Poređenje metaheuristika u optimizaciji dozne raspodele za simulacije zračenja karcinoma oka

Aleksandar Todorović
Fizički fakultet, Univerzitet u Beogradu
Računarski fakultet, Univerzitet Union

Apstrakt

Precizno planiranje radioterapije zračenja tumora zahteva određivanje parametara koji utiču na apsorbovanu dozu zračenja u tumoru i okolnim tkivima. Ručno pronalaženje i podešavanje ovih parametara je često vrlo sporo i ne dovodi do zadovoljavajućih rešenja, zato su u relevantnoj literaturi korišćene metaheurističke metode za optimizaciju. Kao primer navodimo genetski algoritam (Genetic Algorithm, GA) i optimizacija kolonijom čestica (Particle Swarm Optimization, PSO). U ovom radu primenjene su tri metaheurističke metode i to, GA, PSO i tabu pretraživanje (Tabu Search, TS), sa ciljem da se što efikasnije i automatizovano pronađu pet ključnih ulaznih parametara u Monte Carlo simulacijama FOTELP-VOX programa za simuliranje zračenja karcinoma oka. Korišćena mera kvaliteta radioterapije predstavlja odstupanje dobijene doze raspodele zračenja od preporučene.

Dobijeni rezultati ukazuju da se korišćenjem PSO algoritma dostiže najmanja apsolutna razlika između preporučene doze i dobijene doze u najkraćem mogućem periodu. TS algoritam postiže rezultate bliske PSO-u uz stabilniju konvergenciju parametara, ali zahteva duže vreme izvršavanja. GA postiže najlošije rezultate među ispitivanim metaheurstičkim metodama. Sve tri metode značajno nadmašuju analitičko traženje parametara i ručno pokretanje simulacija, i vremenski (naročito zbog mogućnosti paralelizacije) i po dobijenoj grešci.

Sadržaj

1 Motivacija

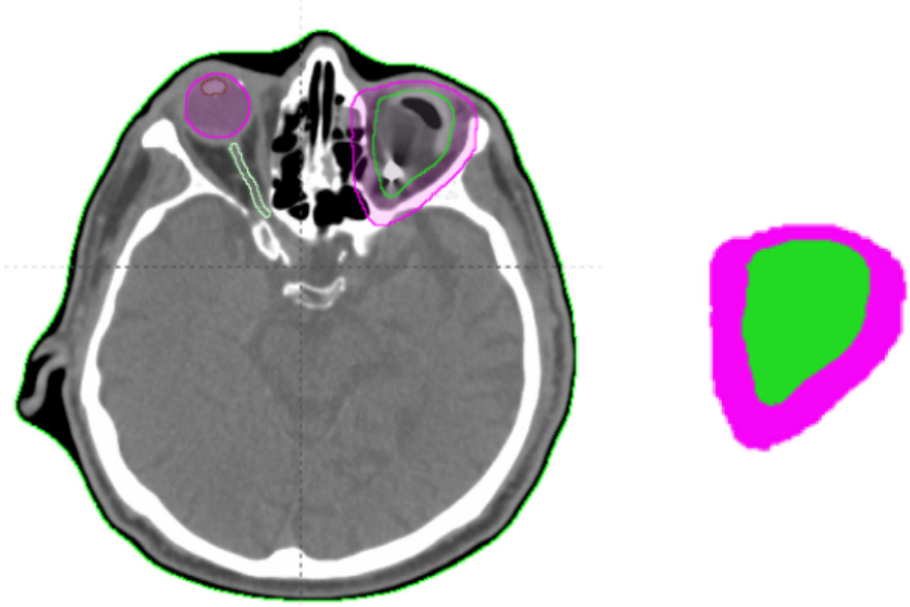
Radioterapija predstavlja ključnu komponentu u lečenju karcinoma oka, gde je neophodno postići visok nivo preciznosti zbog izuzetno male veličine tumora kao i blizine osetljivih organa i tkiva kao što su očni živac, makula i sočivo. Potrebno je postići balans između efikasnog zračenja malignog tkiva i zaštine zaštite zdravih tkiva. Minimalna odstupanja u snopu zračenja mogu značajno da utiču na promenu trodimenzionalne raspodele zračenja, što za posledicu dalje može imati da tumor nije ozračen dovoljnom količinom zračenja ili da su zdrava tkiva ozračena količinom zračenja većom od dozvoljene.

Iako su savremeni modaliteti poput IMRT i VMAT značajno unapredili oblikovanje dozne raspodele, glavni izazov ostaje neizmenjen, a to je precizno definisanje parametara ulaznog snopa zračenja i geometrije tumora pri planiranju radioterapije. Monte Karlo simulacije, o kojima će biti više reči u narednom poglavlju, su uglavnom standard za probleme ovog tipa jer mogu detaljno da modeluju kretanje elektrona i fotona kroz tkiva. Njihova glavna mana je visoka računarska cena i dugo trajanje ovakvih simulacija. Zbog toga, ručno ispitivanje ulaznih parametara snopa za definisanje radioterapije postaje neefikasno.

U kliničkoj praksi, podešavanje ovih parametara je često zasnovano na višegodišnjem iskustvu medicinskih fizičara, međutim ovaj način pronalaska dovoljno dobrih parametara osim što može da traje dosta dugo, takođe je neretko i vrlo subjektivan. Na ovaj način se ne može uzeti u obzir čitav skup rešenja parametara, niti odrediti neki koristan metod za njihovo pronalaženje.

Upravo iz ovih razloga, automatizacija pronalaska ulaznih parametara za definisanje radioterapije se nameće kao prirodan korak u pokušaju za pronalazak rešenja ovog problema. Metaheurističke metode, kao što su genetski algoritmi, optimizacija rojem čestica i tabu pertraga omogućavaju efikasnu pretragu velikog broja parametara bez potrebe za definisanjem konkretne analitičke metode pronalaska sledećeg potencijalnog rešenja. Ipak, računarska cena svake simulacije i njeno vremensko trajanje se ne poboljšavaju korišćenjem ovih metoda (jednoj simulaciji je potrebno isto vreme da se izvrši i uzima istu energiju) pa je sledeće pitanje koje se prirodno postavlja koja od metaheuristika bi bila najpogodnija za korišćenje pri optimizaciji.

Da bi se jasnije prikazala složenost anatomije oka i položaj tumora u odnosu na okolna tkiva, ispod je prikazan CT presek mozga gde je na levoj slici obeležen tumor kao i tkiva i organi koji su najviše osetljivi na zračenje, a na desnoj slici je prikazan izolovan tumor zelenom bojom, ali je za njegovu površinu ipak uzet malo veći deo, koji je označen roze bojom. Slika je preuzeta iz rada [8].



Slika 1: Prikaz izgleda tumora na CT snimku (levo) i prikaz njegovih granica (desno)

2 Teorijski uvod

2.1 Monte Karlo simulacije zračenja i model FOTELP-VOX

Monte Karlo metode predstavljaju numerički pristup koji je zasnovan na velikom broju slučajnih simulacija. Ideja je da se fizički proces, koji u opštem slučaju može biti poprilično složen, modeluje tako što se nasumično generišu mnoge moguće putanje ili događaji. Statistička analiza njihovog ishoda daje približno rešenje problema. Ovakav pristup je posebno koristan kada se rešenja problema ne mogu naći analitički.

U oblasti radioterapije, kod lečenja karcinoma određenih organa, Monte Karlo metode se koriste za simulacije prolaska fotona i elektrona (odnosno zračenja) kroz tkiva organa. Pri tome se za svaku česticu određuje da li će se i gde tačno će se ona raspršiti, apsorbovati, promeniti pravac ili odreagovati na neki drugi način. Time je moguće dobiti trodimenzionalnu raspodelu apsorbovane doze, koja je znatno preciznija od klasičnih analitičkih modela.

FOTELP-VOX je softver koji vrši Monte Karlo simulacije koristeći CT snimke pacijenata, na kojima se nalazi zahvaćena kancerogena regija [8]. CT se pretvara u mrežu, odnosno u vokselni model, gde svaki voksel nosi informaciju o gustini i materijalu tkiva. Nakon toga, snop zračenja se propagira kroz ovu strukturu, a program izračunava koliko je energije apsorbovano u svakom vokselu.

FOTELP-VOX program pre početka simulacije zračenja zahteva definisanje više ulaznih parametara (orijentacija snopa, geometrija izvora, položaj tumora i okoline tumora). Kako pojedina simulacija traje relativno dugo kada se uzme u obzir ukupan broj mogućih kombinacija početnih parametara, ručno podešavanje parametara je neefikasno i često vodi do suboptimalnih rešenja, pa se iz tog razloga za traženje početnih parametara koriste optimizacioni algoritmi.

2.2 Funkcija cilja i merenje greške

Za svaku dobijenu doznu raspodelu na kraju simulacije izračunata je greška u odnosu na preporučenu kliničku vrednost doze. Ta vrednost može da iznosi između $40Gy$ i $60Gy$ za svaki voksel, dok je u ovom radu uzeta za tačno $60Gy$. U simulaciji je korišćeno 27 „slajsova", odnosno 27 preseka xy ravni kroz deo mozga, pri čemu je za svaki presek jasno definisan položaj tumora.

Greška je računata posebno za svaki slajs i to isključivo u delu „pravougaonika" koji okružuje tumor (a o kome će biti reči u narednoj sekciji). Korišćena metoda greške je bila MSE, odnosno Mean Squared Error:

$$E_i = \frac{1}{M} \sum_{j=1}^{M} (D_j^i - 60)^2 \tag{1}$$

gde i ide od 1 do 27 i predstavlja redni broj slajsa, M je ukupan broj voksela u jednom slajsu (u svakom slajsu je jednak, jer je pravougaonik koji okružuje tumor uzet za konstantan u svakom slajsu), D_j^i vrednost apsorbovane doze zračenja u j-tom vokselu, i-tog slajsa.

Prema tome, ukupna greška jedne simulacija iznosi:

$$E_{total} = \frac{1}{27} \sum_{i=1}^{27} E_i \tag{2}$$

Ovakav pristup obezbeđuje stabilno i konzistentno poređenje različitih razultata simulacija, pri čemu niža vrednost greške označava apsorbovanu dozu zračenja koja je bliža preporučenoj i samim tim bolja od neke doze koja ima višu vrednost greške. Funkcija greške svakako nije jedinstvena i moguće ju je definisati na mnogo načina, preko različitih metrika sve dok se održava konzistentno i dosledno rangiranje kvaliteta apsorbovanih doza.

3 Optimizacione metode

Za optimizaciju datog problema, ovde se razmatraju tri različita optimizaciona algoritma: genetski algoritam (GA), koji koristi principe evolucije i populacione pretrage; optimizacija rojem čestica

(PSO), optimizaciona metoda zasnovana na kolektivnom ponašanju rojeva i tabu pretraga (tabu search), deterministička metoda pretrage sa memorijom koja omogućava izbegavanje lokalnih minimuma. Ove metode se koriste u opštem slučaju, kod problema u kojima se za evaluaciju jednog rešenja koriste vremenski relativno skupe simulacije jer omogućavaju brzo usmeravanje pretrage ka parametrima čiji rezultati simulacije daju nižu grešku i to bez potrebe za obilaskom celog skupa parametara.

3.1 GA optimizacija

GA je metoda optimizacije koja je prvenstveno inspirisana biološkom evolucijom. U uopštenom slučaju, radi nad populacijom jedninki gde je svaka jedinka predstavljena vektorom čije su koordinate parametri problema [1]. Svaka jedinka u populaciji predstavlja potencijalno rešenje problema.

Proces GA se sastoji iz evaluacije, selekcije, ukrštanja i mutacije. Evaluacijom se određuje koliko je jedinka kvalitetna i to se obavlja najčešće preko funkcije cilja ili funkcije greške. Selekcijom se biraju jedinke koje će se reprodukovati favorizujući kvalitetnija rešenja. Ukrštanjem se stvara nova jedinka čiji geni predstavljaju kombinaciju gena (koordinata vektora) roditelja te jedinke. Verovatnoća ukrštanja je jedan od parametara GA i on predstavlja koliko često će se primenjivati ukrštanje između roditelja. Mutacijom se na određen način menjaju geni kako bi se održala raznovrsnostpopulacije i izbeglo konvergiranje ka nekom od potencijalnih lokalnih minimuma. Još jedan od standardnih parametara GA je verovatnoća mutacije koja određuje koliko je verovatno da će pojedinačni gen (ili čitava jedinka) biti promenjeni unutar definisanog opsega.

Algoritam često može da koristi i elitizam, pri čemu najbolja rešenja iz prethodne generacije sigurno prelaze u narednu generaciju, kako se ne bi izgubilo najbolje rešenje.

Pseudokod ovog algoritma izgleda ovako:

```
# inicijalizacija
pop = inicijalizuj_populaciju(24)
evaluiraj(pop)

for gen in range(35):
    # cuvanje najboljeg (elitizam)
    elita = nadji_najboljeg(pop)

    # selekcija i ukrstanje
    roditelji = selekcija(pop)
    potomci = []
    for p1, p2 in u_parovima(roditelji):
        if random() < 0.6:
            c1, c2 = crossover(p1, p2)
        else:
            c1, c2 = p1, p2
        potomci.extend([c1, c2])

    # mutacija
    for ind in potomci:
        if random() < 0.4:
            mutacija(ind)

    evaluiraj(potomci)

    # ubacivanje elite umesto najgoreg potomka
    najgori = indeks_najgoreg(potomci)
    potomci[najgori] = elita

    pop = potomci

najbolji = nadji_najboljeg(pop)
```

```
prikazi(najbolji)

def selekcija(pop):
    # Turnirska selekcija: biraj po 2 jedinke nasumicno, najbolja prolazi
    roditelji = []
    while len(roditelji) < len(pop):
        turnir = nasumicno_izaberi(pop, 2)
        najbolji = nadji_najboljeg(turnir)
        roditelji.append(najbolji)
    return roditelji
```

3.2 PSO algoritam optimizacije

PSO algoritam je populacioni algoritam inspirisan kolektivnim kretanjem rojeva u prirodi i već postoje radovi u kojima se vidi njegova primena pri planiranju radioterapije [7]. Svaka čestica predstavlja potencijalno rešenje problema i ima poziciju i brzinu u prostoru rešenja.

U svakoj od iteracija, promena brzine je uslovljena preko više komponenti, od kojih su izdvojene tri najbitnije (u našem slučaju) [3]. Prva komponenta je inercioni faktor koji određuje koliko čestica zadržava svoju prethodnu brzinu i pravac i smer kretanja. Veći inercioni faktor daje stabilnije i sporije promene pri kretanju, dok manji faktor omogućuje brže i bolje prilagođavanje novim uslovima. Druga komponenta je kognitivni koeficijent koj ireguliše koliko čestica „veruje" sopstvenom najboljem rešenju. Veća vrednost pojačava težnju čestice ka sopstvenom, do sada najbolje pronađenom, položaju. Poslednja komponenta je socijalni koeficijent koja reguliše koliko će čestica da teži globalnom najboljem rešenju. Veća vrednost pojačava uticaj najboljeg rešenja na celokupno kretanje populacije.

PSO se koristi za minimizaciju funkcije greške, što je analogno i GA. Za svaku jedinku, evaluacija se vrši kroz pojedinačnu simulaciju, a cilj je smanjenje greške.

Pseudokod ovog algoritma izgleda ovako:

```
# inicijalizacija cestica i njihovih brzina
cestice = nasumicna_resenja()
brzine = nasumicne_brzine()
licna_najbolja = cestice
globalna_najbolja = najbolja_cestica(licna_najbolja)

za svaku iteraciju do max_iteracija:
    za svaku cesticu:
        # azuriranje brzine prema licnoj i globalnoj najboljoj poziciji
        brzina = kombinacija(trenutna_brzina, razlika_do_licne, razlika_do_globalne)

        # pomeranje cestice
        cestica = cestica + brzina

        # evaluacija nove pozicije
        rezultat = oceni(cestica)

        # azuriranje licne najbolje pozicije
        ako je rezultat bolji od licne_najbolje:
            licna_najbolja = cestica

    # azuriranje globalne najbolje pozicije
    globalna_najbolja = najbolja_cestica(licna_najbolja)

vrati globalna_najbolja
```

3.3 Tabu search algoritam

Tabu pretraga je lokalna pretraga kojom se ispituje okolina trenutnog rešenja, pri čemu se uvodi tabu lista kako bi se izbeglo vraćanje u skoro posećene tačke [2]. Ideja je da se u svakoj iteraciji generišu susedna rešenja, a potom izabere najbolje među njima, uz dodatno ograničenje kako bi se izbeglo vraćanje u već posećene tačke. Ovo ograničenje se uspostavlja preko tabu liste, u kojoj se čuva nekoliko prethodno posećenih rešenja. Na ovaj način se smanjuje rizik zaglavljivanja u lokalnom minimumu.

Još jedno svojstvo ovog algoritma je aspiracija, odnosno mogućnost izbora rešenja koje se nalazi u tabu listi ako ono donosi bolje globalno rešenje od dosadašnjeg najboljeg rešenja. Dakle, postoji mogućnost kada algoritam može da pređe preko restrikcija i potencijalno pronađe kvalitetnija rešenja.

Pseudokod ovog algoritma izgleda ovako:

```
current = inicijalno_resenje()
best = current
tabu_list = []

for iteracija in range(1, max_iter + 1):
    susedi = generisi_susede(current)
    najbolji_sused = pronadji_najboljeg_ne_tabu(susedi, tabu_list)

    if najbolji_sused is None:
        # aspiracija: dozvoli tabu ako je bolje od globalnog
        najbolji_sused = aspiracija(susedi, best) or current

    current = najbolji_sused
    if oceni(current) < oceni(best):
        best = current

    # azuriraj tabu listu
    dodaj_u_tabu(tabu_list, current)
    odrzi_velicinu_tabu(tabu_list)

# rezultat
return best
```

4 Rezultati simulacija

4.1 Ulazni parametri simulacije

U okviru FOTELP-VOX simulacija, raspodela apsorbovane doze direktno zavisi od niza ulaznih parametara poput: dimenzija pravougaonika u kome je smešten tumor, centra mase tumora, uglova koji definišu snop zračenja... U ovom radu, fokus je na optimizaciji upravo preko uglova koji definišu snop zračenja, konkretno njih pet [6]. Oni zajedno određuju potpunu orijentaciju snopa u odnosu na karcinom oka. Prva dva predstavljaju standardne Ojlerove uglove odnosno azimut, koji opisuje rotaciju snopa oko vertikalne ose, i elevacija, ugao koji predstavlja rotaciju oko horizontalne ose. Preostala tri ugla predstavljaju dodatne interne rotacije koji omogućavaju fino i precizno oblikovanje snopa i njegovo usmeravanje ka centru tumora. Svaki od ovih uglova u programu FOTELP-VOX može uzimati celobrojne vrednosti u opsegu $[0, 359]$ stepeni.

Može se primetiti da je ukupan broj ulaznih parametara svakako veći u opštem slučaju, ipak, preostali parametri su u ovom radu fiksirani. Za koordinate pravougaonika koji okružuje tumor su uzete takve koordinate da se kreira najmanji mogući pravougaonik koji i dalje okružuje tumor, dok je za centar mase tumora uzet centar takvog pravougaonika.

Takođe, svaka od metoda je pokretana sa istim, odnosno sa analognim, početnim parametrima kako bi upoređivanje rezultata bilo konzistentno. Ove metode mogu da se izvršavaju paralelno,

na više jezgara procesora, te kako bi se ubrzao proces svaka od njih je pokretana na šest jezgara. Upravo zbog sličnosti parametara koji karakterišu optimizacione metode, svaka od njih je trajala približno $22h$ i $30min$.

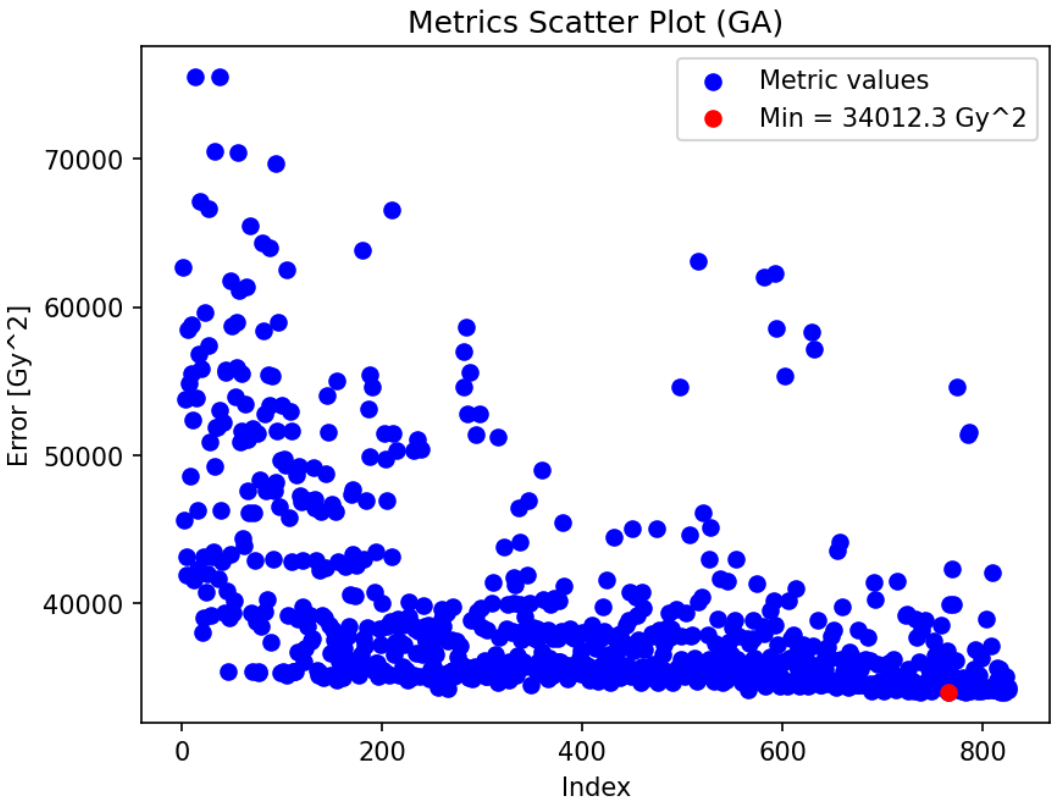
4.2 Rezultati GA optimizacije

U ovoj implementaciji GA, svaka jedinka je predstavljena vektorom od 5 celobrojnih vrednosti koje definišu 5 karakterističnih uglova u simulaciji zračenja. Sam algoritam je koristio pomoćnu Python biblioteku DEAP [1], koja omogućava lakše korišćenje nekih metoda u genetskoj optimizaciji.

Selekcija je realizovana turnirskim putem, pri čemu se iz populacije nasumično biraju grupe od dve jedinke, a najbolja jedinka iz svake grupe prelazi u narednu fazu reprodukcije. Verovatnoća ukrštanja iznosi 0.6, što znači da se 60% parova roditelja ukršta. Verovatnoća mutacije iznosi 0.4 nad svakom jedinkom. Ova relativno velika vrednost verovatnoće je korišćena kako bi se povećala raznovrsnost male populacije od 24 jedinke. Pri manjim vrednostima je primećeno zapinjanje u nekim lokalnim minimumima već nakon tri-četiri generacije.

Kao što je već rečeno, populacije se inicijalizuje sa 24 jedinke, sa unapred zadatim početnim vrednostima uglova za prvu generaciju. Te iste vrednosti su postavljene i u PSO algoritmu i u Tabu search-u, kako bi se obezbedila što manja razlika među parametrima samih algoritama. Evolucija se odvijala kroz 35 generacija. Primenjen je i elitizam, čime se obezbeđuje da najbolja jedinka u trenutnoj generaciji sigurno pređe u narednu.

Na kraju ove optimizacione metode, dobija se sledeći grafik:



Slika 2: Grafik jedinki genetskog algoritma

Na y-osi je prikazana vrednost greške, odnosno razlike dobijene dozne raspodele i preporučene dozne raspodele. Na x-osi je predstavljen redni broj svake od jedinki, gde vrednosti desno predstavljaju kasnije generacije i jedinke u odnosu na vrednosti levo. Crvena tačka predstavlja minimalnu vrednost greške.

[1] `https://deap.readthedocs.io/`

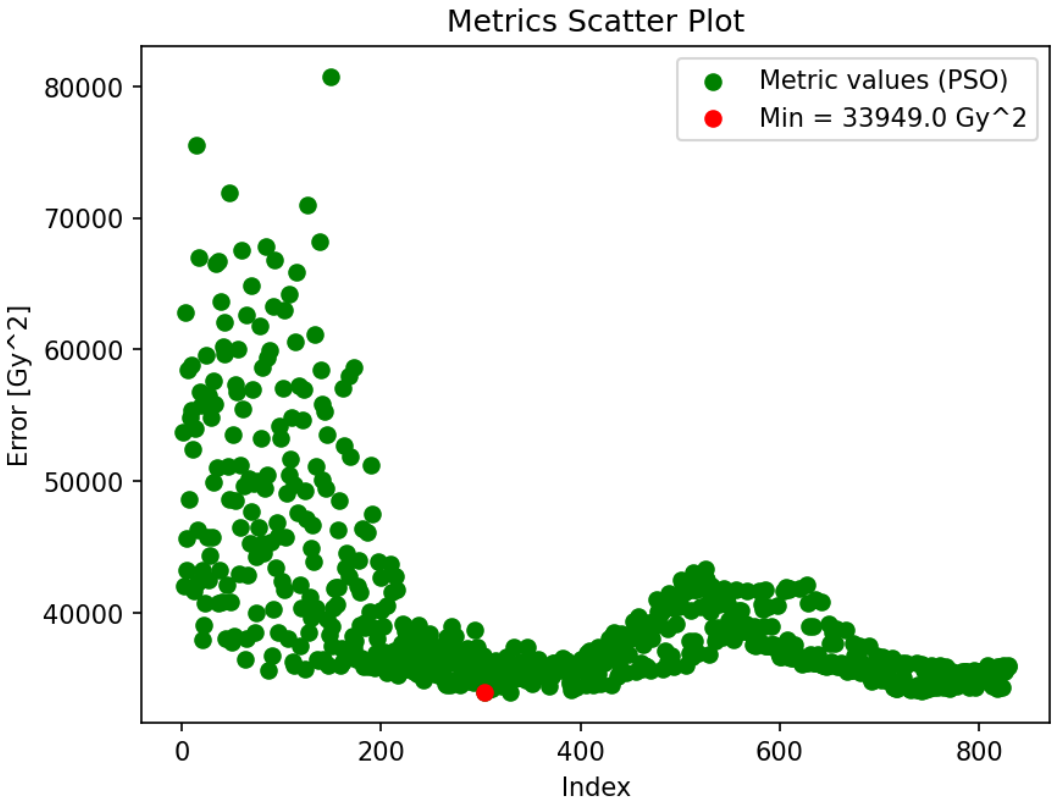
Sa grafika se vidi da se minimalna vrednost greške postiže pri kraju samog programa (pri nekoj od poslednjih generacija). Simulacije relativno brzo konvergiraju ka greškama koje iznose nešto manje od $40000Gy^2$, mada se konstantno uočava prisustvo nekoliko jedinki čije su vrednosti grešaka drastično veće (što se eventualno može pripisati agresivnijim parametrima mutacije i ukrštanja).

4.3 Rezultati PSO optimizacije

U ovoj implementaciji PSO algoritma, analogno GA, korišćene su 24 jedinke i 35 iteracija. Korišćene su neke metode biblioteke PySwarms[2] kako bi se pojednostavio kod. Parametri algoritma su postavljeni u standardne opsege: kognitivni koeficijent iznosi 1,6, socijalni koeficijent iznosi 1,9, dok inercioni faktor iznosi 0,6.

Takođe, kako ne bi bilo prevelike raznolikosti, uvedeno je ograničenje za skok brzine u opsegu (-20, 20), čime se stabilizuje kretanje čestica.

Na kraju ove optimizacione metode, dobija se sledeći grafik:



Slika 3: Grafik jedinki PSO algoritma

Na dobijenom grafiku se primeti još brža i stabilnija konvergencija ka nižim vrednostima greške. Takođe, primećuje se i pokušaj za promenom vrednosti uglova i traženjem nekog potencijalno boljeg rešenja (vrednosti indeksa između 400 i 700), međutim vrednosti se vraćaju na staro jer algoritam nije uspeo naći potencijalno bolje rešenje pri malo većoj promeni uglova. Minimalna vrednost greške se postiže mnogo brže nego u slučaju GA (približno dva puta brže). Pored toga, vrednost greške koja se dobija je nešto manja nego pri genetskoj optimizaciji.

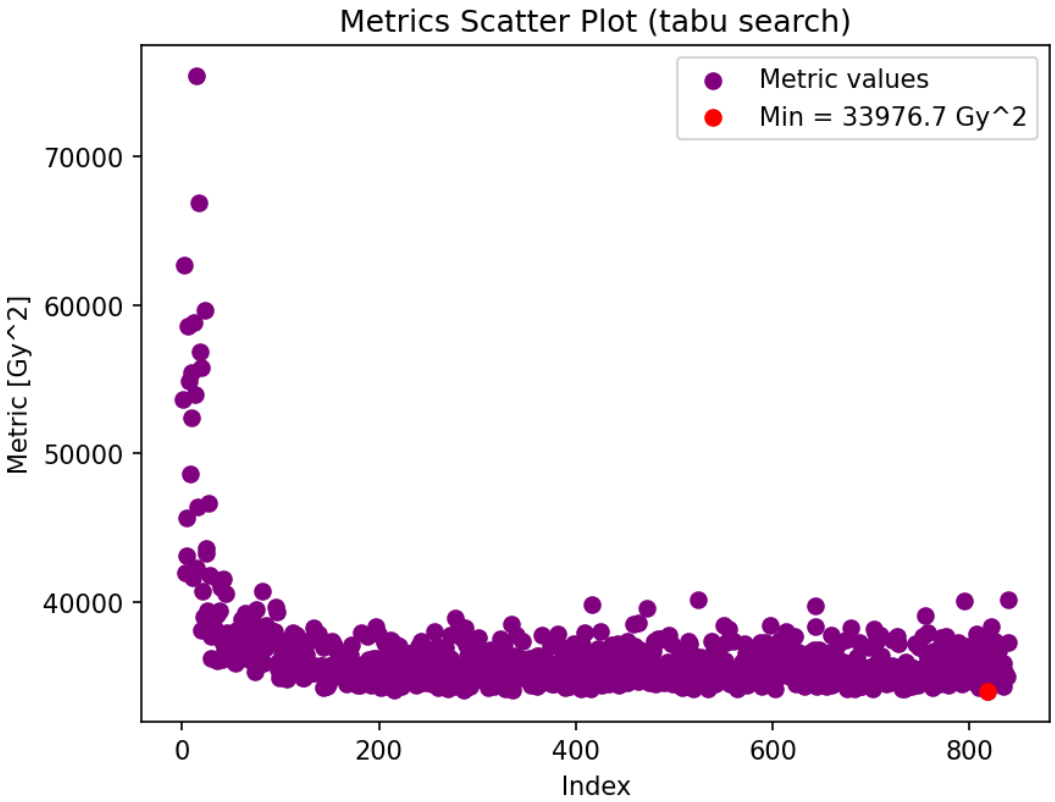
4.4 Rezultati Tabu search optimizacije

U ovom radu, Tabu search algoritam je pokrenut na 35 iteracija. Za svaku iteraciju, generiše se 24 suseda oko trenutnog rešenja (što je donekle analogno broju od 24 jedinke u PSO i GA

[2] `https://pyswarms.readthedocs.io/`

optimizaciji). Početno rešenje je uzeto za najbolje od unapred definisanih rešenja iz prve iteracije. U slučaju da je najbolje tabu rešenje bolje od globalnog minimuma, primenjuje se aspiracija, tj. prevazilaženje tabu restrikcije. Veličina tabu liste se održava fiksnom i iznosi 5.

Na kraju ove optimizacione metode, dobija se sledeći grafik:



Slika 4: Grafik jedinki tabu search algoritma

Sa grafika se primećuje da tabu search algoritam još brže konvergira ka nekim minimalnim vrednostima. Međutim, globalna minimalna vrednost greške se dostiže tek pred kraj samog programa i ona ipak nije bolja od globalne minimalne vrednosti greške dobijene PSO algoritmom, ali je bolja od minimalne vrednosti greške dobijene GA.

5 Diskusija i zaključak

Rezultati sva tri optimizaciona pristupa ukazaju na to da je moguće značajno smanjiti grešku u raspodeli apsorbovane doze pomoću automatizovane pretrage prostora parametara koristeći optimizacione metode. Međutim, razlike u načinu na koji algoritmi pretražuju prostor rešenja dovode do primetnih razlika u performansama.

Genetski algoritam je pokazao zadovoljavajuće ponašanje, ali sa širokim spektrom kvaliteta rešenja i sporijim konvergiranjem. Razlog toga može potencijalno da leži zbog nešto agresivnijih parametara mutacije zbog čega je proces pretrage često „širok”, ali ne nužno dovoljno stabilno usmeren ka minimumu.

Tabu pretraga je davala stabilnije i ravnomernije poboljšanje kvaliteta rešenja kroz iteracije. Zahvaljujući lokalnoj pretrazi i mehanizmu tabu liste algoritam se efikasno kretao kroz okolinu trenutnog rešenja i izbegavao cikluse. Ipak, postavlja se pitanje da li su rezultati tabu pretrage ovakvi zbog povoljnih početnih tačaka koje su dovele do brže konvergencije ka tačkama čija rešenja imaju manje greške.

Najbolje rezultate postigao je PSO. Ovaj algoritam je kombinovao globalno i lokalno usmeravanje čestica, zahvaljujući čemu se prostor rešenja pretraživao sistematičnije u poređenju sa GA.

Istovremeno, za razliku od tabu pretrage, PSO nije zavisio od jednog trenutnog rešenja već od kolektivnog kretanja čitave populacije. To je dovelo do bržeg opadanja greške i postizanja najnižeg globalnog minimuma među svim metodama.

Sveukupno, iako sva tri algoritma uspešno smanjuju grešku, rezultati ukazuju da PSO najbrže i najpouzdanije pronalazi rešenja visokog kvaliteta, dok tabu pretraga zauzima srednje mesto, a genetski algoritam pokazuje najveću varijabilnost.

U narednim istraživanjima bi bilo korisno da se proširi optimizacija na veći skup ulaznih parametara o kojima je bilo reči u Teorijskom uvodu, kao što su na primer geometrija veličina koje definišu granice tumora. Trenutni rad optimizuje isključivo uglove snopa, što daje samo delimičnu kontrolu nad krajnjom doznom raspodelom zračenja, a uključivanjem dodatnih parametara bi moglo da se vidi da li se i u tom slučajo PSO algoritam ponaša najstabilnije.

Pored toga, bilo bi korisno ispitati kako parametri koji figurišu u korišćenim optimizacionim algoritmima (poput broja jedinki, faktora mutacije, brzine kod PSO-a, veličine tabu liste i slično) utiču na konačne rezultate optimizacije. Poseban akcenat može se staviti na fine-tuning ovih parametara [5], odnosno na analizu njihove osetljivosti i međusobne zavisnosti, kako bi se utvrdilo u kojim slučajevima dovode do poboljšanja, a u kojim do degradacije rezultata optimizacije dozne raspodele.

Takođe, efikasan naredni korak bi bio i ispitivanje hibridnih optimizacionih pristupa. Na primer, PSO algoritam bi se mogao iskombinovati sa lokalnom pretragom iz tabu search-a, a to sve zajedno bi potencijalno moglo da da bolje rezultate u odnosu na samu primenu PSO algoritma. Naravno, moguće je koristiti i neke druge hibridne kombinacije, kao i uključiti još koji algoritam optimizacije.

Napomena. Rezultati su dobijeni tokom letnje istraživačke prakse u Matematičkom institutu SANU, pod mentorstvom Milene Živković, Filipa Andrića, Tatjane Davidović i Dragutina Ostojića.

Literatura

[1] **D.E. Goldberg** Genetic Algorithms in Search, Optimization, and Machine Learning. *Addison-Wesley*, 1989.

[2] **F. Glover**, Tabu Search—Part I, *ORSA Journal on Computing*, 1989, 1(3), 190-206.

[3] **J. Kennedy, R. Eberhart**, Particle Swarm Optimization, *Proceedings of IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.

[4] **D. Petrovic, M. Morshed, S. Petrovic** Multi-objective genetic algorithms for scheduling of radiotherapy treatments for categorised cancer patients. *Expert Systems with Applications*, 2011, 38, 6994-7002.

[5] **J. Rađenović** Uputstvo za irace na Linux-u. *Dostupno online na* `https://www.mi.sanu.ac.rs/~tanjad/UputstvoZaIRace.pdf`, 2020.

[6] **V.T. Taasti, L. Hong, J.S.A. Shim, J.O. Deasy, M. Zarepisheh** Automating proton treatment planning with beam angle selection using Bayesian optimization. *Medical Physics*, 2020, 47, 3286-3296.

[7] **G. Yang, W. Li, W. Xie, L. Wang, K. Yu** An improved binary particle swarm optimization algorithm for clinical cancer biomarker identification in microarray data. *Computer Methods and Programs in Biomedicine*, 2024, 244, 107987:1-17.

[8] **M. Zivkovic, F. Andric, M. Svicevic, D. Krstic, L. Krstic, B. Pirkovic, T. Miladinovic, M.E.A. Aichouche** FOTELP-VOX-OA: Enhancing radiotherapy planning precision with particle transport simulations and Optimization Algorithms. *Computer methods and programs in biomedicine*, 2025, 268, 108838:1-15.