

Zahvaljujem mentoru prof. dr. sc. Domagoju Jakoboviću na stručnom vodstvu, korisnim savjetima i pomoći tijekom izrade ovog diplomskog rada.

Zahvaljujem obitelji na svojoj podršci i razumijevanju tijekom cijelog studija.

*Zahvaljujem svojoj djevojci i prijateljima na podršci
tijekom izrade ovog rada.*

Sažetak

Optimizacija upravljanja prometnim signalima evolucijskim algoritmima

U ovom radu razvijen je sustav za stvaranje, učenje genetskim programiranjem i evaluaciju agenata koji upravljaju stanjem semafora na mreži raskrižja. Agenti su zasnovani na postojećoj ideji GPLight+, koja stablom matematičkog izraza procjenjuje korisnost pojedinog prometnog skretanja. Uz pojedinačne agente, istražene su i mogućnosti kombiniranja više agenata u ansamble, generirane četirima različitim postupcima odabira i dva načina donošenja odluke. Sustav je realiziran povezivanjem biblioteke ECF s vlastitom centralnom aplikacijom i našom novom bibliotekom CityFlowCpp, a svi agenti evaluirani su na mreži raskrižja HuangZhou4x4 prema prosječnom vremenu putovanja vozila. Rezultati pokazuju da implementirani GPLight+ agenti dosežu uspješnost usporedivu s referentnim radom, dok ansambli u prosjeku ne nadmašuju pojedinačne agente, premda pojedini primjeri ansambala ostvaruju bolje rezultate.

Ključne riječi: genetsko programiranje; upravljanje prometnim semaforima; ansambli; evolucijsko računarstvo; CityFlow; GPLight+

Summary

Optimization of Traffic Signal Control Using Evolutionary Algorithms

This thesis presents a system for creating, training with genetic programming and evaluating agents that control the traffic light state within a network of intersections. The agents are based on the existing GPLight+ method, which uses a tree-shaped mathematical expression to estimate the usefulness of each turning movement. In addition to individual agents, the possibility of combining several agents into ensembles was explored, generated by four different selection procedures and two decision-making methods. The system was implemented by connecting the ECF library with a custom central application and our CityFlowCpp library. All agents were evaluated on the HuangZhou4x4 intersection network bazen on the average travel time of vehicles. The results show that the implemented GPLight+ agents achieve performance comparable to the reference work, while ensembles on average do not outperform individual agents, although certain ensemble examples achieve better results.

Keywords: genetic programming; traffic signal control; ensembles; evolutionary computation; CityFlow; GPLight+

Sadržaj

Sažetak	1
Summary	2
Sadržaj	3
Uvod	5
1. Upravljanje semaforima	6
1.1. Raskrižje	6
1.2. Faze prometnog semafora	8
1.3. Upravljanje prometnim semaforom	9
1.3.1. Statički ciklus faza	9
1.3.2. Jednostavni reaktivni sustavi	9
1.3.3. Sustavi upravljani pomoću adaptivnih agenata	10
2. Metoda razvoja agenata	11
2.1. Prikaz agenta	11
2.2. Učenje agenata	11
2.2.1. Dobrota sustava za upravljanje prometom	12
2.3. Ansambli	13
3. GPLight+	14
4. Programska implementacija	17
4.1. ECF	18
4.2. CityFlowCpp	19
4.3. Centralna aplikacija	19
4.3.1. Stvaranje i evolucija agenata	20
4.3.2. Evaluacija i analiza naučenog agenta	20
4.3.3. Stvaranje ansambla	21

4.3.4.	Pomoćni sustav	22
5.	Eksperimenti i rezultat.....	23
5.1.	GPLight+.....	23
5.2.	Ansampli	24
5.3.	Referentni primjeri	25
5.4.	Rezultati	25
5.5.	Rasprava.....	28
6.	Zaključak	29
	Literatura	30
	Skraćenice	31

Uvod

Upravljanje prometnim semaforima u urbanim mrežama tradicionalno se temelji na unaprijed definiranim redoslijedima faza koji ne mogu reagirati na eventualne promjene prometnog toka. Ovakvi redoslijedi određeni su na temelju prosječnih podataka koji su dostupni za to specifično raskrižje ili ponekad bez prilagodbe na prosjek specifičnog raskrižja. Posljedica je neiskorišten kapacitet infrastrukture i povećano prosječno vrijeme putovanja, osobito u satima u kojima je povećan broj vozila na cesti i tijekom nepredviđenih prometnih situacijama.

Razvojem modernih računalnih sustava i senzora koji omogućuju prikupljanje i analizu podataka u realnom vremenu, otvorila se mogućnost prilagodbe stanja na semaforima, kao odgovor na promjene u prometu.

Genetsko programiranje, tehnika evolucijskog računarstva, nudi alternativu eksplicitnom inženjerskom projektiranju upravljačke logike. Evoluiranjem upravljačkih funkcija u obliku stabala matematičkih izraza moguće je automatski razviti strategije prilagođene konkretnim prometnim uvjetima, bez ručnog definiranja pravila odlučivanja.

U sklopu ovog rada razvijen je sustav koji omogućava stvaranje i učenje agenata za upravljanje stanjem na semaforu raskrižja. Razvijeni je veći broj agenata koji su trenirani na mreži raskrižja 4x4 simuliranim s pomoću modificirane biblioteke CityFlow. Također su isprobane metode odlučivanja koje uključuju više agenata odjednom. Osnovni agenti su razvijeni na temelju postojeće metode zvane GPLight+ [1].

Rezultati svih razvijenih agenata uspoređeni s postojećim agentima na istom skupu podataka, na temelju prosječnog vremena putovanja.

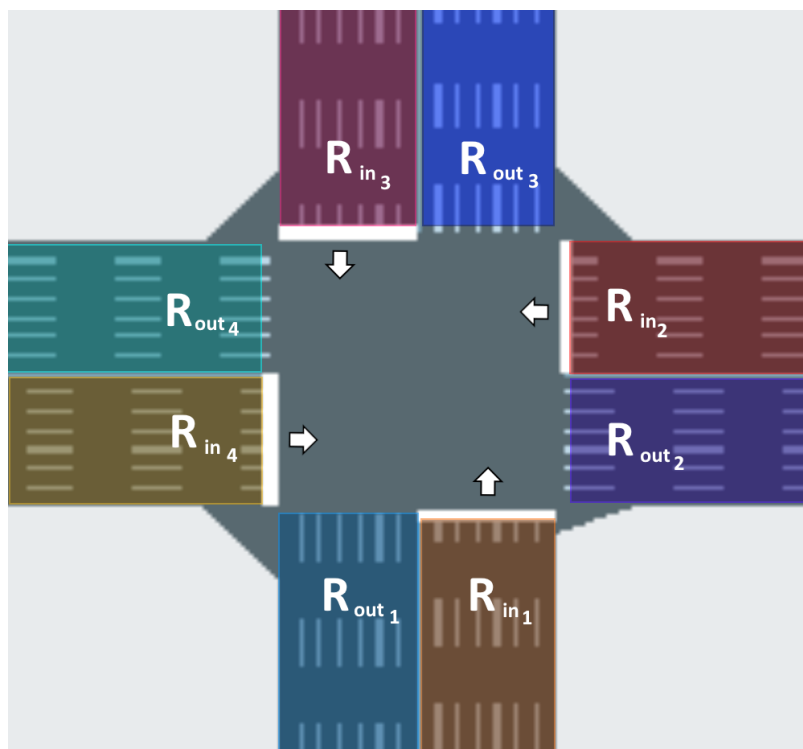
U prvom poglavlju objašnjena je problematika upravljanja raskrižjem i uvedeni su pojmovi potrebi za razumijevanje problema. U drugom poglavlju uvedeni su pojmovi i metode potrebne za razvoj agenata s pomoću genetskog programiranja. Treće i četvrto poglavlje opisuju konkretne implementacije korištene u istraživanju, dok su u petom poglavlju navedeni specifični eksperimenti i prikazani njihovi rezultati.

1. Upravljanje semaforima

Problem upravljanja semaforima moguće je definirati na više načina, ovisno o informacija na raskrižju koje su nam dostupne za donošenje odluke i koliko raskrižja je povezano u dijeljeni sustav. Također postoji više načina na koji te informacije mogu utjecati na stanje semafora. Raskrižja dolaze u različitim veličinama i oblicima, a nekad se pridržavaju različitih pravila odnosno o njihovom prometnom dizajnu. Za potrebe našeg istraživanja naš problem je definiram samo na jednom manjem podskupu raskrižja.

1.1. Raskrižje

Za potrebe našeg rada raskrižje I (eng. *Intersection*) je definirano kao točka u kojoj se susreće osam cesta R (eng. *Road*), ceste mogu biti ulazne (R_{in}) ili izlazne (R_{out}). U našem radu uvijek su korištene četiri R_{in} i četiri R_{out} , po jedan par iz svakog kardinalnog smjera (Sl. 1.1). Svaka cesta R sastoji se od nekoliko prometnih traka L (eng. *Lane*), koje mogu predstavljati jedno od tri moguća skretanja na kraju te ceste: lijevo (L_l), ravno (L_r) i desno (L_d). Svaka cesta R može imati različit broj traka L koje predstavljaju skretanje u istom smjeru.



Sl. 1.1 Oznake cesta koje sačinjavaju raskrižje

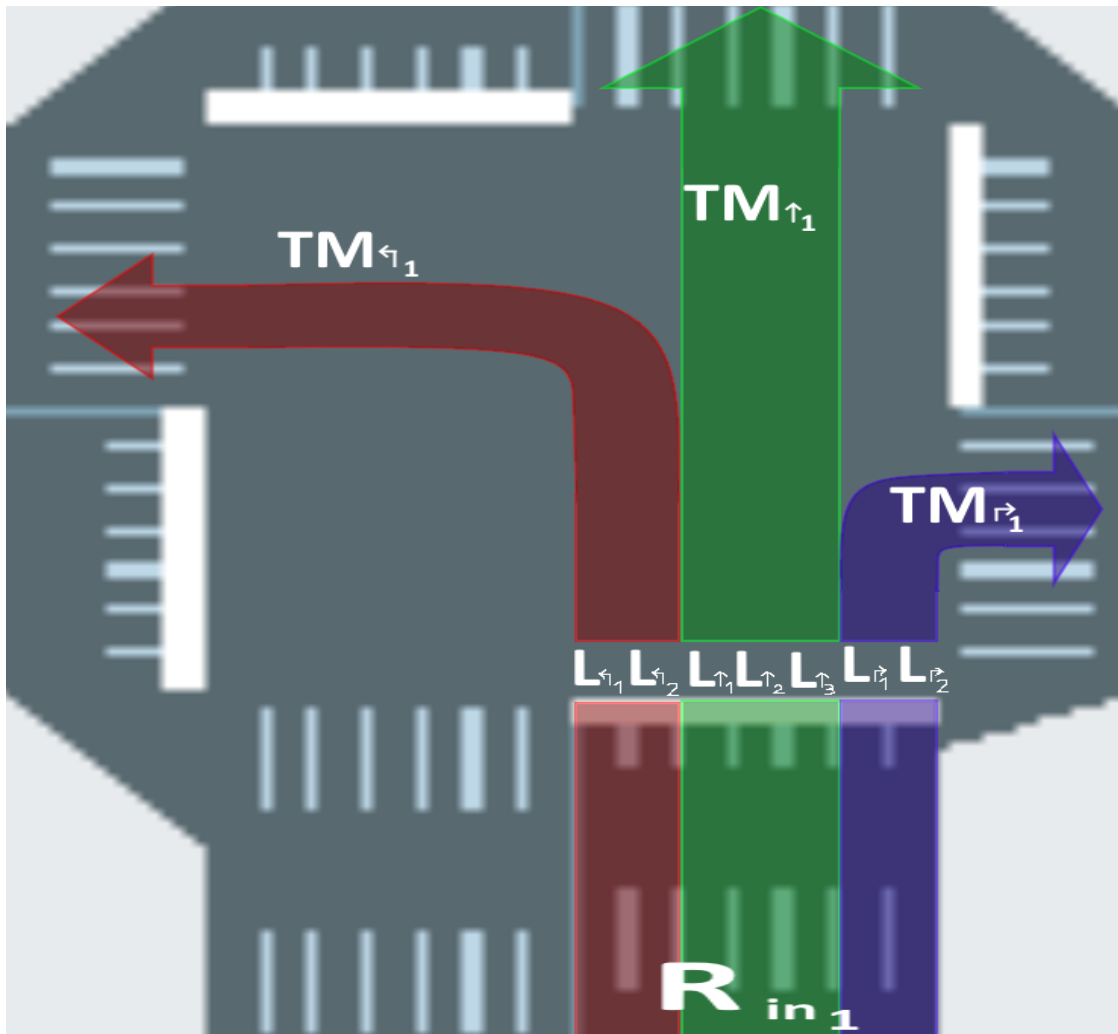
$$I = \{R_{in_1}, R_{in_2}, R_{in_3}, R_{in_4}, R_{out_1}, R_{out_2}, R_{out_3}, R_{out_4}\}$$

$$R = \{L_{\uparrow_1}, L_{\uparrow_2} \dots, L_{\uparrow_1}, L_{\uparrow_2} \dots, L_{\uparrow_1}, L_{\uparrow_2} \dots\}$$

$$P = \{TM_1, TM_2 \dots\}$$

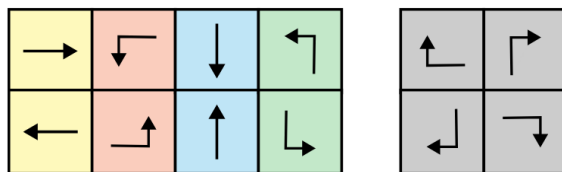
Faza prometnog semafora P (eng. *Phase*) definirana je kao skup dozvoljenih skretanja TM (eng. *Turn Movement*). Skretanje TM definirano je kao par skupa ulaznih prometnih traka L i izlazne ceste R . Skretanje TM je dozvoljeno kada semafor zelenim svjetlom dozvoljava prolaz iz ulaznih prometnih traka tog skretanja u izlaznu cestu tog skretanja. Na primjer, skretanje ulijevo, iz ceste R_{in_i} , u cestu R_{out_j} , gdje cesta R_{in_i} sadrži dvije $L_{\uparrow}$ bi definirali na ovaj način:

$$TM_{\uparrow_i} = \{\text{ulazna cesta: } R_{in_i}, \quad \text{ulazne trake: } \{L_{\uparrow_1}, L_{\uparrow_2}\}, \quad \text{izlazna cesta: } R_{out_j}\}$$



Sl. 1.2 Primjer mogućih skretanja TM koji počinju iz jedne ceste R

Ako odaberemo smjer iz kojeg promatramo raskrižje, moguće je odrediti da ukupno postoji 12 različitih vrsta TM (Sl. 1.3).

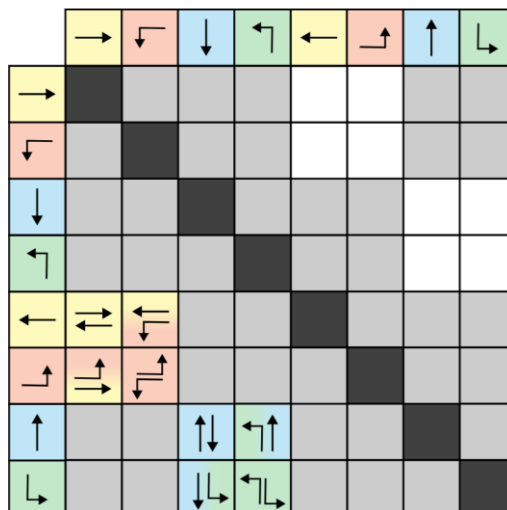


Sl. 1.3 Grafički prikaz 12 mogućih vrsta *TM* [1]

1.2. Faze prometnog semafora

Faza prometnog semafora ne smije sadržavati TM koji svojom putanjom u raskrižju prelazi putanju drugog TM. Ovo pravilo je preuzeto iz originalnog GPLight+ rada [1], kako bi usporedba rezultata bila moguća. Skretanja koja sadrže L_r u klasičnom raskrižju (Sl. 1.1), nikad ne sijeku druge TM. Iz tog razloga svaka faza *P* podrazumijevano sadrži sva četiri TM koji definiraju skretanje udesno iz svoje definirane ulazne ceste R_{in} . Nakon skretanja u desno, u svaku fazu *P* moguće je dodati do maksimalno dva TM prije nego dobivamo presjek u njihovoj putanji.

Ukupno postoji osam validnih kombinacija (Sl. 1.4) te samim time osam faza između kojih možemo birati kod propuštanja vozila.



Sl. 1.4 Moguće dozvoljene kombinacije *TM* (bez prikazanih desnih *TM*) [1]

Također semafori imaju i definirane faze koje koriste za pražnjenje vozila iz prostora raskrižja, u našem slučaju definirat ćemo ovu fazu s četiri desna TM dozvoljena i svim ostalim TM zatvorenima [1]. Korištene faze i njihove oznake nalaze u tablici (Tablica 1.1).

Tablica 1.1 Faze prometnog raskrižja korištene u ovom radu

Oznaka faze	Korišteni TM
P_1 (crveno)	$TM_{\uparrow_1}, TM_{\uparrow_2}, TM_{\uparrow_3}, TM_{\uparrow_4}$
P_2	$TM_{\uparrow_1}, TM_{\uparrow_2}, TM_{\uparrow_3}, TM_{\uparrow_4}, TM_{\downarrow_1}, TM_{\downarrow_3}$
P_3	$TM_{\uparrow_1}, TM_{\uparrow_2}, TM_{\uparrow_3}, TM_{\uparrow_4}, TM_{\downarrow_2}, TM_{\downarrow_4}$
P_4	$TM_{\uparrow_1}, TM_{\uparrow_2}, TM_{\uparrow_3}, TM_{\uparrow_4}, TM_{\downarrow_1}, TM_{\downarrow_3}$
P_5	$TM_{\uparrow_1}, TM_{\uparrow_2}, TM_{\uparrow_3}, TM_{\uparrow_4}, TM_{\downarrow_2}, TM_{\downarrow_4}$
P_6	$TM_{\uparrow_1}, TM_{\uparrow_2}, TM_{\uparrow_3}, TM_{\uparrow_4}, TM_{\downarrow_1}, TM_{\downarrow_3}$
P_7	$TM_{\uparrow_1}, TM_{\uparrow_2}, TM_{\uparrow_3}, TM_{\uparrow_4}, TM_{\downarrow_2}, TM_{\downarrow_4}$
P_8	$TM_{\uparrow_1}, TM_{\uparrow_2}, TM_{\uparrow_3}, TM_{\uparrow_4}, TM_{\downarrow_3}, TM_{\downarrow_1}$
P_9	$TM_{\uparrow_1}, TM_{\uparrow_2}, TM_{\uparrow_3}, TM_{\uparrow_4}, TM_{\downarrow_4}, TM_{\downarrow_2}$

1.3. Upravljanje prometnim semaforom

1.3.1. Statički ciklus faza

Najjednostavniji pristup regulaciji prometa je imati predefiniran ciklus faza i njihovih trajanja koji se ponavlja bez ikakvih promjena neovisno o vanjskim uvjetima. Ovaj pristup omogućuje da imamo dovoljno dobar protok vozila, neovisno o dostupnosti podataka za tok vozila na tom raskrižju. Zbog svoje cikličke prirode ovakvi sustavi su vrlo predvidivi što je poželjno u prometu, jer osobe koje upravljaju vozilima imaju tendenciju reagirati na temelju vlastitih navika na promet u kojem su često, te bi promjenjiv redoslijed faza mogao izazvati zbunjenost [2].

1.3.2. Jednostavni reaktivni sustavi

Statičke cikluse moguće je prilagoditi jednostavnim pravilima koja ne zahtijevaju poseban razvoj agenata te su vrlo jasne za ljudsko razumijevanje. Moguće je koristiti jedan ciklus tijekom cijelog dana, i za vrijeme predviđenih gužvi koje se dešavaju u podjednaka vrijeme svakog dana, uključiti drugi ciklus koji daje veću prednost opterećenijim cestama. Također je moguće i koristiti jednostavan senzor koji nakon što prepozna zastoje dovoljne duljine, produži trajanje adekvatne faze [3]. Ovakve metode su jednostavne za implementaciju i nude poboljšanje osnovnog principa, ali i dalje su ograničene u svojoj prilagodljivosti na nepredviđene situacije.

1.3.3. Sustavi upravljani pomoću adaptivnih agenata

Uz pristup prošlim podacima o prometu na nekom raskrižju, moguće je stvoriti i naučiti agente koji će razviti vlastita pravila za upravljanje fazama u prometu. Agenti mogu upravljati fazama na različite načine, poput promjene trajanja faza ili održavanje postojećeg ciklusa, ili promjene samog redoslijeda faza. Ovaj pristup ovisi o postojećim podacima potrebnim za učenje, dostupnosti adekvatnih kamera i senzora na raskrižjima kako bi sustav mogao biti korišten na pravom raskrižju, također sustav je potrebno dizajnirati tako da poštuje pravila prometa i da ne stvara nesigurne situacije, poput dozvoljavanja preklapanja TM ili puštanja novog TM u optjecaj prije nego se raskrižje stiglo isprazniti. U korištenim agentima obavezno pravilo kojeg su se morali držati je održavanje pet sekundi faze s crvenim svjetlima između svake promjene faza i minimalno trajanje prometne faze od barem 5 sekundi. Sustavi koji koriste ovakve agente mogu reagirati na situacije koje nisu bile predviđene prije postavljanja sustava i nemaju limit kompleksnosti kao kod pravila koja su ručno definirana. Model sustava adaptivnih agenata na kojima je baziran ovaj rad je GPLight+ [1].

2. Metoda razvoja agenata

2.1. Prikaz agenta

U ovom radu fokus je na agente koji svoje odluke baziraju na stablima koja predstavljaju matematičku funkciju ulaznih podataka i konstanti. Sintaksno stablo koje predstavlja matematičku funkciju uređeno je korijensko stablo u kojem svaki čvor pripada jednoj od dvije kategorije ovisno o svojoj ulozi u izrazu. Unutarnji čvorovi nose oznaku matematičke operacije ili funkcije, poput zbrajanja, množenja, sinusa ili eksponencijalne funkcije. Svaki takav čvor ima dvoje djece. Listovi su čvorovi bez djece i predstavljaju elementarne vrijednosti izraza. List može biti ulazna varijabla, odnosno parametar funkcije, ili numerička konstanta. Skup svih listova tako se prirodno dijeli na varijable i konstante. Evaluacija stabla provodi se rekurzivno od listova prema korijenu. Listovi se izravno interpretiraju kao svoje vrijednosti — varijabla kao trenutna vrijednost tog parametra, konstanta kao fiksni broj. Unutarnji čvorovi evaluiraju se primjenom svoje operacije na već izračunate vrijednosti djece. Konačna vrijednost cijelog izraza jednaka je vrijednosti korijenskog čvora nakon što je ovaj postupak u potpunosti proveden. Na taj način svako sintaksno stablo jednoznačno definira jednu matematičku funkciju čiji su argumenti upravo varijable koje se pojavljuju u njegovim listovima [4].

2.2. Učenje agenata

Učenje agenata je provedeno evolucijskim algoritmom. Evolucijski algoritmi su metaheuristike temeljene na idejama prirodnih selekcija i prenošenja gena. Učenje započinje generiranjem populacije napunjene nasumičnim agentima, koji će iz generacije u generaciju temeljem vlastite dobrote biti selektirani za prijenos gena u novu populaciju. Stvaranje početne populacije moguće je obaviti na više načina, o kojima ovisi kakvog će oblika biti početne jedinke. U slučaju stabla pričamo o rasporedu listova, hoće li se nalaziti na maksimalnoj dubiti ili će biti raspoređeni po više dubina. Nakon što je poznata dobrota svih jedinki u populaciji, potrebno je odraditi selekciju jedinki koje ćemo koristiti za generiranje nove populacije. Potrebno je odabrati adekvatan oblik selekcije kako ne bismo previše ograničili potencijalni prostor rješenja ili nepotrebno proširili. Jedan od čestih oblika selekcije je turnirska selekcija koja uzima najbolju od k jedinki da postane roditelj. Nakon

odabira svih roditelja potrebno je uzeti njihova svojstva i kombinirati ih putem križanja. Križanje uzima dio genetskog materijala od svakog roditelja i zatim ih kombinira kako bi stvorio novu jedinku. U slučaju stabla to se često provodi na načina da se u određenoj točki zamjene podstabla oba roditelja, kako bi se stvorile dvije nove jedinke. Kako bi istražili prostor rješenja koji nije prikazan u roditeljima, koristimo mutaciju. Pomoću mutacije mijenjamo vrijednosti listova stabla, mičemo rane ili ih mijenjamo novo generiranim.

2.2.1. Dobrota sustava za upravljanje prometom

Kako bi mogli odabrati jedinke kod učenja, potrebno im je odrediti dobrotu. U slučaju upravljanja prometom, često se koriste ovi:

- Prosječno vrijeme putovanja - srednje vrijednosti trajanja vožnje svih vozila, vrijeme počinje u trenutku kada se vozilo uključi u prometnu mrežu, staje kada iz nje izađe ili simulacija stane [1]
- Propusnost - broj vozila, koja je uspješno izašla iz prometne mreže [1]
- Prosječno kašnjenje - razlika između teoretskog minimalnog vremena potrebnog za prolaz kroz prometnu mrežu i stvarnog vremena [5]
- Prosječna duljina kolone - Prosječni broj vozila u prometnoj traci tijekom crvenog svjetla [1]

U ovom radu, za potrebe učenja agenata kao dobrotu smo koristili prosječno vrijeme putovanja *ATT* (eng. *Average Travel Time*), ali je za potrebe usporedbe zabilježena propusnost *NT* (eng. *Network Throughput*).

2.3. Ansambli

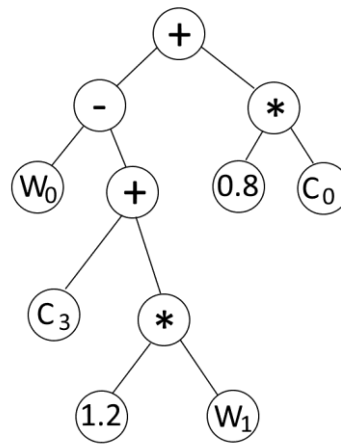
Uz pretpostavku da svaki agent predstavlja jedno kompleksno pravilo, moguće je napraviti ansambl agenata koji će zajedno donijeti odluku. Ansambli u strojnom učenju su modeli koji kombiniraju više pojedinačnih modela kako bi postigli bolju prediktivnu točnost nego što bi je postigao bilo koji od tih modela samostalno. Osnovna intuicija iza ovog pristupa jest da različiti modeli griješe na različitim primjerima, pa njihovim kombiniranjem greške pojedinih modela međusobno se poništavaju ili umanjuju. U ovom radu promatrano je ponašanje jednostavnih kombiniranih ansambila. SEC (eng. *Simple Ensemble Combination*) je najosnovniji oblik kombiniranja modela u ansamblu, gdje se predikcije svih članova ansambila kombiniraju na jednostavan i ujednačen način.

Ovisno o vrsti problema, kombiniranje se provodi na više načina. Za klasifikacijske probleme koristi se većinsko glasovanje, gdje svaki model "glasa" za jednu klasu, a kao konačna predikcija odabire se ona klasa koju je predložio najveći broj modela. Također je moguće odrediti prioritet mogućih akcija za svakog agenta i zatim analizom svih poredaka odabrati adekvatnu akciju.

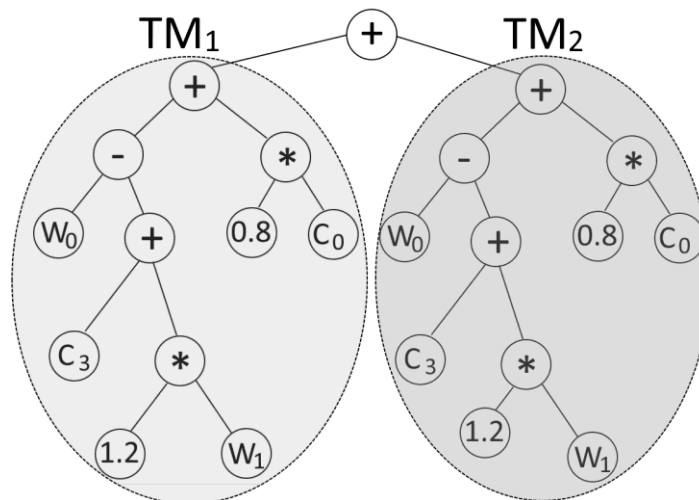
Ključna karakteristika SEC-a jest da svi modeli imaju jednaku težinu u konačnoj odluci, nijedan model nije privilegiran niti se sustav ne prilagođava podacima kako bi naučio kojim modelima više vjerovati. Samim time moguće je naučiti te agente potpuno odvojeno jedne od drugih [6].

3. GPLight+

Osnovni agenti u ovom radu bazirani su na modelu predstavljanom u radu GPLight+ [1]. S obzirom na to da svaka faza sadrži točno šest TM (četiri $\uparrow$ i dva koji određuju propusnost kroz raskrižje), najjednostavniji pristup problemu upravljanja faza semafora bio bi koristiti jedno stablo da procijenimo stanje za cijelu prometnu fazu. Ulazni parametri za ovakvo stablo bi obuhvaćali stanje na cesti za svaki TM u toj fazi. Ali to bi potencijalno značilo da se različiti TM tretiraju različito. GPLight+ pristup predlaže korištenje jednog stabla za izračun korisnosti jednog TM , i zatim sumiranje takvih vrijednosti za sve TM u jednoj fazi.



Sl. 3.1 primjer stabla koje razvijamo tijekom učenja



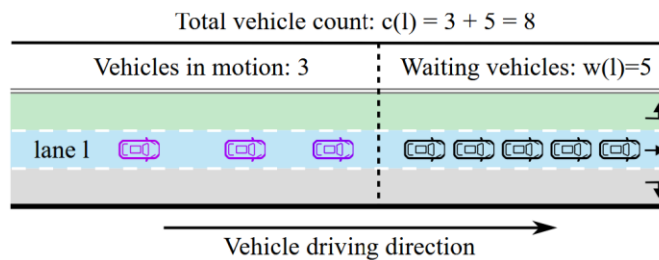
Sl. 3.2 Stablo koje se koristi za izračun korisnosti faze na temelju više TM

Ukoliko se jedan TM nalazi u svim mogućim fazama, u našem slučaju sva skretanja desno, on nije dodan u izračun jer bi jednako pridonosio svim fazama, što smanjuje količinu izračuna kod evaluacije stabala. Ovakav pristup bi osigurao da se svi bitni TM tretiraju jednako unutar faze, i da je potrebno razviti jednostavniju strukturu kod učenja, što bi također ubrzalo samo učenje.

Parametri koji se koriste za dobivanje važnosti jednog TM su prikazani u tablici (Tablica 3.1). Oznaka W predstavlja broj vozila koja trenutno stoje u promatranoj traci, dok oznaka C predstavlja ukupni broj vozila na odabranoj traci (Sl. 3.3).

Tablica 3.1 Opis parametara koji se koriste u GPLight+ stablu

Parametar	Oznaka ceste	Oznaka trake	Opis
W_0	R_{in}	L_{in}	Broj zaustavljenih vozila na ulaznoj traci u TM
W_1	R_{out}	$L_{\swarrow}$	Broj zaustavljenih vozila na izlaznoj traci u TM za skretanje u lijevo
W_2	R_{out}	$L_{\uparrow}$	Broj zaustavljenih vozila na izlaznoj traci u TM za nastavak ravno
W_3	R_{out}	$L_{\searrow}$	Broj zaustavljenih vozila na izlaznoj traci u TM za skretanje u desno
C_0	R_{in}	L_{in}	Ukupni broj vozila na ulaznoj traci u TM
C_1	R_{out}	$L_{\swarrow}$	Ukupni broj vozila na izlaznoj traci u TM za skretanje u lijevo
C_2	R_{out}	$L_{\uparrow}$	Ukupni broj vozila na izlaznoj traci u TM za nastavak ravno
C_3	R_{out}	$L_{\searrow}$	Ukupni broj vozila na izlaznoj traci u TM za skretanje u desno
EPM	-	-	Konstanta, $EPM \in [-1,1]$



Sl. 3.3 primjer izračuna parametara W i C [1]

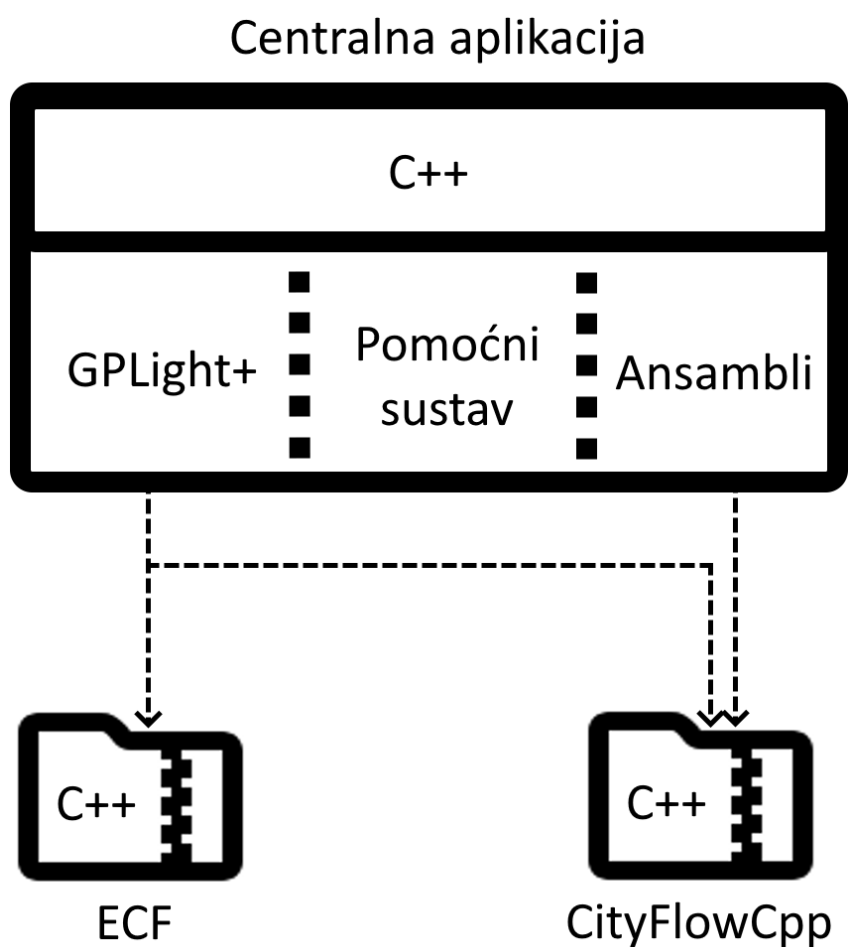
Zbog jednostavnosti prikaza, na cestama koje imaju više traka, koje koriste za skretanje u istom smjeru, vrijednosti takvih traka će biti zbrojene i tretirane kao jedna traka.

$$L_{\eta} = L_{\eta_1} + L_{\eta_2} + L_{\eta_3} \dots$$

Vrijednosti W i C računamo za četiri trake po jednom TM. Prva traka L^{in} je ulazna traka u raskrižje koja dobiva zeleno svjetlo ako odaberemo TM. Sljedeće tri trake ($L_{\eta}^{out}, L_{\uparrow}^{out}, L_{\downarrow}^{out}$) su sve moguće opcije skretanja iz izlazne ceste. Posljednji parametar koji ovakvo stablo prima je težina, vrijednost između $[-1,1]$ koja se koristi kao list za potrebe davanja težine ostalim listovima. Operacije koje se mogu provoditi nad vrijednostima su ograničene na skup $[+, -, *, /]$, pri čemu je dijeljenje osigurano od dijeljenja s nulom.

4. Programska implementacija

Kako bi dobili završni ansambl agenata, potrebno je koristiti nekoliko odvojenih sustava koji nam omogućuju učenje, evaluaciju i manipulaciju agentima. Sustav se sastoji od biblioteke ECF, koja nam omogućuje stvaranje početnih agenata, i učenje istih kroz zadani broj generacija. CityFlowCpp biblioteka se koristi kao simulator prometa. Zatim centralna aplikacija za definiranje agenata i njihovo korištenje koja koristi prethodne biblioteke (Sl. 4.1).



Sl. 4.1 Shema potpunog sustava korištenog tijekom ovog rada

4.1. ECF

ECF je C++ biblioteka, razvijena na FER-u, namijenjena primjeni bilo koje vrste evolucijskog računarstva [8]. Razvijen je kao svestrani alat otvorenog koda koji nudi puno implementiranih algoritama, raznolike genotipove i ono najbitnije, PnP komponente. U biblioteci ECF [9] moguće je odrediti sva svojstva potrebna za učenje stabla koja naši agenti koriste, putem konfiguracijske datoteke (Sl. 4.2).

```
<ECF>
  <Genotype>
    <Tree>
      <Entry key="terminalset">W0 W1 W2 W3 C0 C1 C2 C3 [-1 1]</Entry>
      <Entry key="functionset">+ - / *</Entry>
      <Entry key="maxdepth">6</Entry>
      <Entry key="initmindepth">3</Entry>
    </Tree>
  </Genotype>
  <Algorithm>
    <SteadyStateTournament>
      <tsize> 3 </tsize>
    </SteadyStateTournament>
  </Algorithm>
  <Registry>
    <Entry key="population.size">100</Entry>
    <Entry key="term.maxgen">51</Entry>
    <Entry key="mutation.indprob">0.1</Entry>
    <Entry key="population.demes">1</Entry>
    <Entry key="term.stagnation">30</Entry>
    <Entry key="term.fitnessval">0</Entry>
    <Entry key="batch.repeats">0</Entry>
    <Entry key="log.filename">log.txt</Entry>
    <Entry key="log.level">4</Entry>
  </Registry>
</ECF>
```

Sl. 4.2 Primjer konfiguracijske datoteke – učenje GPLight+ agenata kao u originalnom radu

Za naše potrebe ECF treba izgraditi stabla, i evoluirati ih na temelju funkcije dobrote koju mu predaje centralna aplikacija. ECF sam po sebi ne radi s našim agentima za upravljanje semaforom, već samo evaluira njihovo stablo, dok se primjena i evaluacija tog stabla obavlja odvojeno, što omogućuje vrlo lagano odvajanje komponenti u projektu i zamjenu s drugim komponentama u daljnjem razvoju. Samim time je i zadavanje konfiguracijske datoteke odrađeno kroz centralnu aplikaciju kako bi se bez potrebe za mijenjanjem načina na koji je ECF povezan s ostatkom sustava, mogle zadati nove vrste genotipa ili različiti načini evolucije.

4.2. CityFlowCpp

CityFlowCpp je C++ biblioteka razvijena za potrebe ovog rada kao simulator prometa koji je moguće lagano implementirati i koristiti u skladu s ostatkom sustava [9]. Jezgra biblioteke je uzeta iz originalnog CityFlow projekta [10], koji je namijenjen kao Python biblioteka sa C++ jezgrom. Originalna jezgra je uzeta i omotana u format koji odgovara c++ biblioteci, te je u nju dodano nekoliko funkcija za dohvat stanja u prometu, kako bi se iskoristile prednosti koje CityFlow nudi nativno. Implementacija ovih funkcionalnosti u centralnoj aplikaciji bi nepotrebno često okretala iste podatke te znatno usporila proces učenja i evaluacije agenata.

Biblioteka i dalje nudi vrlo jednostavne mogućnosti upravljanja prometom kako bi se u budućnosti pravila vezana uz agente ili stvarni svijet mogla implementirati, bez potrebe za promjenama na ovoj biblioteci. Mreža prometnih raskrižja sa svim svojim svojstvima može biti definirana preko dvije konfiguracijske datoteke, od kojih prva u sebi definira izgled mreže, cesta, linija i mogućih faza na semaforima, a druga opisuje način na koji se generira željeni tok vozila. Ove konfiguracijske datoteke se proslijeđuju kroz centralnu aplikaciju kod inicijalizacije jezgre simulatora, kako bi se ostavila mogućnost promjena toka za potrebe testiranja ili vršenja različitih testova bez potrebe za ponovnim instaliranjem biblioteke.

4.3. Centralna aplikacija

[12]Centralna aplikacija ima nekoliko načina rada, jer iako načini imaju različitu funkciju, koriste ostale dijelove sustava na jednak način pa je smisleno ih objediniti na jednom mjestu. Centralna aplikacija nam omogućuje definiranje novih vrsta agenata, evoluciju agenata, evaluaciju pojedinih agenata i kasniju analizu njihovog ponašanja s detaljnijim zapisima i vizualnim prikazom. Također nam omogućuje stvaranje ansambala na temelju svih agenata koji su naučeni od postavljanja aplikacije, neovisno o njezinom prekidu rada u međuvremenu.

4.3.1. Stvaranje i evolucija agenata

U centralnoj aplikaciji za potrebe našeg rada definirana su tri vrste upravljanja semafora, GPLight+ agent, upravljač s cikličkim promjenama faza i ansambl. Svi upravljači poštuju određena pravila koja su ugrađena u njih. Različite faze moraju biti odvojene fazom u kojoj su samo desna skretanja dozvoljena punih 5 sekundi kako bi se raščistilo raskrižje. GPLight+ agent i ansambl imaju opciju više puta za redom odabrati istu fazu te se u tome slučaju vrijeme koje bi bilo potrošeno na crvenu fazu, biti dodano trajanju trenutne faze. Kako bi saznali dobrotu određenog agenta potrebno je jednom pokrenuti simulaciju prometa u CityFlowCpp simulaciji, te iz nje na kraju zatražiti stanje prometa i vozila koja su sudjelovala u njemu. Na temelju tih vrijednosti moguće je izračunati dobrotu. U ovom slučaju kao dobrota je korišteno prosječno trajanje vožnje (ATT), koji CityFlowCpp nudi bez potrebe za novim izračunom. Manji ATT označava boljeg agenta. Centralna aplikacija na zahtjev ECF-a, tijekom evolucije, koristi ovu funkciju kako bi ECF mogao adekvatno evoluirati stablo koje se nalazi u GPLight+ agentu. Kada ECF dovrši evoluciju najbolje jedinice, njezino stablo i dobrota se spremaju kako bi se mogle koristiti u ansamblima.

4.3.2. Evaluacija i analiza naučenog agenta

Centralna aplikacija podržava način rada u kojem se već naučeni agent može opet provesti kroz jednu iteraciju simulacije, kako bi se dobile detaljnije informacije o njegovom ponašanju u prometu. Moguće je dobiti detaljne informacije o svakom raskrižju u vrijeme svakog donošenja odluke. Ovi podaci su spremljeni u tekstualnu datoteku. Također uz tu datoteku generiraju se i dvije datoteke koje se mogu koristiti s *frontend* opcijom originalne CityFlow biblioteke.

4.3.3. Stvaranje ansambla

Centralna aplikacija nudi mogućnost prikupljanja podataka o svim do sad izvedenim učenjima agenata i generiranja početnih skupova za stvaranje ansambala. Ansamble stvaramo na četiri distinktna načina [6]:

- RANDOM

Bira se 5 nasumičnih agenata, koje predaje u ansambl.

- WEIGHTED RANDOM

Bira se 5 nasumičnih agenata, pri čemu agenti s boljom dobrotom imaju veću šansu odabira, koje predaje u ansambl.

- GROW

Bira se jedan nasumični agent i predaje ga u ansambl. Zatim evaluira sve ostale moguće agente u kombinaciji s postojećim sadržajem ansambla, kako bi odredila najboljeg agenta za dodati u ansambl. Ovaj postupak se ponavlja dok broj agenata u ansamblu ne dosegne pet.

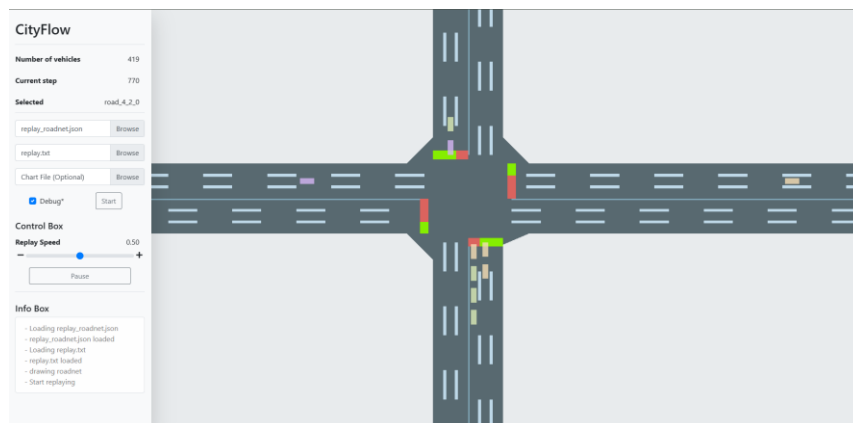
- GROW DESTROY

Ponašanje je isto kao i načinu rada GROW, uz razliku da se stvara ansambl veličine 10, te zatim redom miče agente koji najmanje doprinose ansamblu dok ne dosegne pet agenata.

Svaki ansambl ima dva različita načina donošenja odluke, ne temelju odluka svojih agenata. Agenti u ovom slučaju ansamblu samo predaju informaciju o tome koliko važnost im nosi koja prometna faza. Ansambl s načinom odabira BORDA uzima u obzir važnosti svih faza te sumira njihov položaj na listi, te zatim odabire najbolju. Dok ansambl s načinom rada PLURALITY uzima samo najbolju fazu svakog agenta i odabire onu za koju se odlučio najveći broj agenata. U situaciji jednakog broja glasova, ansambl odabire onaj koji se nalazi prije u listi mogućih faza prema konfiguracijskoj datoteci za CityFlowCpp. U oba načina rada ansambl održava pravilo o crvenom svjetlu između različitih prometnih faza. Svi ansambli generirani na ovakve načine se kasnije evaluiraju zasebno na jednoj simulaciji kako bi mogli zapamtiti njihovu dobrotu i sastav agenata.

4.3.4. Pomoćni sustav

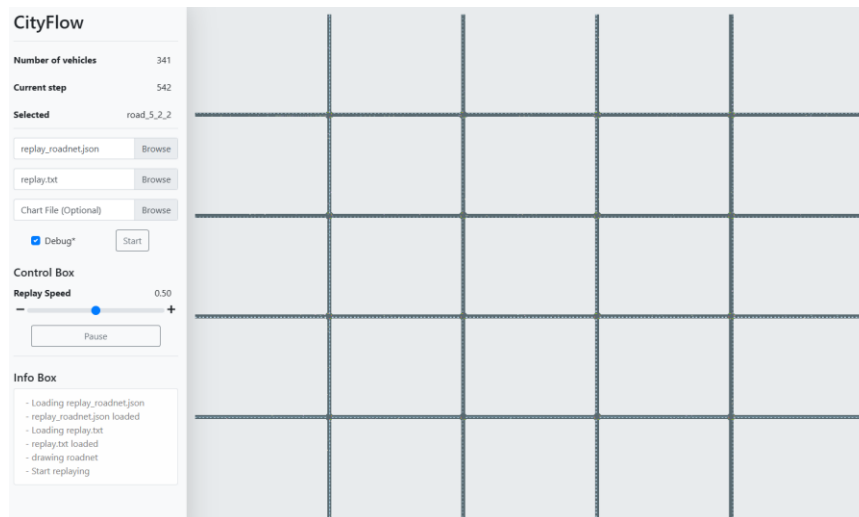
Dio centralne aplikacije koristi se kao pomoćni sustav ostalim dijelovima. Kako bi učenje GPLight+ agenata moglo biti korišteno kod kreiranja ansambala, potrebno je spremati i dokumentirati višestruka pokretanja učenja. Uz spremanje, ovaj modul omogućava paralelno pokretanje više instanci učenja istovremeno, kao i istovremeno pokretanje više instanci evaluacije ansambala. Jedna od mogućnosti dijela programa je priprema za evaluaciju ansambala. Prije pokretanja evaluacije ansambala potrebno je pronaći sve postojeće agente koji će biti korišteni, odabrati početne skupove agenata koji čine pojedini ansambl i pripremiti te agente za evaluaciju. Tu je također i *frontend* dio, koji je preuzet direktno repozitorija originalnog CityFlow projekta [11]. On omogućuje prikaz jedne iteracije simulacije za odabranog agenta, na temelju *replay.txt* i *replay_roadnet.json* koji se generiraju pri svakom pokretanju simulacije. Prikaz se odvija u pregledniku, te je u njemu moguće pregledati točno ponašanje svakog vozila, trake, ceste i semafora u bilo kojem trenutku simulacije.



Sl. 4.3 Primjer grafičkog prikaza simulacije

5. Eksperimenti i rezultat

Učenje i sva mjerenja agenata odvijaju se na mreži raskrižja HuangZhou4x4 sa tokom vozila korištenim u originalnom GPLight+ radu [1]. Simulacija traje 3600 sekundi (1h). Dobrotu svih agenata procjenjujemo na temelju prosječnog vremena putovanja (ATT), gdje bolji agenti imaju manji ATT.



Sl. 5.1 Prikaz mreže raskrižja HuangZhou4x4

5.1. GPLight+

Za potrebe usporedbe s ostalim metodama upravljana semafora i izgradnju ansambala, naučeno je 50 GPLight+ agenata. Agenti su učeni u populacijama od 100 jedinki, pri čemu se učenje zaustavlja nakon 51 generacije. Parametri selekcije i generiranja jedinki, kao i parametri algoritma učenja su opisani u sljedećoj tablici (Tablica 5.1).

Tablica 5.1 parametri učenja GPLight+ agenata

Parametar	Vrijednost
Broj jedinki u populaciji	100
Maksimalan broj generacija	51
Vjerojatnost mutacije	0.2
Broj sačuvanih najboljih jedinki	1
Operator selekcije	<i>SteadyStateTournament</i>
Broj jedinki u turniru	3
Inicijalna minimalna dubina stabla	3
Maksimalna dubina stabla	6
Skup operatora stabla	+, -, *, /
Skup terminalnih simbola	$W_0, W_1, W_2, W_3, C_0, C_1, C_2, C_3, EPM$

5.2. Ansambli

Koristeći naučene GPLight+ agente, generirane su različite varijante ansambala. Testirali smo četiri različite metode biranja agenata i dva načina donošenja odluke. Za svaki od 4 načina generiranja početni agenti su odabrani potpuno nasumično, nezavisno jedno od drugih, dok su za načine donošenja odluke ponovljeni isti početni agenti u oba slučaja. Svaku metodu generiranja i donošenja odluke smo ponovili 50 puta.

5.3. Referentni primjeri

Kako bi dodali perspektivu našim rezultatima, također je implementiran upravljač s cikličkim redoslijedom faza koji koristi predefinirani redoslijed faza s fiksnim trajanjima. Jedan ciklus je definiram ovim redoslijedom (Tablica 1.1):

$$P_1, P_2, P_1, P_6, P_1, P_4, P_1, P_8, P_1, P_3, P_1, P_7, P_1, P_5, P_1, P_9$$

Tijekom ciklusa sve faze imaju trajanje od 30 sekundi, osim P_1 koje traje 5 sekundi. Ovaj agent služi kao usporedba s stvarnim sustavima koji se koriste vrlo često na raskrižjima bez podrške za naprednije agente. Također u rezultate su uključena mjerenja iz originalnog GPLight+ rada [1], kako bi mogli usporediti našu implementaciju istog agenta s originalnom. Usporedba je moguća jer se u oba rada koristi ista jezgra simulatora s istom mrežom raskrižja i tokom vozila.

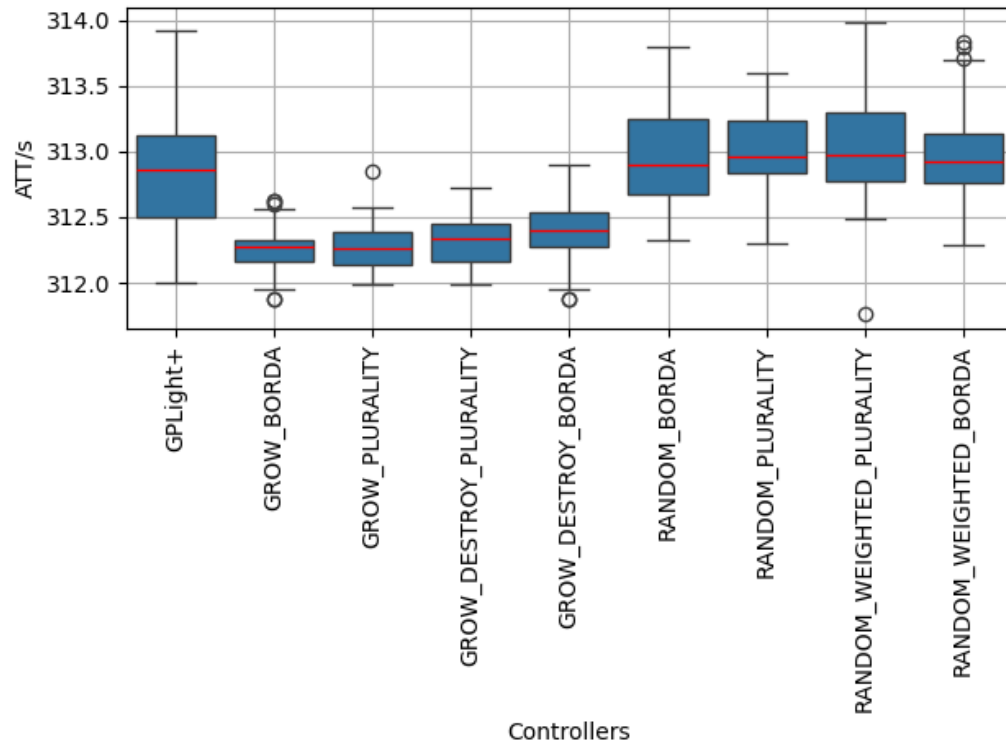
5.4. Rezultati

Rezultati svih provedenih eksperimenata prikazani su u tablici (Tablica 5.2), gdje su za svaku vrstu agenta navedeni najbolji i prosječni ATT te standardna devijacija, uz pripadajuću vrijednost propusnosti (NT). Za ciklički agent prikazana je samo jedna vrijednost ATT-a i NT-a, jer je njegovo ponašanje na korištenom skupu podataka potpuno determinističko, pa pojmovi najboljeg i prosječnog rezultata te standardne devijacije za njega nemaju smisla. Iako je u funkciji dobrote tijekom učenja korišten isključivo ATT, vrijednost NT usko je povezana s njim te je u tablici prikazana kao dodatna mjera razlike u uspješnosti agenata u slučajevima kada je njihov ATT vrlo sličan.

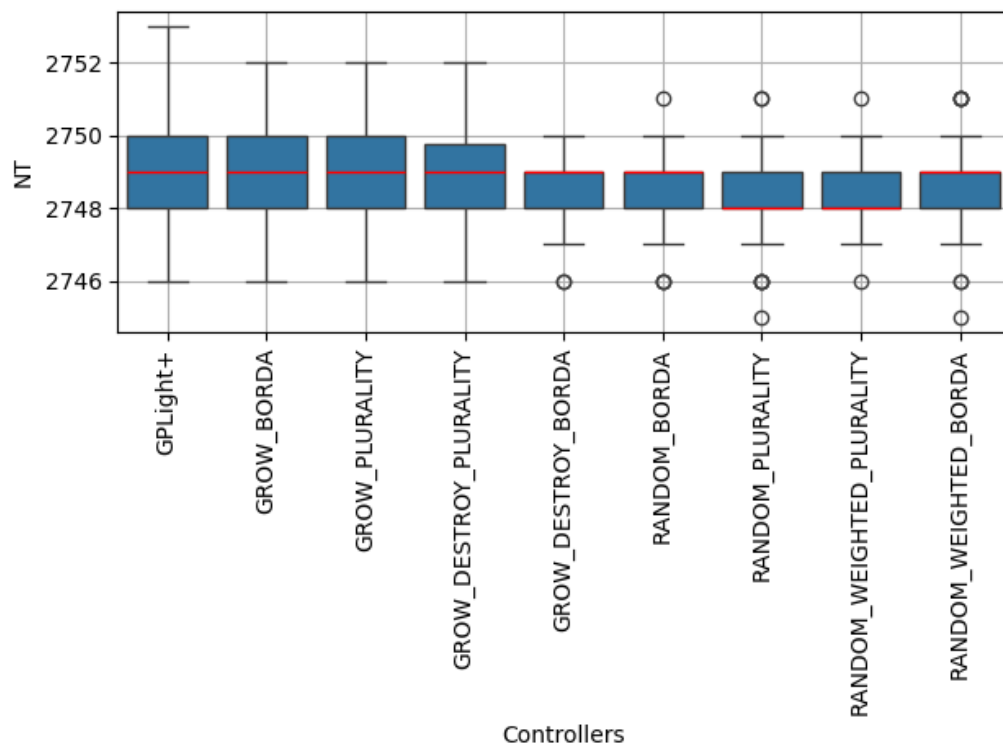
Implementirani GPLight+ agent ostvaruje ATT skoro identičan ATT-u referentnog GPLight+ agenta (311,869 prema 311,888 s), što je vrlo korisno za buduće uspoređivanje naših rezultata s ostalim informacijama iz originalnog rada. Među svim generiranim ansamblima jedan primjer, ansambl RANDOM WEIGHTED s načinom odlučivanja PLURALITY, ostvaruje bolji najbolji ATT (311,766 s) od referentnog i implementiranog GPLight+ agenta. Unatoč tome, ansambli kao pravilo nisu bili bolji od jednostavnog GPLight+ agenta.

Tablica 5.2 Rezultati pojedinačnih eksperimenata

Vrsta upravljača			ATT (sec)	NT	
Upravljač s cikličkim promjenama faza			591.457	2413	
Referentni GPLight+ agent			najbolji	311.888	2935
			prosječni	312.991	2933.6
			std.	1.423	0.8
Implementirani GPLight+ agent			najbolji	311.869	2753
			prosječni	312.305	2748.8
			std.	0.307	1.2
Ansambl	RANDOM	BORDA	najbolji	312.320	2751
			prosječni	312.961	2748.6
			std.	0.367	1.1
		PLURALITY	najbolji	312.296	2751
			prosječni	313.008	2748.3
			std.	0.294	1.3
	RANDOM WEIGHTED	BORDA	najbolji	312.285	2751
			prosječni	312.973	2748.5
			std.	0.372	1.3
		PLURALITY	najbolji	311.766	2751
			prosječni	313.023	2748.3
			std.	0.403	1.0
	GROW	BORDA	najbolji	311.880	2752
			prosječni	312.252	2748.7
			std.	0.171	1.2
		PLURALITY	najbolji	311.989	2752
			prosječni	312.277	2749.0
			std.	0.172	1.4
	GROW DESTROY	BORDA	najbolji	311.880	2750
			prosječni	312.393	2748.6
			std.	0.225	1.1
		PLURALITY	najbolji	311.989	2752
			prosječni	312.321	2748.9
			std.	0.180	1.2



Slika 5.1 Grafički prikaz rezultata različitih agenata prema prosječnom vremenu putovanja



Slika 5.2 Grafički prikaz rezultata različitih agenata prema propusnosti

5.5. Rasprava

Rezultati pokazuju da ansambli, u prosjeku ne ostvaruju značajno bolju uspješnost od pojedinačnog GPLight+ agenta. Jedno moguće objašnjenje jest da su odabrani tok prometa i mreža raskrižja, dosegli svojevrсни prirodni plato uspješnosti, iznad kojeg je daljnje poboljšanje teško ostvarivo bez dodatnih informacija o prometu. Drugo moguće objašnjenje jest da su svi naučeni GPLight+ agenti, zbog sličnog procesa učenja i istog skupa parametara, međusobno vrlo slični, pa njihovo kombiniranje u ansamblu ne donosi dovoljno novih vrsta odluka koje poboljšavaju promet.

Za buduća istraživanja predlaže se proširenje skupa ulaznih parametara stabla. Primjerice, moglo bi se uvesti informacija o tome je li promatrani TM već uključen u trenutnu fazu, s obzirom na to da pravilo o crvenom svjetlu između različitih prometnih faza, izravno utječe na tok vožnje, a trenutna stabla tu informaciju ne koriste. Trenutno je moguća situacija u kojoj dvije različite faze imaju jednaku korisnost, te se odabirom jedne, zbog propuštanja vozila, poveća korisnost druge. Ovakva situacija može završiti u cikličnom korištenju te dvije faze i samim time svaki je put obvezno koristiti pet sekundi crvenog svjetla. Agent s informacijom o trenutno odabranoj fazi, bi mogao prepoznati korist u korištenju iste faze više puta, kako bi se izbjegla potreba za korištenjem crvenog svjetla.

Slično, moglo bi se razmotriti prosječno vrijeme koje je vozilo provelo na traci od trenutka ulaska, ili prosječna brzina vozila, u nadi da bi se time mogla postići neka vrsta „zelenog vala“. Još jedna od mogućnosti je istražiti kombiniranje različitih vrsta agenata unutar istog ansambla, kako bi se osigurali raznolikiji načini odlučivanja unutar njega.

6. Zaključak

U ovom radu razvijen je sustav za stvaranje, učenje i evaluaciju agenata genetskim programiranjem za upravljanje prometnim semaforima, temeljen na metodi GPLight+, te je istražena mogućnost kombiniranja naučenih agenata u ansamble. Sustav je realiziran povezivanjem biblioteka ECF i CityFlowCpp s vlastitom centralnom aplikacijom, što je omogućilo učenje, evaluaciju i analizu velikog broja agenata na mreži raskrižja HuangZhou4x4.

Iako se ansambli u prosjeku nisu pokazali znatno boljima od pojedinačnog GPLight+ agenta, implementirani GPLight+ agenti dosegнули su uspješnost gotovo jednaku onoj iz referentnog rada, što smatramo važnim uspjehom ovog istraživanja. Ovaj rezultat potvrđuje ispravnost razvijenog sustava te ga čini dobrom polazišnom točkom za buduće uspoređivanje s drugim pristupima upravljanja prometnim semaforima.

Literatura

- [1] Liao, X.-C., Mei, Y., Zhang, M. GPLight+: A Genetic Programming Method for Learning Symmetric Traffic Signal Control Policy. IEEE Transactions on Evolutionary Computation, 2025.
- [2] Federal Highway Administration. Balancing Safety and Capacity in an Adaptive Signal Control System - Phase 1 - Chapter 2: Signal timing and safety. U.S. Department of Transportation, Poveznica: <https://www.fhwa.dot.gov/publications/research/safety/10038/002.cfm>; pristupljeno 20.04.2026.
- [3] Federal Highway Administration. Signalized Intersections: Informational Guide – Chapter 4: Traffic design and illumination. U.S. Department of Transportation, Poveznica: <https://www.fhwa.dot.gov/publications/research/safety/04091/04.cfm>; pristupljeno 20.04.2026.
- [4] Poli, R., Langdon, W. B., McPhee, N. F. A Field Guide to Genetic Programming. Lulu Press, 2008.
- [5] Ricalde, E., Banzhaf, W. Evolving Adaptive Traffic Signal Controllers for a Real Scenario Using Genetic Programming with an Epigenetic Mechanism. Proceedings of the 16th IEEE International Conference on Machine Learning and Applications, 2017
- [6] Đurasević, M., Jakobović, D. Creating Dispatching Rules by Simple Ensemble Combination. Journal of Heuristics, vol. 25, 2019.
- [7] Jakobović, D., Đurasević, M., Picek, S., Gašperov, B. *ECF: A C++ Framework for Evolutionary Computation*. SoftwareX, vol. 27, 101640, 2024.
- [8] Evolutionary Computation Framework (ECF). Poveznica: <http://ecf.zemris.fer.hr/>; pristupljeno 29.05.2026.
- [9] Pristav M. CityFlowCpp. GitHub repozitorij. Poveznica: <https://github.com/s-budget/CityFlowCpp>; pristupljeno 18.06.2026
- [10] Zhang, H., Feng, S., Liu, C., Ding, Y., Zhu, Y., Zhou, Z., Zhang, W., Yu, Y., Jin, H., Li, Z. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. Proceedings of The Web Conference, 2019.
- [11] CityFlow Project. *CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario*. Poveznica: [CityFlow Project](#); pristupljeno 20.04.2026.
- [12] Pristav M. Optimizacija upravljanja prometnim signalima evolucijskim algoritmima. GitHub repozitorij. Poveznica: https://github.com/s-budget/ECF_poc; pristupljeno 25.06.2026

Skraćenice

TM	<i>Turn Movement</i>	prometno skretanje
P	<i>Phase</i>	faza semafora
R	<i>Road</i>	cesta
L	<i>Lane</i>	prometna traka
I	<i>Intersection</i>	raskrižje
R_{in}	<i>Incoming Road</i>	ulazna cesta
R_{out}	<i>Outgoing Road</i>	izlazna cesta
L_{in}	<i>Incoming Lane</i>	ulazna traka prometnog skretanja
W	<i>Waiting vehicles</i>	broj zaustavljenih vozila
C	<i>Count (vehicles)</i>	ukupni broj vozila
EPM	<i>Ephemeral terminal</i>	konstanta (težinski faktor lista)
ATT	<i>Average Travel Time</i>	prosječno vrijeme putovanja
NT	<i>Network Throughput</i>	propusnost (broj vozila na odredištu)
SEC	<i>Simple Ensemble Combination</i>	jednostavno kombiniranje ansambla
ECF	<i>Evolutionary Computation Framework</i>	okvir za evolucijsko računarstvo
FER	Faculty of Electrical Engineering and Computing	Fakultet elektrotehnike i računarstva