Idei Proiecte cursul Tehnici de Optimizare

Beniamin Bogosel

March 4, 2025

Pentru fiecare proiect ales se aşteaptă:

- o prezentare detaliată a problemei/algoritmului propus (incluzând exemple)
- discutarea etapelor algoritmului şi justificarea lor
- prezentarea simulărilor şi rezultatelor
- livrarea codului (inclusiv instrucţiunile de executare)
- se recomandă utilizarea `Python`. Dacă doriţi se pot folosi şi alte limbaje de programare.
- **grup de maxim 3 persoane**: se va detalia implicarea fiecărui membru din proiect în rezolvarea acestuia
- **susţinerea proiectului:** va avea loc în timpul cursului (maximum 20-30 de minute). Fiecare membru al grupului va trebui sa intervină în prezentare.
- Este posibil ca mai multe grupe să lucreze pe acelaşi proiect, cu condiţia să se prezinte aspecte diferite (în mod rezonabil, nu toată lumea pe acelaşi proiect).

Un proiect de calitate va duce la **echivalarea examenului**.

1. **Project Euler Problem 185**: `https://projecteuler.net/problem=185`
   - trebuie găsit un cod de cifre având o lungime dată
   - pentru o încercare dată avem ca si output numărul de cifre corecte
   - pentru rezolvare se poate folosi programarea liniară (se pot utiliza algoritmi din `scipy`)
   - (Challenge) Se poate crea un script prin care introduceţi succesiv coduri care se adaugă la datele ce le aveţi deja. Programul se opreşte când suficiente informaţii au fost recuperate pentru a găsi codul.
2. **Simulated Annealing algorithm**: `https://www.baeldung.com/cs/simulated-annealing`
   - Algoritmul propune modificarea variabilei curente folosind o metodă ce include numere aleatorii
   - Se acceptă variabila următoare cu o probabilitate mai mult sau mai puţin ridicată, depinzând de ”temperatura” sistemului (la început permitem perturbări mai mari, care apoi sunt reduse tot mai mult).
   - Se reţine de fiecare dată soluţia cea mai bună.
   - **(Proiect 1)**: implementarea algoritmului şi exemplificarea lui pe cel puţin două probleme date

- **(Proiect 2)**: implementarea algoritmului şi aplicarea lui pentru **Travelling salesman problem**.

3. **Rezolvarea Sudoku folosind programarea liniară**:

   - https://towardsdatascience.com/solve-sudoku-using-linear-programming
   - modelizarea problemei
   - sublinierea faptului că este nevoie de programare liniară folosind numere întregi
   - implementare folosind librăria `PuLP` in `Python`
   - Exemplificare.

4. Determinarea razei pământului ştiind distanţele între 4 oraşe:

   - https://mathproblems123.wordpress.com/2023/11/28/is-the-earth-flat/
   - Testarea pentru cel puţin 3 alte configuraţii care nu sunt în documentaţia dată
   - Se poate deasemenea folosi timpul de zbor al avioanelor (şi viteza medie) pentru a avea noi date disponibile.
   - **(Proiect 1)** determinarea razei folosind triunghiuri sferice.
   - **(Proiect 2)** determinarea razei folosind o problemă de optimizare.

5. Implementarea de algoritmi genetici:

   - Se menţine o populaţie de N "indivizi" (variabile); se evaluează funcţia obiectiv pe această populaţie şi se reţine o proporţie de $p\%$ cei mai "buni" indivizi.
   - Folosind această sub-populaţie se generează o nouă populaţie de N indivizi folosind o procedură de combinare (inventată de voi)
     - Exemplu: se iau doua variabile si se intercalează cifrele acestora
   - Pentru un anumit număr de indivizi noi se introduc "mutaţii" aleatorii (inventate de voi)
     - Exemplu: schimbarea aleatorie a unei cifre dintr-un număr, adăugarea unui număr real aleatoriu în $[-\varepsilon, \varepsilon]$, etc.
   - În fiecare generaţie se păstrează un număr prescris de cei mai buni indivizi.
   - Scopul proiectului este să exploraţi opţiunile pentru partea de "combinări" şi "mutaţii".
   - Algoritmul propus se va exemplifica pe o problemă în dimensiune 2.
   - Se va totaliza numărul de evaluări ale funcţiei obiectiv pentru a ajunge la rezultatul dorit.

6. Rezolvarea 15-puzzle (sau 8-puzzle) într-un număr minimal de mutări, folosind intuiţia dată de algoritmul Dijkstra (sau varianta acestuia, algoritmul A^*): https://www.cs.princeton.edu/courses/archive/spring09/cos226/assignments/8puzzle.html

   - Pornind de la o configuraţie iniţială, se doreşte listarea mutărilor ce duc la soluţie într-un număr minimal de mutări
   - Se va testa algoritmul simultan pentru două iniţializări: o configuraţie şi o alta obţinută prin inversarea poziţiei a două celule adiacente; doar una din configuraţii va permite ajungerea la configuraţia dorită.

7. Determinarea traiectoriei de lungime minimala a unui robot în jurul unui obstacol:

- Robotul poate ocupa o poziţie (i, j) cu i, j numere naturale cuprinse între: $0 \leq i \leq N_x$, $0 \leq j \leq N_y$
- Ca şi input se vor da poziţiile iniţiale şi finale;
- Robotul poate trece din nodul (i, j) în noduri vecine (lateral, vertical sau în diagonală)
- Obstacolul este modelizat cu o matrice A: $A[i, j] = 0$ în afara obstacolului, $A[i, j] = 1$ în interiorul obstacolului
- Implementarea se poate realiza folosind algoritmul Dijkstra. (Construcţia grafului nu este necesară).
- Traiectoria optimală se va afişa împreună cu obstacolulul, punctul de pornire şi punctul final.