Primjena evolucijskih algoritama u latentnom prostoru modela dubokog učenja

Katić, Maria

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 497

PRIMJENA EVOLUCIJSKIH ALGORITAMA U LATENTNOM PROSTORU MODELA DUBOKOG UČENJA

Maria Katić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 497

PRIMJENA EVOLUCIJSKIH ALGORITAMA U LATENTNOM PROSTORU MODELA DUBOKOG UČENJA

Maria Katić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 497

Pristupnica:	**Maria Katić (0036513770)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Domagoj Jakobović

Zadatak: **Primjena evolucijskih algoritama u latentnom prostoru modela dubokog učenja**

Opis zadatka:

Opisati tematiku dubokih generativnih modela, najznačajnije predstavnike te njihova područja primjene. Programski ostvariti duboki generativni model i ispitati njegovu učinkovitost na dostupnim skupovima podataka. Programski ostvariti operatore evolucijskog algoritma i primijeniti evolucijski algoritam na latentni prostor generativnog modela. Posebnu pažnju posvetiti mogućnosti višekriterijske optimizacije u svrhu poboljšanja učinkovitosti dubokog modela i povećanja raznolikosti podataka u latentnom prostoru. Usporediti različite oblike evolucijskih algoritama te ostvarene rezultate generativnog modela s postojećim rezultatima na odabranim skupovima podataka. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Rok za predaju rada: 28. lipnja 2024.

Ovim putem htjela bih se zahvaliti svom mentoru na njegovoj suradnji i znanju.

Veliko hvala mojoj obitelji, Andrei i Dinu na neizmjernoj podršci i ljubavi, bez vas ne bih uspjela.

Sadržaj

Uvod

S porastom računalne snage tijekom posljednjih desetljeća, raspon problema koji je moguće riješiti računalom značajno se povećao. Uz napredak tehnologije, povećava se i složenost problema koji se mogu postaviti i rješavati. Današnji optimizacijski algoritmi sposobni su efikasno pretraživati veliki prostor rješenja, te pronaći kvalitetna rješenja za probleme koji bi tehnikom grubom sile (eng. brute force) ostali neriješeni. Unatoč sposobnosti da rješavaju velike i kompleksne probleme, optimizacijski algoritmi imaju svoja ograničenja. Oni često ne garantiraju konvergenciju niti pronalazak globalnog optimuma, te se istražuju alternativni pristupi optimizaciji.

Jedan od takvih pristupa jest optimizacija u latentnom prostoru, koja nudi novi pristup u rješavanju optimizacijskih problema. Ovaj pristup koristi latentne reprezentacije problema kako bi se omogućilo efikasnije pretraživanje prostora rješenja. U sklopu ovog rada razvijen je i testiran algoritam za rješavanje problema usmjeravanja (eng. routing problem), koji je posebno aktualan u suvremeno doba zbog njegove primjene u logistici, mrežnom inženjeringu i sličnim područjima.

Razvijeni algoritam usmjeren je na optimizaciju problema trgovačkog putnika (TSP), vrste problema navođenja koji predstavlja značajan optimizacijski izazov. Opisan je varijacijski autoenkoder (VAE) koji uči generalizirani latentni prostor za TSP probleme, što omogućuje primjenu genetskog algoritma unutar tog prostora. Dodatno, istražena je i kombinacija genetskog algoritma s Hooke-Jeeves postupkom, čime se dodatno poboljšava efikasnost pronalaska rješenja.

Tijekom istraživanja provedeni su brojni eksperimenti za optimizaciju hiperparametara, kao i usporedbe različitih optimizacijskih algoritama, kako bi se utvrdio najbolji pristup za rješavanje problema. U zaključku rada navedene su prednosti i ograničenja različitih pristupa te su navedeni smjerovi daljnjeg razvoja algoritma.

1. Varijacijski autoenkoderi (VAE) i latentni prostor

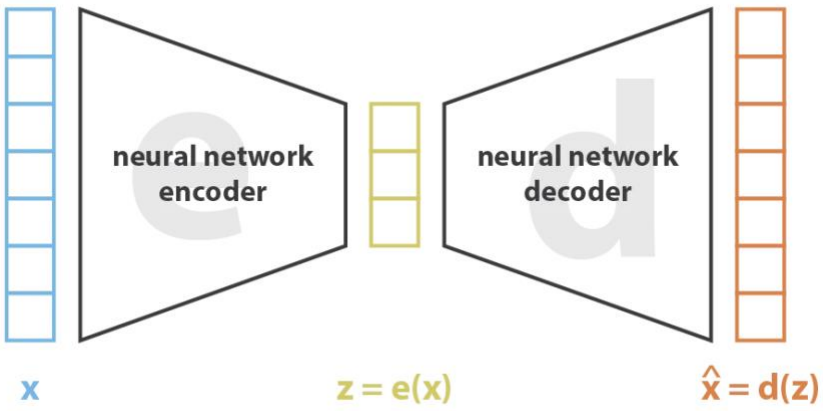
Postoje tri pristupa strojnom učenju – nadzirano učenje (eng. supervised learning), nenadzirano učenje (eng. unsupervised learning) i podržano učenje (eng. reinforcment learning). Svaki od tih pristupa strojnom učenju koristi različite tehnike za obradu podataka.

Nadzirano učenje koristi označene podatke (parove podataka) kako bi model naučio predviđati oznake za nove, neviđene podatke. Podržano učenje, s druge strane, fokusira se na učenje strategija donošenja odluka kroz interakciju s okolinom, gdje model uči na temelju nagrada koje dobiva za svoje akcije. Za razliku od ovih pristupa, nenadzirano učenje bavi se analizom podataka koji nisu označeni, otkriva skrivene strukture i obrasce bez prethodno definiranih oznaka. Tipične primjene nenadziranog učenja uključuju grupiranje sličnih primjera unutar podataka (eng. clustering) i redukciju dimenzionalnosti.

Unutar domene nenadziranog učenja, autoenkoderi predstavljaju ključnu tehniku za učenje efikasnih reprezentacija podataka. Ovi modeli sastoje se od dvije komponente:

- Enkodera - ulazne podatke transformira u vektor u latentnom prostoru
- Dekodera - iz latentnog vektora pokušava rekonstruirati originalne podatke što je preciznije moguće

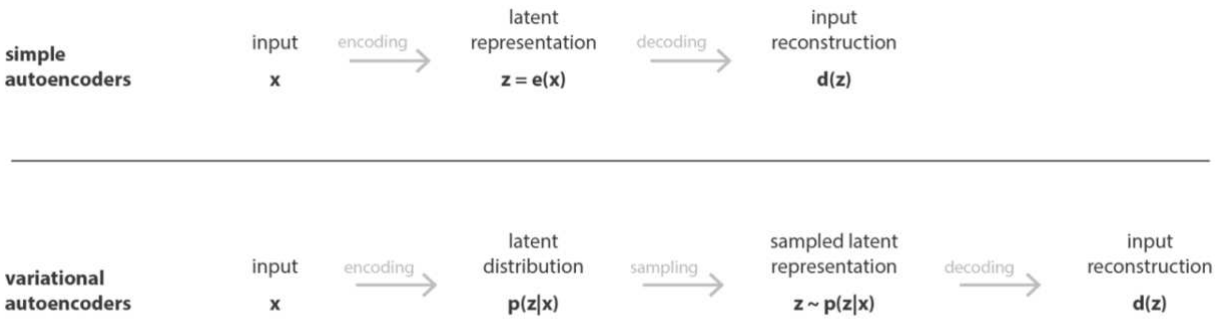
Primarna svrha autoenkodera nije savršeno kopiranje ulaza na izlaz, već da nauči korisnu reprezentaciju (latentni prostor) koja može olakšati daljnju analizu i obradu. Na slici (Slika 1.1) prikazan je izgled autoenkodera.



Slika 1.1 Arhitektura standardnog autoenkodera [2]

1.1. Varijacijski autoenkoder (VAE)

Varijacijski autoenkoderi (VAE) predstavljaju napredak s obzirom na standardne autoenkodere tako što modeliraju latentni prostor kao distribuciju vjerojatnosti, omogućujući generiranje novih podataka koji su slični onima na kojima je model treniran. Enkoder varijacijskog autoenkodera preslikava ulazni podatak kao distribuciju u latentnom prostoru, a ne kao točno određeni vektor. Prilikom dekodiranja izlaznog podataka uzima se uzorak iz distribucije u latentnom prostoru te se taj uzorak dekodira. Zbog toga podatak koji je dekodiran na izlazu nije isti kao ulazni podatak. Razlika u procesu kodiranja i dekodiranja VAE i standardnog autoenkodera prikazana je na slici (Slika 1.2).



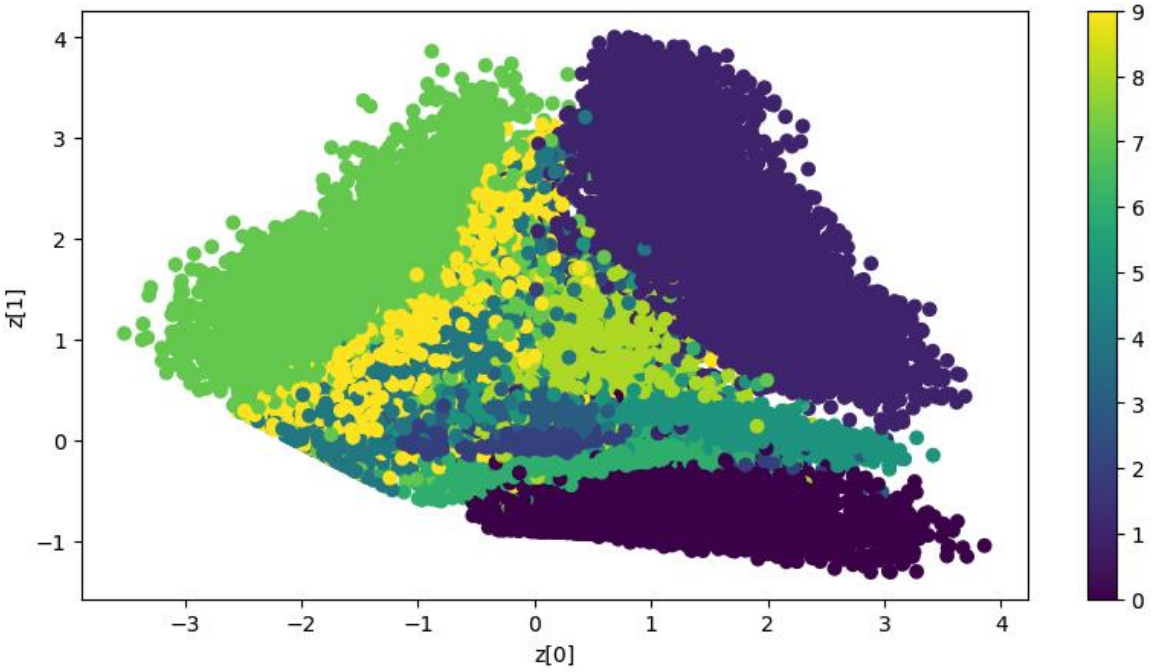
Slika 1.2 Razlika između VAE i standardnog autoenkodera [2]

Glavna prednost VAE u odnosu na standardni autoenkoder je što nema samo mogućnost učenja latentne reprezentacije podataka, već može i generirati nove neviđene podatke. Ova karakteristika čini VAE primjenjivim u optimizacijskim problemima, gdje se takvi modeli mogu koristiti za pronalaženje novih rješenja na temelju naučenih reprezentacija problema, kao što je demonstrirano u razvoju algoritma za rješavanje TSP problema u ovom radu.

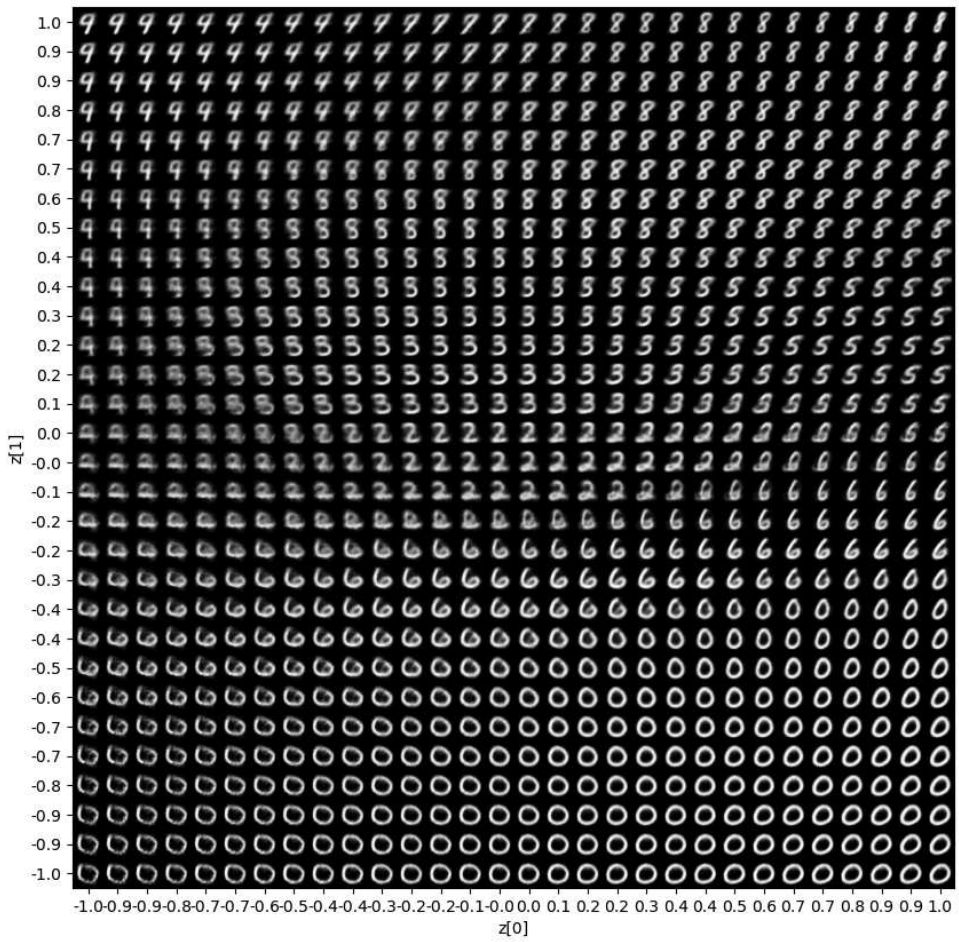
1.1.1. Latentni prostor varijacijskog autoenkodera

Latentni prostor varijacijskog autoenkodera je naučena reprezentacija ulaznih podatka. Modeliran je kao distribucija, i to najčešće normalna distribucija koja je određena parametrima srednje vrijednosti i varijance. Opisani način modeliranja latentnog prostora

omogućuje generiranje novih primjera koji nisu potpuno slučajni, nego imaju određena svojstva početnog primjera. Na slici (Slika 1.3) prikazan je latentni prostor varijacijskog autoenkodera koji je naučen nad MNIST rukom pisanim znamenkama. Skala na desnoj strani slike prikazuje kojom bojom je kodirana svaka od MNIST znamenki. Iz slike se jasno vidi da je VAE uspio pronaći 2D latentnu reprezentaciju u kojoj su izgledom slične znamenke bliže jedna drugoj. Na slici (Slika 1.4) prikazana je skup slika (znamenki) koje se dobije na izlazu dekodera prilikom dekodiranja različitih latentnih vektora.



Slika 1.3 Latentna reprezentacija MNIST znamenki



Slika 1.4 Izlaz dekodera za različite latentne vektore

1.2. Optimizacija u latentnom prostoru VAE

Postoji mnogo tradicionalnih algoritama optimizacije koji se bave rješavanjem složenih optimizacijskih problema. Međutim, povećanjem složenosti problema, vrijeme pronalaska optimalnog rješenja je duže te tradicionalni algoritmi postaju nepraktični. Primjena tradicionalnih optimizacijskih algoritma u naučenom latentnom prostoru umjesto u originalnom prostoru rješenja inovativan je pristup rješavanju složenih optimizacijskih problema.

U radu [3] predstavljen je postupak integracije evolucijskog učenja u latentni prostor varijacijskog autoenkodera, čime se smanjuje vrijeme pronalaska optimalnog rješenja problema. Prednost latentnog prostora u odnosu na originalni prostor rješenja je što se može istrenirati da bude smanjenih dimenzija i da sadrži pretežno visokokvalitetna rješenja problema.

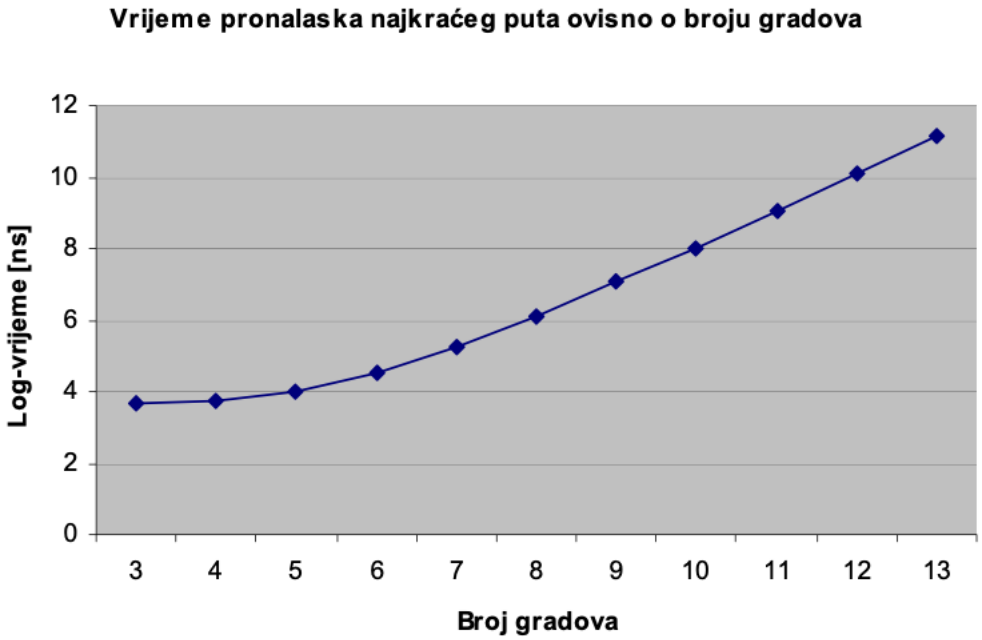
Za optimizaciju u latentnom prostoru potrebno je razviti odgovarajući enkoder, dekoder te optimizacijski algoritam (evolucijski algoritam) -

- Enkoder obrađuje primjerke problema i rješenja problema, učeći pritom latentni prostor bogat visokokvalitetnim rješenjima
- Dekoder na ulazu prima primjerak problema i latentni vektor koji predstavlja jednu od reprezentacija rješenja problema u latentnom prostoru te na izlazu daje potencijalno rješenje problema za koji je moguće ocijeniti kvalitetu rješenja
- Evolucijski algoritam koristi te latentne vektore kao populaciju, nad kojima se provode operacije selekcije, križanja i mutacije, a za izračun fitness funkcije koristi se dekoder koji svaku jedinku (latentni vektor) dekodira u originalni prostor u kojem se može ocijeniti kvaliteta rješenja

2. Evolucijsko učenje

Razvojem procesne moći računala, ljudi su počeli rješavati sve teže probleme koji su prije bili nerješivi. Najjednostavniji pristup rješavanju bilo kakvog problema je tehnikom grube sile (eng. brute force), tj. pretraživanjem cjelokupnog prostora rješenja. Današnja računala su ekstremno brza, pretražuju milijune potencijalnih rješenja u jednoj sekundi. Međutim, čak i uz današnja brza računala, tehnikom grube sile najčešće ne možemo riješiti složene optimizacijske probleme. [4]

NP-teški problemi su optimizacijski problemi čija složenost raste eksponencijalno s obzirom na povećanje veličine problema zbog čega predstavljaju veliki izazov prilikom pronalaska optimalnog rješenja. Problem trgovačkog putnika (eng. Travelling Salesman Problem, TSP) jedan je od najpoznatijih NP-teških problema gdje je cilj pronaći najkraći mogući put koji povezuje skup gradova uz uvjet da je svaki grad posjećen jednom i da se vrati u početni grad. U knjizi [4] prikazan je jednostavan program koji opisani problem rješava tehnikom grube sile te je na slici (Slika 2.1) prikazano vrijeme izvođenja (u logaritamskoj skali) s obzirom na broj gradova. Jasno se vidi da problem postaje eksponencijalno teži kako se broj gradova povećava.



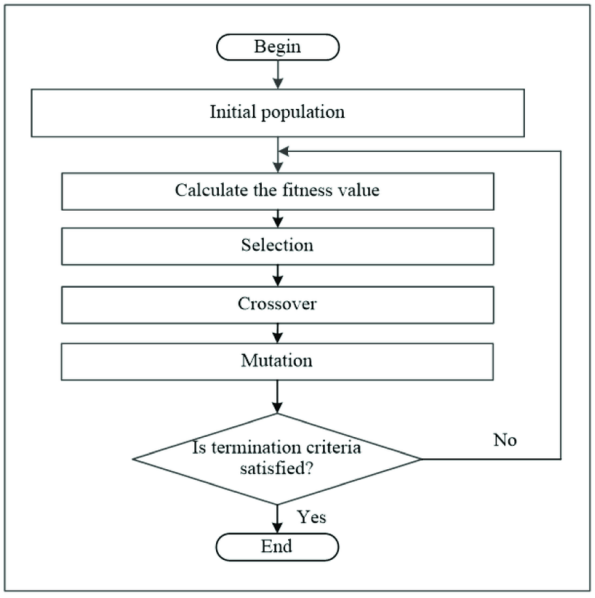
Slika 2.1 Vrijeme pronalaska najkraćeg puta kod problema trgovačkog putnika (logaritamska skala) [4]

Prilikom rješavanja optimizacijskih problema, najčešće nije potrebno doći do najboljeg rješenja, već dovoljno dobrog rješenja. Algoritmi koji pronalaze rješenja koja su zadovoljavajuće dobra, ali ne nude nikakve garancije da je pronađeno rješenje najbolje moguće nazivaju se heuristike. Metaheuristika je skup algoritama koji se koristi za definiranje heurističkih metoda, heuristika opće namjene. Evolucijski algoritmi su glavni predstavnici metaheuristika. [4]

2.1. Genetski algoritam

Darwinova teorija evolucije temelji se na ideji da će bolje i jače jedinke imati veću vjerojatnost preživljavanja i razmnožavanja. Prilikom razmnožavanja, djeca su određena većinom genetskim materijalom roditelja, no postoji i određeno odstupanje. Genetski algoritam inspiriran je tom idejom. Temelji se na mehanizmima genetske varijacije i prirodne selekcije, te simulira proces evolucije koristeći populaciju potencijalnih rješenja.

Genetski algoritam započinje inicijalizacijom nasumično odabranih jedinki koje čine početnu populaciju. Algoritam zatim iterativno provodi selekciju, križanje i mutaciju kako bi stvorio novu generaciju jedinki. Pri tom bolje jedinke imaju veću vjerojatnost selekcije te se time stvara generacija koja je potencijalno bolja od prethodne. Dobrota jedinke je mjera koliko je ta jedinka uspješna ili efikasna u rješavanju određenog problema. Proces se ponavlja kroz niz iteracija dok se ne zadovolji uvjet zaustavljanja (maksimalan broj evaluacija). Na slici (Slika 2.2) prikazan je opisani postupak.

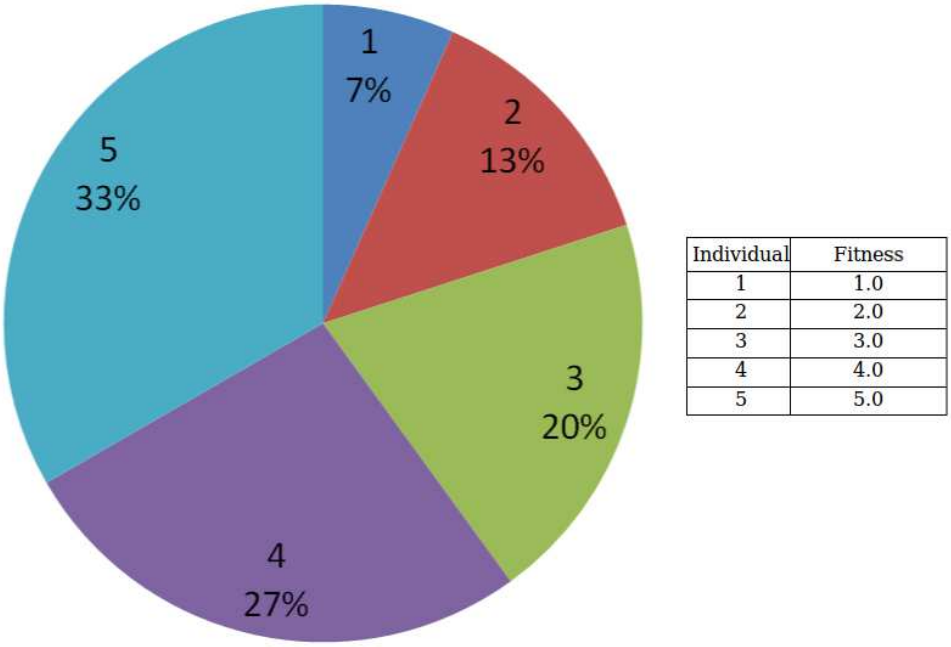


Slika 2.2 Dijagram tijeka genetskog algoritma [5]

2.1.1. Operator selekcije

Selekcija je postupak kojim se odabiru jedinke iz populacije, pri čemu prednost imaju bolje jedinke. Najčešći algoritmi selekcije su turnirska selekcija i proporcionalna selekcija.

- **k-turnirska selekcija** (eng. Tournament Selection) – nasumično se odabere skupina od k jedinki iz populacije te se za križanje bira ona jedinka iz skupine koja ima najbolju dobrotu
- **Proporcionalna selekcija** (eng. Roulette Wheel Selection) – svaka jedinka u populaciji ima šansu da bude izabrana proporcionalno njenoj dobroti – jedinke s većom dobrotom imaju veću vjerojatnost da budu odabrane za reprodukciju (Slika 2.3)



Slika 2.3 Proporcionalna selekcija [6]

Dobrota jedinke najčešće je određena mjerenjem funkcije koju optimiramo. Postoji dodatna mogućnost dodavanja selekcijskog pritiska (SP) gdje SP predstavlja omjer vjerojatnosti preživljavanja dobrih i loših jedinki [7]. Uz zadani SP, prikladan način definicije dobrote je:

$$F_i = 1 + (SP - 1)\frac{(f_i - f_{worst})}{(f_{best} - f_{worst})}$$

2.1.2. Operator križanja

Nakon što su odabrana dva roditelja, operatorom križanja stvaraju se djeca. S vjerojatnosti križanja p_c određeno je hoće li se križanje uistinu dogoditi.

Jedinke u genetskom algoritmu mogu se prikazati na različite načine, poput niza binarnih brojeva, realnog broja ili niza realnih brojeva. Operator križanja ovisi o vrsti jedinke. U ovom radu, jedinka je prikazana kao niz realnih brojeva te su ispitani različiti tipovi križanja prilagođeni tom prikazu:

- **Križanje s jednom točkom prekida** (eng. Single Point Crossover) – dva roditeljska kromosoma podijele se na jednom nasumično odabranom mjestu i razmijene segmente
- **Uniformno križanje** (eng. Uniform Crossover) – geni dva roditeljska kromosoma nezavisno se razmjenjuju s određenom vjerojatnošću za svaki gen, potomci nasljeđuju skup karakteristika oba roditelja
- **Simulirano binarno križanje** (eng. Simulated Binary Crossover) – simulira ponašanje križanja u realnom brojevnom prostoru, dva roditeljska kromosoma kombiniraju se na način da generiraju potomke čiji su geni realni brojevi, uzimajući u obzir distribuciju roditeljskih gena

2.1.3. Operator mutacije

Mutacija je proces nasumične promjene vrijednosti jednog ili više gena unutar kromosoma. S vjerojatnosti mutacije p_m određeno je hoće li se mutacija dogoditi. Ovaj mehanizam služi kako bi povećao raznolikost rješenja pomažući time u prevenciji lokalnih optimuma.

Postoje različite metode mutacije. U implementaciji rada se koristi Gaussova mutacija jer su jedinke sastavljene od realnih gena. **Gaussova mutacija** mijenja vrijednost gena jedinke dodavanjem vrijednosti dobivene iz Gaussove distribucije.

2.1.4. Operator lokalne pretrage Hooke-Jeeves

Algoritmi lokalne pretrage rješavanje problema započinju od nekog početnog rješenja, koje potom pokušavaju inkrementalno poboljšati [4]. Cilj je pronaći bolje rješenje istraživanjem neposredne okoline početnog rješenja.

Hooke-Jeeves postupak standardna je metoda lokalne pretrage. Nastoji riješiti problem spore konvergencije na funkcijama čije su konture zakrenute u odnosu na koordinatne osi [8]. U svakoj iteraciji algoritma koriste se tri točke za istraživanje prostora:

x_b – bazna točka

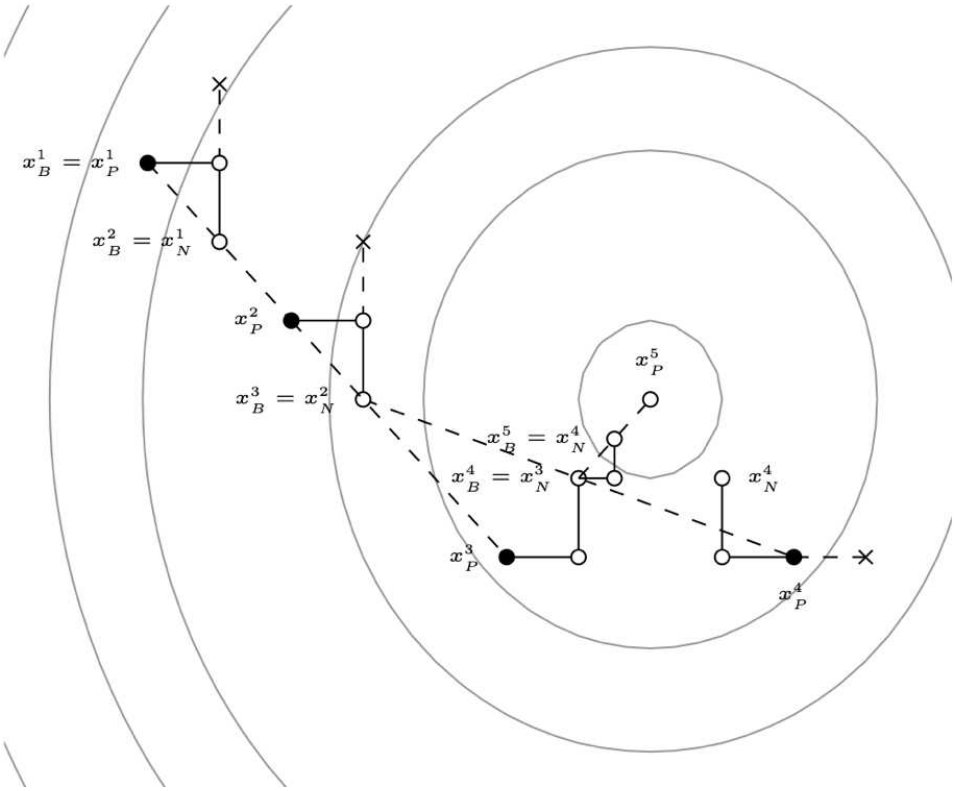
x_p – početna točka

x_n – nova točka

Postupak se sastoji od dvije faze koje se iterativno ponavljaju:

1. Faza lokalnog istraživanja – pretraživanje neposredne okoline početne točke. Početna točka pomiče se za pomak Δx po svakoj dimenziji. Postupak nastoji uštedjeti na broju evaluacija pa ne ispituje sve susjedne točke. Ako je algoritam za jednu dimenziju izračunao da je rješenje bolje u toj točki, odmah se pomiče u tu točku te računa pomak u drugim dimenzijama.
2. Faza refleksije – preslikavanje x_b preko nove točke x_n.. Točka pronađena u prvoj fazi služi kako bi odredila najbolji smjer u okolici početne točke. Druga faza koristi tu informaciju kako bi prebacila baznu točku preko nove točke te time dobila točku pretraživanja za iduću iteraciju.

Na slici (Slika 2.4) prikazan je primjer optimizacije jedne funkcije korištenjem Hooke-Jeeves postupka.



Slika 2.4 Hooke-Jeeves postupak [8]

2.2. Algoritam diferencijske evolucije

Diferencijska evolucija je optimizacijski evolucijski algoritam napravljen za optimiziranje funkcija s realnim varijablama [9]. U diferencijskoj se evoluciji, kao i u genetskom algoritmu, također koristi operator mutacije i križanja, ali postoje razlike u mehanizmu izvođenja.

Na početku se generira početna populacija slučajno odabranih jedinki. Iterira se kroz sve jedinke unutar populacije te se svaka jedinka uspoređuje s novo dobivenim vektorom. Odabiru se tri slučajno odabrana vektora iz kojih se stvara novi vektor na način da se dva od tri vektora oduzmu, razlika se pomnoži s faktorom skaliranje te se sve doda trećem vektoru. Dobiveni vektor uspoređuje se s trenutnim vektorom te se zamjenjuje isključivo ako je bolji od trenutnog. Postupak se ponavlja dok se ne zadovolji uvjet zaustavljanja (maksimalan broj evaluacija).

Razlika između genetskog algoritma i algoritma diferencijske evolucije:

1. Reprezentacija rješenja – genetski algoritam koristi binarne, cjelobrojne i realne gene unutar kromosoma, dok diferencijska evolucija koristi isključivo realne gene
2. Generiranje potomaka – u genetskom algoritmi potomci se generiraju koristeći operacije križanja i mutacije, dok se u diferencijskoj evoluciji potomci stvaraju na način da se odaberu tri slučajno odabrana vektora iz populacije, oduzmu se druga dva, razlika se množi s faktorom skaliranja te se dodaje prvom vektoru
3. Selekcija – genetski algoritam koristi razne metode selekcije (k-turnirska, proporcionalna), dok diferencijska evolucija uvijek koristi jednostavnu selekciju – novo generirani vektor se uspoređuje s trenutnim te u sljedeću generaciju ide bolji

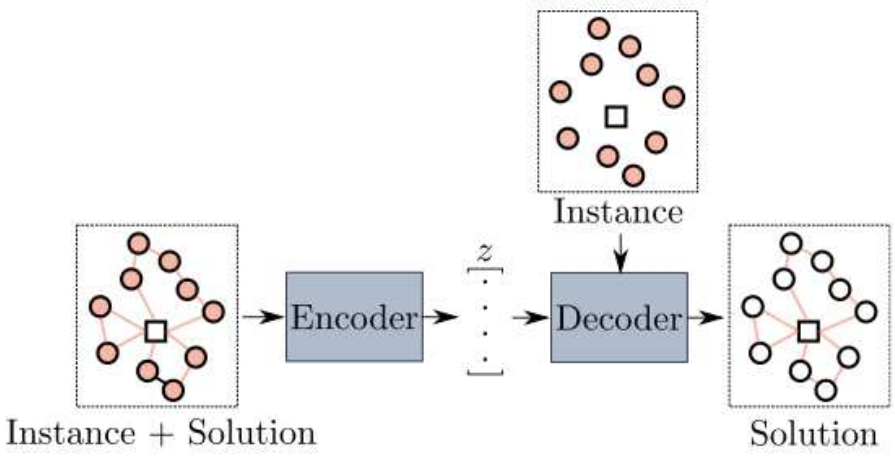
3. Programska izvedba

Ideja programske izvedbe opisana je u radu [3]. Pristup je sljedeći – potrebno je implementirati varijacijski autoenkoder koji mapira optimizacijski problem (problem trgovačkog putnika) u n-dimenzionalni latentni prostor. S obzirom da je VAE treniran samo na visokokvalitetnim rješenjima, latentni prostor je, za razliku od standardnog prostora rješenja, bogat kvalitetnim rješenjima problema. U radu je navedeno da se u istreniranom latentnom prostoru zatim može primijeniti bilo koji kontinuirani optimizacijski algoritam kako bi se pronašlo optimalno rješenje problema.

3.1. Pregled postojećeg rješenja

Postupak treniranja VAE prikazan je na slici (Slika 3.1). Enkoder na ulazu prima primjerak optimizacijskog problema (npr. graf čiji čvorovi predstavljaju gradove, a bridovi udaljenost između gradova) te visokokvalitetno rješenje tog problema (niz gradova po redu koje trgovac mora posjetiti). Enkoder na izlazu daje distribuciju kojom je kodirano ulazno rješenje problema.

Kako bi dekoder mogao konstruirati rješenje problema, potrebno mu je na uzlazu dati primjerak problema i slučajno uzorkovani vektor iz dobivene distribucije (latentnog prostora). Iako je dekoder dovoljno snažan da bi mogao pronaći relativno dobro rješenje problema samo iz primjerka problema, na ulaz mu se također dovodi latentni vektor kako bi se osiguralo da dekoder na izlaz za različite latentne vektore generira različita rješenja.

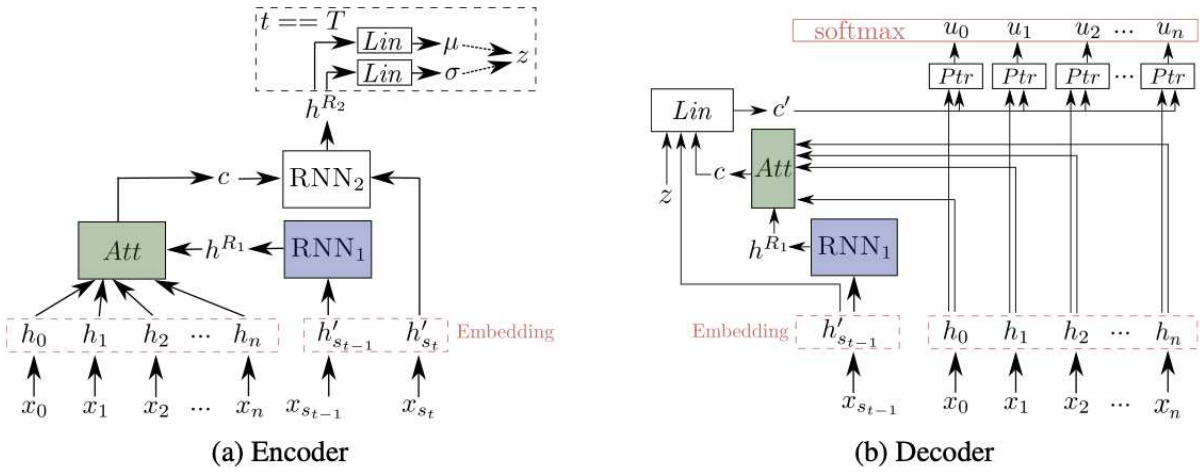


Slika 3.1 Postupak treniranja varijacijskog autoenkodera [3]

3.1.1. Model

Arhitektura enkodera i dekodera prikazana je na slici (Slika 3.2).

Enkoder i dekoder implementirani su kao neuronske mreže. U radu [10] problemi s usmjeravanjem (TSP) modelirani su kao Markovljevi procesi odlučivanja (eng. Markov decision process) gdje se rješenje konstruira nizom radnji (tj. koji čvor treba posjetiti sljedeći). Rad [3] slijedi taj pristup – dekoder se trenira tako da se u svakom koraku odredi koja se lokacija (idući čvor, grad) treba dodati konačnom rješenju. Enkoder također slijedi opisani pristup konstruiranja rješenja nizom radnji – u svakom koraku generira se jedna dimenzija latentne varijable iz danog rješenja problema.

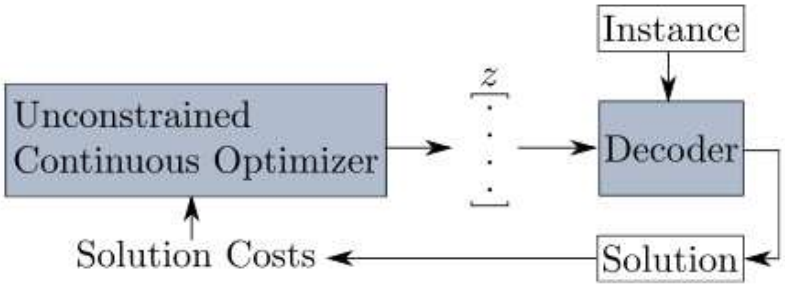


Slika 3.2 Model implementiran u radu [3]

Što se tiče same arhitekture mreže, i enkoder i dekoder koriste *linearni embedding layer* i *attention mechanism* za sekvencijalno kodiranje i dekodiranje rješenja. Težine (eng. weights) mreže se dijele između identičnih komponenti enkodera i dekodera kako bi se omogućilo brže treniranje, ali i zajednički pogled na primjerak problema. Detalji arhitekture opisani su u radu [3].

3.2. Integracija genetskog algoritma u latentni prostor VAE

Kao što je spomenuto ranije, pronalazak optimalnog rješenja problema provodi se u latentnom prostoru istreniranog varijacijskog autoenkodera. Postupak integracije optimizacijskog algoritma u latentni prostor prikazan je na slici (Slika 3.3).



Slika 3.3 Integracija optimizacijskog algoritma u latentni prostor [3]

U radu je kao kontinuirani optimizacijski algoritam implementiran genetski algoritam potpomognut operatorom lokalne pretrage Hooke-Jeeves. Također se za usporedbu rezultata koristi slučajna pretraga prostora rješenja i algoritam diferencijske evolucije.

Algoritam započinje izvođenje definiranjem inicijalne populacije koja je sastavljena od n slučajno uzorkovanih vektora iz latentnog prostora.

Dobrota svake jedinke unutar populacije računa se na način da se na ulaz dekodera dovede latentni vektor (jedinka) i primjerak problema koji je služio za treniranje enkodera i dekodera. Dekoder na izlazu daje rješenje problema koje je kodirano tim latentnim vektorom. Rješenje problema zapravo je niz gradova koje trgovac mora posjetiti, a kvaliteta rješenja određuje se kao euklidska udaljenost niza gradova.

Potrebno je napomenuti da se za evaluaciju populacije koristi mogućnost izračunavanja kvalitete čitave populacije odjednom, budući da dekoder istovremeno prima instancu problema i *batch_size* vektora koji predstavljaju potencijalna rješenja te ih sve zajedno dekodira.

Nakon evaluacije, nad jedinkama se provode operatori selekcije, križanja i mutacije kako bi se napravila nova generacija jedinki. U algoritmu se koristi princip elitizma – određeni

postotak najboljih jedinki iz trenutne generacije direktno prelazi u sljedeću generaciju, osiguravajući tako očuvanje visokokvalitetnih rješenja.

Također, unutar genetskog algoritma dodan je i operator lokalne pretrage Hooke-Jeeves opisan u poglavlju 2.1.4. koji se provodi u svakoj petoj generaciji algoritma. Tijekom izvođenja Hooke-Jeeves postupka potrebno je računati kvalitetnu pojedinačnih jedinki, a ne cijele populacije. S obzirom da je dekoder napravljen kao neuronska mreža koja istodobno prima *batch_size* jedinki, povećanje broja jedinki i vrijeme izvođenja nije linearno povezano – sto evaluacija jedne jedinke mnogo je sporije od jedne evaluacije sto jedinki. Zbog toga se operator primjenjuje samo na najbolju jedinku u svakoj petoj iteraciji te je broj evaluacija operatora ograničen.

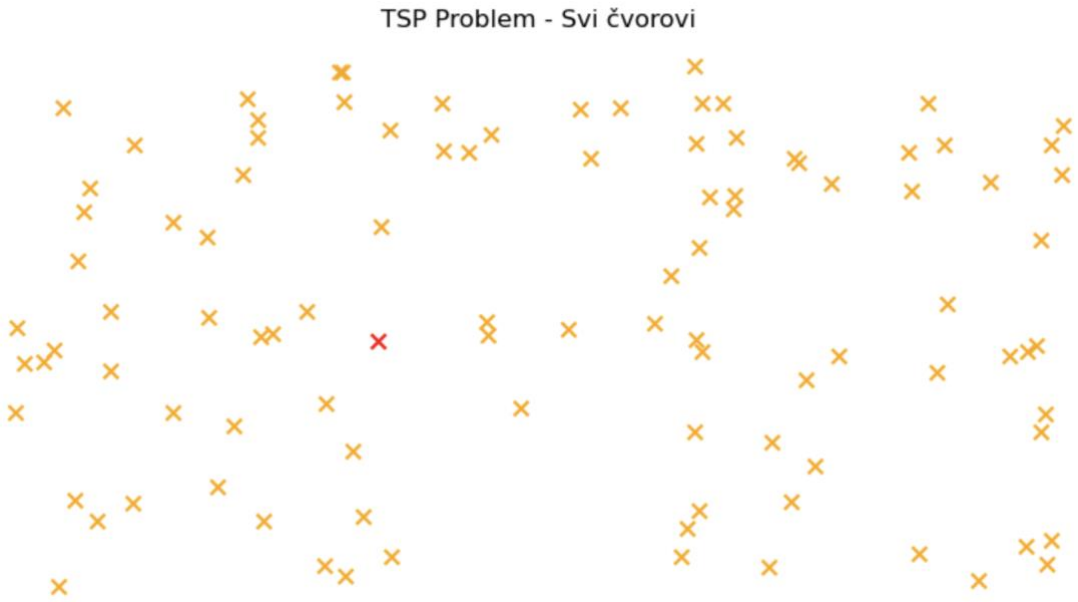
U idućem poglavlju prikazana je optimizacija svih hiperparametara algoritma, analiza efikasnosti genetskog algoritma s optimiziranim hiperparametrima i usporedba s drugim optimizacijskim algoritmima.

4. Rezultati

U ovom poglavlju prikazani su eksperimenti koji su pokrenuti nad opisanom implementacijom.

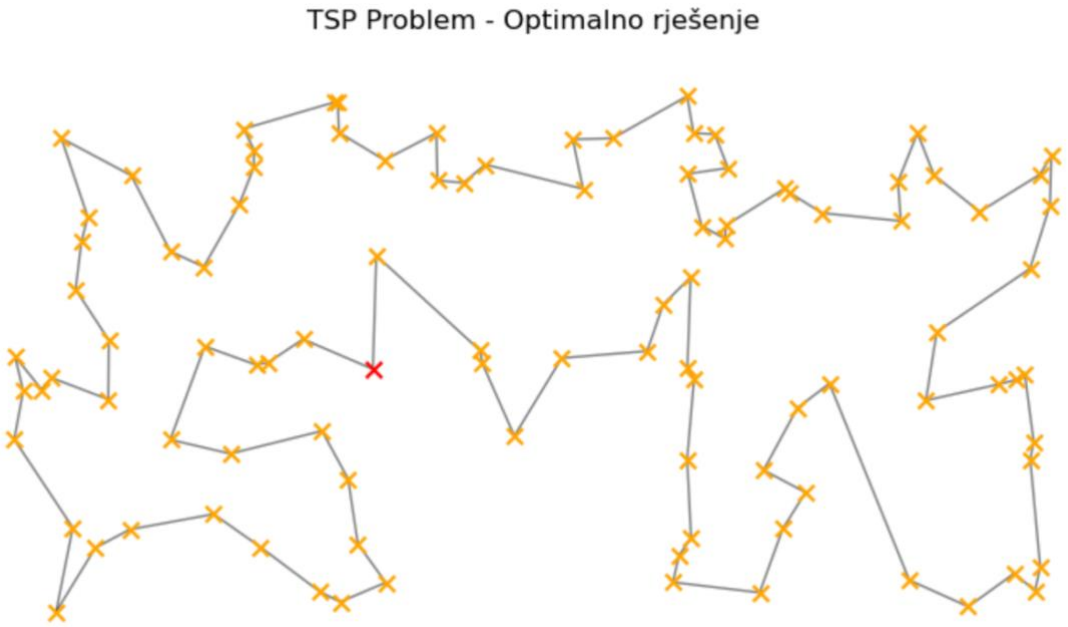
Prvo je napravljena optimizacija hiperparametara genetskog algoritma i Hooke-Jeeves postupka, a zatim je provedena optimizacija problema genetskim algoritmom, algoritmom diferencijske evolucije i slučajnim pretraživanjem prostora.

Optimizacijski problem koji je korišten u implementaciji je problem trgovačkog putnika (TSP) sastavljen od 100 gradova. Problem je prikazan na slici (Slika 4.1). Problem je definiran na način da je svaki grad predstavljen pomoću x i y koordinate u 2D prostoru. Svi gradovi su međusobno povezani. Udaljenost između gradova je definirana kao euklidska udaljenost između dvije odgovarajuće točke u 2D prostoru. Crvenim križićem je označen početni grad – iz kojeg trgovac kreće i u koji se mora vratiti.



Slika 4.1 Svi čvorovi TSP problema

Uz dani problem, poznato je i optimalno rješenje problema prikazano na slici (Slika 4.2). Definirani problem zove se *kroA100*, preuzet je iz [11] te je dokazano da je navedeno rješenje optimalno rješenje danog problema.



Slika 4.2 Optimalna ruta

U idućim poglavljima kvaliteta svih izgeneriranih rješenja prikazana je kao razlika euklidske udaljenosti niza gradova u rješenju i optimalnog rješenja – cilj optimizacijskih algoritama je pronaći jedinku čija je fitness funkcija jednaka nuli.

Važno je istaknuti da je korišteni model varijacijskog autoenkodera treniran pomoću dataseta koji se sastoji od 99,999 različitih TSP problema od 100 gradova i kvalitetnih rješenja tih problema. Kao što je opisano u radu [3], TSP problemi i rješenja problema generirani su pomoću generatora iz rada [13]. Prilikom treniranja modela korišten je latentni prostor od 100 dimenzija. Prilikom učenja, enkoder i dekoder trenirani su na način da enkoder prima instancu TSP problema i rješenje tog problema te na izlazu daje parametre koji definiraju latentni prostor. Dekoder pomoću te iste instance problema i slučajno uzorkovanog vektora iz latentnog prostora generira potencijalno rješenje problema. Cilj treniranja je maksimizirati (log-) vjerojatnost rekonstrukcije rješenja za danu instancu problema. Prilikom treniranja modela korištena su samo visokokvalitetna rješenja i jako velik broj različitih TSP problema. Zbog toga latentni prostor ima svojstvo da za slučajno uzorkovane latentne vektore i neviđene TSP probleme, dekoder generira rješenje koje je mnogo kvalitetnije od slučajno odabranog rješenja u realnom prostoru (detaljnije prikazano u nastavku rada) – latentni prostor bogat je visokokvalitetnim rješenjima.

Jednom istrenirani model korišten je kako bi se pronašlo optimalno rješenje za problem prikazan na slici (Slika 4.1). Prikazani problem nije korišten tijekom treniranja modela, nego se koristi isključivo za testiranje i demonstraciju sposobnosti modela na neviđenim problemima. Evolucijski algoritam koristi slučajno uzorkovane vektore latentnog prostora kao početnu populaciju, nad njima primjenjuje operatore selekcije, križanja i mutacije te koristi dekoder kako bi odredio kvalitetu jedinki. Prilikom određivanja kvalitete jedinki, algoritam na ulaz dekodera dovodi jedinku prikazanu kao latentni vektor i instancu problema (koju dekoder nije vidio tijekom učenja), a dekoder na izlazu generira potencijalno rješenje problema koje se zatim uspoređuje s optimalnim rješenjem prikazanim na slici (Slika 4.2).

4.1. Optimizacija hiperparametara

Optimizacija hiperparametara provedena je za genetski algoritam i Hooke-Jeeves postupak. Eksperimenti su dizajnirani na način da su testirane sve kombinacije hiperparametara kako bi se pronašla kombinacija hiperparametara za koju algoritmi daju najbolje rezultate. Svi testovi su ograničeni maksimalnim brojem evaluacija.

4.1.1. Hiperparametri genetskog algoritma

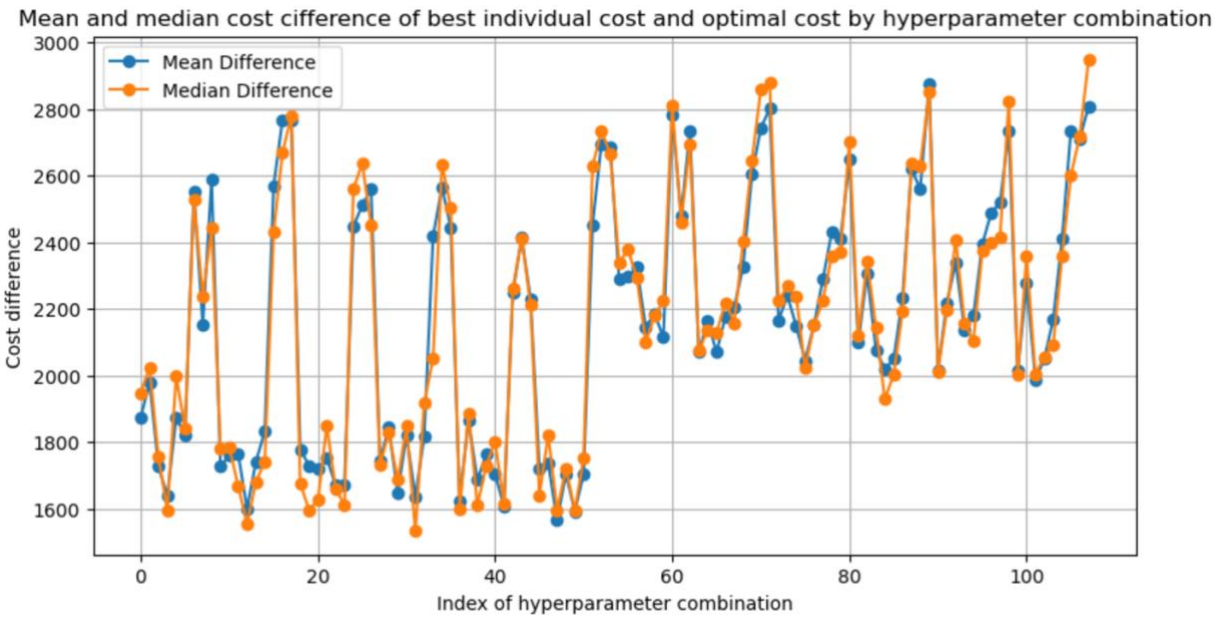
Hiperparametri genetskog algoritma za koje je provedena optimizacija vrijednosti su sljedeći:

- Veličina populacije – *pop_size*
- Veličina elitne skupine jedinki (izražena kao postotak veličine populacije) – *elite*
- Algoritam križanja – *cx_alg*
- Vjerojatnost da će doći do križanja – *cx_rate*
- Vjerojatnost da će doći do mutacije – *mut_rate*

Uzet je maksimalan broj evaluacija od 2000, a vrijednosti hiperparametara za koje je provedeno testiranje su sljedeće:

- *pop_size*: [100, 300]
- *elite*: [0.1, 0.2, 0.3]
- *cx_alg*: [single point crossover, uniform crossover, simulated binary crossover]
- *cx_rate*: [0.8, 0.9]
- *mut_rate*: [0.05, 0.1, 0.2]

Rezultat testiranja prikazan je na slici (Slika 4.3). Za svaku kombinaciju hiperparametara pokrenut je genetski algoritam te je zapamćena kvaliteta najbolje jedinke koja je pronađena. Proces je ponovljen 5 puta za svaku kombinaciju hiperparametara i na grafu je prikazana srednja vrijednost (eng. mean) najboljih jedinki i medijan (eng. median) najboljih jedinki.



Slika 4.3 Rezultat testiranja različitih kombinacija hiperparametara

Analizom dobivenih podataka utvrđeno je da se optimalna kombinacija hiperparametara razlikuje ovisno o tome koristi li se srednja vrijednost ili medijan najboljih jedinki.

U slučaju srednje vrijednosti, optimalni hiperparametri su:

- *pop_size*: 100
- *elite*: 0.3
- *cx_alg*: single point crossover
- *cx_rate*: 0.9
- *mut_rate*: 0.2

koji odgovaraju točki na grafu s indeksom 47.

U slučaju medijana, optimalni hiperparametri su:

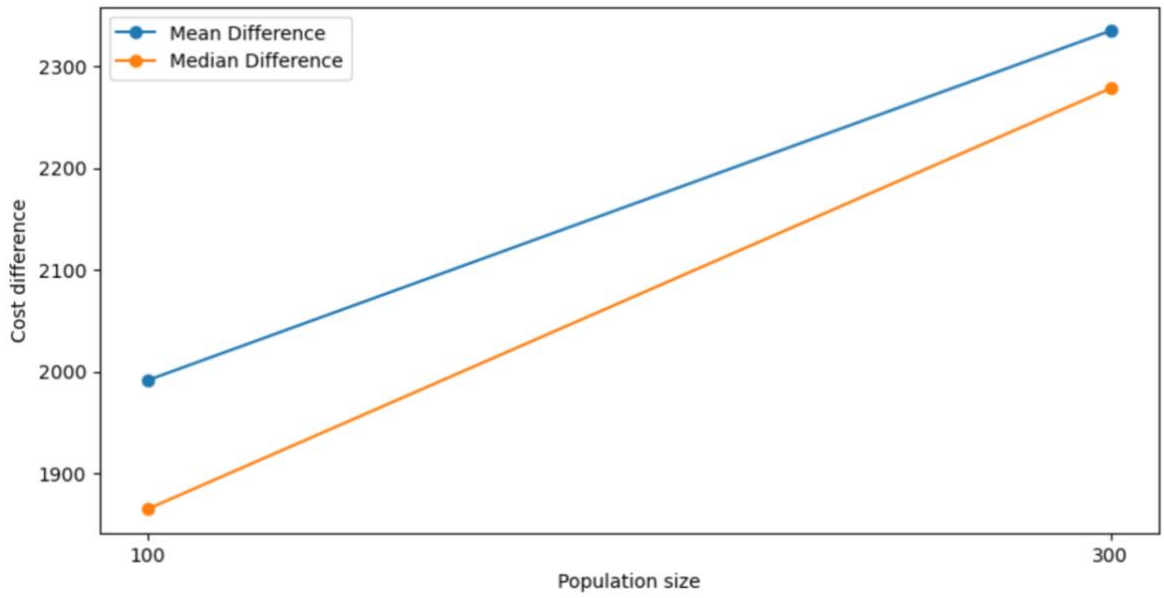
- *pop_size*: 100

- *elite*: 0.2
- *cx_alg*: uniform crossover
- *cx_rate*: 0.9
- *mut_rate*: 0.1

koji odgovaraju točki na grafu s indeksom 31.

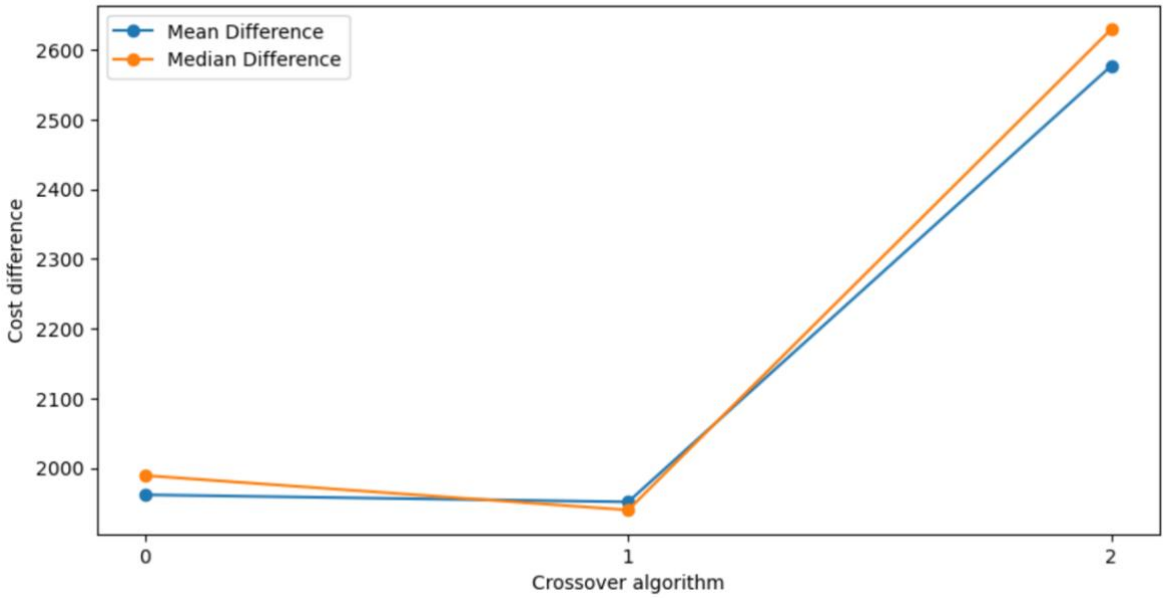
U grafu se pojavljuju dvije zanimljive oscilacije – prva polovica grafa pokazuje znatno nižu vrijednost u odnosu na drugu, te unutar svake grupe od devet uzastopnih podataka, zadnja tri podatka pokazuju znatno veću vrijednost od prethodnih šest.

Detaljnom analizom utvrđeno je da prva polovica kombinacija ima znatno bolje rezultate od druge polovice jedinki zato što je u prvoj polovici korištena veličina populacije od 100, a u drugoj polovici od 300 jedinki. U svrhu analize opažanja, dobiveni rezultati testiranja grupirani su isključivo po veličini populacije te je na grafu (Slika 4.4) prikazana razlika u dobroti jedinki u slučaju veličine populacije od 100 i 300. Sa slike se jasno vidi da su bolji rezultati algoritma ako je veličina populacije 100. Algoritam je ograničen maksimalnim brojem evaluacija te se može zaključiti da ima bolje rezultate ako se broj evaluacija „troši“ na nove generacije, a ne na jedinke iste populacije.



Slika 4.4 Razlika u kvaliteti jedinki prilikom korištenja različitih veličina populacije

Analizom drugog opažanja (unutar svake grupe od devet uzastopnih podataka, zadnja tri podatka pokazuju znatno veću vrijednost od prethodnih šest) utvrđeno je da su nagli skokovi u grafu rezultat različitog algoritma križanja. Dobiveni rezultati grupirani su isključivo po algoritmu križanja i rezultat je prikazan na grafu (Slika 4.5). Vrste križanja su kodirane na sljedeći način: 0 – *Single Point Crossover*, 1 – *Uniform Crossover*, 2 - *Simulated Binary Crossover*. Algoritam daje znatno lošije vrijednosti ako se koristi algoritam križanja *Simulated Binary Crossover*.



Slika 4.5 Razlika u kvaliteti jedinki prilikom korištenja različitih vrsta križanja

Detaljnom analizom ostalih hiperparametara nisu pronađene veća odstupanja.

4.1.2. Hiperparametri Hooke-Jeeves postupka

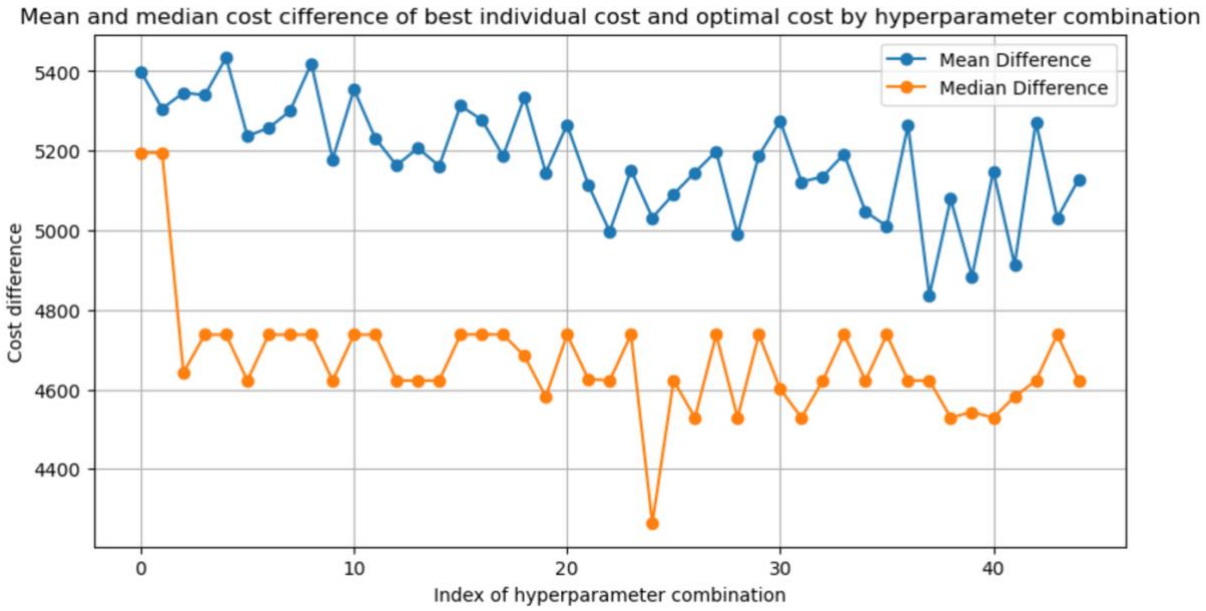
Hiperparametri Hooke-Jeeves postupka za koje je provedena optimizacija vrijednosti su sljedeći:

- Veličina pomaka po svakoj dimenziji – *Dx*
- Postotak dimenzija vektora koji se mijenja – *x_ratio*
- Minimalna vrijednost do koje pomak može ići – *epsilon*

Uzet je maksimalan broj evaluacija od 30, a vrijednosti hiperparametara za koje je provedeno testiranje su sljedeće:

- *Dx*: [0.1, 0.2, 0.3, 0.4]
- *x_ratio*: [0.05, 0.1, 0.2]
- *epsilon*: [0.01, 0.03, 0.05, 0.1]

Rezultat testiranja prikazan je na slici (Slika 4.6). Za svaku kombinaciju hiperparametara pokrenut je Hooke-Jeeves postupak te je zapamćena kvaliteta najbolje jedinke koja je pronađena. Proces je ponovljen 5 puta za svaku kombinaciju hiperparametara i na grafu je prikazana srednja vrijednost (eng. mean) najboljih jedinki i medijan (eng. median) najboljih jedinki.



Slika 4.6 Rezultat testiranja različitih kombinacija hiperparametara

Analizom dobivenih podataka utvrđeno je da se optimalna kombinacija hiperparametara razlikuje ovisno o tome koristi li se srednja vrijednost ili medijan najboljih jedinki. U slučaju srednje vrijednosti, optimalni hiperparametri su:

- *Dx*: 0.4
- *x_ratio*: 0.1
- *epsilon*: 0.01

koji odgovaraju točki na grafu s indeksom 37.

U slučaju medijana, optimalni hiperparametri su:

- *Dx*: 0.3
- *x_ratio*: 0.05
- *epsilon*: 0.1

koji odgovaraju točki na grafu s indeksom 24.

Analizom grafa (Slika 4.6) utvrđeno je da postoji značajna razlika između srednje vrijednosti i medijana, što ukazuje na velike varijacije u rezultatima pet provedenih testiranja. Ovaj fenomen može se objasniti karakteristikama Hooke-Jeeves postupka, koji po teoriji istražuje isključivo lokalni prostor oko početne jedinke. Zbog toga se početna kvaliteta jedinke ne može značajno promijeniti u odnosu na novopronađenu jedinku, tj. ako je postojala velika varijacija između slučajno odabranih jedinki prije provođenja postupka, postojat će i velika varijacija u rezultatima nakon provođenja postupka.

4.2. Analiza dobivenih rezultata

U ovom poglavlju detaljno se analiziraju i uspoređuju rezultati dobiveni genetskim algoritmom kojem je dodan operator lokalne pretrage Hooke-Jeeves, genetskim algoritmom bez lokalne pretrage, te usporedba sa slučajnim pretraživanjem i algoritmom diferencijske evolucije. U svim testovima uzet je maksimalni broj evaluacija od 500,000.

4.2.1. Analiza različitih optimalnih hiperparametara

Zbog razlike u srednjoj vrijednosti i medijanu kod optimizacije hiperparametara, prvo je provedena usporedba genetskog algoritma s operatorom lokalne pretrage s optimalnim hiperparametrima srednje vrijednosti i medijana. Vrijednosti hiperparametara u slučaju srednje vrijednosti su:

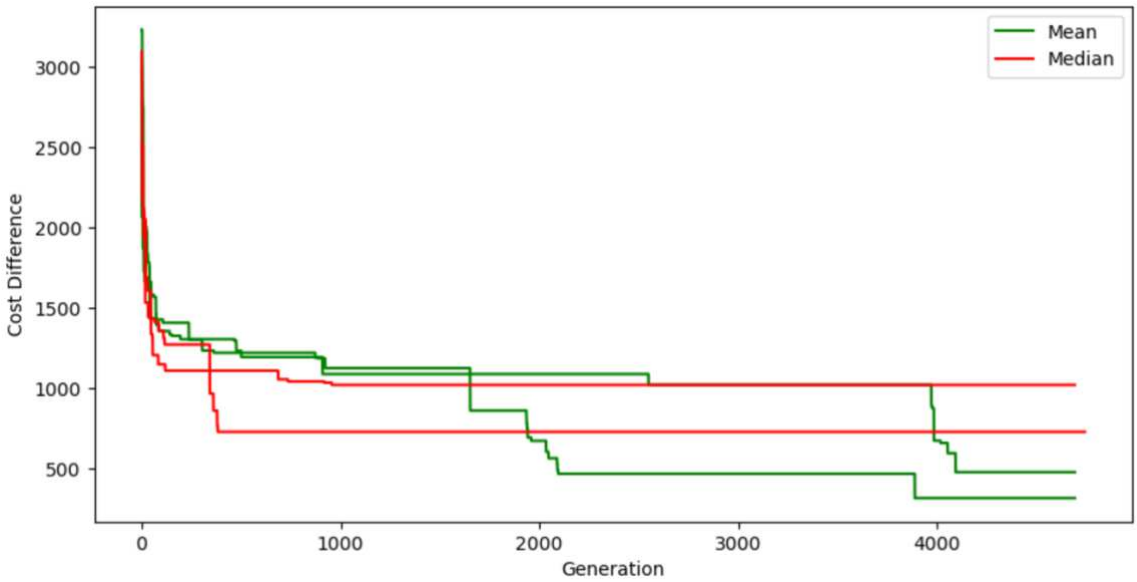
- *pop_size*: 100
- *elite*: 0.3
- *cx_alg*: single point crossover
- *cx_rate*: 0.9

- *mut_rate*: 0.2
- *Dx*: 0.4
- *x_ratio*: 0.1
- *epsilon*: 0.01

a u slučaju medijana su:

- *pop_size*: 100
- *elite*: 0.2
- *cx_alg*: uniform crossover
- *cx_rate*: 0.9
- *mut_rate*: 0.1
- *Dx*: 0.3
- *x_ratio*: 0.05
- *epsilon*: 0.1

Na grafu (Slika 4.7) prikazani su rezultati analize. Genetski algoritam pokrenut je dva puta za svaku kombinaciju hiperparametara te je u oba slučaja došao do boljeg rješenja za optimalnu vrijednosti hiperparametara dobivenu u slučaju korištenja srednje vrijednosti.



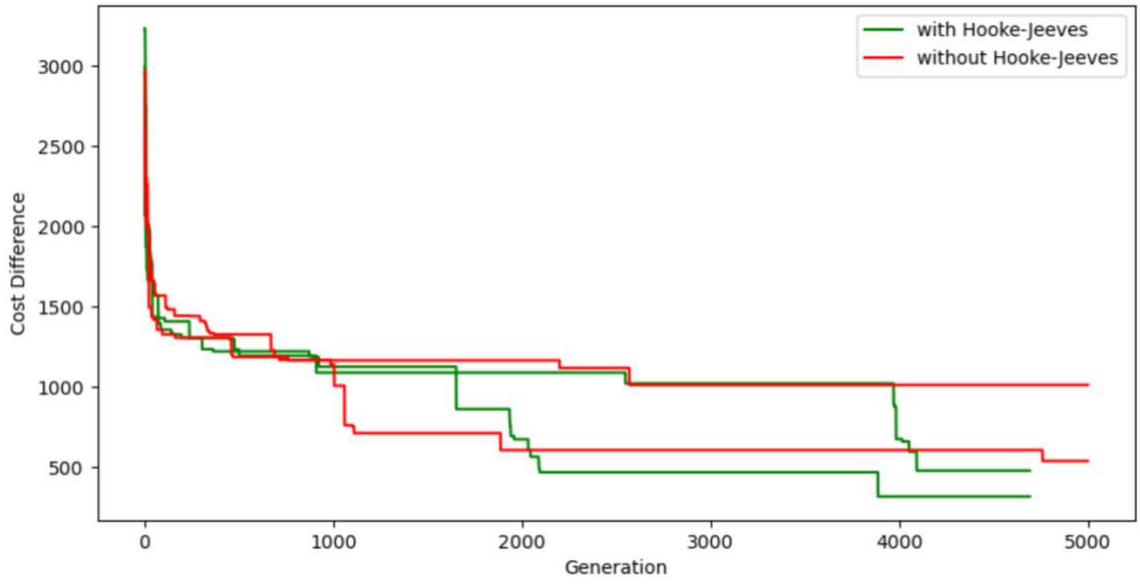
Slika 4.7 Rezultat izvođenja genetskog algoritma za različite vrijednosti hiperparametara

Zbog toga su hiperparametri dobiveni minimizacijom srednje vrijednosti rezultata uzeti kao optimalni hiperparametri za buduća testiranja.

4.2.2. Analiza doprinosa operatora lokalne pretrage Hooke-Jeeves genetskom algoritmu

Također je napravljena usporedba genetskog algoritma s operatorom lokalne pretrage Hooke-Jeeves i klasičnog genetskog algoritma bez operatora lokalne pretrage. Rezultati usporedbe prikazani su na slici (Slika 4.8).

Iz analize prikazane na grafu evidentno je da genetski algoritam ostvaruje bolje rezultate kada se koristi u kombinaciji s Hooke-Jeeves metodom lokalne pretrage. Ova kombinacija dva pristupa optimizacije rezultirala je pronalaskom efikasnijeg rješenja, što ukazuje na prednost integriranja genetskog algoritma i operatora lokalne pretrage.



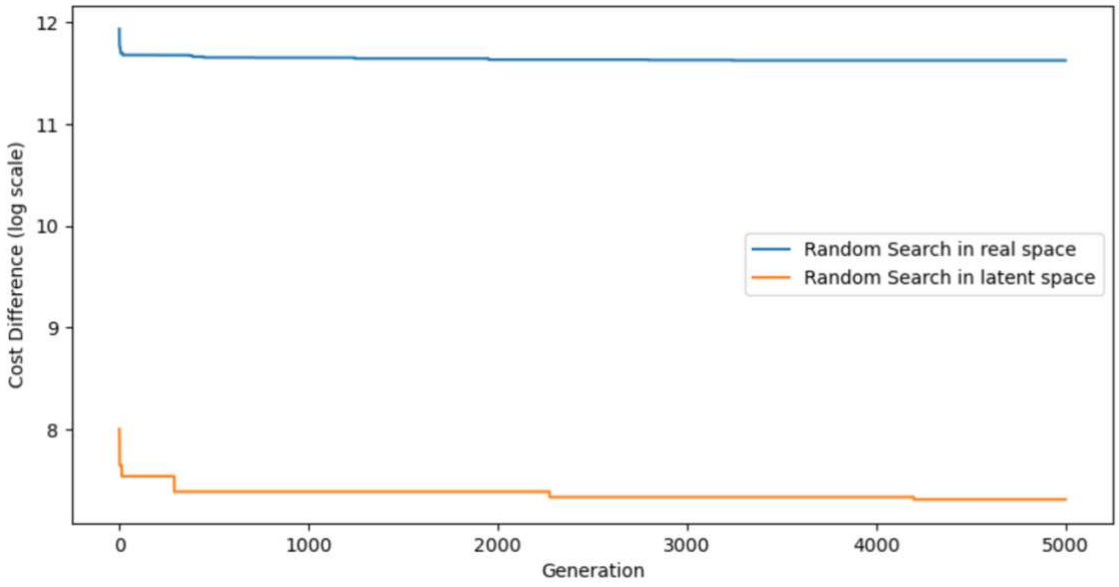
Slika 4.8 Rezultati izvođenja genetskog algoritma s operatorom lokalne pretrage i bez operatora lokalne pretrage

4.2.3. Usporedba različitih optimizacijskih algoritama

Zadnja usporedba koja je napravljena je usporedba različitih algoritama pretraživanja prostora. Pokrenuti su eksperimenti pretraživanja latentnog prostora rješenja pomoću

genetskog algoritma s Hooke-Jeeves postupkom, diferencijske evolucije te slučajnog pretraživanja u latentnom prostoru i realnom prostoru rješenja. Za svaki algoritam eksperiment je pokrenut dva puta i na grafovima su prikazane srednje vrijednosti rezultata za pojedini algoritam.

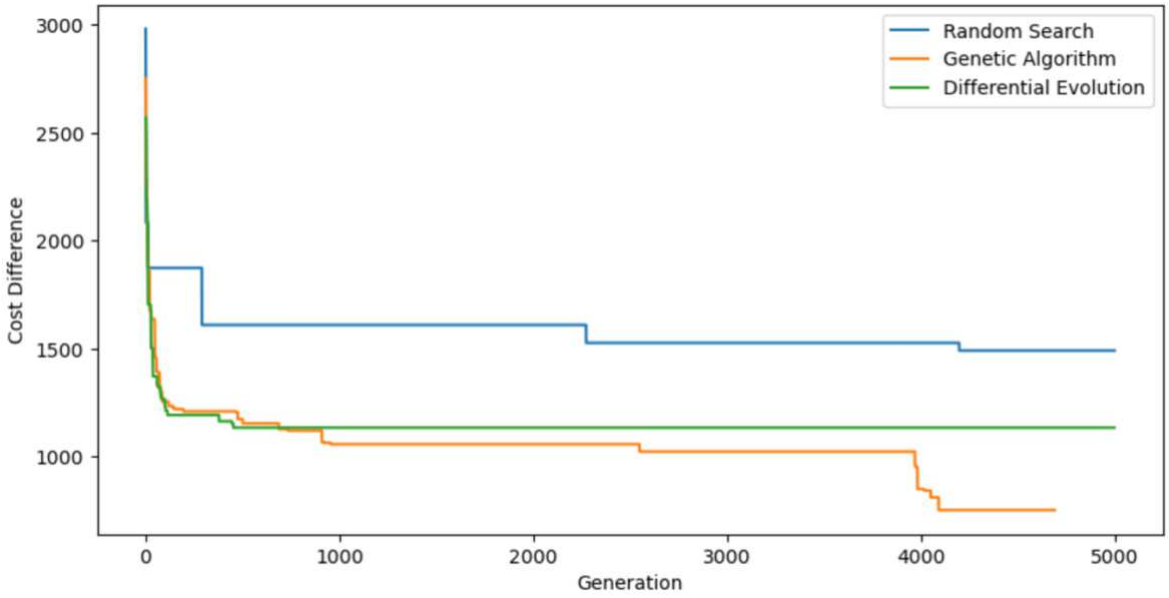
Graf na slici (Slika 4.9) prikazuje usporedbu metoda slučajnog pretraživanja kroz dva različita prostora: latentni prostor i realni prostor rješenja. Kvaliteta jedinki prikazana je u logaritamskoj skali. Latentni prostor, kako je spomenuto ranije, naučen je da sadrži isključivo visokokvalitetna rješenja. Ova karakteristika je ključna jer omogućuje efikasnije pretraživanje optimalnih rješenja u usporedbi s pretraživanjem realnog prostora, gdje su sva moguća rješenja. Graf prikazuje kako se ova karakteristika latentnog prostora manifestira u praksi – slučajno pretraživanje u latentnom prostoru daje znatno bolje rezultate.



Slika 4.9 Usporedba slučajne pretrage latentnog i realnog prostora rješenja (logaritamska skala)

Graf prikazan na slici (Slika 4.10) prikazuje usporedbu različitih algoritama pretrage prostora rješenja. Svi algoritmi pretražuju latentni prostor, i ograničeni su na 500,000 evaluacija. Iz grafa se jasno uočava da genetski algoritam, u kombinaciji s Hooke-Jeeves postupkom, ostvaruje najbolje rezultate. Usporedbom genetskog algoritma i algoritam diferencijske evolucije, može se zaključiti kako oba pristupa imaju slične performanse sve do završne faze (oko generacije 4000), gdje genetski algoritam uspješno pronalazi nove

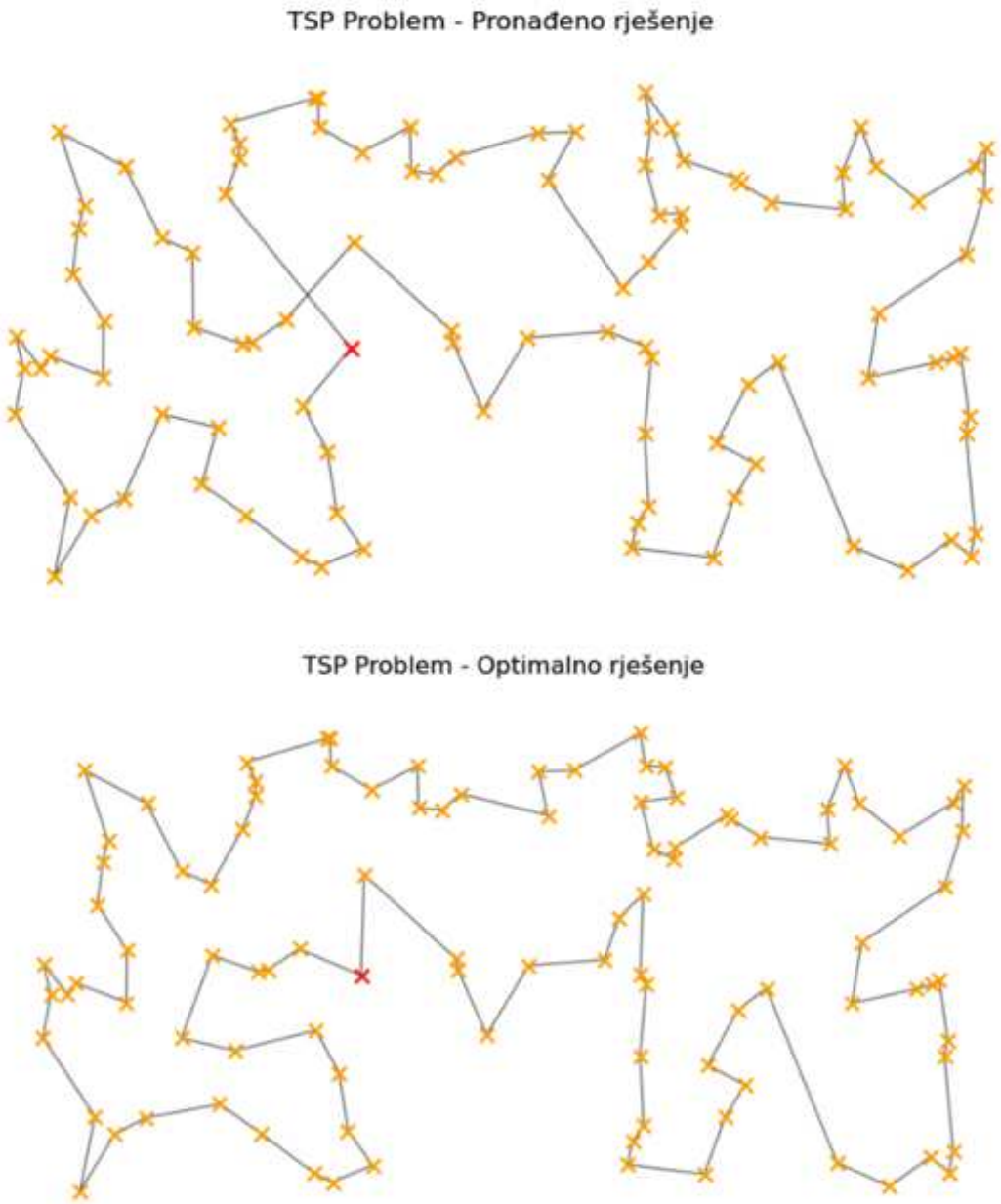
najbolje jedinke i izlazi iz lokalnog optimuma, a algoritam diferencijske evolucije ostaje u lokalnom optimumu. Ovo opažanje je zanimljivo jer se razlikuje od rezultata dobivenih u radu [12], iz kojih se zaključuje da genetski algoritam češće ostaje zarobljen u lokalnim optimumima od algoritma diferencijske evolucije.



Slika 4.10 Usporedba različitih optimizacijskih algoritama

4.3. Prikaz najbolje pronađene jedinke

Nakon analize testova, pokazalo se da genetski algoritam u kombinaciji s Hooke-Jeeves postupkom daje najbolje rezultate. Međutim, algoritam nije uspio pronaći optimalnu jedinku koja predstavlja najbolje rješenje problema. Na slici (Slika 4.11) prikazana je usporedba najbolje pronađene jedinke u ovom radu (najkraća ruta koju je algoritam uspio pronaći) i optimalno rješenje danog problema (najkraća postojeća ruta za dani problem). Fitness prikazane najbolje pronađene jedinke iznosi 21766 (dok je fitness optimalne jedinke 21285).



Slika 4.11 Usporedba najboljeg pronađenog rješenja i optimalnog rješenja problema

Zaključak

U okviru ovog diplomskog rada, razvijen je i implementiran novi pristup optimizaciji problema trgovačkog putnika (TSP) koristeći genetski algoritam ojačan Hooke-Jeeves postupkom unutar latentnog prostora. Ovaj pristup temelji se na korištenju varijacijskog autoenkodera koji uči iz primjeraka TSP problema i njihovih rješenja te kreira latentni prostor koji omogućuje efikasnije pretraživanje rješenja. S obzirom da se za treniranje modela koriste optimalna rješenja problema, latentni prostor je bogat visokokvalitetnim rješenjima i za neviđene TSP probleme.

Analiza je pokazala da optimizacija unutar latentnog prostora, osposobljenog da sadrži visokokvalitetna rješenja, dovodi do brzog pronalaska kvalitetnih rješenja problema. Ovakav pristup omogućuje algoritmu da izbjegava manje optimalna rješenja te se fokusira na ona koja su u skladu s naučenim karakteristikama dobrih rješenja. Dodatno, u istraživačkom radu [3] napravljena je usporedba s tradicionalnim pristupima optimizacije u realnom prostoru te je potvrđena superiornost optimizacije u latentnom prostoru.

Iz eksperimenata se također može zaključiti da je korištenje Hooke-Jeeves postupka unaprijedilo performanse genetskog algoritma te da algoritam uspijeva pronaći bolje rješenje od običnog genetskog algoritma za isti broj evaluacija. Razlog tome može biti zbog mogućnosti finijeg podešavanja rješenja, smanjujući vjerojatnost zadržavanja na lokalnim optimumima i potičući bržu konvergenciju prema globalnom optimumu.

Unatoč navedenim prednostima, postoje i neki izazovi koji se javljaju prilikom korištenja opisanog pristupa optimizaciji koji ostavljaju mogućnost idućim istraživanjima. Jedno od ključnih ograničenja je inicijalna faza treniranja modela. Model korišten u ovom radu istreniran je nad 1000 instanci TSP problema od 100 gradova i kvalitetnih rješenja svakog od tih problema. Za optimizaciju većih TSP problema potrebno je skupiti veliki dataset primjeraka i rješenja većih problema kako bi model mogao naučiti novi latentni prostor. Iako je korišteni model istreniran nad TSP problemima od 100 gradova, može se koristiti i za veće i manje probleme, ali mu pada efikasnost. U budućim istraživanjima bilo bi zanimljivo istražiti poveznicu između razlike u veličini problema i efikasnosti pronalaska rješenja (koliko dobro algoritam pronalazi rješenje za probleme od 80 ili 120 gradova, ako pritom koristi model istreniran nad isključivo problemima od 100 gradova). Također, moguće je

istražiti mogućnost treniranja modela nad TSP problemima različitih dimenzija kako bi se postigla bolja generalizacija modela.

Druga mana opisanog postupka je činjenica da algoritam nije uspio pronaći optimalno rješenje u 500,000 evaluacija prilikom nekolicine pokretanja. Algoritam bi se mogao poboljšati ako se provede detaljnija optimizacije hiperparametara, ili se jednostavno poveća maksimalni broj evaluacija.

Dodatno, u budućnosti se može istražiti korištenje opisanog modela i algoritma optimizacije za druge optimizacijske probleme.

Uzimajući sve navedeno u obzir, ovaj rad uspješno demonstrira mogućnosti i prednosti integracije evolucijskih algoritama u latentne prostore modela dubokog učenja te postavlja temelje za daljnja istraživanja ovog područja. Daljnja istraživanja bi trebala težiti ne samo poboljšanju tehničkih aspekata algoritma, već i njegovoj primjenjivosti na širi spektar stvarnih problema koji zahtijevaju robustna i skalabilna rješenja.

Literatura

[1] Šnajder J., *1. Uvod u strojno učenje*, (Ak. God. 2023/2024), Poveznica: https://www.fer.unizg.hr/_download/repository/SU1-2023-P01-Uvod.pdf .

[2] Rocca J., *Understanding Variational Autoencoders (VAEs)*, Poveznica: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f7; pristupljeno 9. lipnja 2024.

[3] Hottung A., Bhandari B., Tierney K., *Learning a Latent Search Space for Routing Problems using Variational Autoencoders*, ICLR 2021 Conference, Beč, (2021, siječanj).

[4] Čupić M., *Prirodom inspirirani optimizacijski algoritmi*, Zagreb, (2009-2010).

[5] Albadr M., Tiun S., Ayob M., Al-Dhief F.T., *Genetic Algorithm Based on Natural Selection Theory for Optimization Problems*, (2020).

[6] Santos Amorim E.P., Xavier C.R., Campos R.S., Santos R.W., *Comparison between Genetic Algorithms and Differential Evolution for Solving the History Matching Problem*, (2012).

[7] *Genetski algoritmi – predavanje*, (2020, studeni), Poveznica: https://www.fer.unizg.hr/_download/repository/GApredavanje.pdf; pristupljeno 9. lipnja 2024.

[8] Đurasević M., Jakobović D., *Postupci numeričke optimizacije*, Poveznica: https://www.fer.unizg.hr/_download/repository/book%5B1%5D.pdf; pristupljeno 9. lipnja 2024.

[9] Dodlek Z., *Diferencijska evolucija*. Projekt. Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2008.

[10] Nazari M., Oroojlooy A., Snyder L., Taka´c. M., *Reinforcement learning for solving the vehicle routing problem*. In *Advances in Neural Information Processing Systems*, pp. 9839–9849, 2018.

[11] Ruprecht, Karls, *TSPLIB*, Universitat Heidelberg, Poveznica: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/; pristupljeno 9. lipnja 2024.

[12] Hegerty B., Hung C.-C., Kasprak K., *A Comparative Study on Differential Evolution and Genetic Algorithms for Some Combinatorial Problems*, Southern Polytechnic State University

[13] Kool W., van Hoof H., Welling M., *Attention, learn to solve routing problems!*, International Conference on Learning Representations, 2019.

Sažetak

Primjena evolucijskih algoritama u latentnom prostoru modela dubokog učenja

U ovom diplomskom radu istražena je primjena evolucijskih algoritama u latentnom prostoru modela varijacijskog autoenkodera (VAE). Za demonstraciju rada algoritma korišten je optimizacijski problem trgovačkog putnika (TSP). Implementacija rada temelji se na korištenju varijacijskog autoenkodera koji uči iz primjeraka i rješenja TSP problema te kreira latentni prostor koji se zatim koristi za efikasno pretraživanje rješenja. U sklopu rada napravljena je detaljna analiza hiperparametara koji utječu na performanse algoritma te usporedba različitih optimizacijskih algoritama unutar latentnog prostora. Analiza je pokazala da korištenje kombinacije genetskog algoritma i metode lokalne pretrage Hooke-Jeeves u latentnom prostoru postiže značajne rezultate prilikom pronalaska optimalnog rješenja za dani problem. Rezultati istraživanja ukazuju na prednosti i mane optimizacije u latentnom prostoru i potiču na daljnje istraživanje tog područja.

Ključne riječi: duboko učenje, varijacijski autoenkoder, latentni prostor, genetski algoritam, diferencijska evolucija, Hooke-Jeeves postupak, problem trgovačkog putnika

Summary

Evolutionary algorithm optimization in latent space of deep learning models

This thesis explores the application of evolutionary algorithms in the latent space of the variational autoencoder (VAE) model, specifically looking at solving the Traveling Salesman Problem (TSP). The implementation of the work is based on the use of a variational autoencoder that learns from examples and solutions of TSP problems and creates a latent space that is then used for an efficient search for solutions. As part of the work, a detailed analysis of the hyperparameters that influence the performance of the algorithm and a comparison of different optimization algorithms within the latent space was made. The analysis showed that the use of a combination of the genetic algorithm and the Hooke-Jeeves local search method in the latent space achieves significant results when finding the optimal solution for a given problem. The research results indicate the advantages and disadvantages of optimization in the latent space and encourage further research in this area.

Keywords: deep learning, variational autoencoder, latent space, genetic algorithm, differential evolution, Hooke-Jeeves method, traveling salesman problem