Arbore de acoperire capacitat de cost minim

27 mai 2025

1 Definirea problemei

Fie un graf conex neorientat $G = (V, A, b, c)$ cu noduri $V = \{0, 1, 2, ..., n\}$ si multimea de muchii A unde 0 este nodul sursă (rădăcina). Fiecare nod $i \in V \setminus \{0\}$ are o cerere b_i, iar fiecare arc $(i, j) \in A$ are un cost c_{ij}.

Se cere să se construiască un arbore de acoperire T care:

- Respectă constrângerea: suma cererilor din fiecare subarbore incident în rădăcină să nu depășească o capacitate K.
- Minimizează suma costurilor totale ale arcelor din arbore.

2 Definitia in contextul programarii liniare

$$
\begin{aligned}
\min \quad & \sum_{i=0}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \\
\text{a.i.} \quad & \sum_{i=0}^{n} x_{ij} = 1, \quad j = 1, \ldots, n, \\
& \sum_{i=0}^{n} y_{ij} - \sum_{i=1}^{n} y_{ji} = 1, \quad j = 1, \ldots, n, \\
& x_{ij} \leq y_{ij} \leq (K - b_i) \cdot x_{ij}, \quad \forall\, i, j, \\
& x_{ij} \in \{0, 1\}, \quad y_{ij} \geq 0 \quad \forall\, i, j.
\end{aligned}
$$

3 Aplicații

Problema CMST apare în:

- Proiectarea rețelelor de telecomunicații și cablare.
- Organizarea distribuției în rețele logistice.
- de adaugat aici pentru licenta finala mai multe..

4 Metode de rezolvare

Fiind o problemă NP-dificilă, algoritmii exacți sunt eficienți doar pentru instanțe mici, insa putem folosi euristici si meta-euristici pentru o solutie aproximativa

5 Euristica Esau-Williams

Euristica Esau-Williams este mentionata in mai multe articole legate de CMST si este des folosita ca un punct de plecare pentru algoritmi mai complexi sau meta-euristici, aceasta se bazeaza pe conceptul de savings si incearca sa aleaga muchia cu diferenta maxima intre costul drumului de la componenta actuala la radacina si costul muchiei.
Formula de baza pentru saving s_{ij} pentru conectarea componentelor C_i si C_j este $s_{ij} = max(x_i, x_j) - c_{ij}$, daca avem o conexiune viabila si $s_{ij} = \infty$, in caz contrar. In cadrul acestei formule x_i reprezinta costul celui mai scurt drum de la componenta C_i la radacina, iar c_{ij} costul minim al unei muchii intre C_i si C_j.

6 Implementare

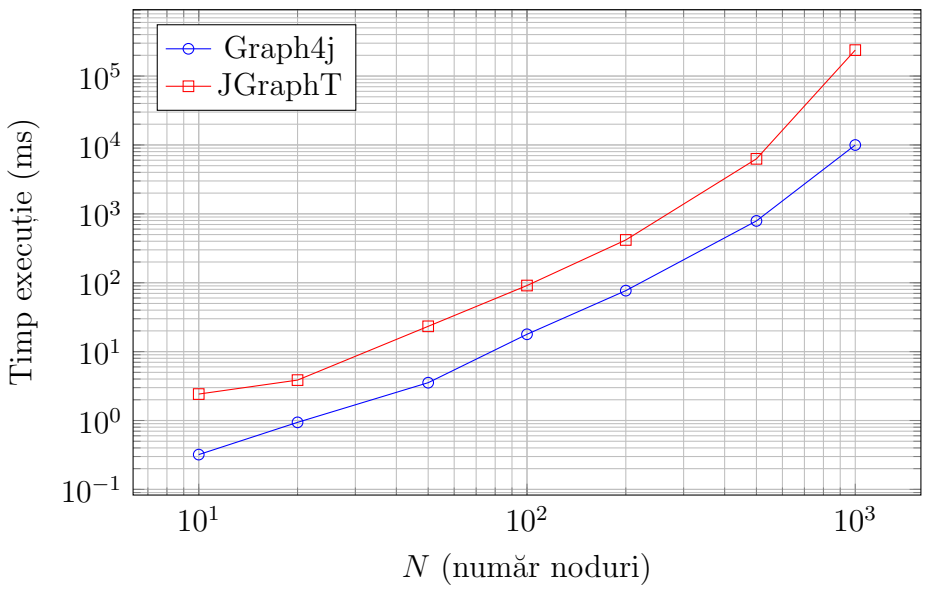
Implementarea facuta foloseste mai multi vectori primitivi in java pentru a stoca informatii ce ne vor ajuta sa facem alegerea si care sunt actualizati constant la fiecare schimbare in graf. O structura de tip UnionFind este folosita pentru gestionarea componentelor in cadrul algoritmului si a indexa in vectorii specifici:

- savings
- gates
- closest
- subtreeCosts

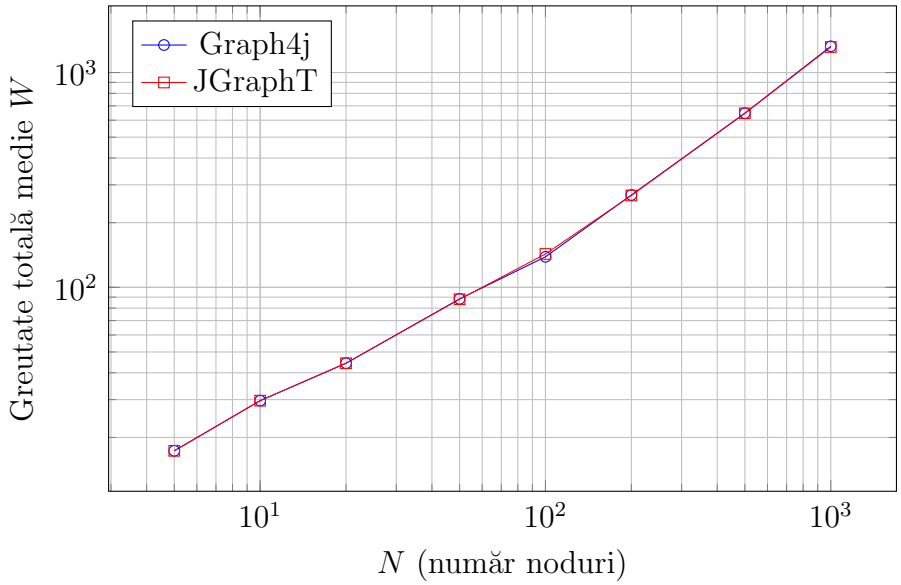
astfel muchia aleasa la un pas va fi $(argmax(savings), closest[argmax(savings)])$, atunci cand avem cel putin un saving pozitiv, altfel algoritmul se opreste. gates si subtreeCosts sunt folositi pentru calculul savings

7 Comparatie cu JGraphT

Comparație timp execuție: Graph4j vs JGraphT



Comparație greutate totală: Graph4j vs JGraphT



Diferentele de avg. weight sunt minime(implementeaza aceeasi euristica), dar diferentele in timpul de executie sunt notabile. Totusi JGraphT iti da posibilitatea de a rula o versiune randomized a euristicii (versiunea implementata in graph4j nu)

Bibliografie

TODO