SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 490

RJEŠAVANJE PROBLEMA USMJERAVANJA VOZILA KORIŠTENJEM NEURONSKIH MREŽA

Tin Jukić

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 490

RJEŠAVANJE PROBLEMA USMJERAVANJA VOZILA KORIŠTENJEM NEURONSKIH MREŽA

Tin Jukić

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

ZAVRŠNI ZADATAK br. 490

Pristupnik:	**Tin Jukić (0036527458)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Marko Đurasević

Zadatak: **Rješavanje problema usmjeravanja vozila korištenjem neuronskih mreža**

Opis zadatka:

Proučiti neuronske mreže te načine na koji se evolucijski algoritmi mogu iskoristiti za treniranje neuronskih mreža. Poručiti problem usmjeravanja vozila te jednostavne heuristike koje se koriste za rješavanje zadanog problema. Primijeniti neuronsku mrežu učenom genetskim algoritmom za određivanje idućeg korisnika prilikom izgrađivanja rješenja za problem usmjeravanja vozila. Usporediti dobivene rezultate s jednostavnim heurističkim postupkom za stvaranje rješenja. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Rok za predaju rada: 10. lipnja 2022.

Sadržaj

Uvod

Problem usmjeravanja vozila (Vehicle Routing Problem, VRP) jedan je od često rješavanih problema prilikom rada s algoritmima strojnog učenja, zbog načina današnjeg načina života (ljudi sve više naručuju stvari putem interneta, koje im je potom potrebno dostaviti). Upravo zbog tog razloga, rješavanjem problema usmjeravanja vozila rješavamo problem dostavljanja naručenih stvari korisnicima. Prilikom njegovog rješavanja, korisniku je na raspolaganju ograničeni broj vozila s pretpostavkom da su sva vozila jednaka i da im je doseg neograničen, odnosno da imaju neograničenu količinu goriva na raspolaganju. Zadatak je naći optimalni put prilikom kojeg se obiđu svi korisnici s najpovoljnijim ishodom (npr. korišteno je najmanje vozila, prijeđen je najkraći put, kapacitet svakog vozila je iskorišten do maksimuma…). Solomon benchmark ima već definirane različite instance za rješavanje ovog problema, koje se razlikuju po pojedinim svojstvima. U ovom radu je za rješavanje problema usmjeravanja vozila korištena neuronska mreža, koja računa koje korisnike je potrebno obilaziti kojim redoslijedom, a za učenje neuronske mreže korišten je genetski algoritam.

1. Opis problema i metode za rješavanje

1.1. VRP

VRP (Vehicle Routing Problem) jedan je od najčešće rješavanih optimizacijskih problema. VRP problem sastoji se od čvorova. Svaka instanca ima čvor izvorište, čvorove korisnike te vozila. Prilikom rješavanja ovog problema, vodimo se pretpostavkom da su svi automobili identični te da svaki automobil na raspolaganju ima neograničenu količinu goriva, odnosno da mu je doseg neograničen. Svi automobili kreću iz čvora izvorišta te dalje obilaze čvorove korisnike, sve dok se ne dosegnu ograničenja. Ograničenja mogu biti razna te upravo zbog toga postoje i brojne varijante ovog problema, npr. VRP s vremenskim prozorom (VRPTW), VRP s ograničenim kapacitetom automobila (CVRP), VRP s više izvorišta te brojne druge. U ovom radu fokus je na CVRP-u, odnosno pretpostavka je da svaki automobil ima ograničeni kapacitet i da su kapaciteti svih automobila jednaki.

Rješenje VRP problema je put, kojim vozila moraju obići korisnike. Kako postoji više putova kojima je moguće obići korisnike, postavlja se pitanje koji od njih je optimalan te kako ga pronaći. Zbog kombinatorne kompleksnosti ovog problema, za njegovo rješavanje se često koriste metode strojnog učenja. Za implementaciju VRP problema često se koriste umjetne neuronske mreže (ANN) i genetski algoritam (GA). Oni omogućuju treniranje modela te pronalazak optimalne rute kojom bi se automobili trebali kretati, kako bi troškovi bili što manji. Troškovi također mogu biti razni, npr. u slučaju VRPTW-a, trošak je vrijeme te algoritam teži utrošiti što je manje moguće vremena za obilazak korisnika, kod CVRP-a trošak je kapacitet te se algoritam trudi što je bolje moguće iskoristiti kapacitet danog automobila te time minimizirati trošak, te brojni drugi. Valja napomenuti da se troškovi mogu međusobno kombinirati. Tako npr. algoritam može težiti maksimalnom iskorištenju kapaciteta automobila i pritom minimizirati prijeđeni put (problem koji se rješava u nastavku rada).

Svaki VRP problem ima definirano koliko automobila ima na raspolaganju te nije nužno iskoristiti sve automobile. Također se može dogoditi da ne postoji put kojim bi algoritam uspio obići sve korisnike.

U ovom radu rješava se problem VRP s pretpostavkom da svaki automobil ima ograničeni kapacitet i da su svi automobili identični. Algoritam na raspolaganju ima 25 automobila,

svaki kapaciteta 200. Potrebno je obići 100 korisnika uz maksimizaciju kapaciteta te prelazak minimalne udaljenosti.

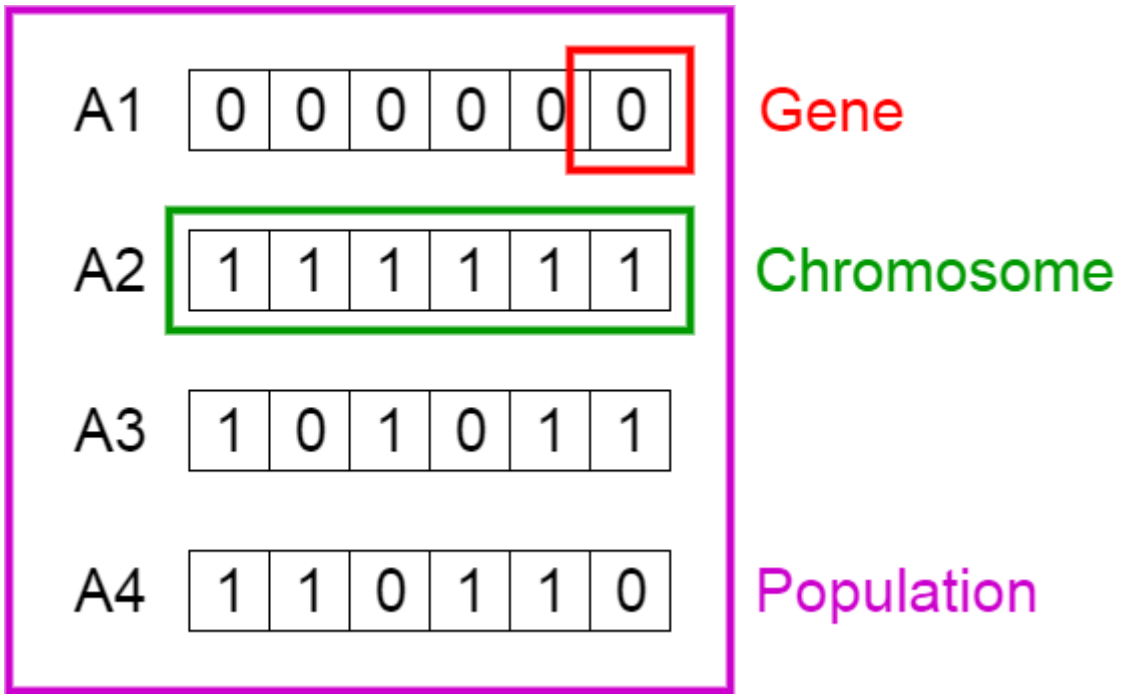
1.2. Metode za rješavanje

Kao što je već ranije navedeno, rješenje VRP problema nije jedinstveno, odnosno može postojati više rješenja. Također, rješenja ne moraju biti jednako kvalitetna te je cilj naći najbolje, odnosno optimalno rješenje konkretnog problema. Zato je za rješavanje ovakvih problema pogodno koristiti algoritme strojnog učenja. Jedan od danas češće korištenih algoritama za rješavanje ovog problema su umjetne neuronske mreže, koje su korištene u ovom radu. U nastavku slijedi kratki opis pojedinih metoda rješavanja problema.

1.2.1. Genetski algoritam (GA)

Genetski algoritam (GA) heuristička je metoda optimiranja. Ideja iza ovog algoritma, kako samo ime kaže, dolazi iz prirode. Ovaj algoritam je inspiriran istraživanjem biologa Charlesa Darwina o prirodnoj evoluciji. Glavni temelj ovog algoritma je pretpostavka o prirodnoj selekciji i preživljavanju samo najboljih jedinki („survival of the fittest“). Jedinke koje su preživjele daju potomke za iduću generaciju.

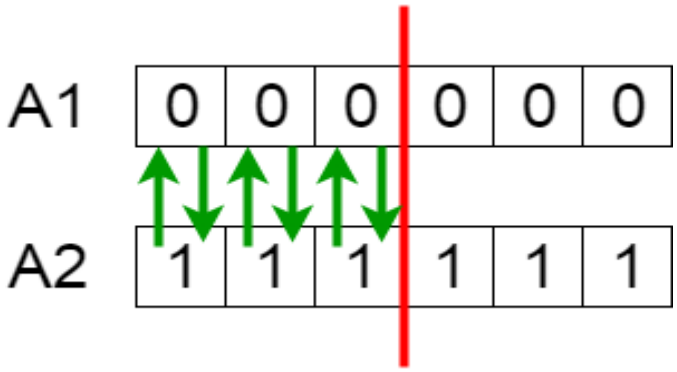
Temeljna jedinica kod rada s ovim algoritmom je jedinka. Svaka jedinka se sastoji od kromosoma. Kromosom je matrica koja se sastoji od svojstava, odnosno gena, vezanih uz nju. Geni su najčešće predstavljeni najčešće decimalnim vrijednostima u rasponu između 0 i 1 i često se još nazivaju težinama.



Sl. 1.1 Prikaz glavnih dijelova genetskog algoritma (preuzeta s GA)

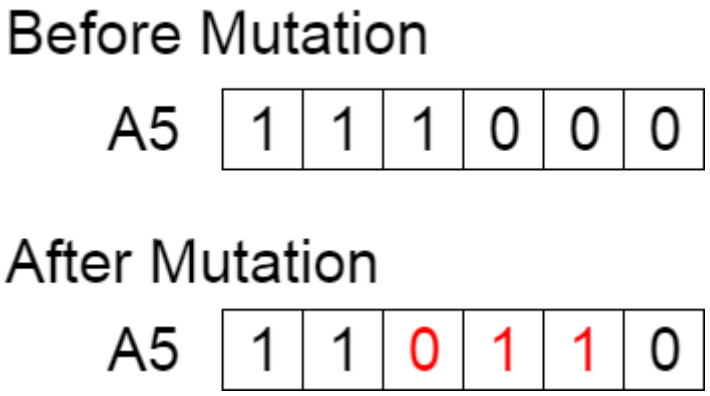
Genetski algoritam sastoji se od 5 faza:

1. Početna populacija, koja se inicijalizira s nasumično odabranim vrijednostima za gene u kromosomima.
2. Funkcija dobrote ocjenjuje koliko je pojedina jedinka dobra, npr. koliko dobro određena jedinka rješava neki problem.
3. Faza selekcije, tijekom koje se uzimaju one najbolje jedinke (one čija je funkcija dobrote najveća) te se one koriste za stvaranje nove populacije jedinki. Prilikom faze selekcije, izabiru se dva roditelja koja imaju dobra svojstva (koji je dobiven pomoći fitness function-a) kako bi napravila novu jedinku, koja sadrži kombinaciju gena obaju roditelja (križanje).
4. Križanje, koji se smatra najbitnijim dijelom genetskog algoritma. Za svakog od dva roditelja koja su izabrana u fazi selekcije za stvaranje potomka, uzima se nasumična cjelobrojna vrijednost unutar gena. Nova jedinka se pravi na način da se geni dvaju roditelja zamijene sve do točke prekida (prethodno izabrana cjelobrojna vrijednost). Iza crossover pointa nema nikakvih promjena u genima. Time nastaju dvije nove jedinke koje se pohranjuju u novu populaciju.



Sl. 1.2 Prikaz izmjene gena između dvaju roditelja do nailaska na crossover point (vertikalna crvena crta) (preuzeta s GA)

5. Zadnji korak prilikom stvaranja jedinki je mutacija. Mutacija omogućuje promjenu pojedinih gena jedinke uz vrlo malu vjerojatnost dolaska do mutacije. Na taj se način održava različitost (diversity) unutar populacije te se sprječava nastanak prerane konvergencije.



Sl. 1.3 Geni jedinke prije (iznad) i nakon mutacije (ispod) – ovdje su radi jednostavnosti prikaza izabrano da geni mogu biti samo 0 ili 1 (binarni) (preuzeta s GA)

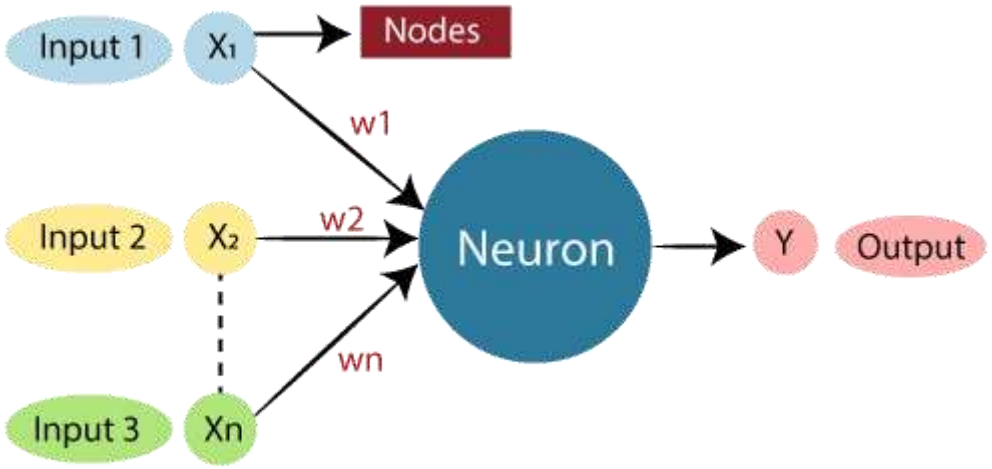
Algoritam završava s radom kada dostigne maksimalni broj generacija koje je korisnik zadao ili kada se dostigne konvergencija, odnosno, kada se počinju stvarati nove jedinke koje se vrlo malo razliku od prethodnih. Tada možemo reći da je genetski algoritam dao rješenje za dani problem.

1.2.2. Umjetna neuronska mreža (ANN)

Umjetna neuronska mreža (ANN) je model strojnog učenja koji također, kao i genetski algoritam od prije, inspiraciju nalazi u živim bićima. Naime, neuronska mreža temelji se na izgledu i funkcioniranju ljudskog mozga. Najmanja građevna jedinica kod neuronske mreže je neuron, koji, baš kao i ljudski neuroni, prima neke informacije (kod živih organizama su to podražaji, ovdje su to brojevni ulazi) i daje određeni izlaz. Ideja je da se rješavanje kompleksnog problema razdvoji na manje pod probleme, kako bi se ubrzalo njegovo rješavanje. Neuroni kod umjetne neuronske mreže su međusobno povezani (kao i kod živih organizama) u različitim slojevima mreže, koji se nazivaju čvorovima.

1.2.2.1 Neuron

Neuron je najmanja građevna jedinica kod umjetnih neuronski mreža. Svaki neuron se sastoji od ulaza u njega (ekvivalentno podražajima kod živih organizama) koji se nazivaju težinama (weights, `w`), sastoji se od funkcije aktivacije, koja konačno određuje izlaz neurona.



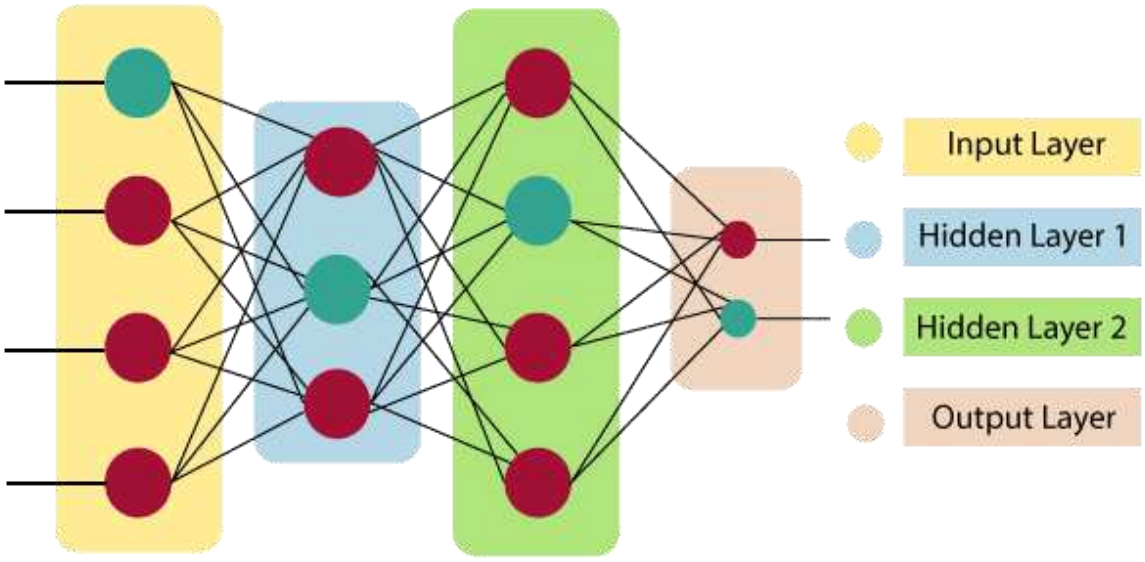
Sl. 1.4 Prikaz neurona zajedno s njegovim ulazima, težinama i izlazom (preuzeta s Artificial neural network)

1.2.2.2 Slojevi (Layers)

Svaka umjetna neuronska mreža sastoji se od slojeva. Broj slojeva koje neuronska mreža može imati nije ograničen i najčešće ovisi o problemu koji se rješava. Bitno je za napomenuti da slojevi neuronske mreže ne moraju imati isti broj čvorova, nego se oni mogu razlikovati.

Hijerarhijski gledano, postoje tri vrste slojeva unutar svake umjetne neuronske mreže.

1. Ulazni sloj (input layer), uz pomoć kojeg se omogućuje primanje podataka u neuronsku mrežu i njihovo računanje. Ulazni sloj najčešće ništa ne računa, nego samo prosljeđuje podatke daljnjim slojevima neuronske mreže, gdje se daljnja obrada događa. Broj čvorova u sloju za unos je također proizvoljan i ovisi o broju podataka koji se žele obraditi.
2. Skriveni sloj (hidden layer) je sloj kojeg, kao što samo ime kaže, „ne vidimo“. U njemu se događa obrada podataka koji su primljeni na njegovim ulazima te se rezultat prosljeđuje na njegove izlaze. Prilikom prosljeđivanja rezultata, najčešće se koristi neka od funkcija aktivacije, npr. relu ili softmax. Uloga funkcija aktivacije je odlučiti hoće li se čvor aktivirati ili neće.
3. Zadnji sloj u hijerarhiji je izlazni sloj (output layer) neuronske mreže. Taj sloj daje konačni rezultat koji se dobije kao rezultat posljednjeg skrivenog sloja



Sl. 1.5 Hijerarhijski prikaz jednostavne neuronske mreže (preuzeta s Artificial neural network)

1.2.2.3 Prednosti i nedostatci

Umjetne neuronske mreže nalaze brojne primjene u mnogim problemima i vrlo su često korišten model. Prednosti korištenja neuronskih mreža su: otpornost na pogreške, mogu raditi s nepotpunim bazama znanja, mogu obrađivati više zadataka istovremeno, itd.

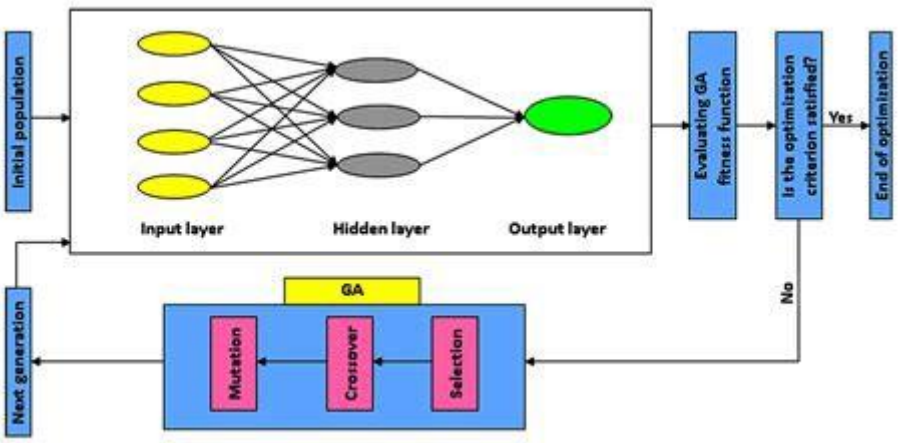
Međutim, kao sve u životu, i one imaju nedostataka, npr.: zahtijevaju određeni hardver (procesore koji omogućuju paralelizam), može biti teško odrediti strukturu neuronske mreže, itd.

1.2.3. Umjetna neuronska mreža potpomognuta genetskim algoritmom

Možda jedno od najvećih ograničenja umjetnih neuronskih mreža su hiperparametri (Hyper-parameters), koji su zapravo vrijednosti težina koje neuronska mreža zahtijeva kako bi mogla riješiti zadani problem. Problem predstavlja dobar izbor hiperparametara. Kako bi se riješilo ovo ograničenje neuronskih mreža, napravljen je novi algoritam učenja neuronski mreža: umjetna neuronska mreža potpomognuta genetskim algoritmom.

Ideja ovog pristupa je napraviti kombinaciju neuronske mreže i genetskog algoritma. Neuronska mreža se koristi za rješavanje zadanog problema, dok se genetski algoritam koristi kako bi se mogle računati nove težine za pojedine slojeve neuronske mreže. Na taj se način omogućuje izmjena težina i traženje rješenja koja bolje opisuju zadani problem.

Prilikom ovog pristupa rješavanju, potrebno je stvoriti nekoliko modela neuronskih mreža koje imaju različite hiperparametre (težine). One se zatim pokreću (istovremeno ili jedna po jedna) te se, nakon što završe s radom, neuronske mreže ocjenjuju, uz pomoć funkcije dobrote. Zatim se uzima nekoliko najboljih neuronskih mreža, kako bi se izračunale nove težine za neuronske mreže koje nastaju iz dvije prethodno izabrane neuronske mreže.



Sl. 1.6 Prikazan odnos između umjetne neuronske mreže i genetskog algoritma (preuzeta s ANN + GA)

1.2.4. Korištena metoda

Algoritam koji je korišten za rješavanje VRP problema s ograničenim kapacitetom vozila je umjetna neuronska mreža potpomognuta genetskim algoritmom, upravo zbog mogućnosti mijenjanja težina za pojedine čvorove neuronske mreže te mogućnosti lakšeg učenja i implementacije rješenja za problem.

Na taj način je omogućen lakši pronalazak optimalne (ili što bolje) rute za zadani problem.

2. Opis implementacije

Kao što je u prethodnom dijelu rečeno, za rješenje ovog konkretnog VRP problema, korištena je umjetna neuronska mreža potpomognuta genetskim algoritmom.

No, prije nego što se opiše algoritam koji je korišten, potrebno je reći da fitness funkcija radi uz pomoć UCS implementacije rješenja problema.

2.1. UCS (Uniform Cost Search)

UCS (Uniform Cost Search) je algoritam koji na jednostavan način odlučuje koji čvor obići prvi. Tu odluku UCS donosi na osnovu cijene puta do određenog čvora, prilikom čega teži minimizirati troškove obilaska čvorova, odnosno, traži put s najmanjom cijenom. U slučaju ovog VRP problema, UCS čvorove ocjenjuje na način da pokušava odabrati čvor koji je na minimalnoj udaljenosti od prethodnog čvora, ali pritom pazi da je zahtijevani kapacitet tog čvora maksimalan, odnosno, najveći moguć za trenutno odabrani automobil. UCS u ovom slučaju kao idući čvor za obilazak uzima upravo onaj čvor koji je najvećeg mogućeg kapaciteta za trenutno vozilo. Uz to, algoritam uspoređuje i udaljenosti. Ako se pronađe čvor s istim zahtijevanim kapacitetom kao i prethodno odabrani, ali je udaljenost do tog čvora manja nego udaljenost do prethodno odabranog čvora, algoritam izabire taj novi čvor istog kapaciteta, ali manje udaljenosti.

Algoritam 1.1 računa najveći mogući kapacitet uz minimalni put.

```
odaberi čvor i
provjeri možeš li ga obići i je li već obiđen
ne:
      odaberi drugi čvor
da:
      izračunaj udaljenost i zahtjev za njega
      odaberi čvor j
      provjeri možeš li ga obići i je li obiđen
      ne:
            odaberi drugi čvor
      da:
            izračunaj udaljenost i zahtjev za njega
```

```
                ako je za čvor j udaljenost manja, a zahtjev
veći:
                        postavi na mjesto čvora i čvor j
                        ima još čvorova za obići
                        da:
                              vrati se na početak
                        ne:
                              vrati čvor i kao rezultat
```

Algoritam 1.1 Funkcija koja računa minimalnu udaljenost i maksimalni kapacitet za čvor

Prethodno objašnjeni algoritam će igrati bitnu ulogu u daljnjoj implementaciji algoritma.

2.2. Implementacija umjetne neuronske mreže potpomognute genetskim algoritmom

Za implementaciju algoritma korišten je programski jezik Python, zato što ima brojne biblioteke koje imaju već gotove implementacije pojedinih algoritama. Korištene su biblioteke tensorflow i keras, koje sadrži gotove implementacije genetskog algoritma i neuronske mreže. Za rješavanje odabrana je jedna konkretna Solomonova instanca VRP problema.

Kao prvi korak u implementaciji, potrebno je definirati oblik neuronske mreže, odnosno, potrebno je odrediti koliko će imati ulaznih podataka, koliko će imati slojeva između te koliko će oni imati izlaza, potrebno je odrediti koliko izlaza ima izlazni čvor i na kraju je potrebno odrediti koje će se aktivacijske funkcije koristiti na pojedinim slojevima u neuronskoj mreži.

Primjer jedne od mogućnosti koja je korištena za rješavanje ovog konkretnog problema dana je u kodu 1.2.

```
inputLayer = keras.layers.Input(shape=(1,))
hiddenLayer1 = keras.layers.Dense(4,
activation="relu")(inputLayer)
hiddenLayer2 = keras.layers.Dense(8,
activation="relu")(hiddenLayer1)
```

```
outputLayer = keras.layers.Dense(1,
activation="linear")(hiddenLayer2)
numOfSolutions = 3
```

Kod 1.2 Izrada modela neuronske mreže

Idući korak u implementaciji je izrada modela neuronske mreže uz pomoć pygad.keras biblioteke koja sadrži metodu za pravljenje modela. Ona prima dva parametra: ulazni i izlazni sloj neuronske mreže.

Nadalje je potrebno instancirati genetski algoritma koji će se koristiti za pomoć prilikom treniranja neuronske mreže. Njega instanciramo pomoću pygad.keras biblioteke, koja sadrži metodu za to.

```
model = keras.Model(inputs=inputLayer, outputs=outputLayer)
    keras_ga = pygad.kerasga.KerasGA(model=model,
num_solutions=numOfSolutions)
```

Kod 1.3 Izrada modela genetskog algoritma i postavljanje prethodno kreiranog modela neuronske mreže genetskom algoritmu

Sada je potrebno definirati druge parametre koji su potrebni za rad genetskog algoritma. Potrebno je definirati kada genetski algoritam treba prestati s radom, odnosno, potrebno je definirati koliko će se generacija stvoriti prije prestanka rada genetskog algoritma. Zatim je potrebno definirati koliko će se roditelja koristiti za stvaranje novih jedinki. Na kraju je potrebno definirati koja je početna populacija genetskog algoritma. Početnu populaciju čine prethodno kreirane neuronske mreže.

```
numOfGenerations = 5
numParentsMating = 2
initialPopulation = keras_ga.population_weights
```

Kod 1.4 Definiranje parametara za genetski algoritam

Zatim je potrebno postaviti te parametre genetskom algoritmu koji će se koristiti prilikom treniranja zadanog modela neuronske mreže. Svaka jedinka genetskog algoritma ima onoliko kromosoma koliko svaka neuronska mreža ima ukupno težina. Svaki čvor neuronske mreže sadrži određeni broj težina, koji je definiram brojem ulaza u taj čvor. Ukupan broj težina

koje svaka neuronska mreža sadrži dobije se tako da se zbroji broj težina koje svaki čvor sadrži.

Na kraju je samo potrebno pokrenuti instancu genetskog algoritma i program započinje s radom.

```
gaInstance = pygad.GA(num_generations=numOfGenerations,
num_parents_mating=numParentsMating,

initial_population=initialPopulation,
fitness_func=fitnessFunc,
                          on_generation=callbackFunc)
    gaInstance.run()
```

Kod 1.5 Postavljanje parametara genetskog algoritma i njegovo pokretanje

Još je bitno opisati korištenu funkciju dobrote. Pomoću nje se ocjenjuje rješenje dane neuronske mreže. Unutar fitness funkcije nalazi se algoritam UCS uz izmjenu da prilikom odabira rute za neki konkretni automobil, neuronska mreža ocjenjuje izabrano rješenje i odlučuje hoće li izabrati neki drugi put kao optimalni. Upravo će se po tom dijelu razlikovati rješenja za pojedine instance neuronske mreže. Kada se odabere rješenje i neuronska mreža ga ocijeni, taj se rezultat dodaje ukupnom fitnessu za tu određenu instancu genetskog algoritma. Na kraju se vraća kompletna ocjena dane instance neuronske mreže te na osnovu te vraćene ocjene genetski algoritam dalje zna izabrati bolja rješenja od gorih, što mu omogućuje pravljenje novih instanci neuronskih mreža i pronalazak boljih rješenja za dani problem.

U nastavku će biti prikazana rješenja problema dobivena upotrebom gore navedenih postavki.

Valja napomenuti da postavke koje su izabrane za treniranje ovog modela nisu optimalne, no zbog prirode problema i hardvera na kojem se algoritam izvršavao, bilo je potrebno dosta minimizirati postavke, kako bi se algoritam izveo u „razumnom“ vremenu.

Za izvršavanje algoritma s gore navedenim postavkama na korištenom računalu je bilo potrebno oko 3 sata.

3. Rješenja problema

Za početak je bilo potrebno riješiti problem uporabom algoritma UCS, koji obilazi sve čvorove i traži onaj s najmanjom udaljenošću i maksimalnim zahtjevom za čvor. Nakon toga je pokrenuta neuronska mreža potpomognuta genetskim algoritmom, opisana u poglavlju iznad.

Rješenja će biti predstavljena pojedinačno za pojedini algoritam te će se na kraju usporediti i bit će prikazana u obliku tablice, radi bolje vizualizacije. Za usporednu algoritama koristit će se nekoliko vrijednosti: ukupna prijeđena udaljenost, ukupan broj iskorištenih automobila, ukupni iskorišteni kapacitet, ukupni preostali kapacitet za pojedine automobile te maksimalni korišteni kapacitet. Rezultati će biti zaokruženi na dvije decimale te će biti prikazani u obliku tablica.

Jednadžbe 3.1, 3.2 i 3.3 računaju pojedine vrijednosti.

$$ukupniPut = \sum putZaPojediniAutomobil$$

Jednadžba 3.1 Računanje ukupnog prijeđenog puta

$$preostaliKapacitet = \sum preostaliKapacitetAutomobila$$

Jednadžba 3.2 Računanje ukupnog preostalog kapaciteta za korištene automobile

$$ukupniIskorišteniKapacitet = \sum iskorišteniKapacitet$$

Jednadžba 3.3 Računanje ukupnog iskorištenog kapaciteta za korištene automobile

Bitno je još navesti karakteristike instance problema koji se rješavao. One su navedene u tablici 3.1.

Broj automobila	Kapacitet pojedinog automobila	Broj korisnika (čvorova)	Broj odredišta (depot-a)
25	200	100	1

Tablica 3.1 Karakteristike instance VRP problema koji se rješava

3.1. Rješenje korištenjem algoritma UCS

Rezultati rješenja VRP problema uz korištenje algoritma UCS prikazana su u tablici 3.2 te se primjer ispisa može vidjeti na slici 3.1.

Broj korištenih automobila	Ukupna prijeđena udaljenost	Ukupni preostali kapacitet	Ukupni iskorišteni kapacitet	Maksimalni iskorišteni kapacitet
8	1202,09	257	1343	198

Tablica 3.2 Rješenja problema VRP korištenjem algoritma UCS

```
Vehicle number: 1:
Traveled distance: 188.32267325334718
Remaining capacity: 10.0
Used capacity: 190
Visited customers: 14
Route: 27 -> 28 -> 80 -> 12 -> 26 -> 58 -> 40 -> 53 -> 90 -> 32 -> 63 -> 64 -> 49 -> 100

Vehicle number: 2:
Traveled distance: 134.8851546648472
Remaining capacity: 19.0
Used capacity: 181
Visited customers: 12
Route: 81 -> 9 -> 51 -> 71 -> 35 -> 65 -> 66 -> 79 -> 33 -> 77 -> 3 -> 68

Vehicle number: 3:
Traveled distance: 192.2060519316261
Remaining capacity: 22.0
Used capacity: 178
Visited customers: 14
Route: 78 -> 29 -> 24 -> 55 -> 25 -> 54 -> 75 -> 56 -> 4 -> 39 -> 23 -> 67 -> 99 -> 59
```

Slika 3.1 Prikaz rješenja VRP problema uz UCS algoritam za prva tri automobila

3.2. Rješenje korištenjem umjetne neuronske mreže potpomognute genetskim algoritmom

Rezultati rješenja VRP problema uz korištenje umjetne neuronske mreže potpomognute genetskim algoritmom prikazana su u tablici 3.3 te se primjer ispisa može vidjeti na slici 3.2.

Broj korištenih automobila	Ukupna prijeđena udaljenost	Ukupni preostali kapacitet	Ukupni iskorišteni kapacitet	Maksimalni iskorišteni kapacitet
7	1444,71	89	1311	200

Tablica 3.3 Rješenje VRP problema korištenjem umjetne neuronske mreže potpomognute genetskim algoritmom

```
Vehicle number: 1:
Traveled distance: 150.35391916792148
Remaining capacity: 11.0
Used capacity: 189
Visited customers: 14
Route: 1 -> 50 -> 51 -> 9 -> 20 -> 65 -> 71 -> 35 -> 66 -> 81 -> 33 -> 3 -> 76 -> 28

Vehicle number: 2:
Traveled distance: 115.83295264319516
Remaining capacity: 27.0
Used capacity: 173
Visited customers: 12
Route: 2 -> 21 -> 40 -> 26 -> 53 -> 58 -> 13 -> 75 -> 56 -> 4 -> 39 -> 23

Vehicle number: 3:
Traveled distance: 228.30854494941326
Remaining capacity: 0.0
Used capacity: 200
Visited customers: 19
Route: 5 -> 84 -> 17 -> 18 -> 52 -> 7 -> 19 -> 11 -> 47 -> 36 -> 46 -> 6 -> 8 -> 45 -> 83 -> 60 -> 10 -> 62 -> 88
```

Slika 3.2 Prikaz rješenja VRP problema uz neuronsku mrežu potpomognutu genetskim algoritmom za prva tri automobila

3.3. Usporedba dobivenih rješenja

Kao što se može vidjeti, treniranje umjetne neuronske mreže uz pomoć genetskog algoritma za rješavanje zadanog problema VRP dalo je zanimljive rezultate. Radi lakše vizualizacije i usporedbe, svi rezultati zajedno će biti ponovno prikazani u tablici 3.4.

	UCS	ANN + GA
Broj korištenih automobila	8	7
Ukupna prijeđena udaljenost	1202,09	1444,71
Ukupni preostali kapacitet	257	89
Ukupni iskorišteni kapacitet	1343	1311
Maksimalni iskorišteni kapacitet	198	200

Tablica 3.4 Usporedba rezultat za pojedini algoritam

Iz tablice iznad je vidljivo da je treniranje dalo neka poboljšanja, ali i neke kompromise. Ukratko će se prokomentirati dobiveni rezultati.

Vidljivo je da je algoritam UCS koristio 8 automobila, dok je ANN (neuronska mreža) koristila 7, što je poboljšanje, ali je ukupna prijeđena udaljenost zato veća kod neuronske mreže u odnosu na UCS algoritam. Također je vidljivo da je ukupni preostali kapacitet za UCS znatno veći od preostalog kapaciteta za neuronsku mrežu te se može primijetiti da je ukupni iskorišteni kapacitet kod UCS-a također veći nego kod ANN-a. I, na kraju, valja primijetiti da je neuronska mreža uspjela dovesti do maksimalnog iskorištenja kapaciteta automobila, dok algoritam UCS to nije mogao postići.

Bilo bi dobro podesiti korišteni genetski algoritam (opisan u poglavlju iznad) drukčije, npr., umjesto 5 generacija, staviti više (možda 200) te povećati `numOfParentsMating` s 2 na 5. Također bi bilo poželjno povećati broj neuronskih mreža koje se koriste s 3 na recimo 10. Time bi se povećala količina neuronskih mreža koje se treniraju, čime bi se omogućio pronalazak bolje rješenja (ako postoji).

Zaključak

Problem usmjeravanja vozila (VRP) je jedan od najčešće rješavanih problema u domeni strojnog učenja. Posebno je zanimljiv zbog brojnih instanci koje se mogu rješavati. U ovom radu su korištena dva algoritma za njegovo rješavanje: UCS i umjetna neuronska mreža potpomognuta genetskim algoritmom. Rezultati koji su dobiveni rješavanjem ovog problema uz korištenje prethodno navedena dva algoritma su vrlo zanimljivi, jer su pomalo začuđujući.

Bio je očekivan napredak u svim dijelovima koji su korišteni za usporedbu, no NN algoritam je dao u nekim dijelovima poboljšanja, a u nekima baš i ne.

Pretpostavka je da je do toga došlo zbog konfiguracije neuronske mreže koja se koristila. Zbog prirode problema i načina rješavanja, treniranje boljih konfiguracija bi zahtijevalo jako puno vremena. Na primjer, za konfiguraciju od 10 neuronskih mreža, uz 250 generacija genetskog algoritma i 5 roditelja koji se uzimaju za stvaranje potomaka, na korištenom računalu bi bilo potrebno oko mjesec dana za izvršavanje algoritma.

U budućnosti bi trebalo osigurati hardver koji je namijenjen za ovakve algoritme, kako bi se treniranje ubrzalo te kako bi se na kraju mogli dobiti bolji rezultati, a ako rezultati i ne budu bolji, bilo bi moguće isprobati više različitih postavki i korištenje drukčijih neuronskih mreža (npr. konvolucijskih).

Literatura

[1] *Vehicle Routing Problems, Methods, and Applications, Second Edition* by Paolo Toth, Daniele Vigo (2014)

[2] *The vehicle routing problem with time windows, flexible service locations and time-dependent location capacity* by Alexander Jungwirth, Markus Frey, Rainer Kolisch, (2020)

[3] https://www.javatpoint.com/artificial-neural-network

[4] https://towardsdatascience.com/artificial-neural-networks-optimization-using-genetic-algorithm-with-python-1fe8ed17733e

[5] https://towardsdatascience.com/gas-and-nns-6a41f1e8146d

[6] https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

[7] https://blog.paperspace.com/train-keras-models-using-genetic-algorithm-with-pygad/

[8] https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/

[9] https://keras.io/guides/functional_api/

Sažetak

U ovom radu se rješava problem usmjeravanja vozila korištenjem umjetnih neuronskih mreža. Opisuje se VRP problem (problem usmjeravanja vozila) te se pokušava riješiti jedna instanca iz Solomonove baze podataka. Korišteno je 25 automobila, svaki kapaciteta 200 te je bilo potrebno obići 100 korisnika. Zatim se opisuje jednostavni algoritam UCS, koji je naveden radi usporedbe s neuronskom mrežom, ali i radi objašnjavanja implementacije neuronske mreže za rješavanje ovog problema. Nakon toga slijedi opis genetskog algoritma, koji se koristi radi treniranja neuronske mreže. Zatim slijedi opis umjetne neuronske mreže te na kraju opis kako se genetski algoritam koristi za treniranje umjetne neuronske mreže. Slijedi opis konfiguracije korištene neuronske mreže. Na kraju se nalaze rezultati oba algoritma te usporedba dobivenih rezultata za ta dva algoritma. Rezultati su raznoliki te su doveli do zaključka da je u budućnosti potrebno koristiti bolje postavke za treniranje neuronske mreže.

Summary

Solving vehicle routing problem using artificial neural networks. Firstly, VRP problem is described and one instance of VRP from Solomon database is trying to be solved. There were 25 vehicles in the fleet, each with the same capacity of 200, and 100 customers were needed to be visited. Next, simple UCS algorithm was described, which was used for comparing VRP results with neural network results and to explain the implementation of the used neural network. Then, genetic algorithm was described, which is used for the trainign of neural network. After that, artificial neural network was described and then the genetic algorithm and previously described neural network were combined together to create an algorithm which was used for solving this instance of VRP problem. Next, configuration of the used neural network was described and after that given results from both algoritms were shown. At the end, the results were compared and the conclusion was made, which states that, in order to make neural network with genetic algorithm better, in the future, better settings for training the neural network are needed to be used.

Skraćenice

VRP – Vehicle Routing Problem (problem usmjeravanja vozila)

ANN – umjetna neuronska mreža

GA – genetski algoritam

ANN + GA – umjetna neuronska mreža potpomognuta genetskim algoritmom

UCS – Uniform Cost Search