Problème de clique maximum

reda-mohamed.drissi@isty.uvsq.fr
Drissi Mohamed Reda

19 Fevrier 2018

Contents

1 Introduction **2**

2 Théorie **2**
2.1 Algorithme De Coloration De graphe 2
2.2 Trouver la clique max 3

3 Implémentation en C++ **4**

4 Test de performance à l'aide de Valgrind © **5**

1 Introduction

Après avoir testé plusieurs algorithmes (glouton, Bron-Kerbosch, branch and bound), il parait que l'algorithme Clique_Max (trouvé sur wikipedia) en utilisant la coloration du graphe, soit le plus rapide, (du moins au moment de l'implémentation).
Le but est d'utiliser l'algorithme approximé de coloration de graphes pour determiner une borne, ce qui facilite la recherche d'une clique maximum.

2 Théorie

Soit un graphe $G = (V, E)$ tel que $V = \{1, 2, 3, 4.., n\}$. Pour chaque sommet $v \in V$, $\Gamma(v)$ est l'ensemble de tous les voisins de v donc $\deg(v) = |\Gamma(v)|$.
Soit $G(R) = G(R, E \cap R \times R)$ le sous-graphe de G comprenant tous les éléments de R, tel que $R \subset V$.

2.1 Algorithme De Coloration De graphe

Q est l'ensemble des sommets de la clique qui est actuellement construite, Q_{max} est l'ensemble des sommets de la plus grande clique trouvée jusqu'à maintenant.
L'algorithme insère chaque sommet v dans la première classe C_k de couleur possible, si aucune classe n'est possible, nous créons une nouvelle classe C_{max_no+1} puis nous y insérons v.
Pour notre cas nous remarquons que les sommets $v \in R$ ayant une couleur $C(v) < |Q_{max}| - |Q| + 1$ ne vont jamais être rajoutées à la clique actuelle (Q), donc nous calculons cette valeur $k_{min} = |Q_{max}| - |Q| + 1$ au début de l'itération, et nous initialisons le compteur de ces sommets à $j \leftarrow 0$.
Quand un sommet $v = R[i]$ est attribué à une classe C_k et que $k < k_{min}$ nous décalons ce sommet vers la position j, $R[j] \leftarrow R[i]$, puis nous incrémentons j.
Une fois tous les sommets ont été attribués à une couleur, les sommets ayant $k < k_{min}$ sont au début de R. Les autres sommets tel que $k \geq k_{min}$ sont copiés depuis les classes de couleurs C_k vers R dans un ordre croissant de k. Uniquement à ces sommets sont attribués des couleurs $C(v) = k$.

```
procedure COLORATION(R,C)
    max_no ← 1
    k_min ← |Q_max| − |Q| + 1
    if k_min ≤ 0 then
        k_min ← 1
    j ← 0
    C_1 ← ∅ C_2 ← ∅
    for i ∈ [1, |R| − 1] do
        p ← R[i]
        k ← 1
        while C_k ∩ Γ(p) ≠ ∅ do
            k ← k + 1
        if k > max_no then
            max_no ← k
            C_{max_no+1} ← ∅
        C_k ← C_k ∪ {p}
        if k < k_min then
            R[j] ← R[i]
            j ← j + 1
    C[j − 1] ← 0
    for k ∈ [k_min,max_no] do
        for i ∈ [1, |C_k|] do
            R[j] ← C_k[i]
            C[j] ← k
            j ← j + 1
```

2.2 Trouver la clique max

Nous initialisons :

- $Q \leftarrow \emptyset$
- $Q_{max} \leftarrow \emptyset$
- ALLSTP $\leftarrow 1$
- Tous les éléments de S et S_{old} sont initialisés à 0

Trier les sommets de R selon leur degrés nous permet de gagner en performance en théorie, puisque cela permet d'avoir une borne supérieure moins grande, et nous permet de réduire le nombre d'étapes au minimum mais puisque cela prend un temps $O(|R|^2)$, nous ne gagnons en performance que quand R est assez large, donc uniquement dans les premières étapes.
En utilisant lv nous comptons le nombre d'appels récursifs depuis la racine jusqu'à la feuille actuelle.

Le nombre d'étapes max où nous n'allons pas trier R doit être determiné dynamiquement, car la densité est un plus grand facteur que la taille du graphe, dans la taille de la clique max.
Les variables $S[lv]$ et $S_{old}[lv]$ sont des variables globales qui nous permettrons de compter le nombre d'étapes. La variable ALLSTP calcule le nombre d'étapes maximales.
À chaque étape nous calculons $S[lv]$/ALLSTP et nous le comparons à T_{limit} qui est un paramètre choisi en testant differents graphes, et en regardant sur internet.
Quand $S[lv]$/ALLSTP$\geq T_{limit}$ nous ne trions pas R.
À chaque itération la fonction mets à jour S et S_{old} avec

$$S[lv] \leftarrow S[lv] + S[lv-1] - S_{old}[lv] S_{old}[lv] \leftarrow S[lv-1]$$

Algorithm 1 Clique_Max(R,C,lv)

```
procedure CLIQUE_MAX(R,C,lv)
    S[lv] ← S[lv] + S[lv − 1] − S_old[lv]
    S_old[lv] ← S[lv − 1]
    while R ≠ ∅ do
        Choisir un sommet p ayant C(p) max (dernier sommet) dans R
        R ← R \ {p}
        if |Q| + C[index de p dans R] > |Q_max| then
            Q ← Q ∪ {p}
            if R ∩ Γ(p) ≠ ∅ then
                if S[lv]/ALLSTP < T_limit then
                    Calculer le degré des sommets de G(R ∩ Γ(p))
                    Trier les sommets de R ∩ Γ(p) par ordre décroissant
                    (selon leurs degrés)
                COLORATION(R ∩ Γ(p),C')
                S[lv] ← S[lv] + 1
                ALLSTP ← ALLSTP +1
                CLIQUE_MAX(R ∩ Γ(p),C',lv+1)
            else if |Q| > |Q_max| then Q_max ← Q
            Q ← Q \ {p}
        else
            return
```

3 Implémentation en C++

Pour compiler le projet nous pouvons utiliser le `Makefile`.
Puis pour exécuter nous pouvons suivre le format :

```
./maxc <fichier.ls|.mat|.clq>
```

4 Test de performance à l'aide de Valgrind ©

Nous allons tester notre programme avec plusieurs instances de DIMAC.

brock200_1 Commande :

```
time ./maxc bingraph/brock200_1.clq
```

Résultat

```
args = bingraph/brock200_1.clq
|E| = 14834  |V| = 200 p = 0.7417
etape n = 0 taille de clique max= 1
etape n = 13 taille de clique max= 14
etape n = 19 taille de clique max= 15
etape n = 68 taille de clique max= 16
etape n = 147 taille de clique max= 17
etape n = 625 taille de clique max= 18
etape n = 2675 taille de clique max= 19
etape n = 11253 taille de clique max= 20
etape n = 57248 taille de clique max= 21
Clique Max: 134 102 85 39 150 92 136 68 186 178 135 94 90 93 20 18 73 87 81 142 108
Taille = 21
Nombre d'etapes = 232999
```

Temps d'exécution

```
real  0m0.410s
user  0m0.400s
sys   0m0.000s
```

brock200_2 Commande :

```
time ./maxc bingraph/brock200_2.clq
```

Résultat

```
args = bingraph/brock200_2.clq
|E| = 9876  |V| = 200 p = 0.4938
etape n = 0 taille de clique max= 1
etape n = 5 taille de clique max= 6
etape n = 9 taille de clique max= 7
etape n = 14 taille de clique max= 8
etape n = 40 taille de clique max= 9
etape n = 1166 taille de clique max= 10
etape n = 1566 taille de clique max= 12
Clique Max: 135 158 145 105 121 27 55 183 70 48 149 120
Taille = 12
Nombre d'etapes = 3606
```

Temps d'exécution

```
real  0m0.033s
user  0m0.020s
sys   0m0.000s
```

brock400_2 Commande :

```
time ./maxc bingraph/brock400_2.clq
```

Résultat

```
args = bingraph/brock400_2.clq
|E| = 59786  |V| = 400 p = 0.747325
etape n = 0 taille de clique max= 1
etape n = 17 taille de clique max= 18
etape n = 29 taille de clique max= 19
etape n = 184 taille de clique max= 20
etape n = 3604 taille de clique max= 21
etape n = 87268 taille de clique max= 22
etape n = 665019 taille de clique max= 23
etape n = 10259821 taille de clique max= 24
etape n = 20022091 taille de clique max= 29
Clique Max: 388 252 20 39 92 142 276 178 68 380 134 73 348 93 234 260 150 365 262 207 221
18 304 90 108 311 85 186
Taille = 29
Nombre d'etapes = 44365961
```

Temps d'exécution

```
real  1m55.945s
user  1m55.936s
sys   0m0.000s
```

brock400_4 Commande :

```
time ./maxc bingraph/brock400_4.clq
```

Résultat

```
args = bingraph/brock400_4.clq
|E| = 59765  |V| = 400 p = 0.747062
etape n = 0 taille de clique max= 1
etape n = 15 taille de clique max= 16
etape n = 20 taille de clique max= 17
etape n = 54 taille de clique max= 18
etape n = 104 taille de clique max= 19
etape n = 199 taille de clique max= 20
etape n = 309 taille de clique max= 21
```

```
etape n = 24492 taille de clique max= 22
etape n = 162628 taille de clique max= 23
etape n = 1528510 taille de clique max= 24
etape n = 1589415 taille de clique max= 25
etape n = 50859228 taille de clique max= 26
etape n = 51039931 taille de clique max= 33
Clique Max: 161 362 353 294 266 157 393 135 241 324 17 7 343 270 202 112 267 380 389
197 396 394 211 340 147 334 245 186 154 247 8 19 242
Taille = 33
Nombre d'etapes = 54362635
```

Temps d'exécution

```
real  2m1.055s
user  2m0.996s
sys   0m0.004s
```

Nous faisons tourner ce dernier `brock800_2` avec Valgrind

brock800_2 Commande :

```
time ./maxc bingraph/brock800_2.clq
```

Résultat

```
args = bingraph/brock800_2.clq
|E| = 208166  |V| = 800 p = 0.650519
etape n = 13 taille de clique max= 14
etape n = 16 taille de clique max= 15
etape n = 161 taille de clique max= 16
etape n = 986 taille de clique max= 17
etape n = 2943 taille de clique max= 18
etape n = 49586 taille de clique max= 19
etape n = 4320201 taille de clique max= 20
etape n = 303732455 taille de clique max= 21
etape n = 586725133 taille de clique max= 24
Clique Max: 649 417 160 504 460 271 445 348 580 109 155 214 414 792 258 12 84 323 130 383
Taille = 24
Nombre d'etapes = 1315510644
```

Temps d'exécution

```
real  57m18.137s
user  57m17.928s
sys   0m0.008s
```

brock800_4 Commande :

```
time ./maxc bingraph/brock800_4.clq
```

Résultat

```
args = bingraph/brock_800_4.clq
|E| = 207643  |V| = 800 p = 0.648884
etape n = 13 taille de clique max= 14
etape n = 66 taille de clique max= 15
etape n = 130 taille de clique max= 16
etape n = 674 taille de clique max= 17
etape n = 1263 taille de clique max= 18
etape n = 139158 taille de clique max= 19
etape n = 1189470 taille de clique max= 20
etape n = 64329588 taille de clique max= 21
etape n = 199277938 taille de clique max= 26
Clique Max: 333 246 269 623 665 713 156 388 494 508 395 323 111 686 16 393 134 342 680 560
Taille = 26
Nombre d'etapes = 501919040
```

Temps d'exécution

```
real  27m25.811s
user  27m25.664s
sys   0m0.004s
```

keller4 Commande :

```
time ./maxc bingraph/keller4.clq
```

Résultat

```
args = bingraph/keller4.clq
|E| = 9435  |V| = 171 p = 0.645349
etape n = 0 taille de clique max= 1
etape n = 10 taille de clique max= 11
Clique Max: 130 162 48 152 65 44 49 138 68 135 147
Taille = 11
Nombre d'etapes = 8586
```

Temps d'exécution

```
real  0m0.014s
user  0m0.012s
sys   0m0.000s
```

hamming8-4 Commande :

```
time ./maxc bingraph/hamming8-4.clq
```

Résultat

```
args = bingraph/hamming8-4.clq
|E| = 20864  |V| = 256 p = 0.636719
etape n = 0 taille de clique max= 1
etape n = 11 taille de clique max= 12
etape n = 30 taille de clique max= 13
etape n = 316 taille de clique max= 16
Clique Max: 95 162 139 212 60 183 45 158 197 240 99 118 17 8 249 74
Taille = 16
Nombre d'etapes = 11293
```

Temps d'exécution

```
real  0m0.041s
user  0m0.032s
sys   0m0.000s
```