Acta et Commentationes, Sciences of Education, nr. 3(33), 2023 ISSN 1857-0623
p. 15-30 E-ISSN 2587-3636

CZU: 004.9:575 DOI: 10.36120/2587-3636.v33i3.15-30

IMPLEMENTAREA ALGORITMILOR GENETICI. ABORDĂRI PRACTICE

Liubomir CHIRIAC, dr. hab., prof. univ.
https://orcid.org/0000-0002-5786-5828
Eugenia CHIRIAC, dr., conf. univ.
https://orcid.org/0000-0002-5935-0414
Natalia LUPAȘCO, dr., conf. univ.
https://orcid.org/0000-0002-3854-2521
Maria PAVEL, dr., conf. univ.
https://orcid.org/0000-0003-4803-6398
Universitatea de Stat din Tiraspol / Universitatea Pedagogică de Stat „Ion Creangă” din Chișinău

Rezumat. În lucrare, autorii analizează principalele concepte legate de funcționarea Algoritmului Genetic în scopul găsirii unei soluții optime, exprimate prin intermediul funcției de fitness. Prin analogie cu dimensiunea evoluționistă a teoriei lui Darwin ce presupune generarea populațiilor (generații) de cromozomi, algoritmul genetic presupune selectarea celor mai „eficienți”, mai „adaptabili” cromozomi din generația curentă. Astfel, cromozomii noi sunt formați aplicând unul din cei trei operatori genetici (ori toți trei)i: *selecția, crossover și mutația.* În acest sens autorii examinează cele mai populare metode care țin de implementarea operatorilor genetici menționați mai sus. Este examinată o aplicație concretă a Algoritmului Genetic și totodată sunt cercetate condițiile de stopare ale algoritmului respectiv.
Cuvinte cheie: algoritm genetic**,** cromozomi, operatori genetici, selecție, crossover, mutație.

IMPLEMENTATION OF GENETIC ALGORITHMS. PRACTICAL APPROACHES

Abstract. In the paper, the authors analyze the main concepts related to the operation of the Genetic Algorithm in order to find an optimal solution, expressed through the fitness function. By analogy with the evolutionary dimension of Darwin's theory, which involves the generation of populations (generations) of chromosomes, the genetic algorithm involves the selection of the most "efficient", most "adaptable" chromosomes from the current generation. Thus, new chromosomes are formed by applying one of the three genetic operators (or all three): selection, crossover and mutation. In this sense the authors examine the most popular methods related to the implementation of the genetic operators mentioned above. A concrete application of the Genetic Algorithm is examined and at the same time the stopping conditions of the respective algorithm are investigated.
Keywords: genetic algorithm, chromosomes, genetic operators, selection, crossover, mutation.

1. Algoritmul genetic și Teoria Evoluției

Algoritmii genetici sunt inspirați din Teoria Evoluției, dezvoltată de Charles Darwin, care se bazează pe principiul că „specia care se poate adapta cel mai bine la schimbare – supraviețuiește”. Mecanismul de funcționare a algoritmului genetic este similar evoluției naturale. Tot din Teoria Evoluției rezultă că supraviețuirea speciei poate fi menținută prin procesul de selecție, reproducere, încrucișare și mutație. În felul acesta conceptul de

evoluție al lui Darwin este adaptat la funcționarea algoritmului genetic pentru a găsi soluții la o problemă exprimată prin intermediul funcției de fitness (funcție obiectivă ori funcție de adaptare).

O soluție generată de un algoritm genetic se numește *cromozom*, iar un set de cromozomi este numit *populație*. Un cromozom, după cum se știe, este compus din gene iar valoarea sa poate fi fie numerică, binară, simboluri sau caractere, în dependență de problema care se rezolvă.

Populația examinată, alcătuită din cromozomi, va fi supusă la fiecare ciclu unei evaluări. Acest fapt presupune calcularea, prin intermediul funcției de fitness, fitness-ul fiecărui cromozom. Astfel, se determină „posibilele soluții" ori „cât de aproape sunt soluțiile identificate în cadrul populației respective" în raport cu soluția optimă a problemei examinate. În așa mod se clarifică cât de eficienți, „adaptabili" sunt cromozomii din populația examinată. Teoria evoluționistă a lui Darwin presupune generarea următoarei populații (generații) de cromozomi, din care sunt selectați cei mai „eficienți", cei mai „adaptabili" cromozomi din generația curentă, formându-se astfel cromozomii noi aplicând unul din cei trei operatori genetici (ori toți trei)i: ***selecția***, ***crossover*** și ***mutația***.

Astfel, unii cromozomi din populație se vor împerechea printr-un proces numit ***crossover*** (încrucișare), producând noi cromozomi numiți descendenți, a căror compoziție genică este o combinație a părinților. Într-o generație, câțiva cromozomi pot să treacă prin procesul de mutație, ceea ce presupune modificări în locațiunea genelor. De obicei, numărul de cromozomi care vor suferi încrucișări și mutații este controlat de rata de încrucișare și valoarea ratei de mutație. Cromozomul din populația care se va menține pentru următoarea generație va fi selectat pe baza regulii evoluției darwiniene, cromozomul care are o valoare de fitness mai „bună" va avea o probabilitate mai mare de a fi selectat din nou în generația următoare.

După câteva generații, valoarea cromozomilor va converge către o anumită valoare de optim care este cea mai „bună soluție" pentru problema cercetată. În continuare vom examina cei trei operatori genetici importanți: *selecția*, *crossover* și *mutația* [1-10].

2. Selecția

Reamintim că ***selecția*** reprezintă alegerea indivizilor cu cea mai bună aptitudine pentru reproducere (sortarea după valoarea funcției obiective). Cu cât este mai bună fitness-ul unui individ, cu atât sunt mai mari șansele sale de a se încrucișa și de a-și moșteni genele de către generația următoare. Așa dar, selecția reprezintă modul în care anumiți indivizi ies din populația examinată și trec în altă populație (generație). Alegerea unui procedeu de ieșire din populație ține de problema examinată. În așa mod, este firesc, ca pentru o problema de optimizare, clasificarea populației să se facă în funcție de valorile

corespunzătoare ale cromozomilor. Mai jos vom examina unele metode care țin de procesul de selecție:

- selecția cu ajutorul ruletei ponderate;
- selecția după rang;
- selecția după eșantionarea universală stocastică;
- selecția în conformitate cu scalarea liniară a funcției fitness;
- selecția turneu;
- elitismul.

2.1. *Selecția cu ajutorul ruletei ponderate (Fitness proportionate selection (FPS))*

Una dintre cele mai răspândite modalități de selecție a părinților este selecția proporțională de fitness. Din această perspectivă, fiecare individ poate deveni părinte cu o probabilitate proporțională cu propriu fitness. Din aceste considerente, indivizii mai apți, mai în formă, au șanse mai mari de a se împerechea și de a-și promova propriile trăsături la generația următoare. Strategia respectivă de selecție pune presiune pe indivizii mai apți din populația respectivă, favorizându-i în așa fel să evolueze în timp.

Cum vom interpreta matematic acest fenomen biologic? Considerăm un cerc împărțit în *n* sectoare (porțiuni), unde *n* reprezintă numărul de indivizi din populație. Astfel, fiecărui individ i se atribuie un sector din cerc care este proporțional cu valoarea sa de fitness. Deci, fiecărui cromozom i se va pune în corespondență un sector al ruletei, care este direct proporțional, ca mărime, cu fitness-ul cromozomului respectiv. Prin urmare cromozomilor cu fitness mai mare li se vor atribui sectoare mai mari, iar celor cu fitness mic li se vor pune în corespondență sectoare mai mici. Evident, atunci când se va arunca aleatoriu o bila pe ruletă există probabilitate mai mare de alegere pentru cromozomii cu fitness mare. Simularea ruletei se realizează în felul următor:

- se calculează suma tuturor fitness-urilor cromozomilor. Fie suma este S;
- se generează un număr aleatoriu între 1 și S. Fie numărul aleatoriu este r;
- se parcurge populația ordonată crescător după fitness și se calculează suma fitness-urilor cromozomilor parcurși până se depășește valoarea r.
- se alege cromozomul la care s-a ajuns.

Exemplu 1. Examinăm tabelul de mai jos care conține informația pentru 6 cromozomi.

Cromozomii	Funcția de fitness	Proporția	N
A	10	9%	3
B	14	12,7%	4
C	30	27.2%	5
D	9	8%	2
E	40	36%	6
F	7	6.6%	1
Total	S=110	100%	21

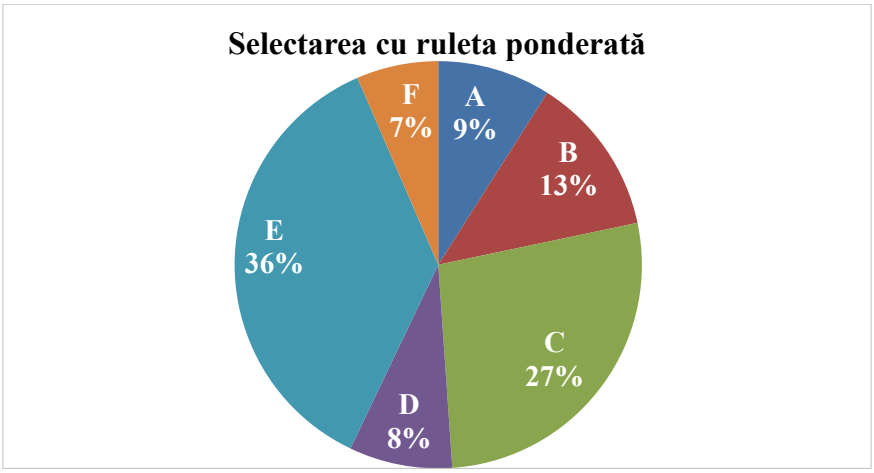

Figura 1. Metoda ruletei ponderate

Se aruncă o bilă pe ruleta și se selectează cromozomul unde se oprește. În mod clar, cromozomii cu valoare de fitness mai mare vor fi selectați de mai multe ori. Aruncând, de exemplu, bila de 6 ori, aleatoriu se obțin următoarele sectoare, deci și cromozomii care corespund sectoarelor respective: A, C, C, E, B, E.

2.2. *Selecția după rang (Rank Selection)*

În cazul în care fitness-ul cromozomilor examinați diferă deosebit de mult, selecția cu ajutorul ruletei ponderate nu este foarte eficientă și în contextul dat nu este recomandată pentru a fi aplicată. Astfel, de exemplu, daca suma fitness-urilor unor cromozomi examinați este de 150 de ori mai mic comparativ cu fitness-ul unui cromozom fixat atunci, evident, probabilitatea cromozomilor cu fitness mic este aproape de zero. Pentru a ***exclude*** această situație, se procedează în felul următor:

- cromozomii se ordonează crescător în funcție de fitness;
- ulterior cromozomii se renumerotează cu numere întregi din intervalul [1, .., n], unde n este dimensiunea populației, iar cromozomul cu fitness-ul cel mai mic are numărul 1 etc., iar cromozomul cu fitness-ul cel mai mare va avea numărul egal cu dimensiunea populației;
- numerele respective vor fi considerate fitness-uri și în situația dată se aplică selecția metodei ruletei ponderate.

Exemplu 2. Examinăm tabelul de mai jos care conține informația pentru 6 cromozomi.

Cromozomii	Funcția de fitness	Rangul	Proporția rangului
A	10	3	14.3%
B	14	4	19%
C	30	5	23.8%
D	9	2	9.5%
E	40	6	28.6%
F	7	1	4.8%
Total	S=110	21	100%

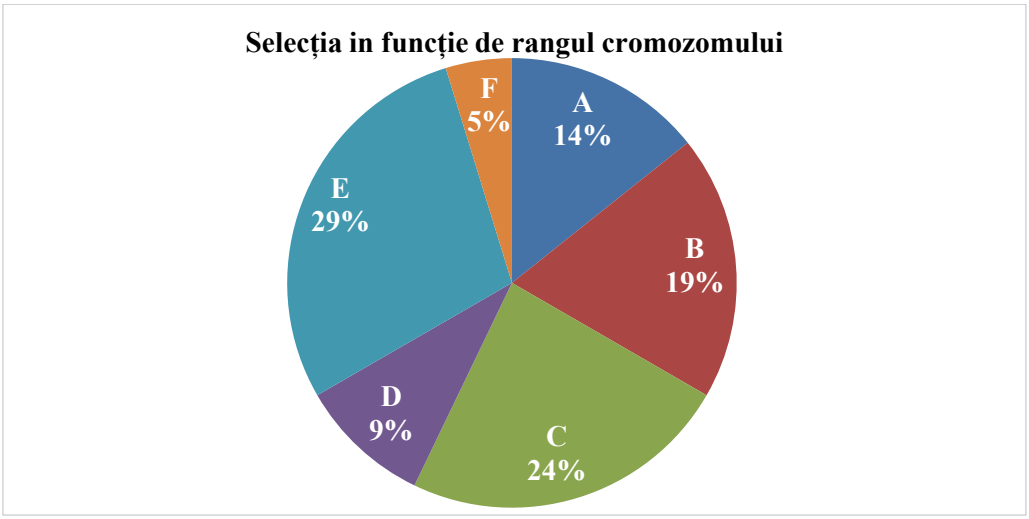

Figura 2. Selecția după rang

Și în această situație sesizăm că cromozomii cu fitness-ul cel mai mare au la fel cele mai mari șanse de a fi selectați, dar de data aceasta nu mai sunt așa mari în comparație cu șansele celorlalți. Aruncând, de exemplu, bila de 6 ori, aleatoriu se obțin următoarele sectoare, deci și cromozomii care corespund sectoarelor respective: D, C, B, E, A, C.

Așadar, dacă o populație inițială conține unul sau doi indivizi foarte apți, dar nu cei mai buni din anumite considerente și restul indivizilor populației nu sunt atât de apți, atunci cei mai apți vor domina rapid întreaga populație și vor împiedica populația să exploreze alți indivizi potențial mai buni. O dominație atât de puternică provoacă o pierdere foarte mare a diversității genetice, ceea ce cu siguranță nu este avantajos pentru procesul de optimizare. În acest sens:

1. Selectarea rangului clasează mai întâi populația și apoi fiecare cromozom primește fitness din acest clasament.
2. Cel mai puțin apt cromozom va avea fitness 1, al doilea, cel mai rău - 2 etc. Iar cel mai apt va avea fitness *n* (numărul de cromozomi din populație).
3. După aceasta, toți cromozomii au șanse mai mari de a fi selectați.
4. Schemele de selecție bazate pe rang pot evita convergența prematură.
5. Dar poate fi costisitor din punct de vedere computațional, deoarece sortează populațiile în funcție de valoarea de fitness.
6. Dar această metodă poate duce la o convergență mai lentă, deoarece cei mai buni cromozomi nu diferă atât de mult de alții.

2.3. *Selecția după eșantionare universală stocastică (Stochastic Universal Sampling (SUS))*

Eșantionarea universală stocastică este destul de similară cu ruleta. Cu toate acestea, în loc să avem un singur punct fix, avem mai multe puncte fixe, așa cum se arată în figura 3. Prin urmare, toți părinții sunt aleși într-o singură rotire a roții. De asemenea, o astfel de configurație „încurajează” și alți cromozomi să fie aleși cel puțin o dată.

Exemplu 3. La o singură rotire a ruletei sunt selectați 6 părinți.

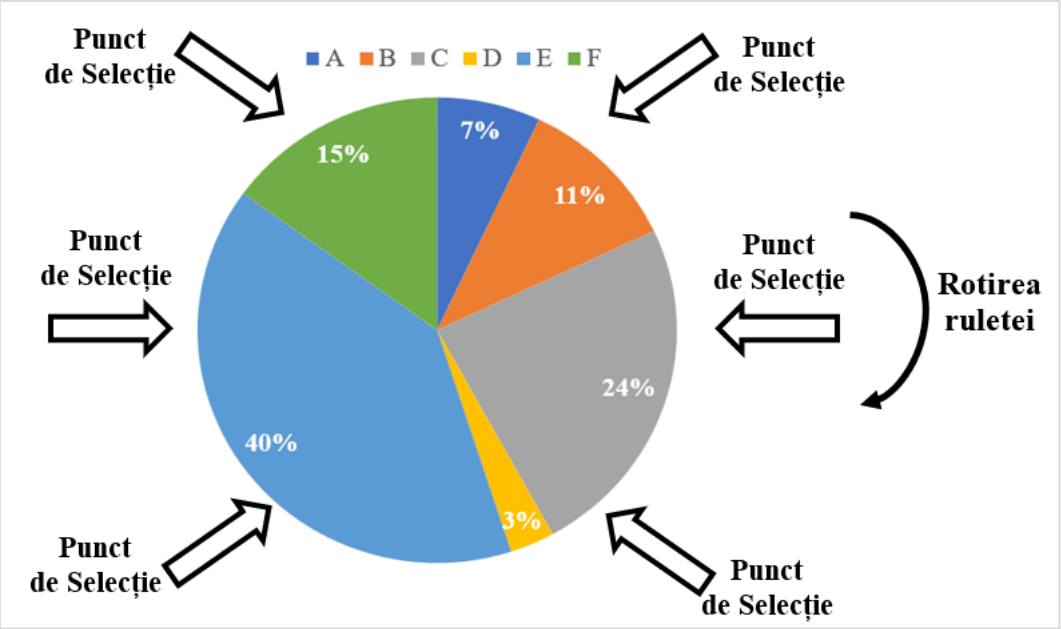

Figura 3. Selecția după eșantionare

2.4. *Selecția în conformitate cu scalarea liniară a funcției fitness (Linear Fitness Scaling in Genetic Algorithm)*

Așa cum s-a menționat anterior, un aspect important al algoritmilor genetici este funcția de fitness, care cuantifică calitatea fiecărei soluții candidate. Funcția de fitness atribuie o valoare numerică, numită și scor de fitness sau valoare de fitness, fiecărui individ din populație.

În unele cazuri, se dorește să se implementeze scalarea liniară a fitness-ului fiecărui cromozom în algoritmul examinat. Scalarea liniară de fitness este o tehnică care urmărește îmbunătățirea presiunii de selecție în Algoritmul Genetic prin scalarea liniară a valorilor de fitness.

Într-un algoritm genetic tipic, valorile funcției de fitness ale indivizilor din populație variază de la zero la o anumită valoare pozitivă. Scalarea liniară a fitness-ului urmărește să transforme aceste valori de fitness într-un nou interval (între zero și unu ori între n_1 și n_2, unde n_1 și n_2 sunt numere reale), într-o manieră liniară. În așa fel toate valorile fitness-ului conținute pe un anumit interval trec în alte valori ale fitness-ului care se conțin pe alt interval.

Motivația scalării liniare a funcției de fitness ține de creșterea presiunii de selecție în Algoritmul Genetic. Prin scalarea liniară a valorilor de fitness, indivizilor cu scoruri de fitness mai mari li se oferă un avantaj mai mare în timpul procesului de selecție. Acest lucru poate ajuta în cazurile în care valorile de fitness nu sunt bine distribuite la nivelul populației sau când este nevoie să se efectueze o ierarhizare a soluțiilor cu fitness mai mare.

Exemplu 4. Examinăm tabelul de mai jos care conține informația pentru 6 cromozomi și valorile funcției de fitness ***f***:

Cromozomi		Funcția de fitness f	Funcția de fitness f^*
A	1	10	Nu se cunoaște
B	2	14	Nu se cunoaște
C	3	30	Nu se cunoaște
D	4	9	Nu se cunoaște
E	5	40	130
F	6	7	80.5
Total		S=110	

Este necesar să se efectueze scalarea liniară a funcției de fitness pentru ca să transforme, să „treacă" valorile funcției de fitness 10, 14, 30, 9, 40, 7 pe un nou interval [80.5, 130], unde vom nota: $d_{min} = 80.5$ și $d_{max} = 130$.

Pentru a efectua operația respectivă vom utiliza relația $f^*(x_i) = a \cdot f(x_i) + b$, unde $i = 1,2,3,4,5,6$ și relațiile: $\mathrm{a} \cdot f_{min} + \mathrm{b} = d_{min}$; $\mathrm{a} \cdot f_{max} + \mathrm{b} = d_{max}$, pentru care vom avea următorul sistem:

$$\begin{cases} \mathrm{a} \cdot f_{min} + \mathrm{b} = d_{min} \\ \mathrm{a} \cdot f_{max} + \mathrm{b} = d_{max} \end{cases} (1)$$

În cazul problemei noastre, avem $f_{min} = 7$ și $d_{min} = 80.5$ iar $f_{max} = 40$ și $d_{max} = 130$. Substituind datele respective în sistemul (3) și rezolvând sistemul dat obținem: $a = 1.5$ iar $b = 70$. În continuare calculăm noile valori $f^*(x_i) = a \cdot f(x_i) + b$. Astfel: $f_1^* = 1.5 \cdot 10 + 70 = 85$; $f_2^* = 1.5 \cdot 14 + 70 = 91$; $f_3^* = 1.5 \cdot 30 + 70 = 115$; $f_4^* = 1.5 \cdot 9 + 70 = 83.5$; f_5^* și f_6^* sunt deja calculate.

Cromozomii		Funcția de fitness f	Funcția de fitness f^*	Proporția nouă f^*	Proporția veche f
A	1	10	85	14.5%	9%
B	2	14	91	15.6 %	12,7%
C	3	30	115	19.7%	27.2%
D	4	9	83.5	14.3%	8%
E	5	40	130	22.2%	36%
F	6	7	80.5	13.7%	6.6%
Total		S=110	585	100%	100%

Observăm că după efectuarea scalării liniare $f^* = 1.5 \cdot f + 70$ diagrama a devenit mai uniformă. În acest caz probabilitatea de a fi selectați cromozomii mai puțin apți este mai mare. Iar pe de altă parte cromozomii mai apți nu vor fi dominanți în noua populație. În așa fel valorile de fitness sunt distribuite uniform la nivelul populației. Iar algoritmul genetic va converge mai sigur către soluția optimă căutată. În diagramele de mai jos se

poate vedea clar diferența de rezultate dintre selectarea cu ruleta ponderată și selectarea liniar scalară.

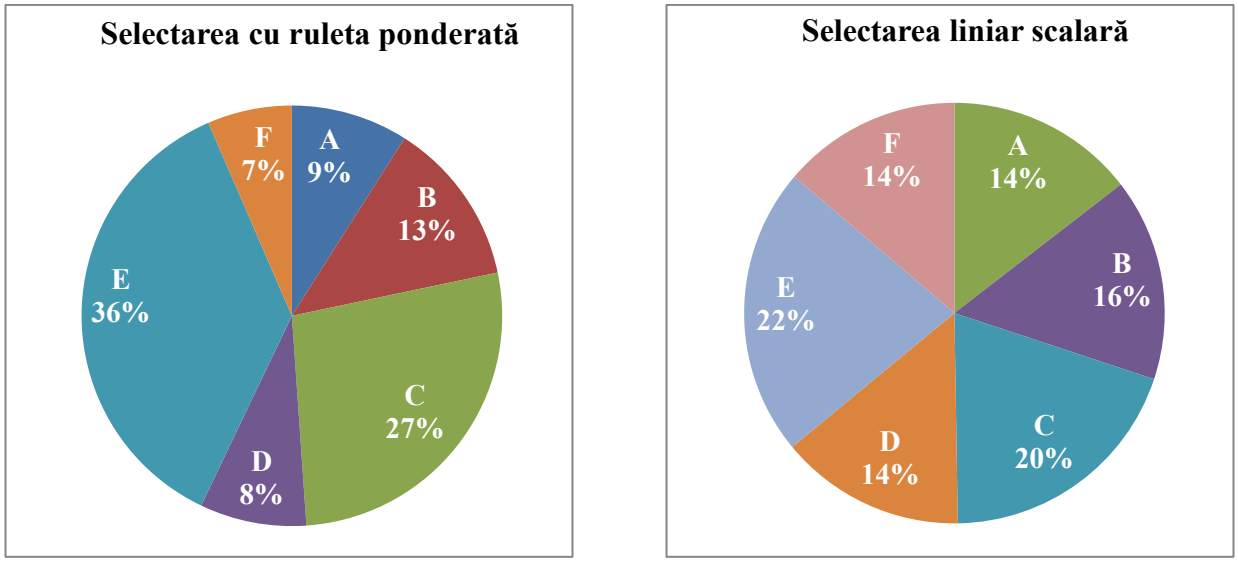

Figura 4. Analiza comparativă dintre 2 metode de selecție

2.5. *Selecția turneu (Tournament Selection)*

Prin intermediul **selecției turneu** se alege în mod aleatoriu *k* indivizi din populație iar dintre aceștia doar cei mai buni *j* sunt selectați pentru a deveni părinți. Aceeași procedură se repetă pentru selectarea următorului părinte. Procedura se repetă până se obține numărul dorit de indivizi. Este cea mai eficientă din punct de vedere al complexității timp. Din punct de vedere al presiunii de selecție se aseamănă selecției bazată pe rang. Selecția turneului este, de asemenea, extrem de populară, deoarece poate funcționa chiar și cu valori negative de fitness.

Exemplu 5. Prin intermediul selecției turneu sunt selectați inițial în mod aleatoriu 3 cromozomi, iar ulterior, la etapa finală, cel mai bun.

Cromozomii	A	B	C	D	E	F	G
Valoarea de fitness	15	10	7	6	9	14	8

Dimensiunea turneului este egală cu 3. În mod aleatoriu sunt selectați 3 cromozomi.

Cromozomii	B	D	G
Valoarea de fitness	10	6	8

Cromozomul cu cea mai bună valoare de fitness este B, deci cromozomul învingător este B.

Cromozomul învingător	B
Valoarea de fitness	10

2.6. *Elitism*

În cazul când se constituie o nouă populație prin mutație și încrucișare, persistă probabilitatea de a pierde cei mai buni cromozomi. Pentru a evita această situație, se recomandă trecerea directă, copierea, fără a fi modificați, a celor mai buni cromozomi din

vechea populație în noua populație. Procesul dat se numește *elitism* și el contribuie semnificativ la creșterea performanțelor algoritmilor genetici. Elitismul, a fost introdus de Ken DeJong (1975) și reprezintă o schemă adițională oricărui mecanism de selecție al cărei scop este de a reține cei mai buni *k* indivizi la fiecare generație. Astfel de indivizi ar putea dispărea din cauză că nu au fost selectați pentru supraviețuire sau fiindcă au fost distruși prin aplicarea operatorilor. Studiile experimentale au dovedit că de multe ori elitismul îmbunătățește semnificativ performanța algoritmului genetic.

Exemplu 6. Prin intermediul selecției de tip elitism sunt selectați cei mai buni n=3 cromozomi.

Cromozomii	A	B	C	D	E	F	G
Valoarea de fitness	15	10	7	6	9	14	8

Dimensiunea selecției de tip elitism este n = 3. În așa mod:

Cromozomii	A	F	B
Valoarea de fitness	15	14	10

3. Încrucișarea (crossover)

Operatorul de încrucișare este analog cu reproducerea și încrucișarea biologică și, de obicei, se aplică asupra indivizilor din populația intermediară. Sunt selectați doi indivizi din populația intermediară și anumite porțiuni din cei doi cromozomi sunt interschimbate. Astfel, prin intermediul operatorului de încrucișare se produc unul sau mai mulți descendenți folosind materialul genetic al părinților (cromozomilor). În felul acesta, operatorul de încrucișare imită încrucișarea inter-cromozomială naturală. De regulă, încrucișarea nu generează descendenți aleatorii și încrucișarea depinde de tipul de codificare a cromozomilor și de problema particulară care este în proces de rezolvare. În continuare examinăm următoarele procedee de încrucișare:

- încrucișare într-un singur punct;
- încrucișare în două puncte;
- încrucișare în *k*-puncte;
- încrucișare uniformă;
- încrucișare ordonată;
- recombinare aritmetică totală;
- încrucișare mixtă.

3.1. *Încrucișare într-un singur punct (One Point Crossover)*

Se alege un punct de încrucișare. De la primul părinte este copiată secvența de la început până la punctul de încrucișare, iar din al doilea părinte este copiată secvența de la punctul de încrucișare până la final.

Exemplu 7. Punctul de tăietură este pe poziția a patra. Urmașul 1 moștenește secvențele 1,0,0,1 de la Părintele 1 și secvențele 1,1,0 de la Părintele 2. Urmașul 2 moștenește secvențele 1,1,0,0 de la Părintele 2 și secvențele 1,0,1 de la Părintele 1.

Părinte 1	1	0	0	1	1	0	1

Urmaș 1	1	0	0	1	1	1	0

Părinte 2	1	1	0	0	1	1	0

Urmaș 2	1	1	0	0	1	0	1

3.2. *Încrucișare în două puncte (Two Points Crossover)*

Sunt alese două puncte de încrucișare. Secvența dintre cele două puncte este aleasă dintr-unul din părinți, iar ce a rămas din celălalt părinte.

Exemplu 8. Considerăm că punctele de tăietură sunt pe poziția a doua și a cincea. Urmașul 1 moștenește secvențele 1,0 și 0,1 de la Părintele 1 și secvențele 0,0,1 de la Părintele 2. Urmașul 2 moștenește secvențele 1,1 și 1,0 de la Părintele 2 și secvențele 0,1,1 de la Părintele 1.

Părinte 1	1	0	0	1	1	0	1

Urmaș 1	1	0	0	0	1	0	1

Părinte 2	1	1	0	0	1	1	0

Urmaș 2	1	1	0	1	1	1	0

3.3. *Încrucișare în k-puncte (Multi Point Crossover)*

Sunt alese *k*- puncte de încrucișare. Secvențele dintre cele *k*- puncte sunt alese în baza principiilor enunțate mai sus.

Exemplu 9. Considerăm că punctele de tăietură sunt pe poziția a doua, a patra și a șasea. Urmașul 1 moștenește secvențele 1,0 și 1,0 de la Părintele 1 și secvențele 1,0, și 0 de la Părintele 2. Urmașul 2 moștenește secvențele 1,1 și 1,1 de la Părintele 2 și secvențele 0,1 și 1 de la Părintele 1.

Părinte 1	1	0	0	1	1	0	1

Urmaș 1	1	0	1	0	1	0	0

Părinte 2	1	1	1	0	1	1	0

Urmaș 2	1	1	0	1	1	1	1

3.4. *Încrucișare uniformă*

În procesul încrucișării uniforme, nu se împarte cromozomul în segmente, ci mai degrabă se tratează fiecare genă separat. În acest sens, în mod aleatoriu pentru fiecare cromozom se decide separat dacă va fi inclus sau nu în cromozomul descendent. De asemenea, se permite de realizat încrucișarea foarte calculat, cu intenție clară și în cazul când este necesar de moștenit mai mult material genetic de la un părinte.

Exemplu 10. Punctele de tăietură se iau aleatoriu ori foarte calculat, cu o intenție clară. Urmașul 1 moștenește secvența 1,3 și secvențele 5 și 6 de la Părintele 1 și secvențele 5,7 și 3 de la Părintele 2. Urmașul 2 moștenește secvența 2,1 și secvențele 4 și 3 de la Părintele 2 și secvențele 4,1 și 1 de la Părintele 1.

Părinte 1	1	3	4	2	5	1	6

Urmaș 1	1	3	5	7	5	3	6

Părinte 2	2	1	5	7	4	3	3

Urmaș 2	2	1	4	2	4	1	3

3.5. *Încrucișare ordonată*

În procesul încrucișării ordonate se ține cont de faptul că toate cifrele care reprezintă Părinții cromozomi trebuie să fie diferite. De exemplu, în cazul examinării problemei comis voiajorului toate orașele, numerotate de la 1 la n, vor fi parcurse doar o singură dată. Deci, toate sunt diferite. În acest caz Urmașii la fel trebuie să posede toate numerele respective dar, în același timp, este necesar ca ele să fie diferite.

Exemplu 11. Punctele de tăietură se iau în funcție de scopul propus. Urmașul 1 moștenește secvența 1, 3 și secvențele 2 și 6 de la Părintele 1, iar secvențele 5, 7, 4 de la Părintele 2. Urmașul 2 moștenește secvențele 7 și 1, cât și secvențele 6 și 3 de la Părintele 2, iar secvențele 4, 2 ,5 de la Părintele 1.

Părinte 1	1	3	4	2	5	7	6

Urmaș 1	1	3	5	7	4	2	6

Părinte 2	2	1	5	7	4	6	3

Urmaș 2	7	1	4	2	5	6	3

3.6. *Recombinare aritmetică totală (Whole Arithmetic Recombination)*

Această metodă este folosită pentru cromozomii care sunt reprezentați prin intermediul numerelor reale. În cazul dat se ia media ponderată a celor doi părinți în baza următoarelor formule:

- $Urmaş\ 1 = \alpha \cdot x + (1 - \alpha) \cdot y$, unde *x* aparține Părinte 1, iar *y* aparține Părinte 2.
- $Urmaş\ 2 = \ \alpha \cdot x + (1 - \alpha) \cdot y$, unde *x* aparține Părinte 2, iar *y* aparține Părinte 1.

Coeficientul $\alpha = 0, \ \dots, \ 1$.

Exemplu 12. Evident, dacă $\alpha = 0.5$, atunci ambii urmași vor fi identici, așa cum se arată în diagramele de mai jos.

P1	0.5	0.5	0.4	0.4	0.3	0.3

U1	0.35	0.2	0.3	0.35	0.25	0.3

P2	0.2	0.3	0.2	0.3	0.2	0.3

U2	0.35	0.2	0.3	0.35	0.25	0.3

3.7. *Încrucișare mixtă (Blend Crossover)*

Luând în considerare valoarea cromozomilor pentru Părintele 1 și Părintele 2, aplicând încrucișarea mixtă, obținem valorile pentru Urmașii 1 și 2, care se iau aleator din intervalul:

$$[x_1 \ - \alpha(x_2 - x_1), x_2 + \alpha(x_2 - x_1)] \ (2),$$

unde $x_1 < x_2$, și x_1 ține de Părintele 1, iar x_2 ține de Părintele 2. În caz că nu se respectă condiția respectivă se va lua $x_1 < x_2$, unde x_1 ține de Părintele 2, iar x_2 ține de Părintele 1. Coeficientul $\alpha = 0, \ \dots, \ 1$. Se recomandă ca $\alpha = 0.5$.

Exemplu 13. Dacă $\alpha = 0.5$, atunci determinând intervalele (2), pentru fiecare pereche $x_1 < x_2$, aleatoriu alegem valorile pentru Urmașii 1 și 2, așa cum se vede în diagramele de mai jos.

P 1	-2.7	4.3	1.2	-0.1	1.5

U 1	-5.3	3.2	-0.7	0.9	2.7

P2	6.8	-5.4	-2.2	1.3	3.5

U 2	-2.9	- 0.3	-1.3	0.7	2.1

Metodele de încrucișare prezentate mai sus sunt destul de generale. De obicei sunt alese metode specifice problemei care se dorește a fi rezolvată și care generează o populație mai bună.

4. Mutație (Mutation)

În termeni simpli, mutația poate fi definită ca o mică modificare aleatorie a cromozomului, pentru a obține o nouă soluție. Mutația este utilizată pentru menținerea și introducerea diversității în populația genetică și se aplică de obicei cu o probabilitate scăzută. Mutația este parte din Algoritmul Genetic care este legată de „explorarea" spațiului de căutare. Din practică, s-a observat că mutația este esențială pentru convergența Algoritmului Genetic. Mutația se utilizează în principal pentru a evita căderea soluțiilor într-un optim local.

În această secțiune, descriem unii dintre cei mai des utilizați operatori de mutație. La fel ca operatorii de încrucișare, aceasta nu este o listă exhaustivă, iar implementatorii Algoritmului Genetic ar putea găsi o combinație a abordărilor examinate ori un operator de mutație specific problemei mai util. Mai jos vom examina câțiva operatori care țin de mutație mai des utilizați:

- mutația de inversare a biților;
- mutația Swap;
- mutația Scramble;
- mutația inversă.

4.1. *Mutația de inversare a biților* *(Bit Flip Mutation)*

Acest operator realizează o inversare a biților în cadrul cromozomilor examinați. Astfel, sunt selectați unul sau mai mulți biți aleatorii și ulterior sunt inversați, în loc de 0 se scrie 1, și în loc de 1 se scrie 0.

Cromozom	1	0	0	1	1	0

Mutația	1	0	1	0	0	0

4.2. *Mutația Swap* *(Swap Mutation)*

În mutația de schimbare (Swap), sunt selectate aleatoriu două poziții pe cromozomi și se schimbă valorile între ele. Tipul respectiv de mutație se aplică în codificările bazate pe permutare.

Cromozom	1	5	6	1	2	3

Mutația	1	3	6	1	2	5

4.3. *Mutația Scramble* *(Scramble Mutation)*

Mutația Scramble este, de asemenea, populară cu reprezentările de permutare. În cromozomul examinat se alege o submulțime de gene și valorile lor sunt amestecate după dorință ori aleatoriu.

Cromozom	1	5	6	1	2	3

Mutația	1	6	5	2	1	3

4.4. *Mutația inversă* *(Inversion Mutation)*

Operatorul de mutație inversă acționează în felul următor: se alege o secvență de gene și ulterior submulțimea respectivă este pur și simplu inversată în cromozomul examinat.

Cromozom	1	5	6	1	2	3

Mutația	1	2	1	6	5	3

În funcție de problema examinată, luând în considerare aspectele specifice, pe parcursul realizării Algoritmului Genetic, implementatorii ar putea aplica și alți operatori care țin de mutație (unii chiar inventați) pentru a asigura convergența soluției către valoarea optimă.

5. Când se stopează în realizare Algoritmul Genetic?

Condiția de stopare a unui algoritm genetic este importantă pentru a determina când se va termina de rulat Algoritmul Genetic. Evident, se dorește o condiție de stopare astfel încât la finele rulării soluția obținută să fie aproape de cea optimă.

În linii mari, Algoritmul Genetic se stopează, atunci când se îndeplinește una dintre următoarele condiții de finalizare:

- În situația când nu există nici o îmbunătățire a populației pentru k – iterații executate.
- În situația când se ajunge la un număr absolut de generații.
- În situația când valoarea funcției de fitness a atins o anumită valoare predefinită.

Pe parcursul realizării Algoritmului Genetic, de obicei, există un contor care contabilizează generațiile create. Totodată, în algoritmul respectiv se ține evidența generațiilor pentru care nu a existat nici o îmbunătățire a populației. De fiecare dată când nu se generează descendenți mai buni decât indivizii din populație se trece la o altă generație ori chiar procesul de implementare poate fi stopat pentru a îmbunătăți anumite aspecte ale mecanismelor de implementare.

Condiția de terminare a unui Algoritm Genetic este foarte specifică problemei, iar implementatorul ar trebui să verifice diferite opțiuni pentru a vedea ce se potrivește cel mai bine în cazul problemei examinate. Algoritmii genetici continuă să fie rulați, de obicei, după ce s-a creat un număr predefinit de populații noi și soluția s-a îmbunătățit de la o populație la alta, dar nu s-a atins scopul propus. Procesul se desfășoară până în situația

când va fi atins obiectivul stabilit. În această situație se afișează cei mai buni cromozomi (cel mai bun cromozom) care reprezintă soluția optimă.

6. Aplicații ale algoritmului genetic

Mai jos vom soluționa o problemă practică, care ține de optimizare, aplicând noțiunile și conceptele de bază examinate anterior.

Problemă. Aplicând Algoritmul Genetic, se cere de găsit maximumul funcției $F(x) = x^2 - 1$ pe intervalul de numere întregi $[0, 15]$.

Soluție. Pentru a codifica cromozomii din populația inițială (generația 0) și a opera cu ea ulterior, vom utiliza scrierea numerelor naturale în baza 2 extinsă pe 4 biți. Astfel, codificăm valorile numerelor naturale *x* de la 0 la 15 după cum urmează:

0 *0000*	1 *0001*	2 *0010*	3 *0011*	4 *0100*	5 *0101*	6 *0110*	7 *0111*
8 *1000*	9 *1001*	10 *1010*	11 *1011*	12 *1100*	13 *1101*	14 *1110*	15 *1111*

Creăm populația inițială din 4 cromozomi, aleși în mod aleatoriu: 1001, 0011, 1100 și 0111. Includem cromozomii aleși în tabelul de mai jos. Funcția de fitness în cazul nostru va fi $F(x) = x^2 - 1$. Este necesar să se determine *max F* pe intervalul indicat. În continuare vor fi realizați următorii pași:

Pasul 1. Fiind constituită populația inițială (Generația 0) prin intermediul funcției fitness, se va măsura fitness-ul fiecărei valori și ulterior se vor selecta cei mai „buni" cromozomi.

	Populația inițială	**Valori zecimale**	$F(x) = x^2 - 1$	**Procente**	**Procente/ medie**	**Aproximare**
1	1001	9	80	24.24	1.14	1
2	0011	3	8	2.42	0.11	0
3	1100	12	143	43.33	2.05	2
4	0111	7	48	30	0.69	1
	Suma		**279**	**100**	**4**	**4**
	Media		69.75	25	1	1

La efectuarea primului ciclu, valoarea cea mai bună a funcției fitness este 143, pentru *x=12*. În continuare, încercăm să obținem o îmbunătățire a populației de cromozomi aplicând operatorii genetici: selecția, crossver-ul și mutația.

Pasul 2. Selecția se va face cu ajutorul metodei *ruleta ponderată*. După efectuarea calculelor, se elimină din populație, cromozomul 2, iar cromozomul 3 va fi multiplicat de 2 ori. Cromozomul 1 și 4, vor fi incluși, de asemenea, în noua generație de cromozomi.

În continuare, supraviețuirea speciei (căutarea soluției) se va face prin intermediul procesului de încrucișare și mutație. Încrucișarea se va face într-un singur punct (one point crossover), poziția căruia se va alege aliator. Mutația de tip SWAP se va produce numai

pentru cromozomul 2, pentru ultimele două elemente, așa cum este arătat în tabelul de mai jos.

	Populația după 1 selecție	**One Point Crossover**	**Populația după crossover**	**Mutație inversată pentru cr.2**	**Populația I generație**
1	1001	Poziția 2	1000	1000	1000
2	1100		1101	1110	1110
3	1100	Poziția 2	1111	1111	1111
4	0111		0100	0100	0100

Pasul 3. Obținând o nouă generație de cromozomi (o nouă populație): 1000, 1101, 1111 și 0100, la fel se va calcula fitness-ul fiecărui cromozom și se vor compara.

	Populația I generație	**Valori x**	**$F(x) = x^2 - 1$**	**Procente**
1	1000	8	63	12.7
2	1110	14	195	39.2
3	1111	15	224	45.1
4	0100	4	15	3.01
	Suma		**497**	**100**
	Media		124.25	25

Cea mai mare valoare (cel mai bun fitness) este obținut pentru cromozomul 1111. Astfel, în două cicluri, au fost calculate 8 valori ale funcției fitness pentru cromozomii obținuți. Dintre ele valoarea 224 reprezintă și maximul funcției $F(x) = x^2 - 1$, pe intervalul de numere întregi [0, 15]. Astfel, prin intermediul acestui exemplu s-a arătat la modul practic cum poate fi implementat Algoritmul Genetic pentru a obține o soluție în procesul de optimizare a unei funcții.

Este cunoscut faptul că o bună parte din algoritmi de optim necesită o cantitate enorm de mare de memorie și timpul de execuție crește exponențial. Pentru excluderea acestui inconveniențe, în practică sunt elaborați așa numiții algoritmi euristici care livrează soluții nu în mod obligatoriu optime, dar „suficient de bune” într-un timp relativ mai scurt și folosind o cantitate de memorie acceptabilă. Elaborarea acestor algoritmi are la bază, de regulă, idei empirice, idei naturale, „raționale”, așa cum s-a procedat în cazul Algoritmului Genetic. Algoritmii euristici sunt preferați în situații când efortul de găsire a unei soluții optime este extrem de mare comparativ cu câștigul obținut și în condițiile când efortul mare de programare nu se justifică.

Este necesar de menționat încă odată că ***Algoritmii evolutivi (Evolutionary Algorithms)*** aparțin domeniului denumit calcul evolutiv, care utilizează mecanisme inspirate de biologie (selecție, crossover, mutație etc.) pentru a căuta soluții optime. Iar ***Algoritmii Genetici (Genetic Algorithms)*** sunt algoritmii cei mai pe larg utilizați dintre

toți algoritmii care țin de Algoritmii evolutivi, fiind algoritmi euristici de căutare care imită procesul natural de selecție pentru a alege soluția candidată „cea mai potrivită”.

Articol realizat în cadrul proiectului de cercetări științifice „Metodologia implementării TIC în procesul de studiere a științelor reale în sistemul de educație din Republica Moldova din perspectiva inter/transdisciplinarității (concept STEAM)”, inclus în „Program de stat” (2020-2023), Prioritatea IV: Provocări societale, cifrul 20.80009.0807.20, cu suportul financiar oferit de Agenția Națională pentru Dezvoltare și Cercetare

Bibliografie

1. HOLLAND, J.H. *Adaptation in Natural and Artificial Systems*. Ann. Arbor: University of Michigan Press, 1975. 183 p.
2. MITCHELL, M. *An Introduction to Genetic Algorithms*. Cambridge: MIT Press, 1996.
3. MITCHELL, M. Genetic Algorithms: An Overview. In: *Complexity*, 1995. Nr. 1(1), pp. 31-39.
4. DUMITRESCU, D. *Algoritmi genetici şi strategii evolutive - Aplicatii în inteligența artificială şi în domenii conexe*. Cluj-Napoca: Editura Albastra, 2000.
5. GOLDBERG, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison -Wesley, Reading, MA,1989.
6. GAREY, M.R.; JOHNSON, D.S. *Computers and Intractability: A Guide to NP-completeness*. New York: W.H. Freeman and Company, 1978.
7. KOZA, J.R. *Genetic Programming*. Cambridge: MIT Press, MA, 1992.
8. OLTEAN, M. *Proiectarea şi implementarea algoritmilor*. Cluj-Napoca: Computer Libris Agora, 2000.
9. BEASLEY, D.; BULL, D. R.; MARTIN, R. R. An Overview of Genetic Algorithms: Part 1. Fundamentals. In: *University Computing*, volume 15(2), pp. 58- 69, 1993.
10. RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. Second Edition. Prentice Hall, 2003.