Un algoritm genetic pentru problema comis-voiajorului

Chiș Bogdan

Mai 2023

1 Introducere

Enunțul problemei cere determinarea celui mai scurt traseu care pornește dintr-un punct, vizitează exact n puncte și se întoarce în punctul de pornire. Acest enunț ne duce cu gândul la problema comis-voiajorului care trebuie să viziteze un număr de localități într-o anumită ordine astfel încât distanța parcursă să fie minimă.

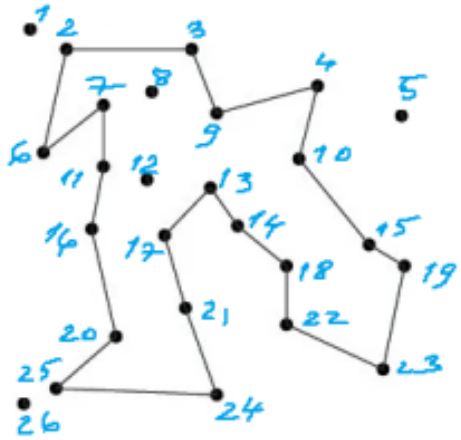

Figure 1: Soluția unei probleme cu 26 de puncte și acoperire 82%

Problema voiajorului are în general o complexitate factorială. Cunoscând faptul că avem x localități, atunci numărul de trasee posibile este $x!$. În cazul enunțului dat în cadrul concursului, voiajorul trebuie să viziteze doar n localități, unde $n = x \cdot p/100$, p fiind un procentaj de acoperire a grafului. Complexitatea este dată de suma permutărilor pentru fiecare submulțime distincă neordonată cu n elemente: $n! \cdot C(x, n)$. Dacă se înlocuiește notația combinărilor cu formula specifică acestora se obervă o simplificare a complexității față de enunțul general al problemei voiajorului.

$$n! \cdot \frac{x!}{n!(x-n)!} = \frac{x!}{(x-n)!} = A(x, n)$$

Astfel, ajungem la formula aranjamentelor. Deci pentru a găsi soluția optimă este necesară verificarea tuturor submulțimilor distincte și ordonate cu n elemente.

2 Starea curentă a problemei

Problema voiajorului este dificil de rezolvat în cazul seturilor de date în care graful prezintă multe noduri. De-a lungul timpului au apărut multiple soluții

deterministice care garantează rezultatul optim. Tehnici precum Backtracking, Branch and Bound, programare dinamică sau programare liniară[1] au fost încercate și au dat rezultate pe seturi de date cu puține noduri. În cazul unui graf complex, aceste soluții sunt lente. Deși este garantată soluția optimă, timpul de calcul pentru obținerea rezultatului este prea îndelungat folosind aceste abordări.

O soluție nedeterministă pentru problemele de optimizare este algoritmul genetic. Acest tip de algoritm face parte din ramura algoritmilor evolutivi ai inteligenței artificiale. Spre deosebire de tehnicile menționate anterior, algoritmii genetici nu garantează soluția optimă, însă pot oferi un rezultat într-un timp scurt. Dezvoltarea lor implică o testare riguroasă pentru a determina implementarea și parametrii care oferă un rezultat cât mai apropiat de cel optim. Această abordare este echilibrată, întrucât timpul de calcul este moderat, algortimul genetic fiind mai lent decât o abordare Greedy și mai rapid decât un Backtracking, iar rezultatul final poate fi apropriat de optim și mai bun decât cel al unui algoritm Greedy care alege optimul local.

3 Modelare teoretică

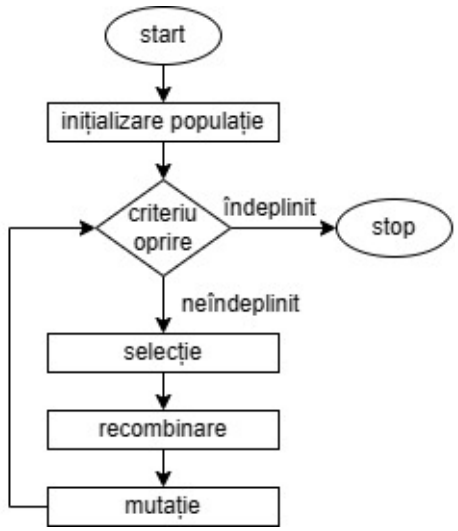

Figure 2: Schema logică a unui algoritm genetic

Etapele unui algoritm genetic sunt puține. Primul pas este inițializarea unui populații de indivizi generați aleator. Individul este o soluție potențială. În acest caz soluția este un traseu. Cât timp nu este îndeplinit un criteriu de oprire, se repetă selecția, recombinarea și mutația. O iterație a buclei reprezintă o generație. Un criteriu de oprire care poate fi folosit este atingerea unui număr maxim de generații.

Selecția este procesul prin care indivizii din populație sunt aleși pentru a participa la recombinare. Problema dată face parte din categoria problemelor de minimizare. Fitness-ul unui individ este un indicator de comparare a soluțiilor. Calculând distanța totală a două soluții potențiale putem spune care traseu este mai scurt. Există multiple moduri de a implementa selecția, însă este important să ținem cont de fitness-ul indivizilor pentru a asigura evoluția generațiilor. Indivizii cu fitness mai mare au șanse mai mari de a participa la recombinare.

Mai departe, în etapa de recombinare pot fi folosite diferite tehnici de a împerechea genele indivizilor pentru a obține soluții noi. Genele pot fi în acest caz noduri ale grafului sau arce. Este important de înțeles că algoritmul genetic este un concept care poate fi implementat în moduri multiple. Pe baza a două trasee, poate fi generat un traseu nou prin amestecarea nodurilor vizitate. Trebuie ținut cont de faptul că soluția nouă trebuie să moștenească de la soluțiile părinte secvențe din traseele anterioare. Altfel, riscăm o involuție a generației următoare.

Mutația este ultimul proces pe care îl suferă indivizii. Mutația are un rol important în evoluția indivizilor, întrucât asigură apariția unor gene noi în cadrul populației. Moștenind în mod repetat un set de gene inițiale, după un număr mare de generații indivizii o să semene între ei. Diversitatea insuficientă în cadrul populației va determina algoritmul să se blocheze într-un optim local. Astfel, mutația poate încetini convergența spre optimul local.

4 Implementare

În capitolul anterior a fost descris conceptul general al algoritmului genetic, urmând ca în acest capitol să fie prezentate implementările pentru fiecare operator genetic. Aceste implementări depind în primul rând de modul în care este reprezentat individul. Precum apare în alte publicații [2], individul poate fi reprezentat ca un șir de numere. Un exemplu de individ: 3 1 5 2 4 7 6. Această metodă de codificare poate fi interpretată în următorul fel: punctul de plecare este nodul cu numărul de ordine 3, următorul nod vizitat este 1, din 1 voiajorul pleacă spre 5, apoi spre 2, 4, 7 și la final ajunge în punctul cu numărul de ordine 6 de unde se întoarce în punctul 3.

Selecția utilizată este una de tip turneu. Se aleg aleator două soluții din populație. Pentru fiecare se calculează și se compară distanța totală a traseelor. Soluția cu distanța totală mai mică este selectată pentru recombinare. Distanța totală fiind mai mică, individul are evident un fitness mai mare, deci există în traseul respectiv porțiuni care sunt mai scurte decât în soluția concurentă. Acest proces se repetă până când avem suficiente perechi de indivizi care în urma recombinării vor rezulta o populație de aceeași dimensiune cu cea anterioară.

Recombinarea a doi indivizi părinte se face prin alegerea unui punct de tăietură aleator pe ambii indivizi. Odată ales punctul de tăietură, se copiază prima secvență de gene de la primul părinte în individul copil după care se completează soluția cu genele din al doilea părinte luate de la stânga spre dreapta dublurile fiind excluse. Un exemplu de recombinare:

parent 1	:	**1**	**2**	**5**	**6**	**4**	**3**	**8**	**7**
parent 2	:	1	4	2	3	6	5	7	8
offspring	:	**1**	**2**	4	3	6	5	7	8

Figure 3: Exemplu de recombinare[2]

Mutația se întâmplă în funcție de o rată de mutație stabilită. Se iterează toți indivizi, iar la fiecare individ îi sunt iterate genele. Pentru fiecare genă se generează un număr întreg aleator în intervalul $[0, 99]$. Dacă numărul este mai mic decât rata de mutație, atunci gena respectivă va suferi modificări. Mutația propusă presupune interschimbarea a două noduri aleatoare din traseu. Spre exemplu:

înainte de mutație: 3 1 5 2 4 7 6
după mutație: 3 1 **7** 2 4 **5** 6

Pentru a evita optimul local, în cadrul mutației se folosesc două rate: o rată de mutație mică, folosită la pornirea algoritmului, și o rată de mutație mare, folosită atunci când trece un anumit număr de generații fără să apară un individ mai bun. După găsirea unei soluții mai eficiente se revine la rata de mutație normală.

Criteriul de oprire folosit este timpul de rulare al algoritmului. În enunțul problemei date la concurs, este specificat faptul că există o limită de timp maximă admisă pentru rularea algoritmului. Astfel, algoritmul nu va fi oprit prea devreme când este atins un număr concret de generații. Totodată, această abordare facilitează compararea algoritmului propus cu alți algoritmi creați de ceilalți participanți.

5 Parametrizare

Algoritmul prezintă câțiva parametri ce trebuie ajustați: timpul maxim de rulare, dimensiunea populației, rata de mutație mică, rata de mutație mare și o valoare delta pentru a determina când când trebuie forțată o rată de mutație mai mare. Calibrarea algoritmului a fost făcută încercând valori diferite pentru fiecare parametru în parte. Este necesar un echilibru între dimensiunea populației și timpul de rulare deoarece o populație numeroasă împiedică generarea de soluții noi într-un timp scurt. Conform unui articol [3], în cazul mutației implementate ca interschimb de localități, ratele de mutație mici dau rezultate bune. O rată de mutație mică se încadrează sub valoarea de 5%. Cele mai bune rezultate au fost obținute când rata de mutație inițială a fost 0%, iar rata de mutație mare de 3%.

6 Testare

În cadrul testării algoritmul genetic(AG) a fost rulat de cel puțin 3 ori cu un timp maxim de rulare de 10 secunde. Datorită unor lucrării științifice[4][5] am găsit distanța totală minimă pentru câteva dintre seturile de date de testare.

set de date	nr. puncte	optim	AG
Eil51	51	426	526
Eil76	76	538	851
KroA100	100	21282	56355
KroA200	200	29368	161767

Table 1: Comparație între rezultatele optime și rezultatele obținute

Se observă faptul că diferența între soluția obținută de algorimul genetic și soluția optimă este mai mică în cazul grafurilor cu un număr restrâns de noduri, față de diferența rezultatelor în cazul seturilor de date complexe. Putem menționa faptul că există o corelație între numărul de noduri și valoarea erorii.

7 Concluzii

În concluzie, algoritmul genetic este o abordare potrivită în cazul problemelor de optimizare timpul de calcul fiind redus. Deși soluția propusă în 72 de ore în cadrul concursului nu reușește să se apropie de optim decât pe seturi de date simple, eroarea crescând odată cu complexitatea grafului, acest algoritm poate răspunde într-un timp rezonabil la cerința problemei. Rezultatul final este mai eficient decât alegerea optimului local calculând distanța de la un punct la celelalte puncte nevizitate.

Avantajul acestui tip de algoritm este faptul că nu trebuie să știm rezolva problema în sine, soluția fiind găsită prin amestecarea soluțiilor anterioare. Caracterul nedeterminist implică dificultăți în testare, întrucât trebuie rulate mai multe teste pentru același set de date, rezultatul fiind diferit la fiecare rulare. Rezultatele obținute sunt în strânsă legătură cu parametrizarea acestuia, iar valorile potrivite parametrilor se determiă experimental. Conceptul de algortim genetic este abstract, oferind celui care îl implementează libertatea de a alege dintr-o multitudine de reprezentări posibile ale individului în funcție de problemă, tipuri diferite de selecție, recombinare sau mutație.

Algoritmii genetici sunt doar una dintre căutările nedeterministe existente. Pentru a dezvolta un algoritm care să obțină rezultate cât mai apropiate de optim într-un timp scurt este importantă documentarea și experimentarea.

References

[1] Wikipedia contributors. Travelling salesman problem — Wikipedia, the free encyclopedia, 2023. [Online; accessed 27-May-2023].

[2] Jean-Yves Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63(1):339–370, 1996.

[3] Andrew Dinhobl. Tuning a traveling salesman, November 2018. Towards Data Science.

[4] Blerina Zanaj and Elma Zanaj. Review of traveling salesman problem for the genetic algorithms. *Journal of Information Sciences and Computing Technologies (JISCT). Albania*, 2016.

[5] Mohammad Shokouhifar, Ali Jalali, and Hamid Torfehnejad. Optimal routing in traveling salesman problem using artificial bee colony and simulated annealing. In *1st National Road ITS Congress*, 2015.

[6] Oliviu Matei. *Evolutionary computation: principles and practices*. Risoprint, 2008.

[7] TechTarget Contributor. Traveling salesman problem. 2020.