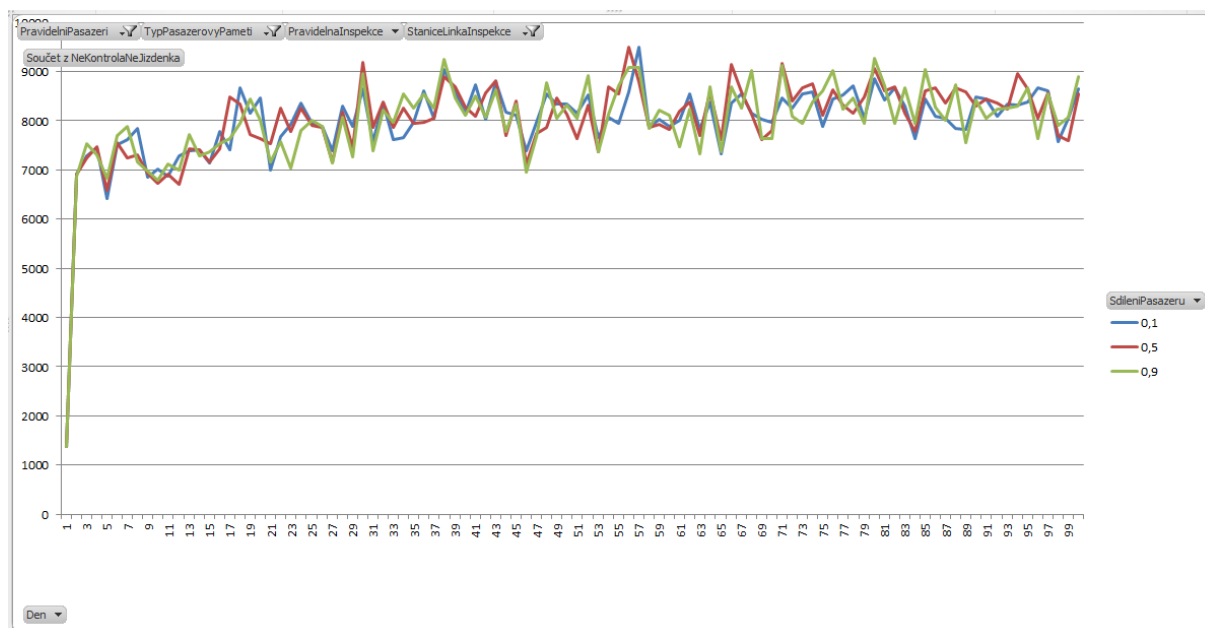


Obrázek 5.2: Graf hodnot metriky „nekontrolován, bez jízdenky“ pro tři míry sdílení informací v průběhu dní (pasažéři s pravidelným rozvrhem)

Sdílení=0.9	Měl jízdenku	Neměl jízdenku
Zkontrolován	2248,6	1148,0
Nezkontrolován	5509,3	3391,3

Sdílení=0.1	Měl jízdenku	Neměl jízdenku
Zkontrolován	2179,7	1186,9
Nezkontrolován	5262,8	3584,4

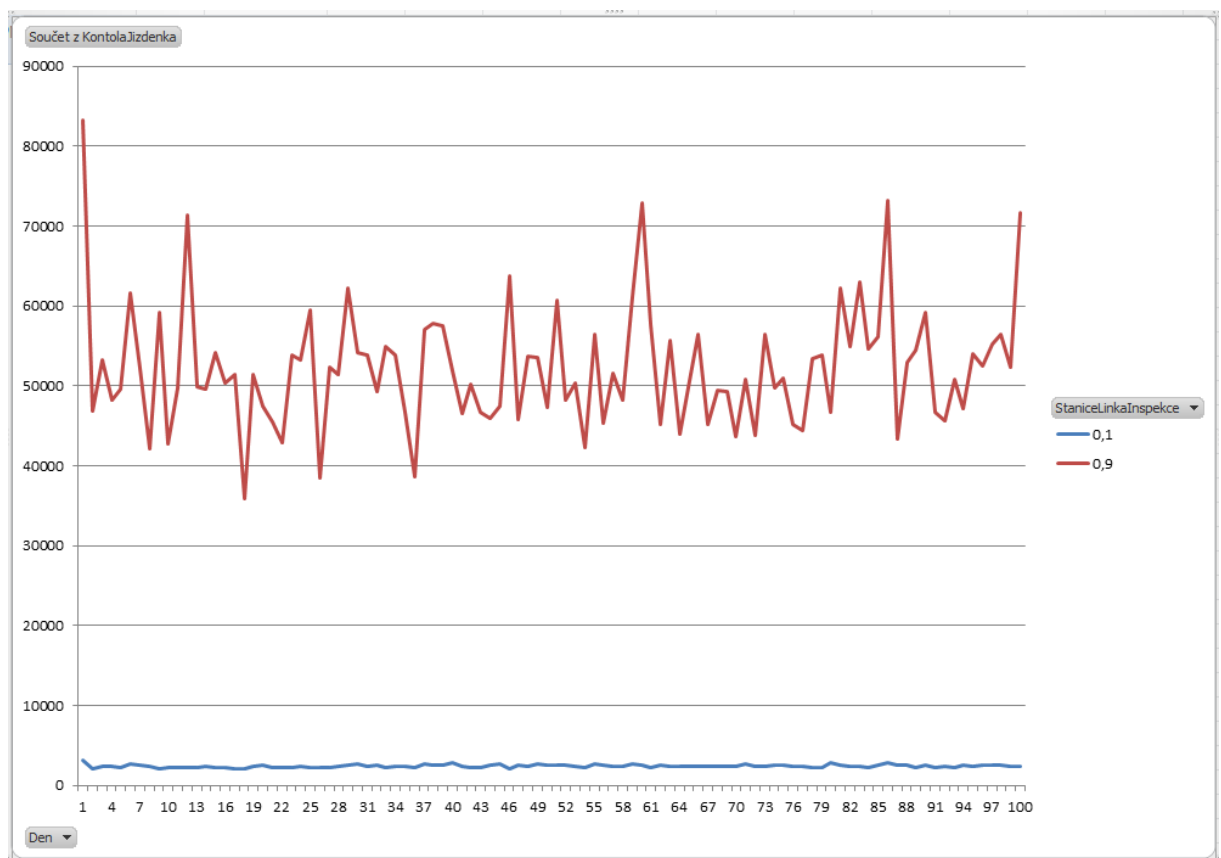
Vidíme, že výsledky se téměř shodují. Pasažér se sdílenou pamětí více inklinuje ke koupi jízdenky - má samozřejmě informaci o více revizorech v systému, nicméně výsledek k posouzení výhodností té či oné taktiky bohužel nestačí.



Obrázek 5.3: Graf hodnot metriky „nekontrolován, bez jízdenky“ pro tři míry sdílení informací v průběhu dní (pasažéři s pravidelným rozvrhem)

5.3.3 Srovnání efektivity různých typů revizorů

V poslední části si ještě krátce srovnáme působení revizora kontrolujícího pouze ve stanicích a revizora, který kontroluje ve vozech metra. Z následujícího obrázku 5.4 vidíme jasně, že největší množství kontrol provádějí inspektoři právě přímo ve vagonech metra. Je to dáno tím, že cestující jezdí poměrně dlouhé trasy a pravděpodobnost, že potkají revizora ve stanici je daleko menší. Ve vagonu metra totiž stráví daleko více času než v jedné stanici metra, kterou navštíví 2x denně.



Obrázek 5.4: Graf hodnot metriky „kontrolován, s jízdenkou“ pro dva typy revizorských kontrol.

Kapitola 6

Závěr

Tato závěrečná práce pro mne byla velikým přínosem z několika hledisek. Zaprvé mi bylo umožněno pracovat na tématu, které je opravdu zajímavé a které není založenou pouze na teorii. Navíc na takovém tématu, s kterým se běžný člověk má možnost setkávat ve svém životě takřka denně, tudíž si velmi dobře dokáže představit jeho problematiku. Byl to ale také veliký trénink z hlediska formě četby a porozumění cizího kódu, na který je potřeba navázat, popřípadě upravit tak, aby byl funkční jak v původním, tak i v novém prostředí. Velmi cenná byla ovšem příležitost zjistit, jak funguje v praxi agentní modelování jako takové, a jakým způsobem se implementuje.

Novinkou pro mne byla práce s třemi různými aplikacemi najednou, které bylo potřeba sladit tak, aby fungovaly jako celek. K tomu bylo nutné zajištění integrity vstupů a výstupu mezi jednotlivými programy tak, aby nedošlo k žádné ztrátě dat, a aby byly výsledky pro jinou aplikaci kompatibilní.

Z hlediska vylepšení práce se jistě asi není potřeba příliš zamýšlet. Bylo by samozřejmě nutné zvážit jiný teoretický model založený na náhodných generátorech se sofistikovanějším rozložením než je rovnoměrné (např. normálním). Evidentně se pasažéři chovají velmi nepředvídatelně a z jejich chování se nedá vyvodit příliš relevantní závěr. Dále by bylo samozřejmě nutné zpracovat na datovém modelu „plné paměti“, neboť i ten se ukázal jako nevyhovující a příliš striktní vzhledem k hodinám.

I přes tato negativa tato práce nabízí výrazné zjednodušení opakování jednotlivých experimentů s proměnlivými faktory. Zavádí implementaci zpracování výsledků v programu Matlab

a přípravu těchto výsledků do formátu .xlsx, kde se pohodlně dají kreslit různé závislosti faktorů (což obstarává aplikace Microsoft Excel). Symbióza těchto tří aplikací tak vytváří zajímavý přístup k zautomatizování samotné simulace a následného zpracování dat z jejích výsledků.

Jak jsem již však v úvodní kapitole zmínil, tato práce si neklade za cíl, jakkoliv optimalizovat chování pasažérů či revizorů, které by bylo použitelné pro zefektivnění přepravní kontroly v reálných systémech. Jejím hlavním cílem bylo teoreticky namodelovat agenta v systému veřejné dopravy a vyzkoušet chování populace na různých sadách experimentů v praxi.

Literatura

- [1] BIJLEVELD, C. Fare dodging and the strong arm of the law. *Journal of Experimental Criminology*. 2007, 3, 2, s. 183–199.
- [2] BONABEAU, E. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*. 2002, 99, Suppl 3, s. 7280–7287.
- [3] BOYD, C. et al. Fare evasion and non-compliance: A simple model. *Journal of Transport Economics and Policy*. 1989, s. 189–197.
- [4] BURMEISTER, B. – HADDADI, A. – MATYLIS, G. Application of multi-agent systems in traffic and transportation. In *Software Engineering. IEE Proceedings-[see also Software, IEE Proceedings]*, 144, s. 51–60. IET, 1997.
- [5] CLARKE, R. V. – CONTRE, S. – PETROSSIAN, G. Deterrence and fare evasion: Results of a natural experiment. *Security Journal*. 2009, 23, 1, s. 5–17.
- [6] DPP. DP kontakt 2008, 2008. Dostupné z: <<http://www.dpp.cz/download-file/1158/01-leden-2008.pdf>>. Online; staženo 15.května 2013.
- [7] KOOREMAN, P. Fare Evasion as a Result of Expected Utility Maximisation: Some Empirical Support. *Journal of transport economics and policy*. 1993, s. 69–74.
- [8] LEUNG, L.-c. *A Study of Fare Evasion in Railway Systems in Hong Kong*. PhD thesis, University of Hong Kong, 2003.
- [9] REDDY, A. V. – KUHLS, J. – LU, A. Measuring and Controlling Subway Fare Evasion.

- Transportation Research Record: Journal of the Transportation Research Board.* 2011, 2216, 1, s. 85–99.
- [10] WOOLDRIDGE, M. Agent-based software engineering. *IEE Proceedings-Software.* 1997, 144, 1, s. 26–37.

Příloha A

Instrukce ke spuštění simulačního prostředí

Ke spuštění simulace slouží k tyto soubory:

- AutomaticExperiments.jar (více experimentů v jednom běhu)
- SimplePatrolMain.jar (experimenty "po jednom")

Soubor AutomaticExperiments.jar má složku s nastavením a daty v *experiments|exp-all|* a její nastavení probíhá ze souboru *config|factors.groovy*. Soubor SimplePatrolMain.jar má složku s nastavením a daty v *experiments|exp-simple|* a její nastavení probíhá ze souboru *config|config|scenario.groovy*. Pro spuštění souboru SimplePatrolMain.jar či AutomaticExperiments.java není nutné dodávat jakýkoliv argument, pokud však budete chtít specifikovat cestu ke složce s experimenty (která musí být stejné struktury), tak jak soubor AutoExperiments.java, tak i build SimplePatrolMain.jar přijímají dva nepovinné argumenty:

- cesta ke složce s experimentem (výchozí hodnota: experiments)
- hodnota Random seedu pro experimenty (výchozí hodnota: 9) (u SimplePatrolMain.jar představuje ID experimentu)

Ke spuštění obou buildů je zapotřebí mít správně nainstalované a nastavené prostředí Java a je možné realizovat například tímto příkazem volaným z command-line:

```
java -jar AutomaticExperiments.jar | SimplePatrolMain.jar experiments 9
```

Zpracování výsledků poté probíhá pomocí Matlab skriptu *parseResults.mat*, který je defaultně uložený tak, aby přečetl nekomprimované výsledky ze složky *results* | *all*. Standardně zpracovává výsledky s ohledem na den události, v tomto skriptu je však také připraveno zpracování s ohledem na hodinu události či bez ohledu na zmíněné veličiny (1 experiment = 1 řádek tabulky). Pokud budeme zpracovávat výstup souboru SimplePatrolMain.jar, je nutné použít verzi Matlab skriptu s odkomentovaným zpracováním souborů z původního formátu výstupu (nekomprimovaného v jednotlivých složkách experimentů). Tato verze se nachází ve složce s výsledky *experiments* | *exp-simple*.

Příloha B

Obsah příloženého CD

- AgentPolis\
 - agentpolis\
 - apgoogleeearth\
 - fareevasion\
 - **fareevasionla**\
 - ...
 - publictransport\
 - Bakalářská práce\
 - Soubory jiných zdrojů\
 - BP - David Petr.pdf
 - Spustitelné soubory\
 - Sestavené soubory\
 - (spustitelná demo simulace)
 - * experiments\
 - (obsahuje nastavení)
 - * AutomaticExperiments.jar
 - * SimplePatrolMain.jar
 - Hotové výsledky\
 - (výsledky, které byly vygenerovány a zpracovány do .xlsx)