Temă la Analiza Algoritmilor
Flux Maxim

Călugăriţoiu Ion-Victor 322CA

Facultatea de Automatică şi Calculatoare
Universitatea POLITEHNICA Bucureşti

ion.calugaritoiu@stud.acs.upb.ro

1 Introducere

1.1 Descrierea problemei rezolvate

În cadrul unei reţele, modelarea unui proces de transport dintre un producător şi un consumator poate fi realizată utilizând un graf orientat. Cantitatea produsă pleacă dintr-un nod sursă şi trebuie sa ajungă la un nod destinaţie. Fluxul transmis prin fiecare muchie trebuie să fie mai mic decât o capacitate maximă asociată laturii. Astfel, considerând ca în momentul iniţial fluxul de pe fiecare muchie este 0, ne propunem să determinăm cantitatea de flux maxim care poate fi transmisă in reţea.

1.2 Aplicaţii practice ale problemei

Diverse probleme pot fi modelate utilizând aceste reţele de transport (grafuri orientate ale căror muchii au o capacitate). Spre exemplu: reţelele de transport (funcţionarea trenurilor/autobuzelor - administrarea corectă a cantităţii de trafic dirijat spre o anumită zonă), reţeaua de distribuire a apei (optimizarea cantităţii de apă care este trimisă într-o anumită arie a unui oraş - în cazul în care nu este realizată corect, apa curge pe distanţe mai mari decât necesar), transmiterea datelor dintr-o reţea de calculatoare (administrarea solicitarilor catre un server), etc.

1.3 Specificarea soluţiilor alese

Pentru a rezolva problema fluxului maxim, se pot utiliza o multitudine de algoritmi, dintre care am ales următorii 3:

Algoritmul Ford-Fulkerson (1956 - L. R. Ford Jr. şi D. R. Fulkerson): reprezintă primul algoritm dezvoltat pentru rezolvarea problemei de flux-maxim, fiind şi cel mai simplu.

Algoritm de tip Push-Relabel (1986 - Andrew V. Goldberg și Robert Tarjan): este considerat ca fiind unul dintre cei mai eficienți algoritmi pentru determinarea de flux-maxim.

Algoritmul Malhotra-Kumar-Maheshwari (MKM) (1978 - V.M. Malhotra, M.P. Kumar și S.N. Maheshwari); este un algoritm bazat pe algoritmul lui Dinic.

1.4 Criteriile de evaluare pentru soluția propusă

Pentru a verifica performanțele fiecărui algoritm, vor fi implementați și rulați pe același set de teste. Testele au fost realizate generând aleator grafuri orientate ale căror muchii au o capacitate asociată. Grafurile vor avea dimensiuni variabile, respectiv dimensiuni variabile relative între numărul de noduri și numărul de muchii ($M = NlogN$, $M = N^2$, $M = N\sqrt{N}$, $M >> N$). O condiție suplimentară pentru a menține proprietatea de rețea de flux este ca în nodul sursă (nodul 1) să nu intre flux, iar din nodul destinație (nodul n) să nu plece flux.

2 Prezentarea soluțiilor

2.1 Descrierea modului de funcționare al algoritmilor

Ford-Fulkerson algorithm:
Este un algoritm iterativ, constă în identificarea succesivă a unor drumuri de la sursă la destinație, drumuri ale căror muchii nu sunt la capacitate maximă de flux (drum de ameliorare). În diverse publicații este menționat ca fiind o **metodă**, nu un algoritm, datorită faptului că obținerea drumurilor de ameliorare nu este specificată concret. Este de tip greedy, și este garantat să ajungă la un rezultat final doar dacă valorile fluxurilor de pe muchii sunt raționale.

Push-Relabel algorithm:
Este bazat pe conceptul de "preflux" (permite fluxului să adune exces în unele noduri) și prezintă 2 operații - "push" - care împinge cât de mult flux se poate printr-o muchie; și "relabel" - care crește "înălțimea" unui nod dacă are exces de flux. În timpul execuției, algoritmul menține un preflux care este gradual transformat în flux maxim prin relocarea fluxului unui nod către nodurile sale vecine(push).

Malhotra, Kumar, Maheshwari (MKM) algorithm:
Este o versiune generalizată a algoritmului lui Dinic, însă diferă prin modalitatea în care se obține fluxul de blocaj ("blocking flow"), folosind conceptul de flux maxim pe care-l poate filtra un nod ("throughput").

2.2 Analiza complexității soluțiilor

Pentru următoarele complexități avem notațiile:
E = numărul de muchii
V = numărul de noduri
$fmax$ = valoarea maximă de flux

Ford-Fulkerson algorithm:
Algoritmul are complexitatea: $O(E|fmax|)$.

Pentru a obține un algoritm a cărui complexitate este independentă de valoarea fluxului maxim, se folosește o căutare în adâncime(BFS) în graf pentru determinarea drumurilor de ameliorare. Algoritmul obținut se numește Edmonds-Karp și are complexitatea $O(VE^2)$.

Push-Relabel algorithm:
Algoritmul de tip Push-Relabel generic are complexitatea: $O(V^2E)$. Există diverse implementări ale algoritmului care îmbunătățesc complexitatea sa. De exemplu, algoritmul implementat este Relabel-to-front, care rulează în $O(V^3)$.

Malhotra, Kumar, Maheshwari (MKM) algorithm:
Algoritmul are complexitatea: $O(V^3)$, fiind mai rapid decat algoritmul pe care este bazat (algoritmul lui Dinic - $O(V^2E)$).

2.3 Prezentarea principalelor avantaje și dezavantaje pentru soluțiile luate în considerare

Se remarcă următoarele avantaje și dezavantaje în utilizarea celor 3 algoritmi:

Deși complexitatea sa este cea mai slabă, utilizarea algoritmului Ford-Fulkerson reprezintă o opțiune bună în majoritatea cazurilor triviale, în practică ajungând la rezultat în timp asemănător (sau chiar mai rapid) decât ceilalți 2 algoritmi. De asemenea, este cel mai simplu și ușor de implementat dintre cei 3.

Pentru rezolvarea fluxului maxim pe grafuri mai complexe, se recomandă utilizarea algoritmilor Push-Relabel sau MKM, având o complexitate mai bună, care în practică oferă o îmbunătățire considerabilă doar în anumite cazuri (după cum se observă din analiza testelor). Dezavantajul acestor algoritmi constă în faptul că sunt mai greu de implementat, complexitatea îmbunătățită (față de FF) fiind resimțită doar pe grafuri specifice.

3 Evaluare

3.1 Descrierea modalităţii de construire a setului de teste folosite pentru validare

Pentru a testa performanţele algoritmilor, am generat 4 seturi de teste cu număr de noduri şi muchii variate, în total însumând 50 de teste. Generarea de fişiere .in şi fişiere .ref a fost făcută folosind script-uri în Bash (obţinerea numărului de noduri şi muchii) şi un script în Python (generarea de grafuri orientate în funcţie de M şi N obţinute anterior).

Pentru **primul set** de 20 de teste am generat folosind funcţia "linspace" din MATLAB un vector de 20 de elemente egal depărtate, cu valorile cuprinse **între 2 şi 2500**. Fiecare element reprezintă numarul de noduri (N) din graful testului respectiv. **Primul set de teste are numărul de muchii (M) egal cu** $N * logN$. Astfel, am generat încă un vector de 20 de elemnte care pentru fiecare valoare N din vectorul iniţial conţine valoarea $N * logN$. După obţinerea numărului de noduri(N) şi de muchii(M) pentru fiecare test, folosind un script Bash sunt trimise N, M ca input unui program care genereaza aleator grafuri orientate (un script scris în Python).

Pentru **al doilea set** de 10 teste, grafurile au fost generate asemănător, însă vectorul de noduri conţine valori **între 2 şi 1186**, iar **vectorul de muchii conţine** $M = N\sqrt{N}$.

Pentru a genera **al treilea set** de 10 teste, am folosit acelaşi vector de noduri de la setul 2, însă numărul de muchii corespunzător fiecarui test a fost **generat aleator între valorile: N - 50 000.**, astfel obţinând

Pentru a genera **al patrulea set** de 10 teste, grafurile au fost generate asemănător, însă vectorul de noduri conţine valori **între 2-150**, iar **vectorul de muchii conţine** $M = N^2$.

După ce generarea celor 50 de grafuri este realizată, script-ul parcurge fiecare linie a fişierelor de test şi adaugă o valoare aleatoare de flux (cuprinsă între 1 şi 500000) la fiecare muchie (la finalul fiecărei linii, mai puţin prima). Astfel, input-ul respectă format-ul specificat în enunţ: pe prima linie - N, M; pe următoarele M linii - x, y, z (muchie de la nodul x la nodul y, cu valorea de flux z).

3.2 Specificaţiile sistemului de calcul pe care au fost rulate testele

Testarea a fost realizată pe o maşină virtuală care rulează Linux. Maşina virtuală a fost importată în software-ul VMWare Workstation 16 Pro.

Specificații complete:

Procesor: Intel(R) Core(TM) i7-10750H CPU @ (2.60GHz - 5.00 GHz) - mașina virtuală dispune de 2 nuclee (4 threads) din cele 6 (12 threads) disponibile pentru host.

Adaptor video: NVIDIA GeForce GTX 1660 Ti

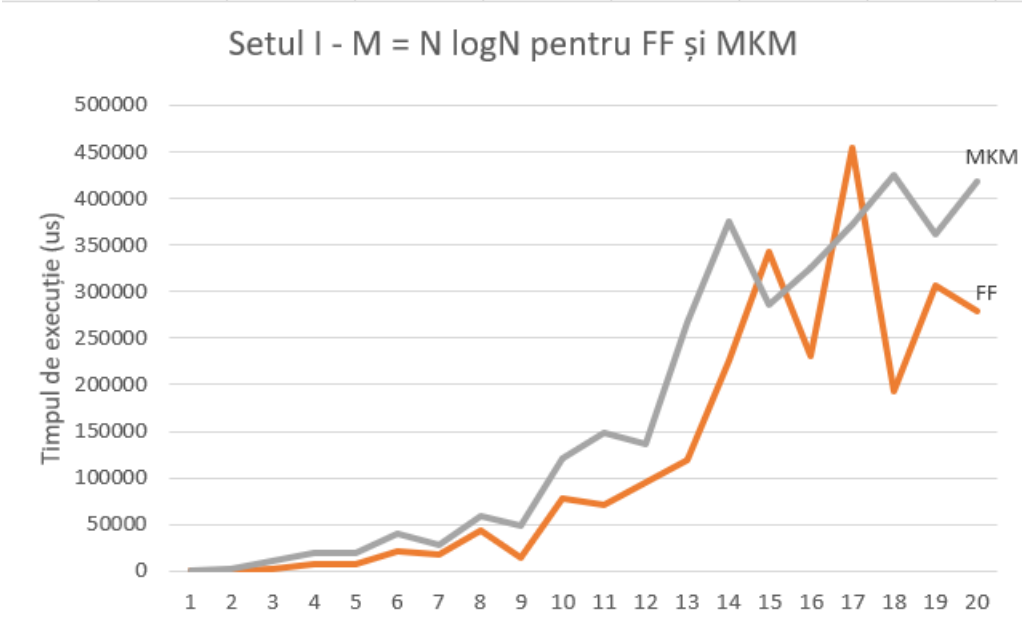
Memorie RAM: 4GB alocați pentru VM din cei 16GB disponibili (2933MHz DDR4 dual channel); în timpul rulării, memoria totală disponibilă pentru testare a fost de aprox. 3500MB, 500MB fiind rezervați pentru sistemul de operare.

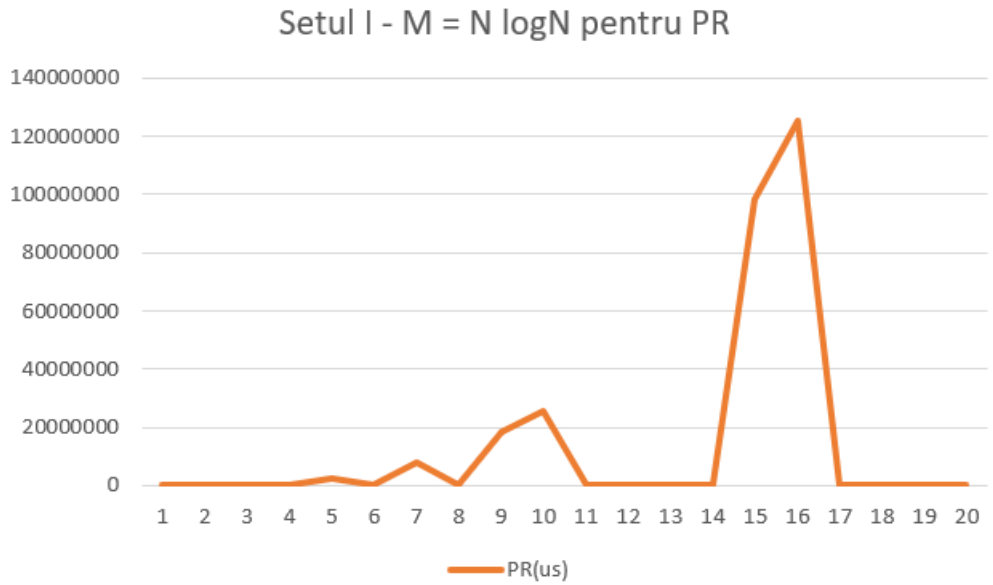
Model SSD: Western Digital PC SN730, mașina virtuală dispunând de 20GB de stocare.

Sistem de operare: Lubuntu (distro bazat pe Ubuntu 20.04.3 LTS)

3.3 Ilustrarea, folosind grafice/tabele, a rezultatelor evaluării soluţilor pe setul de teste

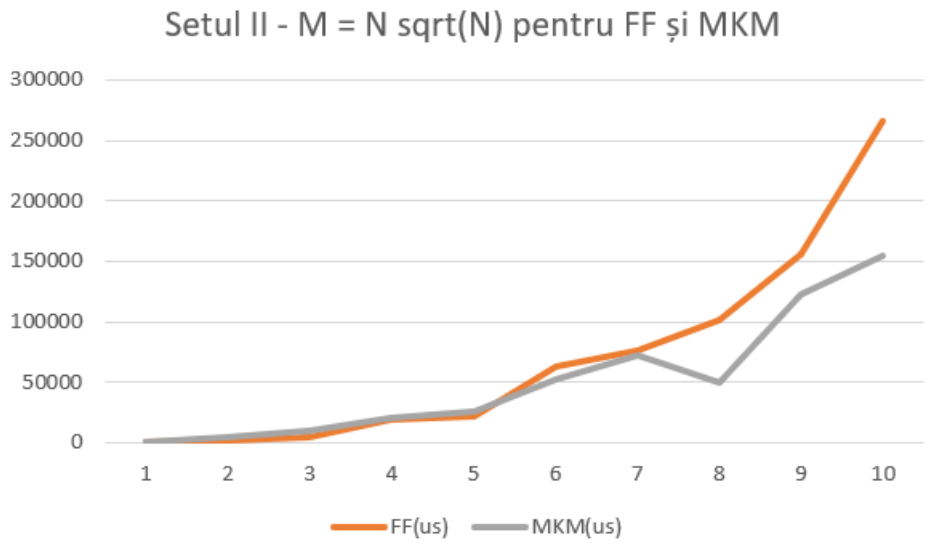
Test	N	M	Ford-Fulkerson	Push-Relabel	MKM
1	2	2	10	2	5
2	134	657	612	46577	1836
3	265	1479	2976	4583	11409
4	397	2376	6973	6908	18567
5	528	3311	7217	2294583	19931
6	660	4285	20233	22204	39302
7	791	5279	17596	7759416	27753
8	923	6302	43486	35514	58330
9	1054	7337	13615	18293894	49249
10	1186	8395	77574	25574032	121080
11	1317	9461	71724	73807	147579
12	1449	10547	95020	82143	135411
13	1580	11637	119425	110745	266936
14	1712	12747	225569	134500	374740
15	1843	13858	341959	98308810	285372
16	1975	14987	230706	125493804	325932
17	2106	16117	455028	239006	372094
18	2238	17263	193028	257127	425804
19	2369	18408	305895	339071	360867
20	2500	19561	278828	292395	417639

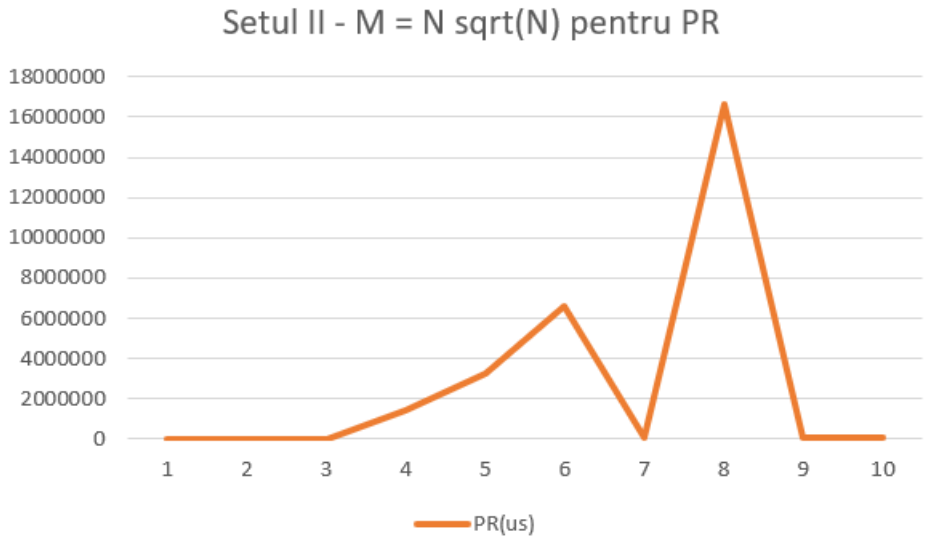




Setul II:

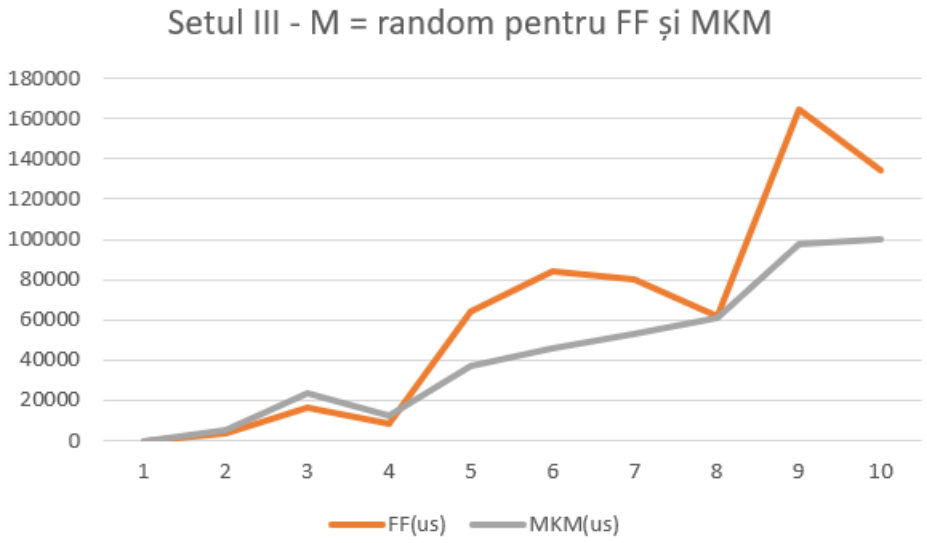
Test	N	M	Ford-Fulkerson	Push-Relabel	MKM
21	2	2	11	2	5
22	134	1552	1337	1332	4152
23	265	4314	4829	3801	9471
24	397	7911	18512	1421951	20777
25	528	12133	21242	3230183	25697
26	660	16956	62778	6565657	51841
27	791	22247	76354	38165	71955
28	923	28042	101794	16632751	50257
29	1054	34219	156250	49361	122675
30	1186	40844	266056	64143	154184

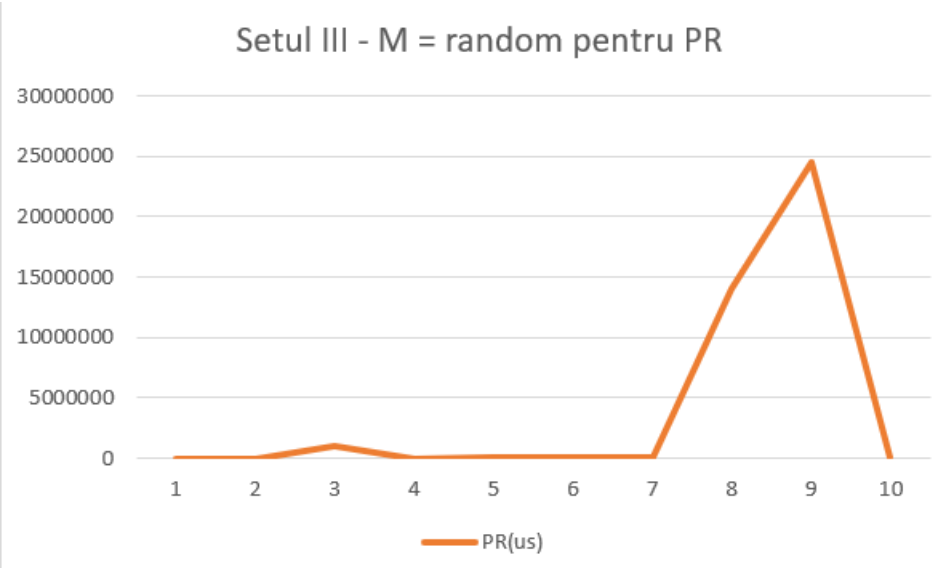

Setul II - M = N sqrt(N) pentru FF și MKM
300000
250000
200000
150000
100000
50000
0
1
2
3
4
5
6
7
8
9
10
FF(us)
MKM(us)


Setul II - M = N sqrt(N) pentru PR
18000000
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
1
2
3
4
5
6
7
8
9
10
PR(us)

Setul III:

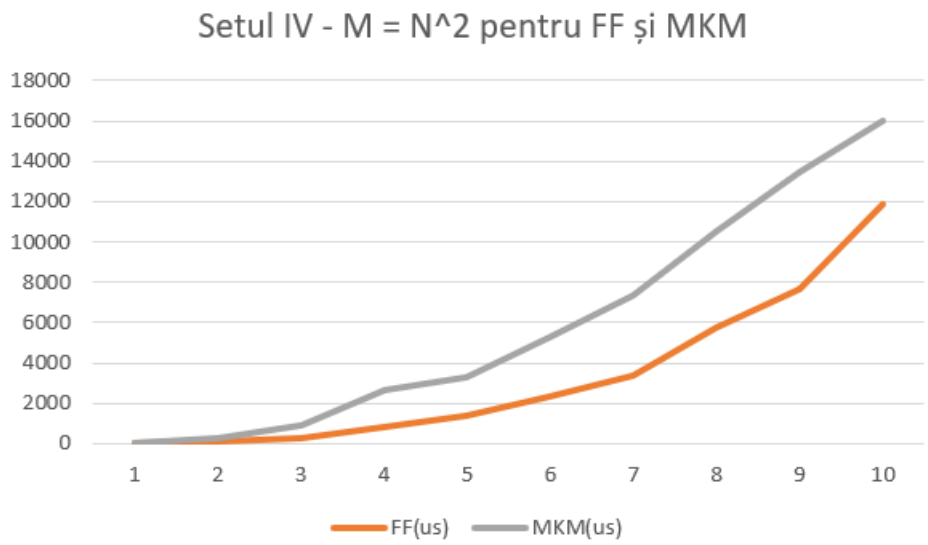
Test	N	M	Ford-Fulkerson	Push-Relabel	MKM
31	2	2	9	2	5
32	134	6319	3395	2430	5515
33	265	20446	16410	1016826	23827
34	397	5028	8356	6987	12770
35	528	32401	64243	18899	37074
36	660	31698	83868	23512	45790
37	791	19008	80246	26805	52792
38	923	15254	62160	13999411	61166
39	1054	38978	164563	24552079	97656
40	1186	24404	134508	55802	99897

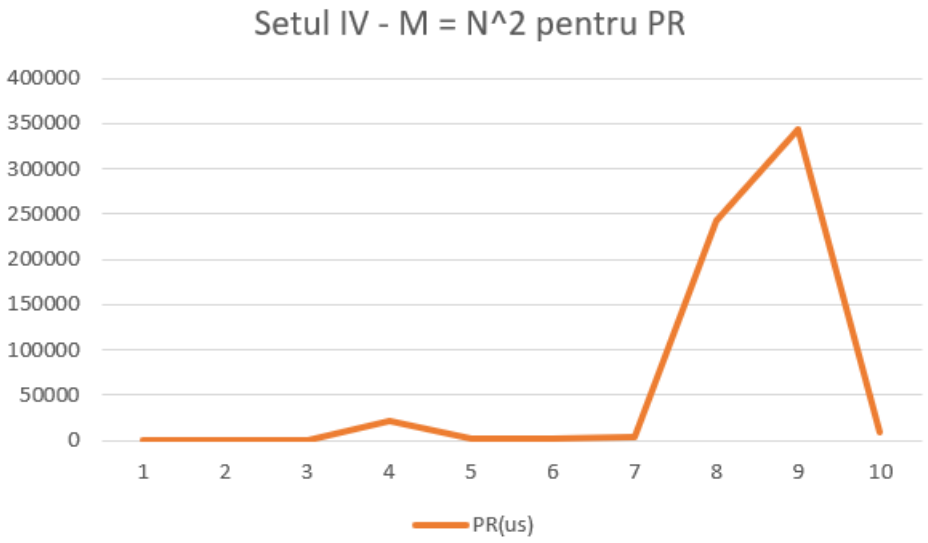




Setul IV:

Test	N	M	Ford-Fulkerson	Push-Relabel	MKM
41	2	2	9	2	5
42	19	342	78	1027	268
43	35	1190	267	453	906
44	52	2652	808	22422	2616
45	68	4556	1350	1656	3305
46	85	7140	2372	2710	5279
47	101	10100	3342	3652	7343
48	118	13806	5769	243090	10532
49	134	17822	7674	343379	13468
50	150	22350	11914	8757	15995





3.4 Prezentarea valorilor obținute pe teste

În urma testării, am observat următoarele particularități ale algoritmilor:

Pentru primul set de teste, în care $M = N * logN$, deși în teorie are o complexitate mai proastă, algoritmul Ford-Fulkerson ajunge mai rapid la un răspuns decât algoritmul MKM.

Pentru al doilea set, în care $M = N\sqrt{N}$, se observă faptul că pentru primele 5 teste algoritmul MKM ajunge la rezultat în timp asemănător cu FF, însă, pen-

tru ultimele 5, MKM este mai rapid. Remarcăm că MKM este mai rapid pentru teste mai mari și mai complexe, cu număr mare de noduri și muchii.

Pentru setul III, în care M este ales aleator, observăm același comportament ca în cazul setului II. Pentru număr foarte mare de muchii, performanțele algoritmului FF sunt vizibil scăzute.

La setul IV, în care $M = N^2$ și N este cuprins între 2 și 150, algoritmul Ford-Fulkerson găsește rezultatul mai rapid decât MKM, fapt datorat numărului relativ mic de noduri comparativ cu celelalte 3 seturi de teste.

Comportamentul algoritmului de tip Push-Relabel a fost reprezentat separat față de ceilalți 2, deoarece rezultatele sale au fost imprevizibile. Pentru unele teste, algoritmul reușește să ajungă la rezultat într-un timp asemănător (sau chiar mai rapid) cu FF și MKM, dar pentru altele, timpul de execuție este neobișnuit de mare. Am considerat că algoritmul se comportă in acest mod fie datorită implementării sale, fie datorită faptului ca muchiile grafurilor sunt generate complet aleator, la unele teste în care numărul de noduri și muchii este foarte mare, execuția terminându-se mai rapid decât ne-am aștepta, și viceversa.

4 Concluzii

În concluzie, în urma analizei testelor, a graficelor și a timpilor de execuție, am remarcat faptul că pentru a rezolva o problemă de flux-maxim în care graful rețelei de flux este relativ simplu și există un număr mic de noduri și muchii, utilizarea algoritmului Ford-Fulkerson este alegerea corectă, date fiind performanțele bune obținute pe acest tip de graf. Pentru a rezolva un graf complex, cu un număr foarte mare de noduri și muchii, utilizarea algoritmului MKM este recomandată, datorită eficienței, fiabilității și predictibilității comportamentului său, spre deosebire de algoritmul Push-Relabel.

5 Bibliografie

1. Design & Analysis of Algorithms - Lecture #14 (Carnegie Mellon University) https://www.cs.cmu.edu/∼avrim/451f13/lectures/lect1010.pdf
2. "An Algorithmic Study of the Maximum Flow Problem: A Comparative Statistical Analysis" - Antonio Sedeño-Noda, Miguel A. González-Sierra, and Carlos González-Martín
3. https://cp-algorithms.com/graph/mpm.html - (ultima accesare 02.11.2021)
4. "An O($|V|^3$) algorithm for finding maximum flows in networks" - V.M. Malhotra, M.P. Kumar, S.N. Maheshwari