Tema 2 ANALIZA ALGORITMILOR 2021-2022 KClique $\leq_p$ SAT

Gîrneț Andrei 321CB

Scopul:

Scopul proiectului este de a compara 2 metode de rezolvare a problemei KClique, una in timp exponențial, in acest proiect backtracking, si alta care implica o transformata polinomiala pentru intrarea SAT.

Backtracking:

- Daca deja sau trecut prin prea multe vârfuri clica nu e valida. Daca numărul de vârfuri parcurse este egal cu k, se verifica daca nodurile formează o clica.
- Trec prin toate nodurile si le adaug într-o clica posibila, la un timp anumit numărul de noduri va fi unul satisfăcător deci este o clica posibila

Reducere la SAT:

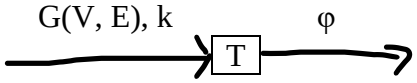
- Ideea transformării (convenție: Numele nodurilor sunt cifre consecutive începând de la 0 la (numarulDeNoduri – 1)) :
  - Așa cum orice nod poate participa într-o clica primele k clauze a SAT sunt formate din x[numeNod][numarPozitieInClica]:
    - i ∈ [0..(nrNoduri – 1)], j ∈[0..(k -1)]:
      $(x_{00} \vee x_{10} \vee x_{20} \vee ... \vee x_{i0} \vee ... \vee x_{(nrNoduri-1)0}) \wedge ... \wedge (x_{0j} \vee x_{1j} \vee x_{2j} \vee ... \vee x_{ij} \vee ... \vee x_{(nrNoduri-1)j}) \wedge ... \wedge (x_{0(k-1)} \vee x_{1(k-1)} \vee x_{2(k-1)} \vee ... \vee x_{i(k-1)} \vee x_{(nrNoduri-1)(k-1)})$
  - Daca un nod ar participa in clica atunci trebuie sa facem ca duplicatele lui sa nu participe si sa fie false, adică daca x_{00} e adevărat, ceea ce înseamnă ca participa in clica nodul 0 pe poziția 0 in clica, atunci x_{01}, x_{02}, x_{03}, ... x_{0k} trebuie sa fie false. Pentru a impune aceasta condiție știind ca xij e adevărat, impunem clauzele:
    - x_{ij} = true, x_{in} = false, n ∈ [0..(k -1)], n ≠ j:
      $(\sim x_{ij} \vee \sim x_{in})$

    Așa pentru i ∈ [0..(nrNoduri – 1)], j ∈[0..(k -1)]
  - Daca intre 2 noduri nu exista o muchie, deci aceste noduri nu pot apărea într-o clica, aceasta o impunem identic ca la cazul 2, doar ca de data aceasta metoda de selecție e daca nu exista muchie intre 2 noduri

  Complexitatea algoritmului in python $O(n^4 + n^3 + n^2)$

- KClique $\leq_p$ SAT

  KClique(G(V, E), k) = 1, daca $\exists E_1, E_2, E_3 ... E_k$ a. i. ele formează o clica.
  SAT(φ) = 1, daca φ este satisfiabila



a. i. KClique(G(V, E), k) = 1 ⇔ SAT(φ) = 1

```
def SAT(expresieBooleana):
        #Implementat
        return valoareBooleana
def KClique(graf, k):
        return SAT(T(graf, k))
def T(graf, k):
        #implementat in python in ./PolTimeRDC/solveKClique.py
        return expresieBooleana
```

Demonstrație KClique(G(V, E), k) = 1 ⇔ SAT(φ) = 1:

KClique(G(V, E), k) = 1 => $\exists E_1, E_2, E_3 \dots E_k$ a. i. ele formează o clica, deci in formula booleana x[nodClica][pozitieInClica] = true => primele k clauze sunt adevărate, daca formează o clica atunci si clauzele care verificau ca sa nu se ia un nod de 2 ori sunt toate adevărate, si toate clauzele care verificau ca in clica sa nu fie noduri care nu au muchie atunci si ele sunt adevărate => SAT(T(G(V, E), k)) = 1

KClique(G(V, E), k) = 0 => $\nexists E_1, E_2, E_3 \dots E_k$ a. i. ele formează o clica, deci formula booleana nu va putea fi satisfăcuta din 2 motive:

1. Nu se vor găsi noduri care sa formeze o muchie pentru a forma clica, deci formula ($\sim x_{ij}$ V $\sim x_{mn}$) nu va da adevărat sub nici o varianta
2. Ar implica utilizarea unui nod de 2 ori, însă in acest caz formula ($\sim x_{ij}$ V $\sim x_{in}$) nu va da adevărat sub nici o varianta

=> SAT(T(G(V, E), k)) = 0

Comparație dintre timpul de rezolvare:

- Categoria 1:
  - Observații:
    - Răspunsul tuturor testelor este false, ceea ce înseamnă ca la backtracking se vor trece prin toate cazurile posibile pentru a da răspunsul false.
    - Numărul nodurilor in graf este relativ mic, ce implica un număr mai mic de clauze pentru SAT, si formarea intrări pentru SAT este mai rapida
    - Număr relativ favorabil de noduri într-o clica, ceea ce ajuta atât la backtracking cat si la reducere
    - Număr mic de muchii, ceea ce înseamnă ca la verificare si parcurgere a grafului probabilitatea ca intre toate nodurile va exista o clica e mult mai mica

- Timp rulare si raportul timpurilor de rulare:

```
andrei@DESKTOP-HK88MDD:/mnt/d/UPB/Analiza Algoritmilor/Tema 2/Tema 2 v1.00$ ./check.sh category1
RUNNING BACKTRACKING category1
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
TOTAL: 4/4
BACKTRACKING TOTAL TIME: 5.644s

RUNNING REDUCTION category1
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
TOTAL: 4/4
REDUCTION TOTAL TIME: 1.258s

REDUCTION / BACKTRACKING: 0.222
```

- Concluzie:
  - Pentru algoritmul de backtracking aceste teste sunt mult mai costisitoare din punct de vedere al timpului așa cum implica încercarea tuturor posibilităților pentru a vedea ca răspunsul este false, care este cel mai rău caz pentru acest algoritm chiar si daca implica puține noduri si o clica de o dimensiune relativ mica, pe când pentru transformata nu are nici o importanta răspunsul ci doar numărul de noduri si mărimea clicii dorite.

- Categoria 2:
  - Observații:
    - Răspunsurile sunt in majoritate true cu un raport(True/False) de 5/1, ceea ce înseamnă ca algoritmul de backtracking se va finisa mai curând decât increarea tuturor posibilităților.
    - Număr mare de noduri, aproximativ 100 noduri pe test, ceea ce implica 100 * k variabile booleene si mult mai multe clauze, pe care transformata trebuie sa le creeze, si apoi SAT sa le evalueze, pentru algoritmul de backtracking atât timp cat da true depinde de cat de repede va găsi clica si de punctul de intrare.
    - Numărul de noduri in clica mic, aproximativ 3/test, ceea ce e foarte favorabil pentru algoritmul de backtracking si relativ favorabil pentru SAT
    - Număr mare de muchii, aproximativ 174/test, ceea ce implica o probabilitate mai mare ca intre 2 noduri alese aleatoriu va exista o muchie, pentru algoritmul de backtracking e foarte favorabil, iar pentru SAT implica mai puține clauze când formam clauzele care nu permit ca 2 noduri care nu au o muchie sa fie într-o clica.
  - Timp rulare si raportul timpurilor de rulare:

```
andrei@DESKTOP-HK88MDD:/mnt/d/UPB/Analiza Algoritmilor/Tema 2/Tema 2 v1.00$ ./check.sh category2
RUNNING BACKTRACKING category2
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
TOTAL: 6/6
BACKTRACKING TOTAL TIME: 0.292s

RUNNING REDUCTION category2
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
TOTAL: 6/6
REDUCTION TOTAL TIME: 22.767s

REDUCTION / BACKTRACKING: 77.969
```

- Concluzie:
  - In aceasta categorie testele sunt foarte complicate pentru metoda de rezolvare reducere, deoarece pentru ea nu prea contează ce răspuns trebuie sa dea, transformata face din intrarea problemei KClique in intrarea SAT, iar pentru un număr mare de noduri transformata are un număr mare de operații de efectuat, iar SAT are un număr mare de variabile de evaluat. Pe când pentru backtracking e foarte favorabil când răspunsul e true, numărul de noduri într-o clica e mic si numărul de muchii in graf e mare.

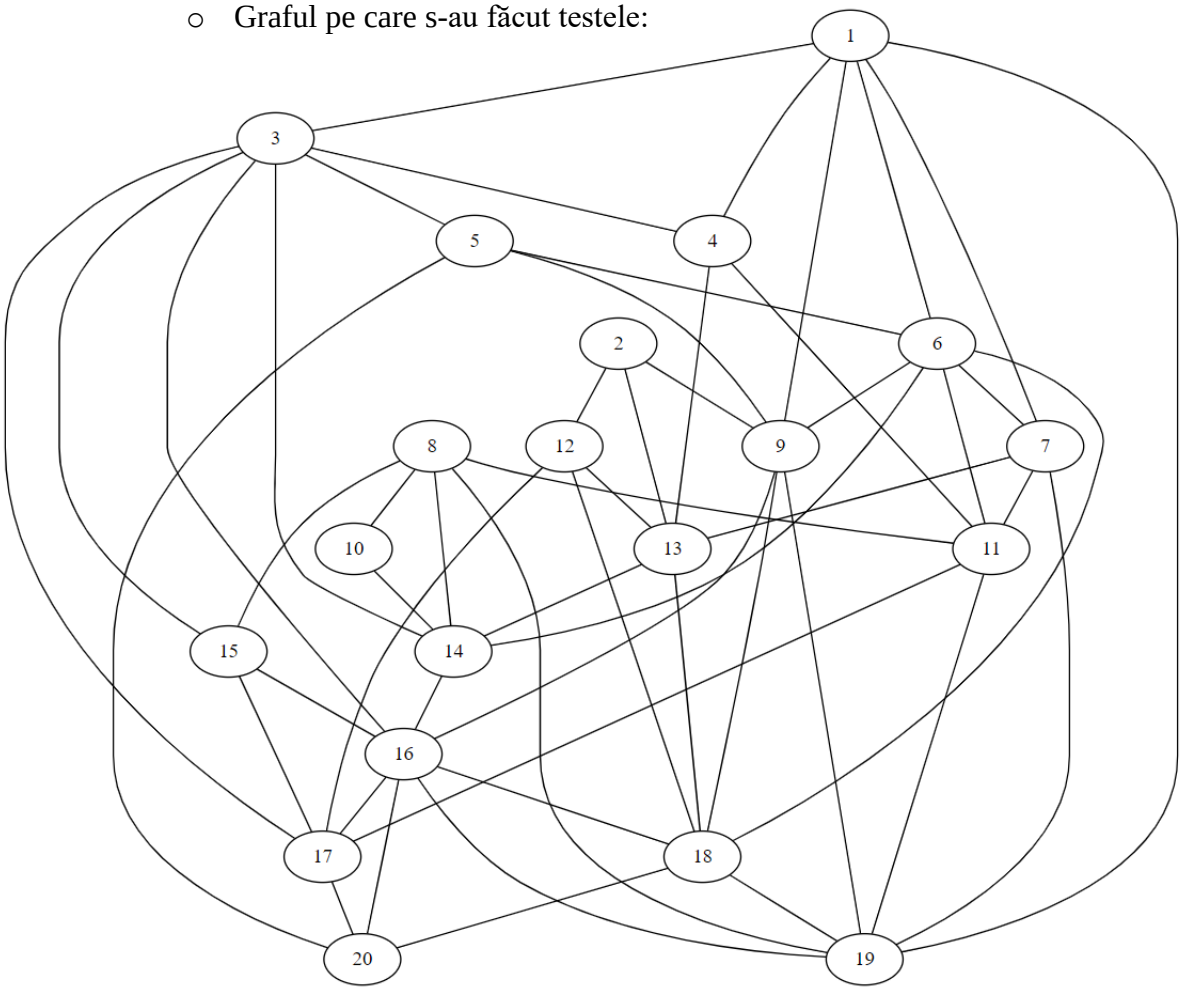
- Categoria 3:
  - Observații:
    - Testele sunt echilibrate din punct de vedere al rezultatului, raportul(True/False) este 1/1, ceea ce nici nu favorizează sau complica operațiile nici unui din algoritmi
    - Număr mic de noduri, aproximativ 7/test, care e favorabil cat pentru metoda cu reducerea la SAT cat si pentru backtracking.
    - Număr favorabil de muchii, aproximativ 13/test, număr favorabil pentru numărul de noduri in teste.
    - Numărul de noduri participante in clica mic, aproximativ 4/test, este favorabil pentru ambele algoritme
  - Timp rulare si raportul timpurilor de rulare:

```
andrei@DESKTOP-HK88MDD:/mnt/d/UPB/Analiza Algoritmilor/Tema 2/Tema 2 v1.00$ ./check.sh category3
RUNNING BACKTRACKING category3
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
[TEST6] - PASSED
[TEST7] - PASSED
[TEST8] - PASSED
[TEST9] - PASSED
[TEST10] - PASSED
TOTAL: 11/11
BACKTRACKING TOTAL TIME: 0.493s

RUNNING REDUCTION category3
[TEST0] - PASSED
[TEST1] - PASSED
[TEST2] - PASSED
[TEST3] - PASSED
[TEST4] - PASSED
[TEST5] - PASSED
[TEST6] - PASSED
[TEST7] - PASSED
[TEST8] - PASSED
[TEST9] - PASSED
[TEST10] - PASSED
TOTAL: 11/11
REDUCTION TOTAL TIME: 2.758s

REDUCTION / BACKTRACKING: 5.594
```

  - Concluzie:
    - Pentru acest set de teste algoritmul de backtracking e mai rapid, datorita faptului ca rezolvarea problemei are loc nemijlocit după citirea datelor, pe când algoritmul care implica reducere face o transformare care probabil nu e una foarte rapid si nici cea mai optimizata.

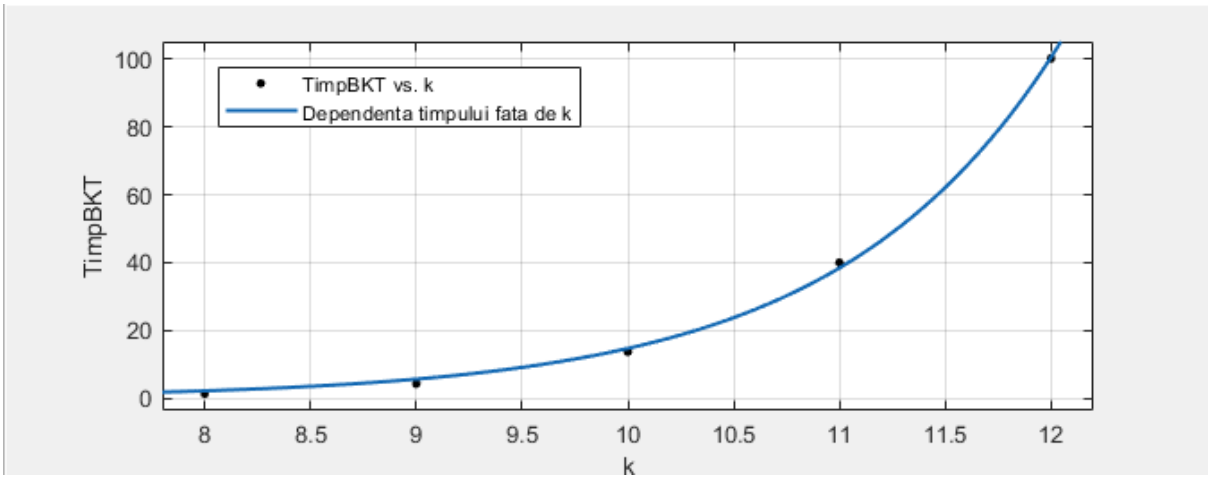
Bonus (Toate testele aveau răspunsul False):

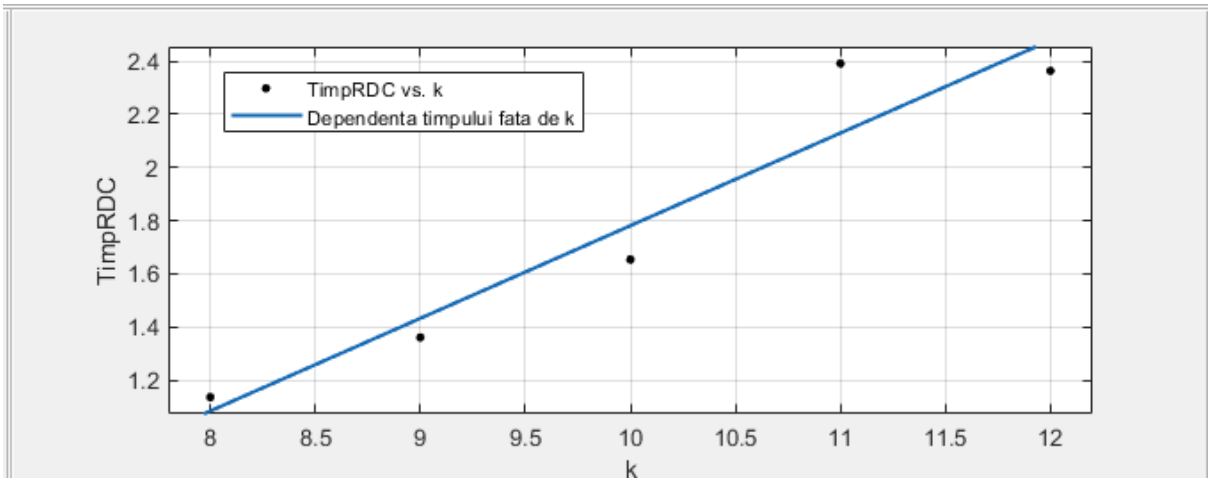
- **Fixez toate datele si modific doar mărimea la clica pe care o caut**
  - Graful pe care s-au făcut testele:



  - Dependenta timpului in funcție de k:

k	*Backtracking*	*Reduction*
8	1.284	1.136
9	4.247	1.361
10	13.671	1.654
11	39.980	2.392
12	100.305	2.364





- Aproximările făcute pentru a reprezenta grafic:
  - Backtracking

    Modelul funcției: a*exp(b*x)

    Coeficienți(cu limite de încredere 95%):

    a = 0.000966 (-0.0001041, 0.002036)

    b = 0.9628 (0.8693, 1.056)

  - Reduction

    Modelul funcției: p1*x + p2

    Coeficienți(cu limite de încredere 95%):

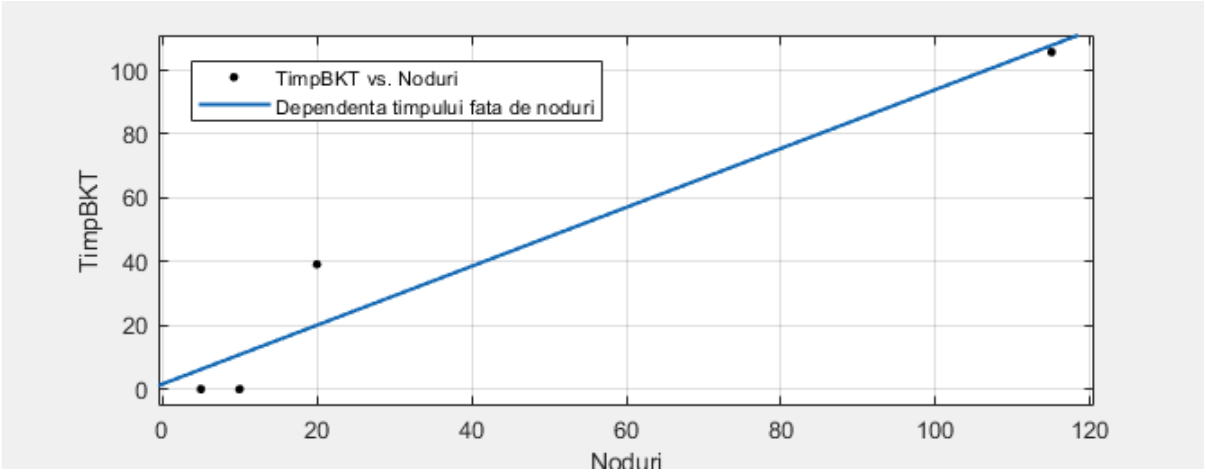
    p1 = 0.3487 (0.1597, 0.5377)

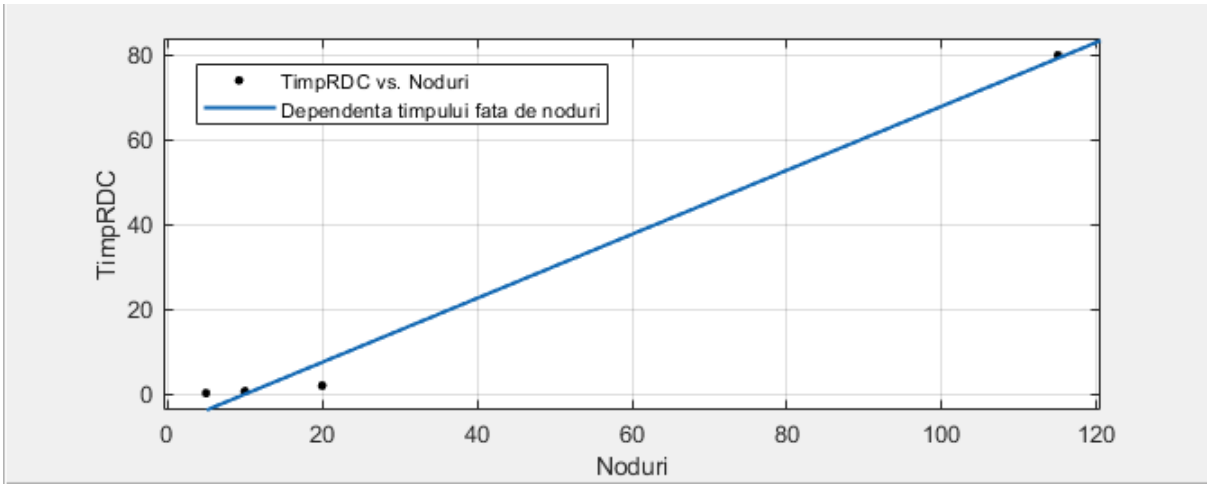
    p2 = -1.706 (-3.615, 0.2035)

- Concluzie
  - Algoritmul de backtracking are tendința de a mari timpul mult mai repede in comparație cu cel al reduceri la SAT, totul datorita faptului ca pana verifica toate posibilitățile, care pe lângă faptul ca sunt multe, sunt si mari când k e un număr mai mare, deci daca se intuiește ca răspunsul este False si k e un număr mai mare de 9 atunci se prefera algoritmul de reducere la SAT.

- **Fixez relativ toate intrările si modific numărul de noduri(k - număr mare):**
  - Dependenta timpului in dependenta de numărul de noduri(k – număr mare)

Nr. noduri	*Backtracking*	*Reduction*
5	0.045	0.228
10	0.041	0.699
20	39.183	1.990
115	105.738	79.903





- Aproximările făcute pentru a reprezenta grafic:
  - Backtracking

    Modelul funcției: p1*x + p2

    Coeficienți(cu limite de încredere 95%):

    p1 = 0.9239 (0.1511, 1.697)

    p2 = 1.607 (-43.7, 46.91)

  - Reduction

    Modelul funcției: p1*x + p2

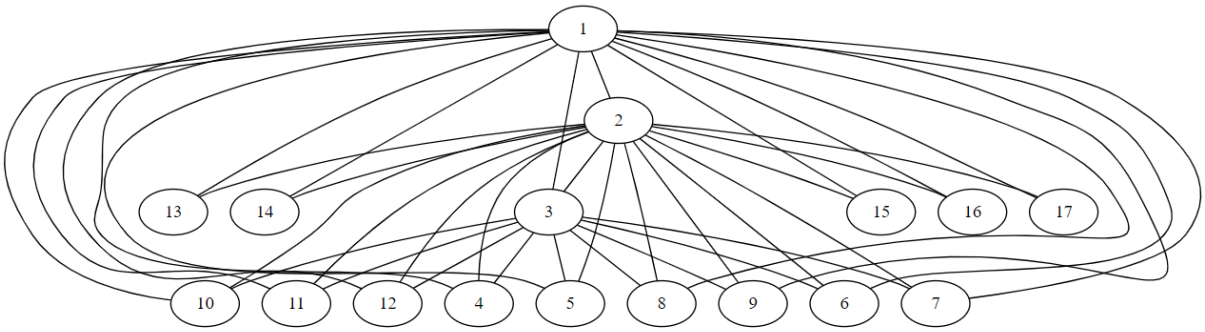
    Coeficienți(cu limite de încredere 95%):

    p1 = 0.7546 (0.5212, 0.988)

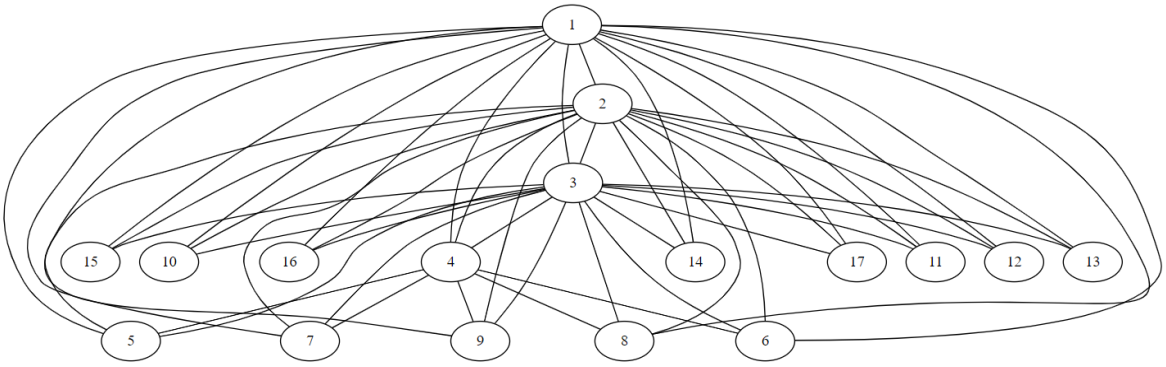
    p2 = -7.592 (-21.27, 6.091)

- Concluzie:
  - Putem observa aici ca la mărirea nodurilor, ambele algoritme au aproximativ același timp de rulare, însă oricum exista o diferența de aproximativ 30s in favoarea algoritmului de reducere la SAT, însă iarăși daca intuim ca răspunsul este False.

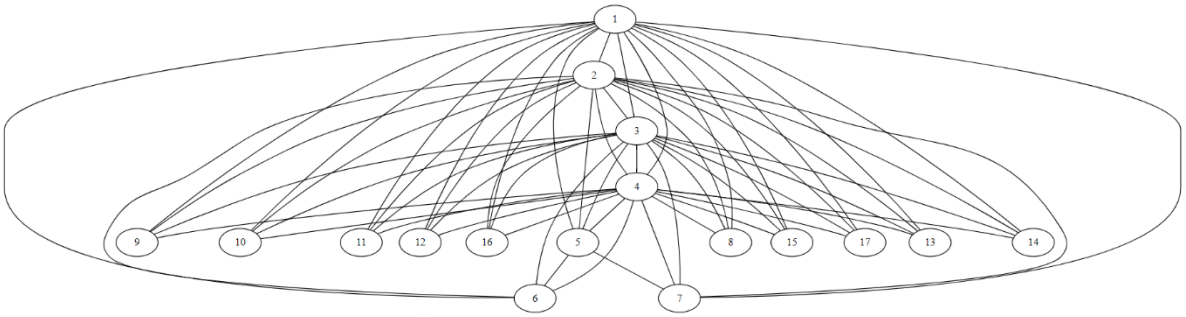
- **Fixez toate datele si modific numărul muchiilor(k - număr mare):**
  - Grafurile utilizate in ordinea numărului de muchii
    - 40 muchii



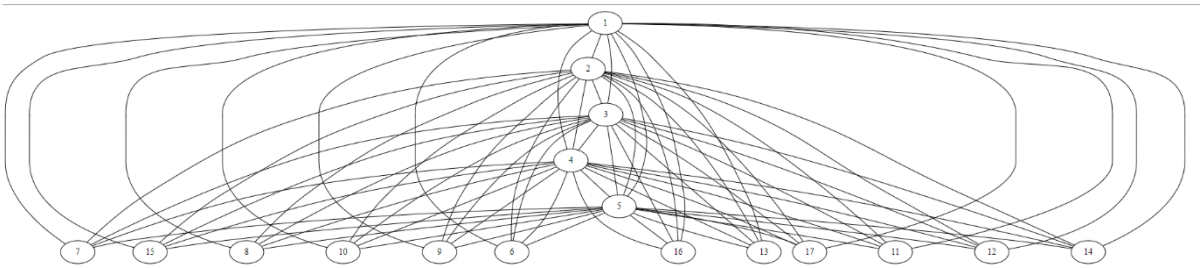
    - 50 muchii



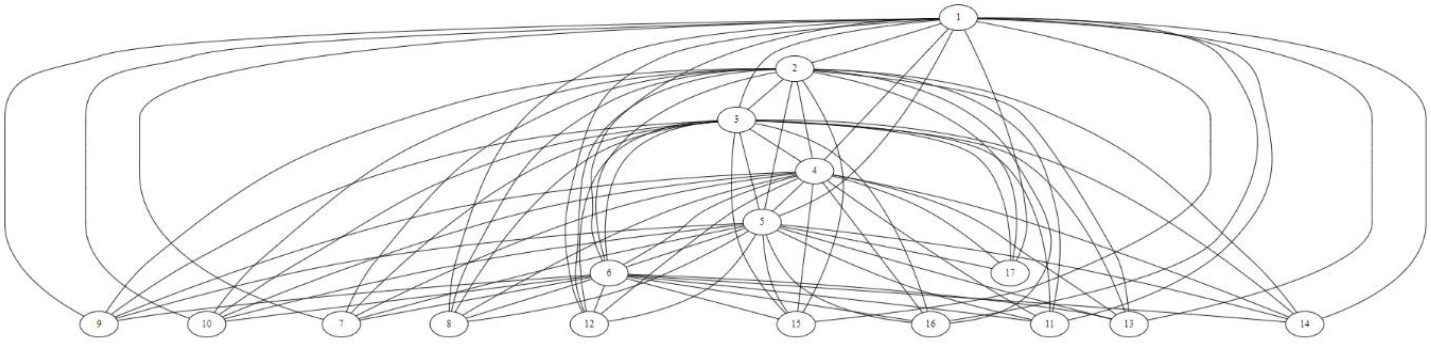
    - 60 muchii



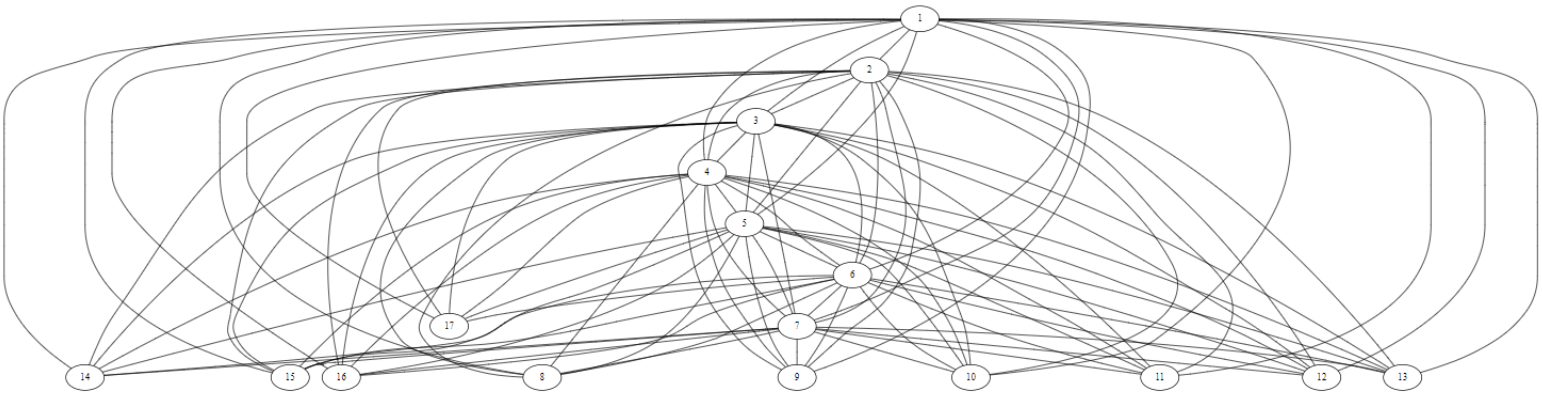
    - 70 muchii



    - 80 muchii

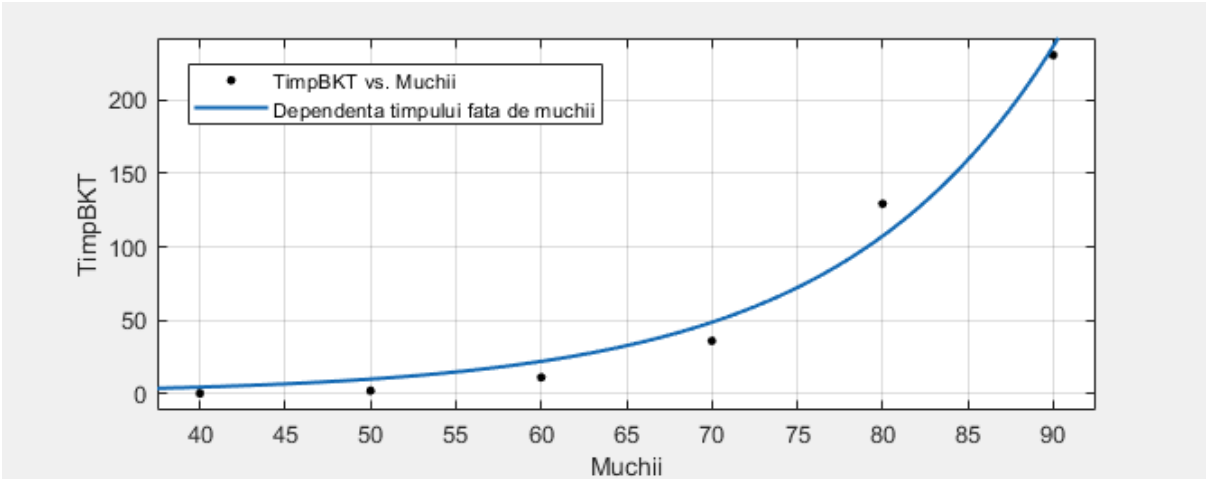


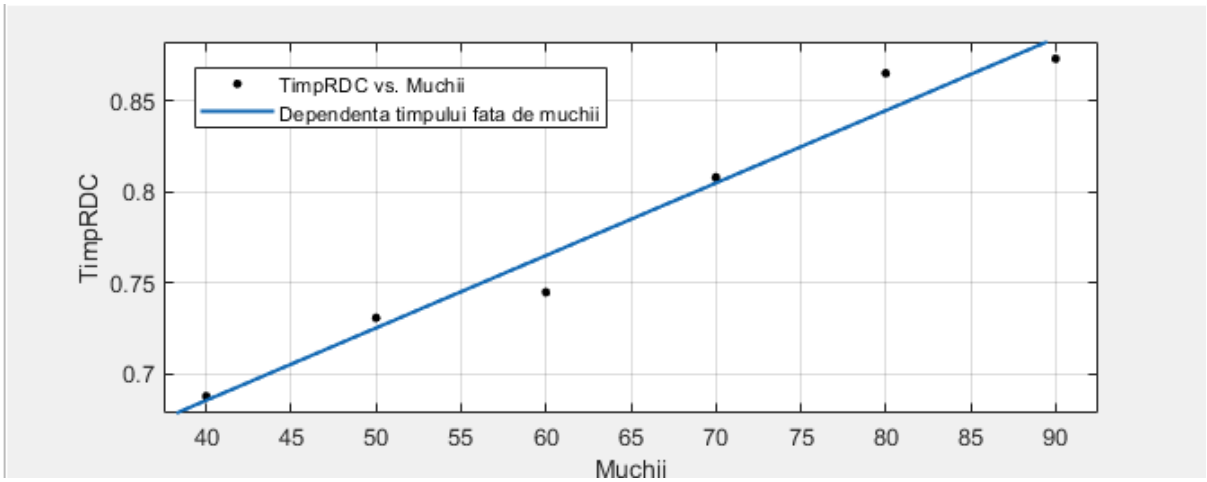
- 90 muchii



- Dependenta timpului fata de numărul de muchii

Nr. muchii	*Backtracking*	*Reduction*
40	0.266	0.873
50	2.136	0.808
60	11.172	0.745
70	36.065	0.688
80	129.492	0.731
90	230.781	0.865





- Aproximările făcute pentru a reprezenta grafic:
  - Backtracking

    Modelul funcției: a*exp(b*x)

    Coeficienți(cu limite de încredere 95%):

    a = 0.1927 (-0.2683, 0.6537)

    b = 0.07904 (0.05173, 0.1064)

  - Reduction

    Modelul funcției: p1*x + p2

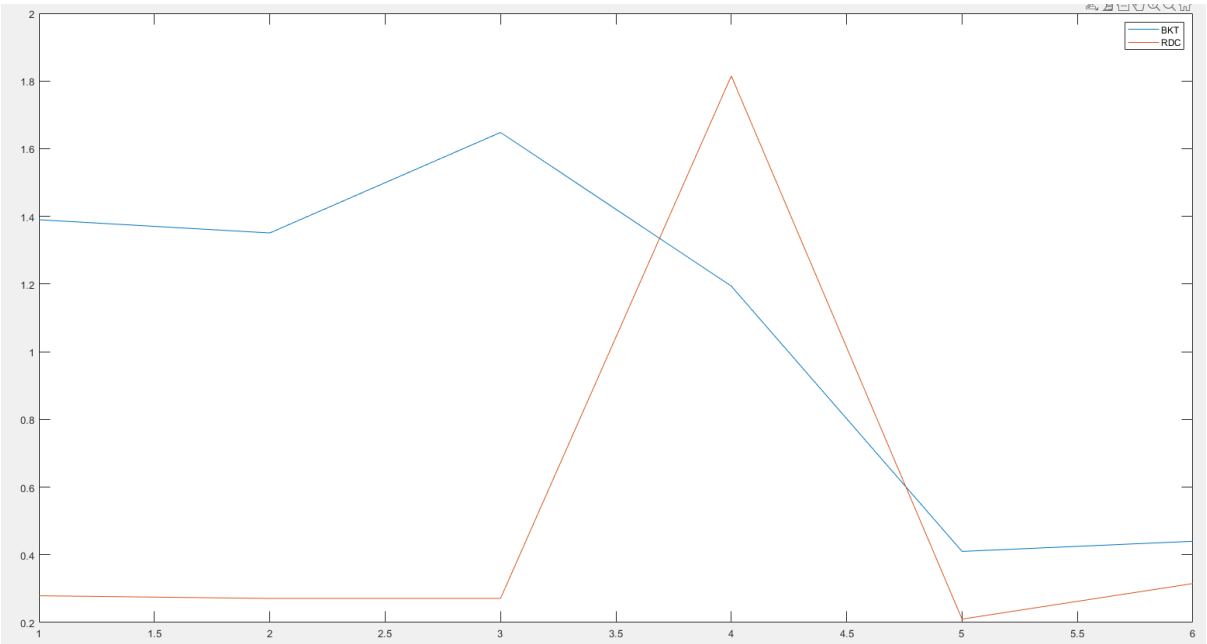
    Coeficienți(cu limite de încredere 95%):

    p1 = 0.003971 (0.002924, 0.005019))

    p2 = 0.5269 (0.4565, 0.5973)

- Concluzie
  - Se vede o dominare clara a algoritmului de reducere fata de cel backtracking, deoarece algoritmul de backtracking depinde mult de legăturile dintre noduri, însă ideea e ca timpul acesta e pentru situația când trebuie sa le verifice pe toate si tot sa nu găsească un răspuns.

- **Teste aliatoare care nu au doar răspunsul False:**



Concluzie Tema:

- Fiecare algoritm are atât avantajele cat si dezavantajele sale. Pentru algoritmul de backtracking problema e situația in care nu exista o clica sau aceasta clica se afla relativ departe de punctul de intrare a algoritmului, ceea ce înseamnă ca va dura mult pana se va găsi acea clica, însă pe lângă asta am mai văzut ca mult depinde timpul de rulare si de mărimea clicii si numărul de muchii, așa cum asta si creează posibilitățile, mai puțin numărul de noduri însă au si ele un rol important, așa cum cu un nod nu poți face 10 muchii. Algoritmul de reducere la SAT este unul identic rapid atât pentru răspunsul True, cat si pentru False, algoritmul depinde de mărimea clicii si numărul de noduri, așa cum de asta depinde si numărul de clauze ce trebuie evaluate de SAT. In concluzie pot spune ca algoritmul de reducere la SAT este unul preferat in rezolvarea acestei probleme, deoarece algoritmul de backtrakcing depinde de prea multe pentru a spune ca va rula mai rapid sau nu ca cel de reducere.

Restul este implementat de Girnet Andrei