SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1486

Optimizacija procjene susjedstva antena mobilne telekomunikacijske mreže

Marko Lukić

Zagreb, lipanj 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 3. ožujka 2017.

DIPLOMSKI ZADATAK br. 1486

Pristupnik: **Marko Lukić (0036469568)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Optimizacija procjene susjedstva antena mobilne telekomunikacijske mreže**

Opis zadatka:

Opisati problem određivanja najmanjeg gubitka signala usmjerenih antena u mobilnoj telekomunikacijskoj mreži. Provesti detaljnu analizu krajolika vezane funkcije cilja u svrhu pronalaska ovisnosti položaja minimuma o parametrima. Prilagoditi postojeće algoritme zadanom problemu te upotrijebiti prilagođene optimizacijske algoritme na osnovu rezultata analize. Ostvariti programski sustav za gradnju modela usmjerenih antena te modela predviđanja pozicije odnosno optimalnog iznosa funkcije cilja. Ocijeniti učinkovitost predikcijskih modela i ovisnost o hiperparametrima postupaka učenja. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 29. lipnja 2017.

Mentor:

Izv. prof. dr. sc. Domagoj Jakobović

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Siniša Srbljić

Zahvaljujem obitelji i prijateljima na podršci i motivaciji te mentoru prof. dr. sc. Domagoju Jakobovića na vođenju i pomoći tokom rada.

SADRŽAJ

1. Uvod

U ovom radu opisuje se proces optimizacije konkretnog problema iz stvarnog svijeta.

Prvi dio govori o samom problemu počevši s tim odakle potiče i zašto je njegovo rješenje potrebno. Zatim se prikazuje modeliranje problema funkcijom čiji se minimum traži što ujedno predstavlja i probleme kod rješavanja. Navode se i ograničenja koja bi konačno rješenje trebalo zadovoljavati te kompromisi do kojih ona dovode.

Drugi dio prolazi kroz korake razvoja i ispitivanja mogućih rješenja problema. Pristupi rješavanju grupirani su prema načinu računanja rezultata te se za svaku grupu navode postupci ispitivanja, ocjenjivanja te analize dobivenih rezultata. Svaki je od pristupa posebno objašnjen, prikazani su njegovi rezultati ispitivanja i raspravljaju se mogući razlozi dobre ili loše uspješnosti.

U trećem dijelu se skupljaju rezultati ispitivanja svih pristupa te se unutar grupa međusobno uspoređuju. Usporedba se vrši s obzirom na jedan ili više odabranih kriterija, a u slučaju da kriteriji nisu proporcionalni grafički se prikazuju pareto frontama. Na posljetku, svi rezultati se raspravljaju i donosi se zaključak o sveukupnom radu.

2. Problem optimizacije procjene susjedstva antena mobilne telekomunikacijske mreže

2.1. Opis problema

Problem koji treba riješiti zadala je tvrtka HashCode. Radi se o mreži telekomunikacijskih antena koju treba posložiti u susjedstvo tako da svaka antena dobije svoju sortiranu listu njoj najbližih susjeda. Procjena bliskosti dvaju antena dobiva se promatranjem gubitka njihovih signala u 2D prostoru. Obje antene su tada određene koordinatama u tom prostoru te karakteristikama njihovog sklopovlja pomoću kojih se modelira gubitak. Time se za svaku točku u prostoru može izračunati iznos gubitka za obje antene što se može interpretirati kao superpozicija dvaju 3D modela gubitka. Za konačnu ocjenu bliskosti para antena, u svakoj točki uzima se samo veći od dvaju iznosa gubitaka te se prostor pretražuje kako bi se dobio najmanji takav iznos. Susjedi se sortiraju prema dobivenoj ocjeni od najmanje, a cijeli postupak se ponavlja za sve parove antena u mreži.

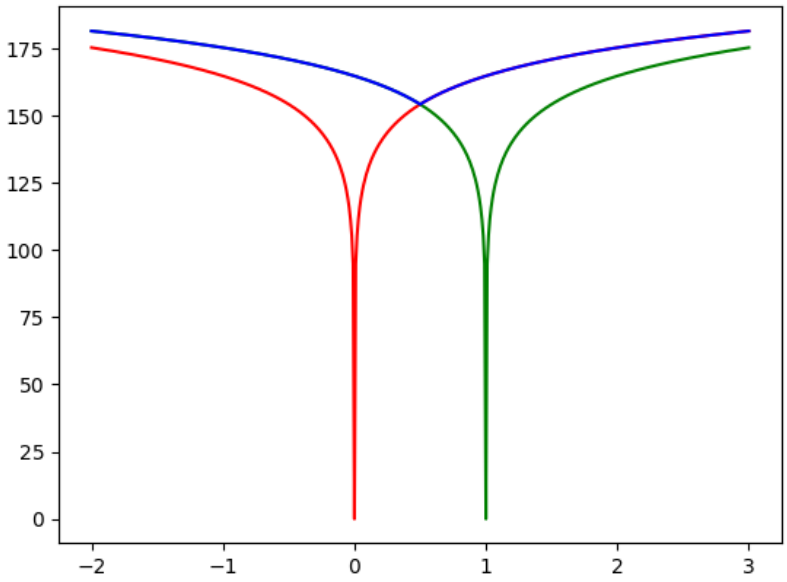
Broj antena u mreži je velik pa je tako i broj računanja ocjena bliskosti velik. Problem koji se rješava je što brže računanje tih ocjena, ali, bez značajnog gubitka preciznosti. Prvi korak u rješavanju već su napravili u HashCode-u: izradu modela gubitka antene u 2D prostoru i jednog mogućeg algoritma za pretragu prostora. Model koristi tri parametra antene za izračun gubitka:

- dobitak (eng. *gain*) - pojačanje, jačina signala
- širina snopa (eng. *beamwidth*)
- azimut (eng. *azimuth*) - relativna usmjerenost središta snopa

Prva dva parametra predstavljaju tvorničke karakteristike dok je treći bitan za određivanje međuodnosa dvaju antena. Radi uklanjanja dodatnih ulaznih parametara u pos-

tupku računanja ocjene bliskosti, antene se prije pretrage računanja gubitaka i pretrage prostora uvijek postavljaju na iste koordinate: (0, 0) i (1, 0). Tako dobivena ocjena se lako preračuna kako bi odrazila stvarnu udaljenost antena, a azimuti su neovisni o stvarnoj orijentaciji.

$$ocjena\_bliskosti = min(\forall x, \forall y | max(gubitak_1(x, y), gubitak_2(x, y)))$$



Slika 2.1: Maksimum gubitaka dvaju antena

Ono što je bilo potrebno napraviti je, koristeći ovaj model za generiranje funkcija pomoću parametara, isprobati alternativne algoritme za pretragu prostora ili zamijeniti cijeli postupak novim modelom. Takav model bi direktno iz parametara dvaju antena izračunao poziciju u kojoj se nalazi globalni minimum funkcije ili pak izračunao samu vrijednost funkcije u minimumu.

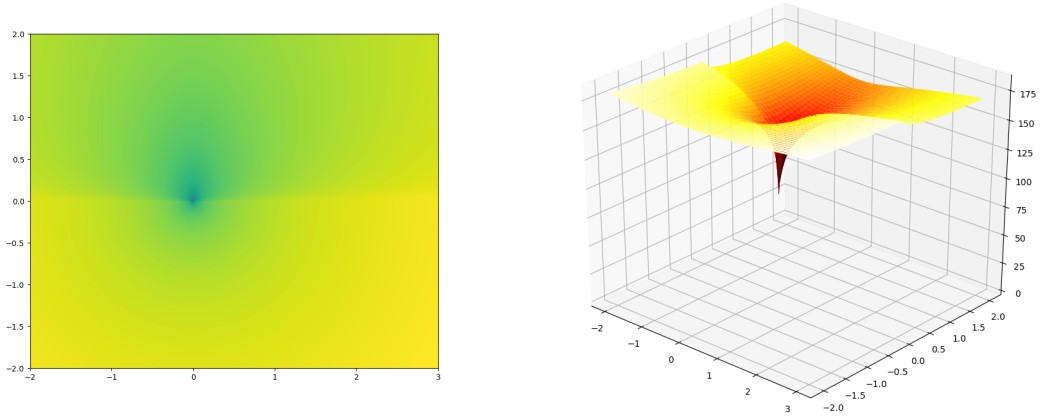
Glavni izazov u pronalasku prihvatljivog rješenja problema su uvjeti koje bi ono trebalo zadovoljavati. Najvažnije, ocjene bliskosti susjeda ne smiju biti netočne u toj mjeri da bi došlo do velikih promjena u mreži. To znači da rješenje mora biti precizno i pouzdano, vrijednosti koje se računaju moraju biti jednake za jednake parametre i dovoljno bliske ispravnim da se poredak u sortiranim listama ne promjeni značajno. Promjene poretka nisu značajne ako, na primjer, slični susjedi zamjene mjesta.

Drugi uvjet rješenju je ograničenje prosječnog vremena potrebnog za računanje. Ovo ograničenje nije strogo određeno, ali je dovoljno kako bi isključilo postupke koji prostor pretražuju iscrpno, koriste velik broj koraka i ponovljenih pokretanja ili koriste nerazumno složene procedure.

Konačno, rješenje mora davati zadovoljavajuće rezultate za sve ulazne parametre. U stvarnom svijetu kombinacije azimuta mogu biti proizvoljne, što znači da je potrebno ispitati što veći broj njihovih kombinacija i pritom osigurati da nema slučajeva koji iskaču. To znači da je, osim prosječne kvalitete, potrebno koristiti i najgore slučajeve kod usporedbi različitih pristupa.

2.2. Analiza problema

Za početak, bilo je potrebno vidjeti kako izgledaju modeli gubitka signala u 2D i 3D prostoru. Kako bi se to postiglo, za generiranje podataka koristila se preuzeta implementacija modela i *python* knjižica (eng. *library*) *matplotlib* za vizualizaciju. Slijedeće slike pokazuju gubitak jedne antene postavljane na koordinate (0, 0) s dobitkom 15, širinom snopa $45°$ i azimutom $0°$. Takav azimut gleda u smjeru *y*-osi, a stupnjevi se gledaju u smjeru kazaljke na satu, npr. s azimutom od $90°$ antena bi gledala u smjeru *x*-osi.



(a) 2D

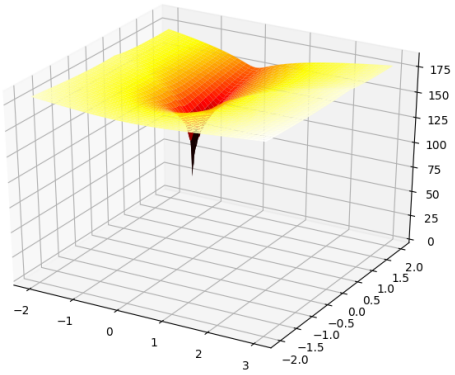
(b) 3D

Slika 2.2: 15_45_0

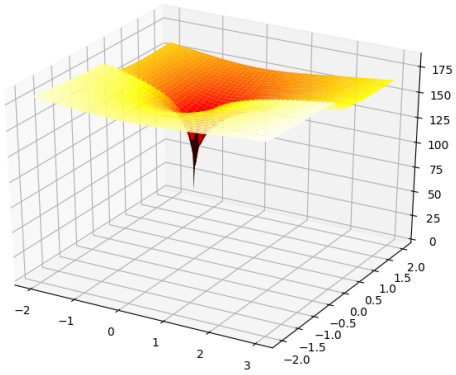
Na slikama se može vidjeti kako gubitak brzo raste udaljavanjem od antene, ali i da taj rast s udaljenošću usporava. Zbog toga možemo pretpostaviti da su vrijednosti gubitka u uskom području oko same antene uvijek niske i da će tu, dodavanjem druge antene, zbog funkcije maksimum taj gubitak biti nadjačan.

2.2.1. Utjecaj širine snopa

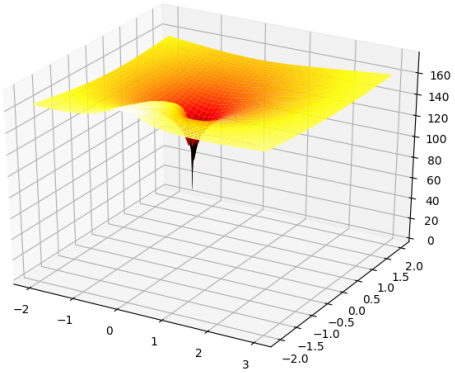
Na pozicijama jednako udaljenim od antene, gubitak je uvijek najmanji u smjeru azimuta koji predstavlja središte snopa. Udaljavanjem od tog središta po kružnici gubitak postepeno raste ovisno o širini snopa. Neke antene šire signal u svim smjerovima, tj. širina snopa im je $360°$ pa je gubitak jednak u svim smjerovima. Slijede primjeri s različitim širinama signala počevši od najmanje pa sve do najveće moguće.



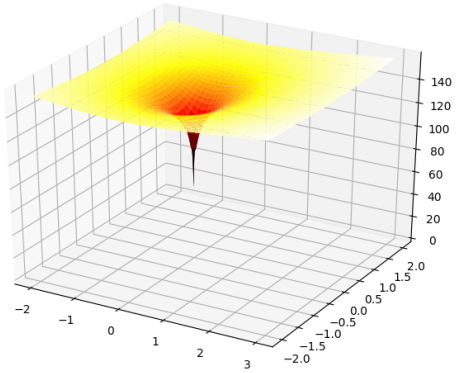
(a) $15°$



(b) $90°$



(c) $270°$

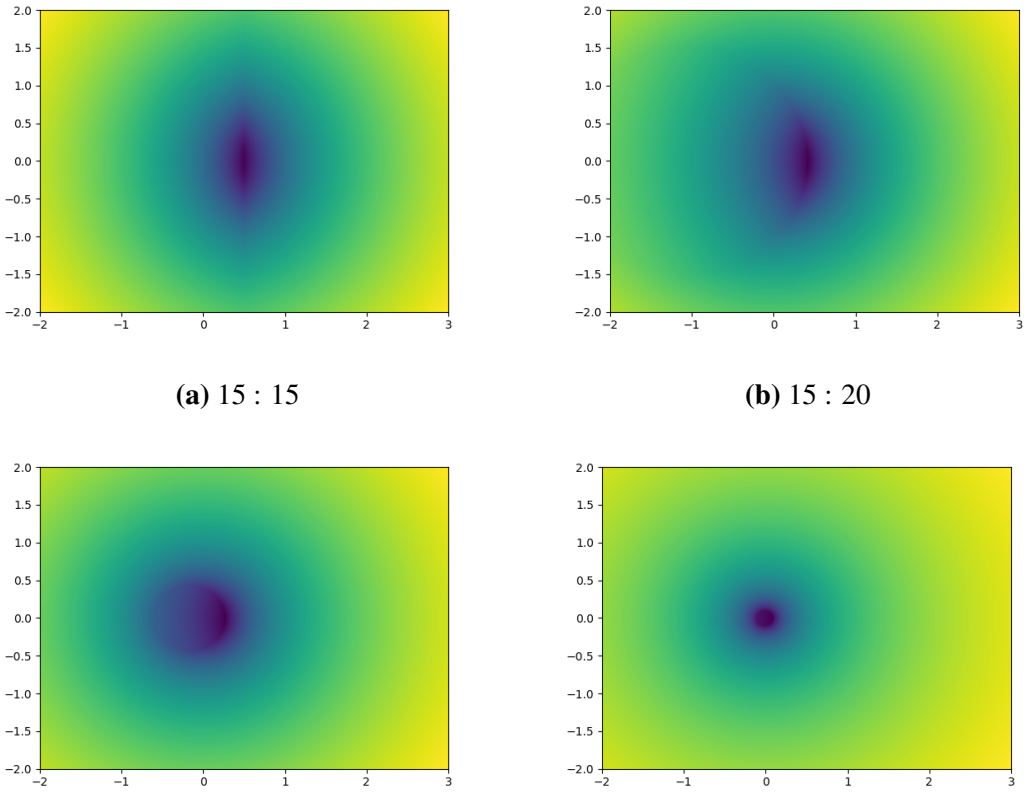


(d) $360°$

Slika 2.3: Različite širine snopova

2.2.2. Utjecaj dobitka

Dobitak predstavlja jačinu signala kojeg antena emitira pa zato ujedno i smanjuje gubitak u cijelom prostoru. U slučaju jedne antene to samo znači da će vrijednosti biti niže dok će 3D model će izgledati jednako. No, kad gledamo površinu funkcije nastale maksimumom gubitaka dvaju antena, razlika vrijednosti dobitka određuje položaj najniže točke. Naime, povećavanjem dobitka jedne od antena njen se gubitak spušta pa se tako i smanjuje njen utjecaj na izgled površine funkcije. U ekstremnim slučajevima, kad je razlika u dobicima velika, sve što preostaje je upravo gubitak antene s manjim dobitkom. Tada je i minimum blizu ili točno na poziciji te antene. Kako bi se ovaj utjecaj najbolje prikazao, u slijedećim primjerima obje antene imaju širinu snopa $360°$ tako da su svi parametri osim dobitka nebitni za poziciju minimuma.



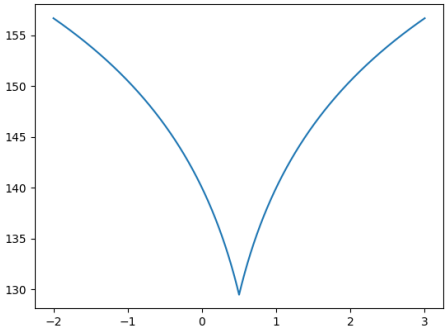
(a) 15 : 15 **(b)** 15 : 20

(c) 15 : 30 **(d)** 15 : 50

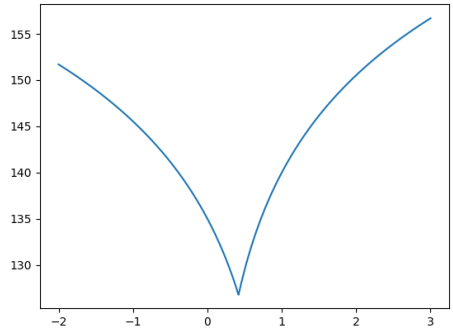
Slika 2.4: Različiti odnosi vrijednosti dobitka

Na slikama se vidi da se u slučaju jednakih vrijednosti dobitka minimum nalazi točno između dvije antene, na poziciji (0.5, 0.0), i povećavanjem dobitka druge antene primiče poziciji prve antene (0.0, 0.0).

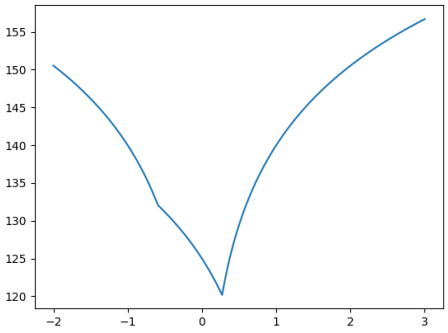
Slijedeće slike prikazuju vertikalne presjeke funkcije od točke (-2, 0) do (3, 0) za iste parametre antena:



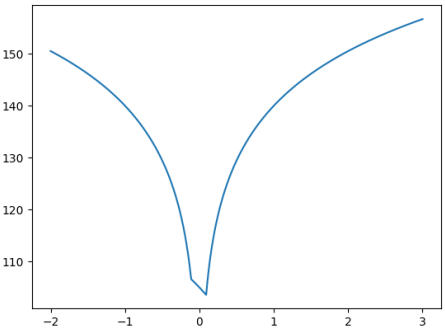
(a) 15 : 15



(b) 15 : 20



(c) 15 : 30

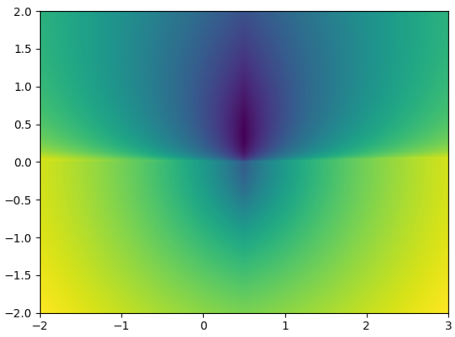


(d) 15 : 50

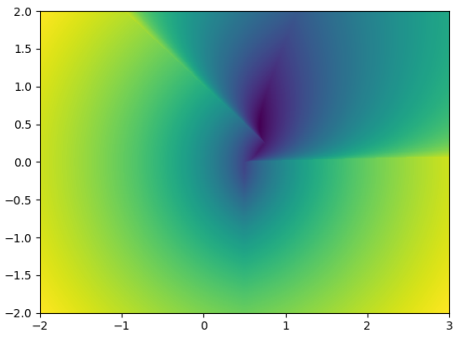
Slika 2.5: Različiti odnosi vrijednosti dobitka (presjeci)

2.2.3. Utjecaj azimuta

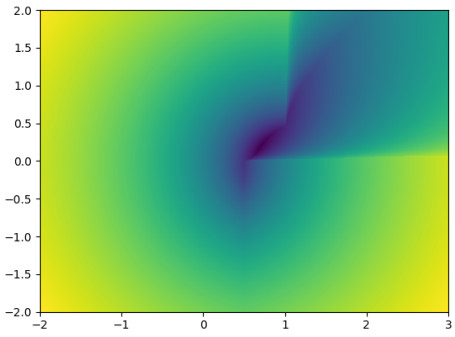
Srećom, sve vrijednosti za dobitak i širinu snopa koje se mogu pojaviti kao i kombinacije ta dva parametra su unaprijed zadane jer moraju podudarati tvorničkim karakteristikama antena. Na žalost, ono što najviše utječe na oblik površine funkcije je kombinacija azimuta. Iako se neke mogu zanemariti, npr. jedna od ovih: $(0° : 180°)$ ili $(180° : 0°)$, broj različitih kombinacija je i dalje jako velik. Slijedeći primjeri predstavljaju samo neke od njih fiksirajući parametre prve antene na dobitak: 15, širina snopa: $45°$ i azimut $0°$ te druge antene na identične uz pomicanje azimuta u koracima od $45°$ u smjeru kazaljke na satu.



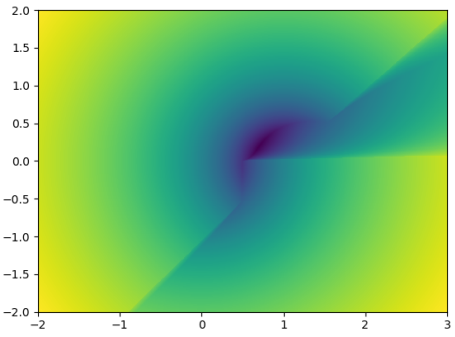
(a) $0° : 0°$



(b) $0° : 45°$

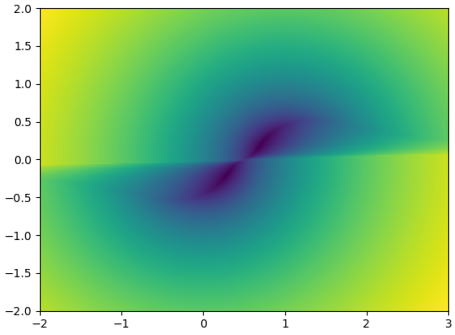


(c) $0° : 90°$

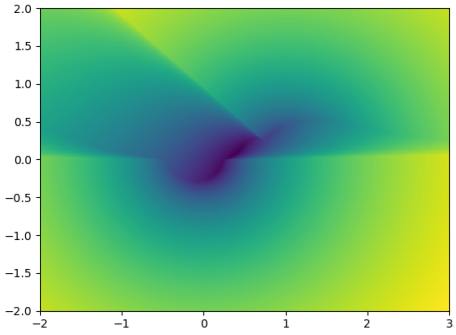


(d) $0° : 135°$

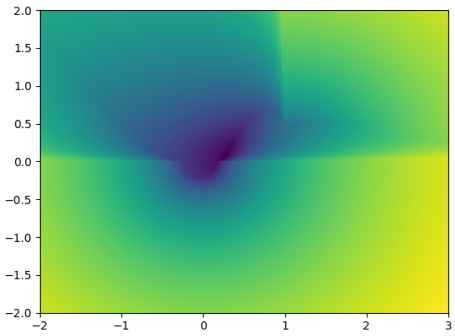
Slika 2.6: Različite kombinacije azimuta



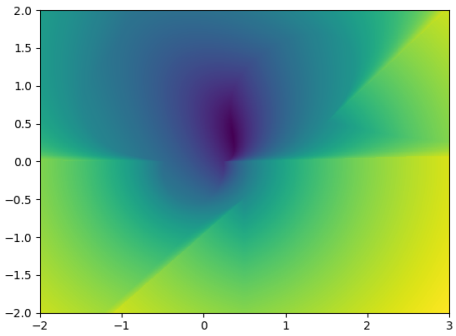
(e) $0° : 180°$



(f) $0° : 225°$



(g) $0° : 270°$



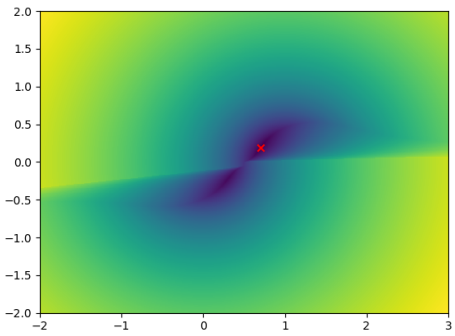
(h) $0° : 315°$

Slika 2.6: Različite kombinacije azimuta

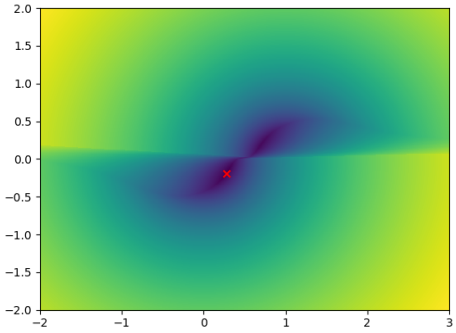
Zbog funkcije maksimum, površine funkcije su pune prijeloma koji nastaju u svim točkama u kojima su gubitci za obje antene jednaki. To stvara probleme za bilo kakve algoritme koji pretražuju prostor ili računaju tražene pozicije iz parametara jer odabir točke s krive strane takvog prijeloma može imati znatno veću vrijednost gubitka ili drugačiji gradijent koji pokazuje prema lošijem lokalnom minimumu.

2.2.4. Kretanje globalnog minimuma

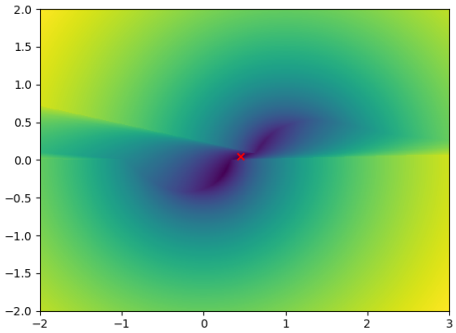
Dodatni problemi nastaju u slučajevima kad postoji više lokalnih minimuma različite kvalitete jer tada i manje promjene u azimutu neke od antena mogu poboljšati jedan i pogoršati drugi. Tada će algoritam pretrage i algoritam koji računa poziciju zbog sličnosti pretpostaviti da je globalni optimum blizu iako je zapravo u sasvim drugoj dolini. Jedan takav primjer, prikazan slikama koje slijede je par antena s uobičajenim parametrima: dobitak 15 i širina snopa $45°$, s prvim azimutom fiksiranim na $0°$ i drugim postavljenim na nekoliko različitih vrijednosti oko $180°$.



(a) $0° : 175°$



(b) $0° : 185°$



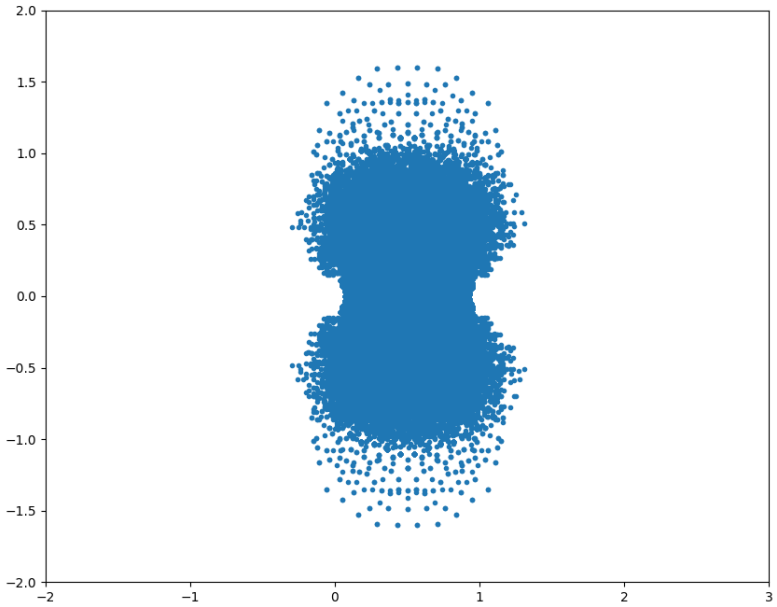
(c) $0° : 195°$

Slika 2.7: Promjene pozicije globalnog minimuma

Ovdje se, u samo dva koraka od $10°$, globalni minimum dvaput "premjesti". Štoviše, ono što dodatno otežava predviđanje ovakvog ponašanja je to što se čak niti ne kreće u istom smjeru. Povećanjem azimuta prvo se smanjuje dolina s prve slike i povećava ona u kojoj je globalni minimum u drugoj slici. Ujedno, prelaskom praga od $180°$ počinje nastajati i širiti se dolina u samom središtu. Kad azimut dovoljno naraste, ona postaje najdublja pa tako ponovno "seli" globalni optimum.

Unatoč ovakvom ponašanju, bilo bi relativno jednostavno napraviti model koji računa poziciju globalnog minimuma kad bi mogućnosti tu poziciju stale na prethodno prikazanima. No ovo je samo jedan slučaj, dodatno pojednostavljen time što obje antene imaju istu tvorničku konfiguraciju. Ono što se može napraviti je ispitati na kojem dijelu 2D prostora je uopće moguće da se globalni minimum pojavi. Svi primjeri do sada su pokazivali dio prostora oko pozicija antena, (0.0, 0.0) i (1.0, 0.0), ograničen na koordinate x-osi u rasponu od -2 do 3 i y-osi od -2 do 2. Da bi se što točnije mogle utvrditi granice unutar kojih je poželjno pretraživati, bilo je potrebno pokrenuti iscrpnu pretragu za što veći broj različitih parova i što veći broj kombinacija azimuta.

Slijedeći primjer prikazuje pozicije globalnog minimuma za preko milijun slučajeva koristeći rezoluciju azimuta od $5°$. Gledajući sve moguće tvorničke konfiguracije antena, velik broj različitih parova i dalje nije uključeno u ovaj primjer koji, bez promjene u rezoluciji azimuta, pokriva oko 0.06% mogućih slučajeva.



Slika 2.8: Pozicije globalnih minimuma za različite konfiguracije

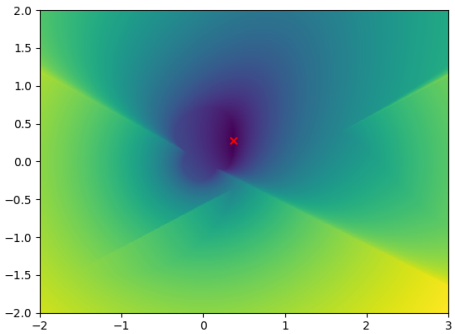
Ovi rezultati ograničili su područje zanimljivo pretragama na koordinate x-osi u rasponu od -0.3 do 1.3 i y-osi od -1.6 do 1.6.

2.2.5. Reduciranje kombinacija azimuta

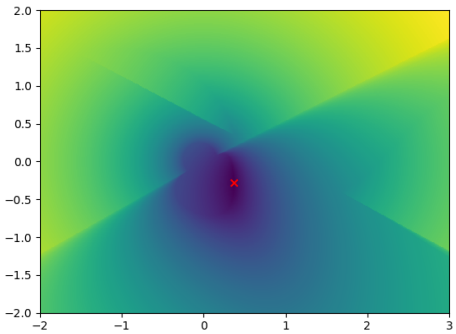
Osim prostora, moguće je i reducirati prethodno spomenute ekvivalente slučajeve gledajući kombinacije azimuta. U konačnom rezultatu nije bitna pozicija globalnog minimuma nego samo vrijednost funkcije u toj točki. Ako su azimuti postavljani na $(45° : 225°)$ ili $(135° : 315°)$, minimalna vrijednost funkcije će biti jednaka iako njena y-koordinata ima suprotan predznak. Zbog ove činjenice moguće je ukloniti sve slučajeve simetrične oko x-osi na način da se npr. azimut druge antene ograniči na vrijednosti između $0°$ i $90°$ te $270°$ i $360°$.

Posebno, ako antene imaju jednake tvorničke karakteristike, ekvivalentnim postaju i slučajevi simetrični oko osi paralelne s y-osi koja prolazi točno između dvaju antena, tj. pravca $x = 0.5$. Reduciranje ovakvih slučajeva se može postići time da opstaju samo koji zadovoljavaju izraz: $azimuth1 < 360° - azimuth2$.

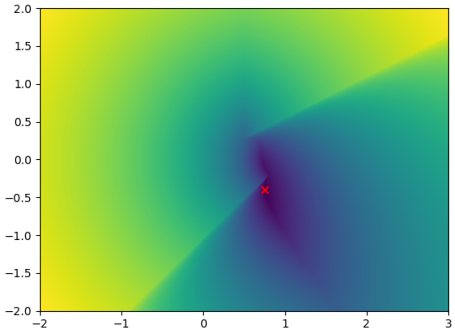
Slike redom pokazuju dva slučaja simetrična oko x-osi: (a) i (b) te dva simetrična oko pravca $x = 0.5$: (c) i (d).



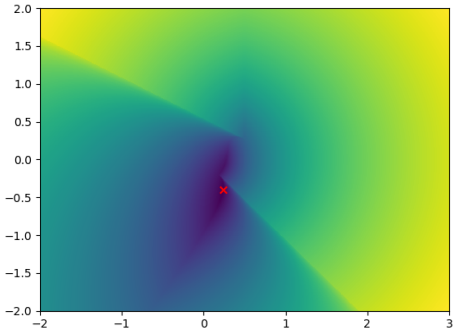
(a) 15 45° 30° : 20 90° 330°



(b) 15 45° 150° : 20 90° 210°



(c) 15 45° 150° : 15 45° 135°



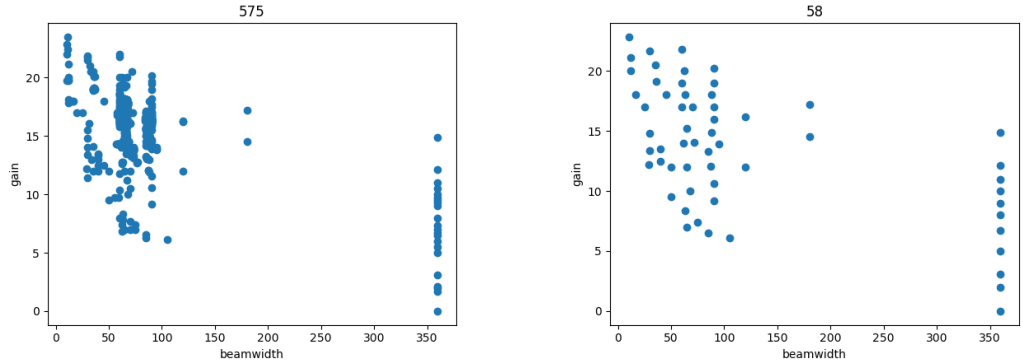
(d) 15 45° 225° : 15 45° 210°

Slika 2.9: Ekvivalenti slučajevi (dobitak širina azimut)

2.2.6. Reduciranje tipova antena

Nakon reduciranja kombinacija azimuta, ono što je preostalo reducirati su nepotrebni tipovi antena. Nepotrebni tipovi su oni koji imaju tvorničke karakteristike gotovo identične drugom tipu, osim, na primjer, male razlike u vrijednosti dobitka. To se može postići filtriranjem tipova po karakteristikama tako da se, nakon što se jedan tip prihvati, više ne prihvaćaju drugi kojima su i dobitak i širina pojasa unutar nekih granica sličnosti.

Sličnost nije dovoljno definirati jednostavnom euklidskom udaljenosti od vrijednosti karakteristika jer dobitak i širina pojasa imaju različite domene i gustoću. Zato je bilo potrebno za svaku od dvaju udaljenosti posebno odrediti najbolju moguću vrijednost takvu da se ukloni što više tipova antena bez gubitka bitnih primjeraka. Drugim riječima, ono što mora ostati nakon filtriranja su predstavnici cijelog skupa tipova antena.



(a) Izvorno **(b)** Preostalo nakon filtriranja

Slika 2.10: Prikaz reduciranja tipova antena

Odabrane vrijednosti za udaljenosti korištene u primjeru su 10° za širinu snopa i 1.0 za vrijednost dobitka. Time je preostalo oko 10% tipova, što znatno smanjuje broj mogućih parova antena koji o njima ovisi kvadratno.

3. Pristupi rješavanju

3.1. Algoritmi lokalne pretrage

Prvi pristup optimizaciji procjene susjedstva - deterministički algoritmi za pretragu prostora - logičan su izbor s obzirom na prirodu problema. Ovakvi algoritmi su stvoreni za brzu, učinkovitu i pouzdanu pretragu za pozicijom u kojoj leži najmanja vrijednost funkcije. Stohastički algoritmi poput simuliranog kaljenja ili genetskih algoritama ne osiguravaju da će kod svakog pokretanja za iste parametre antena davati isti rezultat. Dodatno, genetski algoritmi su bolji kod funkcija s više od jednog lokalnog optimuma upravo zbog korištenja djelomično nasumične pretrage te populaciju potencijalnih rješenja, ali time zahtjeva veći broj evaluacija vrijednosti funkcije.

S druge strane, nije prikladno niti korištenje algoritama koji se kreću samo pomoću gradijenta funkcije iako su vrlo brzi i učinkoviti. Osim toga što nije jednostavno dobiti gradijent funkcije koja modelira gubitak, funkcija dobivena maksimumom dvaju gubitaka nije derivabilna u točkama gdje su gubitci jednaki (prijelomima). Čak i kad bi se taj problem uklonio, gradijentni postupci uvijek pronalaze samo najbližu stacionarnu točku. To otvara nova pitanja poput: koliko puta ponoviti postupak, kako odabrati početne točke ili kako tokom izvođenja odrediti da li se postupak kreće prema prethodno pronađenoj stacionarnoj točki?

Za ocjenjivanje svih algoritama koristi se isti postupak kao i kod generiranja podataka za grafički prikaz funkcije: iscrpna pretraga po rešetci (eng. *Grid Search*). Algoritam uzorkuje 2D prostor u jednoliko razmaknutim točkama paralelnim s koordinatnim osima. Za grafički prikaz pamti sve izračunate vrijednosti funkcije, dok za pretragu koristi samo poziciju i iznos najmanje. Veličinu koraka kod uzorkovanja odredili su u HashCode-u i iznosi 0.01. Ta se veličina ujedno koristi kao prihvatljiva preciznost svih algoritama za pretragu.

3.1.1. *Moving Grid Search*

Prvi algoritam koji se koristi za pretragu prostora nastao je kao moguća metoda za reduciranje iscrpne pretrage po rešetci. Koristi rešetku proizvoljnih dimenzija, najčešće 3x3, koja se kreće ovisno o vrijednostima funkcije u svim njenim točkama. U svakom koraku pomiče se u smjeru točke s najmanjom vrijednosti dok god ta točka nije u središtu. Kad je "najniža" točka u središtu, rešetka se skuplja oko nje. Postupak se ponavlja do trenutka kad razmak između točaka rešetke odgovara zadanoj preciznosti. Ovakvim ponašanjem dobiva se početno široka i rijetka rešetka koja istražuje cijeli prostor, a kasnije se smanjuje pa time i postaje gušća. Takva može precizno pregledati dio prostora koji u prvim koracima određen kao potencijalno kvalitetan.

---

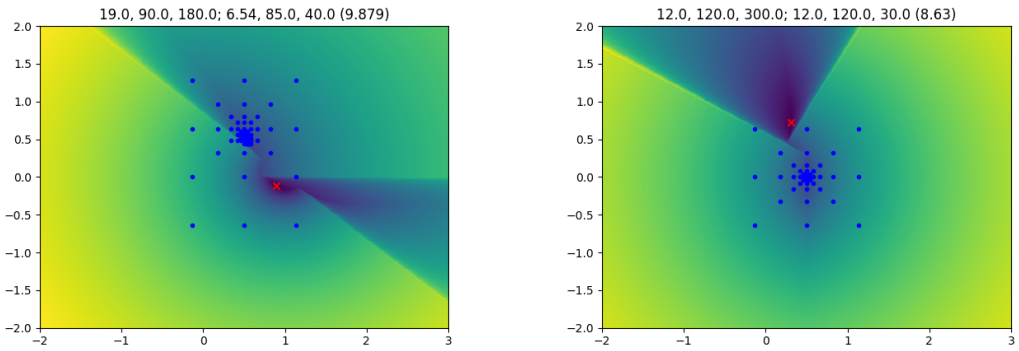
while $razmak > preciznost$ **do**
  $evaluiraj(resetka)$
  $najbolja\_tocka \leftarrow pronadi\_najbolju\_tocku(resetka)$
  if $najbolja\_tocka < sredisnja\_tocka$ **then**
    $pomakni(resetka, najbolja\_tocka)$
    if $nije\_rubna(najbolja\_tocka)$ **then**
      $prepolovi(resetka)$
    end if
  else
    $prepolovi(resetka)$
  end if
end while

U pseudokodu se vidi i dodatna heuristika koja skuplja rešetku čak i kad "najniža" točka nije u središtu, ali nije niti na rubu. U takvom slučaju nema puno smisla pomicati veliku rešetku jer su rubne točke lošije pa se može pretpostaviti da se u tom smjeru funkcija "uspinje".

Početan razmak je, kao i dimenzije rešetke, proizvoljan jer zajedno određuju početno ponašanje algoritma: koliko veliko područje gleda, koliko velike korake radi, koji je oblik rešetke i koliko je ona gusta. Poželjno je da rešetka u početku pokriva što veći dio prostora unutar kojeg je moguće da se nalazi globalni minimum. Iz slike 2.10 je poznato da su dimenzije tog prostora veće u smjeru *y*-osi nego *x*-osi pa je prikladna i rešetka takvog oblika. Što je rešetka gušća, to je veća šansa da će pronaći dolinu koja pripada upravo globalnom minimumu, ali u tom slučaju ona ostaje nepotrebno

gusta nakon smanjivanja pa tada samo usporava rad algoritma. Suprotno tome, rijetka rešetka umanjuje preciznost pa su rezultati takve konfiguracije slabiji u slučajevima u kojima su bitne doline relativno male.

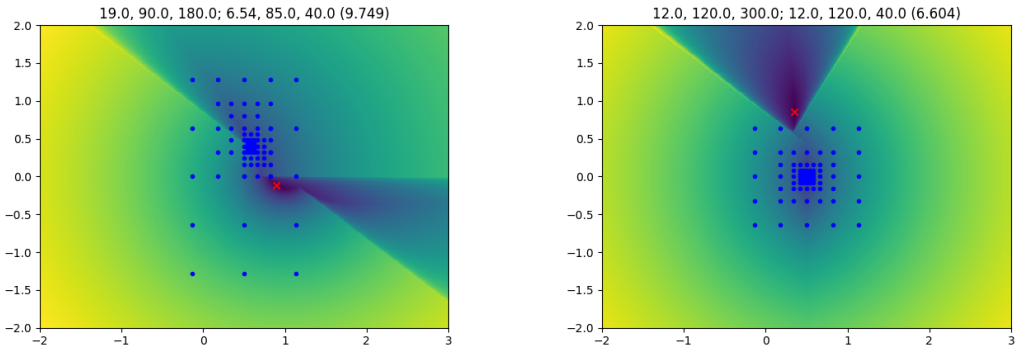
Slijedeći primjeri prikazuju proces prilagodbe parametara *Moving Grid Search* algoritma, brojevi iznad slika predstavljaju parametre antena (dobitak, širina snopa, azimut) te razlika između najmanje pronađene vrijednosti funkcije i najmanje moguće:



(a) **(b)**

Slika 3.1: Dimenzije: 3x3, razmak: 0.64

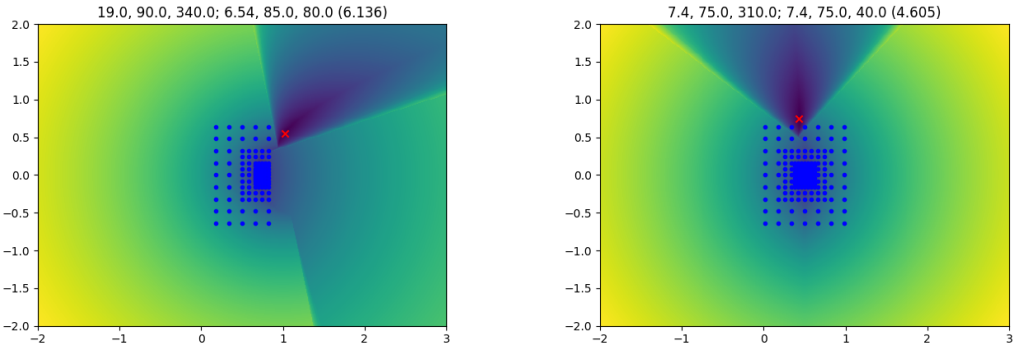
Na slikama 3.1 vide se dva loša slučaja za navedene parametre algoritma. Algoritam kreće iz sredine prostora (0.5, 0) i u prvom slučaju se kreće u smjeru y-osi, dok u drugom ostaje na početnoj poziciji. U oba slučaja rešetka nema ni prikladan oblik ni gustoću da bi pogodila dolinu globalnog minimuma pa završava u najboljem pronađenom lokalnom.



(a) Dimenzije: 5x3, razmak: 0.64 **(b)** Dimenzije: 5x5, razmak: 0.32

Slika 3.2: Prve preinake parametara

Slike 3.2 prikazuju promjenu oblika rešetke dodavanjem redova u smjeru y-osi i promjenu gustoće dodavanjem redaka u oba smjera i smanjivanjem razmaka. Prvi pristup rješava slučaj (b) sa slike 3.1, ali je jednako neuspješan u slučaju (a). Drugi pristup samo ublažava slučaj (b) smanjujući broj problematičnih parova azimuta.



(a) Dimenzije: 9x5, razmak: 0.16 **(b)** Dimenzije: 9x7, razmak: 0.16

Slika 3.3: Druge preinake parametara

Posljednje slike, 3.3, prikazuju dodatne dvije iteracije prilagodbe parametara algoritma. U prvoj se dodatno povećava gustoća rešetke, ali se smanjuje njena veličina u smjeru x-osi kako bi se izbjelo veliko povećanje broja točaka koje se evaluiraju. U drugoj se ispravlja ta pogreška jer je širina rešetka bila nedovoljna. Prvom iteracijom popravljeni su nedostaci u slučajevima poput (a) na slici 3.1, ali čak i nakon druge iteracije postoje slučajevi tipa (b) koji prolaze između uzorkovanih točaka prvog koraka algoritma.

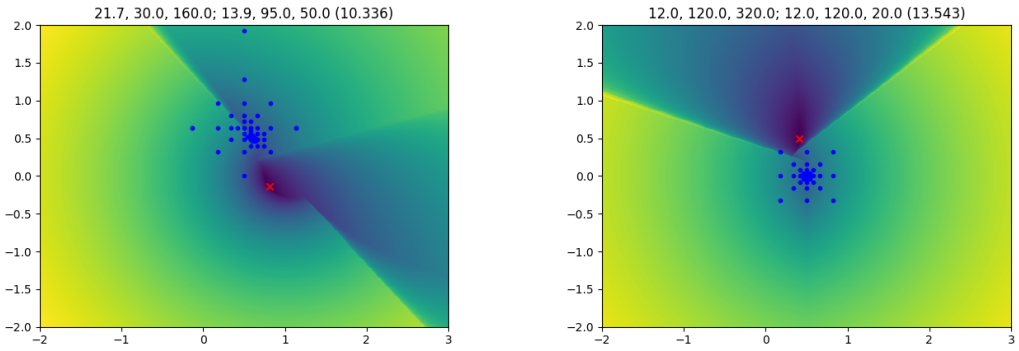
Ovakvi slučajevi bi se mogli dalje ispravljati dodavanjem još redova u smjeru y-osi, ali i trenutni parametri već predstavljaju probleme nakon jednog ili dva sužavanja rešetke. Prvi korak algoritma je već pomoću guste rešetke ustanovio da točke oko najbolje nisu zanimljive za dalja istraživanja pa više nije potrebno da koristi koliko puno redova/stupaca. Osim toga, prosječan broj evaluacija povećao se s reda nekoliko desetaka na nekoliko stotina. Algoritam tako gubi svoju početnu učinkovitost i degradira u iscrpnu pretragu koju pokušava reducirati.

Algoritam je dobar u slučaju nekoliko većih dolina nepravilnog oblika, ali zahtjeva izmjene suprotne njegovoj zamišljenoj ulozi kad su doline manje i strme. Kod funkcija ili područja gdje postoji samo jedan minimum, postoje drugi prikladniji algoritmi koji zahtijevaju primjetno manji broj evaluacija da bi postigli jednake rezultate.

3.1.2. Prošireni Hooke Jeeves algoritam

Za razliku od *Moving Grid Search* algoritma koji u slučaju 3x3 rešetke uvijek pretražuje sve smjerove paralelne i dijagonalne s koordinatnim osima, Hooke Jeeves algoritam pretpostavlja da se nalazi u području sa samo jednim minimumom, tj. jednom dolinom, pa koristi manji broj istraživačkih koraka. Naime, zbog te pretpostavke, ako u jednom smjeru vidi poboljšanje, niti ne provjerava suprotan nego se odmah pomiče u evaluiranu točku. Dodatno, kad u okolici nađe bolju poziciju, isproba i dodatan korak u istom smjeru. Zbog ovakvog ponašanja se prema minimumu kreće brže i uz manje evaluacija. Detaljniji opis algoritma s ilustracijama njegovog ponašanje može se naći u [3].

Nedostatak Hooke Jeeves algoritma je to što je još osjetljiviji na oblik funkcije nego *Moving Grid Search*. Kod neuspjelih istraživanja okoline u smjerovima paralelnim s koordinatnim osima niti ne pokušava ispitati pozicije u dijagonalnim smjerovima. Zbog toga se ovdje koristi proširena inačica algoritma koja dodaje upravo tu funkcionalnost. Nakon evaluacije početne i četiri lošije okolne pozicije, redom se evaluiraju i četiri dijagonalne pozicije. U najgorem slučaju algoritam se ponaša identično kao i *Moving Grid Search*, ali inače se evaluacije zaustavljaju čim se pronađe prva pozicija bolja od početne.



(a) Veličina koraka: 0.64

(b) Veličina koraka: 0.32

Slika 3.4: Loši slučajevi za Hooke Jeeves algoritam

Hooke Jeeves algoritam nema puno parametara, moguće je zadati veličinu početnog koraka i preciznost. Na slikama 3.4 prikazano je ponašanje uz dvije različite veličine koraka. U slučaju (a) algoritam postiže neuspjeh na sličan način kao i *Moving Grid Search* na slici 3.1 (a), ali uz manji broj evaluacija funkcije gubitka. Slučaj (b) izgleda gotovo isto kad kad bi se koristila rešetka dimenzija 3x3 s razmakom jednakim veličini koraka jer predstavlja najgori scenarij u kojem niti dijagonalne provjere

ne uspijevaju.

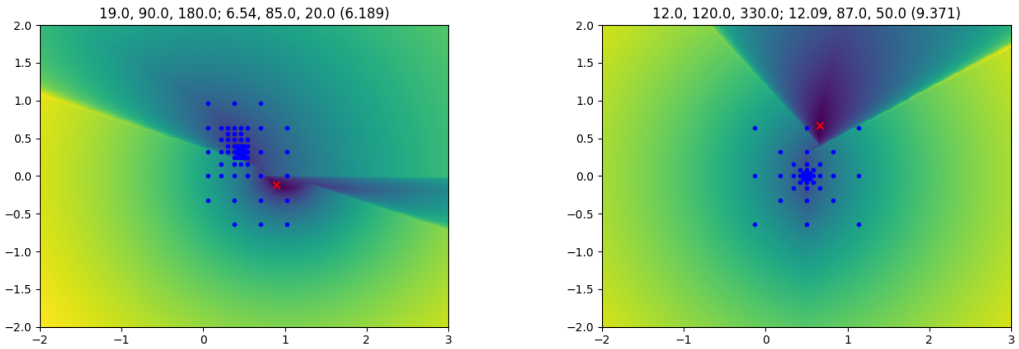
Dodatna karakteristika koja može utjecati na uspješnost algoritma je redoslijed kojim istražuje različite smjerove. Kako algoritam mora biti deterministički i pretpostavlja da je funkcija koju pretražuje unimodalna, taj redoslijed je najčešće fiksiran. Zbog toga će ovdje uvijek postojati neke konfiguracije parametara antena kod kojih će završavati u krivom minimumu kao u slučaju (a). Kad bi se prvo istraživali smjerovi paralelni x-osi, algoritam bi pronašao bolje pozicije koje bi ga odvele u globalni optimum. Ali, tada bi se pojavila nova konfiguracija u kojoj bi bilo bolje istraživati u smjeru paralelnom s y-osi.

Hooke Jeeves je učinkovit algoritam za pretragu jedne doline, a dodavanje smjerova za istraživanje to proširuje i na doline prethodno nepovoljnog oblika. No funkcije maksimuma gubitaka nisu takve da bi nagradile tu učinkovitost, nego uzrokuju pogoršanje kvalitete rezultata. Doline globalnih minimuma mogu pokrivati mali dio prostora koji se pretražuje i mogu se nalaziti bliže ili dalje početnoj točki pa se uvijek dogodi da, ovisno o veličini početnog koraka, algoritam tu dolinu ili preskoči ili ne dosegne. Zbog svega toga on nije prikladan za samostalno korištenje kao rješenje ovog problema.

3.1.3. Heuristike za odabir početne točke

Odabir početne točke za algoritme lokalne pretrage igra veliku ulogu u njihovoj uspješnosti kod funkcija s više lokalnih minimuma, a može i pomoći u smanjivanju potrebnog broja koraka kad postoji samo jedan. Pretpostavka da je dovoljno pronaći granice prostora koji se pretražuje i postaviti početnu točku u središte uz prikladne parametre algoritma pokazala se krivom. Uzrok tome su svojstva parametara antena i funkcije maksimum da stvore površinu funkcije vrlo nepravilnog oblika, sa strmim dolinama koje često prekrivaju relativno mali udio promatranog područja.

Prva jednostavna ideja je koristiti činjenicu poznatu iz analize (2.2.2) da je minimum bliži anteni s manjom vrijednosti dobitka i približiti početnu točku toj anteni. To se može postići korištenjem težinske sume koordinata dvaju antena u kojoj se težinski faktori računaju iz vrijednosti dobitka. Problem kod ovog je što za poziciju minimuma nisu bitne točne vrijednosti dobitka nego samo njihova razlika pa faktore nije moguće računati trivijalno $(a/(a+b))$. Na slici 2.10 vidi se da se vrijednosti kreću od 0 do skoro 25 pa bi ovakvim računom faktori za parove $(0:5)$ i $(15:20)$ bili vrlo različiti unatoč jednakoj razlici. Umjesto toga, za računanje faktora koristi se sigmoidalna funkcija čiji je ulaz upravo razlika skalirana dijeljenjem s 15. Funkcija osigurava da će težinski faktor uvijek biti između 0 i 1, a skaliranje da vrijednosti razlike na ulazu daju očekivane izlaze. Vrijednost kojom se dijeli razlika dobivena je empirijski koristeći poziciju minimuma kod parova antena sa širinom snopa od $360°$ i različitim razlikama vrijednosti dobitka.



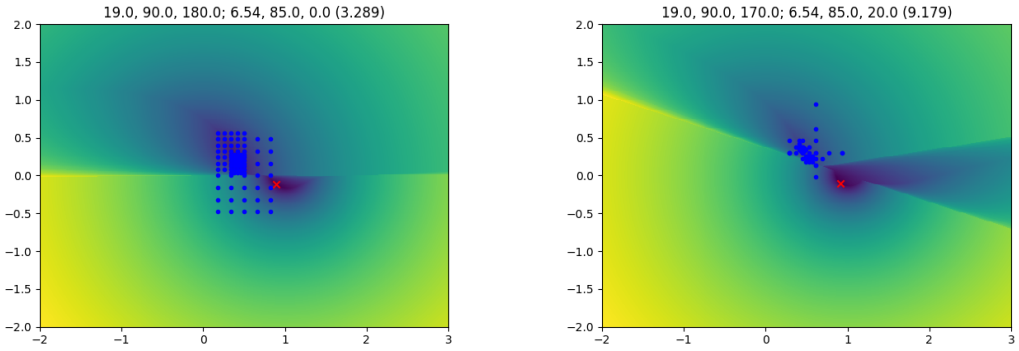
(a) *Moving Grid Search* **(b)** Hooke Jeeves

Slika 3.5: Određivanje početne točke korištenjem vrijednosti dobitka

Slike 3.5 pokazuju kako samo pomicanje na pravcu između dvije antene nije dovoljno. U slučaju (a), izračunata x-koordinata smanjuje udaljenost do globalnog minimuma, ali njegova dolina je uska i minimalno pomaknuta u smjeru y-osi što je dovoljno

da ju rešetka zaobiđe. Slučaj (b) pokazuje ispravno pogođen lokalni i udaljenu dolinu globalnog minimuma čija se pozicija čini sasvim neovisna o odnosu vrijednosti dobitka.

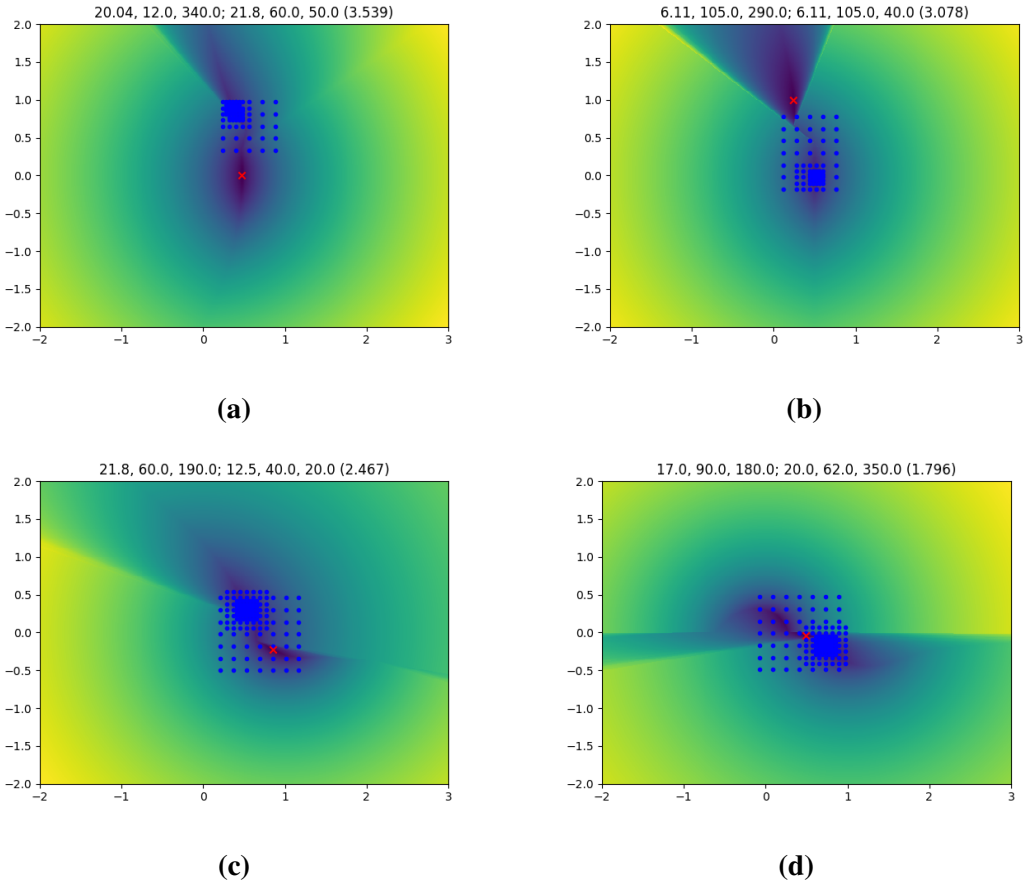
To dovodi do slijedećeg postupka određivanja početne točke koji koristi vrijednosti azimuta. Kombinacije azimute najviše utječu na izgled funkcije u svim slučajevima u kojima širina snopa iznosi manje od $180°$. Zato se početnoj točki koja se nalazi u središtu prostora mogu zbrojiti prosječni sinusi azimuta u smjeru x-osi i kosinusa u smjeru y-osi pomnoženi s udaljenosti središta od granica prostora. Tako će, na primjer, azimuti $(45° : 45°)$ postaviti početnu točku u "gornji desni" kut prostora, a $(45° : 315°)$ i $(315° : 45°)$ u "gornje" središte. No nije poželjno da se početna točka nalazi na samom rubu prostora jer se tada u istraživačkom koraku algoritmi pretrage samo djelomično prekriva prostor unutar granica pa se gubi na učinkovitosti. Iz tog se razloga vektor pomaka iz središta skalira vrijednošću manjom od 1 pa se tako početna točka primiče središtu.



(a) *Moving Grid Search* **(b)** Hooke Jeeves

Slika 3.6: Određivanje početne točke korištenjem vrijednosti azimuta

Iz primjera (3.6) se vidi kako nisu ni vrijednosti azimuta dovoljne same za sebe. U slučajevima poput (a) gdje se azimuti poništavaju $(0° : 180°)$ globalni minimum se može nalaziti u bilo kojoj od dolina ovisno o ostalim parametrima. U istom je primjeru je također vidljivo koliko je u takvom slučaju presudna razlika u vrijednostima dobitka pa je logično koristiti metodu koja ju koristi u kombinaciji s vektorom pomaka dobivenim iz azimuta. Slijedeći primjeri prvo pomiču početnu točku na pravcu između antena, a zatim zbrajaju vektor pomaka skaliran na polovicu svoje veličine.



Slika 3.7: Određivanje početne točke kombinacijom metoda

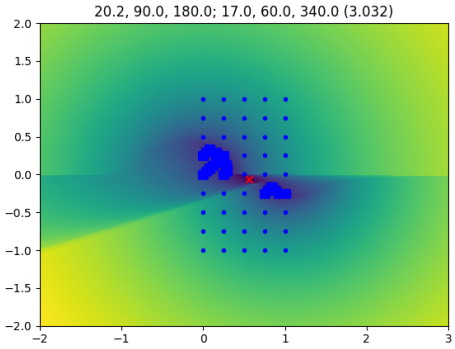
Moving Grid Search algoritam u slučajevima (a) i (b) na slici 3.7 koristi manju rešetku i nailazi na probleme da vektor pomaka ima ili previše jak ili previše slab utjecaj na početnu točku. Relativno malen pomak prema središtu u slučaju (a) i dalje od središta u slučaju (b) bi odveo pretragu prema globalnom minimumu, ali ovako ona u oba slučaja odlazi u suprotnom smjeru. U slučajevima (c) i (d), algoritam koristi veću rešetku pa takvi problemi nisu zastupljeni, ali s takvom rešetkom posebno određivanje početne točke nije niti potrebno.

No ovdje opstaju problemi uskih strmih dolina globalnih minimuma koje se ne istražuju iako se nalaze unutar područja koju rešetka pokriva. Puno je veća vjerojatnost da će neka od evaluiranih pozicija pogoditi nisku točku neke druge, šire doline pa će pohlepni algoritam pogrešno usmjeriti svoju pretragu. Slučaj (c) pokazuje kako je upravo iz ovog razloga kriva dolina proglašena boljom.

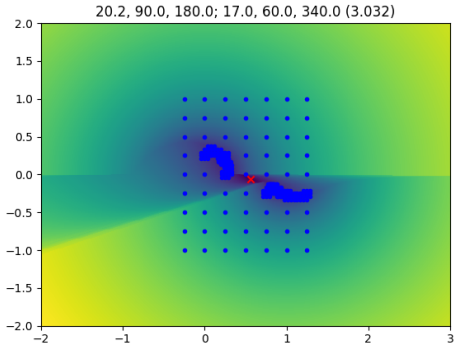
3.1.4. *Multiple Start Grid*

Multiple Start Grid je heuristika za odabir početne točke nastala iz nedostataka *Moving Grid Search* algoritma. Dio teksta koji govori o tim nedostacima (3.1.1) prikazuje kako je za povećanje preciznosti algoritma potrebna rešetka koja je što veća kako bi se dostigle udaljenje doline i gušća kako se manje doline ne bi preskočile. Istovremeno, na taj način se gubi učinkovitost algoritma jer takva rešetka nakon smanjivanja nije potrebna. Ovaj problem se može otkloniti time da se u prvom koraku koristi velika i gusta rešetka, a nakon prvog smanjivanja manja i rjeđa.

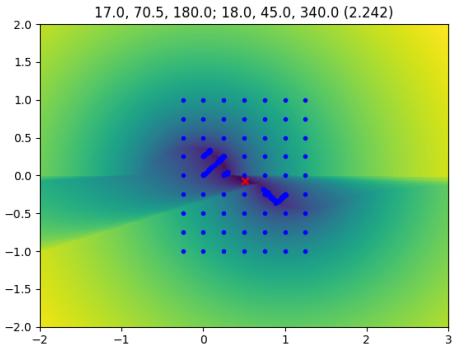
Nedostatak koji preostaje opisan je pri kraju prethodnog dijela, 3.1.3, koji govori o pohlepnoj prirodi algoritma zbog kojeg od dvije otkrivene doline može često odabrati pogrešnu. Ovo je, ali ne u potpunosti, riješeno dodavanjem novog parametra *Multiple Start Grid*-u iz kojeg ujedno dolazi i ime heuristike: broj točaka rešetke iz kojih se započinje pretraga. Osim toga, moguće je odabrati i koji algoritam će iz tih točaka izvršavati pretagu.



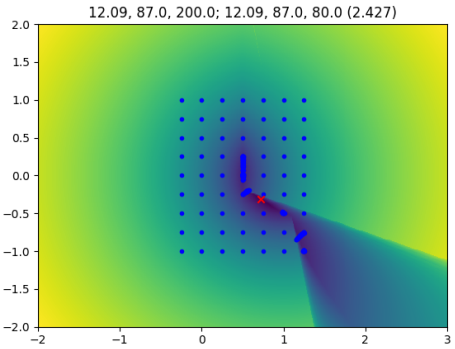
(a) 9x5, *Multiple Start Grid*



(b) 9x7, *Multiple Start Grid*



(c) 9x7, Hooke Jeeves



(d) 9x7, Hooke Jeeves

Slika 3.8: *Multiple Start Grid*

Prvi par primjera na slici 3.8, (a) i (b), pokazuju kombinaciju *Multiple Start Grid*-a s *Multiple Start Grid* algoritmom koji koristi rešetku malih dimenzija i nešto veću preciznost nego kod samostalnog korištenja. To je zato što je uloga algoritma ovdje samo fino pretraživanje, bez potrebe za širokim pregledom prostora. Pretrage se pokreće iz 6 različitih točaka što rezultitra pronalaskom dna nekoliko dolina. Razlika u dimenzijama *Multiple Start Grid*-a između primjera (a) i (b) prikazuje i utjecaj broja i rasporeda točaka na različit odabir čak i kad su parametri antena jednaki.

Primjeri (c) i (d) koriste Hooke Jeeves kao algoritam pretrage i sad vidljivo dolazi do izražaja njegova učinkovitost po pitanju broj potrebnih evaluacija za pronalazak minimuma. Mali početan korak i odabir početne točke smještaju algoritam unutar točno jedne doline za što je i originalno osmišljen.

Na primjerima (a), (b) i (c) svih 6 pretraga pokrenuto je u dvije od tri doline zbog njihovih razlika u veličini, ali za te parametre antena najniža točka se nalazi baš u toj, najmanjoj. Da bi i najmanja dolina bila zastupljena bilo bi potrebno puno više pokretanja pretrage, pogotovo ako ju točke rešetke većinom preskaču.

Unatoč učinkovitosti i preciznosti kombinacije heuristike za odabir početne točke i algoritama pretrage, i dalje ima puno slučajeva u kojima globalni minimum nije pronađen. Ako je rešetka heuristike rijetka, velika je vjerojatnost da će pretraga biti pokrenuta u različitim dolinama, ali i da će neka dolina biti preskočena. S druge strane, ako je rešetka gusta, puno najboljih točaka se nalazi unutar istih dolina pa se i broj pokretanja mora povećati kako bi se osiguralo da će više dolina biti zastupljeno. Zato ponovno dolazi do kompromisa između preciznosti i učinkovitosti.

3.1.5. Adam

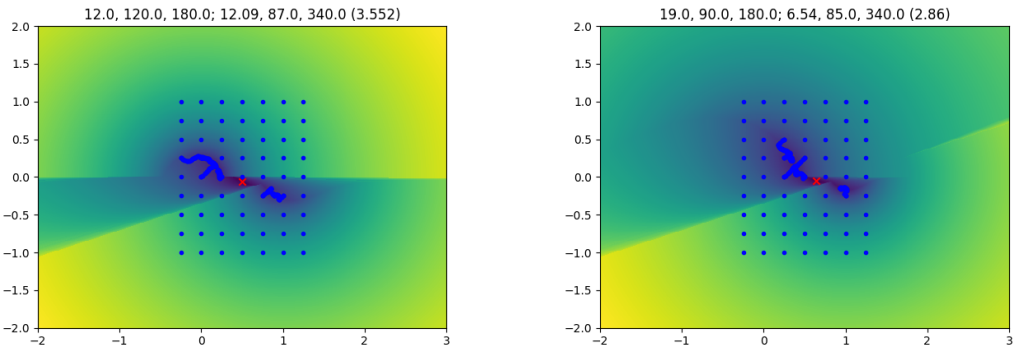
Uz pomoć *Multiple Start Grid* heuristike za određivanje početnih točaka pretrage moguće je koristiti i algoritme koji su još više od Hooke Jeeves-a specijalizirani za brz pronalazak optimuma unimodalnih funkcija. U praksi se za tu svrhu najčešće koriste gradijentni postupci poput algoritma Adam koji je trenutno popularan u učenju (dubokih) neuronskih mreža u bibliotekama poput *tensorflow*-a. Više o postupcima optimizacije, uključujući formule i pseudokodove može se naći u [4].

Kako nije jednostavno, a u nekim točkama niti moguće dobiti gradijent, on se računa pomoću malih pomaka u smjeru x i y-osi. Za računanje jednog gradijenta zato su potrebne tri evaluacije što je lošije od originalnog algoritma, ali jednako najboljem slučaju jednog koraka u Hooke Jeeves algoritmu. No korak za Adam algoritam je manji nego kod Hooke Jeeves-a pa njegova učinkovitost ovisi duljini puta koju mora proći do minimuma.

Osim uobičajenih parametara algoritma opisanih u literaturi, koristi se i maksimalan broj koraka kao uvjet zaustavljanja. Kako algoritam koristi moment, za njegov krajnji rezultat se umjesto krajnje točke uzima ona s najmanjom vrijednosti funkcije iz cijelog pređenog puta. "Stopa učenja" kojom se skalira gradijent ovdje se može biti prilagodljiva s obzirom na korake tako da je početno veća, a pri kraju manja.

Slijedeći primjeri koriste parametre postavljene na:

- $\rho_1 = 0.75$, $\rho_2 = 0.625$
- $pocetna\_stopa\_ucenja = 0.025$
- $konacna\_stopa\_ucenja = 0.0075$
- $maksimalno\_koraka = 20$



(a) **(b)**

Slika 3.9: *Multiple Start Grid* + Adam

S ovakvim parametrima Adam se uz *Multiple Start Grid* ponaša slično kao Hooke Jeeves, ali da bi dostigao njegovu preciznost treba koristiti manju veličinu koraka pa zato i veći maksimalan broj koraka. U konačnici koristi daleko veći broj evaluacija od ostalih algoritama zbog potrebe za računanjem gradijenta ispitivanjem okoline. Kad ne bi bilo ovog problema, algoritam bi se mogao pokazati boljim od ostalih zbog velike prilagodljivosti koju omogućuju njegovi parametri.

3.1.6. *Bisection Grid Search*

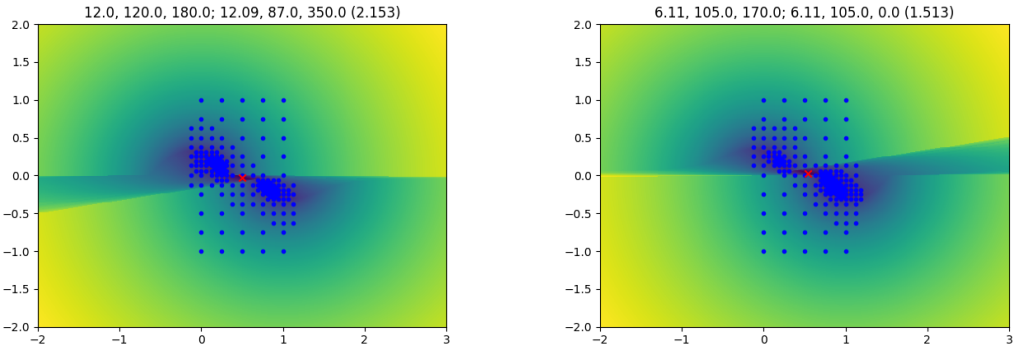
Višestruko pokretanje pretraga unutar istih dolina kod *Multiple Start Grid*-a bilo bi manji problem za učinkovitost kada bi se moglo otkriti prije ili tokom izvođenja. Tada bi se mogao spriječiti nastavak svih osim jedne pretrage i preseliti fokus na druga područja. Prva opcija, otkrivanje prije izvođenja, nije izvediva zbog velike raznolikosti oblika funkcija generiranih parametrima antena. No otkrivanje tokom izvođenja je moguće dodavanjem koordinacije pretraga koja bi se izvodila između svakog njihovog koraka. Ona bi, uz optimizaciju višestrukog pokretanja unutar istih, mogla i zaustaviti nepotreban nastavak istraživanja lošijih dolina.

Jedna implementacija takve koordinacije je algoritam *Bisection Grid Search*. Baziran je na kombinaciji *Multiple Start Grid*-a i *Moving Grid Search* pretrage, a ponašanjem sliči na generacijski populacijski algoritam. Ideja bisekcije je da u svakom koraku oko n najboljih točaka rešetka postane dvostruko gušća sve dok se ne dostigne zadana preciznost.

Prvu generaciju predstavljaju točke velike početne rešetke. Odabire se parametrom zadan broj najboljih i istražuje njihovo susjedstvo. Susjedstvo točke odgovara 3x3 rešetci s razmakom između točaka upola manjim od onog u početnoj rešetci. Odabrane točke sa svojim susjedstvom tvore slijedeću generaciju. Moguće je da se neke točke ponavljaju jer postoje slučajevi u kojima su lijevi susjedi jedne točke desni susjedi druge. Zato se između generacija uklanjaju duplikati. Točke nove generacije se evaluiraju i postupak se ponavlja s još jednom prepolovljenom veličinom razmaka u novom susjedstvu.

$tocke \leftarrow resetka$
while $razmak\_x > preciznost \bigvee razmak\_y > preciznost$ **do**
    $ukloni\_duplikate(tocke)$
    $evaluiraj(tocke)$
    $razmak\_x \leftarrow 0.5 * razmak\_x$
    $razmak\_y \leftarrow 0.5 * razmak\_y$
    $nove\_tocke \leftarrow prazna\_lista()$
    for all $tocka \leftarrow tocke$ **do**
        $susjedstvo \leftarrow stvori\_susjedstvno(tocka, razmak\_x, razmak\_y)$
        $nove\_tocke.dodaj(susjedstvo)$
    end for
    $tocke \leftarrow nove\_tocke$
end while

Slijedeći primjeri koriste početnu rešetku dimenzija 9x5 i razmakom 0.25, preciznost od 0.0025 i u svakoj generaciji uzima se 10 najboljih točaka.



(a) **(b)**

Slika 3.10: *Bisection Grid Search*

Bisection Grid Search algoritam se pokazao precizniji od svih prethodno korištenih. Gledajući prosječan broj evaluacija, za tu preciznost čak niti ne žrtvuje učinkovitost nego ju uvećava. To postiže uklanjanjem višestrukih evaluacija istih područja u slučajevima kad nekoliko pokrenutih pretraga završi u dnu iste doline. Osim toga, visoku učinkovitost postiže boljom raspodjelom točaka koje se evaluiraju kada se istovremeno pretražuje više dolina različite kvalitete. Na primjerima slike 3.10 vidi se kako su u prvom koraku pretrage susjedstva sve doline jednako zastupljene. Ali, po-

gotovo u slučaju (b), kroz nekoliko slijedećih generacija fokus se postepeno prebacuje na dolinu koja većim dijelom sadrži niske točke.

Iako je *Bisection Grid Search* poboljšanje u odnosu na kombinacije *Multiple Start Grid*-a i lokalnih pretraga, neki od nedostataka su i dalje ostali neriješeni. Na primjerima se vidi kako područje oko globalnog minimuma u prvim generacijama nije preskočeno nego preciznije istraženo, ali i dalje u nedovoljnoj mjeri. Kako su vrijednosti azimuta kontinuirane, uvijek je moguća nova kombinacija u kojoj će dolina globalno minimuma biti dovoljno mala da ju svaki algoritam osim iscrpne pretrage preskoči.

Drugi nedostatak algoritma je sporost izvođenja uzrokovana potrebom za sortiranjem evaluirane populacije i uklanjanjem duplikata. Iz tog razloga brzina algoritma nikad nije razmjerna njegovoj učinkovitosti.

3.2. Predikcija modelima strojnog učenja

Drugi pristup rješavanju problema pokušava potpuno zamijeniti pretragu prostora funkcije generirane maksimumom gubitaka dvaju antena modelom koji iz parametara antena izravno računa vrijednost funkcije u globalnom minimumu ili njegovu poziciju. Za tu svrhu vrlo su prikladni modeli koji se koriste u strojnom učenju, pogotovo regresijski. Takav model može na ulazu dobivati 6 parametara antena: 2 x dobitak, 2 x širina snopa i 2 x azimut i na izlazu vraćati vrijednost funkcije, odnosno jednu ili obje koordinate globalnog minimuma.

Modeli koji su se koristili nalaze se kao gotove implementacija u *scikit-learn* biblioteci, osim genetskog programiranja koje je dio ECF-a (*Evolutionary Computation Framework*). Iz bogatog izbora *scikit-learn*-a koristi su se:

- Linearna regresija
- Polinomijalna regresija
- Neuronske mreže (regresijski višeslojni perceptron)
- Regresijski SVM tj. SVR (*Support Vector Machine*)
- Regresijsko stablo odluke (eng. *Decision Tree*)
- KNN: k najbližih susjeda (eng. *K Nearest Neighbors*)
- Odborni strojevi: *AdaBoost* i *BaggingRegressor*

Više o genetskom programiranju može se naći u [1], o neuronskim mrežama i odbornim strojevima u [2], a ostalim modelima u [5].

Parametri na ulazu modela korišteni su u neizmijenjenom obliku, s povećanom dimenzionalnošću koristeći polinomijalnu transformaciju i s normaliziranim kutevima. Normalizacija kuteva preslikava raspon od $0°$ do $360°$ na raspon od -1 do 1. Ulazni podaci su također reducirani postupcima opisanim u 2.2.5 i 2.2.6. Izlazni podaci korišteni za učenje i ispitivanje modela dobiveni su iscrpnom pretragom prostora funkcije s odgovarajućim parametrima antena. Optimizacija hiperparametara svih modela za koje je to značajno izvodila se pomoću *Grid Search* unakrsne validacije koja dijeli skup za učenje *K-Fold* metodom na 5 dijelova. Skup za učenje koristi 70% nasumično odabranih podataka. Model s dobivenim hiperparametrima ponovno se uči nad cijelim skupom za učenje.

3.2.1. Linearna i polinomijalna regresija

U oba slučaja koristi se model linearne regresije s hiperparametrom koji određuje hoće li se koristiti normalizacija regresora. Polinomijalna regresija se koristi transformacijom ulaznih podataka pa stupanj te transformacije predstavlja dodatni parametar.

Običan linearni regresor se brzo uči i brzo izvodi predikciju, ali rezultati koje daje nisu dovoljno dobri. Problem nije linearan pa se od linearnog modela niti ne očekuje uspjeh. Polinomijalna transformacija svakim većim stupnjem više usporava učenje i predikciju, a koristi i više memorije. Usprkos povećanoj složenosti modela, rezultati nisu znatno bolji. Da bi se smatrali prihvatljivim rezultati moraju biti barem bolji od lokalnih pretraga s početnom točkom u središu prostora uz manje vrijeme izvođenja. Ovdje su oni lošiji za nekoliko redova veličine.

3.2.2. Neuronske mreže

Neuronske mreže učene su sa sigmoidalnim i ReLU prijenosnim funkcijama, Adam algoritmom za učenje ($\rho_1 = 0.7$ ili 0.9, $\rho_2 = 0.999$), maksimalnim brojem iteracija od oko 4000, sa i bez ranog zaustavljanja u slučaju porasta pogreške na skupu za validaciju te različitim vrijednostima tolarancije i konfiguracijama slojeva. Brojevi slojeva su u rasponu od 1 do 5, veličine slojeva od 2 do 100.

Bez obzira na velik broj parametara i mogućnost velike složenosti modela, rezultati nisu bili prihvatljivi. Trajanje treniranja modela ovisi o složenosti i korištenju ranog zaustavljanja dok je predikcija uglavnom brza.

3.2.3. SVR

Neki od *kernel*-a isprobanih za SVR nisu uspjeli završiti učenje čak niti nakon više od 10h izvođenja pa su preostali samo RBF (radijalne bazne funkcije) i sigmoidalni. Isprobane vrijednosti parametra C i ϵ su 10, 1, i 0.1, γ se računa automatski, a tolerancije su 1, 0.1 i 0.01.

Treniranje za ove *kernel*-e trajalo je prihvatljivo dugo, ali predikcija, čak i da daje dobre dobre rezultate, je prespora za korištenje u odnosu na trajanje pretrage prostora. Zbog ovog razloga, model niti nije dalje razmatran.

3.2.4. Regresijsko stablo odluke

Stablo odluke isprobano je uz različite parametre koji ograničavaju način njegove izgradnje i dubinu, ali najboljim se pokazalo bez ikakvih ograničenja. Kao takvo,

ono može "napamet" naučiti sve primjere iz skupa za učenje i pomoću njih približno odrediti izlaze za nove primjere.

Predikcija vrijednosti optimuma funkcije cilja pokazala se lošom, ali predikcija pozicije je za većinu slučajeva uspješna. Stablo se vrlo brzo uči i jednako brzo izvodi predikciju, ali zauzima količinu memorije proporcionalnu broju primjera za učenje.

3.2.5. KNN

KNN se pokazao najbolji kad koristi tri najbliža susjeda, udaljenost kao težinski faktor i Manhattan vrstu udaljenosti. Osim ovih vrijednosti isprobani su i 2 do 5 najbližih susjeda, uniformni težinski faktori i viši stupnjevi kod računanja Minkowski udaljenosti: 2 i 3. Algoritam za dohvaćanje susjeda bira se automatski između *Brute Force*, *BallTree* i *KDTree* algoritama.

Ovaj model se pokazao nešto boljim u predikciji pozicije od ostalih modela, ali je i dalje red lošiji i sporiji od stabla odluke. Kao i stablo, ima mogućnost učenja "napamet" svih primjera, ali uz manju učinkovitost.

3.2.6. Odborni strojevi

Odborni strojevi su korišteni kao pokušaj poboljšanja već relativno uspješnog stabla odluke. No najmanji pomak zahtijeva dodavanje po 10ak instanci modela u odbor što nije dovoljno dobro da bi opravdalo korištenje dodatne memorije i sporijeg vremena izvođenja.

3.2.7. Genetsko programiranje

Parametri koji određuju načine izgradnje populacije stabala ograničavaju dubinu između 3 i 8. Korišteni funkcijski čvorovi su: '+', '-', '$*$', '/', 'sin' i cos', a završni varijabilni čvorovi odgovaraju parametrima antena. Algoritam koji se koristio za optimizaciju stabla je *Steady State Tournament* s veličinom turnira 3. Za veličinu populacije su isprobane manje poput 30 pa sve do većih poput 1000. Maksimalan broj generacija uvijek je postavljen tako da dođe do konvergencije. Kako bi se izbjegli poznati problemi genetskog programiranja s dosezanjem točnih vrijednosti izlaza, one su prije optimizacije centralizirane oko 0 uz opcionalno skaliranje na raspon od -1 od 1.

Sva dobivena stabla nisu pokazala prihvatljive rezultate, nego je većina stavila 'sin' ili 'cos' kao korijen garantirajući da su svi izlazi bliski nuli i u rasponu od -1 do 1. Neki parametri antena često nisu ni bili iskorišteni dok su se drugi pojavljivali više puta.

3.2.8. Korištenje prenaučenih modela

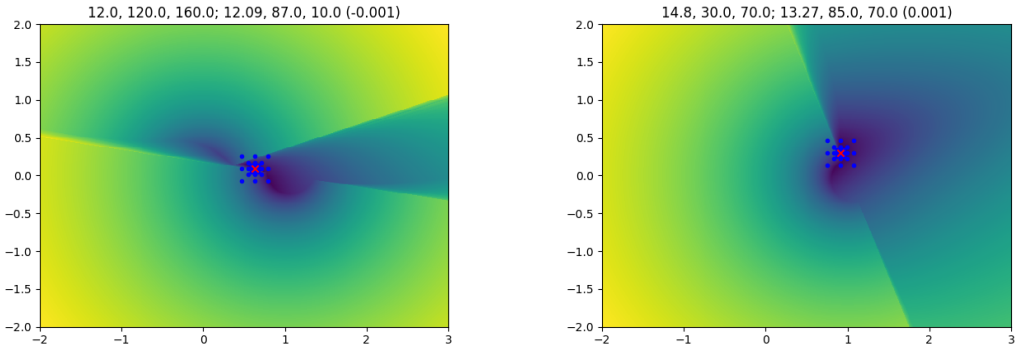
Problem koji se ovdje rješava nije tipičan problem strojnog učenja jer nema velike mogućnosti pojavljivanja novih neviđenih primjera. Tvorničke konfiguracije antena su u potpunosti zadane, a azimuti imaju ograničen raspon. Stoga je moguće koristiti modele koji se skupu za učenje prilagode što bolje moguće ili ga čak nauče u cijelosti tj. "napamet". Sve što preostaje je naći model koji će najbolje naučiti primjere uz najkraće prosječno vrijeme izvođenja predikcije.

Kao što je navedeno u njihovim opisima, za ovu ulogu su prikladni regresijsko stablo odluke i KNN. Oba mogu zapamtiti skup za učenje tako da im je pogreška predikcije nad njim jednaka nuli. Zbog drugačije strukture podataka i načina dohvaćanja rezultata iz naučenih primjera, stablo odluke je dva reda veličine brže od KNN-a i zauzima oko 15% manje memorije.

Kod računanja vrijednosti na izlazu oba modela koriste najsličnije primjere iz skupa za učenje što može u nekim slučajevima biti problematično. Kao što je prikazano na slici 2.7, manje promjene u parametrima mogu preseliti globalni minimum u drugu dolinu pa će pogreška predviđene pozicije biti velika. No kvaliteta krajnjeg rezultata tada ovisi o razlici dubina tih dolina, a ne njihovoj udaljenosti.

3.3. Kombiniranje predikcije i lokalne pretrage

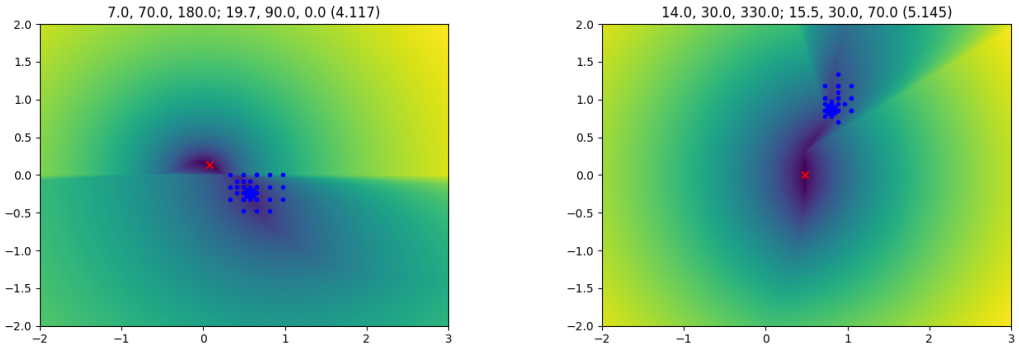
Kako lokalne pretrage jako ovise o kvaliteti odabira početne točke, a modelima za predikciju je potrebna dodatna pretraga prostora kad naiđu na neviđene primjere, njihova kombinacija može davati bolje rezultate od pojedinačnog korištenja. Prije svake pretrage pokrene se predikcija pozicije globalnog minimuma čiji se rezultat koristi kao početna točka pretrage. U najgorem slučaju, algoritam pretrage brzo će pronaći dno doline nešto lošije od najbolje. Ta dolina ne bi trebala biti bitno lošija jer je za bliske parametre antena opravo ona bila najbolja.



(a) *Moving Grid Search* **(b)** Hooke Jeeves

Slika 3.11: Kombinacija predikcije i lokalne pretrage

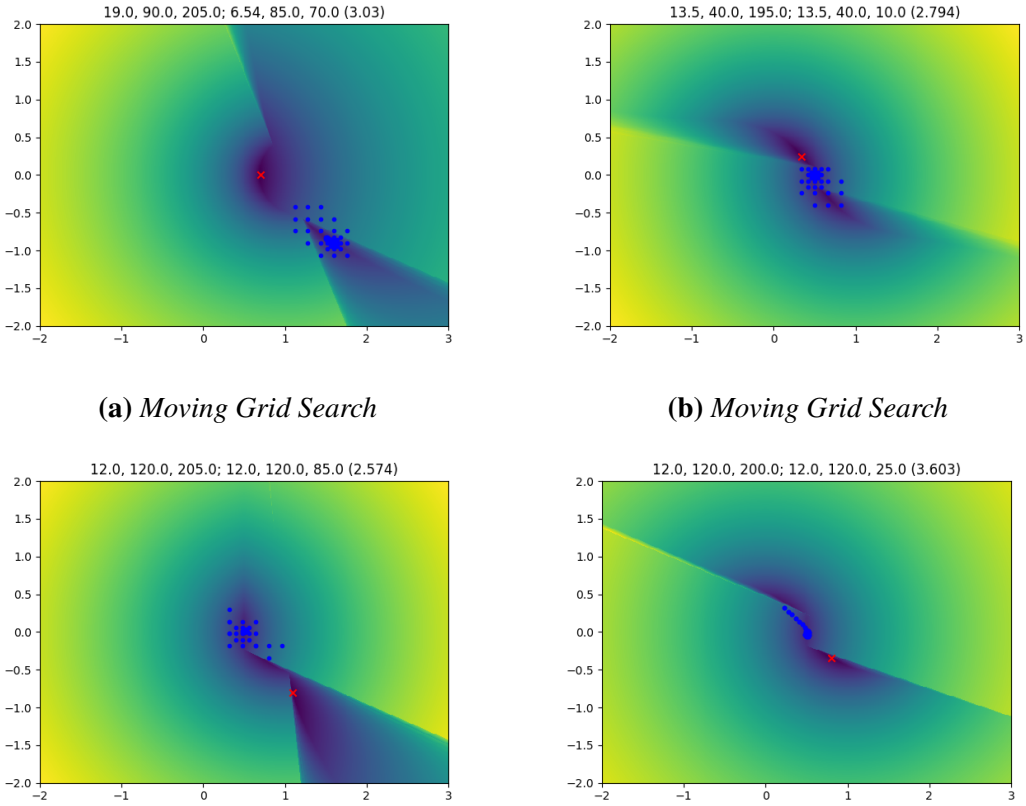
Kao što se vidi na slici 3.11, za naučene primjere lokalne pretrage odrade samo minimalan broj koraka. Ako je preciznost lokalne pretrage veća od iscrpne pretrage koje je generirala podatke za učenje, moguće je da se pronađe i nešto bolje rješenje.



(a) *Moving Grid Search* **(b)** Hooke Jeeves

Slika 3.12: Nove tvorničke konfiguracije antena

Dodavanje novih tvorničkih konfiguracija uzrokuje značajno lošije rezultate od očekivanih. Dolina lokalnog minimuma u kojoj se pokreće pretraga zna biti toliko lošija da bi ju sporiji algoritmi s većim početnim korakom uvijek uspjeli zaobići. Dva takva primjera prikazana su na slici 3.12.



(a) *Moving Grid Search*

(b) *Moving Grid Search*

(c) Hooke Jeeves

(d) Adam

Slika 3.13: Nove kombinacije azimuta

Slučajevi s novim kombinacijama azimuta također uzrokuju lošije rezultate, ali ponekad iz drugih razloga. Naime, manji pomak jednog od azimuta pomiče i rub doline preko pozicije gdje se prethodno nalazio minimum. Pretraga tako počinje s pogrešne strane prijeloma nastalog zbog funkcija maksimum i kreće se u suprotnom smjeru. Ovakvo ponašanje vidljivo je na primjerima (c) i (d) slike 3.13.

Zbog loše sposobnosti generalizacije, predikcija pomoću stabla odluke pokazale se dobrom samo kao učinkovit model za pamćenje viđenih podataka. Zato je najbolje iskorišten ili za višestruko ispitivanje rezultata istih parova antena ili uz kompleksniju lokalnu pretragu koja može pronaći i druge doline u prostoru.

4. Rezultati i usporedbe

Generiranje ispitnih primjera počinje filtriranjem tvorničkih konfiguracija opisanim u 2.2.6. Preostale konfiguracije spremaju se u cikličnu listu i slažu u parove. Svaka konfiguracija dodaje se u par sa svojim desnim susjedom i sa samom sobom. Tako dobivenim parovima dodjeljuju se kombinacije azimuta tako da se mogući raspon [$0°$, $360°\rangle$ uzorkuje zadanom veličinom koraka i zatim nad uzorcima vrši kartezijski produkt. Kada bi veličina koraka bila npr. $10°$ dobilo bi se 36 uzoraka iz kojih produkt stvara 1296 različitih kombinacija. Kombinacije azimuta reduciraju se postupkom opisanim u 2.2.5.

Parametri koji definiraju kako će se generirati podaci su:

- udaljenosti vrijednosti dobitka kod filtriranja tvorničkih konfiguracija
- udaljenost širine snopa kod filtriranja tvorničkih konfiguracija
- veličina koraka uzorkovanja azimuta

Skupovi podataka koji se ovdje koriste dobiveni su slijedećim vrijednostima parametara:

1. Bazni skup podataka:
   - udaljenost vrijednosti dobitka: 1.0
   - udaljenost širine snopa: $10°$
   - korak uzorkovanja azimuta: $10°$

   Veličina skupa: 57884

2. Skup podataka s dodatnim tvorničkim karakteristikama:
   - udaljenost vrijednosti dobitka: 0.5
   - udaljenost širine snopa: $5°$
   - korak uzorkovanja azimuta: $10°$

   Veličina skupa: 106786

3. Skup podataka s dodatnim kombinacijama azimuta:
   - udaljenost vrijednosti dobitka: 1.0
   - udaljenost širine snopa: 10°
   - korak uzorkovanja azimuta: 5°

   Veličina skupa: 228578

4. Skup podataka s dodatnim tvorničkim karakteristikama i azimutima:
   - udaljenost vrijednosti dobitka: 0.5
   - udaljenost širine snopa: 5°
   - korak uzorkovanja azimuta: 5°

   Veličina skupa: 421687

Algoritmi lokalne pretrage i predikcije ocjenjuju se prema nekoliko različitih mjerila. Ona najvažnija su vrijeme izvođenja, ukupna kvadratna pogreška, prosječna kvadratna pogreška i maksimalna apsolutna pogreška na jednom primjeru. Ukupna, odnosno prosječna kvadratna pogreška pokazuju koliko je algoritam prilagodljiv različitim oblicima funkcije. Zbog sortiranja konačnih rezultata bolje je imati više slučajeva s manjom greškom nego nekoliko s velikom pa se zato koristi kvadratna pogreška umjesto apsolutne. Maksimalna pogreška pokazuje koliko će loše u najgorem slučaju bliskost nekog para antena biti ocijenjena tj. kolika će biti najveća promjena u mreži susjedstva.

Prosječna kvadratna pogreška računa se samo za primjere gdje algoritam daje pogrešan rezultat. Kao takva može biti mala uz velik broj pogrešnih primjera ili veća uz malo pogrešnih primjera. Iz tog razloga broj pogrešnih primjera se navodi uz prosječnu pogrešku. Također, kako bi bilo razumljivo kolika je veličina najveće pogreške u odnosu na raspon mogućih vrijednosti bilježe se i najveća i najmanja pronađena vrijednost.

Osim pogreške u vrijednostima funkcije mjere se i prosječna i maksimalna udaljenost pozicije rješenja od globalnog minimuma koristeći euklidsku udaljenost. Ovo je pogotovo bitno za uspoređivanje različitih metoda određivanja početne točke. Uz to, mjeri se koliko je udaljenja pozicija rješenja u slučaju s najvećom pogreškom vrijednosti funkcije i koliko je pogreška vrijednosti funkcije u slučaju s najvećom udaljenosti. Ovo pokazuje koliko su općenito korelirani pogreška vrijednosti funkcije i udaljenost od globalnog minimuma.

Uz vrijeme mjeri se i prosječan broj evaluacija funkcije radi usporedbe učinkovitosti algoritama pretrage, ali ovdje ne predstavlja presudno svojstvo.

Sva vremenska mjerenja izvedena su na Intel Core i7-3630QM procesoru koristeći 8 dretvi za paralelnu obradu skupa podataka podijeljenog na 8 jednakih dijelova.

Podaci slijedeće četiri tablice (4.1, 4.2, 4.3, 4.4) dobiveni su koristeći skup podataka 1. uz navedene metode određivanja početne točke. Parametri algoritama odabrani su tako da osiguraju najmanju ukupnu kvadratnu pogrešku za slučaj početka iz sredine prostora.

Za ovaj skup podataka minimalna vrijednost globalnog minimuma je 121.624, a maksimalna 163.125. Radi usporedbe, ukupno vrijeme izvođenja 10 evaluacija za svaki primjer je oko 0.03s.

Za tablice 4.1 i 4.2 parametri *Moving Grid Search* algoritma su dimenzije rešetke 13x5, početan razmak između točaka 0.16 i preciznost 0.01. Za Hooke Jeeves veličina početnog koraka je 0.64, a preciznost također 0.01.

Tablica 4.1: Rezultati za početne točke u središtu prostora

	Moving Grid Search	Hooke Jeeves
Vrijeme izvođenja (s)	1.468	0.245
Prosječan broj evaluacija	378.31	54.4
Ukupna kvadratna pogreška	224.63	41833.591
Prosječna kvadratna pogreška	0.1562	1.3692
Broj pogrešnih rješenja	1438	30554
Maksimalna pogreška	2.794	13.538
Prosječna udaljenost rješenja	0.007	0.057
Maksimalna udaljenost rješenja	1.187	1.235
Pogreška maksimalno udaljenog	0.214	1.73
Udaljenost maksimalne pogreške	0.233	0.508

Tablica 4.2: Rezultati za početne točke dobivene korištenjem razlike vrijednosti dobitka

	Moving Grid Search	Hooke Jeeves
Vrijeme izvođenja (s)	1.412	0.193
Prosječan broj evaluacija	379.51	53.7
Ukupna kvadratna pogreška	266.158	43888.694
Prosječna kvadratna pogreška	0.014	1.1959
Broj pogrešnih rješenja	19028	36700
Maksimalna pogreška	2.799	13.538
Prosječna udaljenost rješenja	0.014	0.062
Maksimalna udaljenost rješenja	1.187	1.235
Pogreška maksimalno udaljenog	0.169	1.73
Udaljenost maksimalne pogreške	0.291	0.508

Za tablice 4.3 i 4.4 rešetka *Moving Grid Search* algoritma ima dimenzije 7x5 jer time ne gubi kvalitetu rješenja.

Tablica 4.3: Rezultati za početne točke dobivene korištenjem vrijednosti azimuta

	Moving Grid Search	Hooke Jeeves
Vrijeme izvođenja (s)	0.991	0.186
Prosječan broj evaluacija	263.17	53.25
Ukupna kvadratna pogreška	290.432	9656.628
Prosječna kvadratna pogreška	0.0103	0.239
Broj pogrešnih rješenja	28099	40396
Maksimalna pogreška	2.465	9.183
Prosječna udaljenost rješenja	0.021	0.049
Maksimalna udaljenost rješenja	1.214	1.181
Pogreška maksimalno udaljenog	0.749	0.219
Udaljenost maksimalne pogreške	0.583	0.617

Tablica 4.4: Rezultati za početne točke dobivene korištenjem kombinacije vrijednosti dobitka i azimuta

	Moving Grid Search	Hooke Jeeves
Vrijeme izvođenja (s)	1.075	0.197
Prosječan broj evaluacija	263.31	53.25
Ukupna kvadratna pogreška	289.894	8636.568
Prosječna kvadratna pogreška	0.0101	0.2127
Broj pogrešnih rješenja	28714	40595
Maksimalna pogreška	2.47	6.865
Prosječna udaljenost rješenja	0.02	0.05
Maksimalna udaljenost rješenja	1.214	1.182
Pogreška maksimalno udaljenog	0.749	0.264
Udaljenost maksimalne pogreške	0.583	0.924

Najvažnije što je potrebno napomenuti za podatke u ovim tablicama je velika promjena nastala dodavanjem korištenja vrijednosti azimuta. Ono je omogućilo pomicanje početne točke u smjeru y-osi i time uvelike smanjilo ukupnu kvadratnu pogrešku Hooke Jeeves algoritma. *Moving Grid Search* nije vidio velike promjene kod ukupne pogreške, ali je zato dobio mogućnost uklanjanja 6 redaka iz rešetke što je rezultiralo manjim potrebnim vremenom izvođenja.

Podaci slijedeći tablice (4.5) dobiveni su korištenjem skupa podataka 1. i početnom točkom postavljem u središte prostora. Dimenzije rešetke *Multiple Start Grid*-a su 9x7 s razmakom između točaka od 0.25 i započinje 6 pretraga prostora. *Moving Grid Search* koji koristi za pretragu ima rešetku dimenzija 3x3, početan razmak između točaka 0.04 i preciznost 0.0025. Hooke Jeeves ima veličinu koraka 0.08 i istu preciznost. Adam koristi $\rho 1 = 0.75$, $\rho 2 = 0.625$, početnu stopu učenja 0.005 i konačnu 0.0005 te maksimalan broj koraka 35. *Bisection Grid Search* koristi rešetku dimenzija 9x5, početan razmak između točaka 0.25 i preciznost 0.0025, a u svakoj generaciji bira 10 najboljih točaka.

Tablica 4.5: Rezultati *Multiple Start Grid* i *Bisection Grid Search* algoritama

	MSG + MGS	MSG + HJ	MSG + Adam	BGS
Vrijeme izvođenja (s)	2.124	1.516	2.759	2.333
Prosječan broj evaluacija	508.39	424.23	699	355.38
Ukupna kvadratna pogreška	177.368	176.989	275.009	37.613
Prosječna kvadratna pogreška	0.04	0.0443	0.0713	0.0058
Broj pogrešnih rješenja	4430	3991	3858	6493
Maksimalna pogreška	3.033	2.748	3.727	2.149
Prosječna udaljenost rješenja	0.013	0.013	0.014	0.013
Maksimalna udaljenost rješenja	1.067	1.067	1.067	1.031
Pogreška maksimalno udaljenog	0.141	0.141	0.141	-0.047
Udaljenost maksimalne pogreške	0.351	0.318	0.183	0.328

Iz ovih podataka vidi se kako uz jednako kvalitetne uvjete Hooke Jeeves može biti i brži i imati manju ukupnu kvadratnu pogrešku od *Moving Grid Search*-a. Također je vidljivo koliko bolje rezultate *Bisection Grid Search* ima u odnosu na sve ostale algoritme pretrage, a i manji broj evaluacija od svih *Multiple Start Grid* inačica. Ali, sporost izvođenja ga sprječava da bude proglašen nedvojbeno najboljim.

Negativna vrijednosti u tablici znači da je algoritam zbog povećane preciznosti u drugoj dolini našao bolje rješenje od iscrpne pretrage.

U tablicama 4.6 i 4.7 za ispitivanje modela koriste se navedeni skupovi podataka, a oba modela su naučena nad cijelim skupom 1. Regresijsko stablo koristi nabolje parametre učenja navedene u 3.2.4, a KNN one navedene u 3.2.5.

Tablica 4.6: Rezultati ispitivanja za KNN

	Skup 2.	Skup 3.	Skup 4.
Vrijeme izvođenja (s)	1.141	1.881	4.645
Prosječna udaljenost	0.0268	0.0114	0.0325
Maksimalna udaljenost	1.036	0.96	1.08

Tablica 4.7: Rezultati ispitivanja za regresijsko stablo odluke

	Skup 2.	Skup 3.	Skup 4.
Vrijeme izvođenja (s)	0.017	0.028	0.054
Prosječna udaljenost	0.0246	0.0135	0.0325
Maksimalna udaljenost	1.16	1.16	1.16

Oba modela ovdje imaju slične rezultate gledajući kvalitetu predikcije, ali regresijsko stablo odluke ima daleko veću brzinu izvođenja. Iz tog razloga ono je bolje i za samostalno korištenje i u kombinaciji s algoritmima za pretragu prostora.

Tablice 4.8 i 4.9 koriste skup podataka 4. za dobivanje rezultata i regresijsko stablo odluke naučeno na cijlom skupu 1. za odabir početne točke. Algoritmi pretrage u prvoj tablici koriste iste parametre kao za tablice 4.3 i 4.4, posebno Adam koristi iste parametre kao u kombinaciji s *Multiple Start Grid*-om u tablici 4.5 osim što koristi makimalan broj koraka 50. U drugoj tablici svi algoritmi koriste iste parametre kao za tablicu 4.5.

Za skup 4. najmanja pronađena vrijednost globalnog minimuma je 120.964, a najmanja 163.125. Ukupno vrijeme izvođenja 10 evaluacija za svaki primjer je oko 0.14s.

Kako bi se bolje vidio pridonos algoritama lokalne pretrage u tablicu 4.8 dodaje se i samo regresijsko stablo odluke. Najbolje vrijednosti za svaki redak obje tablice su podebljane.

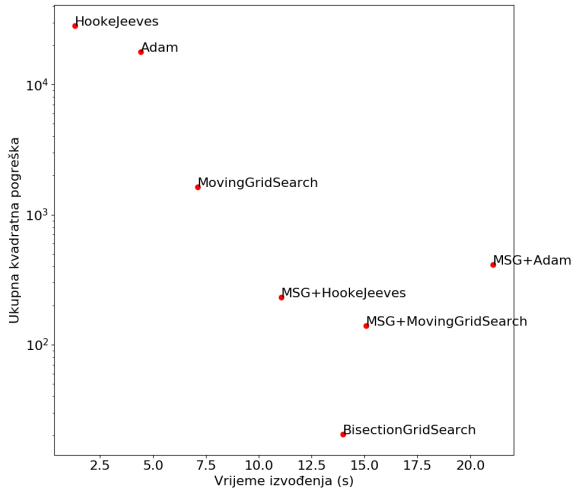
Tablica 4.8: Rezultati predikcije početne točke (1/2)

	Stablo	MGS	HJ	Adam
Vrijeme izvođenja (s)	0.06	7.111	**1.321**	4.416
Prosječan broj evaluacija	1	261.32	**51.72**	151
Ukupna kvadratna pogreška	2560494.224	1633.125	28418.148	17830.451
Prosječna kvadratna pogreška	8.2692	0.0812	0.1482	0.4421
Broj pogrešnih rješenja	309642	**20100**	191796	40331
Maksimalna pogreška	26.961	4.673	10.164	8.302
Prosječna udaljenost rješenja	0.052	**0.008**	0.034	0.025
Maksimalna udaljenost rješenja	1.215	1.329	1.352	1.29
Pogreška maksimalno udaljenog	0.251	1.302	3.142	3.806
Udaljenost maksimalne pogreške	0.295	0.578	0.475	**0.226**

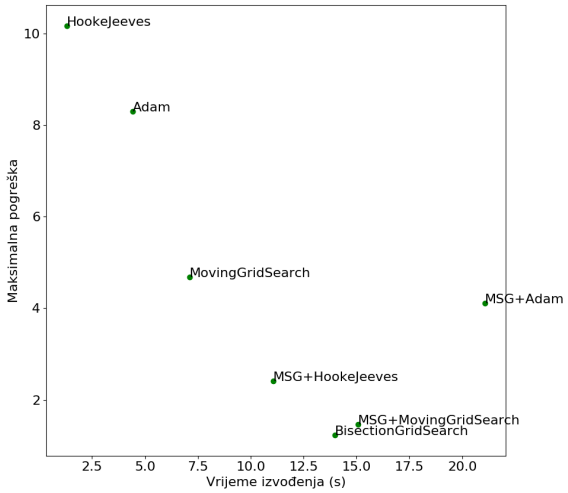
Tablica 4.9: Rezultati predikcije početne točke (2/2)

	MSG + MGS	MSG + HJ	MSG + Adam	BGS
Vrijeme izvođenja (s)	15.073	11.067	21.067	13.988
Prosječan broj evaluacija	509.59	422.09	699	375.02
Ukupna kvadratna pogreška	139.206	231.635	411.244	**20.428**
Prosječna kvadratna pogreška	0.0054	0.0098	0.017	**0.0004**
Broj pogrešnih rješenja	25932	23548	24135	45473
Maksimalna pogreška	1.462	2.409	4.114	**1.232**
Prosječna udaljenost rješenja	0.011	0.011	0.013	0.012
Maksimalna udaljenost rješenja	1.268	1.268	**1.11**	1.108
Pogreška maksimalno udaljenog	0.025	0.025	**-0.090**	-0.071
Udaljenost maksimalne pogreške	0.252	0.571	0.304	0.336

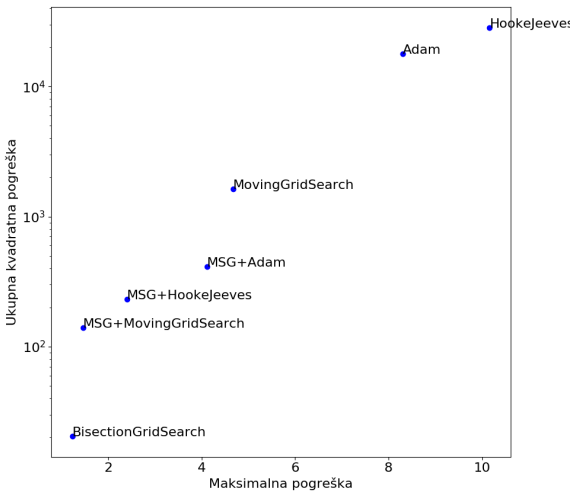
Iz ovih rezultata može se vidjeti da različiti algoritmi pretrage imaju različite prednosti, ali neki od njih su očito bolji od drugih gledajući sva bitna mjerila. Ovakve usporedbe zapravo gledaju pareto učinkovitost algoritama i mogu se prikazati grafički.



Slika 4.1: Pareto prikaz vremena izvođenja i ukupne kvadratne pogreške



Slika 4.2: Pareto prikaz vremena izvođenja i maksimalne pogreške



Slika 4.3: Pareto prikaz maksimalne i ukupne kvadratne pogreške

Na slikama 4.1 i 4.2 vidi se kako veće vrijeme izvođenja u većini slučajeva rezultira manjom pogreškom, dok slika 4.3 prikazuje pozitivnu korelaciju pogreške u najgorem slučaju s ukupnom. I jedno i drugo je istinito kod usporedbe različitih algoritama, ali ne vrijedi uvijek kad se međusobno uspoređuju različite vrijednosti parametara istog algoritma. Posebno treba napomenuti da je *Bisection Grid Search* jedini algoritam od svih prikazanih koji u sve tri slike spada u prvu pareto frontu.

5. Zaključak

Optimizacija procjene susjedstva antena mobilne telekomunikacijske antene pokazala se kao vrlo složen problem. Broj mogućih kombinacija parametara, čak i nakon reduciranja, je dovoljno velik da je teško pronaći algoritam pretrage prostora koji bi funkcionirao dobro u svim slučajevima. Parametri uzrokuju veliku raznolikost i nepravilnost oblika funkcije pa tako ujedno i otežavaju učenje modela za predikciju. Male promjene u vrijednostima parametara mogu uzrokovati velike promjene u očekivanom rezultatu pa je modelima teško konstruirati pravila koja opisuju takvo ponašanje.

Najveći problem kod pronalaska dobrog rješenja predstavljaju uvjeti da ono treba ujedno biti brzo i precizno. Rezultati algoritama pretrage su pokazali kako je odabir između brzine i preciznosti kompromis te da je takvim metodama nemoguće postići oboje. Ono što bi moglo pomoći je metoda odabira početne točke koja se uvijek nalazi unutar doline globalnog minimuma.

Modeli strojnog učenja pokazali su se neuspješnim kod predikcije vrijednosti funkcije izvan primjera viđenih tokom učenja, ali predikcija pozicije koje je samo do određene mjere netočna može se ispraviti dodatnom pretragom prostora. Problem u ovom slučaju leži u tome da za ispravno naučene primjere ta pretraga niti nije potrebna, dok kod se neviđenih primjera ponovno javljaju isti problemi kad predviđena točka nije unutar ispravne doline.

Najbolje pronađeno rješenje problema je kombinacija regresijskog stabla odluke za predikciju pozicije s *Bisection Grid Search* algoritmom pretrage prostora uz redukciju ekvivalentnih kombinacija parametara antena sa svrhim pojednostavljenja modela. No ono ne zadovoljava uvjet prosječne brzine izvođenja koja bi trebala biti približna algoritmu koji samo izvodi 5-15 evaluacija funkcije. Samo stablo odluke je dovoljno brzo, ali bi koristilo veliku količinu memorije i bilo nešto sporije kad bi u svrhu dostizanja potrebne preciznosti naučilo gotovo sve moguće kombinacije parametara.

Rješenje koje bi zadovoljavalo sve uvjete možda bi se moglo pronaći koristeći visoko specijalizirani algoritam pretrage, model predikcije ili oboje. Tako rješenje bi sadržavalo unaprijed određeno znanje o samoj funkciji ili analizom dobivena pravila ponašanja i kao takvo bilo neuoprebljivo izvan svoje domene. Alternativno, umjesto modela predikcije koji bi "napamet" učio sve moguće primjere, mogla bi se koristiti struktura podataka prikladna za takve poslove poput neke od implementacija *lookup* tablica.

LITERATURA

[1] Domagoj Jakobović. *Rješavanje optimizacijskih problema algoritmima evolucijskog računanja u Javi - Uvod u GP*. Fakultet elektrotehnike i računarstva, 2016.

[2] Sven Lončarić i Marko Subašić. *Neuronske mreže - Predavanja*. Fakultet elektrotehnike i računarstva, 2016. URL `http://www.fer.unizg.hr/predmet/neumre_b/predavanja`.

[3] H.-P. Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, Inc., 2001. URL `http://ls11-www.cs.tu-dortmund.de/lehre/wiley/`.

[4] Marko Čupić. *Optimizacija parametara modela*. Fakultet elektrotehnike i računarstva, 2016. URL `http://www.zemris.fer.hr/~ssegvic/du/du3optimization.pdf`.

[5] Jan Šnajder i Bojana Dabelo Bašić. *Strojno Učenje*. Fakultet elektrotehnike i računarstva, 2014. URL `http://www.fer.unizg.hr/_download/repository/StrojnoUcenje.pdf`.

Optimizacija procjene susjedstva antena mobilne telekomunikacijske mreže

Sažetak

U ovom radu opisuje se i analizira problem optimizacije procjene susjedstva antena mobilne telekomunikacijske mreže. Predstavljaju se mogući pristupi njegovom rješavanju: algoritmi lokalne pretrage, modeli strojnog učenja i njihova kombinacija. Za svaki pristup detaljnije se opisuju ispitane implementacije. Rezultati ispitivanja međusobno se uspoređuju te se komentira ukupna uspješnost rješavanja problema.

Ključne riječi: optimizacija, mobilna telekomunikacijska, telekomunikacijske antene, algoritmi lokalne pretrage, strojno učenje, genetsko programiranje

Neighbourhood Assessment Optimization for Mobile Telecommunications Network Antennas

Abstract

This thesis describes and analyses the problem of optimizing the neigborhood assessment for mobile telecomunication network antennas. Possible approaches to its solution are presented: local search algorithms, machine learning models and combination of the two. Tested implementations for each approach are described in more detail. Test results are compared and the overall success in solving the problem is commented on.

Keywords: optimization, mobile telecommunication, telecommunication antennas, local search algorithms, machine learning, genetic programming