UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

PROCEDURAL COMPUTING OF STRING ART

Absolvent
Mincu Adrian-Lucian

Coordonator științific
Profesor Rusu Cristian

București, iunie 2025

Rezumat

Această lucrare de licență abordează tema computației procedurale a ceea ce se numește „String Art", utilizând diverse metode algoritmice. O sursă semnificativă de inspirație a fost lucrarea de certare numită „String Art: Towards Computational Fabrication of String Images" [2].

Conform definiției oferite de Wikipedia [1], „String art este îndemânarea folosirii ației cu scopul de a crea tablouri pe lemn și carton. Acest lucru se poate realiza fie răsucind ața sau sârma, în mod atractiv și simetric, în jurul cuielor de pe o scândură, fie formând desene geometrice, prin coaserea ației pe carton."

În mod normal, artiștii ar trebui să găsească intuitiv un aranjament al ației care să recreeze o imagine dată. De aceea, o să tratez problema găsirii unei căi matematice de a computa string art, reușind să automatizez procesul.

Scopul intregii lucrări este de a implementa mai multe metode algoritmice care să rezolve această problemă, comparându-le pe toate, astfel putând crea o analiză care să evidențieze metoda cea mai eficientă din punct de vedere al timpului, memoriei, cât și calității vizuale al rezultatului.

Astfel, lucrarea va contribui în domeniul procesării de imagini, oferind o nouă perspectivă asupra aplicațiilor artistice ale matematicii și programării.

[1] Pentru mai multe detalii, consultați definiția String Art disponibilă pe Wikipedia

Cuprins

Capitolul 1

Introducere

1.1 String Art și obiectiv

Așa cum am spus mai sus, string art este o tehnică de a crea un efect vizual manipulând ața pentru a forma un model și a reproduce o imagine.

Obiectivul este de a replica acest efect, în mod digital, plecând de la o imagine dată ca input. Trebuie să ținem minte că există o multitudine de metode ca să obținem acest obiectiv, iar pentru fiecare metodă o să trec și peste detaliile mai interesante. Spre exemplu, o să încep cu o imagine alb-negru, dar voi explora și metode pentru a crea versiuni cu spectru complet RGB.

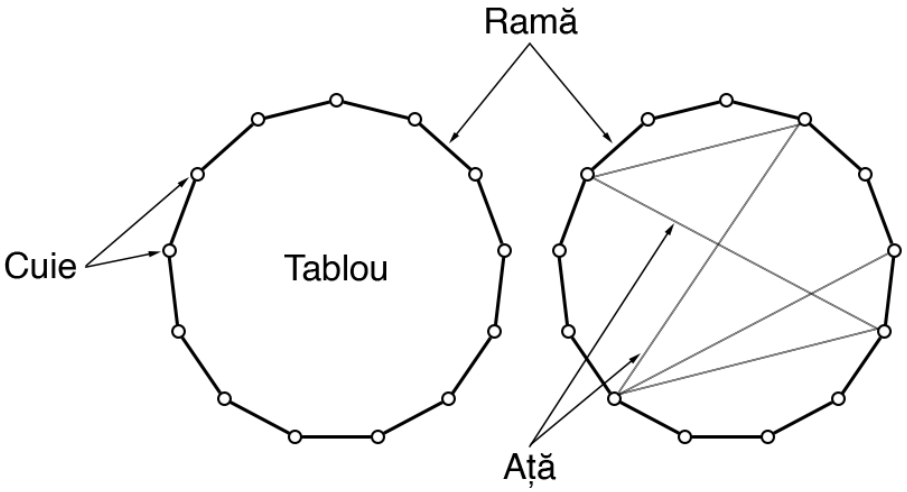

Figura 1.1: Imagine care să illustreze fiecare componentă din procesul de string art.

1.2 Limitări fizice

Definiția string art-ului pe care am adresat-o mai devreme nu o să fie aplicată complet când voi recrea imaginile digital, adică unele reguli vor fi relaxate pentru a explora posibilitățile și limitele acestui domeniu. Asta pentru că, obiectivul nostru este să replicăm un efectul vizual dat de metoda tradiționlală și vrem să ne apropriem de imaginea originală cât mai mult posibil, fără a fi limitați de constrângerile fizice precum folosirea unui fir de ață continuu.

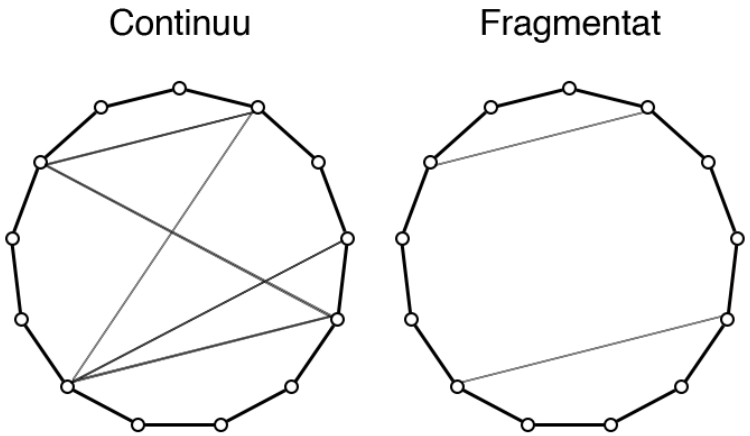

Figura 1.2: Imagine care să illustreze diferite tipuri de aranjamente: continuu versus fragmentat.

1.3 De ce funcționează string art

Tehnica string art este similară cu modul de reprezentare al formelor in fișierele .svg. Asta ne ajută să creăm orice formă dorim. În acest fel, tot procesul devine posibil.

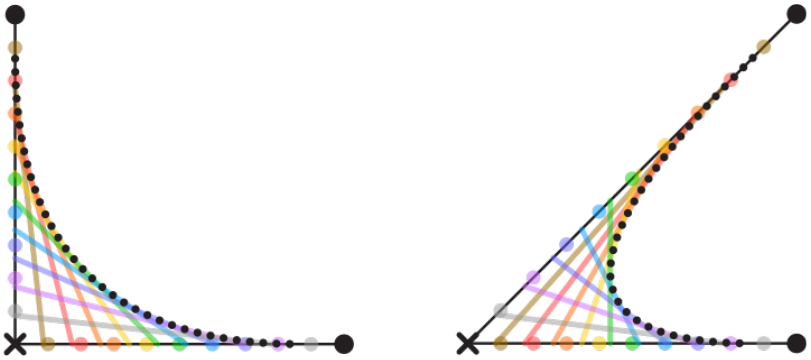

Figura 1.3: Imagine care să ilustreze curbele Bézier cuadratice obtinute dintr-o ață.

Chiar dacă suntem limitați să trasăm numai linii drepte, putem crea iluzia unei curbe utilizând mai multe linii drepte care urmează traseul unei curbe Bézier cuadratice.

Capitolul 2

Preliminarii

2.1 Terminiologie

Putem defini urmatoarele notații:

$$m = \text{Numărul de linii din imagine}$$

$$n = \text{Numărul de coloane din imagine}$$

$$y = \text{Imaginea inițială alb-negru. } y \in \mathbb{R}^{m,\,n}$$

Imaginea de input din pipeline va fi o imagine normală, inițial convertită în spectrul BW(black and white sau alb-negru) pentru a simplifica procesul, iar valorile pixelilor vor fi in intervalul [0, 1]. În plus, culorile imaginii vor fi inversate, astfel încât alb să devină negru și viceversa. Acest pas este necesar deoarece, în grafica pe calculator, valoarea 1 reprezintă alb, dar dorim ca în acest context valoarea 1 să indice o linie desenată cu negru.

$$Y = \text{Versiunea aplatizată pe rânduri și preprocesată a imaginii. } Y \in [0,1]^{m \cdot n} \subset \mathbb{R}^{m \cdot n}$$

$$p = \text{Numărul de cuie folosite}$$

$$l = \binom{p}{2}, \text{ numărul de linii totale care pot fi trase}$$

$$M_i = \text{O matrice care reprezintă a } i\text{-a linie desenată pe tablou}$$

$M =$ Pentru fiecare M_i, coloana corespunzătoare este versiunea aplatizată pe rânduri a M_i. $M \in \mathbb{R}^{m \cdot n,\, l}$

Vectorul coloană X care este output-ul algoritmului nostru, unde fiecare element din el, x_i reprezintă dacă linia i va fi trasată sau nu.

$$x_i = \begin{cases} 1 & \text{dacă linia } i \text{ este trasată} \\ 0 & \text{altfel} \end{cases}, \ X \in \mathbb{R}^l$$

În unele implementări ale metodelor, x_i va putea lua si valori între [0, 1] pentru a simplifica problema.

2.2 Definirea obiectivului

Astfel, obiectivul nostru final este să aflăm valorile elementelor vectorului X care ne aduc cel mai aproape de imaginea inițială. Deci trebuie să minimizăm:

$$\min \| \sum (M_i \cdot x_i) - y \|$$

sau

$$\min \| M \cdot X - Y \|$$

Este important de menționat că există o discrepanță între modul în care calculatorul percepe o imagine calitativă și modul în care acesta este percepută de ochiul uman. În formula de mai sus, calculatorul va considera că imaginea cu cea mai mică eroare este cea de cea mai înaltă calitate.

i03_24_5, MOS=0.4, FSIMc=0.693 i03_13_4, MOS=2.9, FSIMc=0.671

Fig. 27. Example of contradiction between FSIMc and MOS for the test image # 25.

Figura 2.1: Exemplu din lucrarea [1] care evidențiază discrepanța între erorile computate de calculator și calitatea percepută.

Acest fenomen poate fi observat în imaginea de mai sus 2.1, unde poza din stânga, deși are un scor FSIMc [5] mai mare (un scor mai mare indicând teoretic o calitate

mai bună) scorul MOS [1] este mai mic. Acest lucru înseamnă că majoritatea oamenilor au perceput imaginea din dreapta ca fiind mai clară și mai calitativă, contrar evaluării calculate algoritmic. Trebuie menționat că FSIMc este o metodă mai avansată decât MSE (folosită în această lucrare) și, cu toate acestea, poate să dea greș în evaluarea calității.

2.3 Plasarea cuielor

Pentru a plasa cuiele echidistant unul față de celălalt pe o formă circulară, am utilizat ecuația parametrică a cercului.

Pentru un cerc centrat in (x_c, y_c) cu rază r și p cuie:

$$x_k = x_c + r \cdot \cos \frac{2\pi k}{p}$$

$$y_k = y_c + r \cdot \sin \frac{2\pi k}{p}$$

$$\text{for } k = 0, 1, 2, ..., p-1$$

2.4 Trasarea liniilor

Pentru a trasa linii în reprezentarea matricială, algoritmul Bresenham [2] a fost folosit, determinând punctele necesare care să fie selectate între două cuie alese pentru a avea o reprezentare apropriată de o linie dreaptă desenată.

2.5 Obținerea rezultatului

După obținerea vectorului X, imaginea finală este generată înmulțind matricea de linii cu vectorul coloană X, rezultând combinațiile de linii care trebuie trasate: $O = MX$. Totuși, valorile din O pot depăși intervalul $[0, 1]$. Pentru a rezolva acestă problemă trebuie introdusă o funcție de limitare [4]. $C = \mathbb{R}^+ \to [0, 1]$. $C(x) = \max(0, \min(x, 1))$. În plus, pentru a desena imaginea cu culoarea neagră, trebuie inversate valorile culorilor. În final, acestea sunt scalate în intervalul $[0, 255]$, specific imaginilor digitale. Formula completă devine:

$$O = (1 - C(MX)) \cdot 255$$

[1] Pentru detalii suplimentare despre Scorul Mediu de Opinie (MOS), consultați articolul dedicat Mean Opinion Score (MOS)

[2] Pentru implementarea detaliată a algoritmului, consultați articolul despre Bresenham's Line Algorithm

Capitolul 3

Metodologie

3.1 Metoda celor mai mici pătrate (CMMP)

O primă metodă, care a fost inspirată din videoclipul [1] este cea folosind **CMMP**.

Metoda CMMP minimizează funcția menționată mai sus: $\min \|M \cdot X - Y\|$ prin vectorul X astfel încât norma l_2 dintre imaginea originală și cea rezultată să fie minimă. Vectorul X conține valori cuprinse între 0 și 1, indicând intensitatea sau măsura în care fiecare linie este trasată.

Metoda CMMP poate fi implementată prin două abordări.

3.1.1 Reprezentare densă a matricei

În acestă abordare, matricea M este reprezentată ca o matrice densă (plină), cu dimensiunile: $M \in \mathbb{R}^{m \cdot n,\ l}$. Adică are aceeași interpretare ca cea oferită in secțiunea 2.1.

Deși această abordare este simplă și ușor de înțeles, reprezentarea densă este mult mai costisitoare din punct de vedere computațional pentru matrici mai mari, deoarece fiecare element, incluzând și zero-urile, este stocat explicit. În consecință, acest lucru duce la o utilizare mai mare a memoriei cât și a unui timp ridicat de computare.

[1] Inspirația pentru această metodă poate fi găsită în videoclipul The Mathematics of String Art

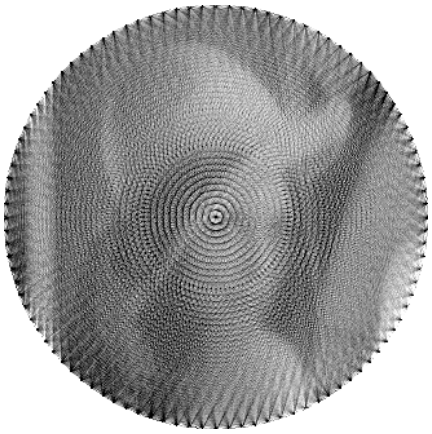

Figura 3.1: Imagine rezultată în urma procesului de CMMP cu reprezentare densă a matricii.

3.1.2 Reprezentare rară a matricei

A doua abordare este cea cu reprezentare rară a matricilor [2].

Știind faptul că vectorii noștri coloană din reprezentarea densă a matricei constituie o matrice aplatizată pe rânduri în care o singură linie a fost trasă, atunci putem spune că matricea este una rară, însemnând că majoritatae elementelor sunt zero.

De exemplu, cel mult $\max(m, n)$ elemente ale matricei M_i au valori diferite de 0, datorită algoritmului Bresenham. Observați imaginea de mai jos.

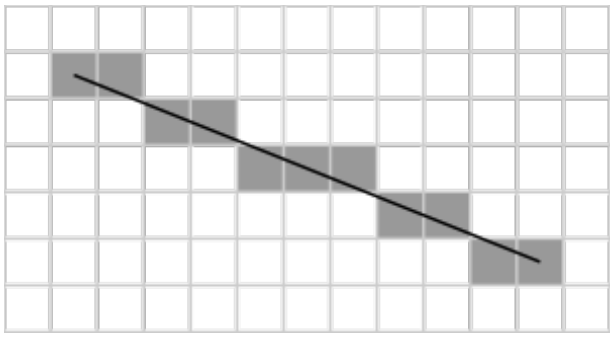

Figura 3.2: Imagine care să demonstreze modul de desenare al algoritmului Bresenham.

În exemplul de mai sus, unde $m = 5,\ n = 1$, linia trasată are exact 11 elemente diferite de zero.

În acest caz, algoritmii CMMP au implementări speciale [3] care să profite de structura rară a matricii. Sunt mai rapizi și folosesc mai puțină memorie pentru că procesează doar elementele diferite de zero.

[2]Pentru detalii despre reprezentarea rară a matricilor, consultați Sparse Matrix

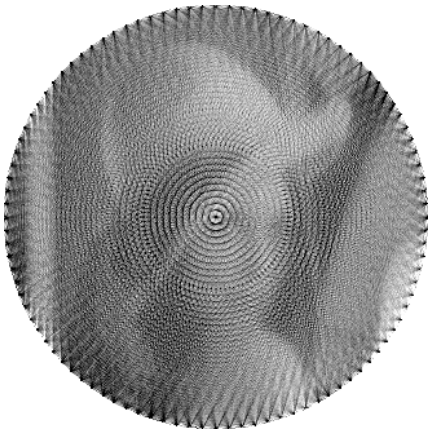

Figura 3.3: Imagine rezultată în urma procesului de CMMP cu reprezentare rară a matricii.

3.2 Metoda Greedy

Metoda Greedy constă în alegerea unei linii la fiecare pas. Această alegere este făcută astfel încât să minimizeze eroarea dintre imaginea originală si imaginea curentă la pasul k.

Dacă urmăm definiția greedy clasică o mică observație ar fi că odată aleasă o linie, alegerea este ireversabilă, adică nu ne putem întoarce să reparăm „greșeli". Dar, având în vedere modul în care o alegere este considerată bună sau rea (pe baza minimizării erorii), putem alege la fiecare pas să scoatem o linie dacă minimizează eroarea.

3.2.1 Euristici

Având în vedere timpul de computare mare, cauzat de necesitatea de a rezolva o problemă de tip CMMP la fiecare pas iterativ, am decis să integrez două euristici care să selecteze un subset al liniilor candidat. O linie este considerată candidat doar dacă nu a fost desenată încă.

1. Euristică Random

   Procesul este simplu, se selectează random k linii candidat.

2. Euristică Produs Scalar

   În acest caz se computează MY și se aleg primele k linii care nu au fost desenate încă și au cel mai mare produs scalar.

3.2.2 Pseudocod

Mai jos poate fi observat pseudocodul folosit în implementare. De menționat că algoritmul se va opri prematur dacă eroarea stagnează (nu este mai mică decât cea de la pasul anterior).

Algorithm 1 Greedy Algorithm

$M \leftarrow$ setul de valori conform terminologiei specificate
$Y \leftarrow$ imaginea aplatizată pe rânduri și preprocesată
$nl \leftarrow$ numărul de linii de selectat (input)

while $nl > 0$ **do**
    $linii \leftarrow$ liniile nedesenate încă
    $linii \leftarrow$ euristica_greedy(linii, k) ▷ selectează random k *linii* sau primele k în ordine descrescătoare a produsului scalar cu Y
    $j \leftarrow 0$ ▷ indexul liniei candidat care reduce cel mai mult eroarea

    for $i = 1$ to $len(l)$ **do**
        if $\|M(X + linii_i) - Y\| < \|M(X + linii_j) - Y\|$ **then**
            $j \leftarrow i$
        end if
    end for

    if $\|M(X + linii_j) - Y\| >= \|MX - Y\|$ **then**
        break
    end if

    $X \leftarrow X + linii_j$
    $nl \leftarrow nl - 1$
end while

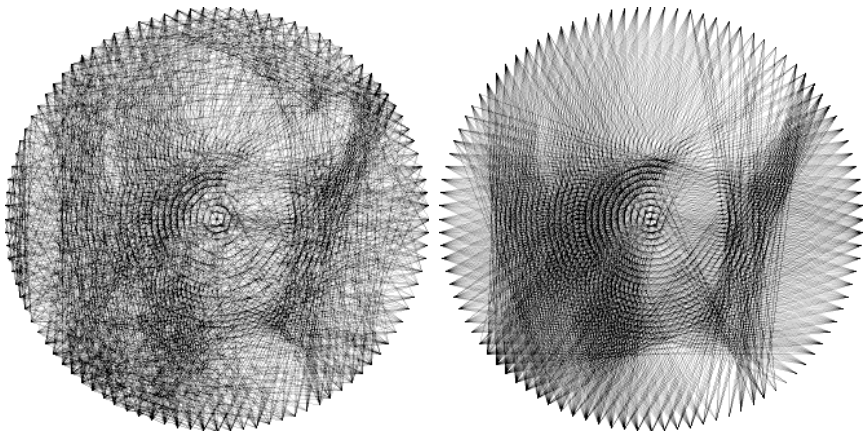

Figura 3.4: Imagini rezultate în urma procesului de Greedy cu euristică random (stânga) și euristică produsc scalar (dreapta).

Capitolul 4

Evaluarea Performanței și Analiză

Figura 4.1: Imaginea originală.

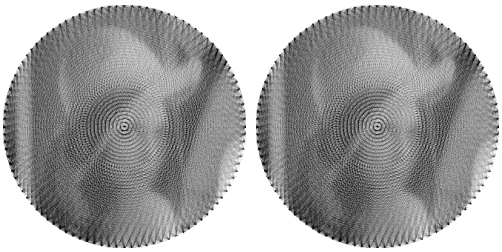

Figura 4.2: Imagini rezultate în urma procesului de CMMP cu reprezentări matriciale dense (stânga) și rare (dreapta).

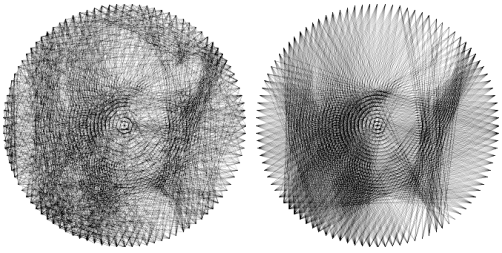

Figura 4.3: Imagini rezultate în urma procesului de Greedy cu euristică random (stânga) și euristică produs scalar (dreapta).

4.0.1 Observații vizuale

Se poate observa încă de la început că cele mai bune rezultate pentru ochiul uman sunt cele care folosesc metoda CMMP. Acest lucru se întâmplă din cauză că ele utilizeaza toate liniile posibile și ajustează intensitatea lor între 0 (alb) și 1 (negru). **Obs.** valorile sunt inversate pentru a putea desena linii negre în loc de albe. În grafica pe calculator, 0 reprezintă negru și 1 alb.

Pe de altă parte, abordările Greedy, care selectează un număr limitat de linii (în acest caz, 1000), produc reconstrucții mai puțin netede. Euristica random (stânga) tinde să fie mai haotică, alegând linii din diferite regiuni ale imaginii. În schimb, euristica produsului scalar (dreapta) se concentrează pe anumite zone, ducând la rezultate mai localizate, dar mai puțin variate.

Creșterea numărului de linii selectate în abordările Greedy ar putea îmbunătăți rezultatele, dar există un compromis legat de costul computațional ridicat.

4.0.2 Imagini Diferență

Imaginile diferență evidențiază discrepanțele dintre imaginea originală și rezultatul fiecărei metode.

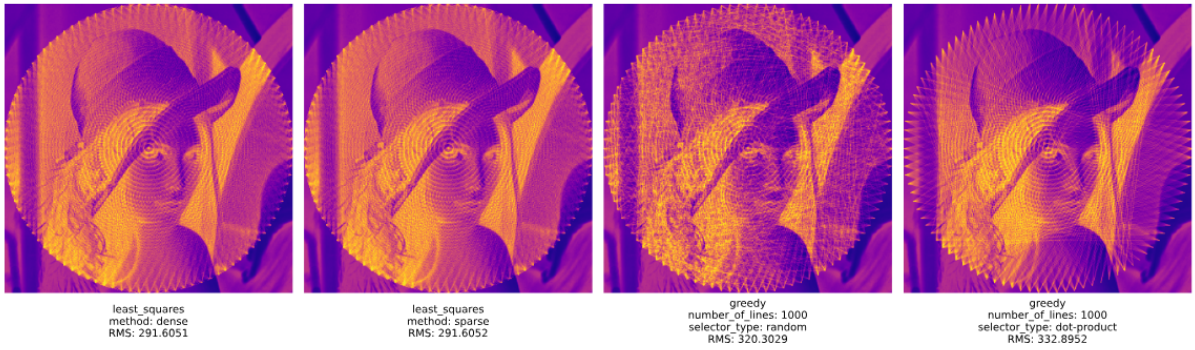

Figura 4.4: Diferența imaginilor rezultate până la imaginea originală.

Așa cum putem observa, imaginile diferență arată destul de asemănător cu imaginile rezultate, deoarece imaginea originală are mai multe nuanțe de gri, iar algoritmul poate selecta doar o valoare consistentă pe fiecare linie. Cel mai bun efect poate fi observat în penultima imagine (abordarea Greedy cu euristica random).

Rezultatele ar putea fi îmbunătățite prin alegerea unei imagini de input cu un contrast mai mare sau prin integrarea unei metode de preprocesare a imaginii (cum ar fi ajustarea contrastului sau comprimarea/extinderea intervalului de luminozitate) înainte.

Abordările CMMP au produs cele mai mici erori. Experimentul ar trebui să fie încercat și cu un număr crescut de linii pentru abordările Greedy, deoarece acest lucru poate

crește acuratețea. Un alt experiment interesant ar fi să se permită scăderea liniilor. Acest experiment funcționează pentru ambele abordări.

4.0.3 Timp de computare

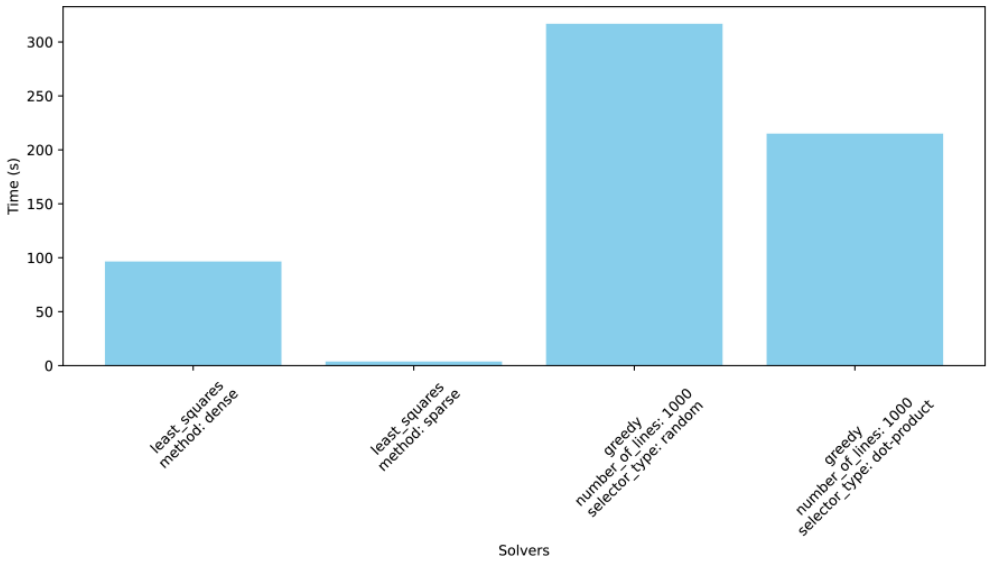

Figura 4.5: Imagine care ilustrează timpul necesar pentru computarea rezultatului în funcție de fiecare metodă.

În ceea ce privește timpul de computare, metoda CMMP rară este cea mai eficientă, finalizându-se în doar câteva secunde. Metoda CMMP densă, deși necesită mai mult timp, rămâne relativ rapidă în comparație cu algoritmii greedy.

Metodele Greedy, în special euristica random, prezintă cele mai lungi timpuri de computare. Acest lucru se datorează faptului că acestea trebuie să calculeze repetat soluția de tip CMMP pentru fiecare linie candidat și să o selecteze pe cea care minimizează eroarea la fiecare pas. Se observă că euristica produs scalar este cu câteva minute mai rapidă decât euristica random. Acest lucru se datorează faptului că produsul scalar între liniile candidat și imaginea originală este calculat o singură dată, în timp ce procesul de selecție aleatorie este calculat de fiecare dată când calculăm eroarea (la fiecare pas și pentru fiecare linie candidat).

4.0.4 Utilizarea maximă a memoriei

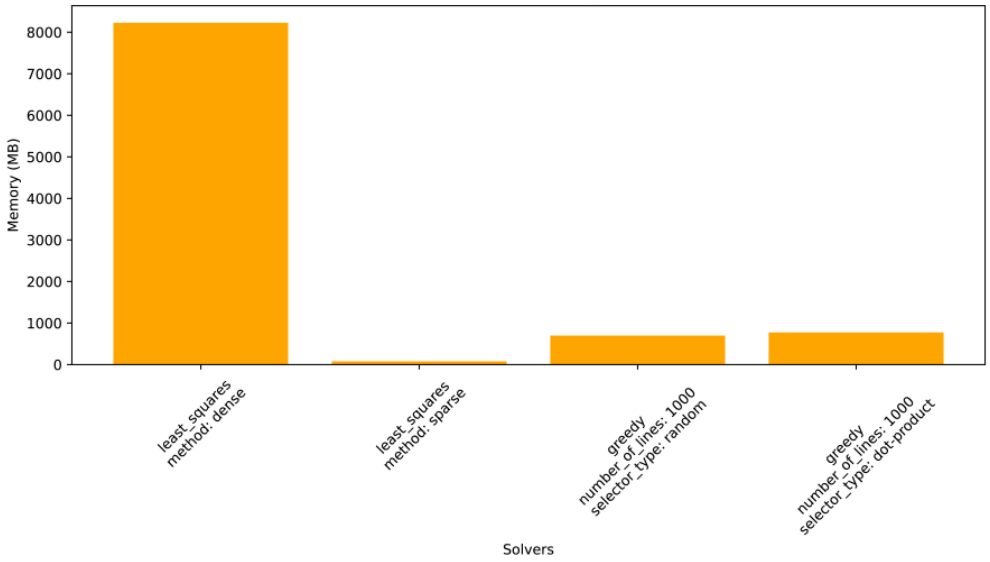

Figura 4.6: Imagine care ilustrează memoria maximă utilizată în timpul computării rezultatului în funcție de fiecare metodă.

Utilizarea maximă a memoriei oferă o perspectivă interesantă asupra eficienței algoritmilor. Toate metodele, cu excepția abordării CMMP densă, utilizează mai puțin de 1GB de memorie.

Metoda densă necesită mult mai multă memorie deoarece trebuie să stocheze o matrice mare de dimensiune: $m = 108.900,\ n = 4.950$. În schimb, metoda rară trebuie doar să stocheze elementele nenule, reprezentând o soluție mai eficientă din punct de vedere al memoriei. Iar abordările Greedy păstrează, în cel mai rău caz, o matrice de dimensiune: $m = 108.900, n = 1000$. Unde 1000 e numarul de linii care trebuie desenate.

Capitolul 5

Concluzii

Această analiză evidențiază trade-off-urile între precizie, timp de calcul și eficiență a memoriei.

Cea mai bună metodă de până acum este abordarea CMMP cu reprezentare rară a matricei. Aceasta se remarcă prin mai multe motive:

Timp de computare cel mai scăzut: Este cea mai rapidă metodă, obținând rezultatul în doar câteva secunde.

Utilizare memorie minimă: Metoda CMMP rară stochează doar elementele nenule, fiind eficientă pentru probleme mari.

Cel mai mic RMS: Oferă cea mai mică eroare RMS, demonstrând o precizie ridicată în aproximarea imaginii originale.

Cea mai bună calitate subiectivă a imaginii: Metoda CMMP rară obține o reconstrucție mai precisă, având o calitate superioară pentru ochiul uman.

În general, abordarea CMMP cu reprezentare rară a matricei atinge un echilibru optim între performanță și eficiență, făcând-o soluția cea mai fiabilă în această evaluare de performanță.

Bibliografie

[1] N. Ponomarenko et al., „Image database TID2013: Peculiarities, results and perspectives", în *ScienceDirect* (2015), p. 76, URL: https://www.ponomarenko.info/papers/tid2013.pdf.

[2] Michael Birsak, Florian Rist, Peter Wonka și Przemyslaw Musialski, „String Art: Towards Computational Fabrication of String Images", în *ResearchGate* (2018), URL: https://www.researchgate.net/publication/322766118_String_Art_Towards_Computational_Fabrication_of_String_Images.

[3] *csr_matrix*, Accessed: 2025-01-16, URL: https://docs.scipy.org/doc/scipy-1.15.0/reference/generated/scipy.sparse.csr_matrix.html.

[4] Ruben Socha, „Computational String Art", în *KTH* (2024), p. 5, URL: https://www.diva-portal.org/smash/get/diva2:1880384/FULLTEXT01.pdf.

[5] Lin Zhang, Xuanqin Mou și David Zhang, „FSIM: A Feature Similarity Index for Image Quality Assessment", în *PolyU* (), URL: https://www4.comp.polyu.edu.hk/~cslzhang/IQA/TIP_IQA_FSIM.pdf.