Tema 2 – Analiza Algoritmilor

Ionut-Cristian Savu

Task2 :

Ca punct de inspiratie pentru transformare, am folosit urmatoarele:

- Computational Complexity: Reductions To SAT
- satisfiability - CLIQUE $\leq_p$ SAT - Computer Science Stack Exchange
- https://www.geeksforgeeks.org/print-subsets-given-size-set/

Se poate testa dacă un graf G conține o clică de k noduri, putându-se găsi astfel orice clică pe care ea o conține, folosind un algoritm cu forță brută. Acest algoritm analizează fiecare subgraf cu k noduri și verifică dacă formează o clică.

Astfel, algoritmul meu de creare a unei expresii SAT care sa verifice existenta unei K-Clique intr-un graf suna astfel:

Se defineste un graf G = (V,E), unde V reprezinta numarul de noduri, si E reprezinta numarul de muchii, si un K fix, pentru ca daca acel K varia, transformarea se realiza in timp exponential si nu in timp polinomial.

Am ales 3 clauze pentru a creea expresiile cu literali:

- O clauza care construieste o expresie in care admite faptul ca exista un al i-lea nod in clica pentru fiecare nod al grafului, unde i-ul apartine intervalului (1,k), iar v-ul apartine (1,n); literalul este adevarat daca aceasta clica continue fiecare nod, iar complexitatea va fi O(k*n); un literal din expresie va fi de forma Xvi (deoarece am considerat ca al i-lea nod din clica este v, si am folosit aceasta notatie pe tot parcursul clauzelor)
- O clauza care construieste o expresie in care se verifica ca un nod este unic in clica, astfel am folosit doi indici, I si j, care sunt cuprinsi in intervalul (1,k), si un indice v, care apartine (1,n). Expresia va fi de forma ~xviV~xvj, pentru fiecare I, j, si v, expresie adevarata daca ambii literali nu sunt adevarati in acelasi timp; Comlexitatea acestei clause este O(n*k^2)
- O clauza care construieste o expresie in care se verifica veridicitatea ca toate nodurile sunt legate intre ele. Astfel, am folosit doi indici, I si j care apartin (1,k) si doi indici, v si u care apartin (1,n), si am verificat daca nu exista muchie intre nodurile v si u. Literalul va arata astfel: ~XviV~xuj, pentru fiecare I, j, v, u, si putem admite ca acest literal este adevarat daca oricare noduri din clica sunt legate intre ele. Complexitatea acestei clauze va fi O(k^2*n^2)

Legat de optimizarea algoritmului de creare a expresiilor cu literali, pot spune ca in cazul in care k == n, complexitatile vor deveni, in functie de ordinea clauzelor mele, urmatoarele: O(k^2), O(k^3) si respectiv O(k^4).

Corectitudinea expresiei construite va fi data daca si doar daca graful G(V, E) va avea o Clica de dimensiune K.

Daca e sa iau corectitudinea algoritmului pe clauze, voi avea astefel: pentru clauza 1,aceasta este adevarata pentru ca eu am presupus ca am o k-clique in graf, deci aceasta este completa (clica), si deci aceasta continue toate nodurile sale in ea. Pentru clauza 2, am considerat ca un nod este unic in clica, deci nu pot exista doua noduri diferite pe acelasi slot, deci clauza ramane adevarata, si nu in ultimul rand, pentru clauza 3, creand legaturi intre toate nodurile din graf, creez si legaturile nodurilor din clica, ceea ce va admite faptul ca si aceasta clauza este adevarata, iar in final putem admite ca algoritmul de constructie a expresiilor cu literali este adevarat, iar pentru ca K este fix, atunci transformarea se rezolva si in timp polinomial, deci putem admite ca problema KClique se reduce la SAT in timp polinomial.

Task3:

Timpi de rulare pentru categoria de teste:

Categoria 1:

- BKT: 0.242s
- RDC: 1.729s

Se poate observa ca pentru categoria 1 de teste, timpii de rulare sunt relativ mici, deoareca atat numarul de noduri si numarul de muchii sunt mici. Momentan, verificarea algoritmului de KClique se realizeaza mai rapid prin backtracking.

Categoria 2:

- BKT: 0.568s
- RDC: 36.614s

Se observa ca diferenta intre backtracking si reducere la SAT este de mai mult de jumatate de minut, deoarece categoria 2 de teste reprezinta categoria in care exista clica in graf, insa numarul de noduri si muchii este si el destul de mare (peste 100 de noduri in unele teste). Intrucat expresia cu literali trebuie sa parcurga de mai multe ori toate nodurile, creandu-se o expresie colossal de lunga, consider ca algoritmul de backtracking ramane, inca, efficient pentru a rezolva problema KClique.

Categoria 3:

- BKT: 0.688s
- RDC: 3.933s
  In aceasta categorie de teste, exista teste in care exista KClique in graf, insa exista si teste in care aceasta nu exista. Astfel, timpul de rezolvare prin backtracking este inca mic comparativ cu cel al reducerii la SAT. Totusi, comparativ cu categoria anterioara de teste, raportul intre cei doi timpi nu este unul mare, intrucat nici numarul de noduri din teste nu este nici acesta unul mare.