

Algoritmi genetici - structura generală și operatori genetici

noiembrie 2006

1 Introducere

Algoritmii genetici (AG) fac parte din clasa mai generală a algoritmilor evolutivi (AE). AE sunt metode de rezolvare a problemelor bazate pe o căutare în spațiul soluțiilor, în care se folosesc populații de "căutatori" supuse unui proces de *evoluție* oarecum similar celui întâlnit în natură.

AG au fost propuși de către John Holland (în perioada anilor '70), inițial ca modele formale ale proceselor de evoluție din natură, scopul urmărit de Holland fiind dezvoltarea unor sisteme artificiale adaptive. Ulterior AG s-au dovedit metode robuste de rezolvare a problemelor dificile pentru care nu se cunosc algoritmi specifici eficienți.

Punctul de pornire în conceperea algoritmilor genetici îl reprezintă analogia care se poate face între rezolvarea unei probleme și procesul natural de evoluție al unei populații.

Rezolvare prin căutare. Pentru unele probleme, rezolvarea constă în găsirea unei configurații (element al spațiului soluțiilor posibile) care satisface anumite condiții (respectă restricții specifice problemei și/sau optimizează un criteriu). În această categorie intră clasică problemă a comisvoiajorului (configurația căutată fiind un circuit în graful asociat problemei care satisface restricția de a trece o singură dată prin fiecare nod și minimizează costul traseului) precum și alte probleme de optimizare combinatorială. O metodă generală de rezolvare a acestor probleme este o *strategie de căutare* în spațiul configurațiilor posibile. Căutarea se poate realiza folosind un singur *explorator* (ca în strategiile de tip gradient sau în cele de tip "simulated annealing") sau o *populație de exploratori* (în cazul algoritmilor evolutivi).

În oricare dintre situații o căutare eficientă presupune asigurarea unui echilibru între următoarele procese:

- *Explorarea* spațiului de căutare. Permite descoperirea de noi configurații sau de căi care conduc la configurații mai bune. Este asigurată atunci când nu se impun condiții prea restrictive la construirea unei noi configurații. Explorarea creează premisele unei căutări cu caracter global.
- *Exploatarea* informației despre problemă dobândită până la momentul curent al căutării. Este asigurată atunci când în stabilirea unei noi configurații se utilizează informațiile colectate de-a lungul procesului de căutare. Exploatarea pune accentul pe căutarea locală.

Algoritmii de tip gradient se caracterizează printr-o bună exploatare dar printr-o slabă putere de explorare. Algoritmii de căutare pur aleatoare (care nu sunt neapărat metode de descoperire) asigură, dimpotrivă, o bună explorare a spațiului configurațiilor însă nu exploatează deloc informația colectată pe parcursul căutării.

Dacă strategia de răcire este bine aleasă, algoritmii de tip "simulated annealing" asigură un echilibru între explorare (în prima fază, când parametrul T este mare) și exploatare (în a doua fază, când T devine mic).

Strategie de căutare	Proces de evoluție
Informații despre problemă	Mediu
Configurație	Individ (cromozom)
Populație de căutători	Populație de indivizi
Funcție obiectiv	Fitness (grad de adaptare la mediu)
Mecanism de exploatare	Selecție
Mecanism de explorare	Reproducere (încrucișare și mutație)

Tabelul 1: Analogia dintre strategiile de căutare și procesele de evoluție

Evoluția în natură. Principalele noțiuni care permit analogia între rezolvarea problemelor de căutare și evoluția naturală sunt:

- *Populație.* O populație este constituită din *indivizi* care trăiesc într-un mediu la care trebuie să se adapteze.
- *Fitness.* Fiecare individ al populației este adaptat mai mult sau mai puțin mediului. Fitness-ul (adecvarea) este o măsură a gradului de adaptare la mediu. Scopul evoluției este ca toți indivizii să ajungă la o adecvare cât mai bună la mediu.
- *Cromozom.* Este o mulțime ordonată de elemente, numite *gene* ale căror valori determină caracteristicile unui individ. În genetică pozițiile pe care se află genele în cadrul cromozomului se numesc *loci*, iar valorile pe care le pot lua se numesc *alele*.
- *Generație.* Etapă în evoluția unei populații. Dacă vedem evoluția ca un proces iterativ în care o populație se transformă în altă populație atunci generația este o iterație în cadrul acestui proces.
- *Selecție.* Procesul de selecție naturală are ca efect supraviețuirea indivizilor cu grad ridicat de adecvare la mediu (fitness mare).
- *Reproducere.* Este procesul prin care se trece de la o generație la alta. Indivizii noii generații moștenesc caracteristici de la precursorii lor (*părinți*) dar pot dobândi și caracteristici noi ca urmare a unor procese de *mutație* care au un caracter întâmplător. În cazul în care în procesul de reproducere intervin cel puțin doi părinți caracteristicile moștenite ale urmașului (*fiu*) se obțin prin *combinarea (încrucișarea)* caracteristicilor părinților.

Analogia dintre strategiile de căutare și procesele de evoluție naturală este ilustrată în tabelul 1.

2 Structura generală a unui algoritm genetic

Ca majoritatea algoritmilor evolutivi și algoritmii genetici sunt procese iterative prin care o populație inițializată în manieră aleatoare este transformată succesiv prin selecție, mutație și încrucișare până la atingerea unui anumit număr de iterații (generații) sau până la îndeplinirea unui alt criteriu de oprire.

Datorită flexibilității lor, există la ora actuală un număr foarte mare de variante de algoritmi genetici. De fapt pentru fiecare problemă concretă apar elemente particulare în cadrul

Inițializari: $P(0) = (x_1(0), \dots, x_m(0)), t = 0$

Proces iterativ:

Repetă

Evaluarea populației curente: calcul $f(x_i(t))$ pentru $i = \overline{1, m}$

Selecția părinților: $P(t) \rightarrow P^1$

Generarea urmașilor prin încrucișare: $P^1 \rightarrow P^2$

Modificarea urmașilor prin mutație: $P^2 \rightarrow P^3$

Evaluarea populației de urmași: calcul valori ale lui f pentru elementele lui P^3

Selecția supraviețuitorilor: $\{P(t), P^3\} \rightarrow P(t+1)$

Incrementarea contorului de generații: $t = t + 1$

până când $t > t_{max}$

Figura 1: Structura generală a unui algoritm genetic.

algoritmului. Totuși, majoritatea se pot încadra în structura generală ilustrată în figura 1, în care $P(t) = (x_1(t), x_2(t), \dots, x_m(t))$ reprezintă populația corespunzătoare generației t iar $f(x_i)$ reprezintă gradul de adecvare (fitness-ul) elementului x_i .

La proiectarea unui algoritm genetic trebuie să se stabilească:

- *Modul de codificare.* Se specifică modul în care fiecărei configurații din spațiul de căutare i se asociază un cromozom.
- *Funcția de adecvare.* Se construiește funcția care exprimă gradul de adecvare la mediu pornind de la restricțiile și funcția obiectiv a problemei.
- *Dimensiunea și modul de inițializare a populației.* Se pot utiliza populații de dimensiune fixă sau de dimensiune variabilă. De cele mai multe ori populația se inițializează în manieră aleatoare cu elemente din spațiul de căutare.
- *Mecanismul de selecție a părinților și a supraviețuitorilor.*
- *Mecanismul de încrucișare a părinților pentru a genera urmași.*
- *Mecanismul de mutație care asigura perturbarea elementelor.*
- *Criteriul de oprire.* Atunci când nu se cunoaște un criteriu specific problemei se optează pentru un număr maxim de iterații. Se pot folosi informații despre populație, cum ar fi gradul de diversitate al acesteia.

Legatura între problema de rezolvat și elementele algoritmului genetic este ilustrată în figura 2.

Este una dintre etapele cele mai importante ale proiectării unui algoritm evolutiv întrucât determină modul în care se va desfășura procesul de evoluție. Se referă la următoarele aspecte: *structuri de date folosite, regula de codificare și regula de decodificare.*

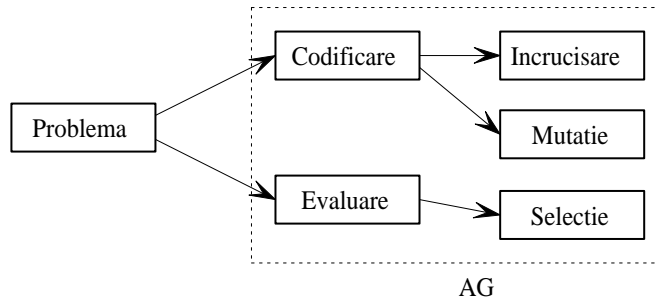


Figura 2: Legături între elementele unui algoritm genetic și problema de rezolvat

2.1 Structuri de date.

În algoritmii genetici și strategiile evolutive cromozomii sunt reprezentați prin structuri liniare cu număr fix de elemente (tablouri) sau cu număr variabil de elemente (liste înlănțuite). Prima variantă este cea mai frecvent folosită. Structuri de alt tip (de exemplu, arborescente) se folosesc în alte metode evolutive (de exemplu, în programarea genetică).

2.2 Reguli de codificare.

Modul în care o configurație este codificată într-un cromozom depinde de problema concretă. Există trei variante principale de codificare:

Codificare binară. Este varianta clasică. În acest caz cromozomii sunt vectori cu elemente din $\{0, 1\}$ iar spațiul de căutare este $S = \{0, 1\}^n$, cu n numărul de gene (n este corelat cu dimensiunea problemei). Este adecvată pentru problemele de optimizare combinatorială în care configurațiile pot fi specificate ca vectori binari.

Exemplul 1. Problema maximizării numărului de biți egali cu 1 (ONEMAX problem). Se pune problema determinării șirului de biți (x_1, \dots, x_n) , $x_i \in \{0, 1\}$ care maximizează funcția $f : \{0, 1\}^n \rightarrow \mathbb{N}$, $f(x_1, \dots, x_n) = \sum_{j=1}^n x_j$. Problema este echivalentă cu a determina configurația de biți care are cele mai multe elemente egale cu 1. Aceasta este evident $(1, \dots, 1)$ problema fiind una artificială folosită pentru a testa algoritmii genetici. Ori de câte ori se pune problema reprezentării unei funcții definite pe $\{0, 1\}^n$ reprezentarea binară este evidentă.

Exemplul 2. Problema submulțimii de sumă maximă limitată de un prag. Se consideră o mulțime $W = \{w_1, \dots, w_n\}$ de valori întregi și M o valoare întreagă. Se caută o submulțime $S \subset W$ cu proprietatea că suma elementelor lui S este cât mai apropiată de M dar nu îl depășește. Orice submulțime S poate fi reprezentată printr-un vector (s_1, s_2, \dots, s_n) cu $s_i = 0$ dacă $w_i \notin S$ și $s_i = 1$ dacă $w_i \in S$. Suma elementelor unei submulțimi S este în acest caz $\sum_{j=1}^n w_j s_j$.

Exemplul 3. Problema rucsacului. Se consideră un set de n obiecte caracterizate prin greutatea (w_1, \dots, w_n) și prin valorile (v_1, \dots, v_n) . Se pune problema determinării unui subset de obiecte pentru a fi introduse într-un rucsac de capacitate C astfel încât valoarea obiectelor selectate să fie maximă. O soluție a acestei probleme poate fi codificată ca un șir de n valori binare în felul următor: $s_i = 1$ dacă obiectul i este selectat, respectiv $s_i = 0$ dacă obiectul nu este selectat.

Exemplul 4. Problema împachetării. Se consideră un set de n obiecte caracterizate prin dimensiunile (d_1, d_2, \dots, d_n) și un set de m cutii având capacitățile (C_1, C_2, \dots, C_m) . Se pune problema plasării obiectelor în cutii astfel încât capacitatea acestora să nu fie depășită iar numărul de cutii uti-

Reprezentare clasică							
0	1	2	3	4	5	6	7
000	001	010	011	100	101	110	111
Codificarea Gray							
0	1	2	3	4	5	6	7
000	001	011	010	110	111	101	100

Tabelul 2: Reprezentări binare: reprezentarea clasică în baza 2 și codul Gray.

lizate să fie cât mai mic. O posibilă reprezentarea binară pentru această problemă este următoarea: se utilizează o matrice cu n linii și m coloane iar elementul s_{ij} are valoarea 1 dacă obiectul i este plasat în cutia j și 0 în caz contrar (obiectul i nu este plasat în cutia j). Prin liniarizarea matricii se ajunge ca fiecare soluție să fie reprezentată printr-un cromozom conținând mn gene.

Exemplul 5. Optimizarea unei funcții definite pe un domeniu continuu. Se consideră o funcție $f : D = [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_n, b_n] \subset \mathbb{R}^n \rightarrow \mathbb{R}$ și se caută $x^* = (x_1^*, \dots, x_n^*)$ care minimizează pe f ($f(x^*) \leq f(x)$ pentru orice $x \in D$). În acest caz codificarea binară nu este evidentă. Fiecare dintre componentele x_i ale lui x sunt transformate după cum urmează:

- (i) se scalează pentru a fi aduse în intervalul $[0, 1)$: $\bar{v}_i = (x_i - a_i)/(b_i - a_i)$;
- (ii) se stabilește numărul de biți utilizați pentru reprezentare (r) și se aduce valoarea v_i în mulțimea $\{0, 1, \dots, 2^r - 1\}$: $u_i = [v_i * (2^r - 1)]$. Valoarea obținută se reprezintă în baza 2. De exemplu, pentru $x = (1.25, 2.3) \in [1, 2] \times [2, 3]$ și $r = 5$ se obține aproximativ $\bar{x} = (0.25, 0.3)$ iar cromozomul asociat va avea $2r = 10$ componente binare: $(0, 0, 1, 1, 1, 0, 1, 0, 0, 1)$. Este evident că această codificare are caracter aproximativ, motiv pentru care în cazul variabilelor reale se preferă utilizarea codificării reale.

Utilizarea reprezentării binare în cazul în care configurațiile corespunzătoare problemei sunt vectori de valori reale prezintă dezavantajul că valori reale aflate la distanță mică au asociate reprezentări binare aflate la distanță mare (diferă în multe poziții binare). De exemplu reprezentarea lui 7 pe 4 biți este 0111 iar reprezentarea lui 8 este 1000. Se remarcă faptul că trecerea de la reprezentarea lui 7 la cea a lui 8 necesită nu mai puțin de 5 complementări de biți. O altă variantă de codificare care evită acest dezavantaj este codificarea de tip Gray caracterizată prin faptul că valori întregi succesive au asociate șiruri de biți care diferă într-o singură poziție.

Pornind de la reprezentarea binară (b_1, \dots, b_r) (cifrele binare sunt specificate începând cu cea mai semnificativă), codul Gray, (a_1, \dots, a_r) , se construiește după regula:

$$a_i = \begin{cases} b_i & i = 1 \\ b_{i-1} \oplus b_i & i > 1 \end{cases}$$

unde \oplus reprezintă adunarea modulo 2.

Codificare reală. Este adecvată pentru problemele de optimizare pe domenii continue (vezi exemplul 5 de mai sus). În acest caz cromozomii sunt vectori cu elemente reale, fiind chiar elementele domeniului de definiție al funcției (pentru exemplul de mai sus cromozomul este chiar $x = (1.25, 2.3)$). Avantajul acestei reprezentări este faptul că este naturală și nu necesită proceduri speciale de codificare/decodificare.

Codificare specifică. Se alege o variantă cât mai apropiată de specificul problemei.

Exemplul 1. Problema împachetării. Considerăm din nou problema împachetării. În varianta de codificare binară propusă mai sus apare dezavantajul că pot fi generate configurații care nu sunt fezabile. Acestea sunt de exemplu matricile care conțin mai multe valori egale cu 1 pe o linie (aceasta ar însemna că un obiect este simultan inclus în mai multe cutii). Pentru a evita astfel de situații se poate utiliza un alt tip de reprezentare: un vector cu n componente (s_1, \dots, s_n) în care $s_i \in \{1, \dots, m\}$ reprezintă cutia în care este inclus obiectul i .

Exemplul 2. Problema comis-voiajorului. Se consideră un set de n orașe și se pune problema găsirii unui traseu care să treacă o singură dată prin fiecare oraș și care să aibă costul minim (dacă costul este proporțional cu lungimea traseului atunci se caută trasee de lungime minimă). O codificare naturală a unei configurații (traseu) este: (s_1, \dots, s_n) unde $s_i \in \{1, \dots, n\}$ reprezintă numărul de ordine al orașului vizitat la etapa i (la fiecare etapa comis-voiajorul se află într-un oraș). Pentru a fi respectată restricția ca fiecare oraș să fie vizitat o singură dată este necesar ca elementele vectorului s să fie distincte ($s_i \neq s_j$ pentru orice $i \neq j$). Astfel fiecare configurație poate fi interpretată ca o permutare de ordin n , motiv pentru care această codificare este numită și codificare de tip permutare.

2.3 Reguli de decodificare.

Decodificarea asigură pregătirea evaluării configurației. Ea realizează translatarea din domeniul de valori specifice genotipului în cel corespunzător fenotipului.

Atenție specială ridică decodificarea doar în cazul codificării binare a unor vectori cu valori reale (vezi exemplul 5 de la codificarea binară). În cazul codificării binare clasice a unei valori din intervalul $[a, b]$ pentru a obține valoarea decodificată pornind de la șirul de r biți obținut prin codificare se folosește relația:

$$\delta(s_1, \dots, s_r) = a + \frac{b-a}{2^r - 1} \sum_{j=1}^r s_j 2^{j-1}$$

În cazul codificării de tip Gray decodificarea se bazează pe relația:

$$\delta(s_1, \dots, s_r) = a + \frac{b-a}{2^r - 1} \sum_{j=1}^r \left(\bigoplus_{k=1}^j s_k \right) 2^{j-1}$$

3 Construirea funcției de adecvare

În procesul de evoluție naturală se urmărește maximizarea gradului de adecvare a indivizilor la mediu. Pentru a ne folosi de analogia dintre procesele de căutare și cele de evoluție din natură este util să reformulăm problemele de optimizare ca probleme de maximizare (orice problemă de minimizare poate fi transformată într-una de maximizare prin schimbarea semnului funcției obiectiv). Pentru o problemă de minimizare de forma: să se determine $x^* \in D$ cu proprietatea că $f(x^*) \leq f(x)$ pentru orice $x \in D$, funcția de adecvare are fi $F(x) = -f(x)$. În cazul unei probleme de maximizare atunci funcția de adecvare poate fi chiar funcția obiectiv.

Lucrurile devin mai complicate în cazul problemelor cu restricții. Să considerăm o problemă de minimizare cu restricții: să se determine $x^* \in D$ cu proprietatea că minimizează funcția obiectiv $f : D \rightarrow R$ și satisface restricțiile:

$$g_j(x) = 0, \quad j = \overline{1, k_1} \text{ (restricții de tip egalitate)}$$

$$h_j(x) \geq 0, \quad j = \overline{1, k_2} \text{ (restricții de tip inegalitate)}$$

O variantă de a trata restricțiile este de a folosi tehnica penalizării care permite includerea acestora în cadrul funcției de adecvare:

$$F(x) = -f(x) - a \sum_{j=1}^{k_1} (g_j(x))^2 - b \sum_{j=1}^{k_2} \varphi(h_j(x))$$

unde

$$\varphi(u) = \begin{cases} u^2 & u < 0 \\ 0 & u \geq 0 \end{cases}$$

iar a și b sunt parametri pozitivi care reflectă importanța relativă a încălcării restricțiilor (cu cât a și b sunt mai mari cu atât restricția este mai importantă și încălcarea ei este penalizată mai mult. Dacă se fac diferențieri între restricții atunci se pot utiliza mai multe valori $(a_1, \dots, a_{k_1}, b_1, \dots, b_{k_2}$ în loc de a respectiv b).

Exemplul 1. Problema ONEMAX. În acest caz funcția de adecvare coincide cu funcția obiectiv a problemei, calitatea unei soluții fiind reflectată de numărul de componente egale cu 1.

Exemplul 2. Problema submulțimii. Un vector $s = (s_1, \dots, s_n)$ este admisibil (soluție potențială) doar dacă satisface $\sum_{i=1}^n w_i s_i \leq M$. Funcția de adecvare trebuie să ia în considerare dacă restricția este sau nu încălcată:

$$F(s) = \begin{cases} \sum_{i=1}^n w_i s_i & \text{dacă } s \text{ este admisibilă} \\ -\sum_{i=1}^n w_i s_i & \text{dacă } s \text{ nu este admisibilă} \end{cases}$$

Modul de specificare a funcției de adecvare sugerează că dacă o soluție este acceptabilă atunci ea este cu atât mai bună cu cât se apropie mai mult de optim (în cazul nostru înseamnă că suma este cât mai apropiată de M). Dacă soluția nu este admisibilă atunci ea este cu atât mai bună cu cât abaterea față de o soluție admisibilă este mai mică. Această tehnică de a trata diferit soluțiile admisibile și cele neadmisibile permite compararea între ele a soluțiilor. Regulile naturale sunt următoarele:

- orice soluție admisibilă este mai bună decât o soluție care nu este admisibilă;
- dintre două soluții admisibile este mai bună cea care are adecvarea mai mare (în cazul problemei submulțimii aceasta înseamnă că suma valorilor elementelor submulțimii este mai mare);
- dintre două soluții neadmisibile cea mai bună este cea care încalcă mai puțin restricțiile (în cazul acestei probleme înseamnă că suma elementelor este mai mică - depășește cu mai puțin pragul M).

Exemplul 3. Problema rucsacului. Fiind o problemă cu restricții trebuie luat în considerare și cazul configurațiilor ce nu sunt admisibile. O posibilă funcție de adecvare este:

$$F(s) = \begin{cases} \sum_{j=1}^n v_j s_j & \text{dacă } \sum_{j=1}^n w_j s_j \leq C \\ \sum_{j=1}^n v_j s_j - b(\sum_{j=1}^n w_j s_j - C) & \text{dacă } \sum_{j=1}^n w_j s_j > C \end{cases}$$

Exemplul 4. Problema comis-voiajorului. În acest caz, dacă se folosește reprezentarea de tip permutare restricția problemei (trecerea o singură dată prin fiecare oraș) este implicit satisfăcută. Dacă matricea C conține costurile $(c(i, j))$ reprezintă costul trecerii de la orașul i la orașul j) atunci funcția de cost poate fi descrisă prin:

$$f(s_1, \dots, s_n) = \sum_{i=1}^{n-1} c(s_i, s_{i+1}) + c(s_n, s_1).$$

În condițiile în care costul trebuie minimizat funcția de adecvare va fi chiar opusa funcției cost: $F(s) = -f(s)$.

4 Selecție

Selecția are ca scop determinarea populației intermediare ce conține părinții care vor fi supuși operatorilor genetici de încrucișare și mutație precum și determinarea elementelor ce vor face parte din generația următoare. Criteriul de selecție se bazează pe gradul de adecvare al configurației la cerințele problemei, exprimat prin valoarea funcției de adecvare (fitnessului).

Nu este obligatoriu ca atât părinții cât și supraviețuitorii se fie determinați prin selecție, fiind posibil ca aceasta să fie folosită doar într-o singură etapă. De exemplu, toți indivizii populației curente sunt potențiali părinți dar după încrucișare și mutație doar cei determinați prin procesul de selecție vor supraviețui. Pe de altă parte este posibil ca părinți să fie doar indivizii selectați iar toți cei generați prin încrucișare și mutație să fie transferați în noua generație.

Procesul de selecție nu depinde de modul de codificare a elementelor populației fiind însă legat de funcția de adecvare.

Există două clase principale de metode de selecție:

- *Metode deterministe.* Se caracterizează prin faptul că elementele cu grad mare de adecvare sunt întotdeauna selectate în defavoarea celor cu grad mai mic de adecvare. Un exemplu de astfel de metodă este *selecția prin trunchiere*.
- *Metode aleatoare.* Se caracterizează prin faptul că în procesul de selecție sunt introduse elemente aleatoare. Variantele cel mai frecvent întâlnite se bazează pe stabilirea unor probabilități de selecție care depind de gradul de adecvare. În felul acesta elementele cu grad mare de adecvare au șanse mai mari de a fi selectate, astfel că numărul de copii ale acestora este mai mare decât al celor cu grad mai mic de adecvare. Metodele cele mai reprezentative sunt *selecția proporțională*, *selecția pe baza rangurilor* și *selecția de tip turneu*.

Diferitele variante de selecție diferă între ele prin *presiunea de selecție*. Aceasta este corelată cu numărul de generații în care populația este constituită din copii ale celui mai bun element. Cu cât presiunea de selecție este mai mare cu atât numărul de generații până când populația devine uniformă (toate elementele sunt identice cu cel mai bun) este mai mic. O presiune prea mare de selecție induce convergența forțată într-o configurație sub-optimală, convergență numită prematură.

Pe lângă presiunea de selecție, metodele de sortare se mai pot diferenția prin proprietatea de *elitism*. Prin elitism se înțelege supraviețuirea celui mai bun dintre elementele generate până la un moment dat. Nu toate variantele de selecție asigură satisfacerea acestei proprietăți. De exemplu, selecția prin trunchiere este elitistă, pe când cea de tip turneu nu este elitistă. O metodă de selecție poate fi transformată într-una elitistă prin amplasarea explicită a celui mai bun element al populației curente în populația corespunzătoare generației următoare.

4.1 Selecția proporțională

Fie $P(t) = (x^1, \dots, x^m)$ populația curentă și fie $F = (F^1, \dots, F^m)$ valorile corespunzătoare ale funcției de adecvare. Presupunem că $F_i > 0$ pentru $i = \overline{1, m}$ (în caz contrar se realizează o

```

i := 1
s := pi
generează u uniform aleator în [0, 1]
WHILE u > s DO
    i := i + 1
    s := s + pi
RETURN i

```

Figura 3: Metoda selecției proporționale.

transformare prin care toți F_i devin pozitivi - se adună la toate elementele o valoare ușor mai mare decât modulul valorii minime). Se construiește distribuția de probabilitate:

$$\begin{pmatrix} 1 & 2 & \dots & i & \dots & m \\ p_1 & p_2 & \dots & p_i & \dots & p_m \end{pmatrix} \quad \text{cu } p_i = \frac{F_i}{\sum_{j=1}^m F_j}$$

Folosind distribuția de mai sus se generează valori aleatoare, utilizând, de exemplu, metoda inversării funcției de repartiție. Ideea de bază a acestei metode este:

- Se construiește tabelul de valori corespunzătoare funcției de repartiție (prin cumularea probabilităților):

$$\begin{pmatrix} 1 & 2 & \dots & i & \dots & m \\ pc_1 & pc_2 & \dots & pc_i & \dots & pc_m \end{pmatrix}$$

unde $pc_1 = p_1$, $pc_2 = p_1 + p_2$, \dots , $pc_i = \sum_{j=1}^i p_j$, \dots , $pc_m = 1$.

- Se generează o valoare aleatoare, u , uniform repartizată în $[0, 1]$.
- Se caută intervalul $(pc_i, pc_{i+1}]$ care îl conține pe u ($pc_i < u \leq pc_{i+1}$). Valoarea $i + 1$ reprezintă indicele elementului care trebuie selectat.

Un model intuitiv simplu al selecției proporționale este cel al ruletei: se consideră o ruletă împărțită în m sectoare, fiecare având aria proporțională cu valoarea funcției de adecvare a elementului corespunzător al populației. Rotirea ruletei este echivalentă generării unei valori aleatoare iar sectorul în dreptul căruia rămâne indicatorul indică elementul din populație care trebuie selectat. O variantă a metodei selecției proporționale, care nu necesită cumularea prealabilă a probabilităților, este ilustrată în fig. 3.

În cazul în care se dorește selecția unei întreg set de părinți sau urmași se poate utiliza o variantă a metodei cunoscută sub denumirea de algoritmul SUS ("stochastic universal sampling"). Acest algoritm returnează un tablou (c_1, \dots, c_m) în care c_i reprezintă numărul de copii ale elementului i al populației iar $\sum_{i=1}^m c_i = q$ unde q este numărul de elemente selectate. Algoritmul este descris în fig 4.

Exemplu. Considerăm problema ONEMAX pentru $n = 100$. Dacă populația are $m = 5$ elemente iar acestea au valorile (numărul de componente egale cu 1): $f(x_1) = 10$, $f(x_2) = 5$, $f(x_3) = 30$, $f(x_4) = 25$ și $f(x_5) = 50$ atunci valorile probabilităților de selecție sunt: $p_1 = 1/12$, $p_2 = 1/24$, $p_3 = 1/4$, $p_4 = 5/24$, $p_5 = 5/12$.

```

generează  $u$  uniform aleator în  $[0, 1/q]$ 
 $s := 0$ 
FOR  $i := 1, m$  DO
   $c_i := 0$ 
   $s := s + p_i$ 
  WHILE  $u < s$  DO
     $c_i := c_i + 1$ 
     $u := u + 1/q$ 
RETURN  $c$ 

```

Figura 4: Algoritmul SUS (Stochastic Universal Sampling).

Numărul mediu de copii ale unui element al populației depinde de valoarea funcției de adecvare. Atunci când există o diferență mare între valoarea funcției de adecvare a celui mai bun element și valorile celorlalte elemente probabilitatea de selecție a acestuia va fi mult mai mare decât cea a celorlalte elemente astfel că populația constituită prin selecție poate conține doar copii ale celui mai bun element (acest lucru elimină diversitatea populației și stopează procesul de evoluție).

Pentru a evita astfel de situații se aplică diverse tehnici de scalare ale funcției de adecvare sau în calculul probabilităților de selecție nu se utilizează direct valoarea funcției de adecvare ci doar relația de ordine existentă între valorile diferitelor elemente. Pe această idee se bazează metoda rangurilor.

4.2 Selecția pe baza rangurilor

Se ordonează crescător valorile funcției de adecvare pentru toate elementele populației. Se rețin valorile distincte și li se asociază câte un rang (cea mai mică valoare are rangul 1, iar cea mai mare are rangul maxim). Se partiționează populația în grupuri de elemente care au aceeași valoare a funcției fitness și li se asociază rangul corespunzător valorii. Presupunând că sunt k grupuri se construiește distribuția de probabilitate:

$$\left(\begin{array}{cccccc} 1 & 2 & \dots & i & \dots & k \\ p_1 & p_2 & \dots & p_i & \dots & p_k \end{array} \right) \quad \text{cu } p_i = \frac{i}{\sum_{j=1}^k j}$$

Se generează câte un rang aleator în conformitatea cu distribuția de mai sus (folosind metoda ruletei) după care se selectează uniform aleator un element din grupul corespunzător rangului.

O altă modalitate de stabilire a probabilităților de selecție pornind de la rangul elementelor este următoarea. Se ordonează crescător cele m elemente ale populației și i se asociază câte un rang (0 pentru elementul cu cel mai mic grad de adecvare și $m - 1$ pentru elementul cu cel mai mare grad de adecvare). Probabilitatea de selecție a elementului i se calculează astfel:

$$p_i = \frac{\alpha + (\text{rang}(i)/(m - 1)) * (\beta - \alpha)}{m}$$

cu $\alpha < \beta$ valori alese astfel încât $\sum_{i=1}^m p_i = 1$, ceea ce implică $\alpha + \beta = 2$. α poate fi interpretat ca numărul mediu de copii ale celui mai slab element iar β ca numărul mediu de copii ale celui mai bun

```

FOR  $i := 1$  TO  $m$  DO
   $rmin =$ valoare aleatoare din  $\{1, \dots, m\}$ 
  FOR  $j := 2$  TO  $k$  DO
     $r =$ valoare aleatoare din  $\{1, \dots, m\}$ 
    IF  $F(x_{rmin}) > F(x_r)$  THEN  $rmin = r$ 
   $ind_i = rmin$ 
RETURN  $ind_1, \dots, ind_m$ 

```

Figura 5: Algoritmul selecției de tip turneu.

element. Folosind probabilitățile astfel calculate selecția poate fi realizată utilizând fie algoritmul ruletei fie algoritmul SUS.

Exemplu. Considerăm exemplul din secțiunea anterioară. Aplicând prima variantă de determinare a probabilităților de selecție se obțin: $p_1 = 2/15$, $p_2 = 1/15$, $p_3 = 4/15$, $p_4 = 3/15$ și $p_5 = 5/15$. În cel de al doilea caz, considerând $\alpha = 0$ și $\beta = 2$ se obțin probabilitățile: $p_1 = 1/10$, $p_2 = 0$, $p_3 = 3/10$, $p_4 = 2/10$, $p_5 = 4/10$.

4.3 Selecția de tip turneu

Se bazează pe simularea unei competiții între elementele populației. Pentru a decide ce element să se selecteze la fiecare etapă se extrag uniform aleator (cu sau fără revenire) k elemente din populația curentă iar dintre acestea se alege cel care are cea mai mare valoare pentru funcția de adecvare. Cel mai frecvent se utilizează $k = 2$. Structura generală a algoritmului de selecție de tip turneu (în condițiile în care se dorește selecția a m elemente) este descrisă în fig. 5. Algoritmul returnează un vector cu cei m indici ai elementelor selectate.

4.4 Selecția prin trunchiere

Se folosește atunci când dintr-o populație (sau o reuniune de populații) trebuie selectată o subpopulație de q elemente, în manieră deterministă. Presupune ordonarea descrescătoare după valoarea funcției de adecvare a tuturor elementelor și reținerea primelor q . Metoda este foarte simplă însă prezintă dezavantajul că necesită sortarea elementelor populației, prelucrare ce determină creșterea complexității prelucrării.

5 Încrucișare

Încrucișarea permite combinarea informațiilor provenite de la doi sau mai mulți părinți pentru generarea unuia sau mai multor urmași. Vom considera doar cazul a doi părinți (notați cu x și y) care generează doi urmași (notați cu x' și y'). Încrucișarea (numită uneori recombinare) depinde de modul de codificare a datelor aplicându-se direct asupra cromozomilor.

Variante specifice *codificării binare* (cromozomul este o succesiune de n cifre binare):

Încrucișare cu un punct de tăietură. Se alege (aleator) un $k \in \{1, \dots, n - 1\}$ numit punct de tăietură (încrucișare) și se construiesc urmașii în modul următor:

$$x' = (x_1, \dots, x_k, y_{k+1}, \dots, y_n) \quad \text{și} \quad y' = (y_1, \dots, y_k, x_{k+1}, \dots, x_n).$$

Este ilustrată în figura 6.

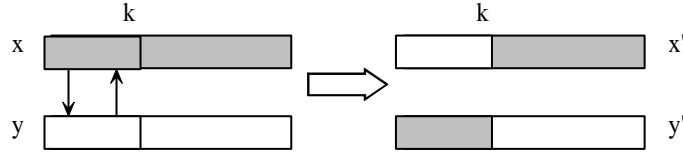


Figura 6: Încrucișarea cu un punct de tăietură

Încrucișare cu două puncte de tăietură. Se aleg (aleator) $k_1, k_2 \in \{1, \dots, n-1\}$ ($1 \leq k_1 < k_2 < n$) numite puncte de tăietură și se construiesc urmașii în modul următor:

$$x' = (x_1, \dots, x_{k_1}, y_{k_1+1}, \dots, y_{k_2}, x_{k_2+1}, \dots, x_n) \quad \text{și} \quad y' = (y_1, \dots, y_{k_1}, x_{k_1+1}, \dots, x_{k_2}, y_{k_2+1}, \dots, y_n).$$

Poate fi extinsă la mai multe puncte de tăietură.

Încrucișare uniformă. La construirea fiecărui urmaș se selectează cu probabilitatea p o genă din primul părinte și cu probabilitatea $1-p$ o genă din al doilea părinte:

$$x'_i = \begin{cases} x_i & \text{cu probabilitatea } p \\ y_i & \text{cu probabilitatea } 1-p \end{cases} \quad \text{iar } y'_i = \begin{cases} x_i & \text{dacă } x'_i = y_i \\ y_i & \text{dacă } x'_i = x_i \end{cases}$$

Cazul cel mai natural este cel în care $p = 0.5$.

O variantă specifică codificării reale este:

Încrucișare convexă. Dacă părinții $x = (x_1, x_2, \dots, x_n)$ și $y = (y_1, y_2, \dots, y_n)$ sunt vectori cu elemente reale atunci elementele urmașilor pot fi calculate prin:

$$x'_i = ax_i + (1-a)y_i \quad \text{și} \quad y'_i = ay_i + (1-a)x_i, \quad a \in (0, 1), i = \overline{1, n}$$

Încrucișare specifică codificării de tip permutare. În cazul codificărilor specifice și operatorii de încrucișare și mutație sunt specifici fiind de regulă bazați pe transformări cu caracter euristic sau pe transformări întâlnite în alte tehnici de rezolvare a problemei. De exemplu în cazul codificării de tip permutare trebuie ca încrucișarea să conserve specificul codificării. Să considerăm problema comis voiajorului cu 7 orașe (identificate prin A, B, C, D, E, F și G) și două configurații cu rol de părinți: (A, B, C, D, E, F, G) și (D, C, F, E, A, B, G) . În acest caz încrucișarea cu un punct ($k = 3$) conduce la (D, C, F, A, B, E, G) respectiv (A, B, C, D, F, E, G) . Primul fiu este obținut preluând secvența formată din primele $k = 3$ orașe din al doilea părinte, iar celelalte orașe sunt tot din al doilea părinte însă amplasate în ordinea în care ele apar în primul părinte. În acest fel se asigură faptul că urmașii conțin aceleași elemente însă amplasate în altă ordine decât cea din cadrul părinților.

Nu e necesar ca încrucișarea să se aplice întotdeauna, modul de aplicare putând fi controlat prin intermediul unei probabilități (numită probabilitate de încrucișare și notată cu p_c). Valori uzuale pentru p_c sunt cuprinse între 0.2 și 0.9.

6 Mutație

Asigură alterarea valorii unor gene pentru a evita situațiile în care o anumită alelă nu apare în populație ca urmare a faptului că nu a fost generată de la început. În felul acesta este asigurată

diversitatea populației. Operatorul de mutație depinde de modul de codificare. Întrucât codificarea reală este caracteristică în special strategiilor evolutive și cum în cazul lor mutația este operația principală, operatorii de mutație specifici acestei codificări vor fi prezentați la strategiile evolutive.

În cazul codificării binare cel mai simplu și frecvent operator de mutație este cel în care se selectează aleator un cromozom, în cadrul acestuia se selectează o genă iar valoarea acesteia este modificată (0 devine 1 iar 1 devine 0). În funcție de modul în care se selectează cromozomii și genele există diverse variante.

De exemplu pentru fiecare cromozom se decide cu o anumită probabilitate (p_m , numită probabilitate de mutație) dacă el va fi supus mutației sau nu. În caz afirmativ se selectează (uniform aleator) o genă și valoarea acesteia se modifică. În felul acesta într-un cromozom poate fi modificată o singură genă.

O altă variantă este aceea în care toate genele se consideră ca făcând parte din aceeași structură și pentru fiecare dintre ele se decide dacă va fi modificată sau nu. În felul acesta este posibil ca mai multe gene din cadrul unui cromozom să fie modificate.

În cazul codificării de tip permutare mutația cea mai simplă constă în schimbarea poziției a două elemente. De exemplu în cazul TSP de la configurația (D, C, F, A, B, E, G) se ajunge la configurația (D, C, E, A, B, F, G) prin interschimbarea elementelor de pe pozițiile 3 și 6.

Probabilitatea de mutație, p_m , se alege mică (de ordinul 10^{-3}), întrucât la algoritmi genetici mutația este un operator secundar, cel principal fiind cel de încrucișare.

Referințe

- [1] D. Dumitrescu; Algoritmi genetici și strategii evolutive - aplicații în inteligența artificială și în domenii conexe, Ed. Albastră, 2000.
- [2] P. Flondor, C. Ionescu (ed.); Introducere în algoritmi genetici, Ed. All, 1999.
- [3] S. Khuri, T. Bäck, J. Heitkötter; An Evolutionary Approach to Combinatorial Optimization Problems, Proc. of CSC4, 1994.
- [4] M. Mitchell; An Introduction to Genetic Algorithms, MIT Press, 1996.
- [5] Back T, Fogel D.B, Michalewicz Z.; Evolutionary Computation I. Basic Algorithms and Operators, IOP Publ., 2000 .