Butnariu Bogdan-Mihai 322CB

Analiza Algoritmilor

Tema 2 - Reducere k-Clique $\leq_p$ SAT
Python

Backtracking:

Pentru a rezolva problema k-Clique in timp exponential am scris un algoritm recursiv. Fisierul kCliqueBKT.py are o functie ‚parseFile' cu rol in interarea in fisierul de intrare dat ca argument in linia de comanda si cu rol in construirea unui dictionar care contine pe campul ‚key' toate varfurile si pe campul ‚items' toate legaturile cu celelalte varfuri. Returneaza ‚k', numarul de varfuri ‚v' si numarul de muchii ‚numOfEdges'. Functia ‚cliques_recursive' primeste ca argument 2 seturi si construieste o clica. Algoritmul implementat are la baza algoritmul lui Bron-Kerbosch si este adaptat pe tipul de input. Acesta implementeaza diferite operatii pe multimile fiecarui varf si gaseste clicile de diferite dimensiuni din graf. Functia ‚kCliques' prin fiecare clica gasita si daca gaseste una cu dimensiunea ‚k' returneaza True, altfel returneaza False.

Reduction:

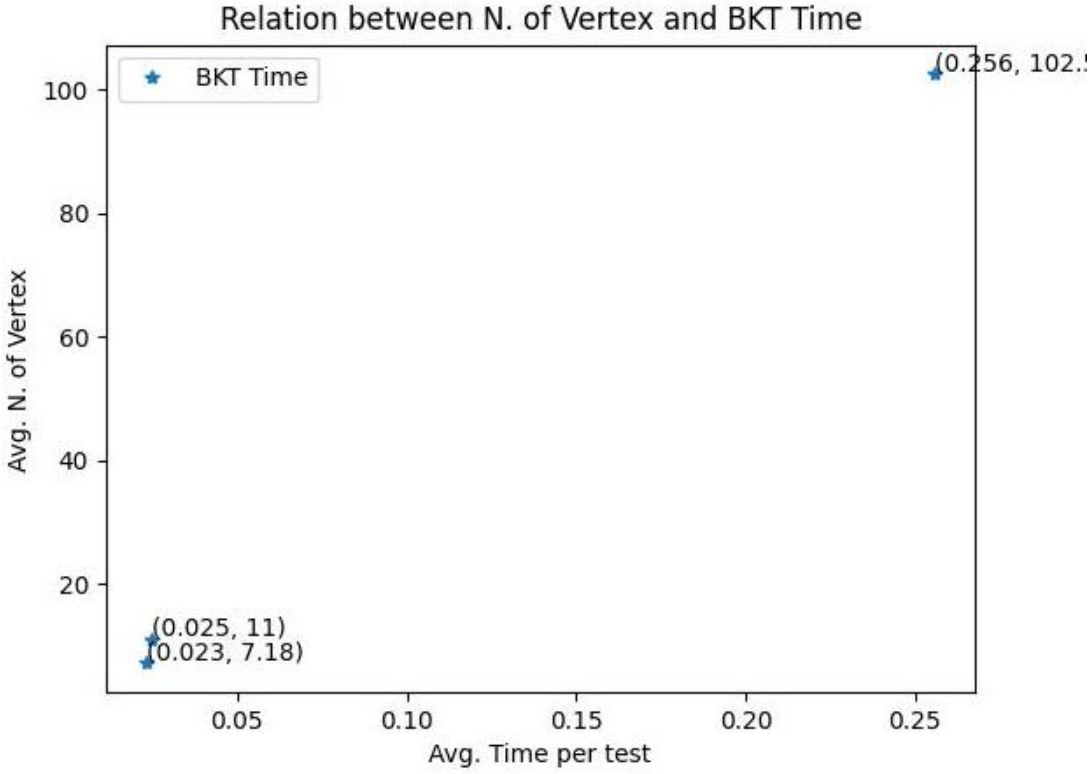
Pentru transformarea polinomiala, sursele mele de insiparie au fost raspunsul de pe StackExchange, raspunsul de pe un Blog, un videoclip de pe YouTube si aceasta lucrare a lui Steven Taschuk. Prima parte a transformarii polinomiale alcatuieste legaturile intre toate varfurile grafului pentru dimnesiunea k a clicii si exclude cazul in care varfurile sunt aceleasi. A doua parte alcatuieste restul de muchii ale grafului, parcurgand matricea de adiacenta. Aceasta parte exclude cazul in care pozitia din matrice este 0 din motive evidente. A treia parte construieste legaturile pentru k-ul impus cu restul de varfuri pentru a putea forma clica de dimensiune k sub forma de SAT. Prima parte are 2 ‚for-uri' care merg pana la k, un ‚for' care merge pana la v si o conditie care sare un numar de pasi, deci prima parte face un numar de kv. A doua parte contine 4 ‚for-uri', 2 care merg pana la k, 2 care merg pana la v. Scenariul cel mai costisitor este scenariul in care nu avem graf, matricea de adiacenta fiind populata cu 0, conditia ar sari un numar mai mic de pasi. Aceasta parte face k^2v^2 – kv pasi. A treia parte face kv pasi, avand 2 ‚for-uri'. Complexitatea rezultata este $\Theta((kv)^2)$, care este un timp polinomial.

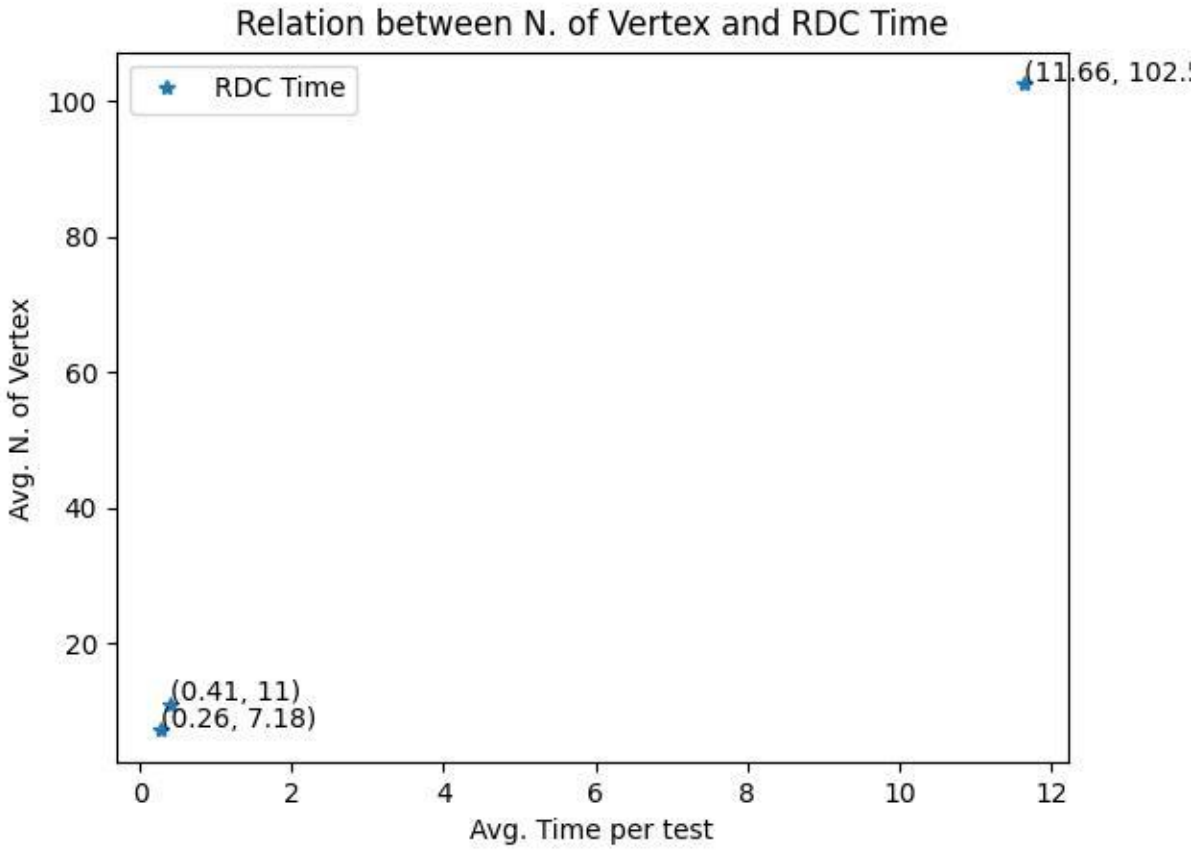
Timpii de rulare:

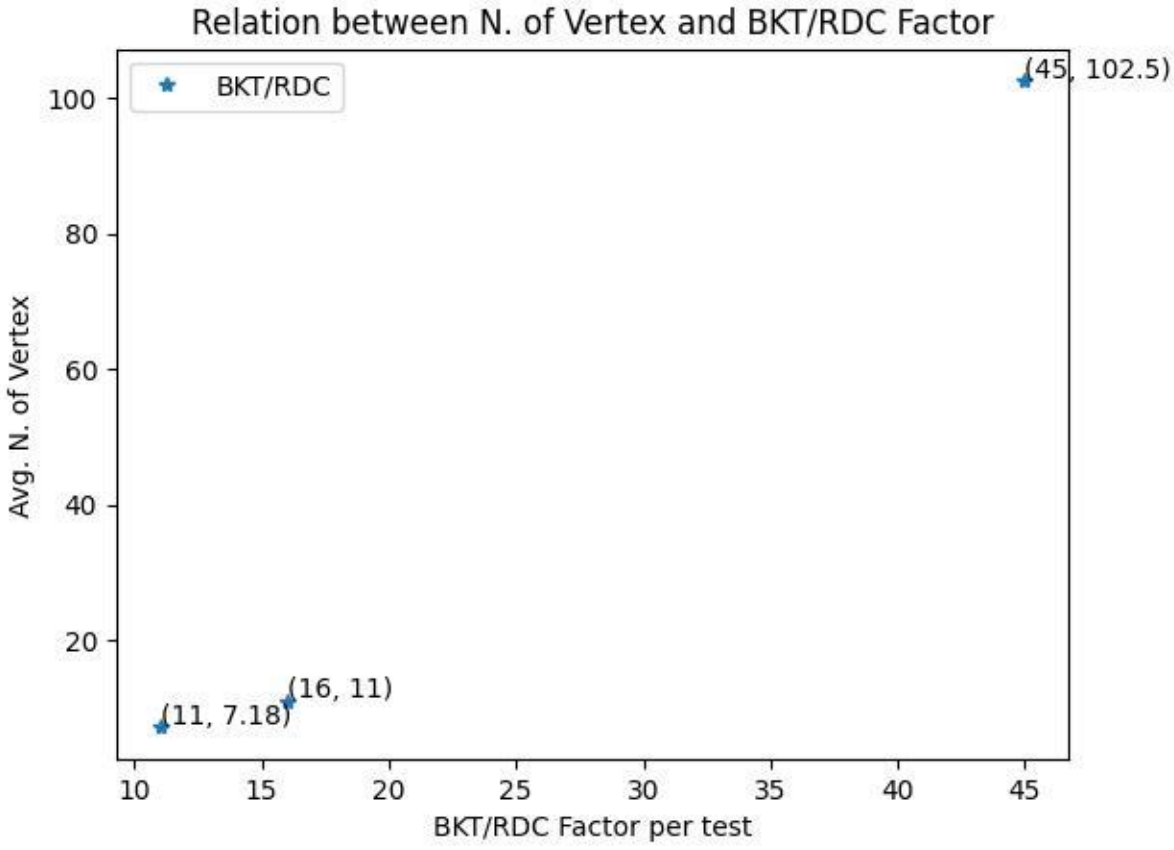
Algortimul implementat de mine are o complexitate $\Theta(k^2v^2)$, unde k reprezinta dimnesiunea clicii cautate si v reprezinta numarul de varfuri prezente in graf. Cum algoritmul de transformare polinomiala are de scris de in jur de 18 caractere la fiecare pas, acesta va genera un sir de caractere foarte mare care trebuie scris si adaugat unei variabile. Dupa ce transformarea este gata, algoritmul de SAT trebuie sa interpreteze tot sirul de caractere si sa returneze un rezultat dupa ce a parcurs tot sirul de caractere. Acest fapt explica de ce SAT are un timp de executie mai mare fata de algoritmul implementat la backtracking. Algoritmul de backtracking itereaza printr-un dictionar si se opreste cand gaseste prima cicla de dimensiune k.

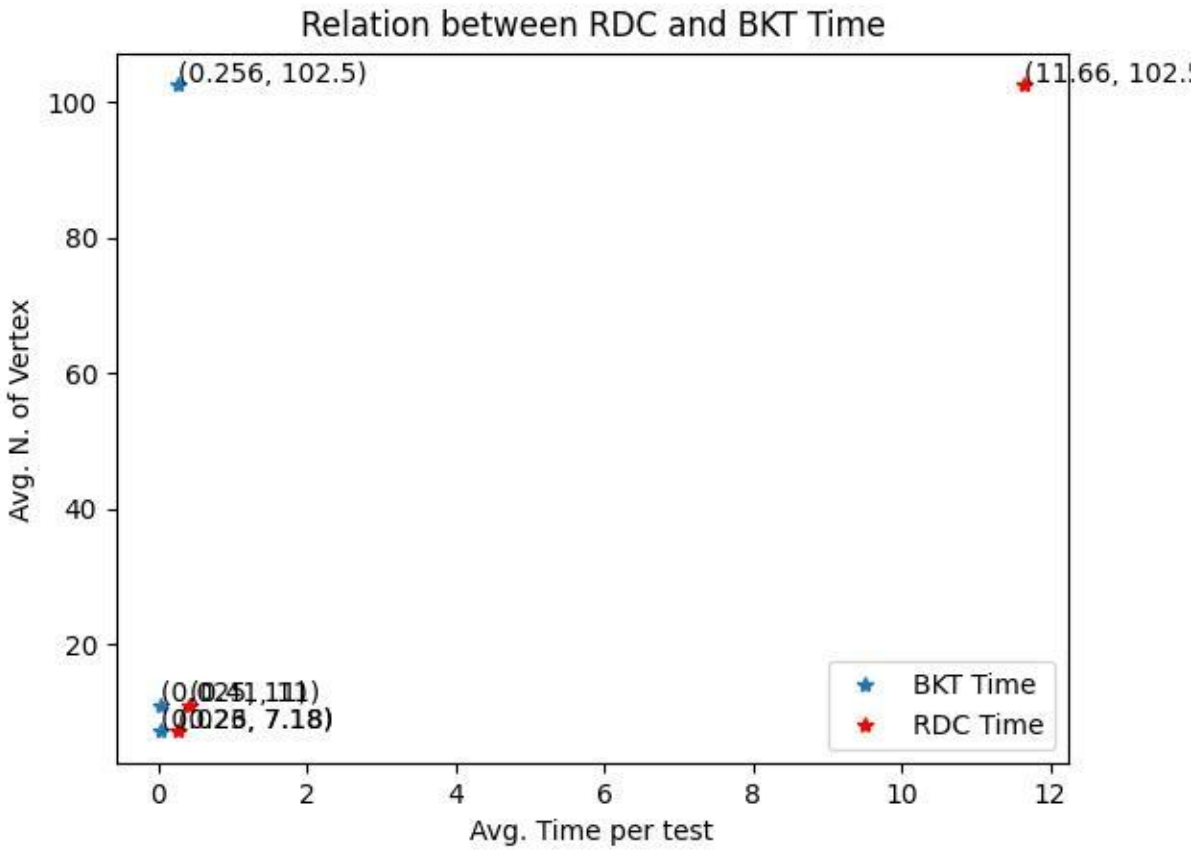
Discutie despre timp:

In testele date pentru checker, k-ul nu variaza asa mult intre categorii astfel incat sa avem un rezultat vizibil pe grafic (vezi complexitate), astfel ca discuta despre timp se va concentra pe variatia numarului de varfuri dintre categorii. Am calculat media nr. de varfuri pe fiecare categorie, astfel incat sa se poata observa pe grafic media unui test. Am rulat checker-ul de 5 ori cu conditii de rulare aleatorii pe laptopul personal si am gasit timpul pentru fiecare categorie in parte de 5 ori, urmand sa gasesc media pentru fiecare categorie. Dupa ce am gasit si timpul pentru fiecare test, am conceput 3 grafice reprezentantive pentru diferentele dintre BKT si RDC.




Relation between N. of Vertex and RDC Time
RDC Time
(11.66, 102.5)
(0.41, 11)
(0.26, 7.18)
Avg. N. of Vertex
Avg. Time per test
0
2
4
6
8
10
12
20
40
60
80
100


Relation between N. of Vertex and BKT/RDC Factor
BKT/RDC
(45, 102.5)
(16, 11)
(11, 7.18)
Avg. N. of Vertex
BKT/RDC Factor per test
10
15
20
25
30
35
40
45
20
40
60
80
100



Din graficele generate cu ‚matplotlib' se poate observa o creste de 10 ori a timpului in cazul BKT pentru o crestere de aprox. 10 ori a numarului de varfuri.

In cazul RDC, se poate observa o crestere de 2 ori a timpului pentru o crestere de 1.53 ori a numarului de varfuri si o crestere de aprox. 38 ori a timpului pentru o crestere de arpox. 10 ori a numarului de varfuri.

In ultimul grafic generat, putem observa cat de agresiv este cu timpul RDC fata de BKT pentru aceeasi medie pe fiecare test.