Sortiranje objekata više atributnim kriterijima uporabom genetskih algoritama

Balko Macsai, Zoltan

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Diplomski sveučilišni studij Računarstvo

SORTIRANJE OBJEKATA VIŠE ATRIBUTNIM KRITERIJIMA UPORABOM GENETSKIH ALGORITAMA

Diplomski rad

Zoltan Balko Macsai

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Zoltan Balko Macsai
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1264R, 07.10.2022.
JMBAG:	0165081261
Mentor:	prof. dr. sc. Damir Blažević
Sumentor:	prof. dr. sc. Tomislav Keser
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 1:	prof. dr. sc. Damir Blažević
Član Povjerenstva 2:	izv. prof. dr. sc. Ivica Lukić
Naslov diplomskog rada:	Sortiranje objekata više atributnim kriterijima uporabom genetskih algoritama
Znanstvena grana diplomskog rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Za zadani skup podataka i zadatak sortiranja po višestrukim kriterijima opisati problem sortiranja, napraviti pregled stanja i mogućih pristupa te analizirati, izložiti i demonstrirati tehnike sortiranja uporabom genetskih algoritama. Napomena: (tema rezervira za Zoltan Balko Macsai)

Datum ocjene pismenog dijela diplomskog rada od strane mentora:	23.09.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	15.10.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	17.10.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 17.10.2024.

Ime i prezime Pristupnika:	Zoltan Balko Macsai
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D1264R, 07.10.2022.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Sortiranje objekata više atributnim kriterijima uporabom genetskih algoritama**

izrađen pod vodstvom mentora prof. dr. sc. Damir Blažević

i sumentora prof. dr. sc. Tomislav Keser

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

U današnjem svijetu, podaci postaju sve složeniji, postoji sve veća potreba za efikasnijim tehnikama sortiranja objekata po više kriterija od jednom. U mnogim područjima, uključujući financije, medicinu i energetiku, nije dovoljno samo sortirati objekte po jednom kriteriju, jer u stvarnom svijetu objekti često imaju više dimenzija i svojstava o kojima ovise. Na primjer, kod fotonaponskih panela, karakteristike kao što su napon i struja mogu značajno utjecati na performanse fotonaponskog sustava. Jedan od izazova sortiranja po više kriterija je pronalazak optimalne kombinacije koja zadovoljava sve zadane zahtjeve.

Genetski algoritmi su napredna metoda koja je pokazala veliki potencijal za rješavanje problema sortiranja u situacijama gdje postoji više atributnih kriterija. Oni se temelje na evolucijskim principima i koriste operacije kao što su selekcija, križanje i mutacija kako bi generirali niz mogućih rješenja i evoluirali prema najboljem mogućem rješenju. Zbog svoje fleksibilnosti i sposobnosti prilagodbe različitim problemima, genetski algoritmi su postali važan alat u optimizacijskim procesima.

Cilj ovog rada je istražiti primjenu genetskih algoritama za sortiranje objekata prema više kriterija. Kroz analizu i demonstraciju, pokazano je kako se genetski algoritmi mogu koristiti za optimizaciju problema sortiranja fotonaponskih panela, uzimajući u obzir njihove ključne karakteristike poput napona i struje. Na ovom konkretnom primjeru omogućeno je bolje razumijevanje potencijala genetskih algoritama u rješavanju problema višekriterijskog sortiranja. Također na konkretnom primjeru, sortiranja fotonaponskih panela, prikazano je kako jedan takav genetski algoritam smanjuje gubitke zbog nesklada fotonaponskih nizova i povećava maksimalnu proizvodnju električne energije.

U nastavku je opisan pregled područja gdje su opisana dva slična postupka sortiranja i grupiranja objekata prema njihovim različitim parametrima. Prvi postupak je korištenje genetskog algoritma za predviđanje kombinacija lijekova za kemoterapije. Dok drugi korištenje analitike karakteristika fotonaponskih panela različitih proizvođača za određivanje koji proizvođač ima najbolje fotonaponske panele U drugom poglavlju se ulazi u pozadinu kako funkcionira fotonaponski panel i od čega je napravljen. U ovom poglavlju istaknuta je važnost pravilnog grupiranja fotonaponskih ćelija i, u konačnici, fotonaponskih panela kako bi se maksimalno iskoristila njihova učinkovitost. Treće poglavlje ulazi u detalje matematičke

analize solarnih panela. U ovom poglavlju su formule koje su bitne za opis fotonaponskih panela te za izračun gubitaka zbog nesklada fotonaponskih panela. Nakon trećeg poglavlja, sljedeće poglavlje razmatra što je genetski algoritam i koje su njegove primjene u svijetu. U petom poglavlju se razrađuje problematika diplomskog rada, postavljaju se ciljevi i kreira se program s genetskim algoritmom za rješavanje konkretnog primjera, pronalaska grupiranih fotonaponskih panela s najmanje mogućim gubitkom snage zbog nesklada. Nakon razrade problema i kreiranja algoritma, u šestom poglavlju obavlja se analiza rezultata koji proizlazi iz genetskog algoritma i običnih sortiranja i grupiranja fotonaponskih panela, po njihovim karakteristikama koje su dobivene od proizvođača. Na kraju, u zaključku je napravljen osvrt na rezultate te kako ih je moguće unaprijediti budućim istraživanjima i poboljšanjima.

1.1 Zadatak diplomskog rada

Za zadani skup podataka i zadatak sortiranja po višestrukim kriterijima opisati problem sortiranja, napraviti pregled stanja i mogućih pristupa te analizirati, izložiti i demonstrirati tehnike sortiranja uporabom genetskih algoritama.

1.2 Područje problema i slična rješenja

Moderni alati pružaju lakše načine za pronalaženje optimalnih nizova objekata. Genetski algoritmi u posljednjih pedesetak godina bilježe značajan razvoj. Genetski algoritmi se primjenjuju i daju dobre rezultate u području učenja kod neuronskih mreža, pri traženju najkraćeg puta, strategiji igara, ali i za optimalno sortiranje podataka prema određenim parametrima sustava. Genetski algoritam pruža više od običnog sortiranja, koje je dostupno pomoću alata kao što su Microsoft Excel, ili jednostavnim pozivom metode *sort()* u programskim jezicima kao *Python*. Koristi se većinom za veće podatkovne skupove i za skupove koji ovise o više čimbenika. Jedan od tih primjera je u kemiji i biologiji.

Lamarckian Genetic Algorithm (LGA) je genetski algoritam koji je korišten u području kemoinformatike, za pronalaženje potencijalno novih spojeva lijekova koji se mogu povezivati s određenim receptorima. U ovom području je računalna problematika pretraživanja baza

podataka za potencijalne kandidate (spojeve lijekova) koji se mogu orijentirati u dobrom smjeru u kombinaciji s kemijskim spojem. Prije poznavanja ovog algoritma, ovakve zadatke bilo je moguće obavljati samo kroz orijentacijska ili kombinacijska pretraživanja, ali ne oba istovremeno. LGA koristi prednosti računalnog ubrzanja i kombiniranja globalnog pretraživanja genetskog algoritma s lokalnim pretraživanjem [1].

Također, u području energetike poznato je korištenje sortiranje fotonaponskih panela. Renomirani proizvođači fotonaponskih panela odavno nude sortirane fotonaponske panele odmah pri narudžbi, naravno uz veću cijenu. Naime takvi sortirani paneli su najčešće samo sortirani po jednom parametru i to je struja na maksimalnoj snazi.

Drugo istraživanje[2] pronalazi analitičke modele za sortiranje fotonaponskih panela, za razliku od pristupa u ovom diplomskom radu, ovi analitički modeli su se fokusirali na kriterije kao snaga, cijena ili masa fotonaponskih panela od različitih proizvođača fotonaponskih panela (*JA Solar, Shinefar, Canadian Solar, Amerisolar*). Svoju metodu su nazivali MEREC (engl. *The Method of Elimination and Removal of Criteria Effects*) metoda, jer je metoda radila eliminaciju i brisanje pomoću određenih kriterija. Proces obrade podataka ove metode je prvotno dodijeliti težinske faktore kriterijima na temelju njihovog utjecaja na izvedbu alternativa. Izvedba alternativa je mjera koja pokazuje koliko je svaka opcija dobra ili loša s obzirom na dane kriterije i kako se međusobno uspoređuju u procesu donošenja odluke. Iza ovoga procjenjuje važnosti svakog kriterija mjerenjem promjena u performansama kada se pojedini kriterij ukloni iz analize. Što je veći utjecaj uklanjanja nekog kriterija to je kriterij važniji za konačnu rang-listu alternativa. Rade se matrice odluke te se normaliziraju, što omogućava usporedbu podataka, bez obzira na jedinice mjerenja. Normalizirana matrica predstavlja izvedbu svake alternative u odnosu na svaki kriterij. Nakon svake izvedbe se radi procjena izvedbe korištenjem logaritamske mjere s jednakim težinskim faktorima za sve kriterije kako bi bila izračunata ukupna izvedba alternativa. Nakon toga se ponovno računa izvedba i svaki se kriterij pojedinačno uklanja. Nakon procjene učinka uklanjanja svakog kriterija određuju se konačne težine. Kriteriji s većim utjecajem na konačni rezultat dobivaju veće težine [2].

U drugom istraživanju korištena je i metoda SPOTIS (engl. *Stable Preference Ordering Towards Ideal Solution*), ova je višekriterijska metoda za donošenje odluka koja je dizajnirana kako bi se izbjegli problemi promjene redoslijeda rangiranja alternativa, što je česta slabost drugih metoda poput AHP (Analitički hijerarhijski proces) ili TOPSIS (Tehnika za određivanje

prioriteta prema sličnosti idealnih rješenja). Glavna prednost SPOTIS metode je stabilnost rangiranja i otpornost na revizije kada se uklone ili dodaju nove alternative. Ključne značajke SPOTIS metode su rangiranje prema idealnom rješenju, pri čemu se fokus stavlja na udaljenost svake alternative od unaprijed definiranog idealnog rješenja, te stabilnost rangiranja koja osigurava nepromijenjen redoslijed ostalih alternativa. Uporaba ograničenja je također bitna značajka SPOTIS metode jer zahtijeva definiranje minimalnih i maksimalnih granica za svaki kriterij. SPOTIS također ne koristi relativne usporedbe već samo radi usporedbe s idealnim rješenjem, što omogućava bržu obradu i pristranost kod ocjenjivanja i čini postupak stabilnijim.

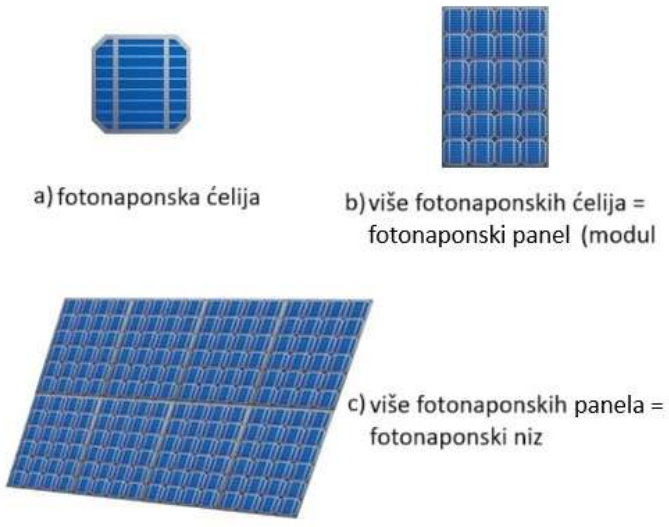
Kombinacijom MEREC metode sa SPOTIS metodom, donositelji odluka mogu dobiti precizno rangiranje alternativa koje odražava najvažnije kriterije koji utječu na performanse. Istraživanje primjenom MEREC metode rezultiralo je rangiranjem panela proizvođača *Shinefar* rangiran kao najboljeg među analiziranim markama fotonaponskih panela [2].

2. STRUČNA POZADINA

Gubitci zbog nesklada fotonaponskih nizova izazivaju probleme u proizvodnji energije fotonaponskih panela. Mnogi znanstveni i stručni radovi obrađuju ove probleme, u cilju smanjivanja gubitaka nastalih zbog nesklada fotonaponskih nizova, ali ovaj problem je neizbježan. No postoje načini ublažavanja gubitaka. Pažljivim odabirom i selekcijom, odnosno sortiranjem po odgovarajućim kriterijima moguće je reducirati gubitke koji nastaju zbog neusklađenosti fotonaponskih panela.

2.1 Fotonaponski panel

Fotonaponski (PV) moduli su poluvodički uređaji koji pretvaraju sunčevu energiju u električnu energiju, fotonaponskim ćelijama. Jedan modul se najčešće sastoji od više fotonaponskih ćelija, te se više modula povezuje u jedan PV panel, iza čega se isti paneli povezuju na različite načine u nizove panela, (engl. *strings of panels*). Povezani nizovi fotonaponskih panela su serijski spojeni paneli. Skupina više tako povezanih nizova fotonaponskih panela čini fotonaponski niz (Slika 2.1) [3].

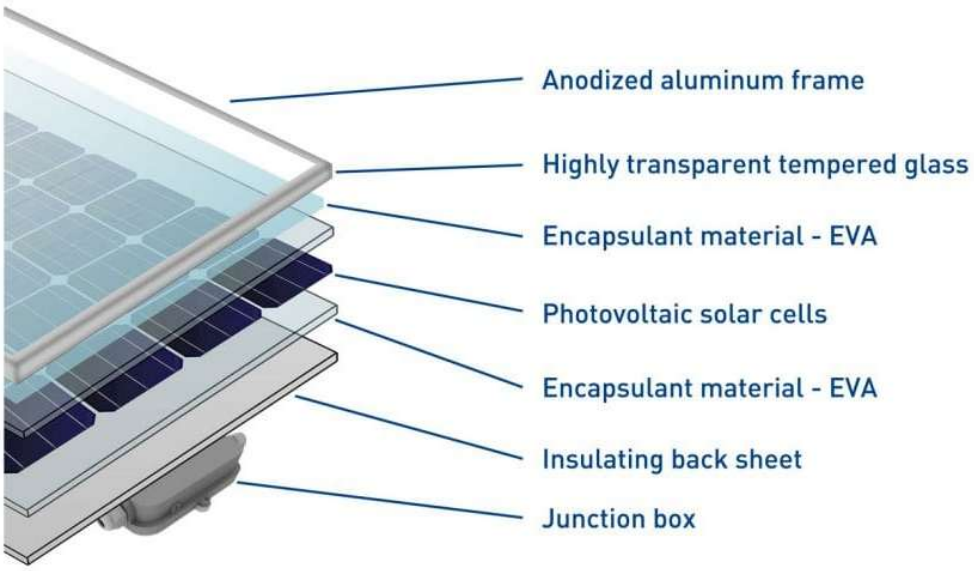


Slika 2.1. a) fotonaponska ćelija b) fotonaponski panel c) fotonaponski niz [3]

Fotonaponska ćelija sastoji se od nekoliko slojeva (slika 2.2), od kojih svaki ima specifičnu svrhu u procesu pretvorbe sunčeve energije. Glavne komponente fotonaponske ćelije su:

- Podloga
- Upijajući sloj
- Prozirni EVA sloj
- Kontaktni sloj

Podloga je osnovni materijal na koji se nanose ostali tanki slojevi fotonaponskog panela. Upijajući sloj je sloj koji obavlja funkciju upijanja sunčeve svjetlosti. Ovaj sloj stvara parove elektron-šupljina i omogućuje protok električne energije, a njegov izbor varira o vrsti fotonaponskog panela. Neke od opcija materijala za upijajući sloj su amorfni silicij ili kadmijev telurid. Nakon upijajućeg sloja, dolazi prozirni EVA (engl. Etilen Vinil Acetat) sloj, njegova uloga je olakšati kretanje elektrona unutar panela, a istovremeno omogućiti propuštanje sunčeve svjetlost. Obično se sastoji od materijala kao što su indijev kositar oksid ili cinkov oksid. Zadnji sloj je kontaktni sloj, on je smješten ispod sloja upijanja i njegova uloga je prikupljanje generiranih elektrona i usmjeravanje istih prema suprotnim krajevima ćelije. Obično je kontaktni sloj napravljen od aluminija [4].



Slika 2.2 Slojevi fotonaponskog panela, prijevod redom: okvir od aluminija, prozirno kaljeno staklo, EVA, fotonaponske ćelije, podloga, razvodna kutija [5]

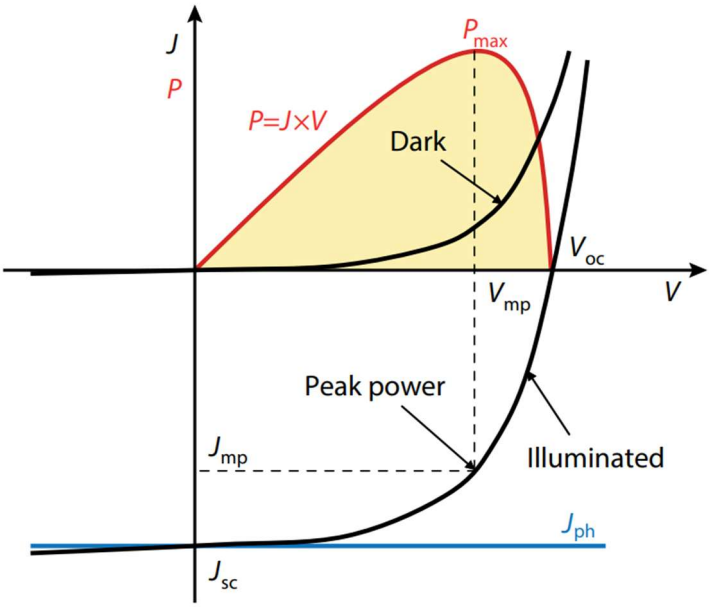
Princip rada fotonaponske ćelije temelji se na načelu fotonaponskog efekta. Sastoji se od dva poluvodička sloja, najčešće kristala silicija. U donji sloj se obično dodaje trovalentni bor

čime se stvara višak pozitivnih naboja (P sloj), a gornji najčešće peterovalentni fosfor koji stvara višak elektrona, tj. višak negativnih naboja (N sloj). Kada fotoni padnu na ćeliju, dolazi do razdvajanja nositelja naboja unutar poluvodičkog materijala. Oslobađaju se elektroni iz oba sloja te se elektroni nastoje gibati prema pozitivnoj ili negativnoj elektrodi. Kada se na vanjske kontakte spoji potrošač, kroz njega poteče struja, što omogućava pretvorbu sunčeve energije u drugi oblik energije, odnosno električnu energiju [6].

U opisanom procesu, ne uspijevaju sve zrake svjetlosti doprijeti do poluvodičkog materijala. Neke se odbijaju od ćelije, dok se druge pretvaraju u toplinsku energiju. Samo zrake odgovarajućih valnih duljina prolaze kroz poluvodički materijal. Na takve gubitke nije moguće utjecati, stoga se oni naknadno uračunavaju [7]. Ipak, zbog različitih čimbenika, poput razlika u proizvodnji, temperaturne varijacije, starenja modula i djelomičnog zasjenjivanja panela, dolazi do nesklada u karakteristikama među ćelijama unutar istog panela ili između panela u nizu. Ovaj nesklad uzrokuje značajne gubitke energije u PV sustavima. Kada karakteristike neke ćelije unutar PV niza variraju, pojedine ćelije mogu raditi slabije od očekivanog.

2.2 Karakteristike fotonaponskih panela

Fotonaponski paneli, kao temeljni elementi fotonaponskih sustava, procjenjuju se prema nekoliko ključnih parametara koji određuju njihovu učinkovitost u pretvorbi sunčeve energije u električnu energiju. Ovi parametri omogućuju usporedbu performansi različitih fotonaponskih modula i važni su za optimizaciju rada fotonaponskih sustava. Glavni parametri koji se koriste za karakteriziranje rada fotonaponske ćelije su vršna snaga (P_{max}), gustoća struje kratkog spoja (J_{sc}), napon otvorenog kruga (V_{oc}), i faktor punjenja (FF) [8, stranica 101], vidljivi na slici 2.3.



Slika 2.3 J-V karakteristika p-n spoja u mraku i na svjetlosti [8, stranica 99]

2.2.1 Vršna snaga (P_{max})

Vršna snaga (P_{max}) je maksimalna snaga koju fotonaponski panel može proizvesti pod standardnim testnim uvjetima (STC), što uključuje sunčevo zračenje od 1000 W/m^2, temperaturu ćelije od 25°C i standardizirani fotonaponski spektar AM1.5. Vršna snaga je ključni parametar jer predstavlja optimalnu snagu izlaza modula. Matematički, vršna snaga može se izraziti formulom (2-1).

$$P_{max} = V_{mp} \times I_{mp} \tag{2-1}$$

Gdje je:

V_{mp} – napon pri maksimalnoj snazi,

I_{mp} – struja pri maksimalnoj snazi.

2.2.2 Gustoća struje kratkog spoja (J_{sc})

Gustoća struje kratkog spoja (J_{sc}) predstavlja količinu struje koja teče kroz fotonaponsku ćeliju kada su izlazi kratko spojeni tj. kada je napon na izlazima jednak nuli. J_{sc} ovisi o spektru upadne svjetlosti i fotonskom fluksu na površini fotonaponske ćelije. Gustoća struje kratkog spoja također ovisi o optičkim svojstvima ćelije kao što su upijanje sunčeve energije u

upijajućem sloju i refleksija. U idealnom slučaju, J_{sc} je jednak foto-generiranoj gustoći struje (J_{ph}) [8, stranice 101-102]. J_{sc} se matematički izražava formulom (2-2).

$$J_{sc} = \frac{I_{sc}}{A} \tag{2-2}$$

Gdje je:

I_{sc} – struja kratkog spoja,

A – površina fotonaponske ćelije.

2.2.3 Napon otvorenog kruga (V_{oc})

Napon otvorenog strujnog kruga (Voc) je maksimalni napon koji se može izmjeriti na izlaznim priključcima fotonaponske ćelije kada nema opterećenja, odnosno kada je struja kroz ćeliju nula. Voc odgovara naponskoj točki pri kojoj struja zasićenja kompenzira fotostruju te ovisi o foto-generiranoj gustoći struje i zasićenoj struji. Napon otvorenog strujnog kruga za kristalno silicijske fotonaponske ćelije obično je u rasponu od 600 do 720 mV, dok je za fotonaponski panel taj raspon od 30 do 42 V, pod standardnim uvjetima osvjetljenja [8, stranica 102]. Napon otvorenog kruga može se matematički izraziti formulom (2-3).

$$V_{oc} = \frac{kT}{q} ln\left(\frac{J_{sc}}{J_0} + 1\right) \tag{2-3}$$

Gdje je:

k – Boltzmannova konstanta

T – temperatura ćelije u Kelvinima

q – električni naboj elektrona

J_{sc} – gustoća struje kratkog spoja

J_0 – inverzna zasićenost struje fotonaponske ćelije

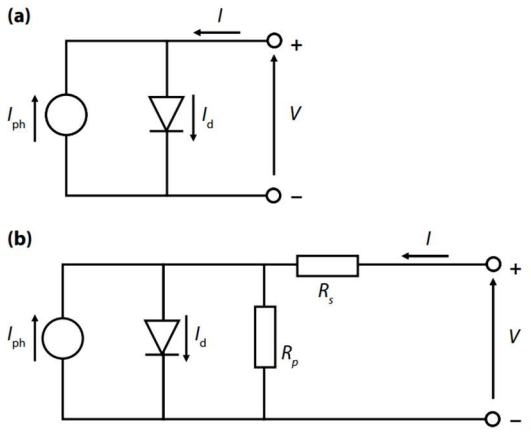
2.2.4 Faktor punjenja (FF)

Faktor punjenja (FF) je omjer maksimalne izlazne snage P_{max} i umnoška V_{oc} i J_{sc}. Faktor punjenja može se aproksimirati kao funkcija V_{oc} (slika 2.3) i utječe na njega, kao i serijski i

paralelni otpori. Za idealne fotonaponske ćelije faktor punjenja je obično između 0.75 i 0.85, ali u stvarnim uvjetima FF je niži zbog različitih gubitaka unutar ćelije [8, stranica 102-103]. FF se može matematički izraziti formulom (2-4).

$$FF = \frac{P_{max}}{V_{oc} \times I_{sc}} \tag{2-4}$$

Razumijevanje ovih parametara je ključno za dizajn i optimizaciju rasporeda fotonaponskih panela. Na slici 2.4 je prikazan ekvivalentni strujni krug fotonaponske ćelije, uključujući serijske i paralelne otpore. Ovaj strujni krug pomaže u razumijevanju utjecaja ovih parametara na performanse ćelije.



Slika 2.4 a) idealni strujni krug fotonaponske ćelije, b) strujni krug sa serijskim otporima R_s i otporom shunta R_p

2.3 Grupiranje fotonaponskih panela

Prije grupiranja fotonaponskih panela u fotonaponske nizove potreban je pažljiv odabir pojedinačnih fotonaponskih ćelija unutar jednog fotonaponskog panela prema njihovim karakteristikama. Postoje tri ključna pravila pri dizajniranju fotonaponskih panela koja se moraju uzeti u obzir kako bi se postigla maksimalna efikasnost i smanjili gubitci [8, stranica 126].

Ta tri pravila su:

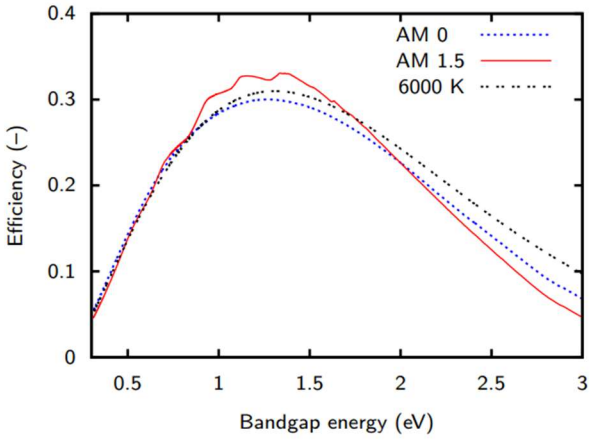
- Iskorištavanje energije iz energetskog procjepa (engl. *Bandgap utilisation*)
- Spektralno iskorištavanje (engl. *Spectral utilisation*)
- Zarobljavanje svjetla (engl. *Trapping light*)

2.3.1 Iskorištavanje energetskog procjepa (engl. Bandgap utilisation)

Fotonaponske ćelije su dizajnirane za maksimalno iskorištavanje energiju fotona čije energije prelaze energetski procjep materijala iz kojeg su izrađene. Energija iz energetskog procjepa se koristi za generiranje elektrona i šupljina koji doprinose proizvodnji električne energije. Korištenje materijala s optimalnim energetskim procjepom, kao što su silicij, galijev arsenid i kadmijev telurid, osigurava maksimalnu učinkovitost konverzije. Međutim, napon otvorenog kruga (V_{oc}) uvijek je manji od napona koji odgovara energetskom procjepu zbog mehanizama rekombinacije nositelja naboja, što predstavlja ograničenje korisnosti energetskog procjepa [8, stranice 126-128].

2.3.2 Spektralno iskorištavanje (engl. Spectral utilisation)

Spektralno iskorištavanje odnosi se na sposobnost fotonaponske ćelije u iskorištavanju što šireg raspona sunčevog spektra. Optimalni energetski procjep za jednostavne fotonaponske ćelije ograničen je Shockley-Queisserovom granicom učinkovitosti, kao što je vidljivo na slici 2.5. Uz pomoć koncepta ulančavanja fotonaponskih ćelija, koje sadrže materijale različitih energetskih procjepa, moguće je značajno smanjiti višak energije i poboljšati spektralno iskorištavanje [8, stranice 128-129].



Slika 2.5 Grafički prikaz Schockley-Queisserove granice učinkovitosti za različite spektre, kao što su 6000K, AM0 i AM1.5 x-os pokazuje učinkovitost, a y-os pokazuje energiju energetskog procjepa [8, stranica 121]

2.3.3 Zarobljavanje svjetlosti (engl. Trapping light)

U zadnjem pravilu, cilj je apsorbirati svu svjetlost koja pada na fotonaponske ćeliju u upijajući sloj. Prema Lambert-Beeraovom zakonu svjetlost prolazi kroz medij, a intenzitet

svjetlosti eksponencijalno se smanjuje kako prolazi kroz upijajući sloj [9]. U kontekstu fotonaponskih ćelija, ovaj zakon je od posebne važnosti jer pomaže u razumijevanju kako svjetlost, koja je potrebna za stvaranje električne energije, prolazi i apsorbira se unutar fotonaponske ćelije. Za maksimalnu apsorpciju, koriste se tehnike koje uključuju korištenjem upijača debljih upijajućih slojeva ili materijala s visokim koeficijentima upijanja [8, stranice 129-131].

Labert-Beerov zakon opisuje eksponencijalno smanjenje intenziteta svjetlosti i može se prikazati formulom:

$$I(z) = I_0 \times e^{(-\alpha z)} \tag{2-5}$$

Gdje je:

α – koeficijent apsorpcije

z – dubina ili udaljenost unutar apsorbirajućeg medija

I_0 – početni intenzitet svjetlosti (na z = 0)

U ovome poglavlju opisano je od čega se sastoji fotonaponski panel, odnosno opisan je njegov najmanji dio, fotonaponska ćelija. Dok se u sljedećem poglavlju prelazi na detaljan opis matematičke analize modela fotonaponskog panela, koji se koriste za procjenu gubitaka nastalih zbog nesklada u fotonaponskim panelima.

3. MATEMATIČKA ANALIZA

U ovom poglavlju analiziran je matematički modeli koji se koriste za procjenu gubitaka nastalih zbog nesklada (engl. *mismatch losses*) u fotonaponskim panelima.

3.1 Matematički prikaz fotonaponskog panela

Fotonaponski paneli su sastavljeni od pojedinačnih fotonaponskih ćelija koje su povezane u seriju ili paralelu kako bi se postigla željena snaga. Međutim zbog različitih faktora kao što su proizvodne tolerancije, degradacija uslijed starenja i okolišni stresovi (npr. sjene), dolazi do gubitaka zbog nesklada. Kako bi se odredio taj gubitak zbog nesklada trebaju se proračunati i izvući varijable kojima se proračunava gubitak zbog nesklada.

Prema [7], [10] i [11] bitne formule su sljedeće:

3.1.1 Jednadžba strujno-naponske (I-V) karakteristike fotonaponskog panela

Jednadžba za određivanje strujno-naponske (I-V) karakteristike prikazana je formulom (3-1).

$$I = I_{pv} - I_0 \times \left(e^{\frac{V+IR_s}{AkTc}} - 1 \right) - \frac{V + IR_s}{R_{sh}} \tag{3-1}$$

Gdje je:

I – struja kroz ćeliju

V – napon na ćeliji

I_{pv} – fotostruja

I_0 – struja zasićenja diode fotonaponske ćelije

R_s – serijski otpor fotonaponske ćelije

R_{sh} – paralelni (shunt) otpor fotonaponske ćelije

e – elementarni naboj (1.6022×10^{-19} C)

A – faktor idealnosti diode

k – Boltzmannova konstanta (1.3854×10^{-23} J/K)

T_c – radna temperatura fotonaponske ćelije [11]

3.1.2 Izračunavanje serijskog otpora fotonaponskog panela R_s

Izračun serijskog otpora fotonaponskog panela (R_s) prikazan je formulom (3-2).

$$R_s = \frac{V_{oc}}{I_{mp}} + \frac{v_t}{I_{mp}} \ln\left(\frac{v_t}{v_t + V_{mp}}\right) - \frac{V_{mp}}{I_{mp}} \tag{3-2}$$

$$v_t = \frac{kATN}{e} \tag{3-3}$$

Gdje je:

N – broj ćelija u fotonaponskom panelu

T – temperatura u Kelvinima

v_t – toplinski probojni napon [11]

3.1.3 Izračunavanje paralelnog (shunt) otpora fotonaponskog panela R_{sh}

Izračun paralelnog (shunt) otpora je prikazan formulom (3-4).

$$R_{sh} = \frac{V_{mp}(V_{mp} + R_{sh}V_{mp})}{V_{mp}I_{pv} - V_{mp}I_0\left(e^{\frac{V_{mp}+R_sI_{mp}}{V_t}} - 1\right) - P_{max}} \tag{3-4}$$

3.1.4 Određivanje omjera idealne snage podniza i idealne snage svih podnizova

Za izračun gubitka snage zbog nesklada potrebno je izračunati omjer snage jednog podniza i srednje vrijednosti snage svih podnizova, ova vrijednost se označava s ω_q i prikazana je forumulom (3-5).

$$\omega_q = \frac{P_{ideal,q}}{\overline{P}} \tag{3-5}$$

Gdje je:

q – podniz

3.1.5 Izračunavanje karakterističnog faktora ćelije C

Prosječni faktor punjenja za fotonaponske panele u konfiguraciji je prikazan pomoću karakterističnog faktora ćelije C, na formuli (3-7), za određivanje faktora C' potrebno je poznavanje svih faktora punjenja fotonaponskih panela, koji ako već nisu poznati od proizvođača se određuju formulom (3-6).

$$\overline{FF} = \frac{(C')^2}{(1+C')[C' + \ln(1+C')]} \tag{3-6}$$

$$FF = \frac{I_{mp}V_{mp}}{I_{sc}V_{oc}} \tag{3-7}$$

3.1.6 Određivanje varijanci struje i napona

U konfiguraciji gdje je više podnizova povezano paralelno, potrebno je izračunati varijance struje i napona fotonaponskih panela, varijanca struje se određuje formulom (3-8), a varijanca napona se određuje formulom (3-9).

$$\sigma_\eta^2 = \left(\frac{\sigma_{I_{mp}}}{\overline{I_{mp}}}\right)^2 \tag{3-8}$$

$$\sigma_\xi^2 = \left(\frac{\sigma_{V_{mp}}}{\overline{V_{mp}}}\right)^2 \tag{3-9}$$

3.1.7 Opća formula za mrežu blokova

Gubitak snage zbog nesklada fotonaponskih panela u mreži paralelnih M podnizova, gdje svaki podniz sadrži L fotonaponskih panela određen je formulom, koja kombinira formule navedene u ovom poglavlju i prikazana je formulom (3-10).

$$E[\Delta P] = \left(1 - \frac{1}{L}\right)\frac{1}{M}\left[\sum_{q=1}^{M} \omega_q \frac{C_q' + 2}{2} \times \sigma_{\eta,q}^2\right] + \frac{C' + 2}{2}\frac{\sigma_\xi^2}{L}\left[1 - \frac{1}{M}\right] \quad (3\text{-}10)$$

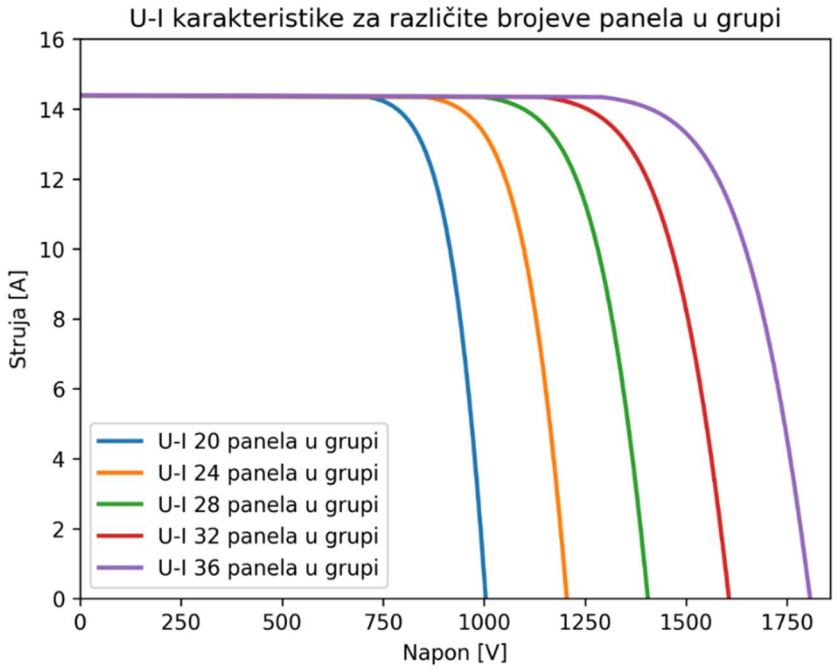
Gdje je:

L – broj ćelija u seriji

N – broj blokova [7]

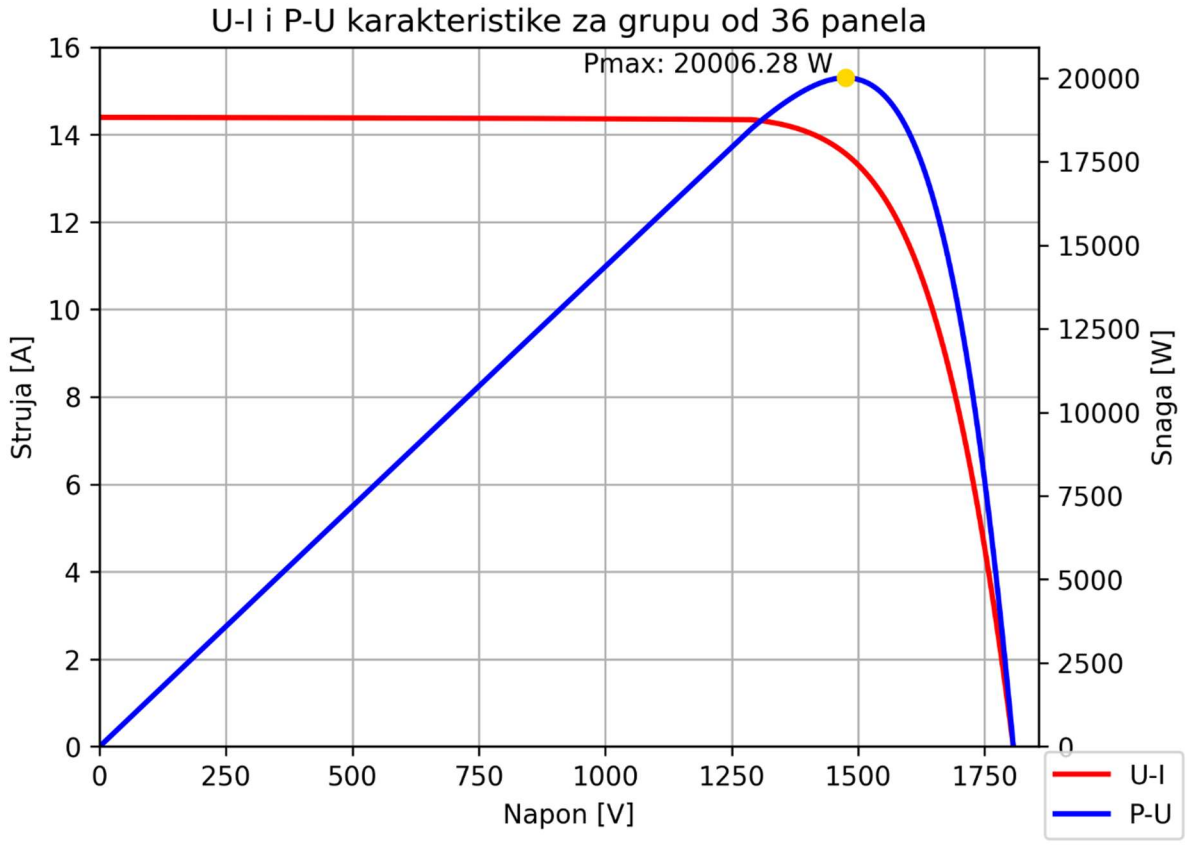
3.2 Grafička analiza fotonaponskog panela

Za učinkovitu analizu i optimizaciju rasporeda fotonaponskih panela, važno je uzeti u obzir sve ove matematičke formule. Razumijevanje formula je prvi dio do razvijanja algoritma, ali uz formule popratno treba proučiti još grafičke modele i karakteristike fotonaponskih panela. Graf I-V krivulje, sa slike 3.1, prikazuje karakterističnu strujno-naponsku (I-V) krivulju fotonaponskog niza panela. Krivulja prikazuje kako struja koja teče kroz panele eksponencijalno opada s porastom napona na terminalima panela. Najvažnije točke na ovoj krivulji su struja kratkog spoja (I_{sc}) i napon otvorenog kruga (V_{oc}). Ova krivulja je osnova za razumijevanje kako fotonaponski paneli rade i kako se gubitci zbog nesklada mogu dogoditi kada se paneli različitih karakteristika povežu zajedno [12].



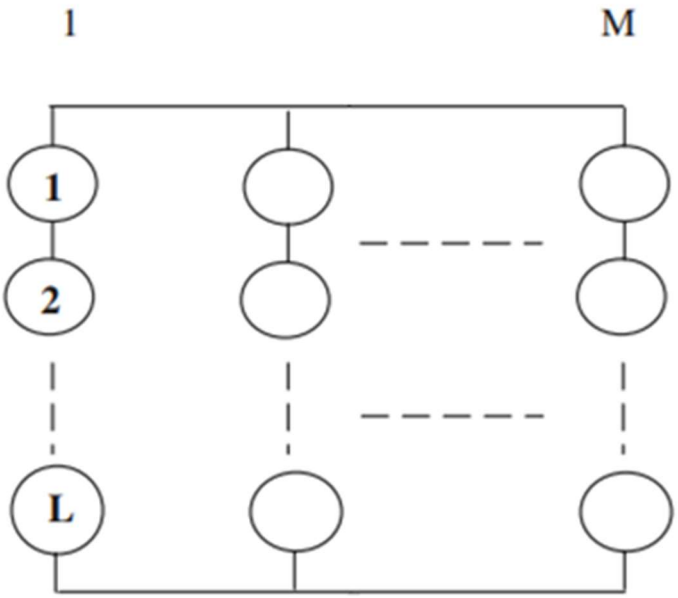
Slika 3.1 U-I karakteristike za različite brojeve panela u grupi (25°C)

Graf na slici 3.2 prikazuje U-I karakteristiku zajedno s P-U karakteristikom grupe od 36 fotonaponskih panela u seriji. Vidljiva je maksimalna snaga i kako snaga varira u odnosu na napon i struju fotonaponskog niza.



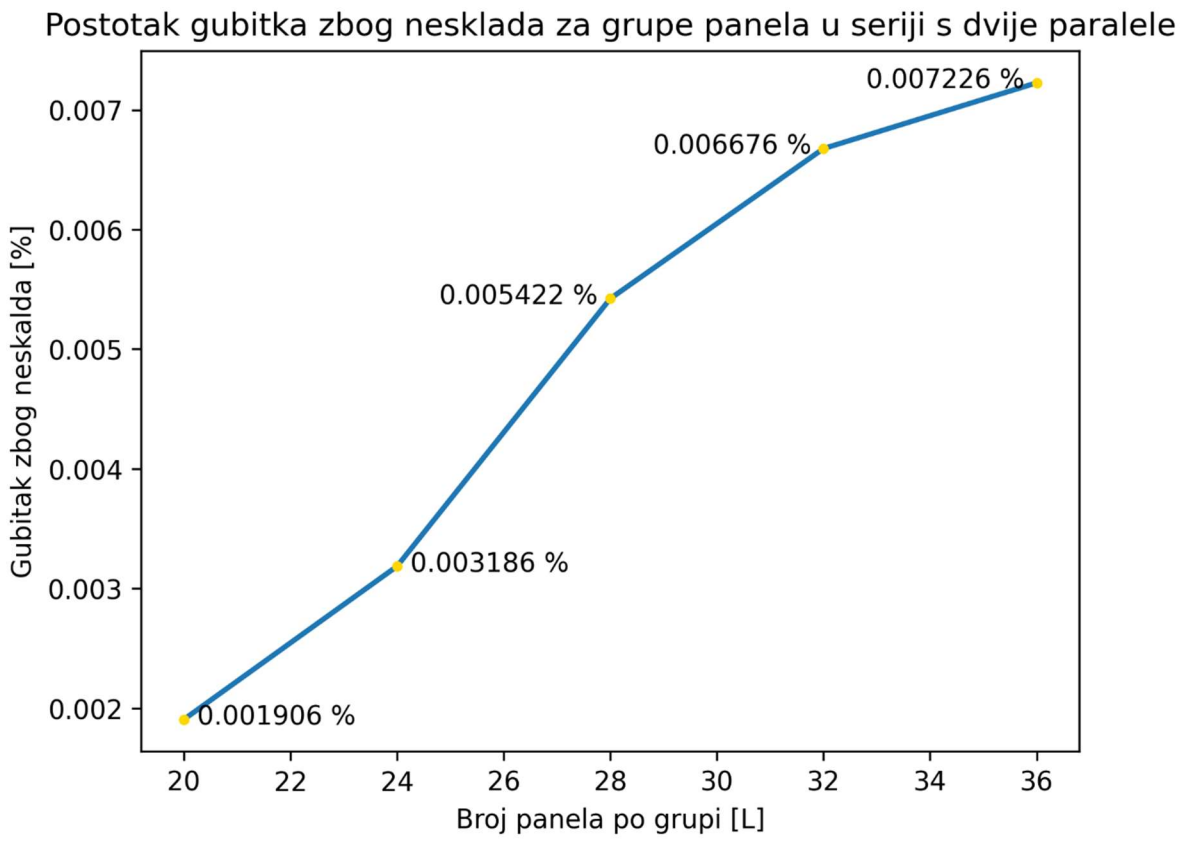
Slika 3-2 U-I i P-U karakteristike za grupu od 36 fotonaponskih panela

Mreža fotonaponskih panela od M paralelnih podnizova, gdje se svaki podniz sastoji od L fotonaponskih panela povezanih u seriji, prikazan je na slici 3.3. Prikazom ovakve mreže, naglašava se kako različiti načini grupiranja panela mogu utjecati na ukupne gubitke zbog nesklada u proizvodnji energije. Posebno je važno primijetiti kako konfiguracija mreže može utjecati na ukupnu izlaznu snagu zbog razlika u karakteristikama pojedinih panela ili modula unutar mreže [10].



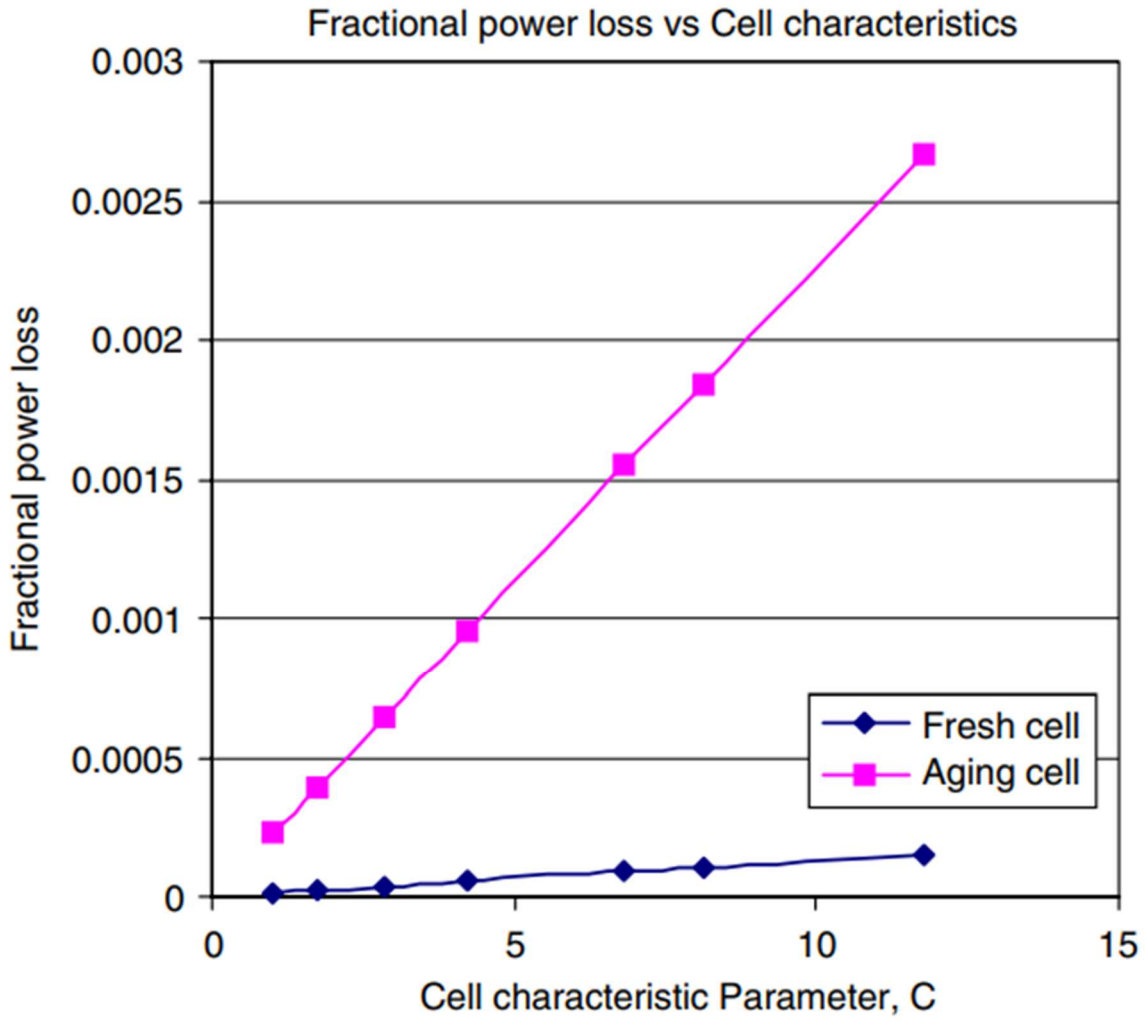
Slika 3.3 Shema mreže fotonaponskih panela od M paralelnih podnizova, te svaki podniz se sastoji od L fotonaponskih panela povezanih u seriji [10, Fig. 1]

Graf gubitka snage zbog nesklada u funkciji broja panela povezanih u seriju, na slici 3.4 prikazuje kako se gubitak snage mijenja s obzirom na broj panela u seriji. Porast gubitaka zbog nesklada vidljiv je povećanjem broja panela u seriji, posebno kada su karakteristike panela unutar serije značajno različite. To je zbog toga što struja kroz serijski spojenu mrežu ne može biti veća od struje ćelije s najnižom strujom, što uzrokuje smanjenje ukupne snage mreže.



Slika 3.4 Graf gubitka snage zbog nesklada fotonaponskih panela povezanih u seriju [10]

Drugi graf, na slici 3.5, pokazuje kako se gubitak snage zbog nesklada mijenja ovisno o parametru karakteristike ćelije (C). Parametar C povezan je s faktorom punjenja (FF) ćelije i varira ovisno o materijalu ćelije, vrsti spoja i kvaliteti izrade [10]. Graf jasno pokazuje kako varijacije u parametru C mogu značajno utjecati na ukupne gubitke zbog nesklada, naglašavajući važnost pažljivog dizajna i optimizaciju rasporeda ćelija kako bi se postigli minimalni gubitci. Također, graf na slici 3.5 pokazuje kako utječe starost ćelije na gubitke energije u ćelijama.



Slika 3.5 Graf gubitka snage zbog nesklada ovisno o parametru karakteristike ćelije, C, na x-osi je karakteristični parametar ćelije C, na y-osi gubitak snage [10]

Razumijevanje kako različiti parametri utječu na ukupne gubitke snage zbog nesklada omogućavaju poboljšanje dizajna i rasporeda fotonaponskih panela, čime se maksimizira učinkovitost i smanjuju gubitci energije.

4. GENETSKI ALGORITAMI

U ovom poglavlju istražuju se genetski algoritmi, njihova primjena u optimizaciji fotonaponskih sustava te primjeri koji ilustriraju njihovu upotrebu u svijetu.

Genetski algoritmi (GA) su stohastičke tehnike optimizacije temeljene na modeliranju prirodne evolucije. Ideja genetskih algoritama predložena je od strane Johna Hollanda još u ranim sedamdesetima. John Holland je prvi predložio ovaj koncept za optimizaciju kompleksnih problema korištenjem principe darvinističke selekcije i Mendelove genetike [13]. Genetski algoritmi su posebno korisni u situacijama gdje su tradicionalne metode optimizacije nepraktične zbog nelinearnosti, diskontinuiteta ili višemodalnih (višedimenzijskih) problema.

4.1 Općenito o genetskim algoritmima

Genetski algoritmi se sastoje od skupa potencijalnih rješenja problema, zvanih populacija, koja evoluira kroz nekoliko generacija prema sve boljim rješenjima. U svakoj generaciji, algoritam primjenjuje operacije kao što su selekcija, križanje (reprodukcija) i mutacija kako bi stvorio novu generaciju rješenja koja bi trebala biti bolja od prethodne. Osnovni cilj genetskog algoritma je pretraživanje velikog prostora rješenja s ciljem pronalaženja optimalnog ili blizu optimalnog rješenja kroz simulaciju prirodne evolucije.

Genetski algoritmi se često primjenjuju u optimizacijskim problemima gdje tradicionalne metode pretraživanja (kao što su gradijentni postupci) mogu zapeti u lokalnim optimumima. U genetskim algoritmima, populacija rješenja kontinuirano se diversificira zahvaljujući mutacijama i križanjima, što omogućuje istraživanje globalnog prostora rješenja i pronalaženje globalnih optimuma [14].

Osnovne komponente genetskog algoritma prema [14] uključuju:

1. **Inicijalizacija populacije:** Populacija predstavlja skup potencijalnih rješenja koja se nasumično inicijaliziraju ili korištenjem neki heuristički pristup. Veličina populacije značajno utječe na performanse algoritma - manja populacija može dovesti do brže konvergencije, ali i većom vjerojatnošću zaglavljivanja algoritma u

lokalnom optimumu. S druge strane, veća populacija omogućuje širi pretraživački prostor, ali zahtijeva više računalnih resursa i vremena.

2. **Kodiranje kromosoma:** Kromosomi se mogu kodirati na različite načine ovisno o prirodi problema. Najčešće se koriste binarni nizovi (npr. nizovi nula i jedinica) koji predstavljaju određene karakteristike ili varijable rješenja. U kontekstu fotonaponskih panela, kromosomi mogu sadržavati podatke o kutu nagiba panela, njihovoj orijentaciji prema suncu, rasporedu i povezivanju. Realni brojčani nizovi također se koriste kada problem zahtijeva kontinuirane vrijednosti.
3. **Funkcija dobrote (engl. *fitness function*):** Funkcija dobrote procjenjuje kvalitetu svakog rješenja u populaciji. Za genetski algoritam, definiranje dobre funkcije dobrote ključno je jer ona usmjerava evolucijski proces prema najboljim rješenjima. Na primjer, u optimizaciji rasporeda fotonaponskih panela, funkcija dobrote može biti ukupna proizvedena energija tijekom dana, ili omjer između proizvedene energije i gubitaka zbog nesklada. Precizna definicija funkcije dobrote može značajno utjecati na uspješnost genetskog algoritma.
4. **Selekcija (engl. *selection*):** Selekcija je proces odabira kromosoma iz populacije za sudjelovanje u stvaranju nove generacije. Kromosomi s većim faktorom dobrote imaju veću vjerojatnost biti odabrani za reprodukciju. Različite metode selekcije uključuju:
    - **Jednostavna selekcija:** Svaki kromosom ima šansu biti odabran proporcionalno svojoj vrijednosti dobrote.
    - **Turnirska selekcija:** Odabire se skupina kromosoma nasumično, a najbolji iz te skupine se koristi za reprodukciju.
    - **Eliminacijska selekcija:** U jednom koraku se uzima najbolji iz prijašnje selekcije i generira se nova jednostavna selekcija, te se uzima bolja od dvije.
5. **Križanje (engl. *crossover*):** Križanje je proces kombiniranja genetskog materijala dvaju ili više roditeljskih kromosoma kako bi se stvorilo jedno ili više potomaka. Ovaj proces imitira biološku reprodukciju, gdje se geni miješaju kako bi se stvorila nova rješenja. U genetskim algoritmima najčešće se koriste metode poput:
    - **Jednostavno križanje:** Jedna točka križanja se odabire i genetski materijal se dijeli između roditelja na toj točki.
    - **Križanje s dvije točke prekida:** Odabiru se dvije točke križanja, između kojih se genetski materijal mijenja.

- **Uniformno križanje:** Svaki gen potomka nasumično se bira iz roditeljskih kromosoma, uz zadržavanje genetičke varijabilnosti.

6. **Mutacija:** Mutacija nasumično mijenja jedan ili više gena unutar kromosoma kako bi se uvele nove varijante u populaciju. Mutacija sprječava preranu konvergenciju algoritma na lokalni optimum i potiče istraživanje novih dijelova prostora rješenja. Stopa mutacije obično je mala kako bi se očuvala stabilnost rješenja, ali dovoljno visoka kako bi pridonijela genetičkoj raznolikosti.
7. **Zaustavljanje algoritma:** Genetski algoritam se obično zaustavlja nakon unaprijed definiranog broja generacija ili kada je postignuto zadovoljavajuće rješenje. Mogući kriteriji za zaustavljanje uključuju postizanje ciljane vrijednosti dobrote, nedostatak poboljšanja nakon određenog broja generacija, ili iscrpljivanje resursa poput vremena ili računalne snage.

4.2 Primjena genetskih algoritama

Primjena genetskih algoritama je vidljiva u različitim područjima, zahvaljujući svojoj sposobnosti rješavanja složenih optimizacijskih problema koji su neprikladni za tradicionalne metode. Fleksibilnost genetskih algoritama i njihova prilagodljivost čine ih idealnim za primjene gdje su problematični nelinearnost, diskontinuitet te gdje je problematika višedimenzionalna i složenije strukture.

U logistici i industriji, genetski algoritmi se koriste za optimizaciju rasporeda resursa, kao što su skladišta, vozni parkovi i distribucijske mreže. Algoritam može pomoći u pronalaženju optimalnog rasporeda koji minimizira troškove transporta, vrijeme isporuke i potrošnju goriva, uzimajući u obzir ograničenja kao što su kapaciteti vozila i skladišta te radno vrijeme vozača i drugi i slično. Genetski algoritmi našli su svrhu i u medicini gdje se koriste za optimizaciju liječenja raka, posebice u optimizaciji kemoterapijskih protokola. Kromosomi u ovom slučaju predstavljaju različite doze i rasporede primjene lijekova. Algoritam se koristi za prilagodbu protokola liječenja kako bi se maksimalizirala učinkovitost liječenja (npr. smanjenje tumora) uz istovremeno smanjenje nuspojava za pacijenta [13].

Svrha genetskih algoritama u području energetike nije nepoznati pojam. Jedan od primjera primjene genetskih algoritama u energetici je optimizacija rasporeda fotonaponskih panela

kako bi se maksimizirala proizvodnja električne energije i smanjili gubitci zbog nesklada fotonaponskih ćelija. U ovom slučaju, genetski algoritmi mogu optimizirati kutove nagiba, orijentaciju i raspored panela u velikim fotonaponskim parkovima, uzimajući u obzir faktore kao što su lokalne vremenske neprilike, sjene i raspoloživost površine.

Genetski algoritmi su, dakle, vrlo svestrani alati koji se mogu prilagoditi širokom spektru primjena, od medicine do financija i logistike. Sposobnost genetskih algoritama za učinkovito pretraživanje velikih prostora rješenja i prilagodbu kompleksnim problemima čini ih idealnim za raznolike primjene.

5. IZRADA GENETSKOG ALGORITMA

U ovom poglavlju razrađuje se postupak izrade genetskog algoritma koji se koristi za optimizaciju rasporeda fotonaponskih panela u cilju maksimalne proizvodnje energije i smanjenja gubitka zbog nesklada. Genetski algoritam detaljno je opisan kroz razradu problema, definiranje funkcije dobrote, te razvoj i implementaciju samog algoritma, uključujući sve popratne funkcionalnosti koje omogućuju njegovo pokretanje i evaluaciju rezultata.

5.1 Razrada problema

U prethodnim poglavljima definirani su matematički modeli fotonaponskih ćelija i predstavljena je problematika gubitaka zbog nesklada u njihovom radu. Ključni izazov je pronaći optimalni raspored i konfiguraciju fotonaponskih panela u sustavu koji minimizira gubitke energije zbog nesklada, dok se istovremeno maksimizira ukupna proizvodnja električne energije.

Genetski algoritam ovdje se koristi za pronalaženje najboljeg rješenja u velikom i složenom prostoru rješenja. Problem se sastoji u odabiru i grupiranju panela kako bi se optimizirao radni učinak, uzimajući u obzir varijabilnost u karakteristikama panela kao što su maksimalna snaga (P_{max}), napon otvorenog kruga (V_{oc}), struja kratkog spoja (I_{sc}), i faktor punjenja (FF) [10].

Stoga treba definirati ciljeve:

Cilj 1: Maksimizacija proizvodnje energije

Genetski algoritam treba pronaći konfiguraciju fotonaponskih panela koja osigurava maksimalnu proizvodnju električne energije tijekom zadanog vremenskog razdoblja. Funkcija dobrote definirana je za favoriziranje konfiguracija s većom ukupnom proizvodnjom energije.

Cilj 2: Smanjivanje gubitaka zbog nesklada

Treba umanjiti gubitke zbog nesklada koji nastaju kada se paneli s različitim karakteristikama povežu u seriju ili paralelu. Funkcija dobrote (engl. *fitness*) također treba uzeti u obzir varijancu snage između panela kako bi se smanjili gubitci.

5.2 Odabir programskog jezika

Od početka, za razvijanje genetskog algoritma ideja je bila koristiti programski jezik *Python*. *Python* nije poznat po brzoj izvedbi jer, jer za razliku od mnogih drugih jezika, nije kompiliran, već koristi tumač koda (engl. *interpreter*) koji izvršava liniju po liniju. Ovaj proces prevođenja koda u funkcije u stvarnom vremenu čini *Python* znatno sporijim u usporedbi s drugim jezicima niske razine (engl. l*ow-level*), koji su optimizirani za brzinu i učinkovitost. Razlog za korištenja *Pythona* je njegova velika dostupnost biblioteka koje olakšavaju programiranje u *Pythonu.*

Iz tog razloga, korištena je biblioteka *Numpy*, jer ova biblioteka programeru pruža niz, dodatnih i korisniku prikladnih, funkcija za rukovanje brojevima i linearnom algebrom [15]. *Numpy* olakšava programeru pristup izradi matematičkih jednadžbi i proračuna jer programer ne mora razmišljati o sintaksi algebarskih funkcija, nego jednostavno proslijedi nizove i vrijednosti funkcijama iz biblioteke i proračun dobiva u trenu.

Neke od korištenih funkcionalnosti su:

- **Random:** Generator nasumičnih brojeva dostupan za postavljanje objekata u nizu na nasumična mjesta, primjer takve funkcije je *shuffle* (slika 5.1), ali to je samo jedna od mnogih matematičkih funkcionalnosti nasumičnih brojeva koja je dostupna u ovoj biblioteci.

```
>>> arr = np.arange(10)
>>> np.random.shuffle(arr)
>>> arr
[1 7 5 2 9 4 3 6 0 8] # random
```

Slika 5.1 Primjer korištenja metode Shuffle

- **Array:** Druga bitna skupina funkcionalnosti je *array*, u ovoj skupini se mogu pronaći matematičke funkcije za zbrajanje vrijednosti niza (*sum*) ili za pronalaženje srednje vrijednosti niza s obzirom na uvjet (*mean*), kao što je vidljivo na slici 5.2.

```
>>> import numpy as np
>>> a = np.ma.array([1,2,3], mask=[False, False, True])
>>> a
masked_array(data=[1, 2, --],
             mask=[False, False,  True],
       fill_value=999999)
>>> a.mean()
1.5
```

Slika 5.2 Primjer korištenja metode mean [16]

- **Ostale funkcionalnosti:** Neke od ostalih funkcionalnosti koje su korisne za razvoj funkcije dobrote i genetskog algoritma su funkcije za izračun varijance (*var*) i pronalaženja minimuma (*min*) i maksimuma (*max*).

```
import numpy as np

# kreiranje 2D matrice
array_2d = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])

# izracun varijance matrice
overall_variance = np.var(array_2d)
print(overall_variance)

# izracun varijance stupaca matrice (axis=0)
column_variance = np.var(array_2d, axis=0)
print(column_variance)

# izracun varijance redaka matrice (axis=1)
row_variance = np.var(array_2d, axis=1)
print(row_variance)
```

Slika 5.3 Korištenje metode za izračun varijance (var) [17]

Osim biblioteke *Numpy*, korištene su biblioteke *Scipy* i *Joblib*. *Scipy* biblioteka omogućava brzo i korisnički prilagođeno izračunavanje kompleksnih jednadžbi, jedna od kojih je izračun karakterističnog parametra ćelije C'. Korištenjem metode *fsolve* biblioteka obavlja proračun iz predane formule za određivanje vrijednosti. Zbog složenih proračuna koje genetski algoritam mora obaviti u svakoj iteraciji, uvedena je biblioteka *Joblib*. Ova korisnički prilagođena biblioteka omogućava multiprocesorsko obavljanje rada, te s obzirom na broj procesora ubrzava izvršavanje dijelova programa jer uvodi paralelizaciju rada, odnosno uvodi paralelne regije programa (slika 5.4).

```
joblib.Parallel(
    n_jobs=3,
    backend="loky",
    verbose=100)(
    (joblib.delayed(globalMethod)() for i in range(1000)
    )
)
```

Slika 5.4 Primjer koda korištenja Joblib biblioteke

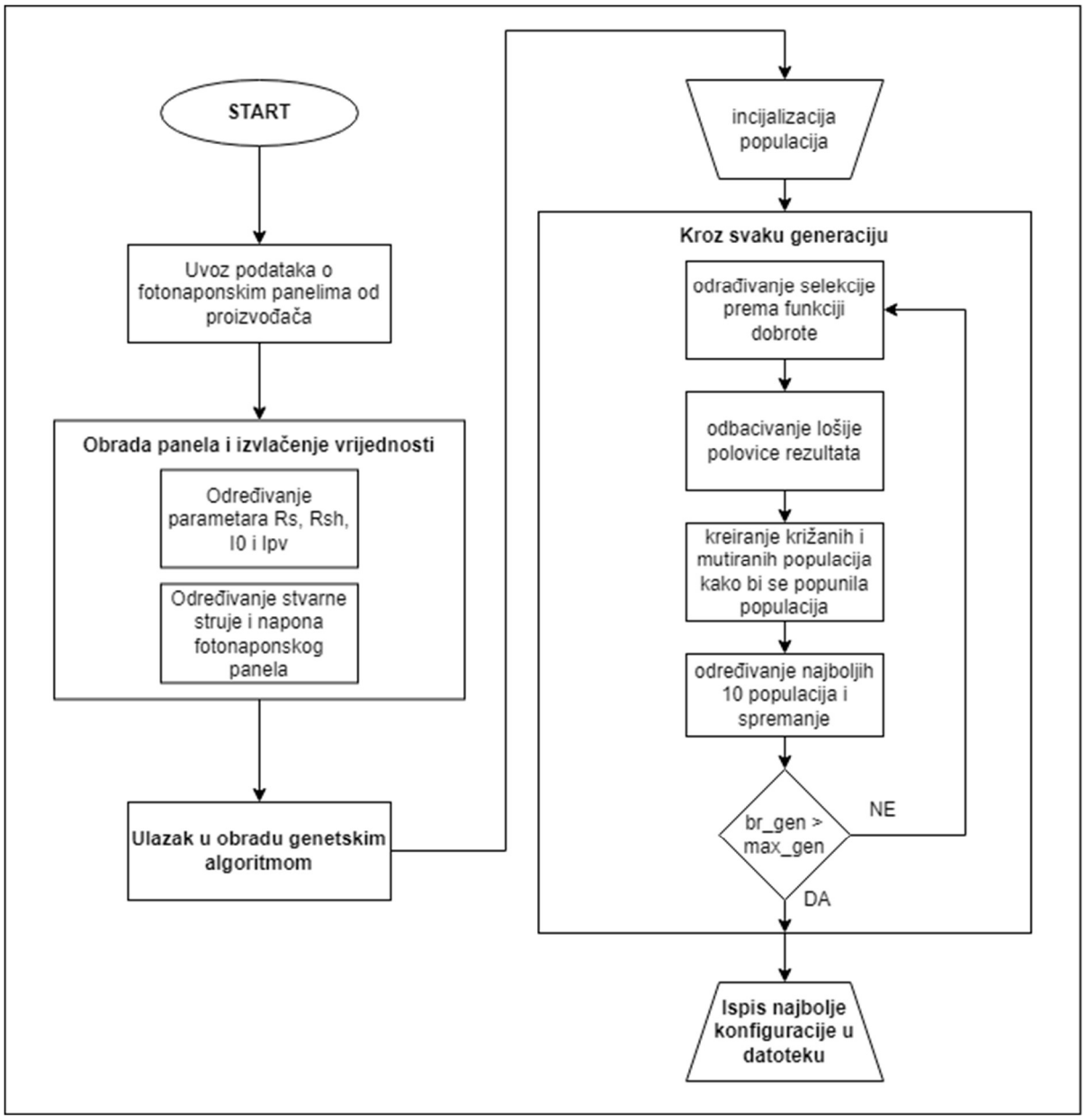
5.3 Kreiranje programa i učitavanje karakteristika fotonaponskih panela

Nakon odabira programskog jezika Python, idući korak u razvoju aplikacije bio je kreiranje konzolnog programa koji može učitati karakteristike dostupnih fotonaponskih panela iz vanjske datoteke. Ovaj korak omogućava pripremu podataka potrebnih za daljnju analizu i optimizaciju pomoću genetskog algoritma.

Program najprije uvozi podatke o fotonaponskim panelima iz .csv datoteke koja sadrži sve relevantne informacije, poput serijskog broja, oznake, izlazne snage, napona i struje panela. Ovi podaci se zatim prosljeđuju genetskom algoritmu koji ih grupira i sortira na temelju faktora dobrote, uzimajući u obzir kriterije kao što su minimizacija gubitaka zbog nesklada i optimizacija faktora punjenja (FF).

Genetski algoritam zatim prolazi kroz iterativni proces, generirajući i evaluirajući različite konfiguracije fotonaponskih panela kako bi pronašao optimalnu grupu koja maksimizira učinkovitost proizvodnje električne energije. Nakon što se pronađe najbolja konfiguracija, rezultati se vraćaju i spremaju u novu datoteku. Ova nova datoteka omogućava daljnje analize ili izvlačenje podataka za praktičnu primjenu.

Kratki dijagram, na slici 5.5, prikazuje tijeka rada programa. Dijagram jasno ilustrira osnovne korake programa: učitavanje podataka o fotonaponskim panelima, obrađivanje uveženih podataka i određivanje karakteristika, izvođenje genetskog algoritma za pronalaženje optimalne konfiguracije, te spremanje sortirane konfiguracije u izlaznu datoteku. Ovaj pristup omogućava jednostavno upravljanje i obradu podataka, uz maksimalnu fleksibilnost za daljnje prilagodbe i optimizaciju.



Slika 5.5 Dijagram tijeka rada programa

Prvi korak u razvoju programa bio je kreiranje klase SolarPanel, koja iz liste podataka sprema sve potrebne podatke o fotonaponskom panelu. Klasa SolarPanel (kod na slici 5.6) definira atributa poput serijskog broja panela (*serialnumber*) te različite elektrotehničke karakteristike kao što su maksimalna izlazna snaga (*pmpp*), napon otvorenog kruga (*uoc*), struja kratkog spoja (*isc*), napon na maksimalnoj snazi (*umpp*), struja na maksimalnoj snazi (*impp*), faktor punjenja (*ff*), broj palete (*palettennummer*). Na kraju su parametri koji su određeni pomoću metode *extraction* sa slike 5.13.

```python
class SolarPanel:
    def __init__(
        self, serialnumber,
        pmpp, uoc, isc, umpp, impp,
        ff, palettennummer
        ):
        self.serialnumber = serialnumber
        self.pmpp = pmpp
        self.uoc = uoc
        self.isc = isc
        self.umpp = umpp
        self.impp = impp
        self.ff = ff
        self.palettennummer = palettennummer
        self.a = 0.0
        self.i0 = 0.0
        self.ipv = 0.0
        self.upv = 0.0
        self.rs = 0.0
        self.rp = 0.0
        self.i = 0.0
        self.u = 0.0
        self.p = 0.0
```

Slika 5.6 Kod za klasu SolarPanel

Ovi atributi omogućuju programskom pristup svim važnim podacima svakog fotonaponskog panela, koji se kasnije koriste u genetskom algoritmu za grupiranje i optimizaciju.

Podaci o fotonaponskim panelima inicijalno su bili dostupni pomoću MS Excel tablice, ali zbog lakšeg rada s podacima unutar programskog okruženja, oni su prebačeni u *.csv* datoteku, odnosno datoteku s vrijednostima odvojenim zarezima (engl. *comma-separated values*). Ovaj format omogućava jednostavno učitavanje i obradu podataka u *Pythonu*.

Primjer podataka iz *.csv* datoteke prikazan je na slici 5.7. Svaka linija u datoteci predstavlja jedan fotonaponski panel s njegovim atributima odvojenim zarezima. Ovaj pristup omogućava učinkovito upravljanje velikim skupovima podataka, što je ključno za optimizaciju u kontekstu genetskog algoritma.

```
Serialnummer,UOC,ISC,UMPP,IMPP,PMPP,FF,WattMarking,CurrentClass,Palettennummer,,,
LRP904094220600401757,50.20 ,13.88 ,41.74 ,13.20 ,550.90 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600401874,50.25 ,13.89 ,41.82 ,13.21 ,552.65 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600406014,50.18 ,13.85 ,41.79 ,13.18 ,550.77 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600405811,50.09 ,13.88 ,41.67 ,13.21 ,550.56 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600405481,50.22 ,13.88 ,42.21 ,13.09 ,552.66 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600401885,50.22 ,13.90 ,41.81 ,13.22 ,552.80 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600401630,50.17 ,13.86 ,41.91 ,13.19 ,553.06 ,0.80 ,550 ,M,LRA904094220661687,,,
LRP904094220600405480,50.20 ,13.87 ,41.92 ,13.21 ,553.75 ,0.80 ,550 ,M,LRA904094220661687,,,
LRP904094220600405475,50.15 ,13.88 ,41.83 ,13.19 ,551.82 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600405466,50.15 ,13.85 ,41.92 ,13.21 ,553.66 ,0.80 ,550 ,M,LRA904094220661687,,,
LRP904094220600401891,50.22 ,13.91 ,41.87 ,13.22 ,553.46 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600405499,50.20 ,13.89 ,41.90 ,13.23 ,554.33 ,0.80 ,550 ,M,LRA904094220661687,,,
LRP904094220600405477,50.13 ,13.83 ,41.78 ,13.16 ,550.01 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600404016,50.24 ,13.89 ,41.79 ,13.21 ,551.92 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600401858,50.19 ,13.87 ,41.78 ,13.19 ,550.95 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600401824,50.22 ,13.87 ,41.82 ,13.19 ,551.52 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600405936,50.13 ,13.89 ,41.78 ,13.21 ,551.80 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600405504,50.17 ,13.83 ,41.86 ,13.17 ,551.46 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600403259,50.10 ,13.84 ,41.77 ,13.17 ,550.27 ,0.79 ,550 ,M,LRA904094220661687,,,
LRP904094220600403137,50.15 ,13.87 ,41.82 ,13.22 ,552.90 ,0.80 ,550 ,M,LRA904094220661687,,,
LRP904094220600401682,50.20 ,13.89 ,41.78 ,13.22 ,552.42 ,0.79 ,550 ,M,LRA904094220661687,,,
```

Slika 5.7 Isječak iz .csv datoteke

Podatke o fotonaponskim panelima iz datoteke s vrijednostima odvojenim zarezima (*.csv*) bilo je potrebno uvesti u program kako bi se dalje obradili u genetskom algoritmu. Za ovu funkcionalnost korištena je *Pythonova* standardna biblioteka *csv*, koja omogućava jednostavno čitanje podataka iz *.csv* datoteka.

Na slici 5.8 prikazana je metoda importPanels koja učitava podatke o fotonaponskim panelima s predane lokacije datoteke. Funkcija započinje otvaranjem datoteke pomoću naredbe open, s postavkom za čitanje ('r') i UTF-8 enkodiranjem, kako bi se osigurala kompatibilnost s različitim znakovnim skupovima.

```python
import csv
from SolarPanel import SolarPanel

def importPanels(location) -> list[SolarPanel]:
    solar_panels = []
    with open(location, mode='r', encoding='utf-8') as file:
        reader = csv.DictReader(file)
        for row in reader:
            solar_panel = SolarPanel(
                row['Serialnummer'],
                float(row['PMPP']),
                float(row['UOC']),
                float(row['ISC']),
                float(row['UMPP']),
                float(row['IMPP']),
                float(row['FF']),
                row['Palettennummer']
            )
            solar_panels.append(solar_panel)

    return solar_panels
```

Slika 5.8 Kod za čitanje iz .csv datoteke

Biblioteka *csv.DictReader* koristi se za čitanje sadržaja datoteke redak po redak, pri čemu svaki redak predstavlja jedan fotonaponski panel sa svim njegovim karakteristikama. Svaki redak se zatim pretvara u objekt klase *SolarPanel* korištenjem podataka iz pojedinačnih stupaca datoteke. Stupci su imenovani prema nazivima ključeva u *.csv* datoteci, poput *'Serialnummer'*, *'Palettennummer'*, itd. Numeričke vrijednosti poput *'PMPP'*, *'UOC'*, *'ISC'*, i ostalih tehničkih specifikacija fotonaponskih panela, konvertiraju se u broj s pomičnim zarezom (engl. *float*) kako bi se mogle pravilno koristiti u daljnjim matematičkim izračunima.

Nakon što je redak pretvoren u objekt *SolarPanel*, taj objekt se dodaje u listu *solar_panels*. Ovaj postupak se ponavlja za svaki redak u datoteci, čime se kreira lista svih panela s njihovim atributima.

Na kraju, metoda vraća ovu listu *solar_panels* kao izlaznu vrijednost koja se prosljeđuje glavnom dijelu programa za daljnju obradu i analizu (slika 5.9). Ova metoda omogućava učinkovito učitavanje podataka o fotonaponskim panelima i njihovu pripremu za daljnje korake u genetskom algoritmu, što je ključno za uspješnu optimizaciju.

```
21      if platform.system() == "Windows":
22          base_dir = Path(__file__).resolve().parent.parent
23          input_file = base_dir / 'longi-dataset.csv'
24          output_file = base_dir / 'new_solar_panels.csv'
25      else:
26          input_file = 'longi-dataset.csv'
27          output_file = 'new_solar_panels.csv'
28
29      # Step 1: Import solar panel data
30      solarPanels = importPanels(str(input_file))
```

Slika 5.9 Kod koji poziva metodu importPanels koja vraća listu fotonaponskih panela

U podnaslovu 5.4 su detalji kako funkcionira genetski algoritam, a nakon što genetski algoritam dovrši svoj rad i pronađe optimalnu konfiguraciju fotonaponskih panela, grupirani fotonaponski paneli moraju se izvesti u novu datoteku za daljnju analizu ili primjenu. Za ovu funkcionalnost potrebno je kreirati metodu za zapisivanje podataka u datoteku s vrijednostima odvojenim zarezima (*CSV*). Na slici 5.10 prikazana je metoda *exportPanels* koja se koristi za ovu svrhu.

```
1   import csv
2   from SolarPanel import SolarPanel
3
4   def exportPanels(solarPanels: list[SolarPanel], output_location):
5       with open(output_location, mode='w', newline='', encoding='utf-8') as csvfile:
6           fieldnames = ['Serialnummer', 'PMPP', 'UOC', 'ISC', 'UMPP',
7                         'IMPP', 'FF', 'Palettennummer']
8           writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
9           writer.writeheader()
10          for panel in solarPanels:
11              writer.writerow({
12                  'Serialnummer': panel.serialnumber,
13                  'PMPP': panel.pmpp,
14                  'UOC': panel.uoc,
15                  'ISC': panel.isc,
16                  'UMPP': panel.umpp,
17                  'IMPP': panel.impp,
18                  'FF': panel.ff,
19                  'Palettennummer': panel.palettennummer,
20              })
21      return 1
```

Slika 5.10 Kod za pisanje u .csv datoteku

Metoda *exportPanels* prima kao parametre niz fotonaponskih panela i lokaciju datoteke u koju treba zapisati podatke. Na sličan način kao kod učitavanja podataka iz datoteke (prikazano na slici 5.8), metoda koristi *Pythonovu* standardnu biblioteku *csv* za pisanje podataka u CSV

formatu. Metoda otvara datoteku za pisanje ('w') s postavkom za UTF-8 enkodiranje kako bi se osigurala kompatibilnost s različitim znakovnim skupovima. Koristi se *csv.DictWriter* za definiranje naziva stupaca (*fieldnames*) i za pisanje zaglavlja datoteke pomoću *writeheader()* metode. Zatim se iterira kroz svaki fotonaponski panel u nizu, a atributi svakog panela se pišu u odgovarajuće stupce korištenjem *writerow()* metode. Nakon uspješnog zapisivanja podataka, metoda vraća vrijednost koja signalizira uspjeh procesa, čime se završava tijek programa. Na slici 5.11 prikazano je kako se ova metoda poziva iz glavnog dijela programa, a ako je datoteka uspješno stvorena, korisniku se prikazuje poruka s potvrdom i lokacijom nove datoteke.

Na ovaj način, korisnik može jednostavno pristupiti grupiranim podacima fotonaponskih panela u novostvorenoj CSV datoteci, što omogućava daljnje analize ili primjene rezultata genetskog algoritma.

```
55  # Step 4: Export the flattened configuration to a CSV file
56  isCreated = exportPanels(flattened_configuration, output_file)
57
58  # Step 5: Provide feedback to the user
59  if isCreated:
60      print(f"Best configuration saved to file: {output_file}")
61  else:
62      print("Failed to save the configuration.")
63
```

Slika 5.11 Kod koji poziva metodu za pisanje u datoteku i ispis izlazne poruke

5.4 Razvoj genetskog algoritma

Glavni dio programa ima inicijalizirane parametre koji se prosljeđuju genetskom algoritmu za optimizaciju konfiguracije fotonaponskih panela. Ti parametri su prikazani na slici 5.12 i uključuju veličinu populacije, broj generacija, stopu mutacije, veličinu grupa fotonaponskih panela, karakteristični parametar povezan s faktorom punjenja (FF) fotonaponskih ćelija kao i parametre povezane s brojem ćelija u seriji, brojem paralelnih podnizova i brojem serijskih blokova. Također na slici 5.12 slici vidljiv je poziv metode *extracion* (slika 5.13), pomoću koje se izračunavaju paralelni R_{sh} i serijski otpor R_s te fotostruja I_{pv}. Izračunati su i stvarna struja i napon svakog pojedinačnog fotonaponskog panela. Izračun tih parametara se obavlja iterativno kroz *while* petlje.

```
# Step 1: Import solar panel data
solarPanels = importPanels(str(input_file))
# Parameters for the genetic algorithm
population_size = 1000              # Number of configurations in each generation
generations = 10000                 # Number of generations the algorithm will run
mutation_rate = 0.2                 # Probability of mutation occurring
number_of_cells = 144               # Number of cells in one panel
L = 36                              # Number of panels in series within each substring
M = 2                               # Number of parallel substrings

#Step 2: Extracting and aproximating panel characteristics
print("Extracting variables...")
for i, panel in enumerate(solarPanels):
    solarPanels[i] = extraction(panel, number_of_cells)
print("Done extracting.")
sorted_solar_panels = sorted(solarPanels, key=lambda x: x.impp, reverse=True)

# #Step 3: Run the genetic algorithm to find the best configuration
print("Starting algorithm...")
best_configuration = doAlgorithm(
    solar_panels=sorted_solar_panels,
    sortType=SortType.IMPP,
    population_size=population_size,
    generations=generations,
    mutation_rate=mutation_rate,
    L=L, M=M)
```

Slika 5.12 Parametri potrebni za metodu doAlgorithm

```python
def extraction(solarPanel: SolarPanel, number_of_cells: int):
    q = 1.6e-19
    k = 1.38e-23
    T = 298
    Isc, Voc, Imp, Vmp = solarPanel.isc, solarPanel.uoc, solarPanel.impp, solarPanel.umpp
    N, Pmax = number_of_cells, Vmp*Imp
    A = solarPanel.ff/100
    if (solarPanel.ff < 1.0):
        A = solarPanel.ff
    Vt = (k*A*T*N)/q
    Rs =  (Voc/Imp) - (Vmp/Imp) + ((Vt/Imp)*log((Vt)/(Vt + Vmp)))
    I0 = Isc/(np.exp(Voc/Vt)) - np.exp(Rs*Isc/Vt)
    Ipv = I0 * (np.exp(Voc/Vt) - 1)
    #firstStep
    iter = 10000
    it = 0
    tol = 0.1
    A1 = A
    VmpC = (Vt * np.log((Ipv + I0 - Imp) / I0)) - (Rs * Imp)
    e1 = VmpC - Vmp
    Rs1 = Rs
    while it < iter and e1 > tol:
        if VmpC < Vmp:
            A1 -= 0.01
        else:
            A1 += 0.01
        Vt1 = (k * A1 * T * N) / q
        I01 = Isc / (np.exp(Voc / Vt1) - np.exp(Rs1 * Isc / Vt1))
        Ipv1 = I01 * (np.exp(Voc / Vt1) - 1)
        VmpC = (Vt1 * np.log((Ipv1 + I01 - Imp) / I01)) - (Rs1 * Imp)
        e1 = VmpC - Vmp
        it += 1
    vt1 = (k * A1 * T * N) / q
    Rs1 = (Voc / Imp) - (VmpC / Imp) + ((vt1 / Imp) * log(vt1 / (vt1 + VmpC)))
    #secondStep
    tolI = 0.001
    iter = 10000
    itI = 0
    I01 = Isc / (np.exp(Voc / vt1) - np.exp(Rs1 * Isc / vt1))
    Ipv1 = I01 * (np.exp(Voc / vt1) - 1)
    Rp = ((-Vmp) * (Vmp + (Rs1 * Imp))) / (Pmax - (Vmp * Ipv1) + (Vmp * I01 * (np.exp((Vmp + (Rs1 * Imp)) / vt1) - 1)))
    I02 = (Isc * (1 + Rs1 / Rp) - Voc / Rp) / (np.exp(Voc / vt1) - np.exp(Rs1 * Isc / vt1))
    Ipv2 = I02 * (np.exp(Voc / vt1) - 1) + Voc / Rp
    ImpC = Pmax / VmpC
    err = abs(Imp - ImpC)
    Rpnew = Rp
    while err > tolI and itI < iter:
        if ImpC < Imp:
            Rpnew = Rp + 0.1 * itI
        else:
            Rpnew = Rp - 0.1 * itI
        I02 = (Isc * (1 + Rs1 / Rpnew) - Voc / Rpnew) / (np.exp(Voc / vt1) - np.exp(Rs1 * Isc / vt1))
        Ipv2 = I02 * (np.exp(Voc / vt1) - 1) + Voc / Rpnew
        def eqn(ImpC):
            return Ipv2 - (I02 * (np.exp((Vmp + (Rs1 * ImpC)) / vt1) - 1)) - ImpC - (Vmp + Rs1 * ImpC) / Rpnew
        current_c = Imp
        ImpC = fsolve(eqn, current_c)[0]
        itI += 1
        err = abs(Imp - ImpC)
    #thirdStep
    vt_new = (k*A1*T*N)/q
    tolerance = 1e-6
    max_iterations = 1000
    previous_I = solarPanel.impp
    iterations = 0
    while iterations < max_iterations:
        term1 = Ipv2 - I02 * (exp((VmpC + previous_I * Rs1) / vt_new) - 1)
        term2 = (VmpC + previous_I * Rs1) / Rpnew
        new_I = term1 - term2
        if abs(new_I - previous_I) < tolerance:
            break
        previous_I = new_I
        iterations += 1
    I solution = new I
```

```
82      #forthStep
83      tolerance = 1e-6
84      max_iterations = 1000
85      # Iteratively find I
86      previous_V = solarPanel.uoc
87      iterations = 0
88      while iterations < max_iterations:
89          term1 = vt_new * log(((Ipv2 + I02) - I_solution - (previous_V + I_solution * Rs1) / Rpnew) / I02)
90          term2 = Rs1 * I_solution
91          new_V = term1 - term2
92          if abs(new_V - previous_V) < tolerance:
93              break
94          previous_V = new_V
95          iterations += 1
96      V_solution = new_V
97      solarPanel.u, solarPanel.i, solarPanel.p = V_solution, I_solution, V_solution * I_solution
98      solarPanel.a, solarPanel.i0, solarPanel.ipv, = A1, I02, Ipv2
99      solarPanel.rs, solarPanel.rp = Rs1, Rpnew
100     return solarPanel
```

Slika 5.13 Metoda (extraction) za izračun potrebnih parametara iz parametara fotonaponskog panela

Parametri se prenose u metodu *doAlgorithm*, što je također prikazano na slici 5.12. Korištenjem ovih parametara, genetski algoritam prilagođava svoju pretragu prostora rješenja s ciljem pronalaženja optimalne konfiguracije koja minimizira gubitke energije zbog nesklada i maksimizira učinkovitost fotonaponskih panela.

Izmjenom ovih parametara moguće je fino podešavati rad algoritma kako bi se dobili različiti rezultati, ovisno o specifičnim zahtjevima ili uvjetima okoline. Na taj način, korisnik može eksperimentirati s različitim vrijednostima kako bi postigao najbolje rezultate za određeni scenarij, čime se omogućuje fleksibilnost i prilagodba algoritma specifičnim potrebama.

Dijagram na slici 5.5 prikazuje tijek izvršavanja genetskog algoritma unutar metode *doAlgorithm*. Algoritam započinje inicijalizacijom populacije, što uključuje stvaranje početnog skupa rješenja koja se dalje koriste u procesu optimizacije. Nakon inicijalizacije, algoritam ulazi u glavnu petlju koja se ponavlja kroz određeni broj generacija.

Petlja generacija nastavlja se sve dok se ne ispuni zadani broj generacija. Na kraju, nakon što su sve generacije dovršene, algoritam vraća najbolje iteracije koje su pohranjene tijekom cijelog procesa. Dijagram jasno prikazuje logiku genetskog algoritma te kombinaciju različitih koraka poput inicijalizacije, križanja, mutacije, procjene i selekcije u cilju postizanja optimalnih rješenja.

Za početak, potrebno je inicijalizirati početnu populaciju fotonaponskih panela koja će se koristiti u procesu optimizacije genetskog algoritma. Inicijalizacija populacije stvara početni skup konfiguracija koje će se dalje evoluirati tijekom iteracija algoritma kako bi se pronašle

optimalne konfiguracije fotonaponskih panela. Ulazni niz fotonaponskih panela razdvaja se u grupe kako bi se formirala početna populacija, iz koje će se pokrenuti proces traženja najboljih permutacija. Najbolja permutacija, nakon obrade svih generacija, je konačno rješenje koje vraća genetski algoritam.

Metoda *initialize_population* (slika 5.14) ima par ulaznih parametra:

- Niz fotonaponskih panela – ovo je niz objekata fotonaponskih panela koji se prosljeđuje iz glavnog programa. Sadrži sve panele koji će biti raspoređeni u grupe unutar populacije.
- Veličinu jednog podniza L – ova vrijednost određuje koliko će fotonaponskih panela biti uključeno u svakom pojedinom podnizu. Odabir veličine podniza važan je jer utječe na raznolikost i kvalitetu rješenja koja algoritam može istražiti.
- Broj podnizova u jednoj grupi M – ova vrijednost određuje koliko će podnizova imati jedna grupa.
- Veličinu populacije – ova vrijednost definira koliko različitih iteracija rješenja će biti u početnoj populaciji. Veća veličina populacije može povećati šanse za pronalaženje optimalnog rješenja, ali također povećava i vrijeme potrebno za izvođenje algoritma.
- Tip sortiranja – ova vrijednost određuje pomoću koje vrijednosti ćemo odrediti početne nizove panela (slika 5.15)
    a. *SHUFFLE* – nasumično generira listu za svaku populaciju
    b. *IMPP* – generirane liste su sortirane prema struji na maksimalnoj snazi panela
    c. *UMPP* – generirane liste su sortirane prema naponu na maksimalnoj snazi panela
    d. *PMPP* – generirane liste su sortirane prema maksimalnoj snazi panela

```python
def shuffleArray(array: list[SolarPanel], sortType: SortType) -> list[SolarPanel]:
    """
    Randomly shuffle the array of SolarPanel objects using numpy.
    """
    np_array = np.array(array)
    match sortType:
        case SortType.SHUFFLE:
            np.random.shuffle(np_array)
        case SortType.IMPP:
            np_array = sorted(np_array.copy(), key=lambda x: x.impp, reverse=True)
        case SortType.UMPP:
            np_array = sorted(np_array.copy(), key=lambda x: x.umpp, reverse=True)
        case SortType.PMPP:
            np_array = sorted(np_array.copy(), key=lambda x: x.pmpp, reverse=True)
    return list(np_array)

def initialize_population(
    solar_panels: list[SolarPanel],
    L: int, M: int, population_size: int,
    sortType: SortType
) -> list[list[list[SolarPanel]]]:
    """
    Create a population of random configurations. Each configuration consists of N series blocks,
    each containing M parallel substrings, and each substring has L panels in series.
    """
    population = []
    panels_per_configuration = L * M
    configurations_per_set = len(solar_panels) // panels_per_configuration
    for _ in range(population_size):
        copied_panels = shuffleArray(solar_panels.copy(), sortType)
        configuration = group_panels(copied_panels, L)
        np_array = np.array(configuration)
        np.random.shuffle(np_array)
        configuration = list(np_array)
        for i, _ in enumerate(configuration):
            configuration[i] = list(configuration[i])
        population.append(configuration)
    return population
```

Slika 5.14 Kod metode za inicijalizaciju početne populacije i metode za nasumični odabir niza

```python
#Types of sort in genetic algorithm
class SortType(Enum):
    SHUFFLE = 0,
    IMPP = 1,
    UMPP = 2,
    PMPP = 3
```

Slika 5.15 Tipovi sortiranja početnih populacija

U svakoj iteraciji inicijalizacije, niz fotonaponskih panela se prosljeđuje u metodu *shuffleArray* (slika 5.15). Ova metoda s obzirom na varijablu *sortType* odabire način sortiranja početne iteracije jedne populacije, u *shuffle* modu koristi biblioteku Numpy kako bi nasumično promiješala redoslijed fotonaponskih panela, stvarajući novi, nasumično poredani niz panela. Ostali tipovi sortiranja su prema karakteristikama samih panela. Nakon toga, sortirani niz se dijeli u podgrupe čija je veličina definirana parametrom *group_size*.

Ako posljednja grupa ne ispunjava traženi broj elemenata (nije potpuna), ta se grupa uklanja kako bi se osigurala konzistentnost veličine svake grupe. Izlaz iz ove metode je 4D matrica, ugniježđena lista koja predstavlja početni niz populacija, gdje svaka populacija predstavlja pojedinačno rješenje (konfiguraciju) za daljnju evoluciju kroz generacije genetskog algoritma. Ova inicijalna populacija je temelj za daljnju optimizaciju jer pruža početne uvjete iz kojih algoritam može tražiti optimalne konfiguracije fotonaponskih panela. Svaku generaciju se radi odabir najboljih populacija, te se uzima gornja polovica najboljih populacija prema faktoru funkcije dobrote (slika 5.16). Ovdje na liniji 80 (slika 5.16) je korištena biblioteka *Joblib*, jer proces izračuna faktora dobrote je procesorski i vremenski zahtjevna metoda.

```
71  def selection(
72      population: list[list[list[SolarPanel]]],
73      L: int, M: int
74  ) -> list[list[list[SolarPanel]]]:
75      """
76      Select the top sets of configurations based on their fitness score.
77      Each set uses all the solar panels.
78      """
79      # Parallel execution of fitness calculation
80      fitness_scores = Parallel(n_jobs=-1, backend="loky")(
81          delayed(fitness)(config, L, M) for config in population
82      )
83      population_with_fitness: List = list(zip(fitness_scores, population))
84      population_with_fitness.sort(key=lambda x: x[0], reverse=True)
85      retain_length = len(population) // 2
86      selected_population = [x[1] for x in population_with_fitness[:retain_length]]
87      return selected_population
```

Slika 5.16 Kod za metoda sortiranja populacije funkcijom dobrote

Funkcija dobrote (*fitness*) procjenjuje kvalitetu svake konfiguracije fotonaponskih panela na temelju izračuna gubitaka zbog nesklada svake grupe. Slika 5.17 prikazuje strukturu funkcije dobrote. Za ovaj genetski algoritam, funkcija dobrote vrši izračuna ukupnog gubitka zbog nesklada svih grupa.

```python
def fitness(configuration: list[list[SolarPanel]], L: int, M: int) -> float:
    """
    Calculate the fitness of the configuration based on minimizing mismatch losses.
    """
    total_loss = 0.0
    index = 0
    while (index+1 < len(configuration)):
        g1 = configuration[index]
        g2 = configuration[index+1]
        flattened_group = flatten_panels_recursively(g1) + flatten_panels_recursively(g2)
        total_loss += calculate_mismatch_loss(flattened_group, L, M, average_max_value(g1, g2))
        index += 2
    fitness_score = 1 / (total_loss + 1e-6)  # Adding a small value to prevent division by zero
    return float(fitness score)
```

Slika 5.17 Kod za funkciju dobrote

Prije slanja fotonaponskih panela u metodu za izračun gubitka zbog nesklada, potrebno je pronaći srednje vrijednosti napona, struje i snage tih grupa, to se obavlja pomoću metode *average_max_value* (slika 5.18). Unutar ove metode obavljaju se aproksimacije najvećih napona podnizova, korištenjem formule 3-1 iz trećeg poglavlja. Ovo je iterativni postupak koji je procesorski zahtjevan, iz tog razloga je cijela metoda za izračun faktora dobrote postavljena u paralelne regije, kako bi se osigurala ubrzana izvedba programa.

```python
def average_max_value(g1: list[SolarPanel], g2: list[SolarPanel]):
    max_values_groups = [find_max_values_of_group(g1), find_max_values_of_group(g2)]
    total_len = len(g1) + len(g2)
    avg_max_values = [0.0, 0.0, 0.0, 0.0]
    avg_max_values[0] = float(np.sum([max_values_groups[0][0] + max_values_groups[1][0]])/total_len)
    avg_max_values[1] = float(np.mean(max_values_groups[0][1]))
    avg_max_values[2] = float(np.mean(max_values_groups[1][1]))
    avg_max_values[3] = float(np.sum([(max_values_groups[0][2] + max_values_groups[1][2])])/total_len)
    return avg_max_values

def find_max_values_of_group(groupedPanels: list[SolarPanel], group = -1) -> list[float]:
    space = 1000
    max_values = [0.0,0.0,0.0] #U I P
    max_isc = max(panel.isc for panel in groupedPanels)
    min_i = min(panel.impp for panel in groupedPanels)
    if (group >= 0):
        max_isc = 14.5
        min_i = 0
    I_values = np.linspace(min_i, max_isc, space)
    V_values = []
    P_values = []
    for I in I_values:
        v = 0
        for panel in groupedPanels:
            v+= compute(I, panel, do_Voltage_Calc, panel.umpp)
        if (v*I > max_values[2]):
            max_values = [v, I, v*I]
        if (group >= 0):
            P_values.append(v*I)
            V_values.append(v)
    if (group >= 0):
        drawCurve(V_values, I_values, P_values, groupedPanels, max_values, group)
        return [V_values, I_values, P_values, max_values] # type: ignore
    return [round(float(value), 5) for value in max_values]
```

Slika 5.18 Kod za aproksimaciju maksimalnih vrijednosti napona, struje i snage konfiguracije

Izračun gubitaka zbog nesklada (*total_loss*) temelji se na formuli 3-10 iz trećeg poglavlja, korištenjem parametara kao što su karakteristika ćelije, broj ćelija u seriji i paraleli. Manji gubitci zbog nesklada ukazuju na bolju konfiguraciju, jer je cilj minimizirati nesklad između fotonaponskih ćelija kako bi se postigla maksimalna učinkovitost.

Metoda *calculate_mismatch_loss (slika 5.19)* ima 5 ulaznih parametara:

- Trenutna konfiguracija.
- C – Karakteristični parametar povezan s faktorom punjenja (FF) fotonaponskih ćelija.
- L – broj panela u seriji u svakom podnizu
- M – broj paralelnih podnizova
- max_values – maksimalne vrijednosti napona struje i snage grupe

```python
def calculate_mismatch_loss(
    flattened_panels: list[SolarPanel],
    L: int, M: int, max_values=[0.0,0.0,0.0,0.0]
) -> float:
    """
    Calculate the mismatch loss for a single series block of solar panels.
    """
    group1 = flattened_panels[:L]
    group2 = flattened_panels[L:]
    # Use fsolve to solve for C
    ff = np.average([panel.ff for panel in flattened_panels])
    ff1 = np.average([panel.ff for panel in group1])
    ff2 = np.average([panel.ff for panel in group2])
    if (ff > 1.0):
        ff = ff/100
        ff1 = ff1/100
        ff2 = ff2/100
    initial_guess = 1.0  # Initial guess for C
    C  = fsolve(find_C, initial_guess, args=(ff))
    C1 = fsolve(find_C, initial_guess, args=(ff1))
    C2 = fsolve(find_C, initial_guess, args=(ff2))
    p_group1 = [panel.umpp*panel.impp for panel in group1]
    p_group2 = [panel.umpp*panel.impp for panel in group2]
    umpp_values = [panel.umpp for panel in flattened_panels]
    i_group1 = [panel.impp for panel in group1]
    i_group2 = [panel.impp for panel in group2]
    if (max_values[0] == 0.0):
        max_values[0] = np.mean(umpp_values)
        max_values[1] = np.mean(i_group1)
        max_values[2] = np.mean(i_group2)
        max_values[3] = np.mean([panel.impp*panel.umpp for panel in flattened_panels])
    p_ideal1 = np.sum(p_group1)
    p_ideal2 = np.sum(p_group2)
    sigma_v = (np.std(umpp_values)/max_values[0])**2
    sigma_i1 = (np.std(i_group1)/max_values[1])**2
    sigma_i2 = (np.std(i_group2)/max_values[2])**2
    alfa = (1 - 1/L) * 1/M
    beta1 = (p_ideal1/max_values[3]) * (C1[0]+2) * sigma_i1
    beta2 = (p_ideal2/max_values[3]) * (C2[0]+2) * sigma_i2
    beta = 0.5 * (beta1 + beta2)
    gamma = 0.5 * (C[0]+2) * sigma_v/L * (1-1/M)
    mismatch_loss = alfa * beta + gamma
    return float(mismatch_loss * 100)
```

Slika 5.19 Kod za metodu izračuna gubitka zbog nesklada fotonaponskih ćelija

Izraz za dobivanje faktora dobrote (engl. *fitness score*, slika 5.17, linija 25) sadrži recipročnu vrijednost ukupnog gubitka kako bi se favorizirale konfiguracije s manjim gubitcima. Recipročna vrijednost (1 / *total_loss*) daje veći rezultat za konfiguracije s manjim gubitcima, što je u skladu s ciljem optimizacije, koji je obrnuto proporcionalan gubitcima. Dodaje se mala vrijednost (1e-6) kako bi se izbjeglo dijeljenje s nulom u slučaju kada bi gubitci bili jednaki nuli.

Na taj način, funkcija dobrote uzima u obzir oba ključna faktora: minimiziranje gubitaka zbog nesklada i maksimiziranje učinkovitosti kroz faktor punjenja, pružajući uravnoteženi kriterij za procjenu kvalitete svake konfiguracije unutar genetskog algoritma.

Nakon izračuna dobiveni faktor dobrote se prosljeđuje natrag u *selection* (slika 5.16) u kojoj se dovršava proces sortiranja. Dobivena sortirana populacija zatim se koristi kao ulaz za daljnje iteracije unutar *while* petlje, koja je ključni dio genetskog algoritma.

U *while* petlji (slika 5.20, linije 154-158), algoritam iterira kroz populacije kako bi generirao novu generaciju potomaka (*offspring*).

```python
def doAlgorithm(
    solar_panels: list[SolarPanel],
    sortType: SortType = SortType.SHUFFLE,
    population_size: int = 50,
    generations: int = 100,
    mutation_rate: float = 0.05,
    L: int = 20,
    M: int = 2
):
    logger = logging.getLogger(__name__)
    logging.basicConfig(filename="./results_longi-test-10000.log", encoding='utf-8', level=logging.DEBUG)
    configurations_per_set = len(solar_panels) // L
    population = initialize_population(solar_panels, L, M, population_size, sortType)
    top_sets = []
    for generation in range(generations):
        g = f"{datetime.datetime.now()}: Generation {generation + 1}"
        logger.debug(g)
        population = selection(population, L, M)
        offspring = []
        while len(offspring) <= population_size - len(population):
            parent1, parent2 = random.sample(population, 2)
            child = crossover(solar_panels, parent1, parent2, L, M, configurations_per_set)
            child = mutate(child, mutation_rate, L, M)
            offspring.append(child)
        population.extend(offspring)
        if len(population) > population_size:
            population = population[:population_size]
        top_sets.extend(population)
        # Parallel execution of fitness calculation
        top_sets = Parallel(n_jobs=-1, backend="loky")(
            delayed(fitness)(config, L, M) for config in population
        )
        population_with_fitness : List = list(zip(top_sets, population))
        population_with_fitness.sort(key=lambda x: x[0], reverse=True)
        top_sets = [config for fitness_score, config in population_with_fitness[:10]]
        if (generation%10 == 0): #print out every 10 generations
            total_loss = 0.0
            index = 0
            print(len(flatten_panels_recursively(top_sets[0])))
            while (index+1< len(top_sets[0])):
                g1 = top_sets[0][index]
                g2 = top_sets[0][index+1]
                avg_max_values = average_max_value(g1, g2)
                flattened_group1 = flatten_panels_recursively(g1)
                flattened_group2 = flatten_panels_recursively(g2)
                flattened_group = flattened_group1 + flattened_group2
                group_loss = calculate_mismatch_loss(flattened_group, L, M, avg_max_values)
                total_loss += group_loss
                index += 2
            result = f"{datetime.datetime.now()} Lowest mismatch = {total_loss:.8f}"
            logger.debug(result)
            print(result)
    winner = max(top_sets, key=lambda config: fitness(config, L, M))
    dup = has_duplicate_panels(flatten_panels_recursively(winner))
    print(f"Winner has {dup} duplicates.")
    return population
```

Slika 5.20 Cijeli kod metode doAlgorithm

Prvo se nasumično odabiru dva roditelja (*parent1* i *parent2*) iz trenutne populacije pomoću funkcije *random.sample()* (slika 5.20, linija 155). Zatim se nad tim roditeljima provodi operacija križanja (crossover), koja kombinira njihove genetske informacije i stvara novo dijete, koje je konfiguracija nastala od križanja podataka iz dvije konfiguracije (slika 5.21).

Nakon toga, na novo dijete se primjenjuje operacija mutacije (slika 5.22), koja uvodi slučajne promjene u genetsku strukturu potomka kako bi se povećala raznolikost populacije i spriječila preuranjena konvergencija algoritma na lokalni minimum.

```python
def crossover(
    all_panels: list[SolarPanel],
    parent1: list[list[SolarPanel]],
    parent2: list[list[SolarPanel]],
    L: int, M: int, configurations_per_set: int
) -> list[list[SolarPanel]]:
    """
    Perform crossover between two parent sets to produce a child set.
    Ensures no duplicate panels across the child set.
    Handles cases where we run out of panels by discarding incomplete configurations.
    """
    crossover_point = np.random.randint(1, configurations_per_set-1)
    child_set = parent1[:crossover_point] + parent2[crossover_point:]
    child_set_panels = flatten_panels_recursively(child_set)
    used_serial_numbers = {panel.serialnumber for panel in child_set_panels}
    missing_panels = [panel for panel in all_panels if panel.serialnumber not in used_serial_numbers]
    required_panels = L * len(child_set)
    total_panels = len(all_panels)
    if total_panels < required_panels:
        full_configs_possible = total_panels // L
        child_set = child_set[:full_configs_possible]
    for panel in missing_panels:
        added = False
        for substring in child_set:
            if len(substring) < L:
                substring.append(panel)
                added = True
                break
            if added:
                break
    child_set_panels = flatten_panels_recursively(child_set)
    used_serial_numbers = {panel.serialnumber for panel in child_set_panels}
    missing_panels = [panel for panel in all_panels if panel.serialnumber not in used_serial_numbers]
    child_set_panels = swap_duplicate(child_set_panels, missing_panels)
    return group_panels(child_set_panels, L)
```

Slika 5.21 Kod metode crossover

```python
def mutate(
    configurations: list[list[SolarPanel]],
    mutation_rate: float, L: int, M: int
) -> list[list[SolarPanel]]:
    """
    Mutate a set of configurations by swapping panels within substrings our within groups.
    Ensures no duplicate panels within the set.
    """
    all_panels = flatten_panels_recursively(configurations)
    if np.random.rand() < mutation_rate:
        i, j = np.random.choice(len(all_panels), 2, replace=False)
        all_panels[i], all_panels[j] = all_panels[j], all_panels[i]
    return group_panels(all_panels, L)
```

Slika 5.22 Kod metode mutate

U metodama *crossover* i *mutate* koriste se pomoćne metode *flatten_panels_recursively* i *group_panels* koje možemo vidjet na slici 5.23. Ove pomoćne metode olakšavaju i održavaju formate konfiguracija potrebne za obradu podataka, u slučajima gdje je lakše i brže samo iterirati kroz jedan dugačak niz, rekurzivno ugniježđene nizove, metodom *flatten_panels_recursively,* spajamo u jedan dugačak niz. Dok kad je potrebna struktura, takav niz, metodom *group_panels*, možemo vratiti u originalnu potrebnu strukturu s ugniježđenim nizovima. Ove metode se koriste kroz cijelu aplikaciju. Također unutar metode *crossover* provjerava se sadrži li dijete duplikate, te se obavljaju zamjene duplikata pomoću metode *swap_duplicate* (slika 5.24).

```
32  def group_panels(solarPanels: list[SolarPanel], L: int) -> list[list[SolarPanel]]:
33      return [solarPanels[i:i + L] for i in range(0, len(solarPanels) - len(solarPanels) % L, L)]
34
35  def flatten_panels_recursively(nested_list) -> list[SolarPanel]:
36      flat_list = []
37      for item in nested_list:
38          if isinstance(item, SolarPanel):
39              flat_list.append(item)
40          elif isinstance(item, (list, np.ndarray)):
41              flat_list.extend(flatten_panels_recursively(item))
42          else:
43              print(f"Error: Found a non-SolarPanel object: {type(item)}")
44      return flat_list
```

Slika 5.23 Pomoćne metode za upravljanje ugniježđenim nizovima

```
13  def swap_duplicate(child_panels: list[SolarPanel], unused_panels: list[SolarPanel]) -> list[SolarPanel]:
14      new_child_panels: list[SolarPanel] = child_panels.copy()
15      seen_serials = set()
16      for i, panel in enumerate(child_panels):
17          if panel.serialnumber in seen_serials:
18              unused_panel = unused_panels.pop(0)
19              new_child_panels[i] = unused_panel
20          seen_serials.add(new_child_panels[i].serialnumber)
21      return new_child_panels
22
23  def has_duplicate_panels(panels: list[SolarPanel]) -> int:
24      seen_serials = set()
25      count = 0
26      for panel in panels:
27          if panel.serialnumber in seen_serials:
28              count += 1
29          seen_serials.add(panel.serialnumber)
30      return count
```

Slika 5.24 Pomoćne metode za zamjenu duplih panela u nizu

Dijete (*child*), koje je rezultat ovih izmjena, se zatim dodaje u niz *offspring* (slika 5.20 linija 158). Ovaj postupak se ponavlja sve dok se ne generira dovoljno potomaka za popunjavanje

nove generacije, odnosno dok veličina niza *offspring* ne postane jednaka razlici veličine populacije (*population_size*) i trenutne duljine populacije.

Nakon što je generirana nova generacija potomaka, ona se dodaje metodom *extend* postojećoj populaciji (slika 5.20, linija 159). Zatim se sve populacije dodaju u niz *top_sets* te se najboljih 10 populacija zadržavaju, a ostale se odbacuju, to se pomoću biblioteke *Joblib* unutar paralelne regije sortiraju funkcijom dobrote, sa slike 5.17, kao ključ za određivanje najboljeg rješenja.

Na kraju svake generacije, ispisuje se trenutna generacija i najmanji postignuti gubitak zbog nesklada, koji je prvi član niza *top_sets* (slika 5.20, print na liniji 189). Nakon što sve generacije prođu, funkcija vraća najbolju konfiguraciju iz niza *top_sets* (slika 5.20, linija 190).

6. ANALIZA REZULTATA

Razvoj genetskog algoritma nije posljednji korak izrade ovog diplomskog rada. Posljednji korak ovog diplomskog rada je pokazati rezultate genetskog algoritma, te tablično prikazati o čemu ovise različiti rezultati genetskog algoritma. Tijekom razvoja genetskog algoritma kreirane su metode koje bez genetskog algoritma sortiraju ulazne podatke, odnosno fotonaponske panele po njihovim određenim karakteristikama. Za demonstrativne svrhe korištene su tri karakteristike.

IMPP, odnosno struja na maksimalnoj snazi, je prva karakteristika koja se uzimala kao parametar za sortiranje fotonaponskih panela, iza kojeg je izašao i najbolji, odnosno najmanji postotak gubitka zbog nesklada fotonaponskih panela. Fotonaponski paneli srodnih struja imaju manji postotak gubitaka zbog manjih izmjena struja u serijskom spoju fotonaponskih panela. Drugi parametar koji je i usko vezan za IMPP, je UMPP, odnosno napon na maksimalnoj snazi, ovo sortiranje je dalo znatno veći gubitak u odnosu na IMPP, iako nije toliko loš za razliku od ostalih parametara sortiranja. Kao treći parametar uzeta je maksimalna snaga, odnosno PMPP, ali taj parametar je dao najlošiji rezultat zbog snažnog utjecaja ovisnost struje i napona na sortiranje panela. Ključno je osigurati srodnu struju među panelima u seriji, dok paneli u paraleli moraju imati srodan napon. Snaga je kombinacija tih vrijednosti pa ju je i teško izjednačiti. Za četvrti parametar uzet je serijski broj panela, odnosno SN. Ovaj parametar je dao malo bolji rezultat od maksimalne snage, iako je ovaj rezultat samo koristan za testiranje i ne bi se trebao koristiti u stvarnim konfiguracijama.

6.1 Ispis rezultata genetskog algoritma

Prilikom ispisa rezultata korištena je metoda za izračun gubitaka zbog nesklada fotonaponskih ćelija sa slike 5.20. Izrađivanje zasebnih metoda za sortiranje panela je napravljeno repetitivnom metodom, odnosno kopiranjem logika u zasebne metode, vidljivo na slici 6.1. Kreirane su tri zasebne metode preko kojih se odrađivalo sortiranje ulaznih podataka, odnosno fotonaponskih panela. Sve tri metode završavaju u metodi *doFirstConfiguration* (slika 6.1 linije 68-71).

```python
def doIMPPConfiguration(solarPanels: list[SolarPanel], L, M, config, output_file):
    sorted_solar_panels = sorted(solarPanels, key=lambda x: x.impp, reverse=True)
    doFirstConfiguration(list(sorted_solar_panels), L, M, config, output_file)

def doUMPPConfiguration(solarPanels: list[SolarPanel], L, M, config, output_file):
    sorted_solar_panels = sorted(solarPanels, key=lambda x: x.umpp, reverse=True)
    doFirstConfiguration(sorted_solar_panels, L, M, config, output_file)

def doPMPPConfiguration(solarPanels: list[SolarPanel], L, M, config, output_file):
    sorted_solar_panels = sorted(solarPanels, key=lambda x: x.pmpp, reverse=True)
    doFirstConfiguration(sorted_solar_panels, L, M, config, output_file)

def doFirstConfiguration(solarPanels: list[SolarPanel], L, M, config, output_file):
    start_configuration = initialize_first_population(solarPanels, L)
    printConfiguration(start_configuration, L, M, config)
    exportPanels(flatten_panels_recursively(start_configuration), output_file)
```

Slika 6.1 Implementacija repetitivnih metoda s obzirom na tip sortiranja

Sastavlja se ugniježđena lista listi kao i u genetskom algoritmu, samo u ovom slučaju, vidljivo na slici 6.2, radi se samo jedna konfiguracija koja se nakon toga ispisuje metodom *printConfiguration* na isti način, kroz petlju, koja prolaze kroz konfiguraciju kao iz genetskog algoritma, odnosno metodom *calculate_mismatch_loss* (slika 6.3).

```python
def initialize_first_population(
    solar_panels: list[SolarPanel],
    L: int
    ) -> list[list[SolarPanel]]:
    configuration = group_panels(solar_panels, L)
    return configuration
```

Slika 6.2 Metoda za kreiranje konfiguracije prema ulaznim parametrima L, M i N

```python
def printConfiguration(
    configuration: list[list[SolarPanel]],
    L: int, M: int, config
):
    counter = 0
    cc = 0
    total_loss = 0
    print(f"{config} sorted configuration")
    for i in range(len(configuration)//2):
        fi = i * 2
        si = fi + 1
        total_loss+= calculate_mismatch_loss(
            flatten_panels_recursively(configuration[fi])
            +flatten_panels_recursively(configuration[si]),
            L, M)
    print(f"mismatch loss: {total_loss:.9f}")
```

Slika 6.3 Metoda za ispis postotaka gubitaka svake konfiguracije

Takav jedan ispis vidljiv na slici 6.4 s obzirom na početne parametre L i M nam daje početne gubitke za svaku konfiguraciju, te za ukupan gubitak svih konfiguracija zajedno.

```
IMPP sorted configuration
mismatch loss: 0.036365059
UMPP sorted configuration
mismatch loss: 1.027889691
PMPP sorted configuration
mismatch loss: 4.008216067
SN sorted configuration
mismatch loss: 3.753169167
SHUFFLE sorted configuration
mismatch loss: 4.041301584
```

Slika 6.4 Ispis izračuna gubitaka zbog nesklada fotonaponskih panela sortiranih po parametrima IMPP, UMPP, PMPP, SN i nasumičnog sortiranja

6.2 Usporedba rješenja početnih sortiranja i genetskog algoritma

U tablici 6.1 su rezultati sa slike 6.4 koji prikazuju usporedbu rezultata gdje nije korišten genetski algoritam kako bi se bolje prikazala razlika pri korištenju genetskog algoritma. U algoritam i sortirane nizove stavljeno je 612 fotonaponskih panela, od kojih su rađene grupe od 2 podniza u paraleli, gdje je svaki podniz imao 36 fotonaponskih panela. Tako je upareno 8 grupa, od 72 panela.

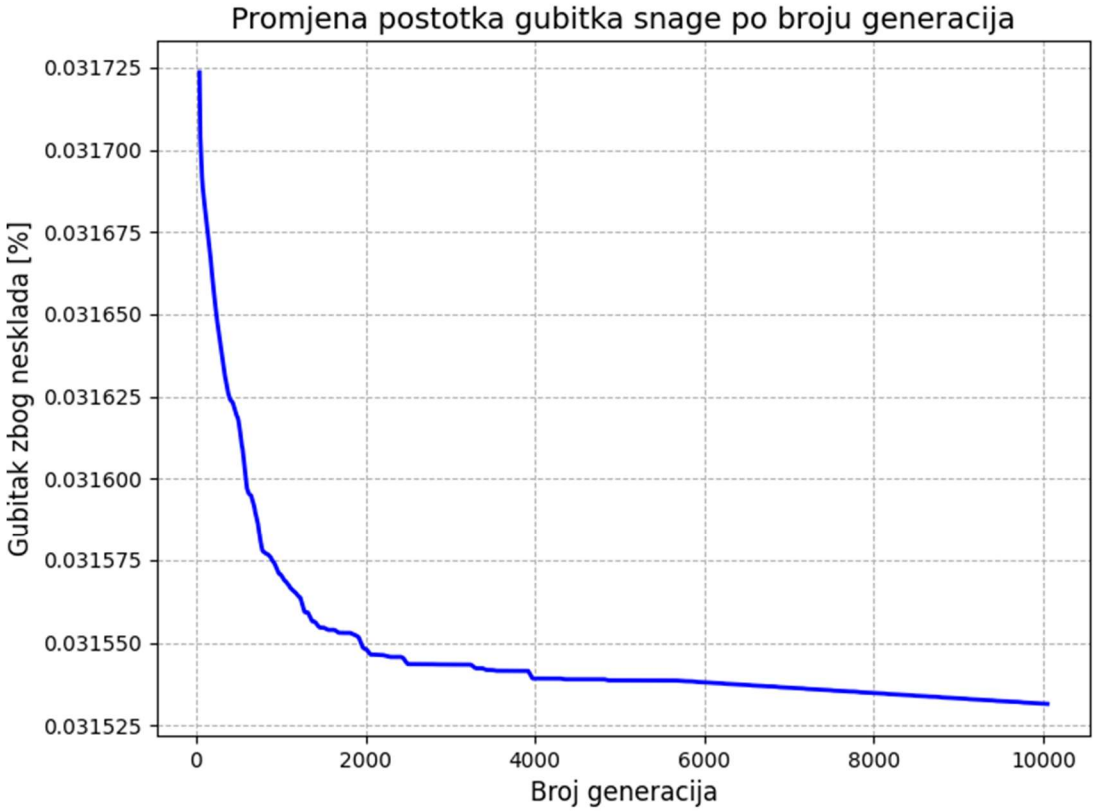
Tablica 6.1 Rezultati različitih sortiranja i obračun gubitka zbog nesklada bez genetskog algoritma i s genetskim algoritmom

	IMPP	**UMPP**	**PMPP**	**SN**	**SHUFFLE**	**GA**
ΔP_ε	0,0363651	1,0278897	4,0082161	3,7531692	4,041301584	0,03153142
Ratio	1,1533	32,5989	127,1182	119,0305	128,167	1

Genetskim algoritmom je moguće nakon određenog broja generacija pronaći konfiguraciju bolju od početne konfiguracije sortirane po struji na maksimalnoj snazi. Zato je za testiranje ovoga pokrenuta velika petlja, odnosno genetski algoritam je pokrenut s deset tisuća generacija, kako bi se s vremenom pronašla bolja kombinacija od početne. Nakon izvedbe programa od deset tisuća generacija, gdje je svaka generacija imala tisuću populacija uz stopu mutacija od 15%, pronađen je rezultat 128,17% bolji od rezultata gdje su fotonaponski paneli nasumično odabrani u grupe. Također u odnosu na rezultat gdje je početni niz panela sortiran po struji na

maksimalnom naponu, ovaj rezultat je samo 1,1533% bolji, iako u odnosu na ostale rezultate koji su vidljivi u tablici 6.1 je znatno bolji.

Gubitak snage zbog nesklada panela u grupama tijekom rada genetskog algoritma, odnosno kroz generacije prikazan je na slici 6.5.



Slika 6.5 Graf promjena postotka gubitka snage po broju generacija.

Na grafu je vidljivo kako u prvim generacijama promjene su znatne, u odnosu na početnu vrijednost. S druge strane, na grafu se vidi kako nakon 5000 generacija algoritam krene stagnirati, odnosno teže pronalazi bolje iteracije, ali pronalazi male promjene s vremenom. Sve dok jednom ne dođe do optimuma, odnosno do točke gdje se ne može više smanjiti nesklad.

7. ZAKLJUČAK

U ovom diplomskom radu istražena je problematika gubitaka zbog nesklada u fotonaponskim nizovima. Naglasak je na primjenu genetskih algoritama kao metode optimizacije. Kroz analizu različitih parametara, poput veličine populacije, broja generacija i stope mutacije, pokazano je kako genetski algoritam može znatno smanjiti gubitke zbog nesklada. Optimizacija pomoću genetskog algoritma omogućava rješavanje složenih problema koji nastaju zbog varijabilnosti karakteristika fotonaponskih panela i vanjskog utjecaja poput starenja i okolišnih faktora.

Jedan od glavnih rezultata ovog rada je pokazuje kako povećanje veličine populacije i broja generacija ne donosi linearno smanjenje gubitaka zbog nesklada fotonaponskih nizova. S druge strane, promjena stope mutacije pokazala se kao važan faktor, gdje čak i male promjene mogu značajno utjecati na konačni rezultat. Jedino ograničenje za ovakve izmjene je vremensko ograničenje, jer porastom vrijednosti broja generacija i populacija, raste i trajanje izvođenja programa. Također uočene su promjene rezultata ovisno o sortiranju početnih populacija u inicijalnoj metodi. Na ovaj način pronađeni su najbolji rezultati cijele analize, jer je otkriveno kako početna populacija, sortirana po struji na maksimalnoj snazi (*IMPP*), rezultira boljim rezultatima. Tijekom analize rezultata, nisu sva rješenja, odnosno sve grupe nisu imale slične postotke gubitaka. Ovi slučajevi ukazuju na potrebu za dodatnim pristupima optimizaciji, kao što je uvođenje elitizma, gdje bi se najslabije konfiguracije prve križale i mutirale dok bi se konfiguracije bliže prosjeku čuvale za daljnje generacije. Ovaj pristup bi u budućnosti poboljšao rezultate genetskog algoritma i pružio smjernice za daljnji rad na području optimizacije fotonaponskih nizova u stvarnim uvjetima pomoću genetskih algoritama.

Ovaj diplomski rad predstavlja temelj za daljnja istraživanja na području grupiranja i sortiranja fotonaponskih panela s ciljem optimizacije energetske učinkovitosti. Genetski algoritmi su se pokazali kao pouzdan alat za smanjenje gubitaka zbog nesklada fotonaponskih nizova te predstavljaju obećavajuće rješenje za izazove u fotonaponskim sustavima u stvarnim uvjetima.

SAŽETAK

U ovom diplomskom radu obrađena je problematika gubitaka električne energije zbog nesklada fotonaponskih ćelija. Razrađene su karakteristike fotonaponski ćelija i također kako se izračunavaju gubitci nastali zbog nesklada fotonaponskih ćelija. Objašnjena je uloga i funkcionalnost genetskih algoritama. Razvijen je genetski algoritam, u programskom jeziku *Python,* za grupiranje i sortiranje fotonaponskih panelima, korištenjem matematičke analitike i karakteristike fotonaponskih panela. Osim genetskog algoritma izrađen je temeljiti ispis izlaznih podataka i rezultata. Napravljena je analiza grupiranih podataka sortiranih genetskim algoritmom. U analizi pokazano je kako različiti ulazni parametri daju različite i bolje odnosno lošije rezultate.

Ključne riječi: energetika, genetski algoritam, grupiranje, fotonaponski paneli, sortiranje

ABSTRACT

This final thesis addresses the issue of mismatch loss of power in photovoltaic cells. The characteristics of a photovoltaic cell are described. Also, it is described how to calculate the mismatch losses caused by connected photovoltaic cells. The role and functionality of genetic algorithms are explained. Using Python, a genetic algorithm was developed for grouping and sorting solar panels, using mathematical analysis and the characteristics of the solar panels. In addition to the genetic algorithm, a thorough printout of the output data was created. An analysis was performed on the grouped data sorted by the genetic algorithm. In the analysis it is shown that different entry parameters created different and better or worse results.

Key words: energetics, genetic algorithm, grouping, solar panels, sorting

LITERATURA

[1] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, A. J. Olson, Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function, SAD, 1998.

[2] Célio Manso de Azevêdo Juniora , Enderson Luiz Pereira Júnior a, Tullio Mozart Pires de Castro Araujoa , Marcos dos Santosb , Carlos Francisco Simões Gomesa , Daniel Augusto de Moura Pereirac, Ordering of Solar Photovoltaic Panels using the MEREC-SPOTIS Hybrid Analytical Model, Brazil, 2023

[3] The Go Greena Team, How Solar Panels Work – A Guide For Dummies, What Is A Solar Cell? [online], Go Greena UK, 20.08.2012, dostupno na: http://gogreena.co.uk/how-solar-panels-work-a-guide-for-dummies/ [16.9.2024.]

[4] H. Annie, Istraživanje tankoslojnih solarnih panela: učinkovitost, cijena i usporedba [online, automatski prijevod stranice], Shielden Channel, 14.03.2014. Kina, dostupno na:https://hr.shieldenchannel.com/blogs/solar-panels/thin-film-solar-panels [16.9.2024]

[5] Ecoprogetti , What is the raw material that composes a photovoltaic module, slika [online], Ecoprogetti, 2014, UK, dostupno na: https://ecoprogetti.com/the-structure-of-photovoltaic-module/ [16.9.2024.]

[6] A. Kirin, F. Žugčić, Princip rada i primjena fotonaponskih ćelija, Karlovac, Rujan, 2018.

[7] J. Webber, E. Riley, Mismatch Loss Reduction in Photovoltaic Arrays as a Result of Sorting Photovoltaic Modules by Max-Power Parameters, Švedska i SAD, 12.5.2013.

[8] K. Jäger, O. Isabella, Arno H.M. Smets, René A.C.M.M., van Swaaij, Miro Zeman, Solar Energy, Fundamentals, Technology, and Systems, Delft University of Technology, Nizozemska, 2014.

[9] Leksikografski zavod Miroslav Krleža, Beerov zakon, Hrvatska Enciklopedija, 2024, dostupno na: https://www.enciklopedija.hr/clanak/beerov-zakon [16.9.2024.]

[10] N.D. Kaushikaa, Anil K. Raib, An investigation of mismatch losses in solar photovoltaic cell networks, India, 12.8.2005.

[11] V. Stornelli, M. Muttillo, T. de Rubeis, I. Nardi, A New Simplified Five-Parameter Estimation Method for Single-Diode Model of Photovoltaic Panels, energies MDPI, Italija, 2019

[12] B. Prasad Koirala, B. Sahan, N. Henze, Study On Mpp Mismatch Losses In Photovoltaic Applications, German Academic Exchange Service (DAAD), Hamburg, 18.11.2009.

[13] J. McCall, Genetic algorithms for modelling and optimisation, School of Computing, Robert Gordon University, Aberdeen, 7.7.2004.

[14] M. Golub, Genetski Algoritmi prvi dio, Zagreb, 27.9.2004. dostupno na: https://www.zemris.fer.hr/~golub/ga/ga.html [16.9.2024.]

[15] Numpy paket, dokumentacija, 2024, dostupno: https://numpy.org/doc/stable/index.html [16.9.2024]

[16] Primjer korištenja Numpy metode za izračun srednje vrijednosti niza, dostupno na: https://numpy.org/doc/stable/reference/generated/numpy.ma.masked_array.mean.html#numpy.ma.masked_array.mean [16.9.2024.]

[17] Primjer korištenja Numpy metode za izračun varijance niza, dostupno na: https://www.codecademy.com/resources/docs/numpy/built-in-functions/variance [16.9.2024.]

[18] T. Matić, D. Blažević, D. Bajer, I. Vidović, Differential Evolution Based Mismatch Loss Optimisation in Photovoltaic Arrays, Osijek, 2018

[19] C. E. Chamberlin, P. Lehman, J. Zoellick, G. Pauletto, Effects of Mismatch Losses in Photovoltaic Arrays, SAD, 1995.

[20] L. L. Bucciarelli Jr., Power Loss in Photovoltaic Array Due to Mismatch in Cell Characteristics, SAD, 1979.

[21] N. Agarwal, A. Agarwal, Mismatch Losses in Solar Photovoltaic Array and Reduction Techniques, India, 2014.

[22] D. Sera, R. Teodorescu, P. Rodriguez, PV panel model base on datasheet values, Danska i Španjolska, 2007

PRILOZI

Kod programa dostupan u GitHub repozitoriju na linku: https://github.com/Zokky2e/mismatch-loss-genetic-algorithm

Spomenuti dokumenti (.xsls i .csv) i izvorni kod programa dostupni su na CD-u u prilogu.