Rezolvarea problemei acoperirii cu vârfuri (VCP) folosind metaeuristici

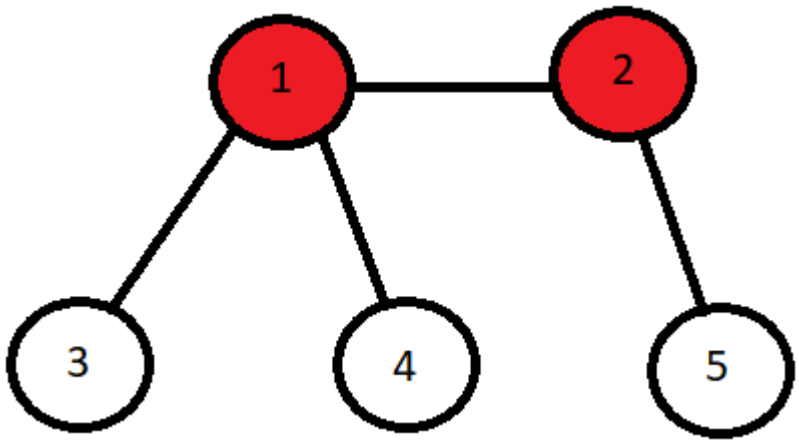
David Andreea Raluca - Grupa 10LF221

Mai 2024

1 Prezentarea Problemei Vertex Cover

Problema vertex cover este una fundamentală în teoria grafurilor. Ea se concentrează pe găsirea unui set de vârfuri într-un graf astfel încât fiecare muchie din graf să aibă cel puțin unul dintre capetele sale în acest set. Mai simplu spus, este vorba despre a alege un număr minim de vârfuri astfel încât să fie garantată acoperirea tuturor muchiilor din graf.

Pentru a înțelege mai bine această problemă, haideți să analizăm un exemplu simplu. Să luăm în considerare un graf cu câteva vârfuri și muchii:



Exemplu de set de noduri minim a unui graf folosind Vertex Cover.

În acest graf, putem observa că pentru a acoperi toate muchiile, putem alege vârfurile 1 și 2. Astfel, fiecare muchie are cel puțin un capăt în acest set de vârfuri. Însă, acesta nu este singurul set de vârfuri care ar acoperi toate muchiile. De exemplu, putem alege și vârfurile 3, 4 și 5, dar acesta nu este setul de noduri minim.

Scopul în problema vertex cover este să găsim un astfel de set de vârfuri care să fie cât mai mic posibil, deoarece, de cele mai multe ori, este preferabil să avem un număr cât mai mic de vârfuri în acest set pentru a economisi resurse și a optimiza anumite procese.
Deși problema este simplu de enunțat, ea este cunoscută ca fiind NP-completă, ceea ce înseamnă că nu există o soluție eficientă (polinomială) pentru a găsi întotdeauna soluția optimă într-un timp rezonabil, mai ales în grafuri mari. Acest lucru face ca problema vertex cover să fie un subiect de cercetare activ în domeniul informaticii teoretice.

2 Exemple de aplicații practice ale problemei vertex cover în diverse domenii

- **Biologie și bioinformatică:**
  În bioinformatică, grafurile sunt utilizate pentru a modela relațiile între molecule, proteine sau gene. Problema vertex cover poate fi folosită pentru a identifica un set de molecule sau proteine care interacționează strâns, ceea ce poate oferi informații despre funcționarea sistemelor biologice.

- **Optimizare în logistică:**
  În domeniul logistică, problema vertex cover poate fi aplicată pentru a găsi un set minim de locații de depozitare sau de distribuție care să acopere toate zonele în care trebuie să fie livrate produsele. Alegerea strategică a acestor locații poate duce la economii semnificative de timp și costuri în lanțurile de aprovizionare.

- **Managementul resurselor financiare:**
  În domeniul financiar, problema vertex cover poate fi aplicată pentru a identifica portofoliile optime de investiții. Alegerea unui set diversificat de active financiare care să acopere riscurile și oportunitățile pieței poate fi esențială pentru gestionarea eficientă a portofoliului.

3 Algoritm genetic

În continuare, vom prezenta abordarea noastră de rezolvare a problemei vertex cover folosind un algoritm genetic. Vom detalia modul în care am adaptat și implementat algoritmul genetic pentru a găsi o soluție optimă sau satisfăcătoare pentru problema vertex cover într-un timp eficient. Prin explorarea metodelor noastre de selecție, încrucișare și mutație, precum și prin ajustarea parametrilor algoritmului genetic, vom demonstra modul în care această metodă poate fi utilizată cu succes pentru rezolvarea problemelor de acoperire cu vârfuri în grafuri.

3.1 Codificare individ

Am codificat indivizii ca șiruri de biți, unde fiecare bit reprezintă un vârf din graf. Dacă bitul este setat la 1, înseamnă că vârful respectiv este inclus în soluție (face parte din vertex cover), iar dacă este setat la 0, înseamnă că nu este inclus în soluție.

$$f(x) = \begin{cases} 1 & \text{dacă vârful } i \text{ este inclus în soluție} \\ 0 & \text{dacă vârful } i \text{ nu este inclus în soluție} \end{cases}$$

3.2 Funcția de fitness

Am folosit o funcție de fitness simplă care evaluează o soluție în funcție de numărul de muchii neacoperite de către acea soluție.
Mai precis, pentru fiecare vârf din graf care nu este inclus în soluție, numărăm câte dintre vecinii săi (adică vârfurile la care este conectat prin muchii) nu sunt incluși în soluție. Suma acestor valori reprezintă numărul total de muchii neacoperite de soluție.

$$\text{fitness(soluție)} = \text{numărul total de muchii neacoperite de soluție}$$

Scopul nostru este să minimizăm această valoare, deoarece vrem să găsim un vertex cover care să acopere cât mai multe muchii posibil. Astfel, funcția de fitness este esențială în determinarea cât de bine se potrivește o soluție la problema vertex cover.

3.3 Selecția

Folosim o formă simplă de selecție numită "turnir de selecție". Această metodă de selecție implică alegerea aleatorie a unui sub-set de indivizi din populație și selectarea celui mai bun individ (cel cu cea mai bună valoare de fitness) din acest sub-set. Cum funcționează turnirul de selecție:

1. Se alege un număr fix de participanți (numit "turneu") din populație.
2. Se evaluează fiecare participant în funcție de funcția de fitness.
3. Cel mai bun participant (adică cel cu cea mai bună valoare de fitness) este selectat pentru a fi părinte.

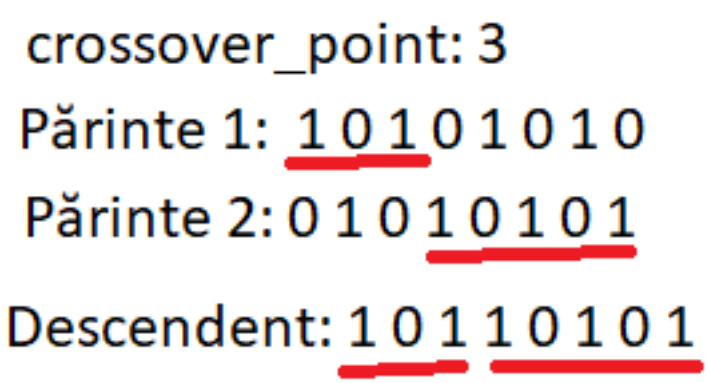
Acest proces este repetat de două ori pentru a selecta doi părinți pentru reproducere.
În esență, turnirul de selecție oferă o modalitate simplă și eficientă de a selecta părinți în algoritmii genetici, fără a necesita sortarea întregii populații.

3.4 Cross-Over

Crossover-ul, sau încrucișarea, este unul dintre operatorii principali în algoritmul genetic, și este folosit pentru a combina informația genetică a doi părinți pentru

a genera un descendent (sau mai mulți) care prezintă caracteristici ale ambilor părinți.
Cum funcționează crossover-ul nostru:

1. Se alege un punct random pentru împărțirea părinților.
2. Se ia prima parte până la punctul respectiv dintr-un părinte și se ia a doua parte de la punctul din celălalt părinte.
3. Cele două părți se unesc, rezultând un descendent.



Exemplu de crossover între 2 părinți.

3.5 Mutația

Mutația reprezintă unul dintre operatorii principali și este o operație care modifică aleatoriu o mică parte din materialul genetic (sau soluția) a unui individ din populație. Acest lucru ajută la introducerea diversității în populație și poate contribui la explorarea unor noi regiuni ale spațiului de căutare.
Cum funcționează mutația noastră:

1. Pentru fiecare bit din individ se generează o valoare random.
2. Dacă valoarea este mai mică decât rata de mutație, atunci bitul respectiv va fi înlocuit cu valorea sa complementară (1 devine 0 sau 0 devine 1).

Mutația este aplicată fiecărui individ din populație cu o anumită rată de mutație. Aceasta înseamnă că fiecare bit din soluție are o șansă dată de rată de mutație de a fi inversat (de la 0 la 1 sau de la 1 la 0), contribuind astfel la diversitatea populației și explorarea spațiului de căutare.

3.6 Parametrii predefiniți ai algoritmul

Algoritmul nostru are următorii parametrii predefiniți:

- **population_size:** mărimea populației
- **tournament_size:** numărul de indivizi luați în turnir
- **crossover_rate:** rate de incrucișare
- **mutation_rate:** rate de mutație
- **max_generations:** numărul maxim de generații pe care îl facem

3.7 Pași algoritm

Algoritmul nostru arată astfel:

Algorithm 1 Algoritm Genetic - Vertex Cover

```
1: Generare populație curentă.
2: for generation în range(max_generations) do
3:    Aflăm cele mai bune soluții din populație folosind funcția de fitness.
4:    if Dacă găsim deja o soluție optimă then
5:       Returnăm cea mai bună soluție de o avem.
6:    end if
7:    Soluțiile găsite devin noua populație.
8:    while mărimea noii populați <population_size do
9:       Folosim Selecția pentru a avea 2 părinți.
10:      Folosim random Cross-over pe descendent.
11:      Folosim Mutația pe descendent.
12:      Adăugăm descendentul în noua populație.
13:   end while
14:   Luăm populația nouă ca cea curentă.
15: end for
16: Returnăm cea mai bună soluție.
```

3.8 Setul de date

Setul de date este generat random la fiecare rulare având un număr de vârfuri și o probabilitate de muchii predefinite, astfel generându-se un graf nou random.

References

[1] *Genetic Algorithms*. GeeksforGeeks. Disponibil la: https://www.geeksforgeeks.org/genetic-algorithms/. (accesat la data de 12.05.2024)

[2] Lucian Mircea Sasu. *Calcul evoluționist - Algoritmi Genetici*. Curs - Inteligență Artificială.

[3] Luciana Majercsik. *Rezolvarea problemei acoperirii cu vârfuri (VCP) folosind metaeuristici*. Laborator - Automate, Calculabilitate, Complexitate.