În cazul problemelor pentru care nu se cunosc algoritmi eficienţi (polinomiali) de rezolvare (din categoria tehnicilor studiate până acum), se pot utiliza alte tipuri de algoritmi, care oferă soluţii aproximative (algoritmi genetici, algoritmi probabilişti).

Algoritmi genetici

- Sunt utilizaţi în probleme pentru care
  - spaţiul de căutare a soluţiilor posibile este mare
  - nu se cunosc algoritmi exacţi mai rapizi
- Furnizează o soluţie aproximativă.
- Denumirea lor se datorează preluării unor mecanisme din biologie: moştenirea genetică şi evoluţia naturală pentru populaţii de indivizi.
- Aplicaţii - probleme de optim, de căutare, de planificare etc (v. [3], [4])

Exemplu – Maximul unei funcţii.

Fie $f:D \rightarrow \mathbf{R}$. Să se calculeze $\max\{ f(x) \mid x \in D \}$, unde $D = [a, b]$.
Vom presupune că f este pozitivă pe D: $f(x) > 0, \ \forall x \in D$. Această presupunere nu este restrictivă (putem aduna o constantă mare, astfel încât f să devină pozitivă).

Vom prezenta principalele noţiuni care permit analogia dintre procesul de căutare a soluţii optime într-un algoritmi genetic şi evoluţia naturală, exemplificând aceste noţiuni pentru problema găsirii maximului unei funcţii pe un interval.

- **Cromozom** = mulţime ordonată de elemente (gene) ale căror valoare (alele) determină caracteristicile unui individ
- **Populaţie** = mulţime de indivizi care trăiesc într-un mediu la care trebuie să se adapteze
- **Fitness** (**adecvare**) = măsură a gradului de adaptare la mediu pentru fiecare individ (funcţie de fitness)
- **Generaţie** = etapă în evoluţia unei populaţii (o iteraţie în cadrul procesului de evoluţie al populaţiei)
- **Selecţie** = proces de selecţie naturală, prin care supravieţuiesc indivizii cu grad ridicat de adaptare la mediu
- **Reproducere** = proces prin care se trece de la o generaţie la alta

  Indivizii din noua generaţie se obţin prin încrucişarea (combinarea) părinţilor şi moştenesc caracteristici ale acestora, dar pot dobândi şi caracteristici noi, ca urmare a unor procese de mutaţie.
- **Operatori genetici**:
  - încrucişare (combinare)
  - mutaţie

Structura unui algoritm genetic (J. Holland, 1970)

t = 0
considerăm o populaţie iniţială P(0) - multiset al lui D
cât timp nu este îndeplinită condiţia de terminare
construim o populaţie nouă P(t+1) din P(t) astfel
- prin **selecţie** obţinem o populaţie intermediară P'(t)
- aplicăm **operatorul de încrucişare** pentru indivizii din P'(t) şi obţinem o nouă populaţie intermediară P''(t)
- aplicăm **operatorul de mutaţie** (şi obţinem populaţia P(t+1))

t = t + 1

- **Maximul unei funcţii**

1. Populaţia

- **Dimensiune (număr de cromozomi) :** fixă, dată; o vom nota n
- **Codificare =** cum asociem unei configuraţii din spaţiul de căutare un cromozom
  - În general: codificare binară, lungime fixă
  - Pentru D = [a,b] şi o **precizie p dată** (ca număr de zecimale):
    - se împarte intervalul D în subintervale de lungime egale cu 10^{-p} şi se consideră fiecare subinterval ca fiind un punct (**discretizarea intervalului**); obţinem $(b-a)*10^p$ subintervale
    - lungimea cromozomului l se determină astfel încât să poată codifica $(b-a)*10^p$ valori:

$$2^{l-1} < (b-a)10^p \le 2^l \quad \Rightarrow l = \lceil \log_2((b-a)10^p) \rceil$$

    - **Un cromozom X astfel codificat binar corespunde valorii din D=[a,b]**:

$$\frac{b-a}{2^l-1} X_{(10)} + a$$

unde $X_{(10)}$ reprezintă valoarea lui X în baza 10

- **Populaţia iniţială** se generează aleator

2. Funcţia de fitness f

- f – funcția de optimizat
- se pot folosi distanțe cunoscute (euclidiană, Hamming)

3. Selecţia

- **Selecţie proporţională**
  - Presupunem $P(t)=\{X_1,\ldots,X_n\}$
  - asociem fiecărui individ X_i o probabilitate p_i de a fi selectat, în funcţie de performanţa acestuia (dată de funcţia de fitness)

$p_i = \frac{f(X_i)}{F}$, unde $F = \sum_{j=1}^{n} f(X_j)$ este performanţa totală a populaţiei

- folosind **metoda ruletei** selectăm n indivizi (!clone), cu distribuţia de probabilitate (p_1, p_2, …, p_n)

Etapa de selecţie:

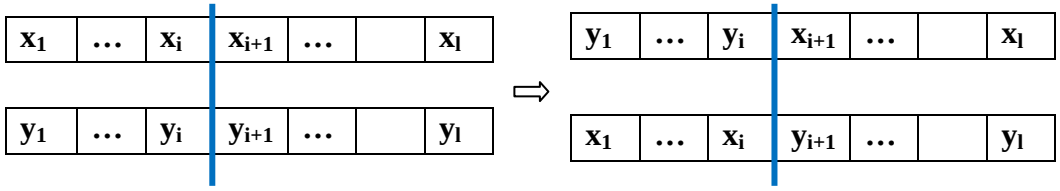
```
for i = 1,n
      genereaza j cu probabilitatea (p1,p2,…,pn) folosind
      metoda ruletei:
          • genereaza u variabila uniformă pe [0,1)
          • determină indicele j astfel încât u este între
            q_{j-1}= p1+…+p_{j-1} şi q_j= p1+…+p_j (cu convenţia q_0=0)

      adauga la populaţia selectată P'(t) o copie a lui X_j
```

- **Selecţie elitistă** = trecerea explicită a celui mai bun individ în generaţia următoare
- **Selecţie turneu** = se aleg aleatoriu k indivizi din populaţie şi se selectează cel mai performant dintre ei
- **Selecţie bazată pe ordonare:** se ordonează indivizii după performanţă şi li se asociază câte o probabilitate de selecţie în funcţie de locul lor după ordonare

4. Încrucişare (crossover) – permite combinarea informaţilor de la părinţi

- Doi părinţi dau naştere la doi descendenţi
- **Încrucişare cu puncte de tăietură generat aleator:** 2 părinţi ⇒ 2 indivizi noi care iau locul părinţilor în populaţie



i – punct de rupere generat aleator

- Nu toţi cromozomii participă la încrucişare. Un cromozom participă la încrucişare cu o probabilitate fixată **pc** (probabilitate de încrucişare – dată de intrare)

Etapa de încrucişare:

Notăm P'(t) = {X'_1,…,X'_n}

```
for i = 1,n
     genereaza u variabila uniformă pe [0,1)
     daca u<pc atunci marcheaza X'_i(va participa la incrucisare)

formeaza perechi disjuncte (X'_s, X'_s) de cromozomi marcaţi şi
aplică pentru fiecare pereche operatorul de încrucişare;
descendeţii rezultaţi înlocuiesc părinţii în populaţie
```

5. Mutaţie

- schimbarea valorilor unor gene din cromozom
- asigură diversitatea populaţiei
- probabilitatea de mutaţie pm – dată de intrare

Etapa de mutaţie:

Notăm P"(t) = {X"$_1$,…,X"$_n$} populația obținută după încrucișare

```
for i = 1,n
    genereaza u variabila uniformă pe [0,1)
    daca u<pm atunci generează o poziţie aleatoare p și trece gena
    p din cromozomul X"i la complement 0 ↔ 1
```

6. Condiţie de oprire

- număr maxim de iteraţii (numărul de generaţii) – data de intrare
  - diversitatea populaţiei
  - stabilizarea performanţei

Amintim **parametrii de intrare**:

- intervalul [a, b]
- precizia p
- dimensiunea populaţiei n
- numărul de generaţii
- probabilitatea de încrucişare pc
- probabilitatea de mutaţie pm

Alte exemple de probleme cunoscute:

1. Algoritmi genetici pentru problema rucsacului – varianta discretă (genetic algorithm for 0-1 knapsack problem)
2. Problema comis-voiajorului (TSP – travelling salesman problem)

Teorema schemei (suplimentar)

- **Schema** = tipar care surprinde similarităţi dintre cromozomi
  - Formal = cuvânt de lungime l peste {0,1,*}
    * = "don't care symbol" (înlocuiește un 0 sau un 1)

  Exemplu: Cromozomul 11010001 aparține schemei **1*01*00***
  - **Ordinul schemei** H o(H) = numărul de poziţii fixe din schemă ⇒ particularitatea schemei
  - **Lungimea schemei** H δ(H) = distanţa de la prima la ultima poziţie fixă ⇒ cât de compactă este informaţia
  - m(H,t) = numărul de exemplare ale schemei H în P(t)

- f(H,t) = fitness pentru schema H = media funcţiei de fitness pentru indivizi din P(t)

- **Teorema schemei:** Algoritmul genetic bazat pe selecţie proporţională, încrucişare cu un punct de tăietură şi mutaţie rară încurajează înmulţirea schemelor mai bine adaptate decât media, de lungime redusă şi de ordin mic:

$$m(H,t+1) \geq m(H,t)\frac{f(H,t)\cdot n}{F(t)}\left(1 - pc\cdot\frac{\delta(H)}{l-1} - pm\cdot o(H)\right)$$

- **Ipoteza blocurilor constituente**: Un algoritm genetic caută soluţia suboptimală prin juxtapunerea schemelor scurte, de ordin mic şi performanţă mare, numite ***building blocks*** (blocuri constituente/constructive)

Bibliografie. Aplicaţii

1. Michalewicz, Zbigniew (1999), Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag.
2. Mitchell Melanie, An Introduction to Genetic Algorithms, MIT Press
3. Aplicaţii https://en.wikipedia.org/wiki/List_of_genetic_algorithm_applications
4. http://courses.cs.washington.edu/courses/cse473/06sp/GeneticAlgDemo/
5. http://ocw.mit.edu/courses/engineering-systems-division/esd-77-multidisciplinary-system-design-optimization-spring-2010/lecture-notes/MITESD_77S10_lec11.pdf