Simulated Annealing şi Tabu Search

1 CERINŢE

1. Să se implementeze algoritmul *Tabu Search* pentru problema rucsacului.
    a. Generare soluţie iniţială şi vecin
    b. Algoritm şi parametrizare
    c. Experimente pe aceleaşi instanţe de la Tema 1
    d. Comparaţii cu rezultatele obţinute de căutarea locală

2. Să se implementeze algoritmul *Simulated Annealing / Tabu Search* (la alegere) pentru problema TSP (instanţa este indicată la laborator).
    a. Generare soluţie iniţială şi vecin
    b. Algoritm şi parametrizare
    c. Experimente pe o instanţă TSP din lista de mai jos

Observaţii (neîndeplinirea cerinţelor din observaţii conduce la scăderea punctajului acordat):

- ✓ Instanţa TSP considerată din lista de mai jos = aceeaşi cu numărul din grupă.
- ✓ Aplicaţia trebuie să fie modularizată, să permită parametrizarea algoritmului şi afişarea soluţiei.
- ✓ Testaţi algoritmul în diverse variante pentru comparaţii pe instanţa TSP primită.
- ✓ Rezultatele experimentelor trebuie salvate (cu indicarea setărilor folosite: algoritmul, valori parametri, număr rulări – minim 10 rulări/configuraţie, calitatea soluţiei – best/avg).

2 TERMEN DE PREDARE

Lab 3

Total Punctaj Tema 2 = 150p

3 PREDAREA TEMEI PRIN MS TEAMS

Incarcati urmatoarele fisiere **INAINTE** de a incepe lab-ul in care este setat termenul de predare:

1. O arhiva cu codul sursa
2. Un document (Word/PDF) care sa contina:
   - ✓ Descrierea pe scurt a algoritmului implementat (pseudocod) si principalelor componente (reprezentare solutie, functie de fitness, operatori, etc)
   - ✓ Indicarea parametrilor algoritmului
   - ✓ Tabele/grafice cu rezultatele obtinute (comparatii pentru cel putin 3 seturi de valori ale parametrilor pentru fiecare instanta de problema)
   - ✓ Analiza rezultatelor

4 PROBLEMA COMIS-VOIAJORULUI

Este una dintre cele mai cunoscute probleme de optimizare combinatorială cu multe aplicaţii (vezi http://www.math.uwaterloo.ca/tsp/apps/index.html).

Se consideră n oraşe şi un comis-voiajor care trebuie să viziteze toate oraşele, trecând o singură dată prin fiecare şi să se întoarcă în oraşul iniţial astfel încât costul total să fie cât mai mic.

Soluţia poate fi reprezentată ca o permutare de n oraşe. Costul unei soluţii este suma distanţelor dintre oraşe în ordinea în care apar în permutare.

„*The simplicity of the statement of the problem is deceptive -- the TSP is one of the most intensely studied problems in computational mathematics and yet no effective solution method is known for the general case. Indeed, the resolution of the TSP would settle the P versus NP problem and fetch a $1,000,000 prize from the Clay Mathematics Institute.*”
[http://www.math.uwaterloo.ca/tsp/problem/index.html]

5 LISTA PROBLEME TSP

Instanţele TSP din lista de mai jos sunt disponibile în librăria online: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/

Documentaţie: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf

Listă probleme *Symmetric traveling salesman problem* (TSP):

1. eil101.tsp
2. eil176.tsp
3. berlin52.tsp
4. bier127.tsp
5. kroA100.tsp
6. kroB100.tsp
7. kroC100.tsp
8. kroD100.tsp

9. kroE100.tsp
10. lin105.tsp
11. pr76.tsp
12. p107.tsp
13. pr124.tsp
14. rat99.tsp
15. st70.tsp

6 SIMULATED ANNEALING

```
begin
    t = 0
    initialize T
    select a current point c at random
    evaluate c
    repeat
        repeat
            select a new point x from the neighborhood of c
            if eval(c) < eval(x)
                then c ← x
                else if random[0,1) < e^((eval(x)−eval(c))/T) then c ← x

        until (termination-condition)
        T ← g(T, t)
        t ← t + 1
    until (halting-criterion)
end
```

- Vecinătatea lui c poate fi dată de 2-swap / 2-opt

- Schema de răcire dată de funcţia g poate fi:
  - T(k+1)= alpha * T(k), cu alpha o valoare subunitară dar apropiată de 1 (de exemplu, alpha=0.99)
  - T(k+1)=T(k)/log(k+1) ➢ T(k+1)=T(k)/k

  unde T(k) este temperatura la iteraţia k.

  Valoarea iniţială a temperaturii trebuie să fie suficient de mare ex. T=10000.

- Criteriul de oprire poate fi:
  - Când valoarea temperaturii ajunge la o valoare minimă prestabilită
  - Numărul de iteraţii parcurse ajunge la un maxim
  - Valoarea atinsă de funcţia de evaluare

7 TABU SEARCH

Un algoritm standard de tabu search este dat mai jos.

```
c = initSolution()
best = c
M = initMemory()
while (stop-criterion not met)
        x = getBestNeighbourNonTabu(c)
        updateMemory()
        c = x
        update best
return best
```

Condiția de oprire este dată de un număr maxim de iteraţii. Memoria reține numărul de iteraţii în care nu este permisă obţinerea unui vecin prin 2-swap sau 2-opt pe anumite oraşe.