

Zahvaljujem mentoru na stručnom vodstvu i korisnim savjetima tijekom izrade ovog rada.

Zahvaljujem svojoj obitelji i zaručniku na podršci i strpljenju kroz cijeli studij.

Sadržaj

1. Uvod	3
2. Problem upravljanja semaforima na raskrižjima	4
2.1. Struktura raskrižja sa semaforom	4
2.2. Metrike učinkovitosti	5
2.3. pristupi upravljanju signalnim planom	5
3. Genetsko programiranje	7
3.1. Evolucijski algoritmi	7
3.2. Prikaz rješenja stablima	7
3.3. Inicijalizacija populacije	8
3.4. Selekcija	8
3.5. Križanje	9
3.6. Mutacija	9
3.7. pristupi koevoluciji upravljačkih funkcija	9
3.7.1. GP Light+ — jedno dijeljeno stablo	9
3.7.2. Grupirana koevolucija	10
3.7.3. Kooperativna koevolucija s odvojenim populacijama	10
4. Simulator prometnog toka	12
4.1. CityFlow	12
4.2. CityFlowCpp	13
4.2.1. Arhitektura i višedretveno izvođenje	13
4.2.2. Programsko sučelje biblioteke	13
4.2.3. Prošireni skup prometnih metrika	14
4.2.4. Praćenje stanja na razini vozila	14

4.2.5. Praćenje maksimalnog repa čekanja	15
4.3. Korišteni prometni scenariji	15
4.4. Načini upravljanja simulacijom	15
4.5. Ulazne varijable upravljačke funkcije	16
4.6. Interval odlučivanja i tijek simulacije	17
4.7. Evaluacija u evolucijskom procesu	18
5. Implementacija sustava	19
5.1. Arhitektura sustava	19
5.2. Upravljački agent	20
5.3. GP Light+ evaluator	20
5.4. Grupirana koevolucija	20
5.5. Kooperativna koevolucija s odvojenim populacijama	21
5.6. Organizacija eksperimenata i pohrana rezultata	21
5.7. Parametri evolucijskog procesa	22
6. Eksperimenti i rezultati	23
6.1. Eksperimentalna postava	23
6.2. Referentne vrijednosti	23
6.3. GP Light+	24
6.4. Grupirana koevolucija	25
6.5. Kooperativna koevolucija	27
6.6. Usporedba metoda	27
6.7. Testiranje robusnosti	28
6.8. Rasprava	29
7. Zaključak	32
Literatura	34
Sažetak	36
Abstract	37

1. Uvod

Upravljanje prometnim semaforima jedan je od ključnih čimbenika učinkovitosti gradske prometne mreže. Unaprijed određeni rasporedi faza, postavljeni prema prosječnoj gustoći prometa, ne mogu se prilagoditi promjenama koje su u urbanim sredinama svakodnevna pojava: jutarnjim gužvama, nepredvidivim incidentima ili sezonskim promjenama u protoku vozila. Posljedica je nedovoljno iskorišten kapacitet mreže i povećano prosječno vrijeme putovanja.

Metode temeljene na evolucijskom računarstvu nude mogućnost automatskog pronalaženja upravljačkih strategija optimiziranjem izravno na simuliranom prometnom toku, bez potrebe za ručnim definiranjem pravila. Genetsko programiranje pritom je posebno pogodno jer evoluiraju upravljačke funkcije u obliku matematičkih izraza koje je moguće izravno interpretirati i primijeniti u stvarnom sustavu [1].

U ovom radu istražuje se primjena genetskog programiranja za evoluciju upravljačkih funkcija semafora u mreži od 16 raskrižja. Svako raskrižje upravlja se matematičkim izrazom koji na temelju trenutnog broja zaustavljenih vozila i ukupnog broja vozila po traci odlučuje kojoj signalnoj fazi dati prednost. Središnje istraživačko pitanje jest kako organizirati evolucijski proces za cijelu mrežu. Istražena su tri pristupa: evolucija jednog zajedničkog stabla za sva raskrižja, grupirana koevolucija koja pojedinim grupama raskrižja dodjeljuje zasebna stabla unutar jedne populacije te kooperativna koevolucija s odvojenim populacijama za svaku grupu.

Za pokretanje prometnih simulacija korišten je simulator CityFlow [2], a za njegovo povezivanje s evolucijskim okvirom ECF [3] razvijena je C++ biblioteka CityFlowCpp. Eksperimenti su provedeni na stvarnoj prometnoj mreži iz kineskog grada Hangzhoua, za koju su dostupni stvarni prometni podaci prikupljeni mjerenjima u prometu.

2. Problem upravljanja semaforima na raskrižjima

Promet u urbanim sredinama oslanja se na semafore koji reguliraju tok vozila na raskrižjima. Zadaća semafora nije samo osigurati sigurnost sudionika prometa, već i održati protočnost mreže u uvjetima koji se neprestano mijenjaju: od jutarnjih opterećenja i posebnih događaja poput koncerata ili sportskih utakmica, do nepredvidivih situacija uzrokovanih prometnim nesrećama. S rastom broja vozila u gradovima, raste i potreba za sustavima koji se dinamički prilagođavaju trenutnim uvjetima, što je potaknulo razvoj tzv. pametnih semafora sposobnih za adaptivno upravljanje prometom.

2.1. Struktura raskrižja sa semaforom

Tipično raskrižje sa semaforom sastoji se od dolaznih prometnih traka, kroz koje vozila pristupaju raskrižju, i odlaznih traka kojima ga napuštaju. Trake mogu biti grupirane po smjeru, a broj smjerova raskrižja najčešće iznosi dva do četiri.

Semafor koji upravlja raskrižjem organiziran je kroz signalni plan koji definira redoslijed i trajanje signalnih faza. Svaka signalna faza određuje skup pokreta kojima je istovremeno dopušten prolaz, primjerice, jedna faza može dopuštati prolaz vozilima sa sjevera prema jugu i s juga prema sjever, dok su ostali smjerovi zaustavljeni. Unutar jedne faze signalna svjetla izmjenjuju se sekvencijalno:

- *zeleno* — određenim dolaznim trakama dopušten je prolaz, ostalima nije,
- *žuto* — prelazno stanje kojim se vozači upozoravaju na predstojeće crveno,
- *crveno* — svim dolaznim trakama zabranjen je prolaz.

Trajanje zelene faze ovisi o ukupnom broju faza semafora i prometnom opterećenju

raskrižja. Žuta faza služi kao prelazno upozorenje i tipično traje nekoliko sekundi.

2.2. Metrike učinkovitosti

Za ocjenu kvalitete upravljanja semaforima koriste se različite metrike, od kojih su najčešće:

- Prosječno vrijeme putovanja — srednje vrijeme koje vozilo provede u mreži od ulaska do izlaska; izravno odražava iskustvo vozača u prometnom toku,
- Duljina repa čekanja — broj vozila koja čekaju ispred semafora u danom trenutku; indikator mogućeg zagušenja susjednih raskrižja,
- Propusnost — ukupan broj vozila koji prođu kroz raskrižje u zadanom vremenskom periodu,
- Kašnjenje — razlika između stvarnog vremena putovanja i idealnog vremena bez zastoja.

U ovom radu kao ciljna mjera optimizacije koristi se prosječno vrijeme putovanja svih vozila koja su dovršila putovanje u mreži [4].

2.3. pristupi upravljanju signalnim planom

Metode upravljanja prometnim semaforima dijele se na statičke i dinamičke [5].

Statičko upravljanje temelji se na unaprijed definiranim, nepromjenjivim vremenskim rasporedima faza [6]. Parametri se kalibriraju na temelju povijesnih prometnih podataka. Prednost ovakvog pristupa je jednostavnost implementacije u postojeću infrastrukturu bez potrebe za dodatnim senzorima. Nedostatak je nemogućnost reagiranja na iznenadne promjene prometnog opterećenja. Kao djelomično rješenje, neki sustavi definiraju više programa prilagođenih određenim satima u danu ili posebnim događajima, no ni takav pristup ne uzima u obzir stvarno stanje prometa u realnom vremenu.

Dinamičko upravljanje koristi podatke senzora, kamera ili induktivnih petlji za prilagodbu trajanja faza trenutnom prometnom toku. Jednostavniji oblici, poznati kao *ac-*

tuated metode, primjenjuju unaprijed definirana pravila, primjerice, produžuju zelenu fazu ako detektori bilježe povećan dolazak vozila. Naprednije optimizacijske metode koriste algoritme koji automatski pronalaze učinkovite strategije upravljanja bez ručnog podešavanja pravila. Ovaj pristup posebno je zanimljiv kada se radi o mreži međusobno povezanih raskrižja, gdje odluke na jednom čvoru izravno utječu na prometni tok u susjednim.

U ovom radu istraživana je upravo takva optimizacijska metoda, genetsko programiranje koje evolucijom matematičkih izraza traži upravljačku funkciju sposobnu dinamički donositi odluke o prioritetu prometnih faza na svakom raskrižju, uz istovremenu koordinaciju cijele mreže koevolucijskim pristupom [4, 7].

3. Genetsko programiranje

Genetsko programiranje [1] poseban je oblik evolucijskog algoritma koji evoluira rješenja prikazana kao sintaksna stabla. Za razliku od klasičnih optimizacijskih metoda, genetsko programiranje ne pretpostavlja unaprijed određenu strukturu rješenja već veličina i oblik programa nastaju kao rezultat samog procesa evolucije. Takav pristup čini ga primjenjivim na raznolike probleme, od simboličke regresije i klasifikacije do upravljanja složenim sustavima, a konačno rješenje moguće je izravno interpretirati.

3.1. Evolucijski algoritmi

Evolucijski algoritmi populacijske su metaheuristike inspirirane procesima prirodne selekcije. Svaka jedinka iz populacije predstavlja jedno potencijalno rješenje problema. Populacija se inicijalizira nasumično i mijenja kroz generacije primjenom operatora selekcije, križanja i mutacije. Kvaliteta rješenja ocjenjuje se funkcijom dobrote koju nastojimo maksimizirati, odnosno funkcijom kazne koju nastojimo minimizirati.

Važnu ulogu igra odabir prikaza rješenja jer on izravno određuje koji su operatori križanja i mutacije primjenjivi. U genetskom programiranju rješenja su prikazana stablima, što omogućuje prirodno modeliranje matematičkih izraza.

3.2. Prikaz rješenja stablima

Stablo u genetskom programiranju sastoji se od unutarnjih čvorova koji predstavljaju elementarne funkcije te listova koji predstavljaju terminale. Elementarne funkcije najčešće su aritmetičke operacije poput zbrajanja, oduzimanja, množenja i dijeljenja, ali mogu uključivati i složenije operacije poput trigonometrijskih funkcija, uvjetnih izraza ili logičkih operatora. Terminali su varijable, konstante ili posebni čvorovi zvani ERC

(engl. *Ephemeral Random Constants*) koji se inicijaliziraju na nasumičnu vrijednost pri stvaranju stabla.

Evaluacija stabla odvija se rekurzivno od korijena prema listovima. Svaki unutarnji čvor izračunava svoju vrijednost na temelju vrijednosti svojih potomaka, dok listovi primaju zadane ili izmjerene vrijednosti. Konačni rezultat jest vrijednost korijenskog čvora.

Ključan hiperparametar jest maksimalna dubina stabla. Stablo premale dubine nedovoljno je ekspresivno za opisivanje složenih odnosa, dok prevelika dubina vodi prekomjernom prilagođavanju i pojavi napuhavanja (engl. *bloat*), uvođenja redundantnih podstabala koja ne doprinose kvaliteti rješenja.

3.3. Inicijalizacija populacije

Za generiranje početne populacije stabala koriste se tri pristupa. Metoda *full* gradi potpuna stabla u kojima svi listovi leže na maksimalnoj dubini. Metoda *grow* gradi stabla varijabilne strukture u kojima listovi mogu biti na različitim dubinama. Metoda *ramped half-and-half*, koja se u praksi najčešće koristi, kombinira oba pristupa čime se postiže raznolikost oblika i veličina stabala u početnoj populaciji.

3.4. Selekcija

Operator selekcije određuje koje jedinke sudjeluju u stvaranju potomaka. Seleksijski pritisak mora biti uravnotežen, preveliki pritisak vodi preranoj konvergenciji i gubitku raznolikosti populacije, dok prenizak pritisak čini pretragu previše nasumičnom.

U genetskom programiranju najčešće se koristi turnirska selekcija. Iz populacije se nasumično odabire k jedinki, a pobjednik turnira, jedinka s najboljom vrijednosti funkcije dobrote, postaje roditelj. Vrijednost k hiperparametar je koji izravno kontrolira seleksijski pritisak. U ovom radu koristi se turnirska selekcija s $k = 3$.

3.5. Križanje

Operator križanja kombinira genetski materijal dviju roditeljskih jedinki. U genetskom programiranju najčešće se primjenjuje križanje s jednom točkom presjeka: iz stabla svakog roditelja nasumično se odabire jedan čvor, a podstabla kojima su ti čvorovi korijeni međusobno se izmjenjuju. Rezultat su dvije nove jedinke koje nasljeđuju strukturu obaju roditelja.

Križanje sužava prostor pretrage usmjeravajući evoluciju prema kombinacijama već pronađenih dobrih gradivnih blokova. Pritom je potrebno osigurati da dubina novo nastalih stabala ne prelazi zadanu granicu.

3.6. Mutacija

Operator mutacije uvodi nasumične promjene u pojedinu jedinku. Tipičan oblik mutacije u genetskom programiranju nasumično odabire čvor stabla te briše njega i njegovo podstablo, na čije mjesto postavlja novo, nasumično generirano podstablo. Ostale varijante uključuju zamjenu primitiva čvora, dodavanje ili odrezivanje grane.

Mutacija povećava raznolikost populacije i omogućuje istraživanje dijelova prostora rješenja nedostupnih samim križanjem. Previše agresivna mutacija narušava korisne strukture i degenerira pretragu u nasumično lutanje, pa se stoga mutacija obično primjenjuje s relativno niskom vjerojatnošću. U ovom radu ispitane su dvije vrijednosti vjerojatnosti mutacije po jedinki: 0,1 (sukladno postavkama u [4]) te 0,5.

3.7. pristupi koevoluciji upravljačkih funkcija

Kada se upravljačke funkcije razvijaju za više raskrižja istovremeno, ključno je pitanje kako organizirati populaciju i evaluirati jedinke. U ovom radu istražena su tri pristupa koji se razlikuju po stupnju specijalizacije i načinu evaluacije.

3.7.1. GP Light+ — jedno dijeljeno stablo

Najjednostavniji pristup jest evolucija *jednog dijeljenog stabla* koje se koristi za upravljanje svim raskrižjima u mreži. Svaka jedinka u populaciji sadrži samo jedno stablo,

a ono se evaluira pokretanjem simulacije u kojoj sva raskrižja koriste identičnu upravljačku funkciju. Ovaj pristup, nazvan *GP Light+*, pretpostavlja da postoji jedna generička strategija koja je dovoljno dobra za sve čvorove mreže, neovisno o njihovim specifičnim prometnim karakteristikama [4].

Prednost *GP Light+* je računalna učinkovitost, populacija je mala i svaka evaluacija odgovara jednoj simulaciji. Nedostatak je ograničena izražajnost: jedno stablo ne može uhvatiti razlike između raskrižja različitih prometnih opterećenja ili topoloških položaja u mreži.

3.7.2. Grupirana koevolucija

U *grupiranoj koevoluciji* mreža se dijeli na grupe od g raskrižja, pri čemu svaka grupa dijeli jedno stablo. Parametar g naziva se *veličina grupe* (*GROUP_SIZE*). Svaka jedinka u populaciji sadrži $\lceil N/g \rceil$ stabala, gdje je N ukupan broj raskrižja, a g veličina grupe.

Dobrota jedinke ne ovisi samo o kvaliteti pojedinog stabla, već o tome kako sva stabla zajedno upravljaju prometnim tokom u jednoj simulaciji. Time se implicitno uzimaju u obzir međuovisnosti između susjednih raskrižja. Istovremena evolucija svih stabala unutar jedne populacije čini ovaj pristup prirodnom nadogradnjom *GP Light+* pristupa, a parametar g kontrolira kompromis između specijalizacije i kompleksnosti pretrage:

- $g = N$ — ekvivalentno *GP Light+* pristupu (jedno stablo za cijelu mrežu),
- $g = 1$ — svako raskrižje ima vlastito stablo; maksimalna specijalizacija unutar zajedničke populacije,
- $1 < g < N$ — kompromis između specijalizacije i veličine genoma.

3.7.3. Kooperativna koevolucija s odvojenim populacijama

U *grupiranoj koevoluciji* sva stabla evoluiraju unutar jedne populacije, što znači da selekcijski pritisak djeluje na razini cjelokupne jedinke, a ne pojedinog stabla. *Kooperativna koevolucija* [8] rješava taj problem odvajanjem svake grupe raskrižja u zasebnu populaciju. Svaka populacija evoluira jedno stablo neovisno, a kvaliteta stabla ocjenjuje se pokretanjem simulacije u kojoj to stablo surađuje s trenutno najboljim stablima svih os-

talih populacija.

Evolucija se odvija u rundama. U svakoj rundi redom se evoluiraju svake populacije kroz S generacija, dok ostale populacije miruju i koriste svoja do tada najbolja stabla. Nakon što sve populacije završe jednu rundu, najbolja stabla ažuriraju se i proces se ponavlja za sljedeću rundu. Ukupan broj efektivnih generacija po populaciji iznosi $S \times R$, gdje je R broj rundi.

Kako bi se ubrzala konvergencija, na početku svake runde dio individua u populaciji inicijalizira se najboljim stablom iz prethodne runde (*warm start*), čime se sprječava gubitak znanja stečenog u ranijim rundama.

Prednost ovog pristupa jest izravniji selekcijski signal, svaka populacija optimizira samo jedno stablo, čime se ublažava problem pripisivanja zasluge prisutan u grupiranoj koevoluciji. Nedostatak je veće računalno vrijeme jer se populacije evoluiraju sekvencijalno.

4. Simulator prometnog toka

Za evaluaciju evoluiranih upravljačkih funkcija potreban je simulator koji vjerno modelira prometni tok i omogućuje brzo ponavljanje simulacija kakvo zahtijeva evolucijski proces. U ovom radu koristi se simulator CityFlow [2], a za njegovu integraciju s evolucijskim okvirom ECF [3] razvijena je C++ biblioteka CityFlowCpp.

4.1. CityFlow

CityFlow je simulator prometnog toka otvorenog koda koji modelira kretanje individualnih vozila kroz gradsku prometnu mrežu. Svako vozilo simulira se kao zasebni objekt koji se kreće prometnom mrežom sukladno zadanim parametrima (maksimalna brzina, ubrzanje, kočenje, minimalni razmak između vozila). Prometna mreža i ponašanje simulacije opisuju se trima JSON datotekama: datotekom prometne mreže (*roadnet*) koja definira raskrižja i ceste, datotekom prometnih tokova (*flow*) koja definira vozila, njihove rute i vremena polaska, te konfiguracijskom datotekom (*config*) koja povezuje prethodne dvije i određuje globalne parametre simulacije poput vremenskog intervala koraka i slučajnog sjemenja.

CityFlow izvorno nudi programsko sučelje u Pythonu putem kojeg vanjski upravljački program može u svakom koraku simulacije dohvatiti stanje prometne mreže i zadati nove signalne faze. Jedan korak simulacije odgovara jednoj sekundi simuliranog vremena.

Ključna prednost CityFlowa u odnosu na neke starije simulatore jest brzina izvođenja. Zahvaljujući arhitekturi s više dretvi, CityFlow može simulirati velike prometne mreže znatno brže od stvarnog vremena, što je presudno za evolucijske metode koje zahtijevaju tisuće evaluacija po pokretanju.

4.2. CityFlowCpp

Izvorni CityFlow distribuiran je kao Python modul i nije predviđen za izravnu integraciju u C++ projekte. Budući da je evolucijski okvir ECF implementiran u C++, za potrebe ovog rada razvijena je biblioteka *CityFlowCpp*: samostalna C++ biblioteka koja izlaže sučelje CityFlow simulatora na način koji je izravno upotrebljiv unutar C++ okruženja bez posrednog sloja u Pythonu.

Biblioteka je implementirana u standardnom C++ i distribuira se kao instalabilni paket putem CMake sustava (`find_package(CityFlowCpp)`), što omogućuje njezino uključivanje u bilo koji C++ projekt bez potrebe za ručnim upravljanjem putanjama i zavisnostima.

4.2.1. Arhitektura i višedretveno izvođenje

Jezgra biblioteke jest razred *Engine*, koji koordinira simulaciju kroz strogo definirani redoslijed faza unutar svake iteracije: generiranje vozila prema zadanom toku, planiranje ruta, upravljanje čekanjem na ulaz u traku, detekcija i razrješavanje prometnih konflikata na raskrižjima, izračun akcija vozila, ažuriranje pozicija te napredak semafora.

Simulacija je višedretvena: ceste, raskrižja i vozila raspoređuju se između dretvi pri inicijalizaciji. Sinkronizacija se provodi dvostrukom barijerom (*start/end barrier*) koja osigurava da svaka dretva završi svoju fazu prije nego što se pređe na sljedeću, čime se izbjegava konflikt pri pristupu podacima bez potrebe za stalnim zaključavanjem memorije.

4.2.2. Programsko sučelje biblioteke

Biblioteka *CityFlowCpp* izlaže programsko sučelje kroz razred *Engine*. Sučelje pokriva upravljanje simulacijom (`nextStep`, `reset`), postavljanje signalnih faza (`setTrafficLightPhase`), dohvat stanja prometne mreže (`getLaneVehicleCount`, `getLaneWaitingVehicleCount`) te prikupljanje metrika (`getAverageTravelTime`, `getFinalMetrics`). Upravljački sloj opisan u poglavlju 5. koristi metode `getLaneVehicleCount` i `getLaneWaitingVehicleCount` za izračun ulaznih varijabli W_0-W_3 i C_0-C_3 koje se prosljeđuju upravljačkom stablu.

4.2.3. Prošireni skup prometnih metrika

U odnosu na izvorni CityFlow, u biblioteci CityFlowCpp uveden je strukturirani razred `SimulationMetrics` koji na kraju simulacije izlaže cjelovit skup pokazatelja kvalitete upravljanja:

- *Prosječno i ukupno vrijeme putovanja* — od ulaska vozila u mrežu do izlaska,
- *Prosječno i ukupno vrijeme čekanja* — akumulirano dok je brzina vozila ispod praga od 0,1 m/s,
- *Prosječno i ukupno kašnjenje* — razlika između stvarnog vremena putovanja i teorijskog minimuma pri maksimalnoj brzini vozila:

$$\delta_i = t_i^{\text{stvarno}} - \frac{d_i}{v_i^{\text{max}}}, \quad (4.1)$$

gdje je t_i^{stvarno} stvarno vrijeme putovanja vozila i , d_i ukupna prijeđena udaljenost, a v_i^{max} njegova maksimalna brzina,

- *Propusnost (throughput)* — broj vozila koja su uspješno dovršila putovanje,
- *Maksimalna duljina repa čekanja* — najveći izmjereni rep čekanja na bilo kojem raskrižju kroz cijelu simulaciju.

Kašnjenje definirano izrazom (4.1) mjeri apsolutni gubitak vremena svakog vozila u odnosu na idealno putovanje pri maksimalnoj brzini.

4.2.4. Praćenje stanja na razini vozila

Za izračun metrika opisanih u prethodnom odjeljku, svakom vozilu dodijeljena su dodatna polja koja se ažuriraju inkrementalno u svakom simulacijskom koraku:

- `stoppedSince` — korak od kojeg vozilo stoji, koristi se za izračun trajanja čekanja,
- `totalDistanceTraveled` — ukupna prijeđena udaljenost, koristi se pri izračunu kašnjenja prema izrazu (4.1).

Ove vrijednosti koriste se pri uklanjanju vozila iz simulacije, čime se osigurava da

svako vozilo doprinese konačnim metrikama čak i u slučaju da nije dovršilo putovanje do kraja simulacije.

4.2.5. Praćenje maksimalnog repa čekanja

Maksimalna duljina repa čekanja prati se globalno kroz cijelu simulaciju. Svaki razred `TrafficLight` prima pokazivač na globalnu varijablu `maxQueueLength` u razredu `Engine`. Metoda `updateMaxQueue()` prolazi cestama koje ulaze u dotično raskrižje i broji uzastopna zaustavljenja vozila od kraja trake prema naprijed što odgovara stvarnoj definiciji repa čekanja kao uzastopnog niza nepomičnih vozila. Globalna varijabla ažurira se samo u smjeru povećanja, čime se pamti najveća izmjerena vrijednost kroz cijelu simulaciju.

4.3. Korišteni prometni scenariji

Eksperimenti su provedeni na stvarnoj prometnoj mreži iz kineskog grada Hangzhoua. Mreža obuhvaća 16 raskrižja sa semaforima raspoređenih u četiri retka i četiri stupca (*hangzhou_4x4_gudang*). Prometni tokovi preuzeti su iz javno dostupnog skupa podataka koji sadrži jedan sat stvarnih prometnih mjerenja [9]. Neujednačena raspodjela prometnog opterećenja, različita trajanja signalnih faza po raskrižjima te složeni obrasci kretanja vozila čine ovaj scenarij prikladnim za procjenu sposobnosti generalizacije evoluiranih upravljačkih funkcija.

4.4. Načini upravljanja simulacijom

Kako bi rezultati evolucije bili usporedivi s postojećim pristupima, u ovom radu definirani su različiti načini upravljanja simulacijom koji služe i kao referentne vrijednosti i kao osnova za evaluaciju evoluiranih rješenja.

Kružni način rada (*round-robin*) najjednostavniji je oblik upravljanja u kojemu simulator izmjenjuje signalne faze jednakim, unaprijed određenim redoslijedom bez ikakvih informacija o trenutnom prometnom toku. Ovaj pristup služi kao osnovna referentna vrijednost (*baseline*) s kojom se uspoređuju evoluirana rješenja.

Evolucijski način rada koristi stablo kao upravljačku funkciju. Stablo na temelju ulaz-

nih varijabli koje opisuju stanje prometa izračunava vrijednost urgentnosti za svaku fazu, a faza s najvećom vrijednošću dobiva zeleno svjetlo. Upravljačke funkcije dobivene evolucijom uspoređuju se s prethodno navedenom referentnom vrijednošću.

4.5. Ulazne varijable upravljačke funkcije

Za donošenje upravljačkih odluka, svaka upravljačka funkcija na ulazu dobiva informacije o trenutnom stanju prometnih traka za svaki pokret (*movement*) unutar razmatrane signalne faze. Pokret definira par ulazne i izlazne ceste na raskrižju. Za svaki pokret stablo se evaluira s osam ulaznih varijabli:

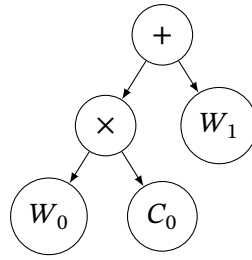
- W_0 — broj zaustavljenih vozila u dolaznim trakama pokreta,
- W_1 — broj zaustavljenih vozila u odlaznim trakama za lijevo skretanje,
- W_2 — broj zaustavljenih vozila u odlaznim trakama ravno,
- W_3 — broj zaustavljenih vozila u odlaznim trakama za desno skretanje,
- C_0 — ukupan broj vozila u dolaznim trakama pokreta,
- C_1, C_2, C_3 — ukupan broj vozila u odlaznim trakama (lijevo, ravno, desno).

Vozilo se smatra zaustavljenim ako mu je brzina manja od 0,1 m/s. Vrijednost urgentnosti faze dobiva se zbrajanjem izlaza stabla po svim pokretima te faze, a faza s najvećom ukupnom urgentnosti postaje aktivna, sukladno pristupu predloženom u [4].

Postupak je ilustriran na primjeru raskrižja s dvije faze (A i B), svaka s dva pokreta, i upravljačkim stablom $T(W_0, C_0, W_1) = W_0 \cdot C_0 + W_1$ (slika 4.1.). Varijabla W_0 predstavlja broj zaustavljenih vozila u dolaznim trakama pokreta, C_0 ukupan broj vozila u dolaznim trakama, a W_1 broj zaustavljenih vozila u odlaznim trakama za lijevo skretanje.

Za svaki pokret unutar faze stablo se evaluira sa specifičnim vrijednostima ulaznih varijabli toga pokreta. Tablica 4.1. prikazuje evaluaciju za konkretno stanje prometa u kojemu dolazne trake pokreta T_1 i T_2 (faza A) imaju znatno više zaustavljenih vozila nego pokret T_3 (faza B), dok pokret T_4 ima umjereno opterećenje.

Urgentnost faze A iznosi $17 + 6 = 23$, a faze B iznosi $1 + 8 = 9$, pa se aktivira faza A.



Slika 4.1. Primjer stabla: $W_0 \cdot C_0 + W_1$

Tablica 4.1. Primjer evaluacije stabla po pokretima

Pokret	Faza	W_0	C_0	W_1	$T = W_0 \cdot C_0 + W_1$
T_1	A	3	5	2	17
T_2	A	1	2	4	6
T_3	B	0	1	1	1
T_4	B	2	4	0	8

Stablo je prepoznalo da faza A zahtijeva prioritet jer dolazne trake njezinih pokreta sadrže više zaustavljenih vozila (W_0). Umnožak $W_0 \cdot C_0$ daje visoku vrijednost kada je i ukupan broj vozila i broj zaustavljenih velik, što ukazuje na zagušenje. Član W_1 dodatno uzima u obzir stanje odlaznih traka, čime stablo implicitno koordinira promet s nizvodnim raskrižjima.

4.6. Interval odlučivanja i tijek simulacije

Upravljačka funkcija ne donosi odluku u svakom vremenskom koraku simulacije, već periodički svakih k koraka, gdje je k parametar koji se naziva *interval odlučivanja*. Između dviju uzastopnih odluka semafor ostaje u trenutnoj fazi. U ovom radu interval odlučivanja postavljen je na 5 koraka, što odgovara 5 sekundi simuliranog vremena. Time je implicitno određeno minimalno trajanje zelene faze od 5 sekundi. Maksimalno trajanje faze nije ograničeno. Ako upravljačka funkcija u uzastopnim koracima odlučivanja odabire istu fazu, ona ostaje aktivna. Pri promjeni faze umeće se crveni interval od 5 sekundi kao prijelazno razdoblje.

Svaka simulacija provodi se na sljedeći način. Nakon inicijalizacije, simulator počinje izvoditi korake prometnog toka. Na svakom koraku koji je višekratnik intervala odlučivanja, upravljački sloj dohvaća trenutne vrijednosti ulaznih varijabli za svako raskrižje, evaluira odgovarajuću upravljačku funkciju i zadaje novu signalnu fazu. Po završetku

svih koraka simulacije, simulator vraća statistike prometnog toka, od kojih se koristi *prosječno vrijeme putovanja* kao mjera kvalitete upravljačke strategije. Ukupan broj koraka po simulaciji postavljen je na 3 600, što odgovara jednom satu simuliranog prometnog toka. Ova duljina odabrana je kako bi se uhvatio cjelokupni prometni scenarij te kako bi prosječno vrijeme putovanja bilo smisljena mjera kvalitete upravljanja.

4.7. Evaluacija u evolucijskom procesu

Svaka jedinka u populaciji evaluira se pokretanjem jedne kompletne simulacije odabranog prometnog scenarija. Upravljačke funkcije sadržane u jedinci koriste se za upravljanje svim raskrižjima u mreži tijekom cijele simulacije. Po završetku simulacije prosječno vrijeme putovanja vozila koja su završila prolazak kroz mrežu postaje vrijednost funkcije kazne te jedinke.

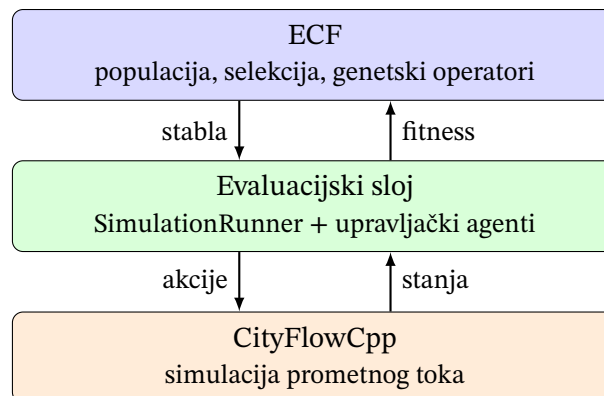
Budući da je simulator deterministički pri fiksnom slučajnom sjemenu, sve jedinke unutar iste generacije evaluiraju se pod identičnim prometnim uvjetima, čime je osigurana usporedivost rezultata. Isto sjeme koristi se kroz cijeli evolucijski proces kako bi poboljšanje funkcije kazne kroz generacije odražavalo stvarno napredovanje kvalitete upravljačkih funkcija, a ne varijacije nastale promjenom prometnih uvjeta.

5. Implementacija sustava

U ovom poglavlju opisana je arhitektura implementiranog sustava te način na koji su pojedine komponente međusobno povezane. Sustav se sastoji od tri glavne cjeline: biblioteke CityFlowCpp koja izvodi simulaciju prometnog toka, evolucijskog okvira ECF te upravljačkog sloja koji ih povezuje. Implementirana su tri načina rada koja odgovaraju trima pristupima opisanim u poglavlju 3.7.

5.1. Arhitektura sustava

Cjelokupni sustav organiziran je kao troslojna arhitektura (slika 5.1.). Na najnižoj razini nalazi se biblioteka CityFlowCpp koja izvodi simulaciju prometnog toka i izlaže programsko sučelje za dohvat stanja mreže i postavljanje signalnih faza. Srednji sloj čine upravljačke komponente implementirane u C++: razred `SimulationRunner` koji vodi tijekom jedne evaluacije i skup agenata koji na temelju stanja prometa donose upravljačke odluke za svako raskrižje. Na najvišoj razini nalazi se ECF koji vodi evolucijski proces: upravlja populacijom, primjenjuje genetske operatore te za svaku jedinku pokreće evaluaciju kroz srednji sloj.



Slika 5.1. Troslojna arhitektura sustava

Sva tri načina rada dijele iste niže razine (CityFlowCpp i agente), a razlikuju se samo u načinu organizacije populacije i evaluacijske logike koja se predaje ECF-u kao EvaluateOp objekt.

Prije pokretanja evolucije svaki način rada automatski mjeri referentnu vrijednost (*baseline*) pokretanjem simulacije u kojoj sva raskrižja koriste kružni (*round-robin*) raspored faza. Ova vrijednost sprema se uz rezultate svake evolucije i koristi se za usporedbu napretka.

5.2. Upravljački agent

Svako raskrižje sa semaforom upravlja se zasebnom instancom ImprovedGPLightAgent razreda. Agent evaluira dodijeljeno stablo za svaku signalnu fazu prema postupku opisanom u odjeljku 4.5. te aktivira fazu s najvećom vrijednošću urgentnosti.

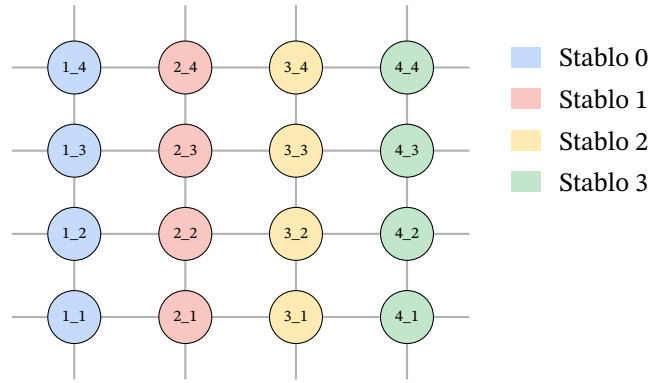
5.3. GP Light+ evaluator

U načinu rada GP Light+ svaka jedinka populacije sadrži jedno stablo koje se dodjeljuje svim agentima u mreži. Evaluacijska logika implementirana je u GPLightEvalOpBasic razredu. Za svaku jedinku evaluator dodjeljuje stablo svim agentima, pokreće jednu simulaciju putem razreda SimulationRunner te kao vrijednost funkcije kazne vraća prosječno vrijeme putovanja vozila koja su prošla kroz mrežu.

ECF se konfigurira s jednim blokom <Tree> unutar genotipa, a konfiguracija se automatski generira ovisno o odabranim parametrima (dubina stabla, veličina populacije, skup simbola). Generirani XML sprema se na disk i predaje ECF-u pri inicijalizaciji.

5.4. Grupirana koevolucija

U načinu rada grupirane koevolucije svaka jedinka populacije sadrži $\lceil N/g \rceil$ stabala, gdje je N ukupan broj raskrižja, a g veličina grupe (*GROUP_SIZE*). Stabla se dodjeljuju agentima prema redoslijedu kojim simulator vraća raskrižja: prvih g raskrižja dijeli prvo stablo, sljedećih g drugo, i tako dalje. Na korištenoj mreži Hangzhou 4×4 to rezultira grupiranjem po stupcima gdje stupac od četiri raskrižja dijeli jedno stablo.



Slika 5.2. Prometna mreža Hangzhou 4×4 s grupiranjem raskrižja po stupcima ($g = 4$). Raskrižja iste boje dijele jedno upravljačko stablo.

Evaluacijska logika implementirana je u razredu `CoevGPLightEvalOp`. Za svaku jedinku, evaluator dodjeljuje sva stabla odgovarajućim agentima, pokreće jednu simulaciju i kao vrijednost funkcije kazne vraća prosječno vrijeme putovanja vozila koja su prošla kroz mrežu.

5.5. Kooperativna koevolucija s odvojenim populacijama

U načinu rada kooperativne koevolucije svaka grupa od g raskrižja ima zasebnu populaciju koja evoluira jedno stablo. Evaluacijska logika implementirana je u razredu `TrueCoevEvalOp`. Pri evaluaciji jednog stabla, ono se dodjeljuje agentima svoje grupe, dok agenti ostalih grupa koriste trenutno najbolja stabla svojih populacija. Time svaka populacija dobiva izravniji selekcijski signal nego u grupiranoj koevoluciji.

Evolucija se odvija u rundama: u svakoj rundi redom se evoluira svaka populacija kroz S generacija, nakon čega se najbolje stablo sprema za korištenje u sljedećoj rundi. Na početku svake runde dio populacije inicijalizira se najboljim stablom iz prethodne runde (*warm start*). Ukupan broj efektivnih generacija po populaciji iznosi $S \times R$, gdje je R broj rundi.

5.6. Organizacija eksperimenata i pohrana rezultata

Svi eksperimenti automatski se numeriraju i pohranjuju u mapu `rezultati/experiment_NNN_<scenarij>/`, gdje je `NNN` globalni redni broj eksperimenta jedinstven kroz sve scenarije. Svaki eksperiment sadrži: zapisnik evolucije (`log.txt`),

cjelokupni ispis (`total_output.txt`), XML datoteke s najboljim stablima (`best_tree_*.xml`) te referentnu vrijednost (`baseline.txt`).

5.7. Parametri evolucijskog procesa

Parametri evolucijskog procesa zajednički su svim načinima rada i prikazani su u tablici 5.1.

Tablica 5.1. Parametri evolucijskog procesa

Parametar	Vrijednost
Veličina populacije	50
Maksimalan broj generacija	51
Operator selekcije	<i>SteadyStateTournament</i>
Veličina turnira	3
Vjerojatnost mutacije	0,1 / 0,5
Maksimalna dubina stabla	6
Minimalna inicijalna dubina	3
Skup terminalnih simbola	$W_0, W_1, W_2, W_3, C_0, C_1, C_2, C_3$, konst. $[-1, 1]$
Skup funkcijskih simbola	$+$, $-$, $\times$, $\div$
Uvjet stagnacije	30 generacija

Operator dijeljenja zaštićen je od dijeljenja nulom: u slučaju da je nazivnik nula, rezultat operacije postavlja se na jedan. Konstante iz skupa terminalnih simbola inicijaliziraju se kao ERC vrijednosti nasumično odabrane iz intervala $[-1, 1]$ pri stvaranju stabla i ostaju fiksne za tu instancu čvora kroz cijelu evoluciju. Osnovna vrijednost vjerojatnosti mutacije (0,1) preuzeta je iz rada [4]. Dodatno je ispitana i povećana vrijednost od 0,5 kako bi se procijenio utjecaj jače mutacije na kvalitetu konačnog rješenja.

Maksimalan broj generacija odnosi se na osnovne eksperimente. Za grupiranu koevoluciju dodatno je provedena evolucija kroz 150 generacija, a za kooperativnu koevoluciju evolucija se provodi kroz $S \times R$ efektivnih generacija po populaciji (odjeljak 5.5.).

Uvjet prijevremenog zaustavljanja aktivira se ako se vrijednost funkcije kazne najboljeg rješenja ne poboljša kroz 30 uzastopnih generacija, čime se sprječava nepotrebno produžavanje evolucije nakon što populacija konvergira.

6. Eksperimenti i rezultati

6.1. Eksperimentalna postava

Glavni eksperimenti provedeni su na scenariju Hangzhou 4×4. Svaki eksperiment pokrenut je s parametrima evolucijskog algoritma navedenima u tablici 5.1., s fiksnim slučajnim sjemenom kako bi rezultati bili usporedivi.

Testirane su četiri metode upravljanja:

- *Kružni agent (round-robin)*, referentna metoda koja izmjenjuje faze jednakim redoslijedom bez uvida u prometni tok,
- *GP Light+*, evolucija jednog dijeljenog stabla za sva raskrižja,
- *Grupirana koevolucija*, evolucija $[N/g]$ zajedničkih stabala unutar jedne populacije, s variranim parametrom veličine grupe $g \in \{1, 2, 4\}$,
- *Kooperativna koevolucija*, evolucija stabala u odvojenim populacijama za svaku grupu raskrižja, s $g = 4$.

Svaka jedinka evaluira se jednom kompletnom simulacijom od 3 600 koraka. Mjera kvalitete jest prosječno vrijeme putovanja svih vozila koja su dovršila putovanje u mreži.

6.2. Referentne vrijednosti

Kao referentna metoda koristi se kružni agent koji ciklički izmjenjuje signalne faze s jednakim trajanjem. Na scenariju Hangzhou 4×4 kružni agent postiže prosječno vrijeme putovanja od 1167,67 s. Budući da je simulator deterministički pri fiksnom slučajnom sjemenu, referentna vrijednost je jednaka u svim eksperimentima na istom scenariju.

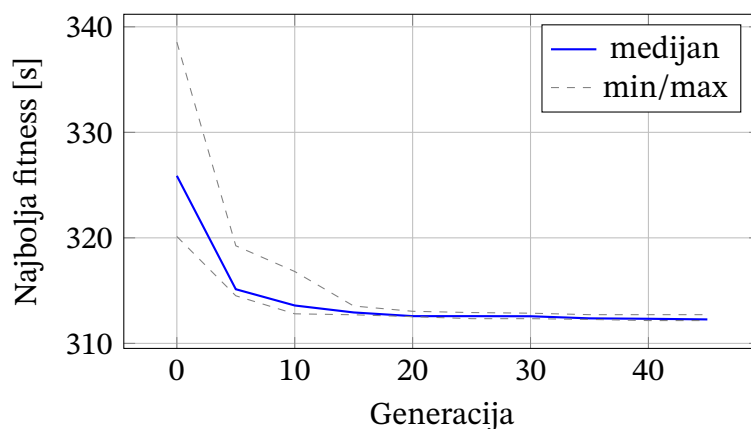
6.3. GP Light+

GP Light+ evoluira jedno dijeljeno stablo koje se dodjeljuje svim 16 raskrižja u mreži. Evolucija je provedena kroz 51 generaciju s populacijom od 50 jedinki. Tablica 6.1. prikazuje konvergenciju za jedno pokretanje, a slika 6.1. prikazuje raspon rezultata kroz 4 neovisna pokretanja s različitim slučajnim sjemenima.

Tablica 6.1. Konvergencija GP Light+ na scenariju Hangzhou 4×4

Generacija	Najbolja fitness [s]
0	319,99
5	315,32
10	314,22
15	313,54
20	313,04
25	312,92
30	312,86
40	312,72
51	312,72

Evolucija konvergira brzo u prvih desetak generacija te potom stagnira. Konačna vrijednost od 312,72 s ostvaruje poboljšanje od 73,2% u odnosu na kružnog agenta. Kroz 4 neovisna pokretanja konačni rezultati variraju između 312,16 s i 312,72 s, što upućuje na visoku stabilnost pristupa.



Slika 6.1. Konvergencija GP Light+ kroz 4 neovisna pokretanja. Puna linija prikazuje medijan, isprekidane linije raspon najboljeg i najlošijeg pokretanja.

Dodatno je ispitan utjecaj povećane mutacije (0,5) na GP Light+. Konačni rezultat iznosi 312,34 s, što je blago poboljšanje u odnosu na mutaciju 0,1 (312,72 s). Razlika je manja nego kod koevolucijskih pristupa, što je očekivano jer GP Light+ ima kompaktan prostor pretrage s jednim stablom.

6.4. Grupirana koevolucija

Grupirana koevolucija testirana je s tri vrijednosti parametra veličine grupe:

- $g = 1$ — svako raskrižje ima zasebno stablo, 16 stabala ukupno
- $g = 2$ — svaka dva raskrižja dijele jedno stablo, 8 stabala
- $g = 4$ — svaka četiri raskrižja dijele jedno stablo, 4 stabla

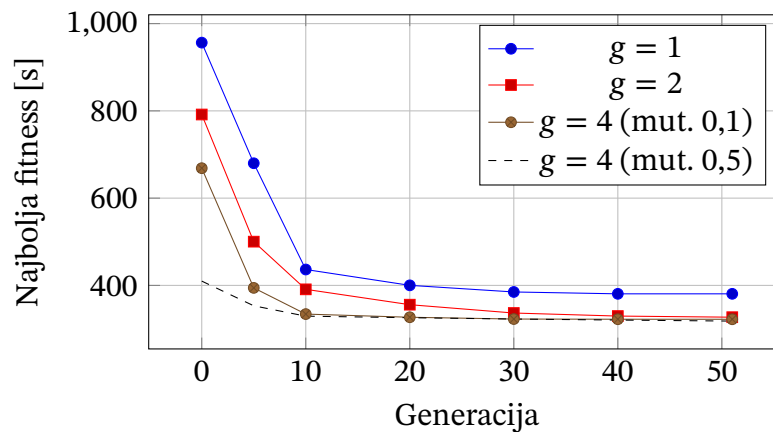
Tablica 6.2. i slika 6.2. prikazuje konvergenciju za sve tri konfiguracije.

Tablica 6.2. Konvergencija grupirane koevolucije na scenariju Hangzhou 4×4 za različite vrijednosti parametra g

Generacija	$g = 1$ [s]	$g = 2$ [s]	$g = 4$ [s]	$g = 4$, mut. 0,5 [s]
0	956,41	791,69	668,64	409,86
5	679,96	500,25	394,28	353,24
10	436,41	390,97	334,23	329,50
20	400,12	355,77	326,97	325,91
30	384,93	336,68	322,88	322,92
40	380,67	329,70	322,44	320,55
51	380,67	327,11	322,36	318,20

Može se uočiti nekoliko karakteristika. Sve tri konfiguracije kreću s višim početnim vrijednostima funkcije kazne nego GP Light+, što odražava veću složenost prostora pretrage uzrokovanu višestrukim genotipima. Konfiguracija $g = 1$ s 16 zasebnih stabala postiže najlošiji konačni rezultat (380,67 s), unatoč tome što ima najviši stupanj specijalizacije. Konfiguracija $g = 4$ postiže 322,36 s i najbliža je GP Light+ rezultatu. Dodatno je ispitan utjecaj vjerojatnosti mutacije na konfiguraciju $g = 4$. S povećanom mutacijom od 0,5 (umjesto osnovne 0,1) konačni rezultat nakon 51 generacije iznosi 318,20 s, što je

poboljšanje od 1,3% u odnosu na istu konfiguraciju s mutacijom 0,1 (322,36 s).



Slika 6.2. Konvergencija grupirane koevolucije za različite vrijednosti parametra g

Kako bi se ispitalo ponaša li se konfiguracija $g = 4$ drugačije uz veći broj generacija, provedena je dodatna evolucija kroz 150 generacija. Tablica 6.3. prikazuje konvergenciju tog pokretanja.

Tablica 6.3. Konvergencija grupirane koevolucije $g = 4$ kroz 150 generacija

Generacija	Najbolji fitness [s]
5	378,54
10	346,84
20	332,60
30	329,86
40	328,71
51	323,32
75	320,65
100	317,21
125	316,07
150	315,98

Za razliku od pokretanja s 51 generacijom, evolucija nije stagnirala već je nastavila poboljšavati rezultat sve do 150. generacije, dostižući 315,98 s.

6.5. Kooperativna koevolucija

Kooperativna koevolucija s odvojenim populacijama testirana je s veličinom grupe $g = 4$ (4 populacije, svaka upravlja jednim stupcem od 4 raskrižja). Evolucija je provedena kroz 8 rundi po 10 generacija (80 efektivnih generacija po populaciji), s veličinom populacije od 50 jedinki.

Tablica 6.4. prikazuje napredak najboljeg rješenja po rundama. U prvoj rundi rezultati su znatno lošiji od ostalih pristupa jer svaka populacija evoluira bez kvalitetnih partnera iz ostalih skupina. Već u drugoj rundi, kada su dostupna prva dobra stabla iz svih skupina, rezultat dramatično pada.

Tablica 6.4. Konvergencija kooperativne koevolucije ($g = 4$) po rundama

Runda	mut. 0,1 [s]	mut. 0,5 [s]
1	352,96	344,04
2	317,96	314,59
3	316,41	313,39
4	314,05	313,39
5	313,58	313,37
6	313,50	313,37
7	313,50	313,37
8	313,50	313,37

Konačna vrijednost od 313,50 s bolja je od svih varijanti grupirane koevolucije i praktički izjednačena s GP Light+ (312,72 s). Odvojene populacije omogućuju izravniji selekcijski signal za svako stablo, čime se ublažava problem pripisivanja zasluge.

Dodatno je ispitan utjecaj povećane mutacije (0,5) na kooperativnu koevoluciju. Konačni rezultat iznosi 313,37 s, što je blago poboljšanje u odnosu na mutaciju 0,1 (313,50 s).

6.6. Usporedba metoda

Tablica 6.5. prikazuje konačne rezultate svih testiranih metoda na scenariju Hangzhou 4×4.

Tablica 6.5. Usporedba prosječnog vremena putovanja vozila na scenariju Hangzhou 4×4

Metoda	Pros. vrijeme putovanja [s]	Poboljšanje vs. baseline
Kružni agent (round-robin)	1167,67	—
Grupirana koev. $g = 1$ (16 stabala)	380,67	67,4%
Grupirana koev. $g = 2$ (8 stabala)	327,11	71,9%
Grupirana koev. $g = 4$ (4 stabla)	322,36	72,4%
Grupirana koev. $g = 4$, mut. 0,5	318,20	72,8%
Grupirana koev. $g = 4$ (150 gen.)	315,98	72,9%
Koop. koevolucija $g = 4$	313,50	73,1%
Koop. koevolucija $g = 4$, mut. 0,5	313,37	73,2%
GP Light+ (1 stablo)	312,72	73,2%
GP Light+, mut. 0,5	312,34	73,3%

GP Light+ s povećanom mutacijom (0,5) ostvaruje najbolji apsolutni rezultat (312,34 s). Kooperativna koevolucija s mutacijom 0,5 praktički ga dostiže (313,37 s), dok grupirana koevolucija s manjim brojem stabala ($g = 4$) postiže rezultat bliži GP Light+ nego konfiguracija s većim brojem stabala ($g = 1$).

6.7. Testiranje robusnosti

Kako bi se procijenila sposobnost generalizacije evoluiranih upravljačkih funkcija na prometne uvjete koji nisu viđeni u procesu evolucije, provedeno je testiranje robusnosti perturbacijom prometnog toka. Za svaku perturbaciju, polazak svakog vozila pomaknut je za nasumičan iznos unutar zadanog vremenskog prozora $[-w, +w]$ sekundi, uz isti skup ruta. Testirane su perturbacije s prozorima $w \in \{30, 60, 120\}$ sekundi i dva različita slučajna sjemena po prozoru.

Tablica 6.6. Prosječno vrijeme putovanja [s] na originalnom i perturbiranim prometnim tokovima, Hangzhou 4×4

Perturbacija	Baseline	GP Light+	Koev. g=1	Koev. g=2	Koev. g=4	Koop. g=4
Original	1167,67	313,85	388,45	327,56	322,77	313,50
±30s s=1	1169,77	315,04	391,09	328,18	323,14	316,35
±30s s=2	1169,83	315,75	402,61	328,99	323,16	316,38
±60s s=1	1171,98	315,27	395,89	330,40	324,02	315,86
±60s s=2	1171,93	315,68	397,61	329,99	324,46	316,09
±120s s=1	1176,21	316,97	393,50	331,29	326,17	317,90
±120s s=2	1175,62	317,67	429,11	331,98	326,30	317,82
Maks. degr.	0,73%	1,22%	10,46%	1,35%	1,09%	1,38%

Većina evoluiranih strategija pokazuje visoku robusnost. Čak i uz perturbaciju od ±120 sekundi, degradacija u odnosu na originalni scenarij iznosi manje od 1,4% za GP Light+, grupiranu koevoluciju ($g = 2$ i $g = 4$) te kooperativnu koevoluciju. Poredak metoda ostaje nepromijenjen kroz sve perturbacije.

Najmanju degradaciju ostvaruje grupirana koevolucija s $g = 4$ (1,09%), a sličnu i GP Light+ (1,22%), što upućuje na to da i jedno zajedničko stablo i stabla na razini grupa pronalaze generičke strategije upravljanja koje nisu osjetljive na specifičan raspored polazaka vozila. Kooperativna koevolucija ostvaruje najbolji rezultat na originalnom scenariju (313,50 s), uz degradaciju od 1,38%.

Iznimka je konfiguracija $g = 1$ sa 16 zasebnih stabala koja pokazuje degradaciju do 10,46%. Velik broj stabala omogućuje prekomjerno prilagođavanje specifičnom prometnom scenariju, što rezultira znatno slabijom generalizacijom na perturbiranim tokovima.

6.8. Rasprava

Dobiveni rezultati pokazuju da GP Light+ s jednim dijeljenim stablom nadmašuje sve varijante grupirane koevolucije pri jednakom broju generacija i jednakoj veličini populacije. Ovaj rezultat može se objasniti složenošću prostora pretrage.

Grupirana koevolucija suočava se s bitno težim problemom pretrage. Svaka jedinka sadrži više stabala, pri čemu selekcijski pritisak djeluje na razini cjelokupne jedinice. Evaluacijski signal koji dobiva jedno stablo zapravo je mjera kvalitete cijelog tima stabala, što otežava usmjeravanje evolucije prema poboljšanju pojedinog stabla. Ovaj problem poznat je kao *problem pripisivanja zasluge* (*credit assignment problem*). Primjerice, u konfiguraciji $g = 1$ jedinka sadrži 16 stabala i ostvaruje fitness od 380,67 s. Iz te jedne vrijednosti nemoguće je utvrditi koje stablo pridonosi lošem rezultatu, a koje funkcionira dobro. S porastom broja stabala taj problem postaje izraženiji, što objašnjava zašto konfiguracija s najviše stabala ostvaruje najlošiji rezultat.

Nasuprot tome, GP Light+ evoluira jedno stablo u kompaktnom prostoru pretrage. Stablo mora naučiti generičku strategiju koja funkcionira za sva raskrižja, ali evolucija ima jasno definirani selekcijski signal i može brzo konvergirati.

Kooperativna koevolucija s odvojenim populacijama ublažava problem pripisivanja zasluge odvajanjem svake grupe u zasebnu populaciju. Svako stablo evaluira se zajedno s najboljim stablima ostalih populacija, čime dobiva izravniji signal o vlastitoj kvaliteti. Rezultat od 313,50 s (mutacija 0,1) potvrđuje ovu prednost, jer je znatno bolji od grupirane koevolucije s istim $g = 4$ (322,36 s) i praktički izjednačen s GP Light+ (312,72 s).

Povećanje vjerojatnosti mutacije s 0,1 na 0,5 poboljšava rezultate kod oba koevolucijska pristupa: grupirana koevolucija s $g = 4$ pada s 322,36 s na 318,20 s, a kooperativna koevolucija s 313,50 s na 313,37 s. Veća mutacija povećava raznolikost populacije, što je korisno u složenijim prostorima pretrage s višestrukim genotipima.

Produžena evolucija konfiguracije $g = 4$ kroz 150 generacija pokazuje da razlika između grupirane koevolucije i GP Light+ nije strukturalna. Fitness vrijednost pada kontinuirano do 315,98 s, što je samo 1% lošije od GP Light+ rezultata. Grupirana koevolucija može dosegnuti razinu GP Light+ uz dovoljno generacija, ali po cijenu znatno duljeg računalnog vremena.

Testiranje robusnosti pokazuje da je većina evoluiranih strategija otporna na perturbacije prometnog toka, s degradacijom manjom od 1,4%. Najmanju degradaciju pokazuje grupirana koevolucija s $g = 4$ (1,09%), a sličnu GP Light+ (1,22%) i kooperativna koevolucija (1,38%). Konfiguracija $g = 1$ s 16 stabala jedina pokazuje značajnu degra-

daciju (10,46%), što potvrđuje da velik broj stabala ne samo otežava evoluciju već vodi i prekomjernom prilagođavanju specifičnom prometnom scenariju.

7. Zaključak

U ovom radu istražena je primjena genetskog programiranja za dinamičko upravljanje prometnim semaforima u gradskoj mreži. Razvijena su i uspoređena tri pristupa različite složenosti: GP Light+ s jednim dijeljenim stablom, grupirana koevolucija s parametriziranim brojem stabala te kooperativna koevolucija s odvojenim populacijama.

Svi evolucijski pristupi značajno nadmašuju kružnog agenta na scenariju Hangzhou 4×4, ostvarujući poboljšanje prosječnog vremena putovanja između 67% i 73%. GP Light+ s jednim dijeljenim stablom postiže najbolji rezultat (312,72 s), dok ga kooperativna koevolucija s odvojenim populacijama praktički dostiže (313,37 s s mutacijom 0,5). Grupirana koevolucija s manjim brojem stabala ($g = 4$, 322,36 s) postiže rezultat bliži GP Light+ od konfiguracije s više stabala ($g = 1$, 380,67 s).

Ovi rezultati upućuju na ključni izazov koevolucijskog pristupa: s porastom broja genotipa unutar jedne jedinice raste i složenost prostora pretrage, a selekcijski signal koji dobiva svako stablo postaje neizravniji zbog problema pripisivanja zasluge. Kooperativna koevolucija ublažava taj problem odvajanjem svake grupe u zasebnu populaciju, čime se postiže rezultat usporediv s GP Light+. Povećanje vjerojatnosti mutacije s 0,1 na 0,5 dodatno poboljšava rezultate kod oba koevolucijska pristupa.

Testiranje robusnosti pokazuje da većina evoluiranih strategija zadržava učinkovitost i na perturbiranim prometnim tokovima, s degradacijom manjom od 1,4%. Iznimka je konfiguracija $g = 1$ koja pokazuje degradaciju do 10,46%, što dodatno potvrđuje da velik broj stabala vodi prekomjernom prilagođavanju.

Buduća istraživanja mogla bi uključiti alternativne strategije grupiranja raskrižja, paralelno izvođenje kooperativne koevolucije te evaluaciju na prometnim scenarijima iz drugih gradova.

Razvijeni sustav, koji objedinjuje biblioteku CityFlowCpp, evolucijski okvir ECF i tri implementirana načina rada, pruža fleksibilnu platformu za daljnje istraživanje evolucijskih metoda upravljanja prometom.

Literatura

- [1] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
- [2] H. Zhang, S. Feng, C. Liu, Y. Ding, Y. Zhu, Z. Zhou, W. Zhang, Y. Yu, H. Jin, i Z. Li, “CityFlow: A multi-agent reinforcement learning environment for large scale city traffic scenario”, u *The World Wide Web Conference*, 2019., str. 3620–3624. <https://doi.org/10.1145/3308558.3314139>
- [3] D. Jakobović, “ECF – evolutionary computation framework”, 2024. [Mrežno]. Adresa: <http://ecf.zemris.fer.hr>
- [4] X.-C. Liao, Y. Mei, i M. Zhang, “GPLight+: A genetic programming method for learning symmetric traffic signal control policy”, *IEEE Transactions on Evolutionary Computation*, str. 1–1, 2025. <https://doi.org/10.1109/TEVC.2025.3578575>
- [5] P. Shaikh, M. El-Abd, M. Khanafer, i K. Gao, “A review on swarm intelligence and evolutionary algorithms for solving the traffic signal control problem”, *IEEE Transactions on Intelligent Transportation Systems*, 2020. <https://doi.org/10.1109/TITS.2020.3014296>
- [6] F. V. Webster, “Traffic signal settings”, Road Research Laboratory, teh. izv. 39, 1958.
- [7] E. Ricalde, “A genetic programming system with an epigenetic mechanism for traffic signal control”, *CoRR*, sv. abs/1903.03854, 2019. [Mrežno]. Adresa: <http://arxiv.org/abs/1903.03854>
- [8] M. A. Potter i K. A. D. Jong, “A cooperative coevolutionary approach to function optimization”, u *Proceedings of the Third Conference on Parallel Problem Solving from*

Nature (PPSN III), 1994., str. 249–257. https://doi.org/10.1007/3-540-58484-6_269

- [9] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, W. Zhang, Y. Zhu, K. Xu, i Z. Li, “CoLight: Learning network-level cooperation for traffic signal control”, u *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019., str. 1913–1922. <https://doi.org/10.1145/3357384.3357902>

Sažetak

Dinamičko upravljanje simuliranim prometnim signalima evolucijskim algoritmima

Paula Močinić

U ovom radu istražuje se primjena genetskog programiranja za dinamičko upravljanje prometnim semaforima u gradskoj mreži. Svako raskrižje upravlja se matematičkim izrazom koji na temelju broja zaustavljenih vozila i ukupnog broja vozila u dolaznim i odlaznim trakama odlučuje kojoj signalnoj fazi dati prednost. Istražena su tri pristupa: GP Light+, koji evoluira jedno zajedničko stablo za sva raskrižja, grupirana koevolucija, koja dodjeljuje zasebna stabla grupama raskrižja unutar jedne populacije, te kooperativna koevolucija s odvojenim populacijama za svaku grupu. Eksperimenti su provedeni na stvarnoj prometnoj mreži iz Hangzhoua s 16 raskrižja. Svi evolucijski pristupi značajno nadmašuju kružnog agenta, ostvarujući poboljšanje prosječnog vremena putovanja između 67% i 73%. GP Light+ postiže najbolji rezultat od 312,72 s, dok ga kooperativna koevolucija praktički dostiže s 313,37 s. Testiranje robusnosti pokazuje da evoluirane strategije zadržavaju učinkovitost i na perturbiranim prometnim tokovima.

Ključne riječi: genetsko programiranje; upravljanje prometnim semaforima; koevolucija; evolucijsko računarstvo; CityFlow

Abstract

Dynamic control of simulated traffic signals using evolutionary algorithms

Paula Močinić

This thesis investigates the application of genetic programming to dynamic traffic signal control in urban road networks. Each intersection is controlled by a mathematical expression that determines signal phase priority based on stopped vehicle counts and total vehicle counts in inbound and outbound lanes. Three approaches are explored: GP Light+, which evolves a single shared tree for all intersections, grouped coevolution, which assigns separate trees to groups of intersections within a single population, and cooperative coevolution with separate populations for each group. Experiments are conducted on a real-world traffic network from Hangzhou with 16 intersections. All evolutionary approaches significantly outperform the round-robin baseline, achieving improvements in average travel time between 67% and 73%. GP Light+ achieves the best result of 312.72 s, while cooperative coevolution nearly matches it at 313.37 s. Robustness testing confirms that the evolved strategies maintain their effectiveness on perturbed traffic flows.

Keywords: genetic programming; traffic signal control; coevolution; evolutionary computation; CityFlow