SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR 2

Optimizacija radnih nedjelja korištenjem genetskog algoritma

Stjepan Đelekovčan

Mentor: Domagoj Jakobović

Zagreb, Svibanj, 2025.

Sadržaj

1. Uvod

Problem radnih nedjelja novo je nastali problem u Republici Hrvatskoj od proglašenja zakona o izmjenama i dopuni zakona o trgovini 22. ožujka 2023. U članku 2 tog zakona pod točkom (4) piše:

„Trgovac može samostalno 16 nedjelja u godini odrediti kao radne, s time da se trajanju radnog vremena prodajnih objekata iz stavka 1. ovoga članka dodaje 15 sati koje raspoređuje od ponedjeljka do nedjelje.“

Navedeno trgovce dovodi do problema odabira 16 radnih nedjelja za svaku poslovnicu koju posjeduju. Trgovcima je, naravno, želja maksimizirati profit dok je građanima želja da i dalje sve ostane „kako je prije bilo“ i da svake nedjelje imaju neku trgovinu za posjetiti u slučaju potrebe za namirnicama i sl. Za primjer, prema nekim izvorima, lanac trgovina ŠPAR trenutno posjeduje 118 trgovina diljem hrvatske. To je 118 trgovina za koje je potrebno odabrati 16 nedjelja u nekoj godini tijekom kojih će raditi. Problem je naravno vrlo zahtjevan, naime samo za jednu poslovnicu postoji $3 \cdot 10^{13}$ mogućih rasporeda i trgovci kao što je ŠPAR trenutno ne posjeduju neki alat koji rješava taj problem, već koristeći tek neke statističke metode pokušavaju pogoditi dobar raspored nedjelja.

U ovom radu pokušati ćemo riješiti navedeni problem uz kriterij dobre pokrivenosti građana kako bi i građani bili zadovoljni rasporedom nedjelja. S obzirom da vjerojatno ne postoje primjerene metode koje rješavaju ovaj problem, problemu ćemo pristupiti iz perspektive genetskog programiranja koji bi uz prikladne heuristike trebao doći do zadovoljavajućih rješenja.

2. Problem radnih nedjelja

2.1 Model problema

Problem se svodi na odabir 16 radnih nedjelja (od njih 55 godišnje) tijekom kojih će neka poslovnica raditi. U obzir za optimizaciju uzimati ćemo geografsku lokaciju poslovnice, njen utjecaj (veličinu) na lokalnom području te projecirani profit za sve nedjelje tijekom godine.
Optimirati će se dva kriterija:

1. Ravnopravna maksimizacija zarade tijekom godine
2. Dobra pokrivenost područja na kojem trgovina djeluje tijekom godine

Više detalja o kriterijima kasnije.

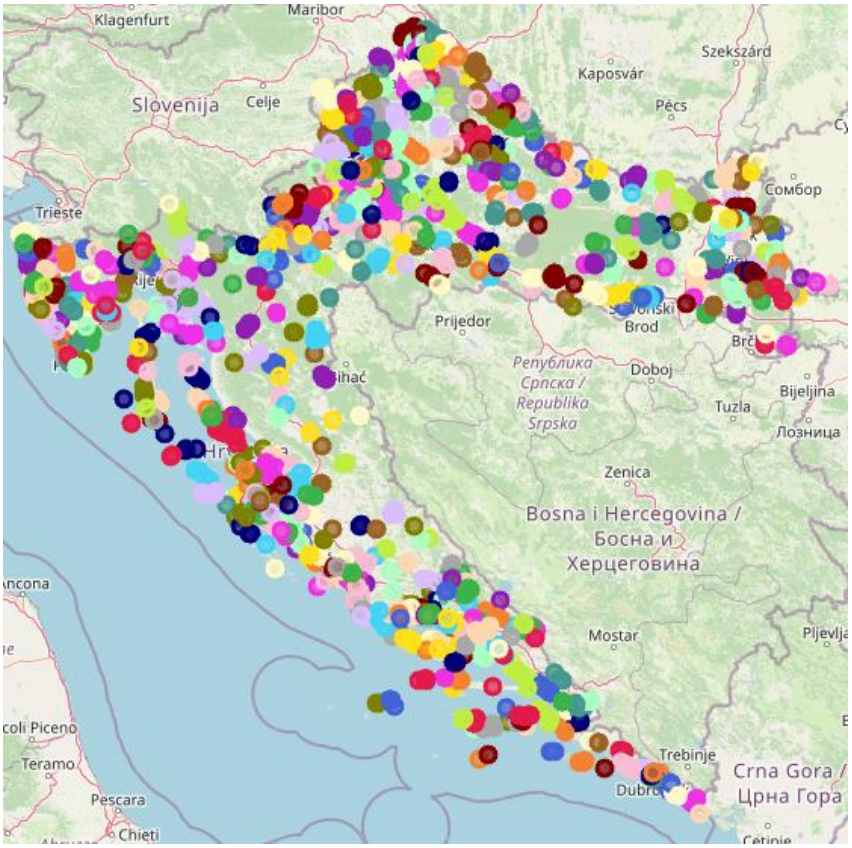
U stvarnosti, neke trgovine sigurno žele raditi specifične nedjelje u godini dok neke nedjelje sigurno ne žele raditi. Na primjer, trgovine na obali moraju raditi čim više tijekom ljeta zbog navale turista koji donose velike profite dok ne žele svoje radnike opterećivati za blagdane i praznike kao što su Božić, Uskrs i praznik rada. Ova ograničenja moći će se definirati i primjeniti za svaku poslovnicu te će algoritam biti prisiljen poštivati ih.

2.2 Podaci

Idealno bi bilo od samih trgovaca prikupiti podatke o geografskim lokacijama njihovih trgovina i stvarne podatke o prometu i zaradi tih poslovnica, no naravno ti podaci nisu, niti bi trebali biti, javno dostupni.

U svrhu testiranja pristupa i algoritma ipak je trebalo prikupiti neke podatke o trgovinama u Hrvatskoj. Prvo su prikupljene lokacije trgovine koristeći OpenStreetMap projekt iz kojeg je moguće povući lokacije svih trgovina koje su korisnici nekad unjeli u sustav (Slika 2). Uz koordinate i adresu trgovine prikupljen je i naziv trgovine kao i lanac kojem trgovina pripada za trgovine za koje je lanac definiran. Podatkovna struktura za jednu trgovinu prikazana je na Slici 2.

S obzirom na geografske podatke iz OpenStreetMap-a, korišten je Google Places API za prikupljanje dodatnih informacija o trgovinama kao što su broj recenzija na Googlu i ocjena trgovine. Ti podaci koristili su se za modeliranje utjecaja trgovine na svom lokalnom području kako bi se taj isti utjecaj koristio u izračunu oba kriterija. Na temelju broja recenzija i ocjeni generirani su mock-podaci za očekivani promet za pojedinu trgovinu.



Slika 1 Karta 4177 prikupljenih trgovina u Hrvatskoj grupiranih u grupe od 10


```
"way/29274000": {
    "name": "SPAR supermarket Zagreb Zagrebačka",
    "brand": "Spar",
    "rating": 4,
    "user_ratings_total": 1213,
    "formatted_address": "Zagrebačka cesta 3, 10000, Zagreb, Croatia",
    "coordinates": [
        15.9322674,
        45.8137023
    ]
},
```


Slika 2 Prikaz podatkovne strukture prikupljenih podataka

3. Genetski algoritam

S obzirom na dimenziju rješenja koja je naivno točno definirana sljedećom formulom

$$veličina = ukupan\ broj\ nedjelja * ukupan\ broj\ trgovina$$

jasno je da je teško pronaći optimizacijski algoritam koji će egzaktno optimirati bilo kakvu dobrotu. Genetski algoritam prihvatljiva je metoda za rješavanje velikih problema, pogotovo ako se dimenzija problema donekle može smanjiti, no genetski algoritam nikako ne može garantirati kvalitetu dobivenog rješenja. Drugi mogući pristup ovom problemu moglo bi biti linearno programiranje, no za to je potrebno dobro modelirati funkciju dobrote što je izvan okvira ovog seminara.

3.1 Konstrukcijski pristup

Kako bi se bolje nosili s dimenzijom problema, problem ćemo rastaviti na manje probleme koje je moguće paralelno optimirati. Dobivena optimalna rješenja dalje će se spajati prema nekom kriteriju i optimirati zajedno sve dok algoritam ne proizvede optimalan raspored za cijeli skup koji je dobio na ulazu. Naveden rastav na manje probleme radimo na sljedeći način:

1. Definiramo hiperparametar MAX_RADIUS_OF_INFLUENCE kao najveći radius na kojem neka trgovina može imati utjecaj
2. Trgovine grupiramo (eng. *cluster*) nekim jednostavnim algoritmom sa ograničenjem da za svaku trgovinu u grupi postoji barem jedna druga trgovina od koje je ta trgovina udaljena najviše MAX_RADIUS_OF_INFLUENCE kilometara i ograničenje da se u svakoj grupi nalazi najviše MAX_IN_CLUSTER trgovina.
3. Podgrupe optimiramo genetskim algoritmom i za njih dobijemo rješenja
4. Prethodne grupe i njihova rješenja spajamo u veće grupe tako da i dalje poštujemo prvo ograničenje o udaljenosti trgovina.
5. Nastalu veću grupu optimiramo istim genetskim algoritmom, ali s populacijom inicijaliziranom jedinkama koje su nastale unijom optimalnih rješenja podgrupa.

3.2 Model jedinke

S obzirom na veličinu problema, jedinku treba modelirati uzimajući u obzir brzinu genetskih operatora i, najbitnije, brzinu funkcije dobrote zanemarujući redundancije koje možda uzrokuju povećanje memorijskog zauzeća. Pa tako svaka jedinka za sebe prati sljedeća svojstva:

- cluster – nepromjenjiva lista identifikacijskih oznaka trgovina koje ta jedinka predstavlja
- constraints – pokazivač na objekt u kojem su definirana sva ograničenja za rad trgovina zadana od strane korisnika
- data – pokazivač na objekt u kojem se nalaze svi potrebni podaci o trgovinama potrebni za optimizaciju

- works – nepromjenjiva lista koja se sastoji od listi works[**i**] u kojima se nalaze indeksi nedjelja koje **i**-ta mora raditi
- model – promjenjiva lista koja se sastoji od listi model[**i**] u kojima se nalaze indeksi nedjelja koje **i**-ta trgovina radi u ovom rješenju
- antimodel – promjenjiva lista koja se sastoji od listi antimodel[**i**] u kojima se nalaze indeksi nedjelja koje **i**-ta trgovina može raditi u ovom rješenju, ali ne radi
- big_matrix – promjenjiva matrica koja predstavlja matričnu sliku listi works i model, a koja na poziciji (**i**, **s**) gdje je **i** trgovina, a **s** neka nedjelja ima 0 ako se **s** ne nalazi niti u model[**i**] niti u works[**i**], a količinu recenzija trgovine inače. Matrica služi za brzi izračun funkcije dobrote i potrebno ju je održavati u konzistentnom stanju kroz primjene genetskih operatora.

3.3 Funkcija dobrote

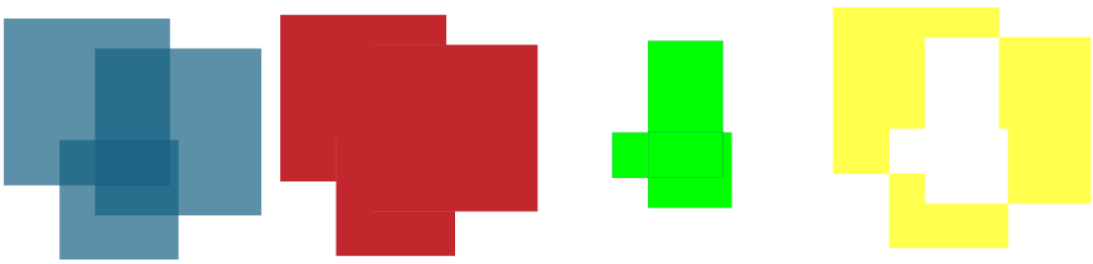
Prisjetimo se, želja nam je maksimizirati pokrivenost područja s trgovinama koje rade za svaku nedjelju, a u isto vrijeme minimizirati natjecanje trgovina kako bi svaka trgovina na kraju imala maksimalan profit s obzirom na utjecaj drugih trgovina. U te svrhe veličinu trgovine modelirati ćemo podacima o količini recenzija te trgovine koje smo dohvatili koristeći Google Places API, nazovimo tu mjeru *influence*(A) za trgovinu A. U svrhu jednostavnosti, za sada nećemo koristiti predviđeni prihod trgovine zato što stvarne podatke za to nemamo.

Modeliramo radius utjecaja trgovine s obzirom na druge trgovine na području na sljedeći način:

$$radius(A) = \sqrt{\frac{influence(A)}{\sum_{B}^{Trgovine\ u\ istoj\ grupi\ kao\ A} influence(B)}} \cdot MAX\_RADIUS\_OF\_INFLUENCE$$

Ovakav model radiusa ima dobra svojstva da će trgovina, ako jedina radi na nekom području, dostići utjecaj od točno MAX_RADIUS_OF_INFLUENCE kilometara, dok će njezin utjecaj približno kvadratno padati s prisutnosti drugih trgovina na području proporcionalno omjeru njenog utjecaja s utjecajima ostalih trgovina na području.

Površinu utjecaja modelirati ćemo kvadratima sa stranicom veličine dva radiusa koja predstavlja bounding-box utjecaja trgovine. Kvadrate koristimo umjesto krugova zato što je izračun površina i presjeka kvadrata mnogo brži od računanja istog za krugove.



Slika 3 Oblici koji se koriste u izračunu funkcije dobrote

Za dane utjecaje trgovina koji se međusobno preklapaju (plavo), izračunati ćemo njihovu ukupno površinu (crveno) i površinu njihovog presijeka (zeleno). Oduzimanjem površine presijeka od unije dobiti ćemo površinu čiji svaki dio pripada točno jednoj trgovini (žuto). Ovo ima smisla zato što ukupna unija (crveno) predstavlja pokrivenost područja dok unija presijeka (crveno) predstavlja međusoban (negativni) utjecaj trgovina. Ako maksimiziramo njihovu razliku (žuto) dobiti ćemo točno ono što želimo, maksimizaciju pokrivenosti područja i minimizaciju interakcije trgovina. Na kraju, računamo prosijek razlike (žuto) za sve nedjelje za danu grupu kako bi dobili normalizirani rezultat.

Programski, prvo se za svaku trgovinu u grupi brzo izračunaju radiusi utjecaja koristeći big_matrix i numpy dok se u svrhu izračuna površina svaki puta na temelju kvadrata gradi segmentno stablo pomoću kojeg se veoma efikasno računaju tražene površine.

3.4 Genetski operatori

Inicijalizacija nasumične jedinke

U svrhu demonstracije, definirajmo jedinku za 3 trgovine (Zero, One, Two) i za 5 nedjelja (indeksi nedjelja su [0, 4]) od kojih svaka trgovina mora raditi tri nedjelje. Trgovina Zero mora raditi nedjelje 2 i 3 te ne smije raditi nedjelje 1, Trgovina One ne mora raditi niti jedne nedjelje i smije raditi sve nedjelje, nedjelja Two mora raditi nedjelje 0 i 1 i ne smije raditi nedjelje 4.

Varijable su onda kako slijedi:

works = { [2,3], [ ], [0,1] }
model = { [ ], [ ], [ ] }
antimodel = { [0, 4], [0, 1, 2, 3, 4], [2, 3] }

Kod inicijalizacije jedinke (punjenje liste model), koristi se heuristika da svaka nedjelja u modelu bude nasumično zastupljena barem jednom, ako je to moguće. Ako to nije moguće, model se puni do potrebnog kapaciteta (works[i] + model[i] = broj radnih nedjelja) uzimajući nasumične nedjelje iz antimodela. Provođenjem navedenog postupka dobijamo inicijaliziranu jedinku.

works = { [2,3], [ ], [0,1] }
model = { [**0**], [**1**,**3**,**4**], [**2**] }
antimodel = { [4], [0, 2], [3] }

Selekcija

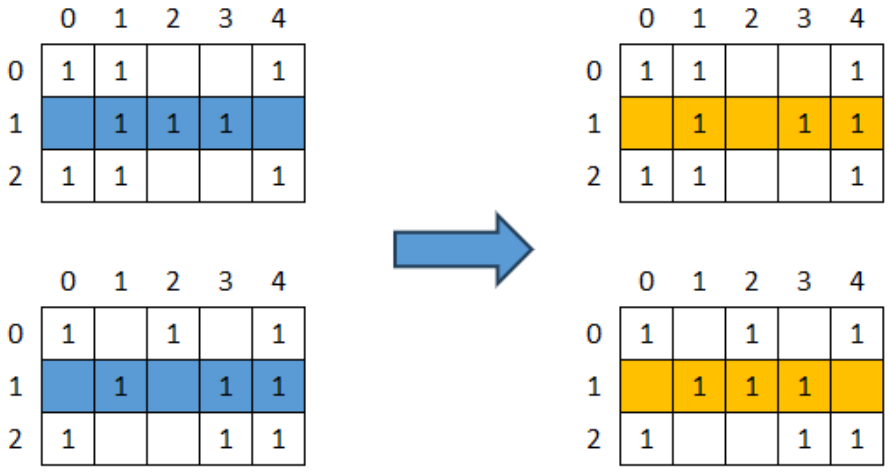
Koristi se jednostavna tro-turnirska selekcija za odabir dva roditelja. Križanjem nastaju dva nova djeteta koja sudjeluju u idućoj generaciji. U iduću generaciju također se prenosi i jedna najbolja jedinka iz prethodne generacije za svrhe elitizma

Križanje

U svrhu demonstracije, od sada nadalje za demonstraciju operatora koristiti će se matrični zapis u kojem retci predstavljaju trgovine, stupci nedjelje u godini pa će na mjestu u matrici biti jedinica ako trgovina radi te nedjelje, a nula inače. Svi operatori i

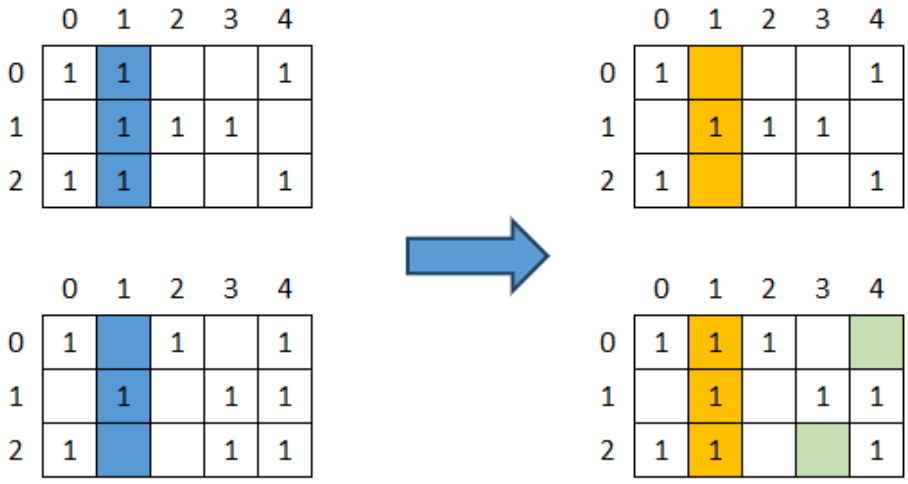
dalje su fizički implementirani koristeći model i antimodel liste zbog lakšeg očuvanja ograničenja, ali je matrični prikaz prikladniji za demonstraciju.

Križanje se radi na više načina. Između dva roditelja može se zamijeniti raspored nedjelja za pojedinu trgovinu (Slika 4) i to se može učiniti za k trgovina odjednom. Intuitivno, ovaj postupak može se zamisliti kao zamjena redaka. Zamjena redaka sama po sebi čuva sva ograničenja pa je ovaj način križanja vrlo jednostavno izvesti.



Slika 4 Operator križanja zamjenom redaka

Analogno, između dva roditelja mogu se i zamijeniti rasporedi za pojedine nedjelje (Slika 5), to se također može ponoviti i za k nedjelja odjednom. Intuitivno, ovaj postupak može se zamisliti kao zamjena stupaca. Međutim, kod zamjene stupaca može doći do kršenja ograničenja broja radnih nedjelja, pa je nakon ovakvog križanja jedinku potrebno „popraviti", micanjem nedjelja tamo gdje ih je previše i dodavanjem tamo gdje ih je premalo. Kršenje ograničenja o obavezi odnosno zabrani rada niti ovim se operatorom ne može napraviti.



Slika 5 Operator križanja zamjenom stupaca

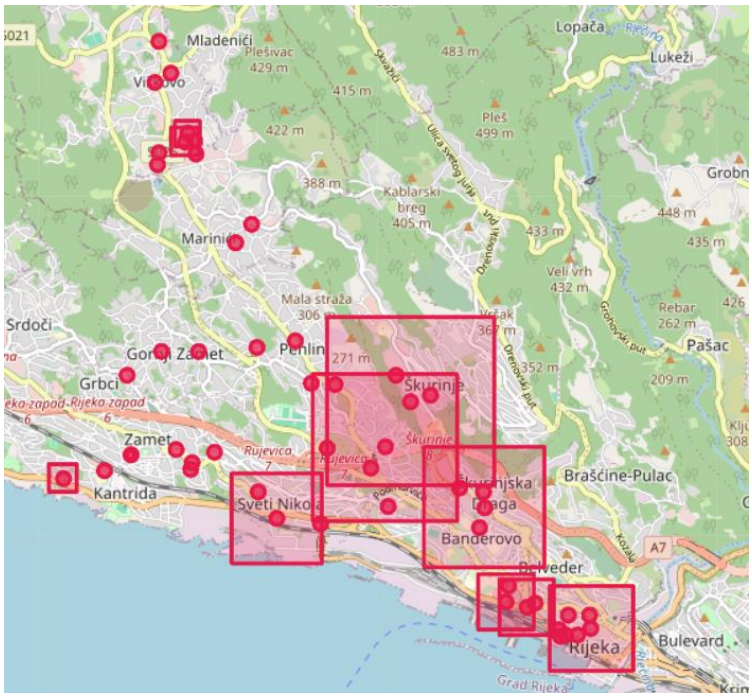
Parametar k koji govori koliko stupaca odnosno redaka se zamijenjuje generiramo prema geometrijskoj razdiobi. Križanje zamjenom stupaca može iskoristiti dobro raspoređene nedjelje iz dobrih rješenja dok koristan efekt križanja zamjenom redaka nije direktno vidljiv.

Mutacija

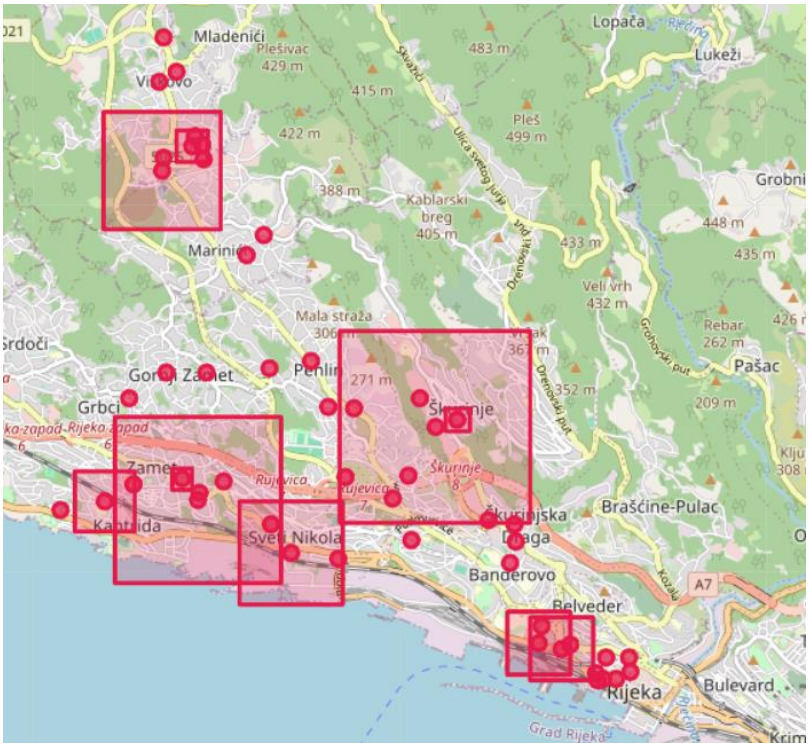
Operator mutacije sam je po sebi stohastički operator koji služi za istraživanje novih prostora rješenja, pa neka semantika iza efekta mutacije nije nužno potrebna. Mutacija je implementirana isključivo nad retcima na način da svaki redak mutira s nekom vjerojatnosti p, a unutar tog retka iz antimodela bira k slučajnih nedjelja i mijenja ih s k slučajnih nedjelja u modelu s ponavljanjem (Slika 6). Mutacija nad stupcima nije implementirana iz razloga što bi mutacije mogle narušavati ograničenja, što bi bilo preskupo popravljati s obzirom na prirodu operatora.

Slika 6 Operator mutacije izmjenom retka

4. Rezultati

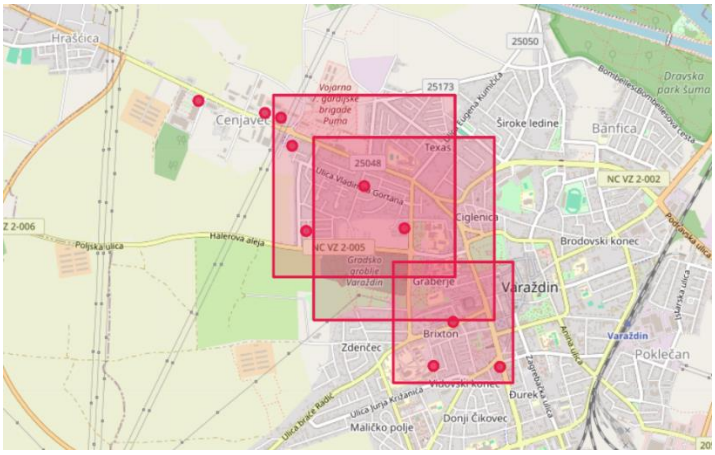


Slika 7 Rijeka – slučajno rješenje

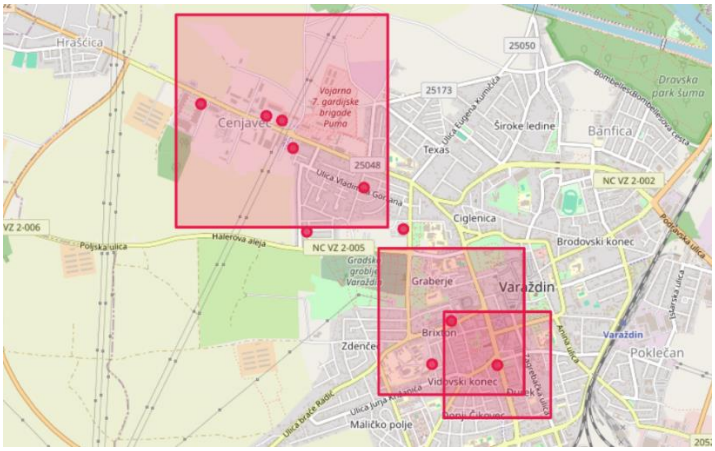


Slika 8 Rijeka – optimirano rješenje

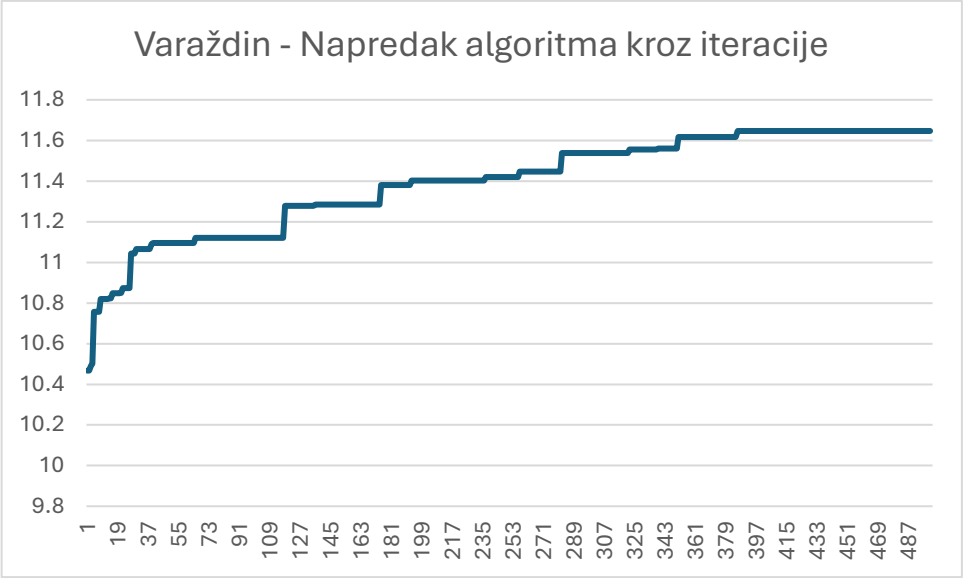
Na Slici 7 prikazano je slučajno rješenje manjeg problema koji se sastoji od ukupno 50 trgovina. Na Slici 8 prikazano je optimirano rješenje istog problema za istu nedjelju. Može se primijetiti kako optimirano rješenje zaista ima manju površinu presijeka te da sve trgovine zajedno pokrivaju veću površinu. Ovo je slučaj za većinu nedjelja u rješenju, no neke nedjelje ipak rezultiraju gorom raspodjelom u svrhu poboljšanja ostalih nedjelja.



Slika 9 Varaždin – slučajno rješenje



Slika 10 Varaždin – optimirano rješenje



Graf 1 Napredak algoritma kroz iteracije

Nadalje, testirano je i 500 generacija algoritma (Graf 1) nad skupom veličine jedne grupe kako bi se utvrdio napredak algoritma na potproblemu. Ponovno, na Slici 9 prikazano je nasumično rješenje problema za neku nedjelju dok je na Slici 10 prikazano optimirano rješenje istog problema za istu nedjelju.

Algoritam je postepeno konvergirao do poboljšanja od otprilike 10%. Poboljšanje od 10% je zapravo veliko s obzirom da se radi u prosijeku 55 nedjelja u godini, konkretno to je poboljšanje od 55 km^2 pokrivenosti kroz cijelu godinu za podskup od samo 10 trgovina.

5. Zaključak

Optimizacijom predložene funkcije dobrote dobili smo vidljivo poboljšanje u pokrivenosti područja nedjeljama kroz godinu. Naivni pristup ispao je uspješan iako su operatori prilično jednostavni. Problem s ovim pristupom je što je funkcija dobrote definirana heuristički sa dobrim svojstvima brze evaluacije. Za stvarni dokaz da je algoritam uspješan trebalo bi funkciju dobrote modelirati uz pomoć ekonomskog i trgovačkog eksperta koji bi dao bolji uvid u bitna svojstva rješenja i tek tada interpretirati uspjeh algoritma.

Postoji mnogo mogućih poboljšanja algoritma koja nisu implementirana niti razmatrana u ovom radu, a voljeli bi ih implementirati nekom drugom prigodom:

1. Implementirati paralelno računanje optimizacije pod-grupa
2. Bolji model za funkciju dobrote
3. Kod odabira operatora koji se primjenjuju uzimati u obzir njihov uspjeh i time češće koristiti uspješnije operatore
4. Smanjiti zrnatost operatora tako da prepoznaju koje specifične nedjelje imaju dodatnog prostora za optimizaciju i njih ciljati
5. Brža funkcija dobrote
6. Optimizacija hiperparametara
7. Bolje programersko i korisničko sučelje za korištenje algoritma

6. Literatura

[1] Čupić, M. Doktorski rad: Raspoređivanje nastavnih aktivnosti evolucijskim računanjem, 20.4.2025. https://ferko.fer.hr/people/Marko.Cupic/files/2011-Cupic-DoktorskiRad.pdf, 13.7.2011.

[2] Čupić, M. Prirodom inspirirani optimizacijski algoritmi, 20.4.2025. https://www.fer.unizg.hr/_download/repository/Cupic2009-PrirodomInspiriraniOptimizacijskiAlgoritmi.pdf, 2009.-2010.

[3] Konishi. H, Concentration of Competing Retail Stores, 11. 2005. https://www.researchgate.net/publication/222541128_Concentration_of_Competing_Retail_Stores, 25.4.2025.