Úvod

V dnešní době je na dálnicích velká snaha o dynamické řízení dopravy obzvláště v místech s uzavírkami jízdních pruhů či při vysoké hustotě provozu. Uzavření jízdního pruhu může nastat prakticky kdykoliv a to plánovaně (např. při pravidelné údržbě) a nebo neplánovaně (dopravní nehoda). Z toho vyplývá úkol pokusit se v tento moment řídit dopravu jak nejlépe to jde. Ani experti se ovšem neshodují, co znamená slovo nejlépe z předchozí věty. Může to být např. konkrétní hodnota intenzity či rychlosti, dojezdový čas či další parametry dopravního proudu.

Pokud jsme schopni získat surová data z daného úseku, např. intenzitu vztaženou na minutu pro každý pruh, a následně je vyhodnotíme, dostaneme přesné informace o úseku. Poté můžeme také tyto data vzít a pokusit se nasimulovat danou oblast. Otázkou zůstává jak. Ideálně tak, aby simulace byla blízká k již získaným informacím o úseku. V této práci se zaměříme na mikroskopické modely, podle kterých budeme simulace provádět. To znamená, že budeme simulovat každé jednotlivé vozidlo.

Po simulacích vyhodnotíme jednotlivé modely a zjistíme, které modely odpovídají reálným situacím. Předpoklad je takový, že některé modely budou přesnější např. při vysokých hustotách, některé zase naopak.

Ve chvíli, kdy získáme tento přehled modelů, můžeme již přesněji analyzovat danou situaci a do budoucna budeme vědět, že daný model simuluje tento stav nejlépe.

Hlavní cíle práce tedy spočívají v simulaci dálničního provozu na základě získaných reálných dat, a to pomocí různých mikroskopických modelů. Dalším krokem je porovnat výsledky z těchto simulací s reálnými daty. Tento způsob lze využít jako další možnost při řízení dané dálniční oblasti či při predikování např. plánovaných uzavírek.

Cíle práce

Cíle práce lze chronologicky formulovat následovně:

- Seznámení se se základními typy mikroskopických modelů (hlavně jejich matematickým popisem) v dopravě pro použití na dálnicích
- Seznámit se s kalibrací těchto modelů
- Seznámit se simulačním softwarem SUMO a možnostmi jeho využití v dané problematice.

Tato témata jsou náplní první, teoretické části práce. V druhé, praktické, části byly cíle práce následující.

- implementace všech mikroskopických modelů z teoretické části do simulačního softwaru SUMO
- kalibrace mikroskopických modelů z teoretické části
- nasimulovaní reálných dat, případně vytvoření vlastních scénářů pro mikroskopické modely
- Makroskopický popis výsledků simulace.
- Mikroskopický popis výsledků simulace.

1. Mikroskopické modely

Jednoduše řečeno „Car Following“ modely vycházejí z předpokladu, že pokud n-té vozidlo následuje $n-1$ vozidlo na homogenní dálnici, trajektorie n-tého vozidla bude stejná jako vozidla před ním $(n-1)$ až na posuny v čase a místě. (Vozidlo bude ve stejném místě jako předcházející v jiný čas a ve stejný čas bude na jiném místě.) Všechny další dispozice jednotlivých modelů v sobě tento předpoklad obsahují.

1.1 Newellův model

V teorii dopravního proudu je Newellův model jeden z modelů, který popisuje chování řidiče dle vozidla před ním. Řidič se snaží držet za vozidlem jedoucím před ním v konstantní vzdálenosti. Tento model vznikl v roce 1961 a postupně se vyvíjel. Vychází z něho novější modely, např. Intelligent Driver Model (IDM) a z tohoto důvodu je zde zmíněn.

V článku [12] je přesně popsán nejen model, ale i jeho verifikace a porovnání s ostatními modely. Pro nás je důležitý popis modelu.

1.1.1 Popis modelu

Jestliže n-té vozidlo jede za $n-1$ vozidlem (které jede za $n-2$ vozidlem atd.), cíl každého Car following modelu je zjistit závislost trajektorie n-tého vozidla $x_n(t)$, jeho pozici v čase t na $n-1$ vozidle, viz. obrázek 1.1. (To také znamená, že není žádná náhodná spojitost mezi těmito vozidly.) Jestliže se $n-1$ vozidlo pohybuje konstantní rychlostí v,

$$x_{n-1} = x_n + vt,$$

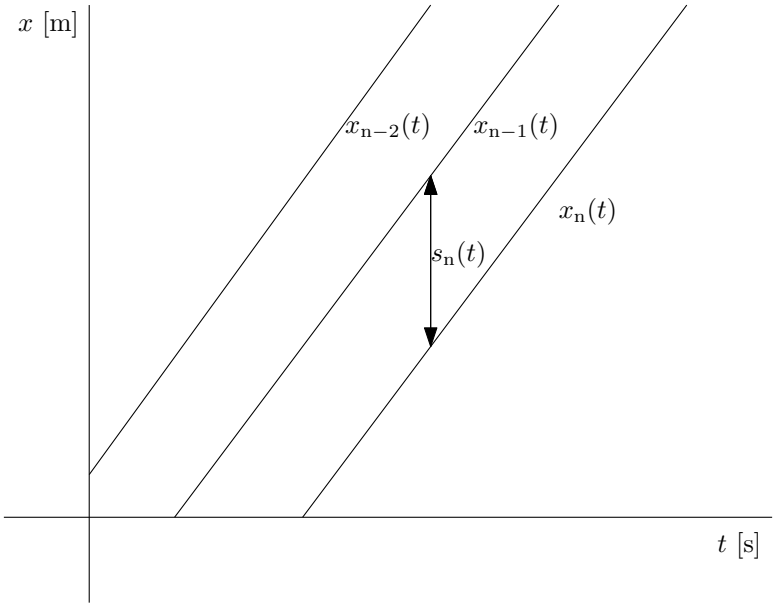
n-té vozidlo také pojede průměrnou rychlostí v. Pokud by n-té vozidlo zrychlovalo, tak by došlo ke kolizi s $n-1$ vozidlem a naopak pokud by zpomalovalo, tak by se n-té vozidlo neustále vzdalovalo. Totéž platí pro všechna vozidla jedoucí za n-tým. Tento model se nezabývá zjištěním hodnoty rychlosti v, předpokládá se, že je určena buďto na základě omezení rychlosti či možnostech vozidla.

Vzdálenost $s_n = x_{n-1}(t) - x_n(t)$ mezi vozidly n a $n-1$ se může měnit v čase. Pokud je dálnice homogenní, tato vzdálenost zůstane konstantní okolo hodnoty s_n. Tato hodnota se mění na základě typu vozidla a také závisí na rychlosti v.

Předpokládejme, že existuje nějaký empirický vztah mezi rychlostí v a vzdáleností mezi vozidly s_n. Pokud rychlost v roste, je logické, že řidiči chtějí dosáhnout většího rozestupu mezi vozidly. Tato závislost mezi v a s_n je znázorněna na obrázku 1.2. Každý řidič má svoji preferovanou rychlost V_n. Jestliže tato rychlost je u $n-1$ vozidla vyšší než u n-tého $v > \mathrm{V}_n$, znamená to, že n-té vozidlo pojede svoji preferovanou rychlostí (na obrázku 1.2 znázorněna čárkovanou čárou) a $n-1$ vozidlo mu ujede. Hodnota rychlosti v nemůže být záporná a vzdálenost mezi vozidly při nulové rychlosti by měla být níže než polopřímka lineární závislosti, viz černá tečka na obrázku 1.2 při rychlosti $v = 0$.

Nyní předpokládejme, že $n-1$ se vozidlo nějakou dobu t pohybuje konstantní rychlostí v[1] a potom náhle změní rychlost na hodnotu v'. Trajektorie vozidel n a $n-1$ mohou poté vypadat jako na obrázku 1.3. Z obrázku lze také vypozorovat jak časovou τ_n, tak prostorovou d_n mezeru

[1] hodnota rychlosti osciluje okolo hodnoty v



Obrázek 1.1: Trajektorie vozidel s konstantní rychlostí, zdroj: [12]

mezi vozidly n a $n-1$. Z čárkovaného obdelníku poté dostáváme vztah vzdálenost mezi vozidly před změnou rychlosti s_n a po změně rychlosti s'_n,

$$s_n = d_n + v\tau_n, \qquad s'_n = d_n + v'\tau_n.$$

Z toho vyplývá, že pokud leží v a v' na polopřímce z obrázku 1.2, sklon této přímky je právě τ_n a hodnota s_n při rychlosti $v = 0$ je d_n.

Ze vztahu mezi v a s_n, jak je zobrazeno na obrázku 1.2, plyne nezávislost mezer d_n a τ_n na rychlostech v, resp. v'. Pokud se tedy změní rychlost z hodnoty v' na v'', d_n a τ_n zůstanou při této změně rychlosti stejné. Lineární trajektorie vozidla $x_n(t)$ potom bude jednoduše posun v čase τ_n a místě d_n,

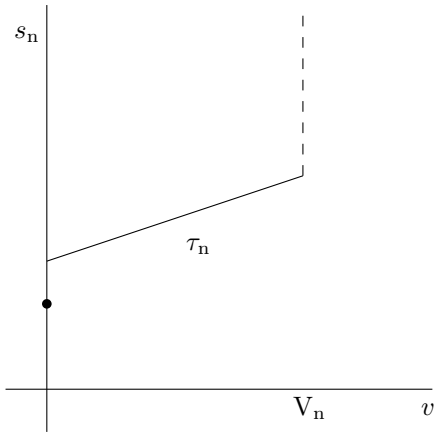
$$x_n(t + \tau_n) = x_{n-1}(t) - d_n. \tag{1.1}$$

U Newellova modelu platí, že n-té vozidlo bude (přibližně) kopírovat trajektorii n-1 vozidla dle vztahu (1.1) při vhodných hodnotách d_n a τ_n. Tím, jak se přesně dokáže n-té vozidlo dodržovat vztah (1.1), se Newellův model nezabývá, pouze předpokládá, že řidič je schopen se tímto vztahem řídit. To není tak těžké, protože pokud se změní rychlost vozidla $n-1$, n-tý řidič nemusí zareagovat okamžitě, ale může počkat dokud se mezera s_n nezvýší (neklesne) na hodnotu, která odpovídá nové rychlosti vozidla n-1 (viz obrázek 1.2).

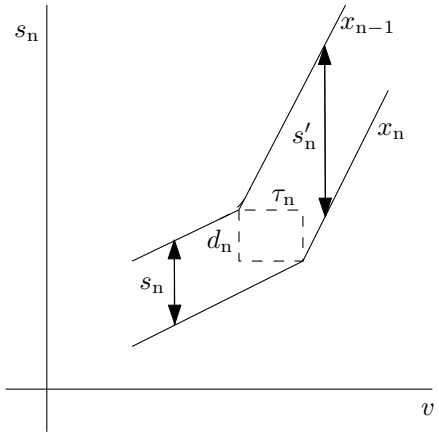
Při pozorování se došlo k závěru, že každý řidič nezkoumá každého jiného řidiče na komunikaci, ale pouze jen vhodnou „makroskopickou“ část. Hodnoty τ_n a d_n jsou přirozené, to vychází ze vztahu (1.1), kde postupným iterováním dostaneme

$$x_n(t + \tau_n + \tau_{n-1} + \cdots + \tau_1) = x_0(t) - d_n - d_{n-1} - \cdots - d_1. \tag{1.2}$$

Hodnoty τ_n a d_n se značně liší mezi jednotlivými vozidly v závislosti na typu řidiče. První skupina řidičů raději jede blíže vozidlu před sebou $(n-1)$ až na minimální bezpečnou vzdálenost,



Obrázek 1.2: Vztah mezi rychlostí v a vzdáleností mezi vozidly s_n, zdroj: [12]



Obrázek 1.3: Lineární aproximace při změně rychlosti vozidel, zdroj: [12]

zatímco druhá skupina naopak preferuje delší vzdálenost pro klidnou reakci na nenadálou událost. Je rozumné, aby se hodnoty τ_n a d_n lišily, a to tak, že každému jednotlivému vozidlu budou hodnoty vygenerovány z nějakého pravděpodobnostního rozdělení, jehož variační koeficient se bude blížit jedné. Při generování rychlostí vozidel by naopak měl být variační koeficient zanedbatelný.

Položíme-li

$$\overline{\tau} = \frac{1}{n}\sum_{k=1}^{n}\tau_k, \qquad \overline{d} = \frac{1}{n}\sum_{k=1}^{n} d_k, \tag{1.3}$$

kde $\overline{\tau}$ a $\overline{d}$ jsou průměrné posuny v čase resp. místě, potom průměrnou vlnovou rychlost spočteme jako podíl $\overline{d}/\overline{\tau}$.

Tento model se dá ovšem popsat nejen mikroskopicky (rychlost, mezera mezi vozidly), ale i makroskopicky (hustota k, intenzita q). Stacionární stav nastane ve chvíli, kdy se všechna vozidla budou pohybovat konstantní rychlostí s rozdílnými posuny (časovými i prostorovými).

Jestliže

$$s_n = d_n + v\tau_n,$$

a rychlost vozidel je konstantní, tak platí

$$\overline{s} = \overline{d} + v\overline{\tau}.$$

Hustotu k lze potom vypočítat jako převrácenou hodnotu průměrné mezery mezi vozidly $k = 1/\overline{s}$ a rychlost jako podíl intenzity a hustoty $v = q/k$. Tudíž platí

$$q = \frac{1}{\overline{\tau}} - \frac{\overline{d}}{\overline{\tau}}k, \tag{1.4}$$

za předpokladu, že rychlost v je menší než preferovaná rychlost jakéhokoliv vozidla V_k.

Rovnice (1.4) propojuje Newellův model s klasickými makroskopickými modely. Problém nastává ve chvíli, kdy průměrná rychlost v je vyssší, než některá s preferovaných rychlostí jednotlivých vozidel. Vozidla se v tomto modelu nemohou předjíždět a proto vznikne kongesce za vozidlem s malou preferovanou rychlostí. V následujících řádcích předpokládáme, že aktuální rychlost vozidla bude menší než preferovaná rychlost V_n.

Uvažujme, že se vozidlo n pohybuje přesně podle rovnice (1.1) a vozidlo za ním $(n-1)$ jede plynule za ním.

Rovnici (1.1) můžeme přepsat do tvaru

$$x_n(t + \tau_n) = x_n(t) + \tau_n v_n(t + T_n). \tag{1.5}$$

Rovnice (1.5) lze pro rovnoměrný pohyb zjednodušit na tvar

$$x_n(t + \tau_n) = x_n(t) + \tau_n v_n(t).$$

Tvar rovnice (1.5) plyne z matematické analýzy[2], přičemž hodnota T_n se nachází někde mezi nulou a τ_n. Pokud je funkce hladká bude přibližně

$$T_n = \frac{\tau_n}{2}. \tag{1.6}$$

Rovnici (1.5) lze přepsat na přibližný tvar

$$x_n(t + \tau_n) = x_n(t) + \tau_n v_n(t) + \tau_n T_n a_n(t), \tag{1.7}$$

který uvažuje zrychlení vozidla a_n[3]. Kombinací rovnic (1.7) a (1.1) dostáváme vztah pro rychlost

$$v_n(t + \tau_n) = \frac{1}{\tau_n}\left[x_{n-1}(t) - x_n(t)\right] - \frac{d_n}{\tau_n}. \tag{1.8}$$

Po zderivování rovnice (1.8) dostáváme vztah pro zrychlení vozidla

$$a_n(t + \tau_n) = \frac{1}{\tau_n}\left[v_{n-1}(t) - v_n(t)\right]. \tag{1.9}$$

[2] věta o střední hodnotě

[3] Předpokládejme vozidlo v klidu, které se rozjíždí. Na začátku zvolíme $\tau_n = 5s, t = 0s, T_n = \tau_n/2 = 2,5s$, potom platí

$$x_n(5) = x_n(0) + 5v_n(0 + 2,5).$$

Pro výpočet dráhy bereme hodnotu rychlosti v čase 2,5s. Nemůžeme vzít hodnotu v nule, vozidlo by v tomto případě stálo na místě. Přibližná hodnota bude tedy opravdu v polovině hodnoty τ_n.

Z rovnic (1.8) a (1.9) je vidět zřejmá závislost na vozidle, které jede před aktuálním, členem $v_{n-1}(t)$, resp. $x_{n-1}(t)$. Řidič volí svoji rychlost na základě odstupu od předchozího vozidla, resp. zrychlení na základě rychlostí. **Konečný vztah pro Newellův model je tedy**

$$a_n(t+T_n) = \frac{1}{\tau_n}\left[v_{n-1}(t) - v_n(t)\right] - \frac{d_n}{\tau_n} - v_n(t). \tag{1.10}$$

1.2 Gippsův model

Další model, kterým se zde budeme zabývat, se jmenuje Gippsův. Tento model byl poprvé popsán v roce 1981 a odvolává se i na zmíněný Newellův model. I v dnešní době se tento model využívá pro simulaci dopravyy. Hlavním zdrojem pro tuto podkapitolu je článek P.G. Gippse[5]. Většina modelů, které vznikly před Gippsovým, jsou různé variace na rovnici

$$a_n(t+T_n) = l_n \frac{[v_{n-1}(t) - v_n(t)]^k}{[x_{n-1}(t) - x_n(t)]^m}, \tag{1.11}$$

kde jednotlivé proměnné mají stejný význam jako při popisu Newellova modelu 1.1. Proměnné l_n, k a m jsou parametry, které je nutno empiricky odhadnout. Tyto modely se nazývají Modely General Motors a jsou blíže rozebrány v [18].

Ačkoliv podávají tyto modely dobré výsledky v mnoha situacích, je žádoucí, aby během přepočtů poloh, rychlostí a zrychlení nějakým způsobem byl zahrnut reakční čas řidiče T_n. To si vyžaduje značné množství historických dat, pokud model bude použit v simulačním programu. Navíc, parametry l_n, k a m nemají přímou souvislost s vlastnostmi vozidla či řidiče.

Gipps se proto rozhodl, že vytvoří model, který bude splňovat následujicí podmínky:

- model by měl napodobovat chování opravdového provozu,
- parametry modelu by měly odpovídat vlastnostem řidiče a vozidla, které jsou jasné; to znamená, že pro nastavení parametrů není potřeba model kalibrovat,
- model by se měl chovat správně i v případě, že doba mezi přepočítáváním polohy a rychlosti je stejná, jako reakční čas řidiče.

1.2.1 Popis modelu

Následující model je odvozen z omezení jak řidiče, tak i vozidla a pomocí těchto omezení se dopočítávají bezepečná rychlost podle předcházejícího vozidla. Předpokládá se přitom, že řidič volí svoji rychlost tak, aby zajistil bezpečné zastavení nebo aby dokázal náhle rychle zastavit při nenadálé události.

První podmínka, kterou aplikujeme na vozidlo n, souvisí s rychlostí vozidla. Ta nepřekročí jeho preferovanou rychlost V_n a jeho zrychlení na volné komunikaci[4] se zvyšuje stejně tak jako rychlost dokud se také zvyšuje kroutící moment motoru vozidla a poté začne klesat na nulovou hodnotu a té nabude ve chvíli, kdy se vozidlo bude pohybovat svoji preferovanou rychlostí V_n

$$v_n(t+T_n) = v_n(t) + 2,5 a_n \tau \left(\frac{1 - v_n(t)}{V_n}\right) \sqrt{\left(0,025 + \frac{v_n(t)}{V_n}\right)}. \tag{1.12}$$

Koeficienty v nerovnici 1.12 byly stanoveny na základě z měření na hlavních dopravních komunikacích při průměrném provozu. Využití nerovnice 1.12 pro tento model je považované za

[4] tj. komunikace, na které se nevyskytuje žádné další vozidlo

přijatelné až do chvíle, kdy se vozidlo přiblíží vozidlu jedoucím před ním. Od té chvíle důraz na tuto podmínku klesá, naopak největší důležitost je ve chvíli, kdy vozidlo nemá před sebou žádné vozidlo či vozidlo před ním se nachází velmi daleko.

Další omezení, které je nutno zmínit, se týká brždění. Jestliže $n-1$ (první) vozidlo zahájí brždění v čase t, zpomalí a následně zastaví v místě $x_n^* - 1$, které dostáváme rovnicí

$$x_{n-1}^* = x_{n-1}(t) - \frac{v_{n-1}(t)^2}{2b_{n-1}}, \tag{1.13}$$

kde se člen b_{n-1} nazývá nejvyšší kritická decelarace, kterou je řidič vozidla $n-1$ schopen vykonat. Její hodnota je vždy záporná.

Vozidlo n jedoucí za ním nebude reagovat na toto zpomalení ihned, ale až v čase $t + T_n$ a tudíž zastaví až v místě x_n^* dané rovnicí

$$x_n^* = x_n(t) + [v_n(t) + v_n(t+T_n)]\,\frac{T_n}{2} - \frac{v_n(t+T_n)^2}{2b_n} \tag{1.14}$$

A proto pro bezpečnost musí řidič vozidla n zajistit, že bude splněna podmínka

$$x_{n-1}^* - s_{n-1} < x_n^*, \tag{1.15}$$

kde s_{n-1} je délka vozidla $n-1$ sečtena s bezpečnou vzdáleností následujícího (n) vozidla. Nicméně pokud tato podmínka neplatí, vozidlo nemá žádný prostor pro případnou řidičovu chybu. Z toho důvodu obsahuje tento model ještě další podmínku; bezpečný odstup s další časovou rezervou, θ. Řidič začne reagovat na vozidlo před sebou až v čase $t + T_n$. Od této chvíle tedy máme reakční čas T_n a bezpečný reakční čas $T_n + \theta$, který budeme ve výpočtech dále využívat. S využitím rovnic (1.13), (1.14) a (1.15) můžeme tuto podmínku přepsat do rovnice

$$x_{n-1}(t) - \frac{v_{n-1}(t)^2}{2b_{n-1}} - s_{n-1} \geq x_n(t) + [v_n(t) + v_n(t+T_n)]\,\frac{T_n}{2} + v_n(t+T_n)\theta - \frac{v_n(t+T_n)^2}{2b_n} \tag{1.16}$$

Bez parametru θ by řidič vozidla byl nucen brzdit až ve chvíli, kdy opravdu musí, to znamená na maximální výkon brzd. Do té chvíle by se stále pohyboval svoji preferovanou rychlostí. Parametr θ slouží k dřívějšímu a přitom ne tak prudkému brždění.

Při skutečném pozorování dopravy jsme schopni změřit všechny parametry vozidla n v rovnici (1.16) kromě členu b_{n-1}. Tento člen nahradíme odhadnutou hodnotou $\hat{b}$, upravíme tuto rovnici a dostáváme

$$-\frac{v_n(t+T_n)^2}{2b_n} + v_n(t+T_n)\left(\frac{T_n}{2} + \theta\right) - [x_{n-1}(t) - s_{n-1} - x_n(t)] + v_n(t)\frac{T_n}{2} + \frac{v_{n-1}(t)^2}{2\hat{b}} \leq 0. \tag{1.17}$$

Relativní hodnoty T_n a θ jsou důležité v určování chování vozidel. Stejně jako v případě Newellova modelu 1.1, (1.6), pokud je hodnota θ rovna právě jedné polovině T_n, vozidlo jedoucí bezpečnou rychlostí bude schopno udržovat tento stav nekonečně dlouhou dobu. Díky tomu můžeme rovnici (1.17) přepsat do tvaru

$$-\frac{v_n(t+T_n)^2}{2b_n} + v_n(t+T_n)T_n - [x_{n-1}(t) - s_{n-1} - x_n(t)] + v_n(t)\frac{T_n}{2} + \frac{v_{n-1}(t)^2}{2\hat{b}} \leq 0. \tag{1.18}$$

Z toho plyne

$$v_n(t+T_n) \leq b_n T_n + \sqrt{\left(b_n^2 T_n^2 - b_n\left(2\left[x_{n-1}(t) - s_{n-1} - x_n(t)\right] - v_n(t)T_n - \frac{v_{n-1}(t)^2}{2\hat{b}}\right)\right)}. \tag{1.19}$$

Z nerovnice (1.18) vychází, že bezpečné rychlosti v_n budou ležet mezi dvěma kořeny této nerovnice, ale pokud bude mít spodní kořen zápornou hodnotu můžeme ho ignorovat, protože nás zajímají pouze kladné hodnoty rychlostí. Podcenění plynulého brždění řidiče vozidla n nastane ve chvíli, kdy

$$v_n(t + T_n) < v_n(t) + b_n T_n.$$

V tuto chvíli musí vozidlo začít brzdit rychleji, než by sám řidič chtěl. Řidič tedy volí rychlost na základě preferované decelarace, ale může brzdit i rychleji pokud by to bylo nutné.

Nerovnice (1.12) pro vozidlo n na volné komunikaci a (1.19) pro vozidlo n před kterým se nachází vozidlo $n-1$ představují dvě hlavní podmínky pro rychlost v_n v čase $t + T_n$ a jestliže řidič přizpůsobí rychlost parametrům vozidla a tak, aby jízda byla bezpečná, dostáváme **konečný vztah pro Gippsův model:**

$$\begin{aligned} v_n(t+T_n) = \min \Bigg(& v_n(t) + 2,5 a_n \tau \left(\frac{1 - v_n(t)}{V_n} \right) \sqrt{\left(0,025 + \frac{v_n(t)}{V_n} \right)}, \\ & b_n T_n + \sqrt{\left(b_n^2 T_n^2 - b_n \left(2 \left[x_{n-1}(t) - s_{n-1} - x_n(t) \right] - v_n(t) T_n - \frac{v_{n-1}(t)^2}{2\hat{b}} \right) \right)} \Bigg) \end{aligned} \tag{1.20}$$

Pokud platí pro téměř všechna vozidla nižší hodnota v rovnici (1.20), vychází se z nerovnice (1.19), vozidla jedou blízko sebe, z čehož vyplývá vysoká hustota provozu. Pokud platí opak, tz. nerovnice (1.12), dopravní proud je volný.

1.3 Kraussův model

Pro simulaci v této práci budeme využívat software SUMO[5] a součástí tohoto programu je předpřipravený Kraussův model vytvořený v rámci disertační práce stejnojmeného autora. V popisu tohoto modelu budeme vycházet z této práce[9].

Tento model se zaměřuje přimárně na všeobecné vlastnosti dopravního proudu, zkoumání chování řidiče zde není primární. Myšlenka autora je taková, že existují obecné vlastnosti dopravního proudu, ze kterých vyplývá chování jednotlivých účastníků silničního provozu a není to chování jednotlivce, které hlavně ovlivňují vlastnosti dopravního proudu.

1.3.1 Obecný přístup

Pokud se dopravní proud modeluje mikroskopicky, musí být brány v úvahu dva typy pohybu vozidla. Prvním typem pohybu je vozidlo jedoucí po vozovce osamoceno, zatímco ve druhém případě vozidlo interaguje s ostatními vozidly. Na základě těchto pohybů předpokládáme následující podmínky. První podmínka má přímo souvislost s rychlostí vozidla. Ta je omezena nějakou maximální rychlostí $v_{\max}$,

$$v \leq v_{\max}. \tag{1.21}$$

Za maximální rychlost může být například zvolena preferovaná rychlost vozidla.

Hlavní důvod, proč mezi sebou vozidla interagují, je fakt, že řidiči nemají v úmyslu srážet se s ostatními vozidly. Další podmínkou tedy bude tzv. „nekoliznost“ systému. Budeme předpokládat,

[5] Simulation of Urban MObility

že řidič vozidla zvolí tedy takovou rychlost vozidla, která nebude vyšší než maximální bezpečná rychlost v_{safe},

$$v \leq v_{\text{safe}}. \tag{1.22}$$

Z maximální bezpečné rychlosti poté plyne interakce mezi vozidly. Její stanovení bude uvedeno dále.

Je možné formulovat modely na základě podmínek (1.21) a (1.22). Nicméně je vcelku vhodné předpokládat také omezující podmínky pro zrychlení,

$$\begin{aligned} &-b \leq \frac{dv}{dt} \leq a, \\ &a, b > 0. \end{aligned} \tag{1.23}$$

Později bude ukázáno, že podmínka (1.23) je nezbytnou složkou, pokud chceme modelovat dopravní proud správně.

Předchozí podmínky můžeme shrnout do následující nerovnice

$$v(t + \Delta t) \leq \min(v_{\max}, v_{\text{safe}}, v(t) + a\Delta t), \tag{1.24}$$

kde se maximální bezpečná rychlost v_{safe} musí splňovat podmínku

$$v(t + \Delta t) \geq v(t) - b\Delta t. \tag{1.25}$$

Celkové povědomí o tom, jak se jednotlivá vozidla pohybují a interagují mezi sebou závisí na výpočtu maximální bezpečné rychlosti v_{safe}. Jakmile je stanovena hodnota v_{safe}, nerovnice (1.24) představuje aktualizované schéma pro simulace dopravního proudu, pokud jsou splněny všechny podmínky zmíněné výše.

1.3.2 Interakce mezi vozidly

Pro vytvoření sofistikovaného dopravního modelu je nutné ukázat, jak vozidla mezi sebou interagují. Podívejme se na dvě vozidla na komunikaci, první vozidlo se nachází v bodě x_1 a jede rychlostí v_1, vozidlo, které ho následuje je v bodě x_f a pohybuje se rychlostí v_f. Jestliže délka prvního vozidla bude l, tak mezera mezi vozidly se vypočte vztahem

$$g = x_1 - x_f - l. \tag{1.26}$$

Jak již bylo zmíněno, hlavní důvod, proč mezi sebou vozidla interagují, je fakt, že řidiči nemají v úmyslu srážet se s ostatními vozidly. Z toho plyne, že mezera mezi vozidly musí být nezáporná. Narozdíl od jiných modelovaných přístupů, zde nezačneme s předpokladem jak může být zrychlení vozidla vypočteno z vozidla jedoucí před ním, protože nekoliznost stejně není splněna automaticky a někdy je velice složité ji dokázat.

Místo toho začneme zavedením toho, že se v spojitých modelech spolu interagující vozidla nesrazí tehdy, pokud je mezera mezi vozidly g větší než nějaká preferovaná mezera mezi vozidly g_{des} a splňuje tuto dynamickou nerovnici

$$\frac{dg}{dt} \geq \frac{g_{\text{des}} - g}{\tau_{\text{des}}}. \tag{1.27}$$

Preferovaný bezpečný časový odstup mezi vozidly τ_{des} a preferovaná mezera g_{des} by mohly funkcemi odstupu mezi vozidly či jejich rychlostmi. Fakt, že v tomto modelu nemohou nastat

kolize je zřejmá, protože pokud položíme g rovné nule časová derivace bude vždy nezáporná a preferovaná mezera g_{des} také.

Předpokládejme případ dvou vozidel jedoucími rychlost v_1 resp. v_f s odstupem g. Druhé vozidlo jede svoji bezpečnou rychlostí a řidič tohoto vozidla může zastavit za jakýkoliv okolností tak, aby nedošlo ke kolizi s prvním vozidlem. To znamená, že pokud řidič má reakční čas τ a brzdná vzdálenost vozidel jedoucích rychlostí v je dána funkcí $f(v)$, je situace bezpečná právě tehdy když

$$f(v_f) + v_f\tau \leq f(v_1) + g. \tag{1.28}$$

Hodnota funkce $f(v)$ nemusí být nutně minimální možná brzdná vzdálenost, ale může to být jiná funkce závisející na způsobu řízení řidiče a pohodlné jízdy. To samé platí pro reakční čas τ.

Úprava nerovnice 1.28 vede k bezpečnostnímu pravidlu pro rychlost druhého vozidla. Nicméně my toto pravidlo neodvozujeme ze dvou důvodů. První z nich je ten, že po úpravě bychom potřebovali znát přesný předpis funkce $f(v)$, který jak uvidíme není možné zjistit. Zadruhé je zbytečné a výpočetně náročné používat typ řízení, které potřebuje předvídat nenádalou událost a vypočítávat decelaraci až k úplnému zastavení každý jednotlivý krok. Proto zavedeme Taylorův rozvoj brzdné vzdálenosti $f(v)$ okolo průměrné rychlosti obou vozidel,

$$\overline{v} = \frac{v_1 + v_f}{2}, \tag{1.29}$$

který budeme v dalším odvození dále používat. Sudé členy tohoto rozvoje se vyruší a zanedbáním členů vyšších řádů dostáváme

$$f'(\overline{v})v_f + v_f\tau \leq f'(\overline{v})v_1 + g. \tag{1.30}$$

Smysl derivace $f(\overline{v})$ vysvětlíme jednoduše. Pokud se podívame na decelaraci vozidla b z rychlosti v na nulu (nemusí být nutně konstantní), tedy $\dot{v} = -b(v)$, kdy $b(v) > 0$, obrdžíme vztah

$$f'(v) = -\frac{d}{dv}\int_v^0 \frac{v'}{b(v')}dv' = \frac{v}{b(v)} \tag{1.31}$$

Nyní můžeme přepsat podmínku bezpečnosti do tvaru

$$v_1 - v_f \geq \frac{v_1\tau - g}{\frac{\overline{v}}{b(\overline{v})} + \tau}, \tag{1.32}$$

kde

$$v_1 - v_f = \frac{dg}{dt}. \tag{1.33}$$

Nerovnice (1.32) se v podstatě rovná nerovnici (1.27), kde preferovaný odstup $g_{\text{des}} = v_1 t$ a preferovaný bezpečný časový odstup mezi vozidly $\tau_{\text{des}} = \tau_b + \tau$, kde $\tau_b = \overline{v}/b(\overline{v})$ je stanoveno na základě decelerací, které řidič využije.

1.3.3 Diskretizace modelu

Nerovnice v části 1.3.1 je potřeba převést na diskrétní rovnice pro dráhy, rychlosti a zrychlení pro možnost simulace na počítači.

Vhodná cesta pro vytvoření rovnic dynamického systému z bezpečnostní podmínky (1.27) je převést spojitou rychlost v_f ve výrazu $\dot{g}(t) = v_1(t) - v_f(t)$ jako rychlost diskrétní s diferencí či integračním krokem $t + \Delta t$. Potom rovnici (1.27) přepíšeme do tvaru

$$v_f(t + \Delta t) \leq v_1(t) + \frac{g(t) - g_{\text{des}}}{\tau_{\text{des}}}. \tag{1.34}$$

Rovnice pro polohu vozidla se potom na stejném principu převede na

$$x(t + \Delta t) = x(t) + v(t + \Delta t)\Delta t. \tag{1.35}$$

Víme, že při $\Delta t \to 0$ a $g_{\text{des}} \geq 0$ rovnice (1.34) a nerovnice (1.35) garantují bezpečnost. Pro konečný čas Δt musí být tato garance opět dokázána.

Mezera $g(t)$ mezi dvěma vozidly je diskretizována vztahem

$$g(t + \Delta t) = g(t) + \Delta t(v_1(t + \Delta t) - v_f(t + \Delta t)). \tag{1.36}$$

Přidáním rovnice (1.34) dostáváme

$$\xi(t + \Delta t) = \xi(t) + \left(1 - \frac{\Delta t}{\tau_b + \tau}\right) + \Delta t \frac{g_{\text{des}} - v_1(t)\Delta t}{\tau_b + \tau}, \tag{1.37}$$

kde

$$\xi(t) = g(t) - v_1(t)\Delta t.$$

Bezpečnost ($g \geq 0$) je tedy garantována tehdy, když platí:

$$\begin{aligned} &\xi(t = 0) \geq 0, \\ &\Delta t \leq \tau, \\ &g_{\text{des}} \geq v_1 \Delta t. \end{aligned} \tag{1.38}$$

To je výsledek, který se ovšem dal předpokládat. Záleží pouze na tom, zda je i po provedení iteračního kroku stále splněna podmínka bezpečnosti, tz. kontrolovat jestli je náš iterační krok menší než reakční čas řidiče τ.

1.3.4 Rovnice modelu

Předpokládáme tedy, nezávisle na náhodném kolísání, že každé vozidlo se pohybuje nejvyšší rychlostí na základě podmínek, které jsou stanoveny níže. **Kraussův model tedy stanovíme následujícími rovnicemi:**

$$\begin{aligned} &v_{\text{safe}}(t) = v_1(t) + \frac{g(t) - g_{\text{des}}}{\tau_b + \tau}, \\ &v_{\text{des}} = \min[v_{\max}, v(t) + a(t)\Delta t, v_{\text{safe}}(t)], \\ &v(t + \Delta t) = \max[0, v_{\text{des}} - \eta]. \end{aligned} \tag{1.39}$$

Preferovaná mezera g_{des} může být zvolena různě. Většinou se volí $g_{\text{des}} = v_1\tau$, kde τ je reakční čas řidiče. Hodnota τ_b je definována jako podíl $\overline{v}/b$, kde $\overline{v}$ plyne ze vztahu (1.29). Nově zde také zavádíme proměnnou η, která je náhodnou výchylkou od běžného řízení. Volba integračního kroku Δt a preferované mezery g_{des} je závislá na podmínkách (1.38).

1.4 Intelligent Driver Model

V této sekci budeme primárně vycházet z práce [18], ve které je Intelligent Driver Model (IDM) do značné míry rozebrán. Využijeme hlavně podkapitolu 5.1 z [18], kterou doplníme o poznatky, které jsou uplatněny z kapitoly 1.

Intelligent Driver Model (IDM) je poměrně nový matematický model z roku 2000, který byl vymyšlen v Německu. Snaží se vylepšit některé nedostatky předcházejících modelů, ale i přispět novými myšlenkami v této oblasti.

1.4.1 Matematický popis

Pro výpočet zrychlení v IDM je použit vztah

$$\dot{v}_n = a_{\text{free}} + a_{\text{int}}, \tag{1.40}$$

kde se aktuální zrychlení účastníka vypočítá jako součet zrychlení při volném proudu a_{free} a zrychlení, ve kterém je zahrnut i účastník jedoucí před ním a_{int}. Zde je vidět určitá paralela s Gippsovým modelem (kapitola 1.2, kde se ovšem vybírá pouze jedna z těchto dvou hodnot. Navíc i rovnice se od IDM odlišují. Pro volný proud je tedy zrychlení $a_{\text{int}} = 0$.

Zrychlení při volném proudu je definováno vztahem

$$a_{\text{free}} = a\left[1 - \left(\frac{v_n}{v_{n\text{free}}}\right)^{\delta}\right], \tag{1.41}$$

kde a je maximální zrychlení, δ je exponent zrychlení, v_n je aktuální rychlost vozidla a $v_{n\text{free}}$ je rychlost, které chce vozidlo dosáhnout.

Pokud v rovnici (1.41) bude zlomek v_n/v_{free} roven jedné, auto se již dostalo na svou preferovanou rychlost v_{free} a celkové zrychlení a_{free} bude rovno nule. Naopak pokud bude v_n rovno nule, znamená to, že a_{free} bude přímo rovno a, a auto pojede svým maximálním zrychlením. Pokud v_n bude menší než v_{free}, pak platí že, čím vyšší hodnotu bude mít nastavenou parametr δ, tím se dosáhne vyšší hodnota a_{free}. Pokud v_n bude větší než v_{free}, bude platit pravý opak.

Auto jedoucí při volném proudu je ale nereálná představa, proto se do rovnice (1.40) přidává interakční člen, který přímo závisí na parametrech vozidla jedoucí před naším účastníkem. Tento člen je roven

$$a_{\text{int}} = -a\left(\frac{s_n^*}{s_n}\right)^2, \tag{1.42}$$

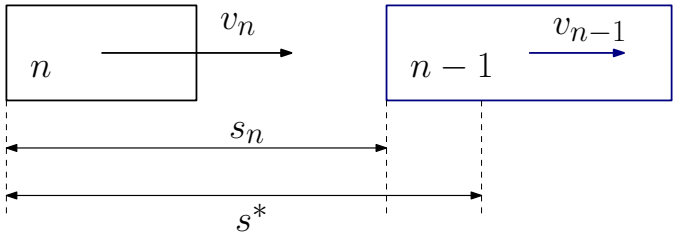
kde a je maximální zrychlení, $s_n = x_{n-1} - x_n$ je rozdíl poloh vozidel a s_n^*

$$s_n^* = s_0 + v_n T + \frac{v_n(v_n - v_{n-1})}{2\sqrt{ab}} \tag{1.43}$$

je preferovaný odstup. Hodnota s_0 se vypočte jako součet minimální bezpečné vzdálenosti mezi vozidly a délkou následujícího vozidla, T je bezpečnostní časový odstup mezi vozidly a b je konstanta brzdění. Situace je schématicky znázorněna na obrázku 1.4.

Člen úměrný $v_n - v_{n-1}$ je kvadratický v proměnné v_n a obsahuje součin $v_n v_{n-1}$. Tyto nelineární členy popisují interakci, v tomto případě dvou vozidel. Pokud tedy bude vzdálenost s_n mezi auty vysoká, potom zlomek s_n^*/s_n bude mít velmi malou hodnotu a interakční člen a_{int} lze zanedbat, takže vozidlo se pohybuje jako ve volném proudu.

Pokud rychlost auta v_n bude vyšší než rychlost auta v_{n-1}, pak interakční člen s_n^* bude kladný a požadovaný odstup se zvětší, takže zrychlení auta $\dot{v}_n$ se musí snížit.



Obrázek 1.4: Proměnné jednotlivých aut

1.5 Shrnutí

Časově spojité mikroskopické modely bez předjíždění jsou v zásadě definovány funkcí zrychlení. V nejstarších modelech může zrychlení $\dot{v}(t+T_r)$ vozidla α zapsáno jako

$$\dot{v}(t+T_r) = \frac{-\lambda v_\alpha^m \Delta v_\alpha}{s_\alpha^l}, \tag{1.44}$$

kde λ, l a m jsou parametry modelu, s je odstup mezi vozidly a Δv_α rozdíl rychlostí.

Kromě toho, může zrychlení může také záviset na vlastní rychlosti vozidla v_α a klesá se vzdáleností s_α k vozidlu před ním,

$$s_\alpha = x_{\alpha-1} - x_\alpha - l_\alpha, \tag{1.45}$$

kde l_α je délka vozidla α.

V závislosti na rovnici (1.44), kdy zrychlení závisí na vedoucím vozidle, tyto modely není možné použít při velmi nízkých hustotách provozu. Pokud není určeno vedoucí vozidlo ($s_\alpha \to \infty$), zrychlení buďto není možné stanovit (pro hodnotu $l = 0$) nebo je rovné nule ($l > 0$) bez ohledu na rychlost vozidla. Dá se předpokládat, že v tuto chvíli se vozidla budou snažit dosáhnout své preferované rychlosti. Chování vozidel v hustém provozu je ovšem poněkud nerealistické. Především odstup mezi vozidly s týkající se vedoucího vozidla nemusí nutně směřovat k rovnovážnému stavu. Ani malé mezery nepřimějí vozidlo k brždění pokud je rozdíl rychlostí Δv_α roven nule.

Tyto problémy byly vyřešeny v Newellově modelu (viz. odstavec 1.1), kde je z rovnice (1.10) vidět, že zrychlení není závislé pouze na rozdílů rychlostí. Newellův model je nekolizní, ale okamžitá závislost na rychlosti na hustotě provozu vede k velmi vysokým hodnotám zrychlení.[16]

Kromě Newellova modelu a dalších modelů pro základní výzkum existují také vysoce komplexní modely jako např. Wiedemannův. Ten se snaží napodobovat dopravu jak nejlépe to jde ovšem za cenu mnoha parametrů modelu.

Další model, který obsahuje realistické brzdné reakce řidičů, je Gippsův model 1.2 (brzdný koeficient b_n) a Kraussův model 1.3 (brzdný koeficient b). Navzdory své jednoduchosti, tyto modely ukazují realistické chování řidiče a jsou nekolizní. Toto chování se bohužel ztrácí ve chvíli, kdy modelujeme dopravu deterministicky[6]. Tím pádem není možné simulovat nestabilní stavy dopravy.

[6] např. když vozidla vjíždí do určité oblasti každých 5 sekund

2. Simulační a podpůrný software

V této kapitole rozebereme simulační software SUMO[1] a další, ať již naprogramované autorem či volně dostupné nástroje.

2.1 SUMO

Po prostudování modelů byla otázka dalšího postupu. Bylo možné pokračovat v aplikaci, kterou autor vytvořil pro svoji bakalářskou práci, další variantou bylo využít softwaru AIMSUN, který by byl také vhodný pro naprogramování vlastních modelů. Nicméně zvítězil open-source simulační software SUMO. Tento produkt se neustále vyvíjí již od roku 2001 a jeho podpora neustále pokračuje. Nejprve se autor pokusil začít pracovat v AIMSUNu, nicméně to skončilo neúspěšně, protože pro naprogramování dalších modelů do tohoto software byl potřeba balíček, který nebyl zdarma.

Další variantou bylo pokračovat v programovacím jazyku JAVA na vytvořené aplikaci pro bakalářskou práci, nicméně po prozkoumání dalších variant se autor rozhodl, že bude výhodnější pokračovat s již vyvíjeným softwarem SUMO.

SUMO je open-source nástroj pro dopravní simulace. Ve výzkumu se dá SUMO využít v mnoha oblastech např. vyhledání nejkratší cesty[2], možnost zkoušení signálních plánů nebo simulace komunikace mezi vozidly. SUMO bylo a je využíváno na mnoha zajímavých projektech[3].

Od roku 2001 se SUMO postupně rozšiřovalo a nyní je z něj již soubor několika programů. Umožňuje na základě dopravních měření, matic vzdáleností využívat různé routovací algoritmy, simulování křižovatek. Obsahuje také rozhraní TraCI pro online simulaci.

Při vývoji SUMO byly hlavní dva důvody proč by měl být opensource. První z nich je snaha podporovat komunitu, která se zabývá simulacemi, volně dostupným nástrojem ve kterém již budou zahrnuty základní algoritmy. Existují některé další volně dostupné simulační softwary, na kterých ovšem nepracovali studenti a přestaly být dříve či později podporovány. Druhým důvodem je neexistující porovnatelnost implementovatelnost modelů a algoritmů.

Tento druhý důvod a také možnost vlastní implementace modelů vedly autora k volbě SUMO. K tomu, aby bylo možné vlastní modely vůbec implementovat, bylo nejprve nutné vlastní SUMO zkompilovat, to znamená, že nestačí pouze verze SUMO, které obsahuje spustitelné soubory, ale přímo zdrojové kódy. Více o tom jak SUMO zkompilovat se čtenář dozví v příloze A.

2.1.1 Jak to funguje

Nyní již víme, čeho všeho je SUMO přibližně schopno. K tomu abychom spuštěna simulace je zapotřebí mít připraveny minimálně tři typy souborů, a to

- konfigurační soubor,
- soubor s podkladem (dopravní síť),
- soubor s vozidly.

[1] Simulation of Urban MObility

[2] či podle jiného kritéria

[3] např. předpověď dopravních proudů v době MS v Německu v roce 2006 v Kolíně nad Rýnem, využití komunitou V2X pro zjišťování dráhy vozidla atd.

2.1.2 Konfigurační soubor

V konfiguračním souboru je možné nastavit všechny možnosti, které obsahuje SUMO jako parametry příkazové řádky. To se může hodit obzvlášť ve chvíli, kdy se SUMO volá s mnoha parametry. Tehdy ho stačí zavolat pouze s konfiguračním souborem.

2.1.3 Soubor s podkladem

Pro diplomovou práci dostal autor data konkrétně průměrné intenzity a rychlosti z mýtných bran Pražského okruhu. Bylo proto vhodné mít jako podklad pro simulaci síť v konkrétních úsecích mezi mýtnými branami. Pro mapu oblasti bylo využito volně dostupných map ze serveru OpenStreetMap, kde se konkrétní úsek stáhnul ve formátu OSM a vymazaly se všechny prvky nesouvisející s Pražským okruhem. Posléze byl použit jeden z nástrojů SUMO a to konkrétně netconvert, který převádí data z předem definovaného formátu do SUMO formátu. Více k tomuto tématu v příloze B.

SUMO také umožňuje využít vlastní nástroj na generování sítí, JTRrouter či DFrouter.

2.1.4 Vozidla pro simulaci

Vozidla mají v SUMO několik parametrů, které mimo jiné závisí také na volbě mikroskopického modelu. Mezi základní patří

- id vozidla,
- typ vozidla,
- dráha, po které se bude vozidlo pohybovat,
- čas vjezdu do oblasti,
- místo vjezdu do oblasti,
- rychlost při vjezdu do oblasti,
- jízdní pruh při vjezdu do oblasti.

Můžeme na začátku definovat typ vozidla, což znamená jeho parametry jako např. délka vozidla, zrychlení, max. rychlost atp. Poté u každého vozidla nastavíme tento typ a vozidlo automaticky dostane všechny přiřazené parametry.

Dráha, po které se bude vozidlo pohybovat, souvisí s odstavcem 2.1.3, kde přímo tyto jízdní úseky vytváříme. Tento parametr může obsahovat více hodnot, v případě Pražského okruhu minimálně část mezi dvě mýtnými branami.

2.2 Příprava a vyhodnocení simulace - program

Aby bylo možné dostat se k jádru celé práce, je nutná příprava dat pro simulaci a také jejich následné vyhodnocení. K tomuto účelu byla vytvořena konzolová aplikace, která umožňuje 4 základní operace

1. převedení formátu dat (MATLAB) z Pražského okruhu na formát podporovaný softwarem SUMO (XML)
2. vytvoření možností, jak vytvářet vstupní soubor vozidel

3. vytvoření souboru, který po následném spuštění programem gnuplot vytvoří graf časové řady

4. vytvoření souboru, který po následném spuštění programem gnuplot vytvoří histogram

Tato aplikace byla vytvořena v jazyku JAVA z důvodu nalezení balíčku pro možnost práce s daty ze softwaru MATLAB. Dále se autor rozhodl, že vytvoří raději jednu přehlednou aplikaci než 5 malých.

2.2.1 Třída AllInOne

V hlavní třídě se nachází konstruktor, který na základě vstupních parametrů zavolá příslušnou třídu, případně vytiskne chybovou hlášku s problémem. Dále je zde funkce na zjištění, zda zadaný parametr je číslo.

- *public AllInOne(String args[])* - konstruktor, větvení na základě vstupního pole *args*
- *public static boolean isInteger(String s)* - funkce na kontrolu, zda je vstupní řetězec *s* číslo

2.2.2 Převedení formátu dat

Pro SUMO je možné vytvořit si jakýkoliv soubor vozidel. SUMO obsahuje přímo manuál, jak tento soubor vytvořit. Není ovšem vhodné vytvářet ho ručně, nýbrž použít nějaký nástroj pro XML.

Třída, která toto umí v Javě, je nazvána *MatlabToXML*. V konstruktoru se vytvoří soubor, jehož název je jeden ze vstupních parametrů a poté ve funkci *createLoopDocument* se přímo vytváří jeho obsah.

- *public MatlabToXML(String fileNameMatlab, String fileNameCars)* - první vstupní parametr určuje soubor ve formátu .mat, který má předem definovanou strukturu, která je shodná s mýtnými branami z Pražského okruhu. Výstupní soubory se potom budou jmenovat na základě vstupní proměnné *fileNameCars*, kde dále bude následovat potržítko a číslo pruhu[4], ve kterém budou informace o rychlostech a intenzitách pouze pro ten daný pruh.
- *public Document createLoopDocument(int lane)* - vstupní parametr *lane* určuje pruh, pro který se právě obsah souboru vytváří.

Výstupní soubor bude shodný s výstupním souborem po simulaci SUMO, takže bude možné přesně jednotlivé hodnoty z mýtných bran porovnat s nasimulovanými.

Příkaz, kterým lze program spustit aby vytvořil tento dokument, je následující

```
1 java -jar out\artifacts\AllInOne_jar\AllInOne.jar -m sokp-0187-20121031.mat det0187.xml
```

Ze souboru *sokp-0187-20121031.mat* vytvoří v závislosti na počtu pruhů pod touto mýtnou branou stejný počet souborů, tedy *det0187_0.xml*, *det0187_1.xml* atd.

Tyto soubory lze následně použít jako vstup pro vytvoření souboru s vozidly pro SUMO a nebo pro vytvoření grafů nebo histogramů pro porovnání výsledků se simulací.

[4]Např. pro vstupní parametr detektorkm0201.xml, budou výstupní soubory detektorkm0201_0.xml a detektorkm0201_1.xml

Literatura

[1] Barceló J, *Fundamentals of traffic simulation*, proceedings, International Series in Operations Research and management science, Springer, 2010

[2] Barrow J, *Nové teorie všeho*, Dokořán, Praha, 2008

[3] Behrisch M, Bieker L, Erdmann J, Krajzewicz D, *SUMO - Simulation of Urban MObility*, Dokořán, Praha, 2008

[4] Brdička M, Samek L, Sopko B, *Mechanika kontinua*, Academia, Praha, 2011

[5] Gipps, P. G. (1981). A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2), 105-111.

[6] Haberman R, *Mathematical models: Mechanical vibrations, population dynamics and traffic flow*, Society for Industrial and Applied Mathematics, Philadelphia, 1988

[7] Helbing D, Herrmann H J, Schreckenberg M, Wolf D E, *Microscopic Simulation of Congested Traffic* v knížce Traffic and Granular Flow, Springer, Berlin, 2000

[8] Horák J, Krlín L, Raidl A, *Deterministický chaos a jeho fyzikální aplikace*, Academia 2003

[9] KRAUSS, Stefan. *Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics.* Köln, 1998. PhD. Thesis. Universität Köln.

[10] Xiaoliang Ma, *A Neural-Fuzzy Framework for Modeling Car-following Behavior* [online], dostupné z `http://www.ctr.kth.se/publications/ctr2006_08.pdf`

[11] May A D, *Traffic flow fundamentals*, Prentice Hall, 1989

[12] Newell, G. F. (2002). A simplified car-following theory: a lower order model. *Transportation Research Part B: Methodological*, 36(3), 195-205.

[13] Přikryl P, *Numerické metody matematické analýzy*, SNTL, Praha, 1988

[14] Scholtz M, *Classical mechanics and deterministic chaos* [online], dostupné z `http://www.fd.cvut.cz/personal/scholma1/`

[15] Scholtz M, Vaniš M, Veselý P, Matějka P, *Applied mathematics on Faculty of Transportation Sciences*, vyjde ve sborníku k výročí Fakulty dopravní ČVUT

[16] Treiber M, Hennecke A, Helbing D, *Congested traffic states in empirical observations and microscopic simulations*, Physical Review E, **62** (2), pp. 1805–1824, 2000

[17] Treiber M, *Microsimulation of road traffic flow* [online], dostupné z `http://www.traffic-simulation.de/`

[18] VANIŠ, Miroslav. *Matematické modelování vybraných problémů v dopravě v jazyce Java.* Praha, 2013. Bakalářská práce. FD ČVUT.

[19] Vitásek E, *Numerické metody*, SNTL, Praha, 1987

A. Příloha A: Kompilace simulačního softwaru SUMO

A.1 SUMO

SUMO je volně šiřitelný dopravní simulátor, který vznikl v roce 2001. SUMO umožňuje modelování intermodálních dopravních systémů. Ty mohou obsahovat dopravní prostředky, městskou hromadnou dopravu či chodce. Dále v SUMO existují nástroje pro vyhledávání cesty, visualizace, import či export map, počítání emisí a další. Nejdůležitější vlastností pro nás je možnost implmentování vlastního dopravního modelu. SUMO samo o sobě již obsahuje některé modely - Kraussův model 1.3, Intelligent Driver Model **??** a Wiedermanův model.

A.1.1 SUMO download

Na stránkách projektu SUMO[1] se stáhne aktuální verze SUMO na pravé straně. Uvažujeme pouze případ dopravního simulátoru SUMO pod Windows, v době vzniku byla na světě verze SUMO 0.22. Ideální je varianta stáhnutí tří zkomprimovaných souborů. Pro potřeby kompilace stačí pouze soubor SUMO 0.22 sources (sumo-src-0.22.0,14 MB). V souboru SUMO 0.22 manual, tutorial and examples (sumo-doc-0.22.0.zip, 35 MB) se nachází podrobná dokumentace ke všem naprogramovaným součástem SUMO a SUMO 0.22 for Windows (sumo-winbin-0.22.0.zip, 33 MB) obsahuje již zkompilované SUMO, které je možné rovnou použít. V tomto souboru se ve složce *docs/userdoc* nachází uživatelská příručka, která obsahuje i návod na zkompilování SUMO pod Windows. Tento návod ovšem v některých částech není dostatečně přesný.

A.2 Visual Studio

Pro zkompilování SUMO je zapotřebí mít nainstalované Visual Studio. To je možné získat přímo na stránkách Visual Studia[2] a po zaregistrování je možno získat Expres edici Visual Studia. Autor pracuje s Visual Studio 2010 Pro. Kompilace SUMO je tedy vyzkoušena na této verzi. **Pro správné fungování Visual Studia je nutné mít nainstalován Microsoft .NET Framework verze 4 ne vyšší!**

A.3 Ostatní závislosti

SUMO ke svému zkompilování potřebuje další knihovny. V první řadě je to **Xerces-C** (xerces-c-3.1.1-x86-windows-vc-10.0.zip, 23 MB)[3], které už je samo o sobě zkompilované. Podstatné je stáhnout verzi x86 a ne 64bitovou, aby s ní byly schopni další součásti pracovat.

Další závislostí jsou **knihovny FOX**, kterou si budeme muset ve Visual Studiu zkompilovat sami. V návodu u SUMO je uvedeno, že je vyzkoušena pouze verze 1.6.36[4]. Přesně pro tuto jedinou verzi se mi povedlo provést kompilaci v pořádku.

[1] http://www.dlr.de/ts/sumo/en
[2] http://www.visualstudio.com
[3] http://mirror.hosting90.cz/apache//xerces/c/3/binaries/xerces-c-3.1.1-x86-windows-vc-10.0.zip
[4] http://ftp.fox-toolkit.org/pub/fox-1.6.36.zip

Poslední částí pro zdárnou kompilaci jsou FWTools. Verze doporučená SUMO ovšem nefunguje a ani nikde na internetu se ji nepodařilo nalézt. Funkční je ovšem verze 2.4.7[5].

Důrazně se doporučuje ani jednu z těchto částí neinstalovat do klasického adresáře Program Files, může to způsobit problémy.

Nutné je také mít nainstalovaný Python (pro něj neplatí předchozí odstavec), ovšem pozor rozhodně ne nejnovější verzi[6], nýbrž verzi 2.7[7]. U Pythonu není nutné žádné další nastavení.

A.3.1 Xerces-C

Nejprve rozbalíme obsah archivu a posléze překopírujeme ze složky *bin* obě dll knihovny[8] do složky *Windows/System32*. Přes veškerou snahu se nepodařilo XERCES naimportovat všem projektům. Tím pádem se musí celý obsah složky *include* překopírovat do složky, kde je nainstalované Visual Studio do složky *vc/include* [9]. Výsledkem tedy bude překopírovaná složka *Xercesc* ze složky *Xerces/include/* do složky */vc/include*. Pro kompilaci a následné spuštění zkompilovaných programů je nutno vytvořit systémovou proměnnou XERCES[10], jejíž hodnota bude nastavena na kořenový adresář Xerces. Poté vyhledáme v systémových proměnných proměnnou PATH a nakonec hodnoty této proměnné přidáme %XERCES%/bin/.

A.3.2 FOX

FOX klasicky rozbalíme z archivu a poté musíme Visual Studiem zkompilovat. Otevřeme v rozbaleném archivu složku *windows/vcpp* a v ní otevřeme Visual Studiem soubor *win32.dsw*. Dle verze Visual Studia se může objevit hlášení o nutnosti konvertování. Potvrdíme pro všechny (možno zaškrtnout) a necháme Visual Studio načíst FOX. Po načtení musíme vybrat možnost kompilace **Release** a zkompilujeme **pouze** projekt foxdll. Pak provedeme to samé s tím rozdílem, že vybereme možnost **Debug**. Při kompilaci můžou vzniknout chyby, ty nás ovšem netrápí, protože to důležité by se mělo vytvořit. V kořenovém adresáři se vytvoří složka *lib* ve které se nacházejí zkompilované knihovny FOXDLL-1.6.dll a FOXDLLD-1.6.dll, které také překopírujeme do složky *Windows/System32*. Do systémových proměnných je nutné přidat proměnnou z názvem FOX16 s cestou do kořenového adresáře FOX[11] a následně na konec hodnoty systémové proměnné PATH přidat %FOX16%/lib.

A.3.3 FWTools

FWTools klasicky nainstalujeme (ideálně ne do složky Program Files). Vytvoříme stejně jako pro FOX systémovou proměnnou s názvem PROJ_GDAL, která bude opět odkazovat do kořenového adresáře FWTools. Poté vyhledáme v systémových proměnných proměnnou PATH a nakonec hodnoty této proměnné přidáme %PROJ_GDAL%/bin.

A.3.4 SUMO

V kořenovém adresáři SUMO ve složce **/build/msvc10** se nachází soubor s názvem *prj.sln*. Ten otevřeme Visual Studiem. Pokud je vše nastaveno správně, je možno spustit kompilaci

[5] http://home.gdal.org/fwtools/FWTools247.exe

[6] 3 a výše

[7] https://www.python.org/ftp/python/2.7.8/python-2.7.8.msi, 16 MB

[8] xerces-c_3-1.dll a xerces-c_3_1D

[9] např. c:/Program Files/Microsoft Visual Studio 10.0/VC/include/

[10] Start - Počítač - Vlastnosti systému - Upřesnit nastavení systému (nabídka na pravé straně) - Proměnné prostředí... - Systémové proměnné - Nová...

[11] např. C:/fox-1.6.36/

SUMO, to znamená zkompilovat všechno (všech 54 projektů). Pokud kompilujeme v režimu Release, dostaneme výstup, který se nachází v kořenovém adresáři ve složce *bin*. Tento výstup je v sumo-winbin-0.22.0.zip.

B. Příloha B: Vytvoření sítě pro simulaci v SUMO

B.1 OpenStreetMap

Pro vytvoření sítě bylo využito serveru OpenStreetMap, který obsahuje kromě pozemních komunikací spoustu dalších objektů. Nejdříve bylo na tomto serveru vybrána oblast, ve které se nacházejí mýtné brány. Posléze tato mapa byla vyexportována do souboru OSM. Ten ovšem obsahoval i další objekty, které nemají pro simulování žádnou relevanci. Tyto objekty bylo nutné odstranit.

K tomu se využil free software JOSM[1], kde se vybraly všechny prvky, které bylo nutné odstanit. Tyto prvky byly sice odstraněny, ovšem velikost souboru se ještě zvýšila, protože ke každému objektu byl pouze přirazen parametr skrytí.

Aby se prvky opravdu smazali byl využit software XMLStartlet[2], který přímo pracuje se souborem XML. Ve stejné složce, kde se nachází XMLStartlet se překopíruje soubor OSM s již odstraněnými prvky (*input.osm*) a s příkazové řádky se spustí příkaz

```
1 xmlstarlet ed -d "/osm/*[@action='delete']" < input > output.osm
```

kde *output.osm* je název výstupního souboru s přímo smazanými prvky.

B.2 netConvert

Pokud máme připravený soubor z odstavce B.1, zkopírujeme ho do složky se souborem *netconvert.exe*. Z příkazové řádky spustíme příkaz

```
1 netconvert --guess-ramps --remove-geometry --junctions.join --osm-files output.osm --
    output-file nodes.net.xml
```

Tento příkaz nám vytvoří soubor *nodes.net.xml*, který obsahuje již přesnou strukturu sítě, kterou bude možné otevřít v SUMO. Existuje mnoho parametrů netconvert, po několika zkouškách autor usoudil, že tyto parametry jsou nejblíže tomu, čeho se chce dosáhnout. Program *netconvert.exe* se pokusí uhádnout nájezdy a sjezdy s Pražského okruhu (*–guess-ramps*), které nejsou zřejmé ze souboru *output.osm*, s tím souvisí vytvoření propojení (*–junctions.join*) těchto nájezdů a sjezdů a vytvoření dalších uzlů (*–remove-geometry*). V tuto chvíli máme již použitelný soubor jako podklad pro simulaci.

SUMO pouští vozidla do oblasti tak, že vezme jeden z parametrů vozidla, který určuje vjezd do oblasti, v dalším parametru jsou informace o dalších hranách, které vozidlo projíždí. Problém výstupního souboru *nodes.net.xml*, se kterým jsou tyto parametry vozidla srovnávány, spočívá v tom, že hrany a uzly jsou 8 až 9ti místná čísla, která nemají na první pohled danou spojitost. Je tedy nutné je přejmenovat.

Nejlepší způsob je asi v použití SUMO-GUI, kde při najetí na prvek zjistíme jeho ID a potom v souboru *nodes.net.xml* nahradíme předem vymyšleným ID z nějakého vlastního systému. Autor zvolil číslování od 1 a v závislosti na směru bud písmeno a nebo b. Takže hrana má např. název 1a, 3b.

[1] https://josm.openstreetmap.de/
[2] http://xmlstar.sourceforge.net/