Un MODEL genetic pentru problema ORARULUI

Radu Vișinescu

Acest articol este structurat în două părți; în prima dintre ele este prezentată noțiunea de algoritm genetic, iar în cea de-a doua se propune un model de algoritm genetic în vederea rezolvării unei cunoscute probleme de programare, cea a generării orarului de funcționare a unei instituții de învățământ.

Algoritmi genetici

Introducere

Algoritmii genetici reprezintă un caz particular al algoritmilor evolutivi. În conceperea algoritmilor evolutivi s-au folosit operații specifice existente în natura la nivelul secvențelor de ADN ce formează cromozomii organismelor vii. Legat de aceste operații există și unele experimente care iau în considerare un nou tip de calculator pe bază de *ADN* [3].

Algoritmii genetici au apărut din dorința de a modela pe calculator diverse situații evolutive și pentru testarea în cadrul acestora a diferitelor ipoteze. De asemenea, o puternică motivație a fost rezolvarea pe această cale a problemelor NP (nedeterminist polinomiale) [2].

Prin modelarea situației din natură putem vorbi despre ***cromozom*** ca fiind un vector de valori de lungime finită. Un element al vectorului cromozom se va numi ***genă***, iar o genă poate lua la rândul său o mulțime finită de valori. În multe cazuri, similar unui bit, gena poate lua valori din mulțimea {0, 1}.

Operația de recombinare a doi cromozomi de aceeași lungime poartă în literatură denumirea de ***crossover***. Are ca rezultat alți doi cromozomi obținuți din secvențe analoage ale cromozomilor inițiali (părinți).

Prezentăm un caz particular al acestei operații în care cromozomii se "taie în două locuri" ($k = 2$).

Fie $k = 2$ și $1 \le t_1 < t_2 \le n$. Din cromozomii

$c_i = x_1x_2\ldots x_{t1}x_{t1+1}\ldots x_{t2}x_{t2+1}\ldots x_N$

și

$c_j = y_1y_2\ldots y_{t1}y_{t1+1}\ldots y_{t2}y_{t2+1}\ldots y_N$

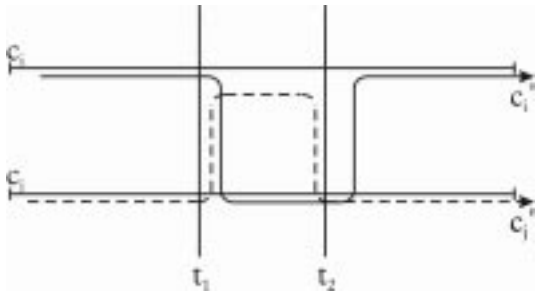
se vor obține cromozomii:

$c_i' = x_1x_2\ldots x_{t1}y_{t1+1}\ldots y_{t2}y_{t2+1}\ldots y_N$

și

$c_j' = y_1y_2\ldots y_{t1}x_{t1+1}\ldots x_{t2}x_{t2+1}\ldots x_N$.

Operația este prezentată grafic în figura următoare:



Structura unui algoritm genetic

Prezentăm cazul de bază al aplicării unui algoritm genetic.

Problema constă în aflarea valorii maxime a unei funcții de două variabile și a punctului în care se obține această valoare. Pe caz general avem: $f:(a, b) \to R$; se poate verifica faptul că ne putem restrânge la cazul în care intervalul (a, b) este intervalul $(0, 1)$.

Considerăm o valoare întreagă pozitivă n, de exemplu $n = 20$. Considerăm cele 2^n șiruri de lungime n cu elemente 0 și 1. Un astfel de șir poate fi considerat ca fiind partea fracționară în baza doi a unui număr din intervalul (0,1). (Pentru $n = 2$ avem: 00 este 0, 01 este 1/4, 10 este 1/2, iar 11 este 3/4). Putem considera că cele 2^n șiruri reprezintă tot atâtea puncte din intervalul (0, 1).

Un astfel de șir îl vom numi ***cromozom***.

O ***genă*** va fi o valoare (0 sau 1) de pe o anumită poziție.

Putem descrie acum structura generală a algoritmului genetic în pseudocod:

```
creare_populație_inițială(n,nr)        // pas 1
g ← 0
repetă
   g ← g + 1
   selectare_populație_intermediară()  // pas 2
   mutații_încrucișate(pc);            // pas 3
   mutații_simple(ps)                  // pas 4
până când g=gmax
afișare_rezultat()                     // pas 5
```

Pasul 1 - crearea populației inițiale

Considerăm o valoare *nr*, spre exemplu *nr* = 100, și generăm aleator un număr de *nr* cromozomi (șiruri) de lungime *n*. Numim această mulțime de șiruri populația inițială de cromozomi; *n* și *nr* sunt doi parametri ai algoritmului.

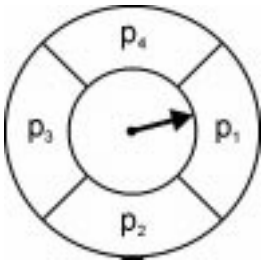
Vom analiza în cele ce urmează pașii din interiorul buclei. La fiecare iterație se trece de la generația curentă la generația următoare; *gmax* este un al treilea parametru reprezentând numărul maxim de generații.

La început generația curentă este populația inițială de cromozomi.

Pasul 2 - selectarea populației intermediare

Se evaluează populația curentă. În cazul de față aceasta constă în calcularea valorilor funcției *f* în cele *nr* puncte. Fie $v_i := f(c_i)$ pentru $i = 1, 2, \ldots, nr$. Un cromozom se va numi ***valoros*** dacă valoarea funcției în respectivul punct este mare. Fie $S := v1 + v_2 + \ldots + v_{nr}$. Considerăm probabilitățile $p_i := v_i / S$;

Aplicăm un mecanism de tip ruletă rusească prin care selectăm cu probabilitate mai mare cromozomii valoroși. Cu alte cuvinte, la fiecare pas vom genera un număr aleator între 0 și 1 și calculăm în ce zonă de probabilitate ne aflăm. Această operație este ilustrată grafic în figura următoare:



Ruleta va avea sloturi proporționale cu probabilitățile p_i, $i = 1, 2, \ldots, nr$ și o vom "întoarce" de *nr* ori .

Pasul 3 - aplicarea mutației încrucișate

Unui procent din populația intermediară i se vor aplica mutații încrucișate. Pentru a selecta acești cromozomi se stabilește un al patrulea parametru și anume *pc* (probabilitatea de mutație încrucișată) - are valoarea de aproximativ 0,1. Pentru fiecare cromozom al populației intermediare generăm un număr aleator între 0 și 1. Dacă acest număr este mai mic decât *pc*, cromozomul va fi selectat pentru mutații încrucișate.

Mutația propriu-zisă este de tip *crossover* și se realizează așa cum se va descrie în continuare.

Fie cromozomii selectați:

$c_i = x_1 x_2 \ldots x_N$

$c_j = y_1 y_2 \ldots y_N$

Generăm aleator un număr *t* din mulțimea 1, 2, …, *n*.

Cei doi cromozomi se vor înlocui cu următorii doi:

$c_i' = x_1 x_2 \ldots x_t y_{t1+1} \ldots y_N$

$c_j' = y_1 y_2 \ldots y_t x_{t1+1} \ldots x_N$

Pasul 4 - aplicarea mutației simple

Unui procent (mai mic) din populația intermediară, procent ce depinde de un parametru *ps* (probabilitate de mutație simplă), i se vor aplica mutații simple. Procedeul are loc așa cum se descrie în continuare.

Se va genera aleator un număr *t* din mulțimea 1, 2, …, *n*. Conținutul genei de pe poziția *t* se va modifica.

Observație

O caracteristică a algoritmilor genetici este că lucrează cu întrega populație; contează prea puțin cum evoluează indivizii, important este ca populația (luată ca un întreg) să evolueze. Nimeni nu garantează că o nouă generație va fi neapărat mai valoroasă, această evoluție are lor ca urmare a unor legi statistice.

În final se va ajunge la o populație care nu va mai evolua. Ultimul pas (al cincilea) este executat o singură dată.

Pasul 5 - selectarea rezultatului

După *gmax* generații, maximul funcției pe intervalul (0, 1) va fi aproximat de cel mai valoros cromozom din ultima generație. Acesta va fi afișat ca rezultat.

Analiza complexității

Crearea populației inițiale prin randomizare poate fi efectuată într-un timp de ordinul $O(n \cdot nr)$.

Presupunem că evaluarea funcției într-un punct necesită un timp constant. Luând în considerare și durata de calcul al punctului din intervalul (0, 1) asociat cromozomului, evaluarea populației curente are loc într-un timp de ordinul $O(n \cdot nr)$.

"Întoarcerea ruletei" se efectuează într-un timp de ordinul $O(nr)$.

Mutațiile de orice tip sunt realizate în timp constant, deci și pentru mutații avem un timp de ordinul $O(nr)$.

Rezultă că bucla repetitivă este executată într-un timp de ordinul $O(n \cdot nr \cdot gmax)$.

În concluzie, algoritmul este polinomial în parametrii *n*, *nr* și *gmax*. Observăm că durata de execuție este direct proporțională cu parametrul *gmax* (numărul maxim de generații).

Cazuri nedorite

Este posibil ca algoritmul să se "blocheze" în maxime locale. Pentru evitarea acestei probleme se recomandă respectarea următoarelor cerințe:

1. Alegerea de funcții de test semnificative;
2. Testarea algoritmilor utilizând și un alt generator (propriu) de numere aleatoare pe lângă apeluri ale funcției `random()`.
3. Alegerea parametrilor *n*, *nr*, *gmax*, *pc*, *ps* astfel încât să se obțină o bună convergență.

Indicație

Problema minimizării unei funcții *f* se reduce la maximizarea funcției *-f*.

Problema orarului

Descrierea problemei

Enunțul general al problemei este următorul [1]:

Se dă matricea încadrărilor, precizând pentru fiecare profesor numărul de ore la fiecare clasă. Cerința principală este aceea de a se obține un orar, adică o repartizare a profesorilor la clase, pentru fiecare dintre zilele săptămânii și, în cadrul fiecărei zile, pentru fiecare oră, astfel încât într-o aceeași oră (dintr-o anumită zi a săptămânii) fiecare profesor să intre la cel mult una din clasele asociate lui prin matricea de încadrare și la fiecare clasă să intre cel mult câte unul din profesorii asociați clasei.

În continuare vom considera un caz simplificat al acestei probleme, caz ce acoperă totuși o mulțime de situații concrete.

Vom considera următoarele date de intrare:

- *nr_cl*: numărul de clase din școală;
- *nr_ore*: numărul maxim de ore pe zi ale unui elev;
- *nr_zile*: numărul de zile în care sunt ore;
- *nr_p*: numărul de profesori;
- încadrarea profesorilor dată printr-o matrice cu *nr_p* linii și *nr_cl* coloane numită matricea încadrărilor notată în continuare cu *A*. Profesorii vor fi codificați cu numere *i* = 1, ..., *nr_p* iar clasele cu numere *j* = 1, ..., *nr_cl;*
- elevii din clasele 9-10 învață după-amiaza, iar cei din clasele 11-12 dimineața.

Exemplu

- *nr_cl* = 40; 40 de clase - acestea ar putea fi clasele 9A, ..., 9H, 10A, ..., 10H, 11A, ..., 11H, 12A, ..., 12H;
- *nr_ore* = 6 (șase ore pe zi);
- *nr_zile* = 5 (cinci zile pe săptămână);
- *nr_p* = 80 (80 de profesori - se păstrează și o listă cu numele lor: Aldea, Andrei, ..., Voicu);
- o linie a matricei încadrărilor ar putea indica, de exemplu, faptul că profesorul Aldea are următoarele ore: 5 ore la 10B, 4 ore la 11A, 3 ore la 12 C, 3 ore la 12D, 3 ore la 12E.

Orarul de funcționare se poate modela printr-o matrice cu *nr_p* linii și $2 \cdot nr\_ore \cdot nr\_zile$. Indicele liniei reprezintă codul profesorului, iar indicele coloanei un anumit moment de timp (zi și oră). Un element al acestei matrice va fi codul unei clase. O linie a matricei va indica programul pe zile al unui profesor, iar o coloană va indica profesorii ce au ore la momentul respectiv.

Condițiile pe care trebuie să la îndeplinească un orar de funcționare sunt următoarele:

- trebuie să fie conform cu încadrarea (linia de orar trebuie să exprime exact și încadrarea profesorului);
- la un anumit moment de timp (zi și oră), la o anumită clasă trebuie să predea cel mult un profesor;
- un profesor nu poate preda simultan la mai multe clase;
- elevii nu vor avea ferestre (ore libere între două ore la care li se predă).

Vom numi un astfel de orar ca fiind un ***orar valid***.

Pe lângă cele patru condiții absolut necesare ar fi de dorit ca profesorii să aibă un număr cât mai mic de ferestre, iar elevii să aibă cel mult două ore la aceeași disciplină în aceeași zi.

Un orar valid cu număr mic de ferestre pentru profesori va fi numit un ***orar bun***.

O posibilă rezolvare

Pentru început va trebui să găsim un orar valid de pornire. Există mai multe posibilități de determinare a unui asemenea orar.

O primă variantă poate fi găsită în [1]; trebuie menționat faptul că s-ar putea ca prin colorarea orelor și ulterior a *S*-orelor, să nu se obțină de la început o repartizare corectă (un orar valid).

De asemenea, se poate observa că obținerea *S*-orelor este realizată cu ajutorul unui algoritm care folosește metoda *backtracking*, deci consumă mult timp.

Putem lua în considerare o altă variantă de generare a unui asemenea orar de pornire.

Concret, folosind funcția `Random`, se vor așeza în matricea orarului, linie cu linie, pozițiile în care respectivul profesor ține orele prevăzute în încadrare.

Pentru fiecare linie se va utiliza un algoritm de generare a unui aranjament aleator de ore conform cu matricea încadrărilor.

Presupunem că orele unui anumit profesor sunt codificate cu numere cuprinse între 1 și *n*.

Un posibil algoritm ar fi:

```
pentru pas ← 1,n execută
  aleator ← random(nr_ore * nr_zile - pas + 1) + 1
  poz ← 0
  nr ← 0
  cât timp nr < aleator execută
    poz ← poz+1
    dacă a[poz]=0 atunci
      nr ← nr+1
    sfârşit dacă
  sfârşit cât timp
  a[poz] ← pas
sfârşit pentru
```

Inițial se vor "masca" pozițiile de dimineață pentru clasele 9-10 și pozițiile de după-amiază pentru clasele 11-12. Apoi, în raport cu numerele aleatoare generate, se va uti-

liza o mască de dimensiunea orarului. Pe fiecare linie pozițiile deja ocupate cu ore la o anumită clasă vor fi "mascate" pas cu pas.

Acestă generare este cu atât mai abordabilă cu cât vor exista clase care învață un număr de ore pe săptămână mai mic decât numărul maxim de ore prevăzut.

După generarea orarului de pornire, acesta trebuie optimizat în raport cu numărul de ferestre.

Un orar valid poate fi considerat ca fiind o matrice *B* cu *nr_p* linii și $2 \cdot nr\_ore \cdot nr\_zile$. Pentru exemplul considerat vor fi 80 de linii și 60 de coloane. 30 dintre coloane vor reprezenta ore de dimineață (cele care au un indice care împărțit la 12 dă un rest cuprins între 1 și 6), iar celelalte 30 ore de după-amiază. Un element al acestei matrice va fi codul unei clase (de la 1 la *nr_cl*) sau valoarea 0 dacă pe linia profesorului *p*, la timpul *t* nu este planificată desfășurarea unei ore.

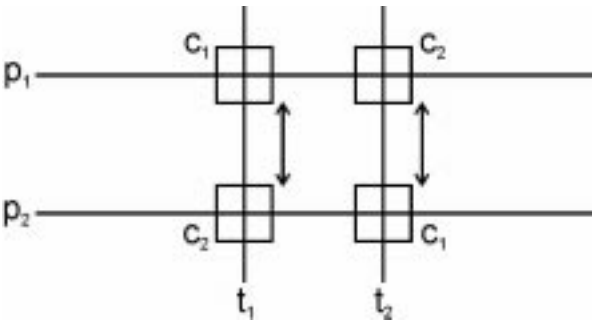
O linie a matricei *B* va fi considerată cromozom. Pentru exemplul ales, lungimea cromozomului este 60.

O genă va putea lua valori cuprinse între 0 și *nr_cl*. Vor putea fi 41 de valori distincte pentru gene.

Dimensiunea populației este egală cu *nr_p*. În cazul acestui exemplu, valoarea este 80.

Ca mutații încrucișate putem considera următorul tip de *crossover*:

- fie profesorii p_1 și p_2 care la timpii t_1 și t_2 au ore la clasele c_1 și c_2; cu alte cuvinte vom avea $B(p_1, t_1) = B(p_2, t_2) = c_1$ și $B(p_1, t_2) = B(p_2, t_1) = c_2$; cromozomii corespunzători liniilor p_1 și p_2 se vor încrucișa interschimbând genele de pe pozițiile t_1 si t_2; operația poate fi reprezentată grafic astfel:



Observăm că prin astfel de mutații se păstrează validitatea orarelor.

Este greu de găsit un tip de mutație simplă care să conserve proprietatea unui orar de a fi valid .

Funcția de minimizat este funcția care calculează numărul de ferestre al unui profesor.

Această funcție se poate eventual pondera, în sensul ca o fereastră de două ore să fie echivalentă, de exemplu, cu trei ferestre de câte o oră.

Construirea unui orar bun va însemna optimizarea prin algoritmul genetic a unui orar valid de pornire. Optimizarea va avea ca scop reducerea numărului de ferestre ale profesorilor.

Observații

- Algoritmul descris poate fi implementat astfel încât să se execute într-un timp de ordinul $O(nr\_p \cdot nr\_ore \cdot nr\_zile \cdot gmax)$.
- Conform celor prezentate anterior, putem varia valorile parametrilor *pc* și *gmax* în vederea obținerii unei bune convergențe (mai puține generații construite până la obținerea unui orar bun).
- Este esențială testarea algoritmului pe date de dimensiune reală, deoarece convergența algoritmilor genetici depinde, într-o măsură destul de mare, de valorile parametrilor *n* și *nr*.

În cele ce urmează vom prezenta câteva modificări care ar putea fi aduse algoritmului prezentat. Fie M_0, M_1, M_2,... șirul de orare generate de algoritm. M_g va reprezenta populația din generația *g*. În scopul îmbunătățirii performanțelor pot fi luate în considerare și următoarele modificări la descrierea generală a algoritmilor genetici.

O posibilitate de optimizare implică introducerea a încă doi parametri întregi pozitivi *p* și *delta*.

Fie *g* generația curentă. În cadrul buclei repetitive se vor executa următoarele operații:

- salvarea populației corespunzătoare generației curente *g* (M_g);
- generarea a *delta* noi populații conform algoritmului prezentat; noile orare vor fi identificate prin M_{g+1}, M_{g+2}, ..., $M_{g+delta}$;.
- păstrarea celei mai valoaroase generații, care va deveni generația $M_{g+delta}$;.
- creșterea numărului de generații: $g \leftarrow g + delta$.

O a doua posibilitate de optimizare implică introducerea parametrului întreg *delta* și a unui parametru *S* care este un șir de numere naturale.

Fie *S* un șir descrecător de numere naturale care conține *gmax* elemente. În cadrul buclei repetitive se vor executa următoarele operații:

- salvarea populației corespunzătoare generației curente *g* (M_g);
- generarea a *delta* noi populații conform algoritmului prezentat; noile orare vor fi identificate prin M_{g+1}, M_{g+2}, ..., $M_{g+delta}$;
- repetarea pasului anterior până când numărul total de ferestre scade sub valoarea S_g; populația corespunzătoare devine generația $M_{g+delta}$;
- creșterea numărului de generații: $g \leftarrow g + delta$.

Bibliografie:

[1]. ***Vlad Bazon***, *Un model pentru problema orarului*, GInfo 9/8 (decembrie 1999);

[2]. ***Ricardo Poli***, *Genetic algorithms*, http://www.cs.bham.ac.uk/~rmp/slide_book/node2.html;

[3]. ***George Păun***, *Calculatoare pe bază de ADN*, GInfo 9/3 (martie 1999).

Dl. prof. Radu Vișinescu este cadru didactic la Liceul "Ion Luca Caragiale" din Ploiești. Poate fi contactat prin e-mail la raduv@lphilc.sfos.ro.