UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Filip Novović

RAČUNARSKI SISTEM ZA ASISTENCIJU PRI PRETICANJU U BEŽIČNIM MREŽAMA VOZILA

master rad

Beograd, 2021.

Mentor:

dr Aleksandar KARTELJ, docent
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Vladimir FILIPOVIĆ, redovni profesor
Univerzitet u Beogradu, Matematički fakultet

dr Mladen NIKOLIĆ, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Naslov master rada: Računarski sistem za asistenciju pri preticanju u bežičnim mrežama vozila

Rezime: Uticaj tehnološkog napretka na računarske sisteme automobila otvorio je vrata raznim aplikacijama koje za cilj imaju povećanje performansi, bezbednosti ili efikasnosti korišćenja resursa u vožnji. U radu je predložen i razvijen računarski sistem zasnovan na bežičnim ad-hoc mrežama vozila koji za cilj ima asistenciju vozaču u vidu upozorenja u slučaju otkrivanja nepogodnog ishoda započetog manevra. Uz sam sistem, u kom su testirana dva algoritma za izračunavanje procene, pružen je i opis razvojnog i simulacionog okruženja, a radi razumevanja šireg konteksta upotrebe modula za procenu predložena je i okvirna modularizacija računarskog sistema koji bi mogao da podrži i veći broj manevara.

Ključne reči: VANET, MANET, preticanje, asistencija, vozila, simulacija, računarski sistem, algoritam

Sadržaj

Glava 1

Uvod

Sve brži i pouzdaniji protokoli, porast korišćenja senzora i njihovo povezivanje utiču u velikoj meri na razvoj različitih industrija. U agrikulturi senzori se koriste za merenje svojstava zemljišta [23]. U proizvodnji, daju osnov za implementaciju modernizovanih procesa (eng. *Industry 4.0*) [18]. Predviđa se da će do 2024. vrednost tržišta povezanih uređaja (eng. *Internet Of Things*) dostići 487,30 milijardi američkih dolara [21]. Jedna od industrija koju tehnološki napredak „gura" u polje raznolikih inovacija, a koja pored materijalne dobiti za cilj ima i zaštitu ljudskih života, jeste automobilska industrija [25]. Kroz intenzivan rad naučnog i inženjerskog kadra, uz podršku proizvođača vozila, ideja o autonomnim vozilima postaje realnost [6, 7]. Međutim, aktuelni sistemi uglavnom se zasnivaju na pojedinačnim upotrebama ugrađenih senzora i obradi vizuelnog okruženja (eng. *computer vision*) [14], koji povlače razne izazove poput variranja efikasnosti u zavisnosti od trenutnih vremenskih uslova [34], neprekidnog praćenja objekata (eng. *object tracking*), identifikacije i označavanja objekata na slici (eng. *semantic segmentation*) [15] itd.

Drugačiji pristup, koji se zasniva na bežičnom umrežavanju vozila i njihovoj komunikaciji, pružajući potencijal za dalje unapređenje računarskih sistema u vozilima, naziva se VANET (eng. *Vehicle Ad-hoc NETwork*). Sama ideja o bežičnoj komunikaciji između vozila nije nova, međutim, tek poslednjih godina, upravo usled zrelosti predložene tehnologije i novih standarda (npr. 5G telekomunikacione mreže [16]), stiču se uslovi za realizaciju njenog potencijala.

U ovom radu opisan je razvoj i testiranje automobilskog računarskog sistema zaduženog za asistenciju vozaču pri preticanju na putevima sa dve trake, zasnovanog upravo na komunikaciji između vozila. U uvodnom delu izloženi su opis, karakteristike i upotrebe VANET-a kako bi bio predstavljen potencijal, ali i izazovi koje

tehnologija pruža. Preticanje svakako nije jedina situacija u kojoj bi komunikacija između vozila mogla da posluži kao osnov sistema za asistenciju pri vožnji, te je u nastavku (glava 2) predložena modularizacija računarskog sistema koji bi mogao da podrži veći broj vozačkih manevara. Na osnovu te modularizacije, konkretan sistem zadužen za podršku pri preticanju biće opisan kroz definisanje zaduženja različitih modula. Ista celina prikazuje motivaciju za obradu manevra preticanja i ciljeve rada, dok su u glavi 3 predložena dva različita algoritma koji na osnovu trenutnog preseka stanja okolnog saobraćaja i njegovog fizičkog modela vrše procenu bezbednosti započetog preticanja. Odabir simulacionog alata i postavljanje realističnog simulacionog okruženja, kao i rezultati pokretanja oba algoritma prikazani su u glavi 4. Na kraju je dat osvrt na potencijalna unapređenja, kako samih algoritama, tako i okruženja u kom su oni testirani i razvijani. U ključnim implementacionim delovima izloženi su najbitniji pseudokôdovi koji će pomoći u sažetom objašnjavanju ideja iza algoritma.

Kako bi pretpostavke u daljim sekcijama bile intuitivne, neophodno je najpre ukratko opisati karakteristike komunikacije između vozila kao i pojasniti motivaciju za njen dosadašnji i budući razvoj. VANET predstavlja specijalni tip bežične mreže koja se naziva MANET (eng. *Mobile Ad-hoc NETwork*), a koja svoje korene dalje vuče sve do 1972. u vidu PRNET (eng. *Packet Radio NETwork*) mreža [5]. Za vreme svog zrelog postojanja, nailazi na širok skup primena, koji uključuje vojne tehnologije [27], akcije spasavanja [29] itd. Za postojanje navedenih mreža nije potrebna mrežna infrastruktura u vidu namenskih rutera već svaki čvor koji je u radijusu drugih može postati aktivni član komunikacije kao i ruter koji bi se koristio za decentralizovane metode prosleđivanja paketa. U glavne karakteristike MANET mreža ubrajaju se: distribuiranost, dinamična topologija i skalabilnost. Prva karakteristika podrazumeva da su sami čvorovi ravnopravni nosioci infrastrukture mreže i da ne postoji centralizovani kontrolni uređaj. Dinamična topologija govori o tome da se, stalnim dodavanjem čvorova i njihovim izlaženjem iz mreže, njena struktura po svojoj prirodi potencijalno neprestano menja. Kada je reč o skalabilnosti, mnoge upotrebe podrazumevaju uključenje velikog broja čvorova u istu mrežu, što po svojoj prirodi MANET i omogućava. Međutim, potrebno je takođe uzeti u obzir da zbog svoje prilagodljivosti i dinamičnosti, pitanje skalabilnosti nije tako jednostavno i da efikasnost rutiranja zavisi od tipa mreže [3].

Nakon sažetog uvoda u širi skup bežičnih ad-hoc mreža koje dele iste karakteristike, pažnju usmeravamo na upotrebu u saobraćajnoj delatnosti drumskog saobraćaja, odnosno VANET mreže. Pre ključnog dela izlaganja koje se odnosi na primenu

VANET mreže u konkretnom, izdvojenom manevru preticanja na putevima sa dve trake, ovde ćemo se još ukratko osvrnuti na njen potencijal, izazove i standarde.

Sposobnost komunikacije između aktera u saobraćaju značajno je pokrenula razvoj velikog broja potencijalnih primena koje za cilj imaju uglavnom povećavanje bezbednosti manevara u kojima je to moguće ili poboljšanje efikasnosti toka saobraćaja. Konkretni primeri kojima se upravo i potvrđuje raznolikost upotrebe uključuju sistem za povećavanje efikasne potrošnje goriva kod kolona transportnih vozila (eng. *platooning)* [2], razvoj pametnog sistema saobraćaja u gradovima (eng. *Intelligent Traffic System*) [17], aplikacije koje asistiraju vozaču pri vožnji itd. Pored mogućnosti da vozila komuniciraju isključivo međusobno V2V (eng. *Vehicle-to-Vehicle*), postoje takođe pristupi i istraživanja koja se bave mrežom koja podrazumeva i komunikaciju sa okolnom infrastukturom V2I (eng. *Vehicle-to-Infrastructure*), kao i kategoriju V2X (eng. *Vehicle-to-Everything*) koja predlaže povezivanje različitih tipova uređaja i vozila. Infrastruktura, u ovom smislu, ne nameće fizičke resurse neophodne za prenos podataka, već se vezuje za mogućnost daljeg proširivanja kapaciteta same VANET mreže (npr. povećavanjem dometa stizanja paketa ukoliko bi se koristila i posredna komunikacija).

Uz velike mogućnosti koje pruža, VANET u svom razvoju nailazi na različite probleme. Primaran izazov i temu mnogih radova predstavljaju privatnost i bezbednost. Poznata je lista slabosti i napada kao što su DOS, lažiranje GPS pozicije, osluškivanje mrežnog saobraćaja, napad posrednika i predlozi njihovih rešavanja iz ugla enkripcije [22]. Nije moguće sve rešiti boljim principima enkripcije pa predlozi za poboljšanje bezbednosti često uključuju i korišćenje proverenog hardvera. Dalje, postoje mnogi problemi koji se vezuju za sam prenos paketa usled postojanja prepreka (eng. *shadowing*)[11], pojavljivanja zagušenja mreže [9] itd. Iz priloženog se primećuje veliki broj otvorenih pitanja, međutim, sve intenzivnijim razvojem telekomunikacionih standarda, algoritama rutiranja i uključivanjem automobilske industrije, rad na aktuelnim problemima ukazuje na spremnost da se sama VANET tehnologija, pored aktivnog korišćenja u industriji, u skorijoj budućnosti nađe i u široj civilnoj upotrebi.

Bitni standardi su IEEE 802.11p WAVE i Cellular V2X. Prvi standard, IEEE 802.11p WAVE, predstavlja prihvaćeno proširenje standarda IEEE 802.11 (WLAN). U Sjedinjenim Američkim Državama, deo je komunikacionog kanala nazvanog DSRC (eng. *Dedicated Short Range Communications*) zadužen za niže slojeve MAC i PHY. Drugi standard, Cellular V2X, koristi 4G LTE i 5G mrežu [32] za prenos podataka.

Evropska komisija je smisleno odbila da protokol vezuje za WiFi u želji da ostane neograničen [10]. Analiza performansi C-V2X protokola prikazana je u [13].

Glava 2

Računarski VANET sistem zadužen za asistenciju pri vožnji

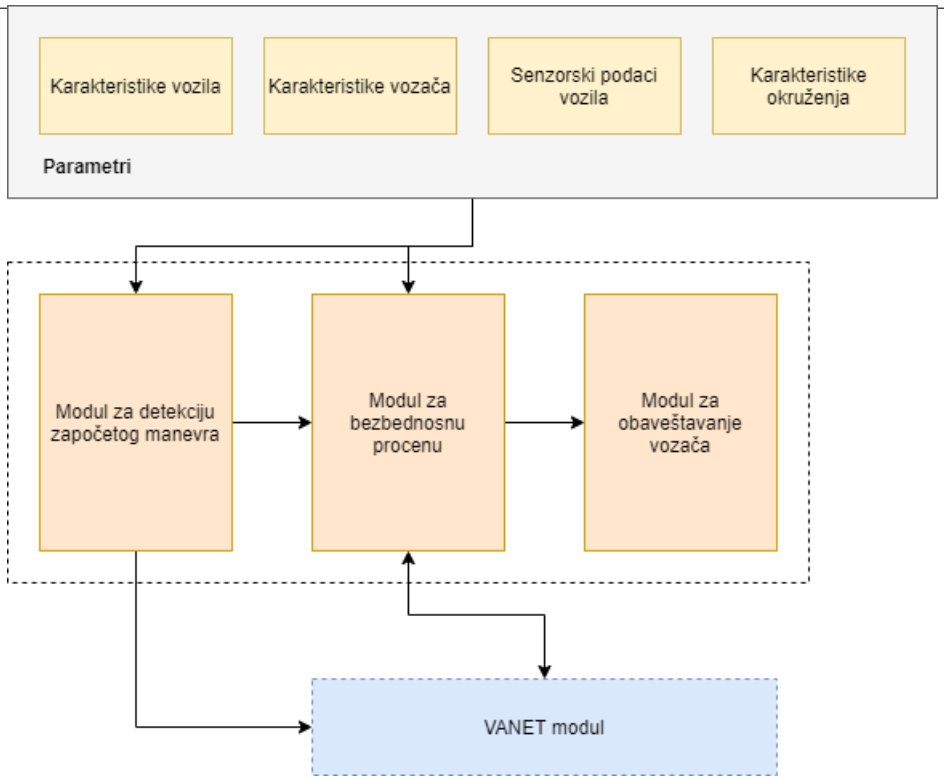
U daljim sekcijama, fokus će biti na primeni VANET mreže u jednom izdvojenom manevru, odnosno preticanju na putevima sa dve trake. Međutim, kako bi se razumeo kontekst komponente razvijene za svrhe rada, napravljena je osnovna modularizacija čitavog računarskog sistema koji bi svojom fleksibilnošću mogao da bude primenjen i u dosta većem broju manevara. Dijagram se može videti na slici 2.1.

Moduli, korišćeni u stvaranju osnovnog okruženja za komponentu koja bi upozoravala vozača na neželjeni ishod akcije, jesu:

1. *Modul za detekciju započetog manevra* - odgovoran je za prepoznavanje potencijalne započete akcije na osnovu trenutnog stanja saobraćaja u lokalnom okruženju. U slučaju preticanja, uključenje signalizacije i tip puta na kome se vozilo nalazi predstavljaju osnov za pretpostavku da se radi o preticanju.

2. *Modul za bezbedosnu procenu* - centralni modul, zadužen za konstrukciju bezbednosne ocene svih mogućih ishoda započete procedure. U radu su obrađena dva pristupa implementaciji algoritama specijalizovanih za proceduru preticanja koji su u srži ovog modula. Izlaz će se direktno koristiti za obaveštavanje vozača o tome koji ishod bi trebalo da izbegne.

3. *Modul za obaveštavanje vozača* - izlazni modul celog računarskog sistema. Njegov cilj je blagovremeno i nenametljivo upozoravanje vozača o fizičkoj neizvodljivosti pojedinih ishoda započetog manevra.



Slika 2.1: Konceptualni pregled modularizacije VANET računarskog sistema za asistenciju pri vožnji

4. *VANET modul* - koristi se kao izlazno-ulazna tačka za komunikaciju sa drugim vozilima. Šalje parametre koji bi drugima bili od značaja i prikuplja sve pakete koji su u radijusu definisanim protokolom koji se koristi.

Na priloženom dijagramu može se takođe primetiti i sekcija koja opisuje parametre. Nije modul za sebe, ali je prikazana zbog svoje intenzivne komunikacije sa drugim modulima i zbog boljeg sagledavanja različitosti parametara koji bi mogli da se koriste od strane različitih procedura, pa i modula za bezbednosnu procenu pri njihovom započinjanju. U kontekstu šire upotrebe računarskog sistema, svaki tip podataka koji bi predstavljali moguća proširenja implementiranog sistema iz rada zavređuje pažnju, tako da neće biti većeg ulaska u dubinu pojedinačnih stavki. Od parametara, u daljim sekcijama koristiće se isključivo fizički parametri u vidu brzine, ubrzanja, traka u kojima se vozila nalaze, kao i njihovih lokacija.

Srž samog rada, koji opisuje predlog računarskog sistema za asistenciju pri preticanju na dve trake, leži upravo u implementaciji specijalizovanog *modula za bezbednosnu procenu* koji na osnovu trenutne situacije u saobraćaju može da proceni

da li je započeto preticanje opasno ili bezbedno. Kako bismo videli šta taj modul može da očekuje u vidu scenarija koji obrađuje, prvo ćemo opisati detaljnije šta je to što preticanje kao manevar podrazumeva, ali i to koji su rezultati dosadašnjih istraživanja koja su se bavila upravo ovom temom.

2.1 Pregled manevra preticanja na putevima sa dve trake i dosadašnji radovi

Do sada je spomenuto više postojećih upotreba VANET mreže kojima je svrha uglavnom pospešivanje performansi toka saobraćaja ili efikasnija upotreba resursa pri vožnji. Pored njih dosta pažnje je usmereno i na povećanje bezbednosti saobraćaja.

Motivacija - Scenario koji svojom opasnošću privlači razna rešenja jeste upravo preticanje na putevima sa dve trake. U Nemačkoj se, na primer, samo u 2014. desilo 73.916 nesreća sa fizičkim povredama na ruralnim putevima [28]. Od svih njih, broj poginulih osoba je 2.019, a dodatno je 25.971 rezultiralo teškim telesnim povredama. Ukupno se oko 6% nesreća desilo pri preticanju, a za njih se vezuje 9% svih fatalnih ishoda i ozbiljnih povreda. To svrstava preticanje na putevima sa dve trake u izrazito opasan manevar. Zaključak rada je da se nesreće pri preticanju najčešće dešavaju na deonicama na kojima je manevar dozvoljen, ali gde je preglednost putanje umanjena. Ideja ovog rada i jeste da se računarskim sistemom, koji se zasniva na bežičnim mrežama vozila nezavisne vizuelne preglednosti, predvidi ishod započetog manevra preticanja.

Aktuelna istraživanja - Sistem za asistenciju pri preticanju zasnovan na bežičnim mrežama nazvan još i VAOAS (eng. *VANET-based adaptive overtaking assistance*) nailazi na različite implementacije i uglove razvoja [4]. Između ostalog, istraživači su se posvetili i „elegantnom" primećivanju započetog preticanja u slučaju nepostojanja povezanosti računarskog sistema sa signalizacijom, a to su učinili tako što su posmatrali promenu ugla volana u određenom vremenskom intervalu [20]. O uključivanju dodatnih parametara poput temperamenta vozača, njegove veštine vožnje, stanja na putu, kao i o pokušajima podrške različitih šablona preticanja mogu se takođe naći podaci u stručnim radovima [24]. Postojeći radovi su uglavnom

okrenuti broju pokrenutih simulacija i podešavanju sistema.

2.2 Ciljevi rada

Posmatranjem postojećih radova, može se videti da su mnoge teme načete, kao i to da upotreba VANET mreže napreduje, delom sve većim brojem primena, a delom i unapređenjima postojećih pristupa. Imajući u vidu navedeni rast industrije, u tom neprekidnom napretku svakako postoji i veliki prostor za poboljšanja i kombinovanje rezultata različitih radova. U nastavku, biće izloženi ciljevi razvoja računarskog sistema opisanog u daljim sekcijama kao i nadogradnje simulacionog okruženja, koji pokušavaju da odgovore na neka od otvorenih pitanja i ponude potencijalna unapređenja.

Glavni ciljevi rada:

1. **Definisanje nacrta šireg sistema za asistenciju pri vožnji** - većina radova fokusirana je isključivo na manevar preticanja, bez osvrtanja na kontekst u kom bi modul za taj izolovani manevar radio. U ranijem delu ove sekcije, napravljen je osnovni koncept u kome razvijeni algoritam funkcioniše. Pozitivna strana ovog pristupa je otvaranje teme kombinovanja algoritama za asistenciju u različitim manevrima kao i postepeno uključivanje mešovitih simulacija i testova u budućnosti.

2. **Realistični scenariji** - Realističnost scenarija je bitna stavka dobrog testa rada sistema. Međutim, nedovoljno pažnje se pruža upravo fizičkoj realističnosti puta. Simulacioni scenariji uglavnom svoju raznovrsnost nalaze u promeni konfiguracije vozila ili stanja puta poput klizavosti, kvaliteta i slično. Upravo zbog želje da se uvede ta nepredvidivost koju oblik puta može da pruži, veoma bitan motiv rada je adaptacija realnih puteva u simulacione scenarije i olakšavanje tog čitavog procesa, automatizacijom, kako bi uz minimalno znanje o formatu čuvanja puteva u simulatoru, novi scenario mogao biti ubačen i od strane lica koje nije upoznato sa programiranjem.

3. **Nepostojanje eksplicitne zavisnosti od konkretnog protokola** - U trenutnoj fazi tranzicije ka primeni, u kojoj se VANET nalazi, vezivanje za jedan jedini protokol nikako nije dobra odluka. U proteklom periodu, neki protokoli

su izgubili poverenje, dok su neki poput C-V2X postali uzdanice. U kontekstu rada, bitno je da korišćen simulator pruža integraciju ka većem broju protokola, kao i da sam modul za bezbednosnu procenu napisan u aplikativnom sloju ne treba da bude svestan nižih slojeva.

4. **Podrška za preticanje dva vozila** - U praksi preticanje dva vozila nije retko, pa je podrška za odabir ishoda koji ima najbolji odnos bezbednosti i efikasnosti u kojima postoji do dva vozila od najveće važnosti. To je još jedan aspekt čiji je cilj doprinos realističnosti scenarija upotrebe.

5. **Stvaranje paralele između predloženih algoritama za procenu bezbednosti pri preticanju** - Još jedan faktor koji može uticati na preciznost rada algoritma je strogost poštovanja granica njegovih proračuna. U radu su predložena, implementirana i testirana dva pristupa koji na osnovu istih ulaznih parametara pokušavaju da što bolje ocene moguće ishode pri preticanju.

Glava 3

Predlog algoritma koji asistira pri preticanju na putevima sa dve trake

3.1 Korišćena notacija

U daljem tekstu koristićemo sledeće skraćenice radi konciznosti i bolje preglednosti:

- *ov* - vozilo koje započinje preticanje (eng. *overtaking vehicle*);
- *ot* - vozilo koje *ov* pretiče (eng. *overtake target*);
- s_2 je bliže, a s_1 dalje vozilo ispred *ov* iz iste trake;
- *op* - vozilo iz suprotne trake u odnosu na *ov* (eng. *opposite lane vehicle*);
- v^{veh} - brzina pojedinačnog vozila;
- t - vreme potrebno da se izvrši neki neka akcija;
- s - dužina deonice puta koja se vezuje za neku određenu fazu toka preticanja;
- a - ubrzanje vozila *ov*;
- l^{veh} - gps lokacija vozila;
- $euclidean(l^{veh_1}, l^{veh_2})$ - funkcija koja izračunava Euklidovo rastojanje između dve zadate lokacije, gde važi da su veh_1 i veh_2 iz istog skupa $\{ov, ot, op, s_2, s_1\}$ i $veh_1 \neq veh_2$.

Kod navedenog nabrajanja važi $veh \in \{ov, ot, op, s_2, s_1\}$. Pored opštih pravila koja su prikazana, na mestima gde je potrebno korišćene su i dodatne varijacije koje su posebno naglašene i opisane u kontekstu upotrebe. Obično su nazivi promenljivih koje se ne vezuju samo za jedno vozilo izabrani tako da svojim imenom opisuju kontekst upotrebe (npr. relativna brzina vozila *ov* u odnosu na *ot* nazvana je $v^{overtake}$ jer je to realna brzina kojom će se odvijati samo preticanje).

3.2 Opis algoritma

Najpre ćemo postaviti okruženje u kom će se algoritam pozivati kao i pretpostavke vezane za obrađeni model preticanja. S obzirom na to da će se oba konkretna pristupa razvijena u radu zasnivati na istim postavkama, u nastavku je reč algoritam pri opisivanju korišćena u jednini, tako da govorimo o algoritmu a ne o algoritmima. Osnovni algoritam zadužen za vršenje procene uspešnosti započetog preticanja, predstavlja minimalistički okvir koji se oslanja samo na brzine kretanja vozila iz neposredne blizine i njihove lokacije.

Pretpostavke, koje se odnose na obuhvaćene scenarije, jesu:

- Podržan je model preticanja na putevima sa dve trake koje idu u različitim smerovima.
- Za jedno vozilo proverljivo je preticanje najviše dva vozila koja su ispred njega.
- U svakom testnom scenariju, preticanje se proverava nad samo jednim vozilom i sva ostala vozila u istom slučaju ne mogu započeti svoje preticanje.
- Brzina vozila iz drugog smera, kao i onih koja se pretiču, konstantna je.
- Sem vozila koje vrši manevar preticanja, vozila iz druge trake, kao i vozila koja se pretiču, ne postoje spoljašnji akteri (pešaci, biciklisti, kamioni i slično).
- Algoritam podržava tzv. leteće preticanje sa ubrzanjem, kod kog se vozilo *ov* od samog početka scenarija kreće većom brzinom od vozila koje ono pretiče.
- Pretpostavlja se ravnomerno ubrzavanje vozila *ov* sve dok ono ne dostigne svoju maksimalnu predefinisanu brzinu.

- U trenutku u kom se preticanje započinje, nemoguće je znati kolika će biti maksimalna vozila *ov*, pa se u algoritmu koristi pretpostavljena razlika između maksimalne i početne brzine vozila *ov* koja iznosi $20\frac{km}{h}$, odnosno $4.16\frac{m}{s}$ i ona će se u daljem toku rada označavati kao v^{top}.

- U ciljnoj traci (onoj koja je suprotnog smera od kretanja vozila *ov*) u jednom testnom scenariju može postojati samo jedno vozilo (*op*).

- Algoritam kao ulazni parametar ne prihvata informaciju o tome gde je isprekidana traka na putu, odnosno da li je uopšte dozvoljeno preticanje na nekoj deonici puta. Razlog za ovu odluku je to što se algoritam i pokreće u trenutku kad je već započeto preticanje, a na *modulu za procenu bezbednosti pri preticanju* biće samo pokušaj upozorenja vozača ukoliko za tim ima potrebe. Dodatno, parametar poput tipa trake povećava u velikoj meri zahteve algoritma o poznavanju okruženja. U svakom slučaju, ta informacija bila bi potencijalno interesantna kao jedno od proširenja ulaznih parametara.

Iz pretpostavki se vidi da se rigidni algoritam u potpunosti oslanja na informacije dobijene u trenutku kada je preticanje započeto. Dalje, on koristi samo osnovne fizičke vrednosti potrebne da se opiše kretanje: brzina, trenutna GPS lokacija, pravac i smer.

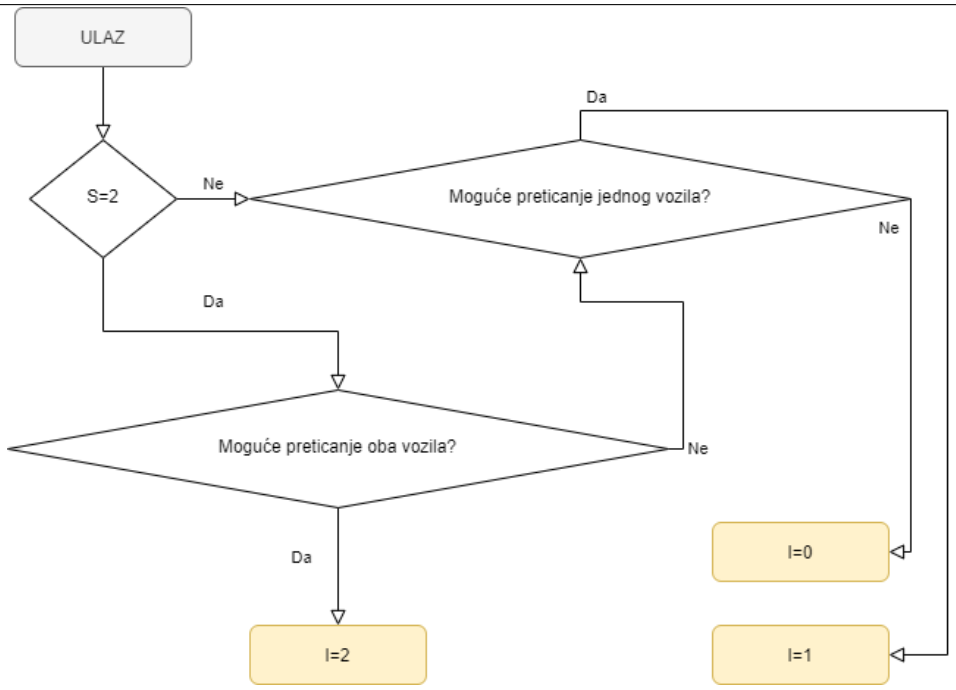
Imajući sve ovo u vidu, u daljem tekstu, pored nadograđenog algoritma, biće kratko opisan i pristup generalizaciji testnih scenarija pri daljem unapređivanju VANET platforme.

3.3 Ulazni podaci

Potrebni parametri, koji predstavljaju stanje vozila u trenutku u kom je započeto preticanje, jesu: (1) brzine vozila koja učestvuju u scenariju $V = [v^{ov}, v^{ot}, v^{op}]$, (2) GPS lokacije vozila koja učestvuju u scenariju $L = [l^{ov}, l^{ot}, l^{op}]$, (3) pretpostavljen kapacitet ubrzanja vozila koje vrši preticanje a i (4) broj vozila iz izvorne trake S.

3.4 Izlaz iz algoritma

Rezultat algoritma je numerička vrednost koja predstavlja broj verifikovanih vozila koje je moguće preteći bezbedno.



Slika 3.1: Konceptualni pregled toka izvršavanja algoritma za procenu bezbednosti započetog preticanja

Razlikujemo tri moguća izlaza:

1. $I = 0$ nije moguće preteći nijedno vozilo
2. $I = 1$ moguće je preteći najviše jedno vozilo
3. $I = 2$ moguće je preteći dva vozila

Kada se vozilo koje vrši preticanje nađe na predodređenoj distanci od najbližeg vozila u istoj traci, sprovodi se provera da li je preticanje moguće, kao i to koliko je vozila moguće preteći. Na slici 3.1 prikazan je tok izvršavanja algoritma.

Ukratko, ukoliko postoje dva vozila ispred nas, potrebno je proveriti da li možemo da preteknemo oba i, ukoliko možemo, verifikovan je slučaj I=2. U slučaju da se po proceni ne mogu bezbedno preteći kola koja su druga po redu u odnosu na nas, potrebno je proveriti da li možemo preteći bar jedno vozilo. Ista provera se vrši direktno i u slučaju da ispred nas postoji samo jedno vozilo. Ukoliko provera pojedinačnog ishoda vrati potvrdan rezultat, izlaz iz algoritma je I=1, a u suprotnom I=0.

Provera bezbednosti preticanja oba vozila lako se svodi na verifikaciju uspešnog preticanja jednog vozila. Koristeći postavljene oznake, ukoliko se ispred vozila *ov*

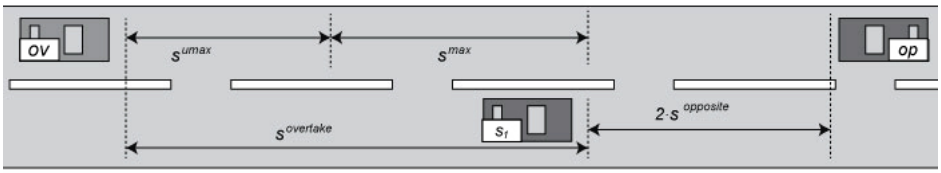
nalaze dva vozila (s_1 i s_2) prvo će se proveriti da li je moguće preticanje vozila s_1. U slučaju da je moguće, smatramo da je scenario preticanja oba vozila procenjen kao bezbedan. Dalje, u slučaju da verifikacija preticanja daljeg vozila nije bila uspešna, proverava se da li je moguće preteći barem ono vozilo koje je bliže (s_2). Scenario u kom se ispred *ov* nalazi samo jedno vozilo je trivijalan, jer je tada ishod jednostavno provera preticanja tog jednog vozila. Sada, kada su oba slučaja svedena na pozivanje jedne iste metode za verifikaciju preticanja jednog vozila, preostaje da se definiše upravo ta metoda.

Prema postavljenom ulazu i izlazu moguće je napisati veći broj algoritama koji bi se trudili da što bolje predvide buduće kretanje svih vozila na osnovu preseka stanja saobraćaja u trenutku početka manevra. U nastavku sekcije opisana su dva različita pristupa koja su za svrhe rada implementirana i testirana u simulatoru. Idejno, njihova osnovna razlika je u tome što se prvi pristup bazira na direktnom izračunavanju da li na osnovu jednostavnih fizičkih parametara vozilo *ov* može bezbedno izvršiti preticanje, a drugi na verovatnoći da do sudara dođe kroz proteklo vreme.

3.5 Rigidni algoritam

U radu je prvi pristup okarakterisan rigidnim jer osnovna ideja na kojoj se on zasniva, ne nudi mnogo fleksibilnosti pri razvoju i njegovom unapređivanju. Kako bi se za određeno vozilo koje se pretiče proverio ishod, izračunava se vreme potrebno *ov* da stigne do pozicije vozila *ot* i vreme potrebno vozilu *op* iz suprotne trake da stigne do iste lokacije. Ukoliko vozilo *op* stigne do te lokacije pre *ov* ishod je sudar, a u suprotnom se smatra da je *ov* uspeo da stigne i vrati se u svoju traku. U konkretnoj implementaciji svakako neće biti dovoljno samo da *ov* stigne *ot*, već i da dodatno pređe predefinisano rastojanje (koje odgovara stizanju ispred vozila koje se pretiče) kako bi zapravo uspešno dovršio manevar. Za izračunavanje razdaljine između vozila, korišćeno je Euklidovo rastojanje. U procenjeno vreme dostizanja pozicije *ot* uzimano je u obzir i ubrzanje. Konceptualni prikaz navedenih rastojanja dat je na slici 3.2.

Ideja i tok rigidnog algoritma - prikaz pristupa izračunavanju rigidnog algoritma priložen je u pseudokôdu 3.3. Prvo se definišu brzina kojom se *ov* kreće u odnosu na vozilo koje pretiče *ot*, odnosno $v^{overtake}$ i distanca koju je potrebno preći tom brzinom da bi se preticanje izvršilo $s^{overtake}$. Na primer, ukoliko se vozilo *ov*



Slika 3.2: Rastojanja korišćena pri proceni uspešnosti preticanja

kreće brzinom $80\frac{km}{h}$, a vozilo *ot* $70\frac{km}{h}$, dok je njihova početna udaljenost $30m$, možemo oceniti da će preticanje biti dovršeno kada vozilo *ov* tu distancu pređe krećući se brzinom $10\frac{km}{h}$. Oznakom *umax* opisane su veličine koje opisuju fazu preticanja dok se još nije ostvarila predefinisana maksimalna brzina, dok se *opposite* vezuje za promenljive koje opisuju udaljenost vozila koje se pretiče (*ot*) i onog koje je u suprotnoj traci (*op*). Udaljenost ($s^{opposite}$) podeljena je sa 2 jer se oba vozila kreću u suprotnom smeru istim brzinama, što znači da će se sresti kada pređu pola distance između njihovih lokacija. Promenljivom v^{avg} označena je prosečna brzina kojom će se *ov* kretati dok ne ostvari maksimalnu brzinu. Dalje, gledamo da li će vozilo *ov* uspeti da ostvari maksimalnu brzinu pre nego što stigne do *ot*. Ukoliko ne uspeva, u računicu nećemo uključiti ubrzanje, već ćemo koristiti brzinu kojom se *ov* kretalo pre ubrzanja. Bitno je napomenuti da će se u simulacionom scenariju ubrzanje odvijati i to da se ono zanemaruje samo u računici kako bismo se više obezbedili jer se tačna brzina kojom će se *ov* kretati ne može lako predvideti u realnom okruženju. U slučaju da *ov* uspeva u fazi preticanja da ostvari maksimalnu brzinu, ukupno vreme preticanja se računa kao suma vremena koje je potrebno da se dostigne maksimalna brzina i onog u toku kog će se *ov* kretati maksimalnom brzinom dok ne stigne do *ot*. Na kraju, ukoliko je vreme potrebno za preticanje ($t^{overtake}$) manje od vremena za koje *op* stiže do *ot*, procenjuje se da će se preticanje uspešno izvršiti. U suprotnom će doći do sudara.

Result: procena da li ov može bezbedno preteći ot

$v^{overtake} \leftarrow v^{ov} - v^{ot}$;
$v^{max} \leftarrow v^{overtake} + v^{top}$;
$s^{overtake} \leftarrow euclidean(l^{ov}, l^{ot})$;
$t^{umax} \leftarrow v^{top}/a$;
$v^{avg} \leftarrow (v^{overtake} + v^{max})/2$;
$s^{umax} \leftarrow t^{umax} * v^{avg}$;
$s^{opposite} \leftarrow euclidean(l^{op}, l^{ot}))/2$;
$t^{opposite} \leftarrow s^{opposite}/v^{op}$;
if $s^{umax} > s^{overtake}$ **then**
    $t^{overtake} \leftarrow s^{overtake}/v^{overtake}$;
else
    $s^{max} \leftarrow s^{overtake} - s^{umax}$;
    $t^{max} \leftarrow s^{max}/v^{max}$;
    $t^{overtake} \leftarrow t^{umax} + t^{max}$;
end
return $t^{overtake} < t^{opposite}$;

Slika 3.3: Pseudokôd koji opisuje ponašanje funkcije checkOvertakingOptionsRigid zadužene za proveru svih mogućih ishoda pri preticanju

3.6 Algoritam zasnovan na verovatnoći

Po svojoj prirodi i modelu na kom se zasniva rigidni algoritam, može se primetiti da jako mala razlika u vrednosti ulaznih parametara može da znači potpuno drugu procenu. Cilj unapređenja bio bi pokušaj da se ishodima daju verovatnoće umesto obaveznog odabira koji će se sigurno desiti a koji ne. Na osnovu tih verovatnoća bi zatim bilo moguće dobrim tempiranjem granice prihvatljivosti ishoda bolje kontrolisati ponašanje algoritma i njegovu preciznost u scenarijima u kojima nije lako odlučiti se za jednu opciju.

Od rigidnog pristupa razlikuje ga samo telo metode za odlučivanje, pa će poglavlje uglavnom biti koncipirano oko modela izračunavanja. Testiranje će se vršiti nad istim scenarijima nad kojima je proveravan i rigidni algoritam, pa su i ovde korišćeni isti parametri i notacija kao i u ranijim sekcijama.

Ideja i tok algoritma - cilj je pretočiti logiku iza odgovora na pitanje da li *ov* može dostići distancu dovoljnu za preticanje *ot* pre vozila iz suprotne trake *op* u odgovor na pitanje kolika je verovatnoća dostizanja određene distance bez sudara.

Upravo na oblik te funkcije koja povećava verovatnoću sudara kroz vreme koje protiče uticaće ulazni parametri. Za svrhe rada i razvijeni algoritam korišćena je polinomijalna funkcija šestog stepena i formula:

$$collisionProbability(t) = (t \times \frac{1}{t^{collision}})^6$$

Bitno je napomenuti da je oblik funkcije svakako opširna tema koja zahteva poseban napor i pažnju, a navedena formula je osmišljena kako bi mogla da se na unapređenom simulacionom okruženju testiraju dva idejno različita algorima koji dobijaju iste ulazne podatke.

Promenljiva $t^{collision}$ izračunava se kao vreme potrebno da vozila *ov* i *op* stignu jedno do drugog. Ulazni parametar t predstavlja vreme koje je potrebno da vozilo *ov* prestigne *ot*. Rezultat predstavlja verovatnoću sudara pre ili u trenutku stizanja do vozila *ot*. Vrednosti navedene funkcije mogu biti i veće od broja 1 (što označava siguran sudar) pa se u tom slučaju zaokružuju na 1. Motiv za korišćenje priložene funkcije jeste želja za testiranjem jednostavne funkcije koja bi ipak imala relativno ubrzano povećanje verovatnoće sudara kako se vozilo *ov* približava *op*. Primer oblika funkcije *collisionProbability* u kom je procenjeni sudar u 12. sekundi prikazan je na 3.4.

Jedini preostali deo podešavanja jeste granica verovatnoće koja je prihvatljiva za preuzimanje rizika. Ta granica se konfiguriše u vidu promenljive nazvane *acceptabilityThreshold* kojoj je vrednost dodeljena u radu jednaka 0.2.

Pseudokôd koji opisuje principe rada algoritma zasnovanog na verovatnoći prikazan je u 3.5. Kao i kod rigidnog algoritma, oznaka *umax* vezuje se za vrednosti faze u kojoj *ov* još nije ostvarilo svoju maksimalnu brzinu (nakon ubrzanja). Pri izračunavanju dužine puta koja je potrebna za završetak ubrzanja do maksimalne brzine, korišćena je srednja vrednost početne i maksimalne brzine, kako bi se dobila dovoljno dobra aproksimacija (iako će pravo ubrzanje biti prepušteno simulacionom okruženju koje nije lako predvidivo). Izračunavamo kombinovanu brzinu v^{colavg} uzimajući srednju vrednost početne i maksimalne brzine za onu kojom se kreće *ov*. Cilj je, dakle, izračunavanje vremena do sigurnog sudara. U uslovu se može primetiti da, ukoliko je vreme potrebno da se ostvari maksimalna brzina vozila *ov* veća od onog za koje će se desiti sudar kada se vozilo *ov* kreće prosečnom brzinom, ne koristi se

Slika 3.4: Primer funkcije *collisionProbability* za scenario u kom su *ov* i *op* na procenjenih 12s vožnje. U 12-toj sekundi sudar je neminovan pa je vrednost funkcije tada 1, a na samom početku scenarija je 0. Tačka B označava trenutak u kom je vrednost funkcije, odnosno verovatnoća sudara jednaka 0.2 što je konfigurisano promenljivom *acceptabilityThreshold*

ubrzanje pri izračunavanju. U suprotnom, slično kao u rigidnom algoritmu računa se ukupno vreme ubrzanja i vremena koje će se voziti maksimalnom brzinom do sudara. Sada kada imamo vreme do sigurnog sudara ($t^{collision}$), preostaje samo da se izračuna i vreme potrebno za preticanje ($t^{overtake}$) i da se te dve vrednosti iskoriste u formuli koja predstavlja osnovu navedenog algoritma kako bi se dobila procena verovatnoće sudara u trenutku kada se preticanje završava. Ukoliko verovatnoća odgovara predefinisanom pragu rizika, procenjuje se da će se preticanje uspešno izvršiti, a u suprotnom da će doći do sudara.

Result: procena da li ov može bezbedno preteći ot u skladu sa predefinisanim pragom prihvatanja rizika *acceptabilityThreshold*

$v^{max} \leftarrow v^{ov} + v^{top}$;
$t^{umax} \leftarrow (v^{max}$ - $v^{ov})$ / a;
$v^{avg} \leftarrow (v^{max} + v^{ov})/2$;
$s^{umax} \leftarrow t^{umax} * v^{avg}$;
$s^{collision} \leftarrow euclidean(l^{ov}, l^{op})$;
$v^{colavg} \leftarrow v^{avg} + v^{op}$;
$t^{colavg} \leftarrow (s^{collision}/v^{colavg})$;
$t^{collision} \leftarrow 0$;
if $t^{umax} \geq t^{colavg}$ **then**
    $v^{collision} \leftarrow v^{ov} + v^{op}$;
    $t^{collision} \leftarrow s^{collision}$ / $v^{collision}$;
else
    $s^{maxcol} \leftarrow s^{collision}$ - s^{umax};
    $v^{maxcol} \leftarrow v^{max} + v^{op}$;
    $t^{max} \leftarrow s^{maxcol}/v^{maxcol}$;
    $t^{collision} \leftarrow t^{umax} + t^{max}$;
end
$v^{overtake} \leftarrow v^{ov} - v^{ot}$;
$v^{maxov} \leftarrow v^{max} - v^{ot}$;
$s^{overtake} \leftarrow euclidean(l^{ov}, l^{ot})$;
$t^{overtake} \leftarrow 0$;
if $s^{umax} > s^{overtake}$ **then**
    $t^{overtake} \leftarrow s^{overtake}/v^{overtake}$;
else
    $s^{maxov} \leftarrow s^{overtake}$ - s^{umax};
    $t^{max} \leftarrow s^{maxov}/v^{maxov}$;
    $t^{overtake} \leftarrow t^{umax} + t^{max}$;
end
$collisionProbability \leftarrow pow((t^{overtake} * (1/t^{collision})), 6)$;
return $collisionProbability < acceptabilityThreshold$;

Slika 3.5: Pseudokôd koji opisuje algoritam za procenu uspešnosti preticanja jednog vozila koji je zasnovan na verovatnoći

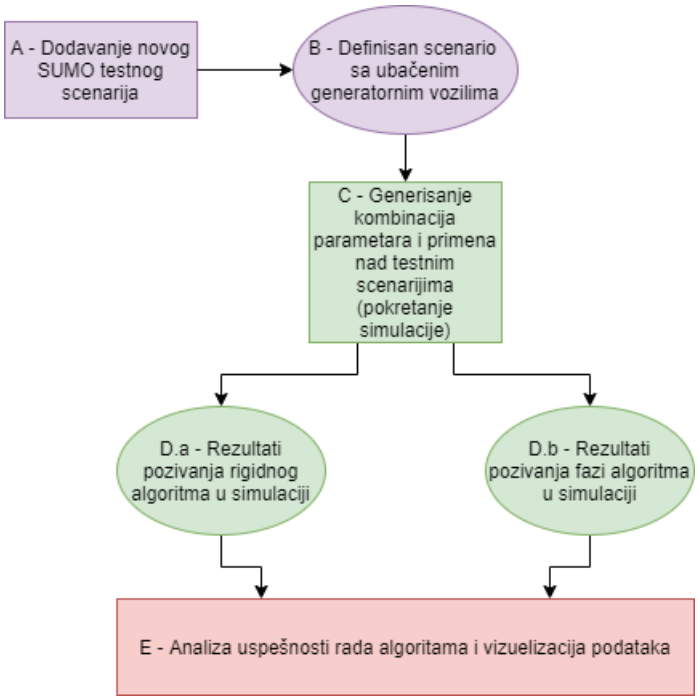
Glava 4

Simulacija predloženih algoritama i rezultati

Do sada, u ranijim poglavljima, opisana je situacija u saobraćaju koja je u radu podvrgnuta testiranju. Naredni korak biće opisivanje implementiranog sistema za simuliranje i proveru tačnosti rada algoritma koji testiramo. Na početku ćemo se osvrnuti na alate koji se koriste, a zatim i na stvaranje jednog u potpunosti novog modula za generisanje realističnih simulacionih scenarija. Na kraju će se vršiti pregled rezultata rada rigidnog i algoritma zasnovanog na verovatnoći. Kako bi se, pri prolasku kroz sekciju, u svakom trenutku mogla imati cela slika koraka potrebnih za uspešnu simulaciju kao i njihova povezanost, priložen je na 4.1 dijagram koji grubo prati sve podsekcije.

Dodatno, bitno je spomenuti da, iako je za svrhe rada bio potreban samo razvoj simulacionih scenarija za situaciju preticanja na putevima sa dve trake, uz manji napor promene izolovanog dela generatora simulacije, opisani sistem bi mogao da se primeni u mnogim drugim algoritmima koji bi se koristili u različitim situacijama iz saobraćaja.

Postavljanje realističnog simulacionog okruženja i automatizacija uvođenja novog scenarija ubrajaju se među glavne ciljeve rada, pa je ceo tok detaljno opisan. Posebna pažnja usmerena je na opisni deo implementiranih skripti, bez ulaženja u duboke tehničke detalje, jer od prirode preuzimanja testnih scenarija zavisi kako kvalitet testiranja algoritama tako i budući potencijal za proširenja.

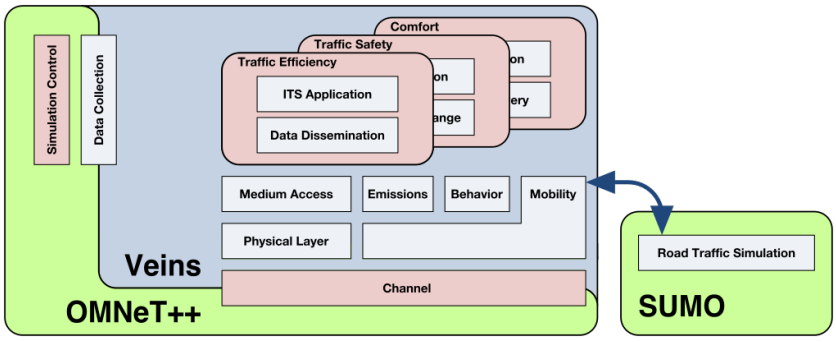


Slika 4.1: Struktura simulatora

4.1 Alati

Alat korišćen za simulaciju VANET mreže je Veins [30]. Unutar sebe Veins kombinuje simulator saobraćaja SUMO [19] i modul OMNeT++ [31] koji daje mogućnost stvaranja realističnih mreža čvorova. Aplikativni kôd je pisan u programskom jeziku C++ u sekciji koda rezervisanoj za to. Ovaj rad u sebi uključuje pozive SUMO simulatora sa različitih mesta pa je modularni prikaz odnosa između komponenti Veins projekta prikazan na slici 4.2, preuzetoj sa web sajta Veins projekta u svrhu razgraničavanja funkcija korišćenih delova platforme.

Veins - projekat čiji je kôd otvorenog tipa, nudi podršku za veliki broj protokola kao što su IEEE 802.11p, IEEE 1609.4 DSRC/WAVE i ARIB STD-T109. Postoji veliki broj svojstava koja čine jako realističnim testiranje mrežnih sistema, kao i računarskih sistema koji unutar sebe koriste podržane protokole. Neki od tih dodataka su Obstacle Shadowing koji implementira model slabljenja mrežnog signala pri prolasku kroz zgradu i Vehicle Obstacle Shadowing koji primenjuje model gubljenja jačine signala uzrokovanog vozilima.



Slika 4.2: Modularni prikaz Veins sistema

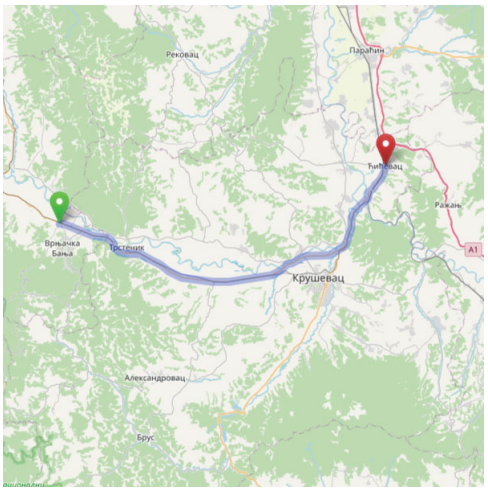
OMNeT++ - C++ biblioteka i projekat koji se koristi za simuliranje mrežnih sistema, nudi podršku za testiranje kako bežičnih tako i mreža koje koriste kablove kao fizičke medijume prenosa paketa, ali i za test optičke mrežne infrastrukture. Takođe, nudi dodatke za integraciju sa bazama podataka, sisteme za rad u realnom vremenu itd.

SUMO - simulator saobraćaja koji omogućava rad sa velikim mrežama kao i rad sa različitim tipovima aktera u saobraćaju (npr. automobili, kamioni, pešaci). Nudi interfejs za kontrolu i upravljanje simulacije kroz TraCI (Traffic Control Interface) koji je prilagođen za različite programske jezike, uključujući: Python, Javu, .NET, C++ i Matlab. U ovom radu TraCI je uglavnom korišćen za kontrolisanje svojstava vozila unutar simulacije i njihovo kreiranje.

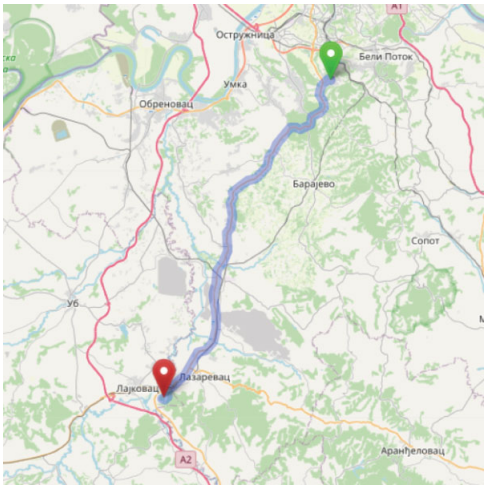
4.2 Testni scenariji

Jedan od bitnih faktora koji utiču na procenu koliko će se precizno algoritam ponašati u realnom svetu jeste odabir testnih scenarija nad kojima se vrši simulacija i primena algoritma. Kako bi ti scenariji bili što realističniji, pristupanjem javno dostupnoj bazi podataka Agencije za bezbednost saobraćaja Republike Srbije [33], odabrane su deonice puteva koje se mogu smatrati opasnim. Konkretno, to su deonice puteva u Srbiji na kojima se od 2015. do 2019. dogodilo preko 20 nezgoda u kojima je ishod nesreće formalno bio: “poginuo na licu mesta”, “preminuo za vreme prevoza do zdravstvene ustanove”, “naknadno preminulo lice u roku od 30 dana od posledica saobraćajne nesreće” ili “teške telesne povrede”.

Na osnovu spomenutih kriterijuma, za potrebe rada izdvojene su sledeće deonice:



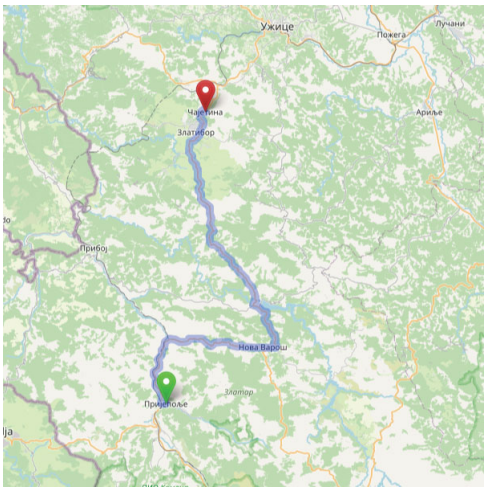
Slika 4.3: Novo Selo - Ćićevac



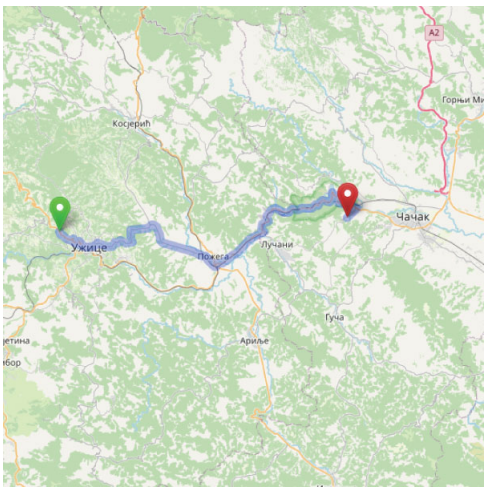
Slika 4.4: Rušanj - Lajkovac

1. Novo Selo - Ćićevac (oznaka puta E-761),
2. Rušanj - Lajkovac (oznaka puta E-763),
3. Prijepolje - Čajetina (oznaka puta E-763),
4. Dubci – Pakovraće (oznaka puta E-763).

Kako bi se dobio bolji uvid u to kako se pristupilo parametrizaciji scenarija i konkretnom obliku puteva koji su korišćeni, priložene su njihove slike preuzete sa *OpenStreetMap* veb aplikacije [1].



Slika 4.5: Prijepolje - Čajetina



Slika 4.6: Dubci – Pakovraće

Format datoteke u kojoj alat SUMO čuva informacije o putevima, uključujući i njihovu podeljenost na najmanje tehničke celine (ivice), ima ekstenziju *net*. Za stvaranje SUMO simulacije na osnovu realnog puta korišćene su ugrađene skripte pridodate alatu, koje pružaju mogućnost da se sa mape definisane projektom *OpenStreetMap* odabere veći pravougaoni region koji se može izvesti u *net* datoteku.

Nakon preuzimanja karte željenog regiona, radi pokrivenosti što većim brojem testova, put nad kojim će se vršiti simulacija biće podeljen na manje delove. Zatim, posle izvlačenja unutrašnjih sekcija puta, svaka od njih koja u sebi sadrži tri ili više ivica (po *net* podeli) biće korišćena za jednu konkretnu grupu testova. Za potrebe stvaranja konkretnih ruta na osnovu niza svih ivica ciljanog puta, napisane su python

skripte koje su priložene uz rad.

4.3 Parametrizacija simulacionog scenarija

U prethodnom koraku opisana je metoda kojom se put nad kojim želimo da testiramo algoritam (testni scenario) može podeliti na najmanje smislene celine nad kojima je moguće pokrenuti simulaciju preticanja. Sada se može posmatrati jedna od tih određenih najmanjih deonica kako bi se definisao drugi korak kome je cilj da obuhvati što veći broj različitih konkretnih scenarija preticanja koje ćemo nazivati simulacioni scenario.

Sada ćemo idejno opisati kako se simulacioni scenario odvija u fazi kada se on u simulatoru tek postavlja. Prvo se na testnu deonicu pušta vozilo koje se pretiče, a zatim se u različitim trenucima njegovog kretanja u simulaciji stvaraju svi preostali akteri. Na taj način se reguliše razdaljina između aktera u različitim scenarijima. Zbog svog zadatka, prednje vozilo koje se pretiče naziva se i *generatorno vozilo* sa oznakom *gen*. Vreme unutar simulatora označeno je kao *simtime*. Sada možemo uvesti sve parametre koji identifikuju jedan scenario.

Ukratko, jedan simulacioni scenario odgovara jednoj kombinaciji vrednosti parametara simulacije. Parametri uzeti u obzir za svrhe rada jesu:

1. Početna brzina *vInit* - opisuje brzinu u kojoj vozilo koje pretiče (*ov*) ulazi u prostor simulacije;

2. Potencijal ubrzanja a - postavlja se kao SUMO parametar na vozilu koje pretiče, vrednosti opsega preuzete su iz posmatranja prosečnih brzina ubrzanja za putničke automobile iz [8];

3. Udaljenost vozila *ov* od prvog sledećeg iz iste trake u trenutku početka preticanja *ovInitialDistance* - parametar koji reguliše koliko je potrebno da se vozilo *ov* približi vozilu ispred sebe da bi započelo proceduru preticanja. Ubrzanje vozila počinje u trenutku kada se napusti početna traka pa parametar utiče na samu procenu algoritma kao i na ishod;

4. Trenutak simulacije (*simtime*) u kom se vozilo s_2 stvara *s2Init* - simulator predviđa testove u kojima se pretiče jedno, ali i dva vozila. U slučajevima kada su oba vozila uključena, *s2Init* se odnosi na vreme stvaranja s_2 vozila u simulaciji nakon generatornog vozila, izraženo u milisekundama;

5. Trenutak simulacije (*simtime*) u kom se vozilo *ov* stvara *ovInit* - simulaciono vreme u kom se stvara *ov* vozilo. U situaciji u kojoj se pretiču dva vozila, ne uzima vrednost iz celog opsega već iz dela opsega vrednosti većih od vremena *s2Init* unutar te kombinacije jer se mora stvoriti nakon njega kako bi bilo iza njega u simulaciji;

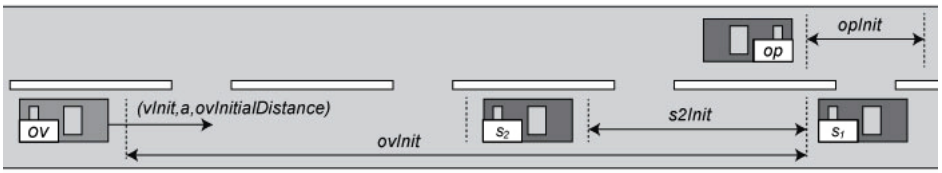
6. Trenutak simulacije (*simtime*) u kom se vozilo *op* stvara *opInit*.

Svi navedeni parametri koji će se koristiti u simulaciji prikazani su na slici 4.7. Najpre ćemo detaljnije opisati priloženu sliku. Vozila s_1 i s_2 kreću se istom trakom i istim smerom kao i vozilo koje vrši preticanje. Parametri, koji se vezuju isključivo za ponašanje vozila koje vrši preticanje, jesu brzina (*vInit*), ubrzanje (a) i udaljenost od vozila, odnosno blizina vozilu ispred koja je potrebna da bi se započelo preticanje ($ovInitialDistance$). Dalje, distanca između dva vozila u opsegu vrednosti parametara definiše kao razlika u generalnom vremenu simulatora (*simtime*) u trenucima stvaranja jednog vozila i drugog čiju razdaljinu opisujemo. Taj pristup je u osnovi jednak direktnom definisanju razdaljina, jer, uz znanje o tome koliko se brzo kreću kola, koliko jedan korak simulacionog vremena traje u sekundama i kada se ko generisao na mapi koja podržava gps lokacije, može se lako izračunati i fizička udaljenost. U smislu generisanja scenarija, svi trenuci stvaranja vozila definišu se kao vreme koje je prošlo od simulacionog vremena stvaranja vozila s_1, jer je ono hronološki prvo koje se pojavljuje na deonici pri pokretanju simulacije. Opsezi vrednosti navedenih parametara nabrojani su u propratnoj tabeli 4.1.

Može se primetiti na osnovu mogućih vrednosti parametara da je broj svih kombinacija koje se mogu koristiti za jednu deonicu jednak 2.430. Parametri vezani za distance između vozila koja učestvuju zavise od dužine deonice scenarija, pa će u praksi broj kombinacija koje će se iskoristiti u vidu simulacija biti nešto manji.

Procesor korišćen za pokretanje simulacija je Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, 1800 Mhz, 4 Core(s), 8 Logical Processor(s) koji je u proseku izvodio do 900 simsec/sec. Za potrebe rada i zahtevanu preciznost procene nad svakim testnim putem predviđeno je pokretanje ukupno do 2.565 simulacija nad svakom od 5 deonica koje su generisane iz testnog puta. To je ukupno 15.390 simulacija za sve testne scenarije. Konfiguracija simulacije se veoma lako može promeniti tako da obuhvata veći broj deonica i različite početne parametre.

Kako bi deonice generisane za testni put odgovarale zahtevima rada razmatrane su one deonice čija je dužina između 300 i 500 metara. Do tih vrednosti, odno-



Slika 4.7: Parametri koji definišu jedan simulacioni scenario.

sno smislenog odnosa dužine minimalnih deonica i odgovarajućih opsega vrednosti parametara, došlo se eksperimentalnim putem jer se u slučajevima gde je put bio veoma kratak ili veoma dugačak dešavalo ili da scenario ne uspe dovoljno dugo da se odvija da bi došlo do preticanja ili je od velikog broja simulacionih vrednosti parametara jako malo kombinacija uključivalo preticanje jer bi se svi akteri usled velikih udaljenosti uglavnom mimoišli, redom.

Naziv parametra	Najmanja vrednost parametra	Najveća vrednost parametra	Korak	Mera
vInit	70	90	10	$\frac{km}{h}$
a	0.5	2.5	1	$\frac{m}{s^2}$
ovInitialDistance	7	11	2	m
s2Init	300	900	300	ms
ovInit	500	3000	500	ms
opInit	0	4000	1000	ms

Tabela 4.1: Parametri koji se koriste za generisanje grupa simulacionih scenarija nad zadatom putanjom.

4.4 Simulacija VANET-a i implementacija mrežnog sloja

U prethodnim poglavljima opisan je uglavnom rad sa saobraćajnim aspektom simulacionog okruženja, odnosno sa vozilima, putevima i njihovim svojstvima. Pre

ulaska u detalje implementacije scenarija preticanja neophodno je opisati mrežni sloj na kom se sistem zasniva. U platformi za testiranje računarskog sistema za asistenciju pri preticanju, ne postoji eksplicitna zavisnost od korišćenog protokola pa je parametrizacija tog dela konfiguracije mrežnog sloja veoma poželjan i smislen potez u daljem razvoju računarskog sistema, kao i platforme. Dodatna olakšica u prilagođavanju testnog okruženja drugim protokolima jeste i činjenica da Veins podržava relativno jednostavno uključivanje dodatnih paketa za standarde poput C-V2X i VLC. Za svrhe testiranja algoritama u radu, korišćen je IEEE 802.11p koji je i podrazumevani protokol u Veins simulatoru. U nastavku biće opisani delovi kôda koji sa aplikativnog sloja prosleđuju podatke nižim mrežnim slojevima u vidu stvaranja paketa.

Princip slanja paketa kroz mrežu - za očekivani rad modula za bezbednosnu procenu pri preticanju potrebno je da svaki akter u saobraćajnoj proceduri ima informacije o drugim vozilima. Korišćeni model mrežnog prenosa paketa koji omogućava oglašavanje informacija svima koji su u dometu pošiljaoca naziva se *beaconing*.

Beaconing - Paketi (eng. beacons) neprekidno se šalju kroz mrežu. Više o efikasnosti slanja paketa beaconing metodom u različitim simulatorima (uključujući OMNeT++) može se pronaći u literaturi [26]. Za komunikaciju između vozila, zadužena je biblioteka OMNeT++ koja je integrisana u Veins simulaciono okruženje. SUMO vozilima se, pri pojavljivanju u simulacionom prostoru, dodeljuje mrežni čvor koji automatski započinje slanje paketa. Deo kôda koji stvara paket prikazan je u kôdu 4.1.

```
1 bsm -> setSenderPos(curPosition);
2 bsm -> setSenderSpeed(std::to_string(curSpeed));
3 bsm -> setSenderDirection(curDirection);
4 bsm -> setSenderID(traciID);
5 bsm -> setSenderLaneID(laneID);
6 bsm -> setPsid(-1);
7 bsm -> setChannelNumber(static_cast < int > (Channel::cch));
8 bsm -> addBitLength(beaconLengthBits);
```

Listing 4.1: Kôd u kom se paket puni informacijama o stanju vozila, ivice kojom se kreće i identifikatorom

Konfiguracija OMNeT++ okruženja za konkretni projekat nalazi se u fajlu omnetpp.ini. Konfigurabilna svojstva uključuju *protok*(u bitovima), *radijus* (u metrima), *jačinu signala* (u megavatima) i razne druge.

4.5 Tok simulacije

Uz sve generisane kombinacije vrednosti parametara simulacije koji će se testirati stvorena je realistična simulacija preticanja. Naredni korak je uključenje poziva algoritma za bezbednosnu procenu uspešnosti preticanja, definisanog u poglavlju 3, u simulaciju. Kako bi se razumeo pseudokôd prikazan u nastavku, potrebno je napomenuti da je unutar Veins simulacije moguće definisati aplikativni modul i unutar njega funkciju koja će biti pozivana za svako vozilo unutar simulacije u svakom novom simulacionom trenutku (*simtime*). Pseudokôd 4.8 prikazuje njeno ponašanje.

Pored upravljanja vozilom, aplikativni sloj je zadužen i za konstruisanje paketa koji će slati svim ostalim vozilima u radijusu propisanom odabranim protokolom. Kako bi paketi koji će se iz aplikativnog modula proslediti infrastrukturi i modulu OMNeT++ zaduženom za niže mrežne slojeve bili upotrebljivi u smislu opisanog računarskog sistema za asistenciju pri preticanju, bilo je potrebno dopuniti postojeće Veins pakete.

```
   Data: simVehicle, simScenario
   Result: stanje vozila simVehicle je ažurno
 1 if outOfBounds(simVehicle) then
 2 |   deleteVehicle(simVehicle);
 3 else
 4 |   if simVehicle.type = 'ov' then
 5 |   |   frontVehicle ← getVehicleInFront(simVehicle);
 6 |   |   if simVehicle.state = 'overtaking' then
 7 |   |   |   updateOvertake(simVehicle);
 8 |   |   else if euclideanDistance(simVehicle,frontVehicle) <=
   |   |    simScenario.ovInitialDistance then
 9 |   |   |   simScenario.predictedOvertakings ← checkOvertake(simVehicle);
10 |   |   |   log(simScenario.predictedOvertakings);
11 |   |   |   simVehicle.state ← 'overtaking';
12 |   |   end
13 |   end
14 end
```

Slika 4.8: Pseudokôd koji opisuje glavnu petlju aplikativnog sloja simulacije. Funkcija *checkOvertake* označava algoritam za procenu bezbednosti preticanja definisanog u glavi 2.

Uspešnost preticanja

Predviđanje uspešnosti preticanja, pored toga što mora biti precizno po pitanju bezbednosti (procenom koliko najviše automobila može uspešno da se pretekne), ne sme da bude suviše ograničavajuće. Ukoliko su upozorenja preterano oprezna, ceo sistem gubi smisao jer bi moglo da dođe do zanemarivanja notifikacija ili bespotrebnih zagušenja saobraćaja.

Kako bi se za jedan testni scenario proverilo koja je zapravo granica opasnosti, odnosno koja su to sporna kola čije preticanje predstavlja opasnost, početno preticanje je trajalo sve dok se zapravo ne bi desio sudar. Nakon toga bi se na osnovu toga gde se desio sudar, trenutnih pozicija i brzina svih aktera u trenutku sudara, lako procenilo da li bi i koliko vozila uspelo da se pretekne.

Lista događaja za ispis

Kako bi se očuvala dinamična priroda aplikacije i ispisa logova, osmišljen je princip kojim bi se u trenutku prolaska pored kola koja se pretiču procenilo da li bi vraćanje

iz ciljne trake bilo uspešno bez stvarne potrebe da se *ov* vrati u startnu traku. Ideja i konceptualni pristup ove funkcionalnosti prikazan je u pseudokôdu 4.13. Ceo tok događaja i prepoznavanja novih fizičkih stanja nema nikakve zavisnosti od mrežnog sloja platforme, već najvećim delom od provere situacije u SUMO komponenti preko TraCI modula. Svaka stavka koja se loguje, pored informacije o tipu događaja, čuva i informacije o stanju svih vozila koja su trenutno aktivna. Stanje ubraja pozicije i brzine svih učesnika.

Potrebno je napomenuti da ceo deo koji se odnosi na tok preticanja i logovanje događaja nije ni na koji način zavisan od rada mrežnog sloja. On se tiče fizičkog preticanja i koristi u velikoj meri TraCI funkcije koje pružaju aktuelno stanje saobraćajnog dela simulacije (uključujući i vozila).

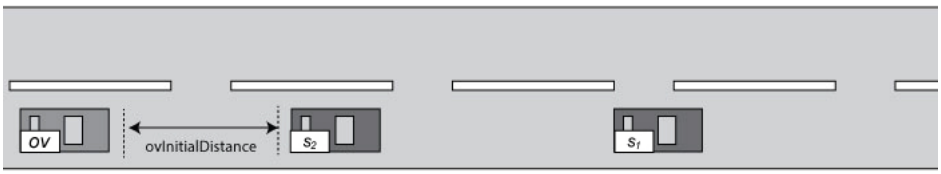
Događaji:

1. *OVERTAKING_STARTED* - vozilo *ov* se u ovom trenutku nalazi na distanci ovInitialDistance od prvog vozila ispred i preko TraCI modula se pokrenulo ubrzavanje vozila, kao i promena trake po kojoj se kreće (slika 4.9).

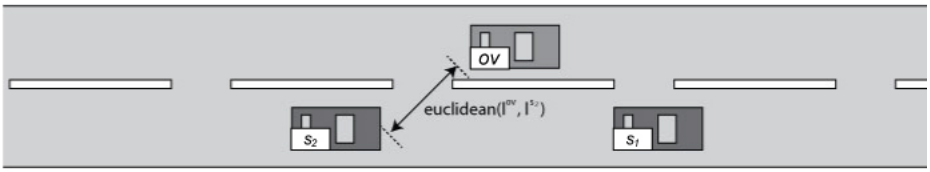
2. *OVERTAKE_UPDATE* - loguje se u trenutku tokom preticanja u kom vektor sačinjen od pozicija *ov* i vozila koje je ispred njega prestane po svom smeru da se poklapa sa smerom kretanja *ov*. To upravo označava trenutak u kom je vozilo *ov* došlo ispred vozila s_2 (slika 4.10).

3. *COLLISION* - opisuje trenutak sudara sa vozilom iz suprotne trake. Zapisuje se id vozila sa kojim se desio sudar (slika 4.11).

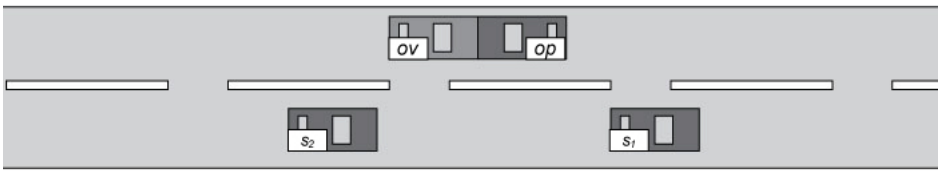
4. *OVERTAKE_DONE* - u trenutku završenog preticanja generatornog vozila *gen* (nezavisno od toga da li je bilo sudara ili ne), simulacioni scenario se smatra uspešno završenim (slika 4.12).



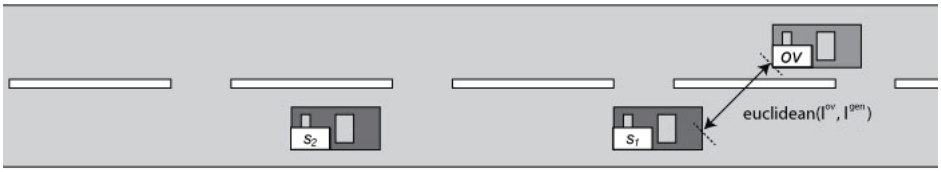
Slika 4.9: Prikaz trenutka logovanja *OVERTAKE_STARTED* događaja



Slika 4.10: Prikaz trenutka logovanja *OVERTAKE_UPDATE* događaja



Slika 4.11: Prikaz trenutka logovanja *COLLISION* događaja



Slika 4.12: Prikaz trenutka logovanja *OVERTAKE_DONE* događaja

Result: u slučaju da je vozilo *ov* preteklo *ot*, događaj je upešno ispisan, a vozilo *ot* je ažurirano ukoliko nije bilo generatorno

```
1  s^overtake ← euclidean(l^ov, l^ot);
2  if infront(ov, ot) ∧ s^overtake ≥ 0 then
3      if ot != gen then
4          log('OVERTAKE_UPDATE');
5          ot ← gen;
6      else
7          log('OVERTAKE_DONE');
8          endSimulation();
9      end
10 end
```

Slika 4.13: Pseudokôd funkcije u kojoj se ažurira status preticanja i loguju situacioni podaci. Funkcija *infront* vraca *true* ukoliko je prvo prosleđeno vozilo ispred drugog, a u suprotnom *false*.

Filtriranje neuspešnih scenarija

Širok opseg vrednosti parametara, ukupnih dužina deonica, kao i prepuštanje kontrole svojstava kao što su brzina i ubrzanje vozila SUMO komponenti, ali i ograničenja koja podrazumeva korišćeni mrežni protokol, utiču na to da može doći do *nevalidnog scenarija* koji se ne treba analizirati jer ne oslikava tačno rad algoritma.

Primećujemo naredne nevalidne scenarije koji se neće razmatrati u koraku analize rezultata:

1. Scenario nema *OVERTAKE_STARTED* događaj - u ovoj situaciji vozilo *ov* nije ni započelo preticanje. Dešava se kada se vozilo *ov* stvori predaleko od vozila koje je moguće preteći pre nego što svi izađu iz simulacionog prostora.

2. Scenario ima *OVERTAKE_STARTED* deo, ali nema *OVERTAKE_DONE* ili *COLLISION* događaje - opisuje stanje u kom je u scenariju preticanje započeto ali se do izlaska aktera iz simulacionog prostora nije desio ni sudar, a ni uspešno preticanje predviđenog vozila. U ovom slučaju ishod je ostao neizvestan.

4.6 Uključivanje novog testnog scenarija

Sada, kada je simulacija postavljena, od fizičkog podešavanja do stvaranja scenarija za svaku testnu deonicu, možemo opisati malo detaljnije korake automatizovanog procesa ubacivanja novog scenarija. Jedan od glavnih motiva rada jeste mogućnost simuliranja algoritma nad realnim testnim scenarijima. Time razvoj algoritma već u početnim fazama dobija praktični značaj. Kako bi se to omogućilo razvijen je ceo tok adaptacije početne *net* mape formatu koji odgovara definisanom modulu za simulaciju.

U nastavku, biće nabrojani koraci koje je potrebno pratiti kako bi se nova testna deonica uključila u mehanizam provere tačnosti algoritma za preticanje.

Koraci:

1. Generisanje neobrađenog SUMO scenarija - korišćenjem *osmWebWizard.py* skripte. Rezultat je folder koji sadrži sve ivice na površini specifikovanoj unutar alata, kao i fajlove potrebne za pokretanje (kao jednostavnu saobraćajnu simulaciju).

2. Uklanjanje puteva koji se uključuju u testnu deonicu puta u SUMO editoru - ovo je neophodan korak, jer automobili u SUMO simulaciji traže najbrži put pri vožnji od jedne do druge tačke, što znači da u slučaju zagušenja saobraćaja na testnoj magistrali oni mogu tražiti okolne puteve kako bi sebi skratili vreme putovanja, a to se mora predupprediti. Ovde je potrebno takođe spremiti se za dalje korake tako što se unutar SUMO editora preuzmu krajnje ivice obe trake ciljane magistrale. Kasnije će one biti korišćene za preuzimanje svih ivica jedne i druge trake koje će se zatim deliti na simulacione poddeonice.

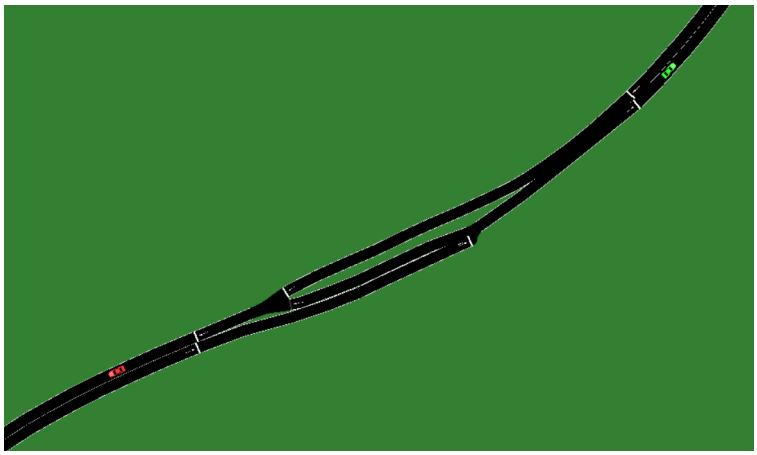
3. Pokretanje skripte *initializeNewScenario.py* - prosleđuje joj se kao argument komandne linije folder koji se automatski kreira u prvom koraku, a cilj joj je prebacivanje novog SUMO projekta u odgovarajući Veins folder u kome se čuvaju svi simulacioni scenariji. Nakon uvlačenja projekta, skripta dodaje sve

konfiguracione fajlove, uglavnom OMNeT++ koji opisuju koji mrežni protokol se koristi.

4. Pokretanje skripte *getEdgesAndDistances.py* - ona opisuje segment kôda koji je razvijen kako bi se dobile sve ivice SUMO mape puta koji nam je od značaja, kao i njihove dužine u metrima. Te dužine ivica će se koristiti kao referentne vrednosti za generisanje simulacionih scenarija prilagođenih svim dužinama testnih deonica. Potrebno je pri pozivu preko komandne linije proslediti parametre koji opisuju redom, početnu ivicu pa krajnju ivicu trake u kojoj se nalazi vozilo koje pretiče, a zatim početnu pa krajnju ivicu trake koja je suprotnog smera u odnosu na prvu. Što je duži testni put to je veća verovatnoća da neće biti moguće podeliti ga sukcesivno celom dužinom, tako da sve simulacione deonice mogu da posluže za simulaciju preticanja (npr deonice koje sadrže raskrsnice ili račvanja na više linija). Primer prekida testnog puta prikazan je na slici 4.14. U tu svrhu, uključen je korak pretprocesiranja čiji je cilj da od svih izvučenih deonica prihvati samo one koje od svog početka do kraja imaju samo dve linije, bez uključenja.

5. Pokretanje skripte *generateSubroutes.py* - na osnovu svih ivica u oba smera, kao i njihove dužine, stvaraju se manje deonice koje predstavljaju osnov za kreiranje grupe simulacija. Uzimajući u obzir granice dužina preticanja [12] čiji rasponi parametara odgovaraju i simuliranom preticanju u radu, čuvane su samo deonice koje po svojoj dužini imaju 300 ili više metara. Rezultat rada ovog finalnog koraka se može videti u primeru 4.2.

```
1  <vType id="veh_slow" vClass="passenger" maxSpeed="16" lcOpposite="0"/>
2  <route id="t0" edges="303782330#5 733564205#0"/>
3  <route id="t1" edges="733564205#9 733564205#11"/>
4  <route id="t2" edges="733564205#11 733564205#12"/>
5  <route id="t3" edges="733564205#11 733564205#12 733564205#14"/>
6  <route id="t4" edges="733564205#12 733564205#14"/>
7  <route id="o0" edges="-733564205#0 -303782330#6"/>
8  <route id="o1" edges="-733564205#11 -733564205#10"/>
9  <route id="o2" edges="-733564205#13 -733564205#11"/>
10 <route id="o3" edges="-733564205#14 -733564205#13 -733564205#11"/>
11 <route id="o4" edges="-733564205#14 -733564205#13"/>
```



Slika 4.14: Primer prekida koji deli put na dve testne deonice

```
12 <flow id="gen0_372_605" color="0.5,0.5,1" begin="0" end="50000"
       period="50" type="veh_slow" route="t0"/>
13 <flow id="gen1_633_1872" color="0.5,0.5,1" begin="50000" end="100000"
       period="50" type="veh_slow" route="t1"/>
14 <flow id="gen2_633_1872" color="0.5,0.5,1" begin="100000" end="150000"
       period="50" type="veh_slow" route="t2"/>
15 <flow id="gen3_537_1440" color="0.5,0.5,1" begin="150000" end="200000"
       period="50" type="veh_slow" route="t3"/>
16 <flow id="gen4_615_1872" color="0.5,0.5,1" begin="200000" end="250000"
       period="50" type="veh_slow" route="t4"/>
17 <flow id="gen5_358_480" color="0.5,0.5,1" begin="250000" end="300000"
       period="50" type="veh_slow" route="t5"/>
```

Listing 4.2: Primer sadržaja datoteke koja se odnosi na rute unutar simulacije. Flow definicija odgovara toku generatornih vozila na jednoj deonici testnog scenarija. Unutar identifikatora elemenata flow liste, prvi deo je naziv toka, drugi je distanca deonice u metrima, a treći broj scenarija na deonici.

Kako bi se verifikacija napravljenih deonica što više olakšala, dodatno je napisana skripta *checkRoutes.py* koja korišćenjem TraCI modula pokreće kola nad generisanim rutama. Uz to da su samim pokretanjem tih automobila generisane rute uspešno verifikovane (ukoliko SUMO ne prijavi nikakvu grešku), korišćenjem „sumo-gui" izvršnog SUMO fajla osoba zadužena za uvođenje simulacionog scenarija može i vizuelno proveriti kretanje automobila po svim deonicama.

Nakon izvršenja navedenih koraka, dobijene su poddeonice na koje direktno može

da se prikači generator konkretnih simulacionih scenarija razvijen u aplikativnom modulu OMNeT++ okvira za svrhe rada.

4.7 Analiza ishoda simulacija i procena preciznosti rigidnog algoritma

Pri analizi ishoda, glavni faktori koji su razmatrani jesu efikasnost, bezbednost i celokupna tačnost. Za detaljnije definicije navedenih karakteristika, korišćena notacija za ukupan broj istih ishoda različitih scenarija jeste:

$$ver_{<broj\ verifikovanih\ vozila>\_<broj\ pretečenih\ vozila>}$$

Na primer, ukoliko je za jedan određeni scenario, algoritam procenio da je bilo moguće preteći dva vozila, a ispostavi se nakon završetka simulacije da nije uspelo da se pretekne nijedno (pre sudara sa *op*), tada će rezultat pokretanja tog scenarija spadati u kategoriju ver_{2_0}. Cilj algoritma je, svakako, pravljenje takvih procena da nakon završetka simulacije scenarija razlika između broja verifikovanih vozila i broja pretečenih vozila bude jednaka 0.

Efikasnost - u ovom kontekstu efikasnost označava uticaj algoritma na tok saobraćaja u vidu izazivanja potencijalnih zagušenja bespotrebnim zabranama preticanja kada bi ona bila moguća. Karakteristika algoritma kojom se ogledaju njegove performanse nazvana je *pesimističnost* algoritma. Bitno je naglasiti da se za računanje procenta nije ubrajao broj slučajeva gde se desio nepredviđen negativan ishod jer se ta situacija nikako ne može nazvati efikasnom. Formule za nabrojane mere jesu:

$$Pes = ver_{0\_1} + ver_{0\_2} + ver_{1\_2}$$
$$Per = 1 - \frac{Pes}{sn}$$

gde je *sn* broj svih scenarija u kojima nije bilo sudara.

Bezbednost - opisuje rezultate scenarija u kojima je algoritam formirao ispravnu procenu ili je pogrešio preporukom da se vrši preticanje koje se ispostavilo opasnim. Karakteristiku algoritma koja negativno utiče na njegovu tačnost nazivamo

optimističnost algoritma. Korišćene formule su:

$$Opt = ver_{1\_0} + ver_{2\_0} + ver_{2\_1}$$
$$Safe = 1 - \frac{Opt}{n}$$

gde je n ukupan broj scenarija.

Tačnost - predstavlja procenat scenarija u kojima je algoritam verifikovao onaj broj vozila koji je i bio uspešno preteknut na kraju simulacije. Formula je:

$$Corr = \frac{ver_{0\_0} + ver_{1\_1} + ver_{2\_2}}{n}$$

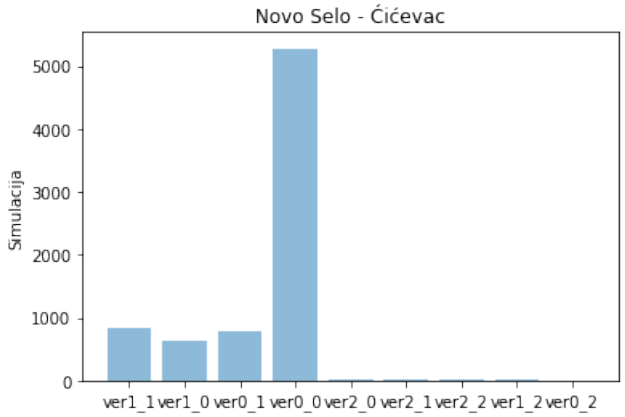
gde je n ukupan broj scenarija

Pregled obrađenih informacija Da bi se što bolje sagledali rezultati na testnim deonicama, priložena je tabela 4.2 u kojoj se može videti koliko je simulacija uspešno završeno i ocene efikasnosti i tačnosti izračunatih po navedenim formulama. Dodatno, detaljno prikazani svi ishodi se mogu videti na slikama 4.15, 4.16, 4.17 i 4.18. Zajednički detalj za sve testne scenarije je ubedljivo najveća stopa zabranjenih preticanja, što označava da su korišćenim opsegom parametara obuhvaćeni baš ti granični slučajevi u kojima bi se desio sudar. Takođe, na svim putevima, broj ishoda ver_{1_1} veći je od ver_{1_0}.

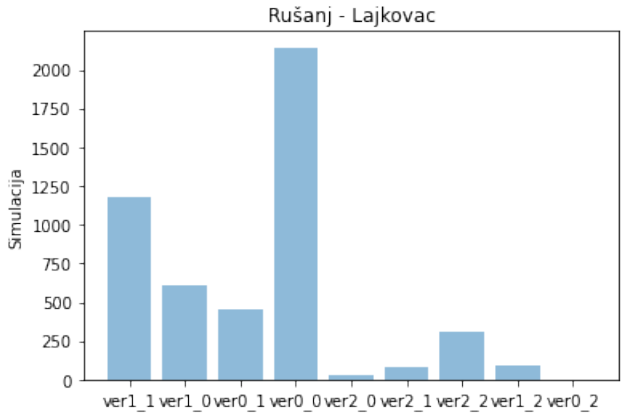
Deonica	Uspešne simulacije	Per (%)	Safe (%)	Corr (%)
Novo Selo - Ćićevac	7626	88.40	91.26	80.68
Rušanj - Lajkovac	4893	87.02	85.22	74.16
Prijepolje - Čajetina	10717	85.52	97.07	83.02
Dubci - Pakovraće	9446	87.54	96.95	84.87

Tabela 4.2: Pregled obrađenih informacija za sve scenarije

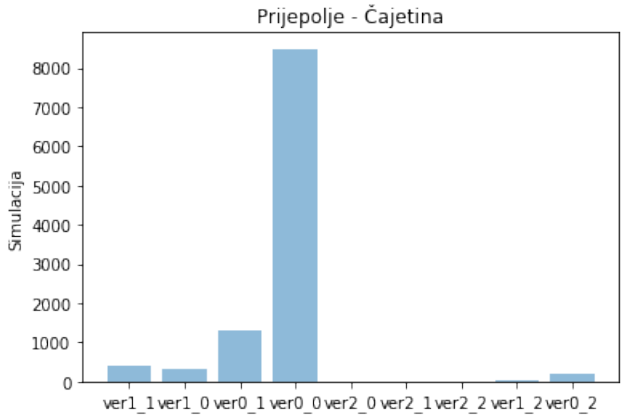
Pregled odnosa broja promašaja i uspešnih procena Iako je algoritam bio približno efikasan na svim deonicama, interesantno je zapažanje da je odnos



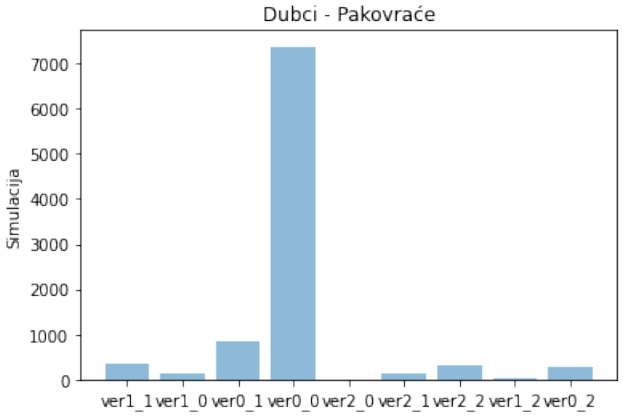
Slika 4.15: Pregled svih ishoda na deonici Novo Selo - Ćićevac



Slika 4.16: Pregled svih ishoda na deonici Rušanj - Lajkovac



Slika 4.17: Pregled svih ishoda na deonici Prijepolje - Čajetina



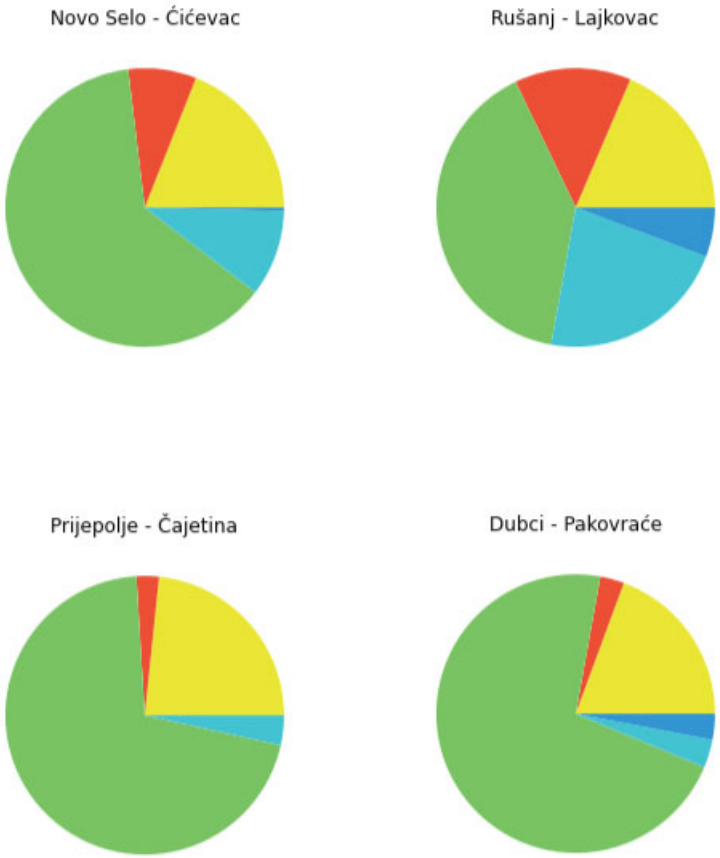
Slika 4.18: Pregled svih ishoda na deonici Dubci - Pakovraće

ispravno verifikovanih dozvoljenih preticanja, optimističnosti i pesimističnosti imao primetne razlike po vrednostima za različite testne deonice. Grafički prikaz tih odnosa može se videti na slici 4.19

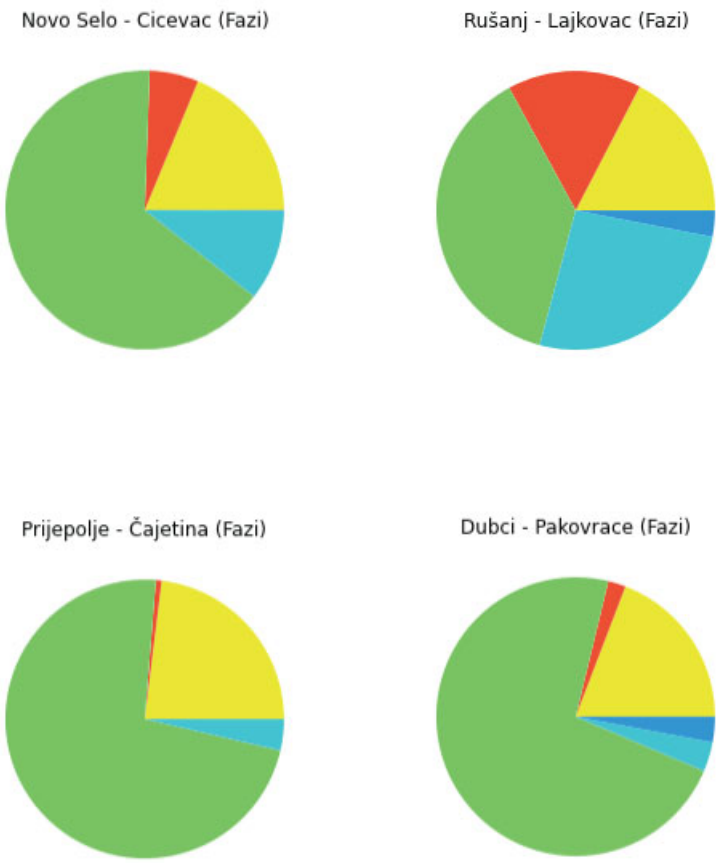
Uzimajući u obzir to da je algoritam zasnovan na osnovnim fizičkim ulaznim vrednostima, efikasnost algoritma se može smatrati zadovoljavajućom osnovom za dalji razvoj i proširivanje modela. Karakteristika koja otežava rad algoritma u graničnim slučajevima, onim u kojima je vremenska razlika između uspešnog preticanja i sudara minimalna, jeste njegova rigidnost i unapred postavljena obaveza da za svaki ishod da odgovor: da (dozvoljeno je preticanje) ili ne (nije dozvoljeno). U narednom poglavlju biće opisan jedan pristup nalaženju finijeg prelaza između potencijalnih ishoda.

4.8 Analiza ishoda simulacija i procena preciznosti algoritma zasnovanog na verovatnoći

Pokretanje razvijena dva algoritma nad istim scenarijima pruža mogućnost poređenja njihovih efikasnosti. Kako bi se ta razlika prikazala, za svaku deonicu, pored karakteristika koje su proveravane i u sekciji 4.7, u nastavku će pored dobijenih vrednosti biti prikazana i razlika u odnosu na rezultate rigidnog algoritma. Rezultati i poređenje prikazani su u tabeli 4.3. Može se primetiti da je bezbednost primetno povećana na svim putanjama praćena tačnošću a zatim i performansama, sem na



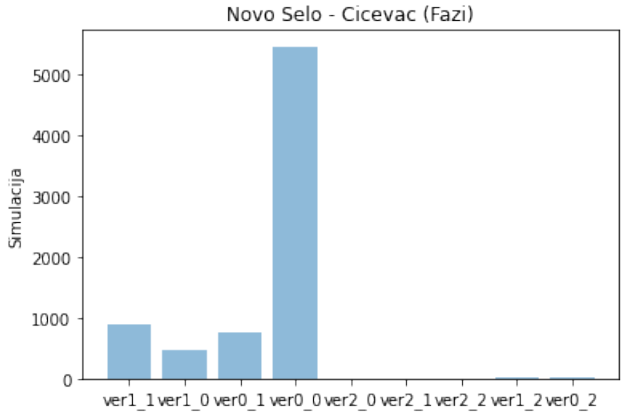
Slika 4.19: Odnos uspešnih procena i promašaja. Zelenom bojom prikazan je procenat uspešnih zabrana preticanja, svetlo plavom uspešnih procena da se može preteći jedno vozilo, tamno plavom da se uspešno mogu preteći dva vozila, dok crvena označava zbir svih procena u kojima se desio sudar koji nije otkriven (optimističnost), a žutom zbir slučajeva u kojima su bila zabranjena preticanja koja bi se uspešno odvila (pesimističnost)



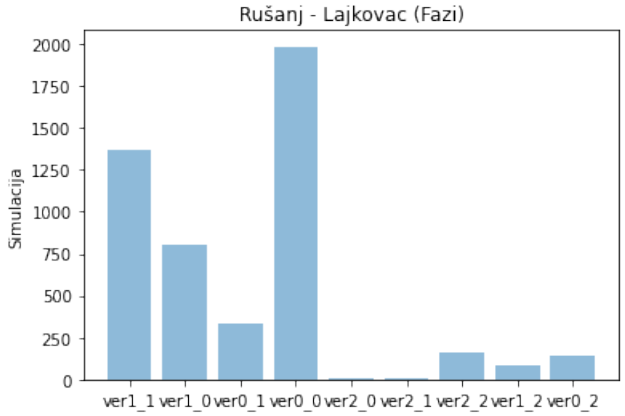
Slika 4.20: Odnos uspešnih procena i promašaja za algoritam zasnovan na verovatnoći

deonici Rušanj-Lajkovac koja je i kod samog rigidnog algoritma imala za preko 10% manju tačnost od ostalih. Prikaz svih ishoda dat je na slikama 4.21, 4.22, 4.23 i 4.24, a odnos optimističnosti, pesimističnosti i tačnosti dat je na slici 4.20.

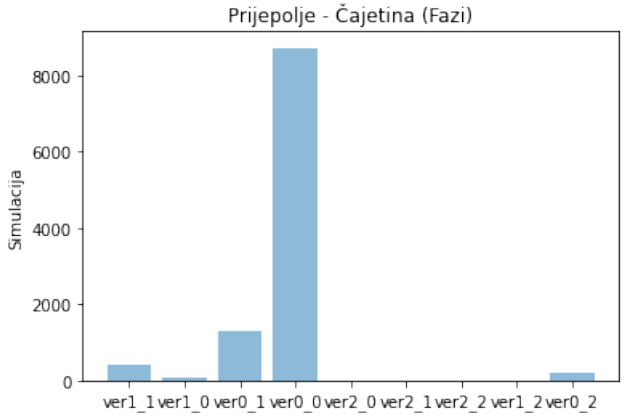
Dobijeni rezultati ukazuju na potencijal razlaganja kritične granice ishoda pri donošenju odluke na širi, kontrolisaniji skup. Suštinski, promenom osnovnog oblika *collisionProbability* funkcije, kao i uvođenjem parametara koji bi je prilagodili lokalnim uslovima i okolnostima, mogli bi se očekivati još bolji rezultati. O mogućim nadograđivanjima i proširivanjima algoritama i platforme za simulacije biće reči u narednom poglavlju.



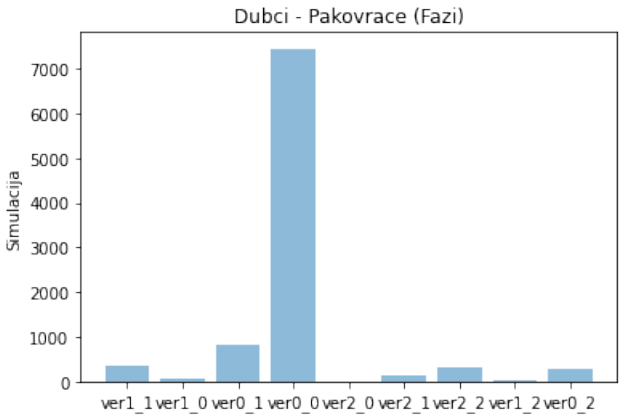
Slika 4.21: Pregled svih ishoda na deonici Novo Selo - Ćićevac za algoritam zasnovan na verovatnoći



Slika 4.22: Pregled svih ishoda na deonici Rušanj - Lajkovac za algoritam zasnovan na verovatnoći



Slika 4.23: Pregled svih ishoda na deonici Prijepolje - Čajetina za algoritam zasnovan na verovatnoći



Slika 4.24: Pregled svih ishoda na deonici Dubci - Pakovraće za algoritam zasnovan na verovatnoći

Deonica	Uspešne simulacije	Per (%)	Safe (%)	Corr (%)
Novo Selo - Ćićevac	7626	88.71 (+0.31)	93.73 (+2.47)	83.14 (+2.46)
Rušanj - Lajkovac	4893	85.97 (-1.05)	83.39 (-1.83)	71.70 (-2.46)
Prijepolje - Čajetina	10717	86.02 (+0.5)	99.24 (+2.17)	85.37 (+2.35)
Dubci - Pakovraće	9446	87.72 (+0.18)	97.76 (+0.81)	85.76 (+0.89)

Tabela 4.3: Pregled obrađenih informacija za sve scenarije

Glava 5

Zaključak

Kombinacijom svih razvijenih i prilagođenih komponenti u radu, napravljeno je celokupno razvojno okruženje za automobilski računarski sistem koji asistira pri preticanju. Dodatno, predložena je modularizacija, po kojoj je i navedeni modul razvijen, a koja stvara korak ka spajanju rada većeg broja modula zaduženih za različite saobraćajne procedure. Interesantna nadogradnja postojeće platforme mogao bi da bude modul za mašinsko učenje koji bi, osmatranjem konkretnih vožnji osobe koja koristi instancu računarskog sistema za asistenciju pri preticanju, pružio još preciznije ulazne parametre za komponentu zaduženu za odlučivanje ishoda i procenu bezbednosti. Dalje, uključivanjem informacija koje bi se dobijale iz senzora automobila, algoritmima bi se pružilo kompleksnije okruženje u kom oni ne bi morali da se zasnivaju isključivo na prostim fizičkim veličinama poput lokacija, brzina i ubrzanja, već npr. i na potencijalu kočenja, trenutne vidljivosti itd.

Uopštenije, preciznost i upotrebljivost samog algoritma koji se testira zavisi najvećim delom direktno od kvaliteta platforme na kojoj je vršena procena njegove tačnosti i stepena realističnosti simulacije. Što su mrežna komunikacija, kretanje vozila i magistrale koje se koriste u testnim scenarijima bolji u dočaravanju realnog sveta, to će lakše biti vršenje tranzicije na terensko testiranje, a na kraju i na korišćenje samog algoritma. Upravo zbog toga, poželjno je posvetiti približno jednaku pažnju kako konkretnim algoritmima koji procenjuju verovatnoće ishoda tako i samoj razvojnoj platformi.

Automatizacijom procesa generisanja simulacionih scenarija omogućen je lak unos novih testnih deonica koji značajno utiču na kvalitet testiranja razvijenih algoritama. Međutim, u trenutnoj verziji razvojne platforme, novi scenariji se stvaraju pozivanjem skripti koje su zadužene za različite delove procesa putem komandne li-

nije pa bi prirodan dalji korak koji bi još više olakšao kontrolu simulacionih testova bio integracija grafičkog interfejsa za podešavanje i pokretanje simulacija. Odabirom Veins simulacionog okruženja obezbeđena je fleksibilnost po pitanju upotrebe protokola, jer podržava mnoge aktuelne standarde. Jedan od najbitnijih produkata, jeste baza algoritma zasnovanog na verovatnoći koja je prikazala zadovoljavajuće testne rezultate u svojoj početnoj fazi, a svakako ima mnogo prostora za unapređenja.

Automobilska industrija otvorena je za razne inovacije i primene tehnologija koje mogu da povećaju performanse, efikasnost korišćenja resursa ili bezbednost. Sadašnji trenutak, u kome jedna takva tehnologija kao što je VANET još nije doživela vrhunac primene, pruža dobru priliku za razmatranje budućih aplikacija i njihov razvoj. U ovom radu, prikazan je računarski sistem koji nastoji da svojim pristupom podrži taj razvoj.

Bibliografija

[1] Open street map (https://www.openstreetmap.org/).

[2] Mani Amoozadeh, Hui Deng, Chen-Nee Chuah, H. Michael Zhang, and Dipak Ghosal. Platoon management with cooperative adaptive cruise control enabled by vanet. *Vehicular Communications*, 2(2):110–123, 2015.

[3] Huda Al Amri, Mehran Abolhasan, and Tadeusz Wysocki. Scalability of manet routing protocols for heterogeneous and homogenous networks. *Computers & Electrical Engineering*, 36(4):752–765, 2010. Signal Processing and Communication Systems.

[4] Ihn-Han Bae and Jae-Kon Lee. Design and performance evaluation of a vanet-based adaptive overtaking assistance system. In James J. (Jong Hyuk) Park, Han-Chieh Chao, Hamid Arabnia, and Neil Y. Yen, editors, *Advanced Multimedia and Ubiquitous Engineering*, pages 59–69, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[5] Annkur Bang and Prabhakar Ramteke. Manet: History,challenges and applications. 09 2013.

[6] BBC. German bundestag adopts autonomous driving law.

[7] BBC. Self-driving vehicles to be trialled in cambridge.

[8] P.S. Bokare and A.K. Maurya. Acceleration-deceleration behaviour of various vehicle types. *Transportation Research Procedia*, 25:4733–4749, 2017. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016.

[9] Mohamed Salah Bouassida and Mohamed Shawky. On the congestion control within vanet. In *2008 1st IFIP Wireless Days*, pages 1–5, 2008.

[10] Capacitymedia. Eu ambassadors reject 'wifi-only' move for autonomous cars.

[11] Scott E. Carpenter. Obstacle shadowing influences in vanet safety. In *2014 IEEE 22nd International Conference on Network Protocols*, pages 480–482, 2014.

[12] Tushar Choudhari, Anuj Budhkar, and Avijit Maji. Modeling overtaking distance and time along two-lane undivided rural highways in mixed traffic condition. *Transportation Letters*, 0(0):1–9, 2020.

[13] Manuel Gonzalez-Martín, Miguel Sepulcre, Rafael Molina-Masegosa, and Javier Gozalvez. Analytical models of the performance of c-v2x mode 4 vehicular communications. *IEEE Transactions on Vehicular Technology*, 68(2):1155–1166, 2019.

[14] Shantanu Ingle and Madhuri Phute. Tesla autopilot: semi autonomous driving, an uptick for future autonomy. *International Research Journal of Engineering and Technology*, 3(9):369–372, 2016.

[15] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. 2020.

[16] Ammara Anjum Khan, Mehran Abolhasan, and Wei Ni. 5g next generation vanets using sdn and fog computing framework. In *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6, 2018.

[17] Ganesh S. Khekare and Apeksha V. Sakhare. A smart city framework for intelligent traffic system using vanet. In *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*, pages 302–305, 2013.

[18] Arkadeep Kumar. Methods and materials for smart manufacturing: Additive manufacturing, internet of things, flexible sensors and soft robotics. *Manufacturing Letters*, 15:122–125, 2018. Industry 4.0 and Smart Manufacturing.

[19] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[20] Alireza Marefat, Rozita Aboki, Ali Jalooli, Erfan Shaghaghi, Mohammad Reza Jabbarpour, and Rafidah Md Noor. An adaptive overtaking maneuver assistant

system using vanet. In *2014 IEEE Asia Pacific Conference on Wireless and Mobile*, pages 316–321, 2014.

[21] MarketWatch. Internet of things (iot) market trends outlook 2021- industry analysis by growth segments, opportunity and challenges, future scope and business size, with demand forecast analysis 2024.

[22] Mohamed Nidhal Mejri, Jalel Ben-Othman, and Mohamed Hamdi. Survey on vanet security challenges and possible cryptographic solutions. *Vehicular Communications*, 1(2):53–66, 2014.

[23] Mahammad Shareef Mekala and P. Viswanathan. A survey: Smart agriculture iot with cloud computing. In *2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS)*, pages 1–7, 2017.

[24] Chunmei Mo, Yinong Li, and Zheng Ling. Simulation and analysis on overtaking safety assistance system based on vehicle-to-vehicle communication. *Automotive Innovation*, 1, 06 2018.

[25] Nikkei. Cheaper lidar sensors brighten the future of autonomous cars.

[26] H. Noori and B. Badihi Olyaei. A novel study on beaconing for vanet-based vehicle to vehicle communication: Probability of beacon delivery in realistic large-scale urban area using 802.11p. In *2013 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, volume 01, pages 1–6, 2013.

[27] Mamata Rath, B. K. Pattanayak, and Bibudhendu Pati. Energy efficient manet protocol using cross layer design for military applications. *Defence Science Journal*, 66:146–150, 2016.

[28] Thomas Richter, Stephan Ruhl, Jörg Ortlepp, and Emmanuel Bakaba. Causes, consequences and countermeasures of overtaking accidents on two-lane rural roads. *Transportation Research Procedia*, 25:1989–2001, 2017. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016.

[29] William H. Robinson and Adrian P. Lauf. Resilient and efficient manet aerial communications for search and rescue applications. In *2013 International Conference on Computing, Networking and Communications (ICNC)*, pages 845–849, 2013.

[30] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing (TMC)*, 10(1):3–15, January 2011.

[31] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems; Workshops*, Simutools '08, Brussels, BEL, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[32] Pengfei Wang, Boya Di, Hongliang Zhang, Kaigui Bian, and Lingyang Song. Cellular v2x communications in unlicensed spectrum: Harmonious coexistence with vanet in 5g systems. *IEEE Transactions on Wireless Communications*, 17(8):5212–5224, 2018.

[33] Agencija za bezbednost saobraćaja Republika Srbija. https://www.abs.gov.rs/.

[34] Shizhe Zang, Ming Ding, David Smith, Paul Tyler, Thierry Rakotoarivelo, and Mohamed Ali Kaafar. The impact of adverse weather conditions on autonomous vehicles: How rain, snow, fog, and hail affect the performance of a self-driving car. *IEEE Vehicular Technology Magazine*, 14(2):103–111, 2019.