Proiect: Implementare modernă a metodelor lui Kantorovich în Python

1. Ce articole există pe tema aceasta și ce s-a mai făcut până acum

Articole și lucrări semnificative:

1. **L. V. Kantorovich (1939)** – *"Mathematical Methods of Organizing and Planning Production"* – lucrare fondatoare care introduce concepte de optimizare liniară în economie.
2. **G. B. Dantzig (1947)** – Dezvoltarea algoritmului simplex, care a făcut programarea liniară practic aplicabilă.
3. **T. C. Koopmans (1951)** – A extins aplicațiile în economie, împărțind Premiul Nobel cu Kantorovich.
4. **Lucrări moderne** – În domeniul *Operations Research* și *Management Science* există numeroase studii care aplică programarea liniară în:
    a. Planificarea producției
    b. Optimizarea lanțului de aprovizionare
    c. Managementul resurselor

Ce s-a făcut până acum:

- Problemele lui Kantorovich au fost rezolvate inițial manual sau cu metode numerice rudimentare.
- În prezent există biblioteci software (CPLEX, Gurobi, PuLP) care rezolvă eficient probleme de programare liniară.
- **Lacuna identificată:** Implementări moderne educaționale care să conecteze lucrarea istorică cu instrumentele actuale, într-un format accesibil.

2. Ce date vom folosi și un exploratory data analysis

Date utilizate:

1. **Date simulate** bazate pe exemplele din lucrarea lui Kantorovich:
    a. Productivitatea mașinilor pentru diferite piese
    b. Timpi de procesare și cerințe de producție

c. Dimensiuni ale materialelor (scânduri, țevi, etc.)

2. **Seturi de date publice** pentru analogii moderne:
    a. Date de producție din UCI Machine Learning Repository
    b. Date de transport/logistică

Exploratory Data Analysis (EDA):

1. **Analiza statistică descriptivă:**
    a. Medii, deviații standard, distribuții
    b. Corelații între productivitate și tipurile de mașini
2. **Vizualizări:**
    a. Heatmaps pentru productivități
    b. Diagrame de dispersie pentru relații între variabile
    c. Grafice de bare pentru comparații între metode
3. **Identificarea patternurilor:**
    a. Care mașini sunt cele mai versatile
    b. Unde apar pierderi mari de material

3. Ce modele vom folosi și necesarul de calcul

Modele de programare liniară:

1. **Probleme de alocare optimă** :
    a. Maximizarea producției cu resurse limitate
    b. Formulare: Maximizează (z) cu restricții (z_k = z) 2.

Probleme cu restricții suplimentare:

a. Limitări de energie, forță de muncă, etc.

3. **Probleme de minimizare a deșeurilor**:

a. Minimizarea pierderilor la tăierea materialelor

Implementare:

- Folosim **biblioteci Python**:
    - PuLP pentru modelare intuitivă
    - SciPy.optimize.linprog pentru rezolvări eficiente

- NumPy și Pandas pentru manipularea datelor
- Matplotlib/Seaborn pentru vizualizări

Necesar de calcul:

- **CPU standard** – suficiente pentru probleme de dimensiune mică/mijlocie (până la 1000 de variabile)
- **Memorie RAM**: 4-8 GB
- **Nu este necesar GPU** – programarea liniară se rezolvă eficient pe CPU
- **Timp de calcul estimat**: sub 5 secunde per problemă pentru exemplele din Kantorovich

4. Ce metode de evaluare și comparație a modelelor vom folosi

Metode de evaluare:

1. **Comparație cu soluțiile originale**:
   a. Verificarea acurateței prin compararea cu rezultatele lui Kantorovich
   b. Analiza diferențelor și a surselor acestora
2. **Metrici de performanță**:
   a. Timp de execuție pentru fiecare algoritm
   b. Precizia soluției (diferența față de optim)
   c. Scalabilitatea la dimensiuni mai mari
3. **Analiza fezabilității**:
   a. Verificarea dacă soluțiile satisfac toate constrângerile
   b. Analiza sensibilității la variații în datele de intrare

Comparație între abordări:

1. **Metoda lui Kantorovich (multiplicatori rezolvatori)** vs. **Algoritmul Simplex** vs. **Metode de punct interior**
2. **Compararea bibliotecilor**:
   a. PuLP (cu diferiți solvere)
   b. SciPy.optimize.linprog
   c. CVXOPT (pentru probleme convexe)
3. **Evaluarea ușurinței de implementare** și clarității pentru educație

Validare:

- Testarea pe multiple seturi de date simulate
- Verificarea consistenței rezultatelor
- Benchmark împotriva unor probleme standard din literatură

Structura proiectului final:

1. Jupyter Notebook cu tutorial de programare liniară
2. Implementarea problemelor lui Kantorovich
3. Analiza comparativă a metodelor
4. Visualizări interactive
5. Documentație și referințe