**Universitatea de Medicină, Farmacie, Științe și Tehnologie „George Emil Palade” din Târgu Mureș**

**Facultatea de Inginerie și Tehnologia Informației**

**Departamentul de Studii Masterale: Programul de Studiu „Inteligență Artificială”, anul 1**

**NUME: Manoila Bogdan Liviu**

**Titlu**

**Optimizarea plasării stațiilor de bază într-o rețea celulară folosind algoritmi genetici**

**1.Rezumat**

Odată cu creșterea constantă a traficului de date mobile, devine tot mai importantă optimizarea plasării stațiilor de bază LTE pentru a asigura o acoperire cât mai bună și servicii de calitate. Scopul acestui proiect a fost să identific o selecție optimă a stațiilor dintr-un set real de date, astfel încât să maximizăm acoperirea zonei studiate, folosind cât mai puține stații active ,reducând in acest fel consumul de energie, dar și eventualele intervenții umane la SITE-uri în caz de deranjamente.

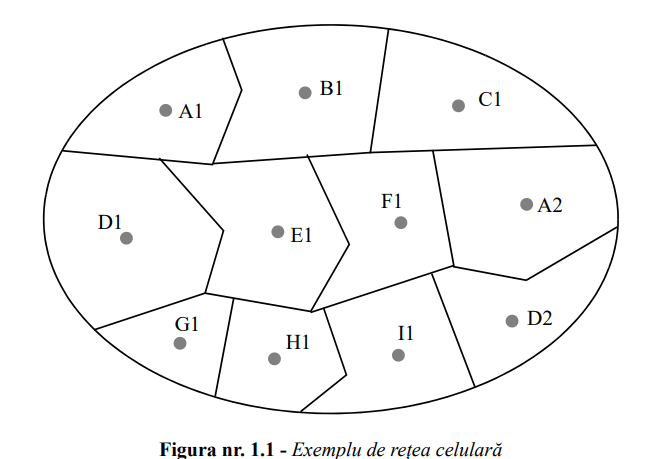
Am aplicat un algoritm genetic implementat în Python, cu ajutorul bibliotecii DEAP. Fiecare soluție generată de algoritm este un vector binar, care indică ce stații sunt active. Funcția de evaluare combină distanța medie între punctele dintr-o grilă fixă și stațiile active, adăugând penalizări pentru zonele neacoperite și pentru numărul mare de stații activate. Astfel, algoritmul caută un echilibru între acoperire și eficiență.

Populația evoluează prin selecție turneu, încrucișare în două puncte și mutație flipbit. Datele provin dintr-un fișier CSV cu coordonatele geografice și raza de acoperire a stațiilor LTE din România, ceea ce face ca testarea să fie relevantă pentru condiții reale.

Rezultatele obținute arată că este posibil să reducem semnificativ numărul de stații active fără a compromite acoperirea. Graficul rezultat ilustrează clar pozițiile stațiilor alese și punctele de testare, confirmând eficiența metodei.

În concluzie, algoritmii genetici pot fi o soluție practică și flexibilă pentru optimizarea rețelelor LTE, putând fi adaptați ușor în funcție de diferite condiții și cerințe specifice.

**2. Introducere**

* Problema abordată: Distribuția ineficientă a stațiilor LTE poate duce la acoperire slabă, interferențe sau redundanță.
* 

În principiu, distribuţia emiţătoarelor nu trebuie să fie regulată, iar celulele nu trebuie să aibă o formă anume. Celulele notate cu litere diferite utilizează seturi diferite de canale pentru evitarea interferenţelor izocanal. Deci, o celulă are şi semnificaţia suplimentară că este zona în care este utilizat pentru comunicaţie un set dat de canale. Celule suficient de îndepărtate (precum, de exemplu, A1, A2 şi respectiv, D1, D2) pot utiliza simultan acelaşi set de canale. Prin reutilizarea spaţială a frecvenţelor o reţea celulară de comunicaţii mobile poate susţine simultan un număr de comunicaţii mult mai mare decât numărul total de canale alocate. Factorul de multiplicare este o funcţie de mai mulţi parametri ai reţelei, dar, în principal, de numărul total de celule. Scopul principal al definirii celulelor într-o reţea de comunicaţii mobile este acela de a delimita zonele în care fie un set dat de canale, fie un emiţător dat sunt utilizate de mobile cel puţin preferenţial, dacă nu exclusiv. O limitare geografică rezonabilă a utilizării unui canal este impusă de necesitatea evitării interferenţei izocanal. O structură de celule rezultată din proiectare se implementează practic prin poziţionarea corespunzătoare a emiţătoarelor, alegerea convenabilă a diagramelor de radiaţie a antenelor şi prin alegerea corectă a emiţătorului ce serveşte un apel de comunicaţie. O celulă reprezintă aria de acoperire a unui emiţător. Dacă acesta utilizează o antenă omnidirecţională şi terenul este perfect plan, atunci celula rezultă de formă circulară. Deşi considerentele de propagare, recomandă forma circulară a celulelor, aceasta este improprie pentru proiectare, deoarece un ansamblu de cercuri conduce la ambiguităţi, cu puncte ce aparţin fie mai multor celule, fie la nici o celulă.[1]

* Motivație: Plasarea optimă a stațiilor de bază în rețelele LTE este o problemă complicată, mai ales când numărul de stații și suprafața de acoperit cresc. Metodele clasice care încearcă să verifice toate combinațiile devin prea lente și greu de folosit în astfel de situații. De aceea, am apelat la metaheuristici, cum sunt algoritmii genetici, care pot explora eficient un spațiu foarte mare de posibile soluții fără să le testeze pe toate. Acești algoritmi găsesc rapid variante bune, aproape optime, reducând timpul și resursele necesare pentru planificarea rețelei și asigurând o acoperire mai bună pentru utilizatori.

**3.Literatură scurtă**

**Studiu de caz 1: Jianpo Li , Jinjian Pang \*, Xiaojuan Fan , Optimization of 5G base station coverage based on self-adaptive mutation genetic algorithm.**

Articolul tratează problema optimizării amplasării și configurării stațiilor de bază 5G, cu scopul de a maximiza acoperirea și eficiența rețelei. Deoarece 5G vine cu provocări specifice — o densitate mare de stații, frecvențe înalte care acoperă distanțe mici și necesitatea unei acoperiri continue — această problemă devine destul de complexă și are multe fațete.  
Pentru a aborda această situație, autorii propun un algoritm genetic care folosește o rată de mutație auto-adaptativă, adică rata de mutație se ajustează pe parcursul procesului de căutare. Astfel, în fazele inițiale, algoritmul explorează mult, încercând soluții diverse pentru a evita să rămână blocat în soluții suboptime, iar pe măsură ce avansează, scade rata de mutație pentru a se concentra pe îmbunătățirea și rafinarea celor mai bune soluții găsite.

**Modelarea problemei**

Spațiul de căutare este reprezentat prin vectori ce codifică pozițiile și parametrii stațiilor de bază prin coordonate, putere de emisie.

**Reprezentarea soluțiilor (codificare):**  
Fiecare individ în populație reprezintă o configurație a pozițiilor stațiilor de bază, codificate ca un set de coordonate reale (xi,yi)

**Inițializarea populației:**  
Se generează aleator un set de poziții inițiale pentru stațiile de bază în zona planificată.

**Funcția de fitness:**  
Se folosește o funcție care combină costul construcției și acoperirea zonei (formula obiectivului general). Aceasta evaluează cât de bună este o soluție (poziția stațiilor).

**Selecția:**  
Se aplică selecția prin metoda ruletei: soluțiile cu fitness mai bun au o probabilitate mai mare să fie selectate pentru reproducere.

**Încrucișarea (crossover):**  
Se folosește încrucișarea aritmetică între doi părinți pentru a produce doi copii noi, combinând aleator coordonatele părinților.  
Aceasta crește diversitatea populației.

**Mutația adaptivă:**  
Pentru a evita blocarea în optimi locale, se aplică o mutație adaptivă care modifică ușor coordonatele indivizilor, în funcție de poziția lor față de limitele admise (sus/jos

**Iterare:**  
Pașii de selecție, încrucișare și mutație se repetă până când se obține o soluție satisfăcătoare sau se atinge un număr maxim de generații.

**Rezultate:**

Simulare s-a facut in MATLAB. Orașul Jilin , suprafata de 25 km patrati va fi ales ca zonă de studiu pentru optimizarea acoperirii stațiilor de bază. Stațiile de bază din zona planificată vor folosi antene direcționale, iar aria de acoperire va avea o formă ideală circulară. Zonele de planificare sunt împărțite în trei tipuri: zone urbane dense si zone urbane generale

1. **Zona urbana densa**

Numărul stațiilor de bază în zonele urbane dense variază între 48 și 62. Atunci când numărul de stații este 56, valoarea fitness-ului funcției obiectiv atinge minimul, rezultând harta optimă de acoperire a stațiilor de bază, care reduce costurile și asigură o acoperire largă.

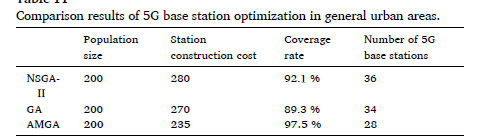
După optimizarea acoperirii stațiilor 5G, sunt incluse 43 de stații noi 5G și 13 stații 3G/4G partajate, ceea ce duce la o rată de acoperire a stațiilor de 92,5 %.

Algoritmul propus în această lucrare reduce costul construcției locațiilor cu cel puțin 7 %, îmbunătățește acoperirea cu cel puțin 3,3 % și reduce numărul de stații cu cel puțin 7 % comparativ cu celelalte algoritme.

**B.Zona urbana generala**

In această zonă există 32 de stații 3G/4G partajate. numărul stațiilor de bază în zonele urbane generale variază între 20 și 36. Cand numarul de statii de baza ajunge la 28 , valoarea fitness a funcției obiectiv atinge valoarea optimă, ceea ce înseamnă reducerea costurilor de construcție a stației de bază, în timp ce se obține o acoperire largă.

Acest algoritm este comparat cu algoritmul genetic tradițional și cu algoritmul de optimizare multi-obiectiv NSGA-II.



**Concluzia:** algoritmul propus în această lucrare reduce costul de construcție a stațiilor cu cel puțin 13%, îmbunătățește acoperirea cu cel puțin 5,4% și reduce numărul de stații de bază cu cel puțin 17,6% comparativ cu celelalte algoritme.

**Studiu de caz 2:** **Proposed Base-Station Location Optimization with Genetic Algorithm Scheme for Lte Network Radio Planning Shahad Nafea Jaafar, Ekhlas Kadum Hamza and Viean Abdulmuhsin Al-Salihi Control & System Engineering Department, University of Technology, Iraq**

**Descriere tehnică**

În această lucrare, GA propus este utilizat pentru a minimiza suprapunerea și consumul de energie în fazele de planificare și operare a rețelei. Aceasta se realizează ținând cont de cerințele utilizatorilor în funcție de diferitele perioade cu încărcare de trafic si de interferențe.

Modelul rețelei este împărțit în trei etape; prima etapă este proiectată pentru rețeaua din cele 24 de locații selectate în cadrul orașului Bagdad, etapa a doua optimizează designul rețelei folosind un algoritm genetic (GA), iar etapa a treia prezintă planificarea rețelei LTE. În funcție de profilurile clienților, rețeaua LTE poate genera o populație de utilizatori mobile.

**Planificarea rețelei LTE**  
Populația determină amplasamentele distribuite în această fază. Cele mai bune locații pentru instalarea turnurilor rețelei LTE sunt alese folosind o abordare de optimizare bazată pe algoritmul genetic. Locațiile sunt selectate astfel încât să reducă interferențele, iar utilizatorii finali să beneficieze de o acoperire cât mai bună. Procentul interferențelor este determinat în funcție de numărul de utilizatori acoperiți.

Studiul analizeaza 24 de locații selectate din cadrul orașului Bagdad care însumează o suprafață de 1078 km patrati.

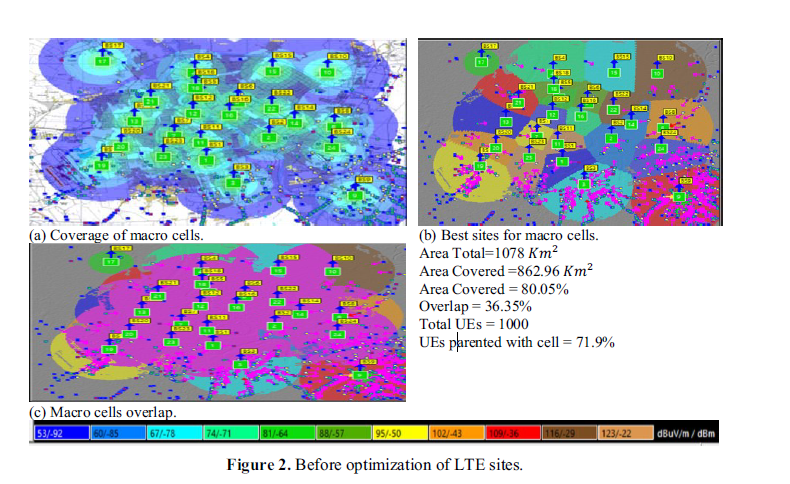
Numărul locațiilor ce vor fi stabilite este determinat de condițiile de acoperire. Simularea, ca metodă de evaluare a rețelei planificate, este realizată pe baza numărului de utilizatori deserviți și a capacității rețelei.

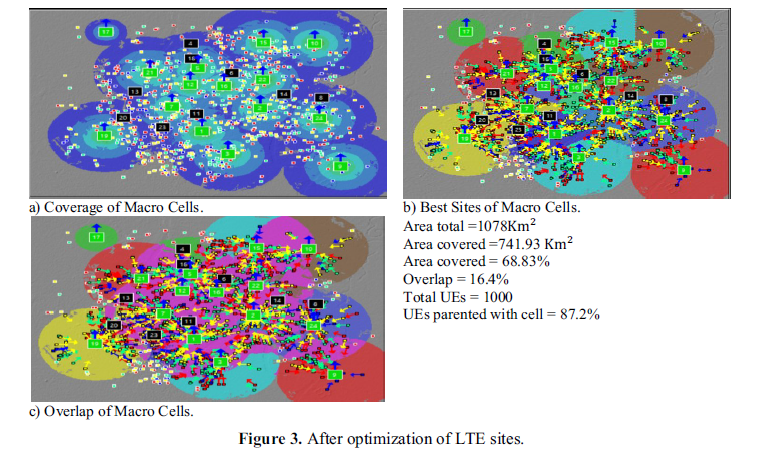
Autorii propun utilizarea unui algoritm genetic (GA), care este un algoritm evolutiv inspirat din mecanismele selecției naturale, pentru a explora eficient spațiul de soluții și a găsi poziții optime pentru stațiile de bază.

Interferența este determinată în funcție de numărul de utilizatori acoperiți. Se identifica puterea maximă recepționată de un utilizator, aceasta fiind reprezentată prin funcția de fitness (stația de bază deservitoare).

Procesul de verificare implică oprirea și repornirea unor seturi de stații de bază (BS), ceea ce conduce la eliminarea celor mai slabe.. Evaluarea planificării rețelei se face în funcție de numărul de utilizatori deserviți și capacitatea rețelei.

Prin utilizarea algoritmilor GA, nouă locații sunt eliminate. Suprapunerea, inițial de 36.35 s-a redus cu 19,95%, folosind cincisprezece locații rămase.





Reprezentarea cromozomilor GA corespunde unor configurații posibile de locații a statiilor de baza.

Funcția obiectiv combină mai mulți indicatori cheie: acoperirea geografică, calitatea semnalului (SINR), și minimizarea interferențelor.

Operații specifice GA, precum selecția, crossover-ul și mutația, sunt adaptate pentru a menține diversitatea populației și pentru a evita blocarea în soluții suboptime.

Modelul ia în considerare caracteristicile specifice ale rețelelor LTE, cum ar fi tipurile de antene, puterea de transmisie, și condițiile de propagare radio.

.

**Puncte cheie**

* **Problema studiată:** Optimizarea poziționării stațiilor de bază în rețele LTE pentru maximizarea acoperirii și performanței radio.
* **Metodologie:** Algoritm genetic (GA) adaptat pentru planificarea radio LTE.
* **Reprezentare:** Cromozomi reprezintă configurații posibile de locații BS.
* **Functia fitness:** ia in calcul puterea semnalului receptionat de utilizatori , interferențele dintre stații, numărul de utilizatori acoperiți eficient, gradul de suprapunere dintre acoperiri.
* **Funcția obiectiv:** Maximizează acoperirea și SINR, minimizează interferența și costurile.
* **Operatori GA**: Selecție, crossover, mutație pentru explorare eficientă și evitarea blocajului. Probabilitate scazuta(0.05).Elitism pentru a pastra cele mai bune solutii.
* **Rezultate:** GA propus oferă soluții superioare față de metodele clasice în simulări urbane, asigurând o acoperire mai bună și reducerea interferențelor.Se reduce numarul de statii de baza.
* **Contribuție:** Demonstrează aplicabilitatea algoritmilor evolutivi pentru probleme complexe de planificare radio LTE, oferind un instrument practic pentru operatorii de rețele.

1. **Obiective specifice**

**Întrebări de cercetare**:

* + Cum putem optimiza plasarea stațiilor LTE pentru o acoperire mai eficientă?

Plasarea optimă a stațiilor LTE/5G poate fi obținută prin aplicarea unui algoritm genetic (ar fi varianta si cu mutație auto-adaptativă )care ajustează automat parametrii algoritmului pe parcurs pentru a maximiza acoperirea și performanța rețele.

* + Ce performanțe oferă AG comparativ cu distribuția reală?

Plasarea stațiilor de bază LTE poate fi optimizată pentru o acoperire mai eficientă prin utilizarea algoritmilor genetici (GA), care permit găsirea unei configurații optime ce reduce interferențele și maximizează acoperirea utilizatorilor finali.

**4.1. Prelucrarea datelor reale LTE pentru România**

* Datele au fost preluate din sursa OpenCellID, care conține coordonate geografice (latitudine, longitudine) și informații despre stațiile de bază LTE, inclusiv raza de acoperire și semnalul mediu.
* Setul de date a fost procesat folosind Pandas, păstrând atributele relevante (lat, lon, range) pentru evaluarea acoperirii rețelei.
* Stațiile au fost convertite într-un array Numpy pentru optimizarea procesării în cadrul algoritmului genetic.

**4.2.Implementarea unui Algoritm Genetic (AG) pentru selecția optimă a stațiilor**

* AG-ul a fost implementat în Python 3.10 folosind biblioteca DEAP.
* Fiecare individ reprezintă un cromozom binar de lungime egală cu numărul total de stații LTE – 1 indică o stație activă, 0 una inactivă.
* Generarea grilei de test
  + am construit o grilă de puncte (grid\_points) pe baza valorilor minime și maxime de latitudine și longitudine
  + Fiecare punct din grilă este folosit pentru a verifica dacă este acoperit de vreo stație activă.
* Funcția de fitness combină:
* Evaluează fiecare individ (o combinație de stații active): Verifică pentru fiecare punct dacă e acoperit de cel puțin o stație activă.
* Calculează: **distanța totală** (sumă de distanțe minime către stații acoperitoare), **penalizări**: dacă punctele NU sunt acoperite, respectiv dacă se folosesc prea multe stații
* Returnează un scor (cu cât mai mic, cu atât mai bine)
* Configurarea DEAP (algoritm genetic):
* Am definit tipul de fitness (FitnessMin) și tipul de individ (listă de 0 și 1).
* Am configurat operatorii: mate: încrucișare în două puncte, mutate: fiecare bit are 5% șansă să se inverseze. select: turneu cu 3 indivizi
* Parametrii folosiți:
* Populație de 50 indivizi
* Se rulează algoritmul timp de 50 generații.Pentru fiecare generatie se evalueaza fitness-ul: adica se verifica ce puncte din grila sunt acoperite de statiile active.Se calculeaza distant fata de punctele acoperite. Iar pentru punctele neacoperite si numar mare de statii se aplica penalizari. La final se obtine un scor fitness pentru fiecare individ.(mai mic e mai bine)
* Selecție – pentru urmatoarea generatie se aleg indivizi printr-un turneu.Se aleg aleator 3 indivizi,castiga cel cu fitness cel mai bun.Se formeaza o noua populatie prin selectie.Se repeat pasii de 50 de ori , ca la final sa obtin 50 de copii selectati.
* Încrucișare- acei 50 de copii rezultati vor suferi o amestecare(imperechere) , adica amestec genele pentru a produce variatie prin **cxTwoPoint** crossover în două puncte — o metodă clasică de recombinare. mutație, evaluare. Iau perechi de cate 2 indivizi si ii combin cu o probabilitate de 0.5.Aleg aleator 2 pozitii din vectorul binar si se inverseaza segmentele dintre aceste pozitii. Rezulta 2 copii cu gene amestecate.Ceilati 50% din parinti sunt copiati ca atare in noua generatie fara modificari.
* Dupa ce am generat cei 50 de indivizi ai noii generații (fie prin crossover, fie copie directă), fiecare dintre acești indivizi este supus posibilității de mutație genetică. Prin **mutFlipBit** fiecare bit (adică fiecare element din vectorul binar al unui individ — deci fiecare stație) are o probabilitate individuală **indpb=0.05** (adică 5%) de a fi inversat. (0->1 , 1->0).Acest lucru ajuta să se exploreze mai bine spațiul soluțiilor și să evite blocarea în minime locale (soluții proaste care par bune doar pe moment).

Dar nu toti indivizii din noua generatie vor suferi mutatie , parametrul **mutpb=0.2** spune doar ca 20% dintre indivizii generatiei vor trece prin mutatie.

* colectare statistici (fitness mediu și minim). Pentru fiecare generatie se calculeaza si se inregistreaza statistici importante despre fitness-ul indivizilor. Astfel ca pentru fiecare individ din populație iau valoarea fitness-ului (este un tuplu cu un singur element, scorul de optimizare) si obtin fitness-ul minim al populatiei pe care il salvez. Deasemenea calculez media fitness-ului fiecarei populatii.
* păstrarea celui mai bun individ în HallOfFame. – cea mai buna solutie se pastreaza in HallOfFrame. După fiecare generație, algoritmul verifică dacă în populația curentă există un individ mai bun decât cel păstrat în hof.Daca da, este inlocuit cu individul mai bun gasit,daca nu , ramane neschimbat.E util pentru a nu pierde solutii bune in unele generatii. Dupa ce algoritmul trece prin toate generatiile hof retine cea mai bună configurație a stațiilor active/inactive găsită pe tot parcursul evoluției
* Afișarea rezultatului
  + Se afișează câte stații sunt active în soluția optimă.
  + Se plotează toate stațiile, stațiile active selectate și grila de test

**4.3. Evaluarea performanței soluției**

* Performanța a fost evaluată pe baza evoluției valorii fitness în timp și identificarea celei mai bune configurații de stații active.
* S-a utilizat o vizualizare grafică pentru a reprezenta:
  + Toate stațiile LTE (în gri),
  + Stațiile selectate ca active (în roșu),
  + Punctele din grilă folosite pentru testarea acoperirii (în albastru deschis).
* Soluția finală optimizează acoperirea geografică cu un număr minim de stații, reducând redundanțele și costurile.

**4.4. Resurse și Literatură de Specialitate**

* Sursa primară de date: OpenCellID (<https://www.opencellid.org>)
* Bibliotecă Python folosită: DEAP – Distributed Evolutionary Algorithms in Python

**5.Compararea algoritmilor genetici cu metodele clasice**

**5.1.Avantaje ale algoritmilor genetici (AG):**

Algoritmii genetici reprezintă o metodă puternică și versatilă pentru descoperirea soluțiilor optime în fața provocărilor complexe. De la începuturile lor, acești algoritmi au evoluat și au contribuit semnificativ la numeroase domenii ale informaticii și tehnologiei. Importanța lor în rezolvarea problemelor reale și inovarea tehnologică este evidentă și va continua să crească în viitor. Concluzionând, algoritmii genetici nu doar că transformă modul în care abordăm problemele complexe, ci influențează și viitorul tehnologiei, având potențialul de a crea soluții inovatoare pentru provocările secolului XXI.[2].

Reducerea costurilor rețelei: AG permite minimizarea costurilor totale ale rețelei prin identificarea amplasamentelor optime pentru stațiile de bază, asigurând în același timp acoperirea și capacitatea necesare.

Îmbunătățirea acoperirii și capacității: Prin optimizarea amplasamentelor, AG contribuie la o mai bună acoperire a utilizatorilor și la o capacitate sporită a rețelei, răspunzând cerințelor de trafic în creștere.

Adaptabilitate la distribuția neuniformă a utilizatorilor: AG este eficient în scenarii în care utilizatorii sunt distribuiți neuniform, permițând o planificare flexibilă și adaptată a rețelei.[3]

**5.2.Dezavantaje ale algoritmilor genetici (AG):**

Algoritmii genetici necesită evaluarea repetată a unei funcții de fitness într-un număr mare de generații și pentru mulți indivizi.

Deși GA sunt concepuți să evite minimele locale, în practică pot converge prematur dacă diversitatea populației scade prea repede.

Performanța algoritmului depinde în mare măsură de alegerea parametrilor precum dimensiunea populației, rata de mutație și rata de încrucișare. Stabilirea valorilor optime pentru acești parametri poate fi dificilă și necesită adesea experimente extensive. Alegerea nepotrivită a acestor valori poate duce la convergență prematură sau explorare ineficientă

Posibilitatea convergenței premature: Algoritmii genetici pot converge rapid către soluții suboptime dacă diversitatea populației nu este menținută, ceea ce duce la blocarea în minime locale și la pierderea potențialului de a găsi soluții mai bune.[4]

**5.3.Avantaje ale metodelor clasice (ex: grid search, clustering):**

Grid Search:

* Simplu de implementat: nu necesită cunoștințe profunde despre funcția obiectivă sau distribuția datelor.
* acoperă în mod regulat întreg spațiul căutării, evitând lacune în explorare.
* rezultatele sunt determinate clar de grila fixă, ușor de replicat.
* Programe Atoll - Simulare a acoperirii radio, capacității, propagării semnalului, optimizarea amplasării celulelor, analiza interferențelor.

**Ranplan Wireless** - platformă completă pentru planificare, proiectare și optimizare a rețelelor celulare, inclusiv 5G. Include simulări avansate și modele radio complex

**NS-3 (Network Simulator 3)-** Simulator open-source de rețele care permite simularea traficului și a performanței rețelelor wireless. Poate fi extins cu scripturi personalizate pentru optimizarea poziționării.

**OpenCellular și OpenAirInterface -** Proiecte open-source pentru prototipare și testare de rețele celulare, utilizate mai mult în cercetare și învățare. Poate fi integrat cu algoritmi de optimizare dezvoltați separat**.**

Clustering (de exemplu, K-means, DBSCAN):

* Clustering-ul, de exemplu, reduce dimensiunea problemei prin gruparea utilizatorilor, ceea ce ajută la identificarea pozițiilor centrale pentru stații..
* Stațiile pot fi poziționate în jurul centrelor clusterelor pentru o acoperire optimizată a grupurilor de utilizatori.
* Funcționează bine și pe seturi mari de date, oferind soluții rapide.
* Flexibilitate: Poate lucra cu diverse metrici de distanță și poate fi combinat cu alte metode (ex: grid search pentru rafinare).

**5.4.Dezavantaje ale metodelor clasice:**

* Pe măsură ce numărul de stații sau dimensiunea zonei crește, metodele clasice devin ineficiente din cauza complexității crescute și a timpului de calcul.
* Rezultatele pot depinde puternic de setările inițiale (ex: numărul și poziția centrelor în clustering), ceea ce poate duce la soluții suboptime.
* Metodele clasice nu se descurcă bine în cazul funcțiilor obiectiv non-liniare, cu multiple constrângeri sau în prezența datelor zgomotoase și dinamice.
* Nu pot ajusta în mod dinamic strategiile de căutare sau soluțiile, spre deosebire de metodele evolutive sau metaheuristice. [5]

**6.Metodologie**

* Metaheuristica folosită: Algoritm Genetic (AG) implementat în Python 3.10, folosind biblioteca DEAP
* Codificare: Cromozomi reprezintă vectori binari (liste de 0 și 1) cu lungime egală cu numărul total de stații LTE (N). Valoarea 1 indică stația activă, 0 inactivă
* Funcție de fitness:

Se minimizează o funcție care combină:

* Distanța medie între punctele dintr-o grilă regulată de verificare (grid\_points) și cea mai apropiată stație activă, pentru a maximiza acoperirea spațială.
* Penalizări pentru punctele neacoperite (care sunt prea departe de orice stație activă).
* Penalizare proporțională cu procentul de stații activate, pentru a evita selectarea unui număr prea mare de stații.
  + Operatori genetici:
  + selecție turneu (selTournament) cu mărimea turneului
  + Încrucișare două puncte (cxTwoPoint).
  + Mutație prin flipbit cu probabilitate pe bit de 5% (mutFlipBit).
* Parametri: Populație de 50 indivizi.50 de generații. Probabilități de încrucișare 0.5 și mutație 0.2.
* Mediu: Python 3.10, biblioteci DEAP, NumPy, Pandas, Matplotlib.
* Date: CSV cu date reale LTE din România (OpenCellID), incluzând coloanele latitudine, longitudine și raza de acoperire a fiecărei stații.
* Metrici de evaluare:
* Valoarea fitness-ului care combină distanța medie ponderată și penalizările.
* Numărul de stații active selectate.
* Vizualizarea grafică a stațiilor selectate și a punctelor de testare

**7.Resurse și Stadiul Actual al Cercetării**

1. Abordări clasice vs metaheuristici:  
   Problemele de amplasare a stațiilor LTE sunt combinatoriale și devin rapid intractabile pentru metode exacte, mai ales pe seturi mari de date reale. Metaheuristicile precum algoritmii genetici oferă un bun echilibru între calitatea soluției și timpul de calcul.
2. Avantaje ale Algoritmilor Genetici:  
   Pot gestiona flexibil diverse criterii în funcția de fitness și se adaptează bine la dimensiuni mari ale problemei. În schimb, necesită tuning al parametrilor și pot consuma resurse semnificative pentru convergență.
3. Utilizarea datelor reale:  
   Spre deosebire de simulări pe date sintetice, acest proiect folosește date reale LTE din România, sporind relevanța practică a rezultatelor.
4. Poziționarea proiectului:  
   Codul implementat combină date reale cu AG, aplicând o metodologie clară pentru a optimiza distribuția stațiilor pe baza acoperirii și a numărului minim de stații active**.**

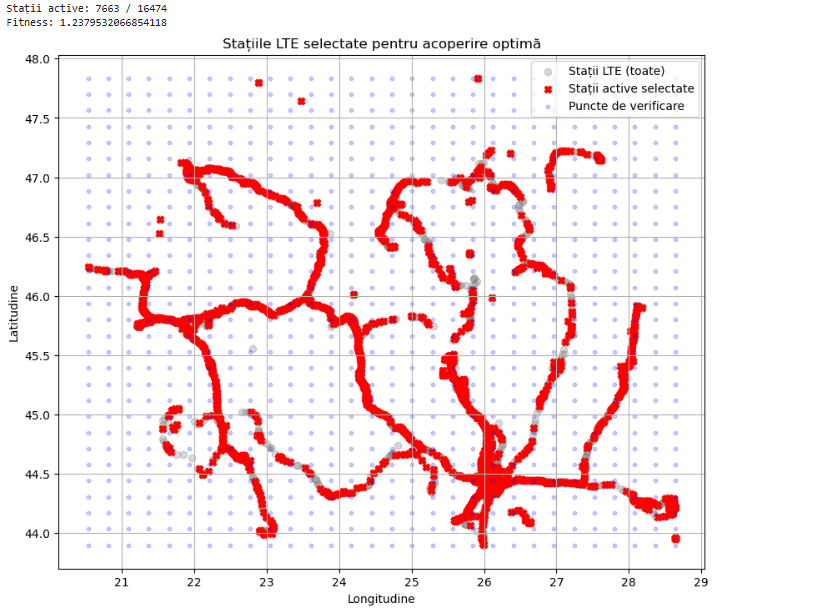
**8.Rezultate și Interpretare**

**Varianta 1.**

* Număr total de stații LTE: 16474
* Număr de stații active selectate: 7663 (~46.5% din total)
* Cel mai bun scor de fitness atins: 1.23795

Pe parcursul celor 50 de generații, algoritmul genetic a reușit:

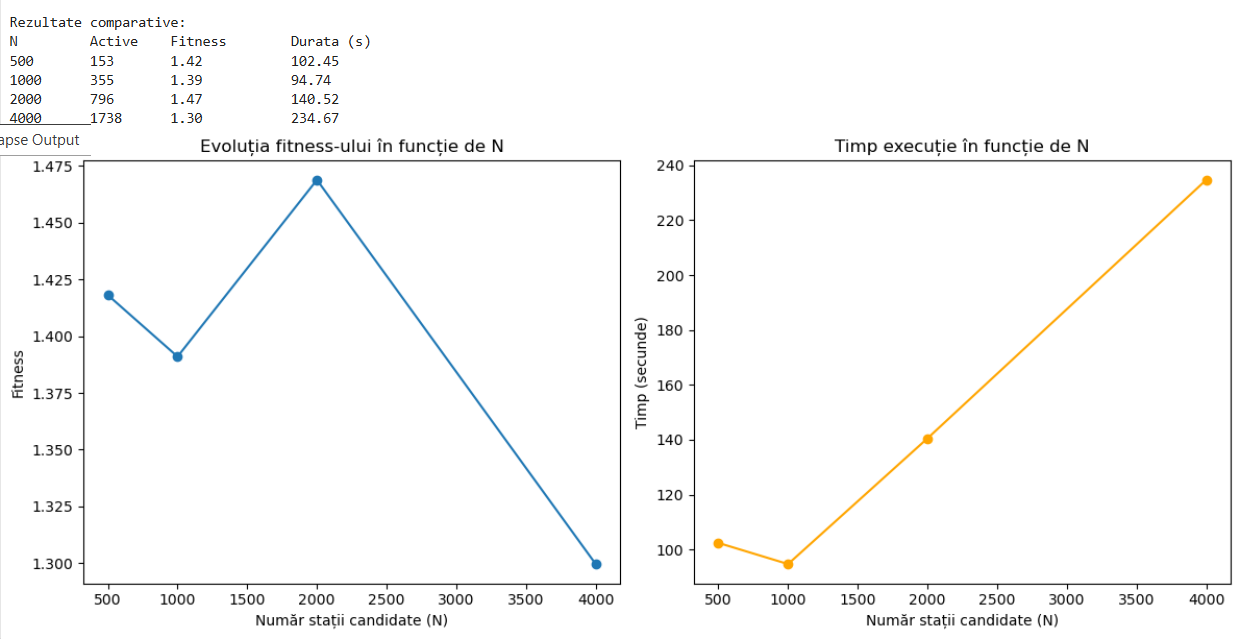
* să scadă scorul de fitness de la 1.313 la 1.238, o îmbunătățire vizibilă;
* să obțină o scădere constantă și stabilă a valorii minime, semn că algoritmul utilizat favorizat soluții din ce în ce mai bune;
* să reducă media populației în mod progresiv, ceea ce înseamnă că întreaga populație a evoluat pozitiv, nu doar indivizii de elită.



* Graficul generat arată clar pozițiile tuturor stațiilor LTE, cele activate (în roșu) și punctele de verificare din grilă.
* Evoluția fitness-ului în cadrul procesului de optimizare indică convergența către soluții mai bune.
* Compararea numărului de stații active înainte și după optimizare arată o reducere semnificativă a redundanței, menținând în același timp o bună acoperire a regiunii.
* Zonele cu acoperire slabă sau neacoperite pot fi identificate prin analiza punctelor de grilă neacoperite sau cu distanțe mari față de stațiile active.

**Varianta 2.**

Am testat algoritmul pentru un numar diferit de statii de baza. Am testat pentru 500,1000,2000,4000 pentru a vedea cum evolueaza fintness-ul. Cel mai bun fitness l-am obținut cu N = 4000 (1.30) — ceea ce sugerează că o diversitate mai mare a stațiilor oferă soluții mai bune.La valoarea lui N = 1000 oferă un fitness comparabil (1.39) dar cu mult mai puțin timp și mai puține stații active → posibil un compromis optim.



Fitness scade , in timp ce timpul de execuție crește.

**Varianta 3.**

Am testat varianta unui numar de statii N=1000 , generate aleator din cele 16474 stații de bază.

-dimensiune populatie 50

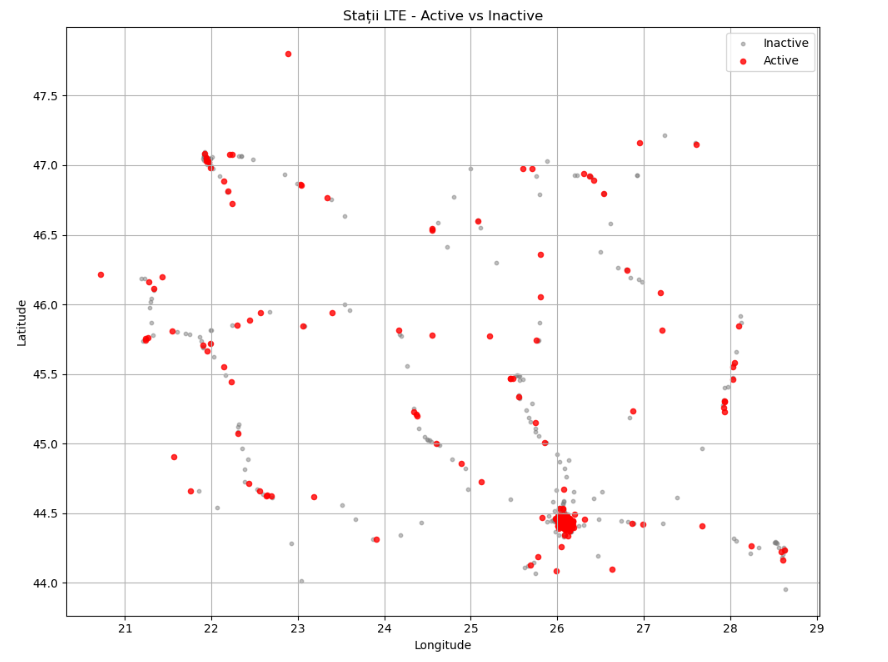
-numarul de generatii 50

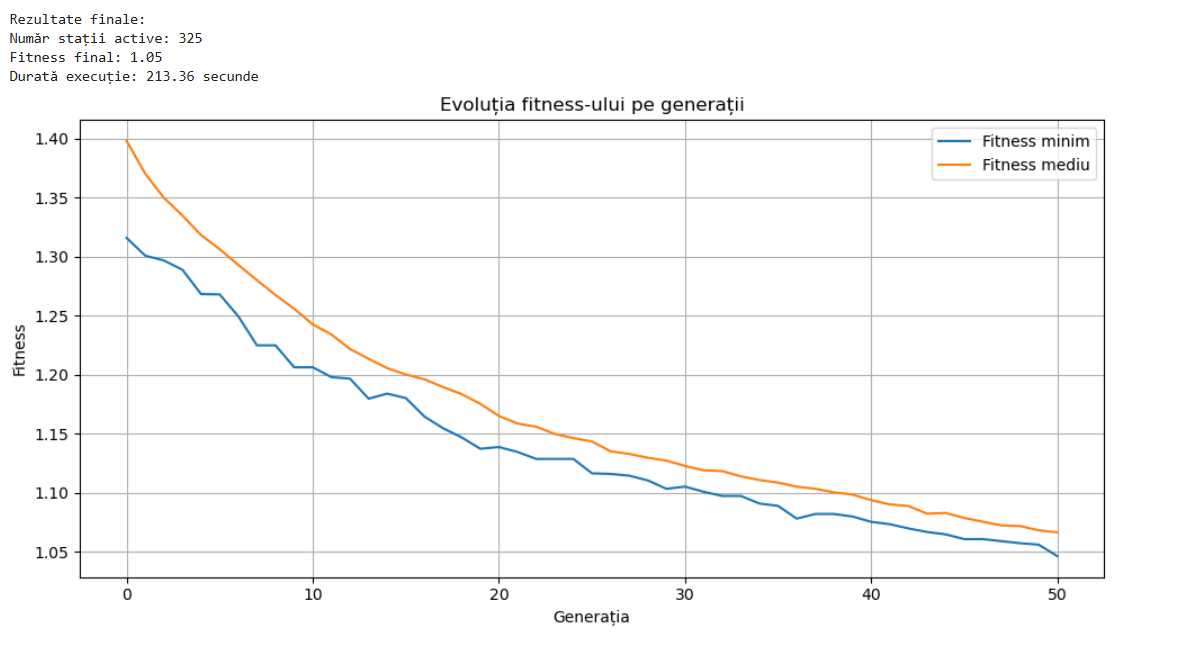
-probabilitatea de crossover (cxpb) = 0.5

-Probabilitatea de mutație (mutpb) = 0.2

-Tournsize în selecție (tournsize) =3

- Probabilitatea de mutație individuală (indpb) =0.05



Se observa o reducere a numarului de statii active de la 1000 la 325.

**Varianta 4 .**

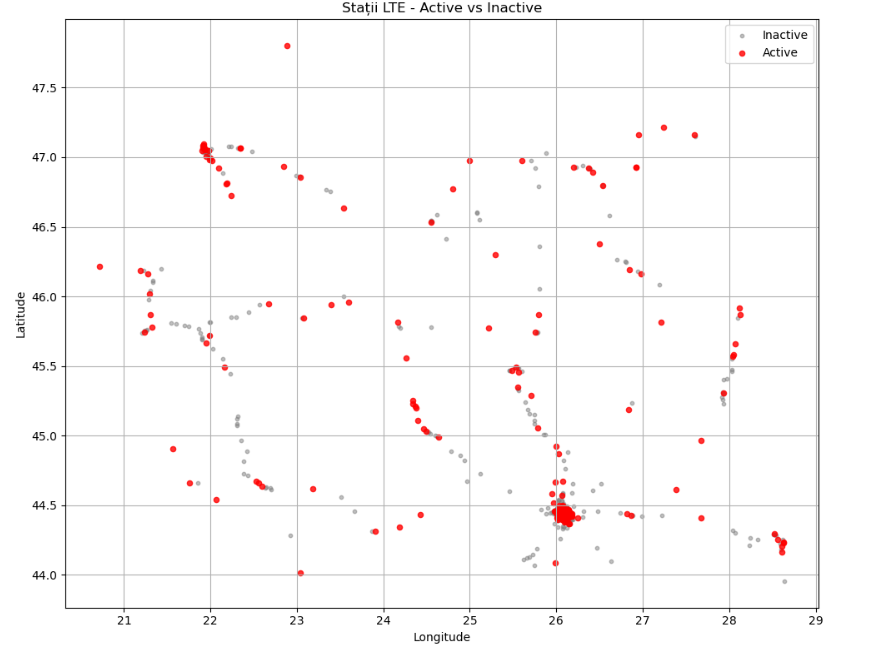
Am testat varianta unui numar de statii N=1000 , generate aleator din cele 16474 stații de bază.

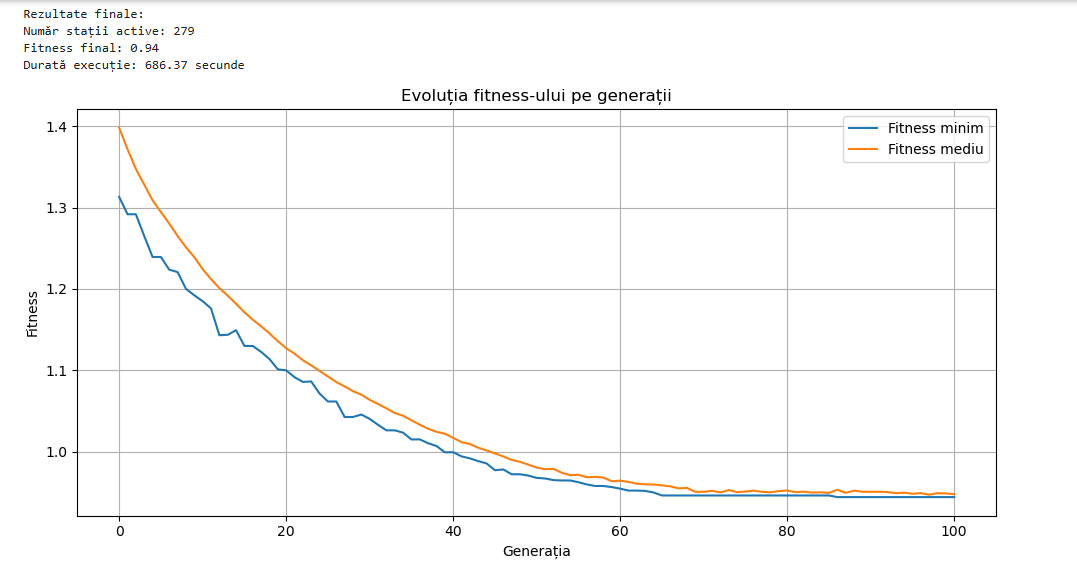
-dimensiune populatie 200

-numarul de generatii 100

-probabilitatea de crossover (cxpb) = 0.7

-Probabilitatea de mutație (mutpb) = 0.1





Observăm că după generatia 70 apare stagnare, ceea ce sugerează că populația a ajuns la un optim local. In acest caz se observa o reducere a numarului de statii active de la 1000 la 279. Un rezultat mai bun in urma ajustării parametrilor decat varianta 3.

**Varianta 5**

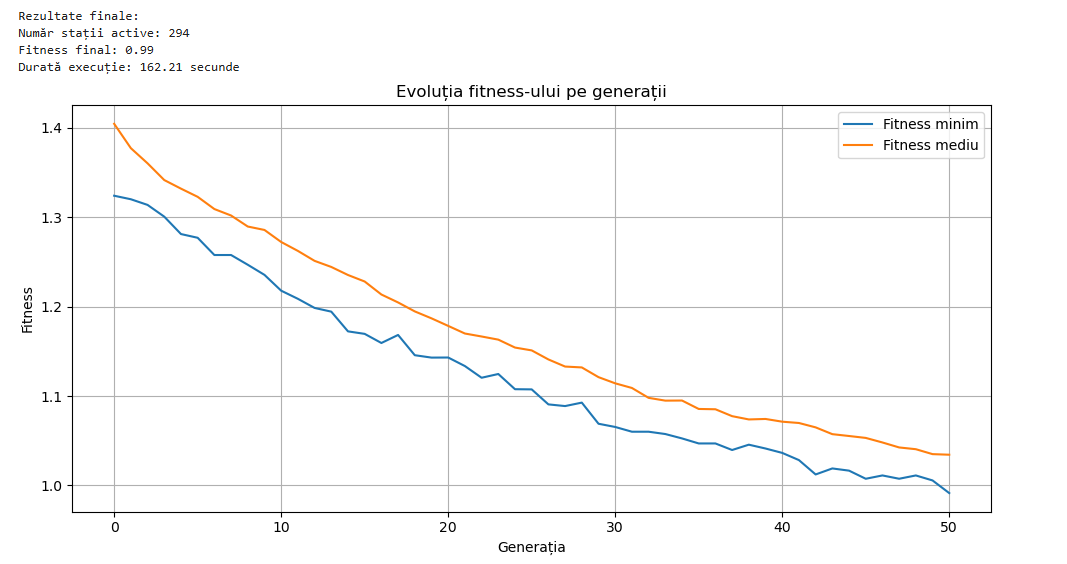
Am testat varianta unui numar de statii N=1000 , generate aleator din cele 16474 stații de bază.

-dimensiune populatie 100

-numarul de generatii 50

-probabilitatea de crossover (cxpb) = 0.9

-Probabilitatea de mutație (mutpb) = 0.4



**Varianta 6**

Am testat varianta unui numar de statii N=1000 , generate aleator din cele 16474 stații de bază.

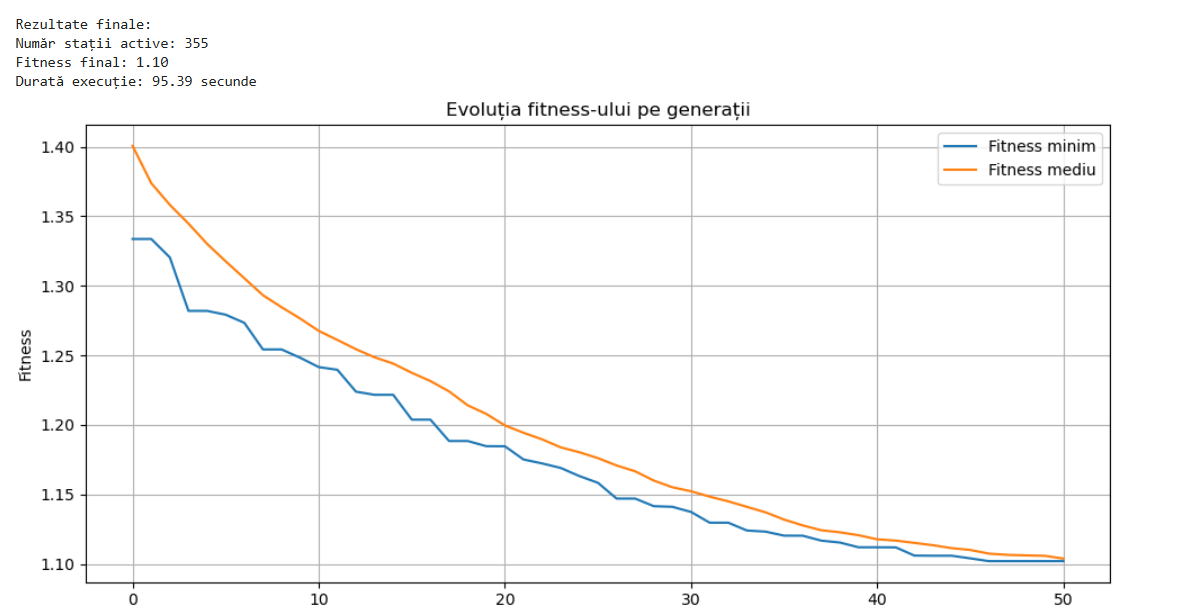
-dimensiune populatie 100

-numarul de generatii 50

-probabilitatea de crossover (cxpb) = 0.5

-Probabilitatea de mutație (mutpb) = 0.05

In acest caz se poate observa ca algoritmul nu s-a blocat intr-un minim local. Se observa o reducere a numarului de statii active de la 1000 la 355. Un rezultat mai slab in urma ajustării parametrilor decat varianta 3 ,4,sau 5 dar fitness-ul s-a blocat intrun minim local la 46 generație.



* Limitări:
  + Lipsa datelor exacte despre utilizatorii reali și trafic.
  + Lipsa metricilor QoS pentru o evaluare mai detaliată a calității rețelei.Daca sunt interferente intre celulele statiilor de baza. Nivelul de semnal.
  + Lipsa reliefului

**9.Concluzii și Contribuții Personale**

* Algoritmul Genetic implementat a demonstrat că este capabil să optimizeze distribuția stațiilor LTE pe baza datelor reale, reducând numărul de stații active fără a compromite acoperirea.
* Am prelucrat și adaptat date reale LTE pentru a fi utilizate în modelul de optimizare.
* Funcția de fitness a fost proiectată pentru a evalua în mod echilibrat acoperirea și numărul de stații active. Astfel că acoperirea maximă a unei statii de baza a fost un criteriu de bază în crearea functiei fitness.Deasemenea penalizări pentru cazul în care nicio stație de bază nu acoperea un anumit punct din grilă- inspirată din metrici QoS (Quality of Service)
* Funcția mea de fitness este inspirată din lucrări legate de plasarea senzorilor în rețele wireless, unde se optimizează atât acoperirea unei zone (prin distanță sau k-coverage), cât și numărul de resurse utilizate. (link spre functia fitness de unde m-am inspirat <https://github.com/Timothy1102/Genetic-Algorithm/blob/master/main.py>) , deasemenea o alta lucrare de unde m-am inpirat este <https://github.com/enessagroglu/NATURE-INSPIRED-ALGORITHM-OPTIMIZATION-FOR-BASE-STATION-LOCATION-ALLOCATION-PROBLEM/blob/main/Project/fitness_function.py>. In acest exemplu este o funcție care ia în calcul cererea acoperită în funcție de distanță față de punctele de interes (de exemplu, străzi sau clienți), cu penalizări pentru stații în exces și cerere neacoperită și care combinnă acoperirea cererii în funcție de distanță (similar cu punctele de pe grid din modelul meu) , limitarea capacității fiecărei stații și penalizări pentru stații în exces și cererea neacoperită. Eu am adaptat aceste idei pentru domeniul LTE, înlocuind cererea cu distanța până la stați și punctele de interes cu o grilă de test pentru acoperire geografică.
* Am testat algoritmul modificat parametri precum numărul total de statii active, dimensiune populatie ,numărul de generatii , probabilitatea de crossover (cxpb) ,probabilitatea de mutație (mutpb) pentru a observa daca algoritmul se blochează intr-un minim local și pentru a compara rezultatele obținute în urma modificărilor.
* Am învățat să combin metaheuristici cu date geografice reale și să interpretez rezultatele din punct de vedere al telecomunicațiilor
* Pe viitor, pot integra date despre utilizatori și trafic pentru optimizări mai precise, pot testa alte metaheuristici și pot integra vizualizări GIS mai detaliate.

**10.Implementare algoritm**

stations = df[["lat", "lon", "range"]].to\_numpy()

Așadar, **doar coordonatele geografice** (lat, lon) și **raza de acoperire** (range) a fiecărei stații sunt utilizate în optimizare.

Se creaza o matrice cu N randuri si 3 coloane.

***# Creez o grilă de puncte pentru testarea acoperirii***

***lat\_vals = np.linspace(stations[:,0].min(), stations[:,0].max(), 30)***

***lon\_vals = np.linspace(stations[:,1].min(), stations[:,1].max(), 30)***

***grid\_points = np.array([[lat, lon] for lat in lat\_vals for lon in lon\_vals])***

Creez o grila de dimensiune 30x30 care cuprinde zona unde sunt amplasate statiile de baza si unde voi testa coperirea stațiilor LTE. Iau 30 de valori care definesc latitudinea si 30 longitudinea , valori uniform distribuite.

Rolul acestei grila este de a vedea daca fiecare punct este acoperit de cel putin o statie, si cat de bine este acoperit un punct.

Funcție fitness() e inima algoritmului genetic – ea decide cât de bună este o soluție (un individ), adică o anumită alegere de stații LTE activate sau dezactivate.

***def fitness(individual):***

***individual:*** o listă de 0 și 1 cu lungime N (numărul total de stații LTE)

* 1 = stația este **activă** (selectată)
* 0 = stația este **inactivă** (ignorată

***active\_stations = stations[np.array(individual) == 1]***

-extrag doar rândurile din stations corespunzătoare pozițiilor unde individul are 1

***if len(active\_stations) == 0:***

***return (1e6,)*** # penalizez maxim

- Dacă un individ nu activează nicio stație → returnăm o valoare foarte mare ca fitness, adică „foarte slab”.

***for point in grid\_points:***

***dists = np.linalg.norm(active\_stations[:, :2] - point,* axis*=1)***

***in\_range = dists <= active\_stations[:, 2]***

-Verific daca un punct din grila este acoperit de o statie , comparand distanta de la acel punct la toate statiile active. In\_range este o variabila tip Boolean(TRUE).

***if np.any(in\_range):***

***total\_distance += np.min(dists[in\_range])***

-in cazul in care punctul e acoperit de cel putin o statie , iau distanta cea mai mica pana la statia respective.

***else:***

***total\_uncovered += 1***

***total\_distance += 1000***

-daca punctul nu este acoperit – il marchez ca neacoperit si penalizez puternic in scor..

***penalty\_size = np.sum(individual) / N***

* Aici calculez procentul de stații activate de către individul current. Avand in vedere ca individual este o listă de 0 și 1 , numar cati de 1 sunt in lista(adica statii active) si impart la numarul total de statii. Rezultatul e un procent între 0 și 1 care arată cât de costisitor este individul (pentru că mai multe stații = mai mult consum de energie / bani).

***penalty\_uncovered = total\_uncovered / len(grid\_points) # Procentul de puncte neacoperite***

* aici calculez procentul de puncte din grilă care NU suntacoperite de nicio stație. ***total\_uncovered*** = câte puncte din grilă nu au fost acoperite (nu se află în raza niciunei stații) , iar ***len(grid\_points)*** = numărul total de puncte din grilă (in cazul de fata 30x30). Rezultatul e un procent între 0 și 1 care arată cât de slab este individul în a acoperi zona.

Definec **funcția obiectiv** pe care algoritmul genetic o minimizează. Scopul este să găsesc un individ (adică o combinație de stații activate) cu scor cât mai mic.

***score = total\_distance / len(grid\_points) + 10 \* penalty\_uncovered + 2 \* penalty\_size***

* ***total\_distance / len(grid\_points)*** - masoara media distanței de la fiecare punct din grilă până la cea mai apropiată stație activă. Scopul este de a încuraja ca punctele din grilă să fie cât mai aproape de stații ,acest lucru inseamna semnal mai bun, latență mai mica.
* Cazul ideal ar fi ca toate punctele sa fie foarte aproape de o statie – valoare mica.
* ***10 \* penalty\_uncovered –***penalizez punctele care sunt in grila si nu sunt acoperite de statii , adica multiplic cu 10 pentru a da o penalizare mare daca multe puncta nu sunt acoperite.
* ***2 \* penalty\_size –*** penalizare pentru numarul de statii activate.

Scopul final este sa minimalizez scorul , tinand cont de calitatea acoperirii,acoperirea completa a grilei si de eficienta in alegerea statiilor.

Algoritmul genetic invata sa aleaga cat mai putine statii care sa acopere toata grila si sa fie cat mai aproape de punctele din grila.

Inițializare DEAP

***creator.create("FitnessMin", base.Fitness, weights=(-1.0,))***

* Clasa FitnessMin care definește o funcție de fitness pe care vreau s-o minimizez.
* weights=(-1.0,) -> scoruri mai mici = mai bune.

**creator.create("Individual", list, fitness=creator.FitnessMin)**

* Creează clasa Individual derivată din list.
* Fiecare individ este o **listă de 0 și 1** (stații oprite/pornite), și are asociată o valoare de fitness.

***toolbox.register("attr\_bool", random.randint, 0, 1)***

***toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr\_bool, n=N)***

***toolbox.register("population", tools.initRepeat, list, toolbox.individual)***

* ***attr\_bool:*** generează un bit aleator (0 sau 1).
* ***individual:*** creează un individ cu N gene (câte o stație).
* ***population:*** creează o listă de indivizi.

***toolbox.register("evaluate", fitness)***

* Înregistrează funcția de **evaluare** care calculează fitness-ul unui individ.
* Când algoritmul vrea să afle cât de bun este un individ (un set de stații activate), apelează această funcție fitness.
* fitness primește un individ (listă de 0 și 1) și returnează o tuplă cu valoarea fitness-ului (de minimizat).

***toolbox.register("mate", tools.cxTwoPoint)***

* Definește operatorul de **crossover** (împerechere / recombinare) folosit pentru a genera copii noi din doi părinți.
* tools.cxTwoPoint e un tip de crossover în două puncte:

Functionare:

* Se aleg aleator 2 poziții în cromozom (individ).
* Segmentele dintre acele două poziții sunt schimbate între cei doi părinți.

***toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)***

* Definește operatorul de **mutație**, care adaugă diversitate prin modificări mici ale indivizilor.
* tools.mutFlipBit este o mutație specifică pentru indivizi codificați binar (listă de 0 și 1):
* Pentru fiecare bit în individ, există o probabilitate indpb să fie inversat (0 devine 1, 1 devine 0). indpb=0.05 înseamnă:
* 5% șansă pentru fiecare bit să fie "flipped" (inversat).

Folosesc mutatie pentru pentru a evita blocarea în minime locale si pentru a introduce noi combinații care nu apar prin crossover.

***toolbox.register("select", tools.selTournament, tournsize=3)***

* Definește operatorul de selecție, adică modul în care alegem indivizii pentru a produce următoarea generație.
* ***tools.selTournament*** este o metodă de selecție prin turneu.

Selectia prin turneu selecteaza un individ astfel incat : se alege aleator ***tournsize*** indivizi din populație (aici 3).Din acesti 3 se alege 1 cu cel mai bun fitness. Este un echilibru intre exploatare si explorare.

Evaluate -> Calculează fitness-ul indivizilor -> fitness -> Măsoară calitatea soluție

mate-> Recombinează doi părinți pentru copii-> cxTwoPoint -> Combină gene, creează variație

mutate -> Inversează biți aleatori din cromozom -> mutFlipBit, indpb=0.05 -> Introduce variație nouă și evită stagnarea

select-> Alege indivizii pentru reproducere -> selTournament, tournsize=3 -> Favorizează indivizii buni cu șansă mică pentru ceilalți

***pop = toolbox.population(n=50) # populație de 50 indivizi***

***hof = tools.HallOfFame(1) # păstrează cel mai bun individ***

***stats = tools.Statistics(lambda ind: ind.fitness.values)***

***stats.register("min", np.min)***

***stats.register("avg", np.mean)***

* creez o populatie initiala de 50 de indivizi(fiecare individ este o lista binara 0 si 1).Pastrez cel mai bun individ găsit pe parcursul evoluției si il retine(1) asa ca pot accesa solutia optima gasita. Cree un obiect pentru a colecta și calcula statistici pe fitness-ul indivizilor.Aici lambda primește un individ și extrage valoarea fitness (tuple) pentru a face statistici pe ea. ***Min*** care calculează valoarea minimă din fitness-urile indivizilor din populație. **avg** - calculează media fitness-urilor populației.

pop = populația curentă de 50 soluții

hof = cea mai bună soluție de până acum

stats = obiect care măsoară min și media fitness-urilor pentru fiecare generație

***pop, log = algorithms.eaSimple(pop, toolbox,***

***cxpb=0.5, mutpb=0.2,***

***ngen=50,***

***stats=stats,***

***halloffame=hof,***

***verbose=True)***

-functie de din DEAP care rulează un algoritm genetic simplu.

**Parametrii funcției:**

* **cxpb=0.5**  
  Probabilitatea ca doi indivizi să se încrucișeze (crossover) în fiecare generație este 50%.  
  Deci, jumătate din noile indivizi rezultați vor fi copii recombinați.
* **mutpb=0.2**  
  Probabilitatea ca un individ să sufere mutație în fiecare generație este 20%.  
  Mutatia modifică câte un bit (0 sau 1) cu probabilitatea indpb=0.05 (definită în toolbox).
* **ngen=50**  
  Numărul de generații (iterații) pe care le rulează algoritmul.  
  Practic, evoluția populației se face timp de 50 de cicluri.
* **stats=stats**  
  Obiectul care colectează statistici (minim, medie fitness) la fiecare generație.
* **halloffame=hof**  
  Obiectul care ține minte cel mai bun individ pe tot parcursul evoluției.
* **verbose=True**  
  Afișează informații despre progresul algoritmului (fitness minim și mediu pentru fiecare generație).

Pentru fiecare generatie(50) :

* Selectează indivizi pentru reproducere (folosind selecția în turneu).
* Aplică crossover cu probabilitatea cxpb la indivizi selectați.
* Aplică mutații cu probabilitatea mutpb.
* Evaluează fitness-ul indivizilor noi (descendenții).
* Actualizează populația cu noii indivizi.
* Actualizează HallOfFame dacă găsește un individ mai bun.
* Colectează statistici.

La final, returnează populația finală (pop) și un log (log) cu istoricul statisticilor pe generații.

***best = hof[0] # cel mai bun individ***

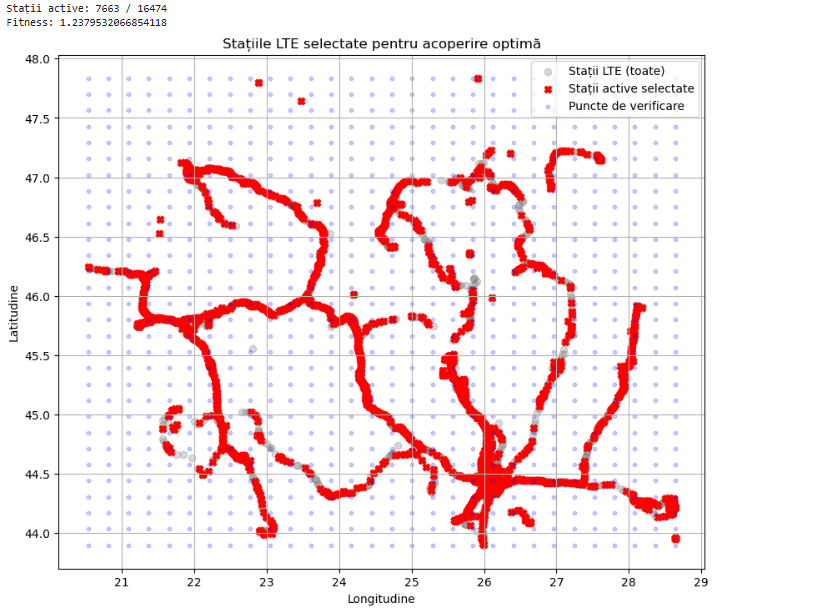
***active = np.where(np.array(best) == 1)[0]***

***print(f"Stații active: {len(active)} / {N}")***

***print(f"Fitness: {best.fitness.values[0]}")***

* Se extrag pozițiile stațiilor activate.
* Se afișez numărul și scorul final.

Rezultat



**10. Bibliografie**

[1]. Ion Bogdan , Comunicatii Mobile

[2]. <https://revistaverso.ro/descoperirea-solutiilor-optime-puterea-algoritmilor-genetici-in-rezolvarea-problemelor-complexe/?utm_source=chatgpt.com>

[3] Proposed Base-Station Location Optimization with Genetic Algorithm Scheme for Lte Network Radio Planning Shahad Nafea Jaafar, Ekhlas Kadum Hamza and Viean Abdulmuhsin Al-Salihi Control & System Engineering Department, University of Technology, Iraq

[4] Genetic Algorithms in Wireless Networking: Techniques, Applications, and Issues Usama Mehboob, Junaid Qadir, Salman Ali, Athanasios Vasilakos

[5] Optimal Location of Base Stations for Cellular Mobile Network , Rahmani, A., & Yeganeh, M. H.,(2015