Sisteme de ecuații liniare

Bahrim Dragoș (464)
Ionescu Alexandru-Theodor (461)
Maxim Tiberiu (461)

Universitatea din București
Calculatoare și Tehnologia Informației
2023

Outline

- descrierea problemei
- specificația soluției
- design și alegerea algoritmului paralel
- implementare
- experimente
- concluzii

Descrierea problemei

- sistemele de ecuații liniare sunt o modalitate de a modela și rezolva probleme precum simularea sistemelor dinamice sau interacțiuni între factori economici
- în general problema este reprezentată sub formă matriceală unde fiecare rând reprezintă o ecuație iar fiecare coloană coeficientul variabilei respective alături de un vector de termeni liberi ce conține numărul cu care fiecare ecuație este egal

$$\begin{cases} x_1 + x_2 + x_3 = 6 \\ -x_2 - 3x_3 = -11 \\ 13x_3 = 39 \\ 0 = 0 \end{cases}$$

Descrierea problemei

- în cazul matricelor de dimensiuni foarte mari pot exista încetiniri din cauza numărului mare de calcule necesare sau constrângeri legate de memorie
- paralelizarea sarcinilor ce pot fi făcute independent poate fi o opțiune de a rezolva aceste probleme

Descrierea problemei

- totuși trebuie găsit un echilibru întrucât comunicările frecvente duc la încetiniri, asta reprezentând o problemă în cazul sarcinilor implicit secvențiale
- este necesară alegerea unei topologii ce complementează algoritmul de calcul și distribuția sarcinilor eficient

Specificatia solutiei

- Obiectivul principal al soluției reprezintă implementarea unor algoritmi ce pot rula pe un număr variabil de procesoare
- Ne conferă avantajul de a rula pe orice tip de arhitectură
- Dezavantajul este reprezentat de consum suplimentar de memorie și incapacitatea de a aduce optimizări problemei ce ar aduce limitări asupra topologiei ce duc la necesitatea unui număr specific de procesoare

Specificatia solutiei

- Implementarea oferă posibilitatea de a rezolva sisteme de ecuații liniare ce sunt reprezentate prin matrici
- Calculul se poate face folosind metoda Gauss prin eliminare, Jacobi și Gauss-Seidel
- Un utilizator va interacționa cu o interfață web pentru a interacționa cu implementarea
- Dezvoltarea trebuie să fie cât mai generică, astfel poate să fie compatibilă cu cât mai multe medii

Specificatia solutiei

- Limitările soluției apar din cauza operațiilor cu numere în virgulă mobilă ce oferă rezultate eronate atunci când sunt necesare aproximări
- Evaluarea se face observând timpul necesar calcului paralel, adică timpul de găsire al unei soluții sau de terminare a numărului de iterații pentru soluțiile secvențiale

Design

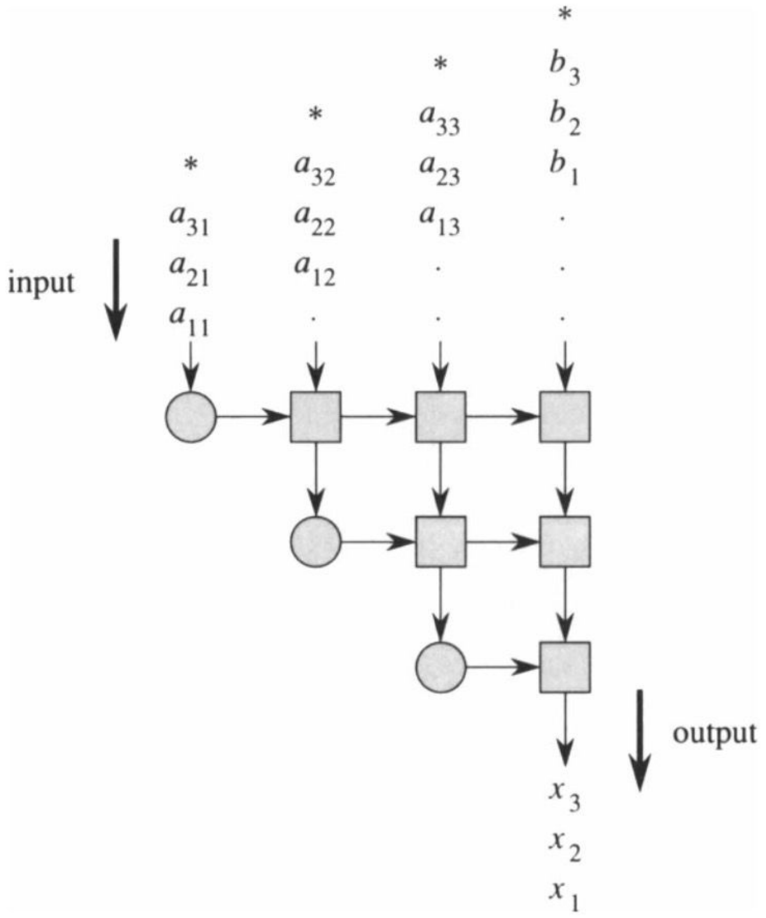
- Metoda Gauss prin eliminare

$$\begin{cases} x + y + z = -9 \\ -y - (-4) = 7 \\ z = -4 \end{cases} \Leftrightarrow \begin{cases} x + y + z = -9 \\ -y = 7 - 4 \\ z = -4 \end{cases} \Leftrightarrow$$

$$\begin{cases} x + (-3) + (-4) = -9 \\ y = -3 \\ z = -4 \end{cases} \Leftrightarrow \begin{cases} x = -9 + 4 + 3 \\ y = -3 \\ z = -4 \end{cases}$$

Design

- Metoda Gauss prin eliminare



Design

- Metoda Gauss prin eliminare

$$C_{jk}^{(i)} = C_{jk}^{(i-1)} - \sum_{\ell=1}^{n} N_{j\ell}^{(i)} C_{\ell k}^{(i-1)} = C_{jk}^{(i-1)} - N_{ji}^{(i)} C_{ik}^{(i-1)} = C_{jk}^{(i-1)} - \frac{C_{ji}^{(i-1)}}{C_{ii}^{(i-1)}} C_{ik}^{(i-1)}.$$

Design

- Metoda Jacobi

$$x_i(0) \; = \; [k\; k\; k\; k\; \dots\; k] \;,\; k \; - \; valoare\; aleatoare\; (de\; obicei\; 0\; sau\; 1);\; i \in [1, n]$$

$$x_i(t + 1) = (b_i - \sum_{j \neq i} (A_{ij} * x_j(t)))/A_{ii}$$

Design

- Metoda Jacobi

❖ Cazul n-linii, n-procese

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ . & . & . & . \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ . \\ b_n \end{bmatrix}$$

❖ Exemplificare caz:
4 linii , 3 procese

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ a_{41} & a_{42} & \cdots & a_{4n} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Design

- Metoda Gauss-Seidel

$$x_i(0) \ = \ [k\ k\ k\ k\ \dots\ k] \ , \ k \ - \ valoare\ aleatoare\ (de\ obicei\ 0\ sau\ 1);\ i \in [1, n]$$

$$x_i(t\ +\ 1) = (b_i - \sum_{j<i} (A_{ij} * x_j(t\ +\ 1) \ - \ \sum_{j>i} (A_{ij} * x_j(t))/A_{ii}$$

Design

- Metoda Gauss-Seidel

exemplu **SPLIT_FACTOR = 2**

P1: A1, b1
P1: A2, b2
P2: A3, b3
P2: A4, b4
P3: A5, b5
P3: A5, b6
…
Pn: An, bn

Implementare

- Metoda Gauss prin eliminare
  - procesul master citește matricea si distribuie elementele corespunzătoare
  - la fiecare iterație fiecare proces calculează parțial matricea C
  - matricea C este sincronizată printr-un broadcast general
  - la final, procesul master face back substitution pentru a afla soluția

Implementare

- Metoda Jacobi
  - procesul MASTER va citi matricea extinsă, verifică dacă este diagonal dominantă și va distribui liniile corespunzătoare celorlalte procese
  - prima asignarea a vectorului de soluții va fi cu valori aleatoare (de obicei 0 sau 1)
  - încep iterațiile și fiecare proces va calcula valoarea x corespunzătoare. La finalul iterației curente, toate valorile vor fi stocate, în vederea utilizării lor la iterația următoare
  - la sfârșitul iterațiilor, procesul MASTER va afișa soluția finală

Implementare

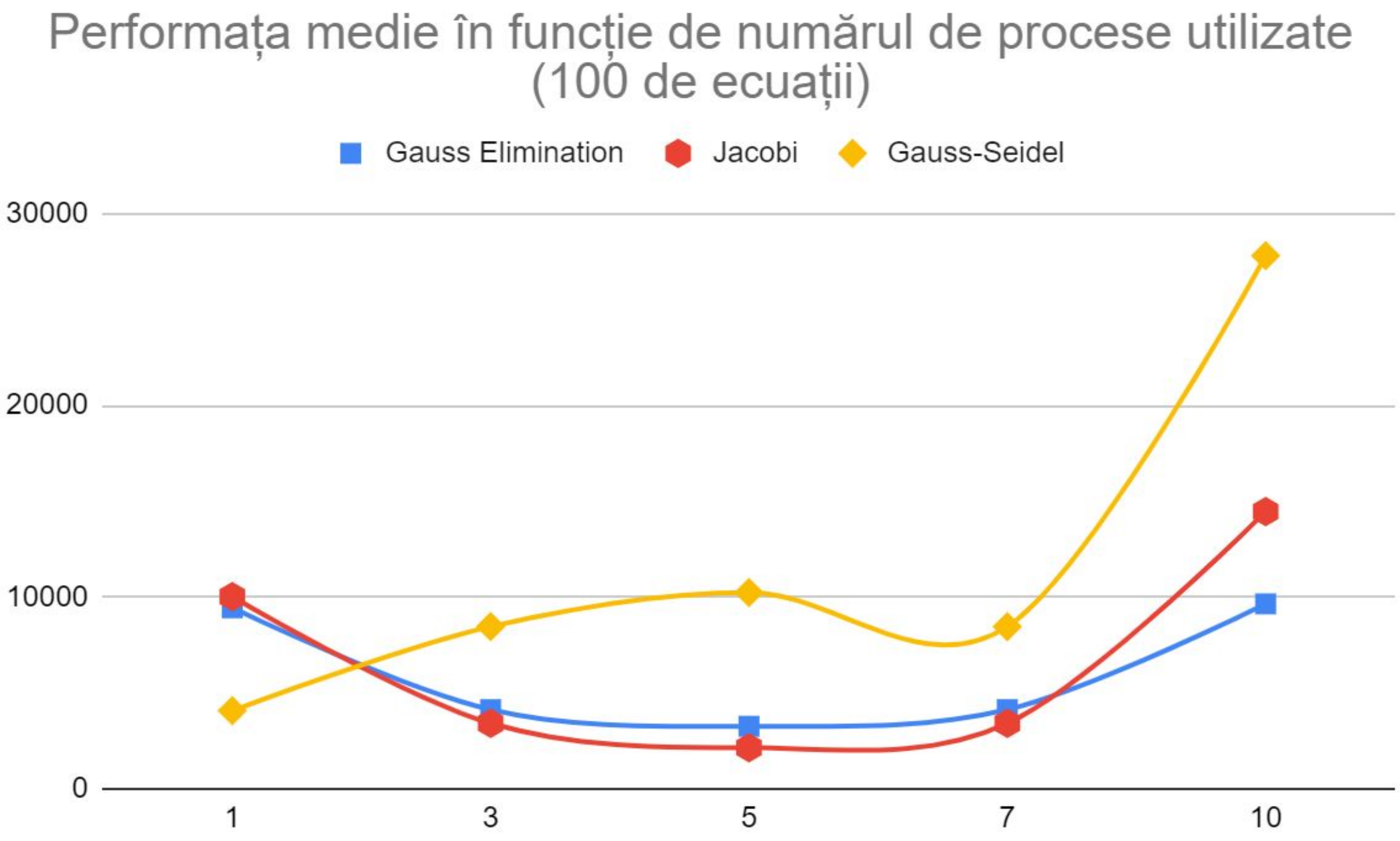
- Metoda Gauss-Seidel
  - procesul MASTER citește datele de intrare, calculează SPLIT_FACTOR și atribuie proceslor numărul de linii aferent acestora
  - se folosesc, în calculul componentelor din soluției locale, două variabile care stochează soluția precedentă și soluția curentă, pe baza formulei
  - acestea sunt trimise mai departe proceselor de rank superior
  - în ultimul proces, soluția curentă suprascrie soluția precedentă și este trimisă la procesul MASTER pentru o nouă iterație
  - la sfârșitul iterațiilor, procesul MASTER va afișa soluția finală

Experimente

- poate fi rulat pe orice dispozitiv datorită lipsei de constrângeri legat de arhitectură
- testele sunt executate pe un procesor cu 10 procesoare fizice

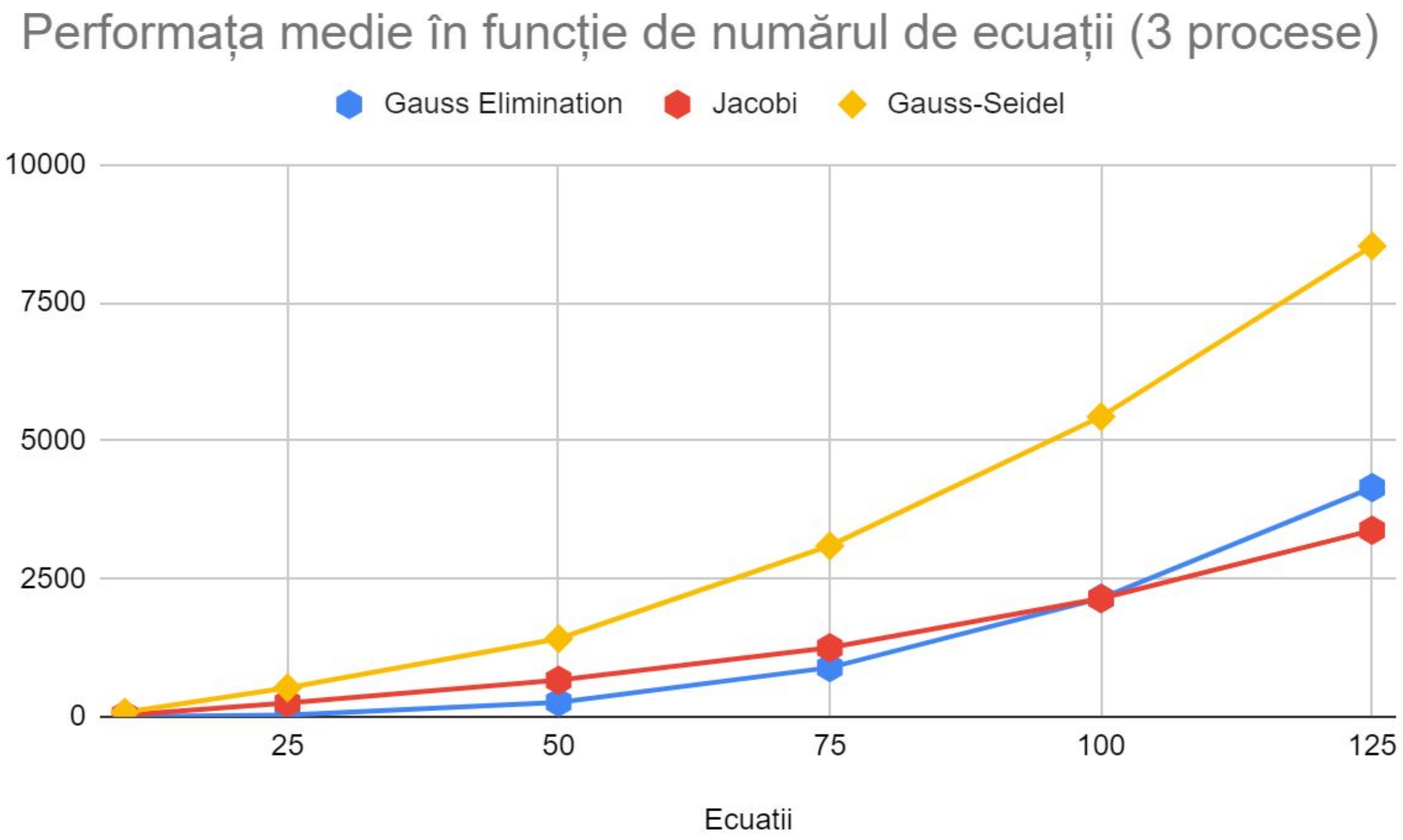
Experimente

- performanța medie în funcție de numărul de procese utilizate (100 de ecuații, 500 iterații)



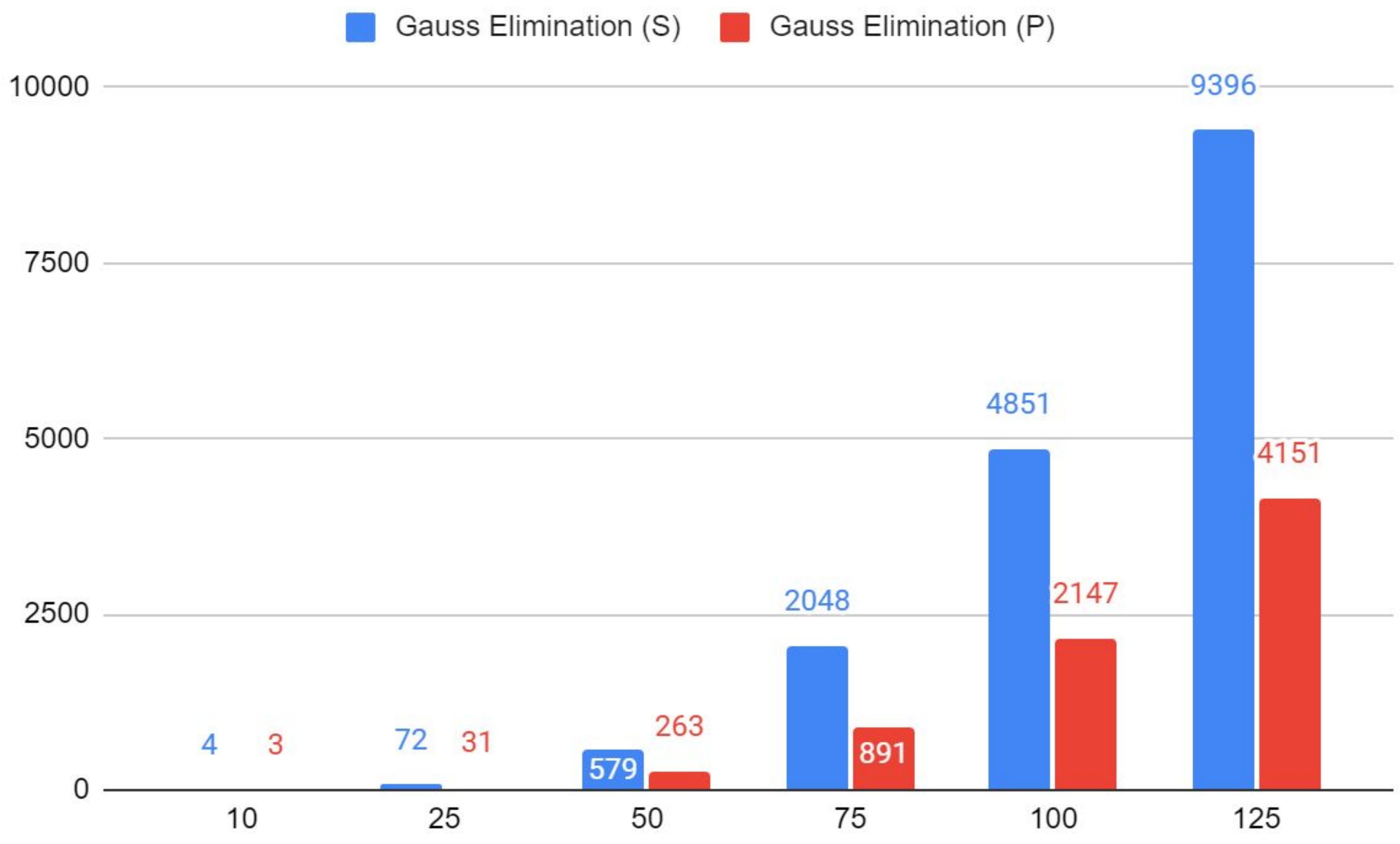
Experimente

- performanța medie în funcție de numărul de ecuații utilizate (3 procese, 500 iterații)



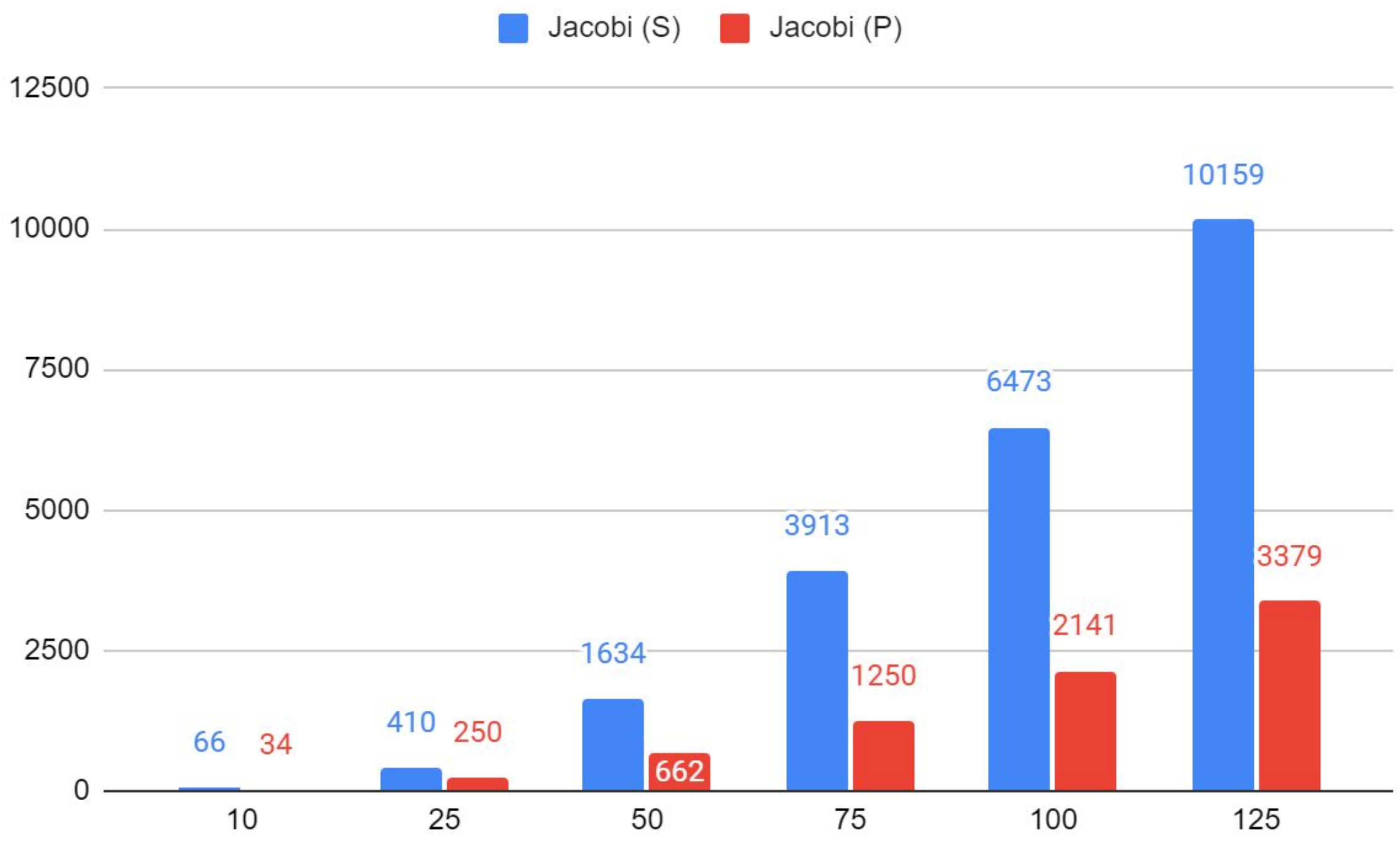
Experimente

- comparația dintre algoritmii secvențiali și cei paraleli, în materie de timp de execuție



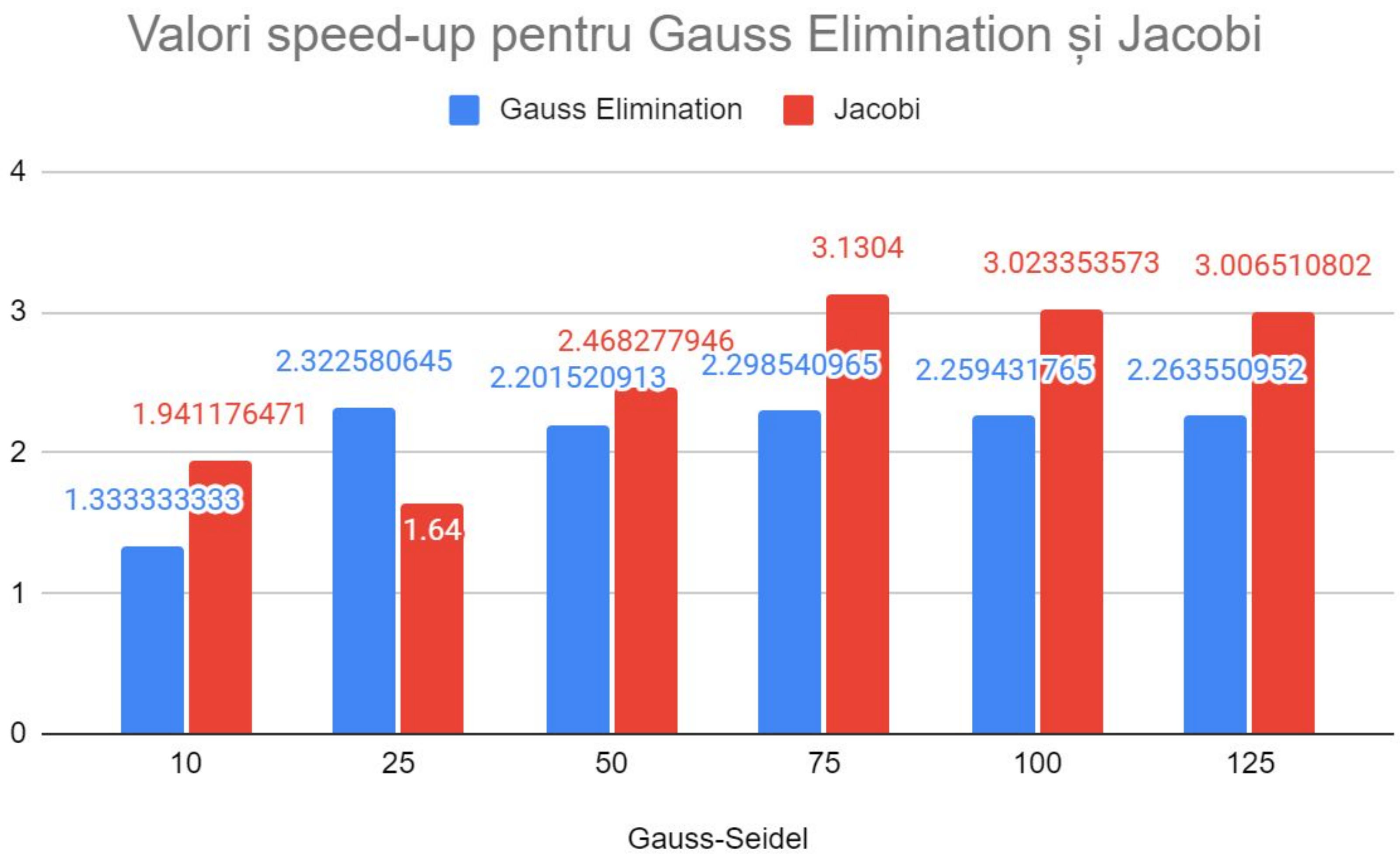
Experimente

- comparația dintre algoritmii secvențiali și cei paraleli, în materie de timp de execuție



Experimente

- valorile calculate pentru speed-up



Concluzii

- Definirea topologiilor este importantă în abordarea paralelă a problemei.
- Creșterea numărului de procesoare nu duce întotdeauna la îmbunătățirea rezultatelor din cauza necesității de comunicare frecventă.
- Calculul cu numere în virgulă mobilă poate fi dificil din cauza pierderii de precizie și a rezultatelor incorecte.
- Comunicarea poate fi îmbunătățită prin utilizarea topologiilor specifice per algoritm.