Rezolvarea problemelor cu ajutorul metodelor de căutare locale

Obiective

Formularea problemelor ca probleme de optimizare a unei funcţii obiectiv numerice şi identificare modalităţilor locale şi evolutive de rezolvare a lor. Specificarea, proiectarea şi implementarea metodelor de optimizare evolutivă.

Aspecte teoretice

Rezolvarea problemelor ca proces de optimizare

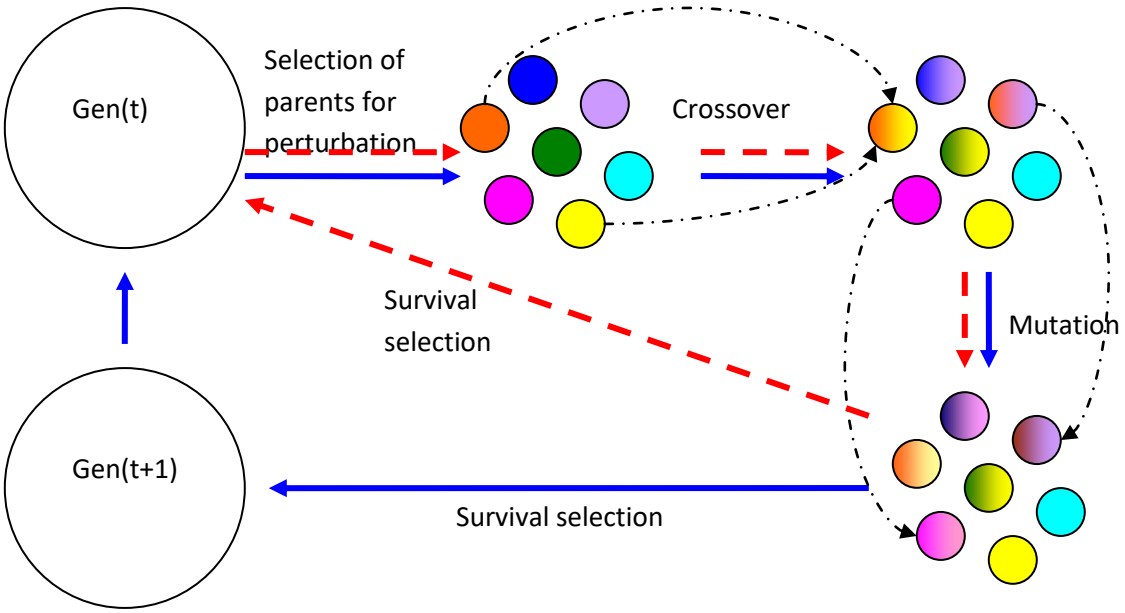
Tipuri de probleme de optimizare.

Modalităţi de rezolvare a problemelor de căutare → Identificarea soluţiei potenţiale optime:

- Stabilirea componentelor problemei
  - Condiţii (constrângeri) pe care trebuie să le satisfacă (parţial sau total) soluţia
  - Funcţie de evaluare a unei soluţii potenţiale → identificareaa optimului
- Definirea spaţiul de căutare
- Stabilirea strategiei de identificare a soluţiei optime în spaţiul de căutare

Probleme abordate

1. Schema generală a unui algoritm evolutiv – discutie privind schema geenrationala si cea steady-state.



Generational GA	Steady – state GA

Initialisation P(0)
Evaluation(P(0))
g = 0;
While (not stopCondition) do
  Repeat
  Select 2 parents p1 and p2 from P(g)
  Crossover(p1,p2) =>o1 and o2
  Mutation(o1) => o1*
  Mutation(o2) => o2*
  Evaluation(o1*)
  Evaluation(o2*)
  Add o1* and o* into P(g+1)
  Until P(g+1) is full.
  g++
EndWhile
	Initialisation P
	Evaluation(P)
	
	While (not stopCondition) do
	
	  Select 2 parents p1 and p2 from P
	  Crossover(p1,p2) =>o1 and o2
	  Mutation(o1) => o1*
	  Mutation(o2) => o2*
	  Evaluation(o1*)
	  Evaluation(o2*)
	  Best(o1*,o2*) replaces Worst(P)
	
	EndWhile

2. Paşi în dezvoltarea unui algoritm evolutiv
   - stabilirea unei reprezentări și a unei *mapări* de la genotip la fenotip
   - inițializarea populației
   - evaluarea calității unui individ (fitness)
   - managementul populatiei (algoritm generațional versus algoritm steady-state)
   - selecția părinților pentru încrucișare
   - încrucișarea
   - mutația
   - formarea noii generații (fără și cu elitism)
   - condiția de oprire a căutării
   - parametrii AG

3. Considerarea unor probleme si rezolvarea lor cu ajutorul AE.
   - Exemple de probleme:
     i. Submulţime de sumă dată – dându-se n numere întregi, să se găsească o submulţime de numere a căror sumă să fie S
        - Reprezentare binară (1 – element ales, 0 – element neales pt submulţime)
        - Fitness = abs(suma submulţimii – S)
        - Optimizare prin minimizare
        - selecție - de orice fel (exemplificarea unei anumite selecții)
        - încrucișare (de ex cu un punct de tăietură)
        - mutație (tare sau slabă)
     ii. Problema rucsacului
        - Reprezentare binară (1 – element ales, 0 – element neales pt rucsac)
        - Fitness = valoarea obiectelor din rucsac – (greutatea rucsacului – greutatea obiectelor puse în rucsac), dacă obiectele încap în

rucsac şi valoarea obiectelor din rucsac – abs(greutatea rucsacului – greutatea obiectelor puse în rucsac)*constantă_penalizare, altfel
- Optimizare prin maximizare
- selecție, încrucișare, mutație - similar cu problema precedentă

iii. Problema comisului voiajor
- Reprezentare permutare
- Fitness = lungimea drumului
- Optimizare prin minimizare
- selecție (similar cu problema precedentă)
- încrucișare
  - cu punct de tăietură, dar cu verificarea corectitudinii descendenților
  - ordonată
- mutație
  - prin interschimbare (swap mutation)

iv. Problema reginelor
- Reprezentare matrice cu 0 si 1 (1 pt locul fiecărei regine) sau reprezentare prin permutare (reprezentarea prin permutarea este mai eficientă)
- Fitness = nr de greşeli de amplasare a reginelor
- Optimizare prin minimizare
- selecție, încrucișare și mutație - similar cu problemele precedente