université de BORDEAUX

UNITÉ DE FORMATION Mathématiques & interactions

PROJET : MÉTA-HEURISTIQUES

Partitionnement d'un graphe en p classes

Equipe :
BAERT NICOLAS
CHEN XIAOYU
JOLIOT ANNA

Enseignant :
MARC-MICHEL CORSINI

M1 ROAD
Année scolaire 2022 - 2023

1 Mise en place / fonctionnement

Ce rapport de projet est completé par le dossier compressé de codes joint. Il nous a semblé que la programmation orientée object proposée par le c++ était la plus adaptée. Notre dossier comprend un fichier `main.cpp`, deux fichiers (un `.cpp` et un `.h`) pour chaque implémentation et un fichier `utiles.cpp` avec son fichier d'en-tête `utiles.h` qui regroupe toutes les structures et fonctions générales au problème.
Afin d'obtenir l'éxécutable, voici les lignes de commandes à écrire :

```
g++ -c utiles.cpp Lecturebis.cpp enumeration.cpp enumerationbis.cpp main.cpp
Dgradient.cpp

g++ -o main.exe main.cpp utiles.o Lecturebis.o enumeration.o enumerationbis.o
Dgradient.o

.\main.exe
```

L' ordinateur utilisé pour faire tourner les algorithmes est :

- Asus ZenBook, processeur intel Core i5, Ram 8G sous windows 10.

2 Description du problème

Le but de ce projet est de trouver une partition optimale d'un graphe en p groupes tout en minimisant le poids des arêtes intergroupe. Nous approcherons le problème par différents algorithmes de calcul. Dans un premier temps, nous élaborerons un algorithme d'énumération. Ce dernier donnera l'optimum, et nous l'utiliserons pour valider nos approches postérieures. Ensuite, nous développerons un algorithme glouton du gradient.

Cependant, il faut avant tout fixer certains critères. Nous considèrerons comme solution réalisable toute solution qui respectera les conditions suivantes :

- Aucune classe n'est vide
- Tous les sommets sont mis dans une classe
- Soit N le nombre de sommets et p le nombre de classes. Chaque groupe devra respecter : $|groupe| = \frac{N}{p} \pm 1\% \frac{N}{p}$.

Ce dernier critère est très important et fera l'objet d'une étude approfondie par la suite. Il s'agit d'un critère de "raisonabilité". En effet, pour un graphe de 600000 sommets, nous pourrions vouloir répartir les sommets par le nombre de sommets divisé par le nombre de classes, prenons $p = 3$; ce qui associerait 125000 sommets par groupe. Cependant, avec autant de sommets, ne serait-il pas raisonnable d'accepter une meilleure solution si un groupe comprenait 125600 sommets et un autre 124400 ? C'est pourquoi, nous considèrerons qu'1% est une marge suffisante pour notre étude. Il pourra être intéressant de faire varier ce taux dans la suite de ce projet, afin de voir l'évolutoin quant aux valeurs optimales.

Enfin, le nombre p de groupes est à déterminer. Nous voilà au début de ce projet et nous pensons établir $3 \leq p \leq 10$. Nous fixerons pour les premiers essais $p = 3$ puis nous étudierons l'impact de ce choix.

3 Structures des données

Les structures des données utilisées sont écrites dans le fichier `utiles.h`.
Une solution est décrite par `vector<vector<int>>` . Ce formalisme nous semble le mieux car il est facile d'accès et que le cardinal des groupes n'est pas constant.
Un graphe est considéré comme une liste d'arcs :

```
struct arc{
    int predecesseur;
    int voisin;
    float cost;
};
typedef vector<arc> graphe;
```

Dans les algorithmes nous avons besoin de lister les arcs donc un vecteur est le plus optimal.

4 Algorithme d'énumération

Nous avons décidé de commencer par un algorithme d'énumération explicite. Il est codé sous le nom `"enuperation.cpp"`. Voici son pseudo-code :

```
Set L ← solution vide;
Set S, S_Best ← ∅;
while L ≠ ∅ do
    Select a subproblem s_parent from L to explore;
    for arc(a,b) in A do
        if a et b sont déjà dans un groupe then
            nothing;
        else if a dans un groupe, pas b then
            for i in nb classes sans la classe de a do
                s_enfant ← ajout de b à la classe i ;
                if s_enfant réalisable then
                    ajout de s_enfant à S ;
                    if S_Best > s_enfant then
                        S_Best ← s_enfant;
                    end
                else
                    ajout de s_enfant dans L ;
                end
            end
        else
            s_enfant ← toutes les combinaisons d'ajout de a et b à des classes ;
            if s_enfant réalisable then
                ajout de s_enfant à S ;
                if S_Best > s_enfant then
                    S_Best ← s_enfant;
                end
            else
                ajout de s_enfant dans L ;
            end
        end
    end
    enlever s_parent de L;
end
```

Amélioration :

Cet algorithme d'énumération donne bien les solutions optimales mais est très long. C'est l'algorithme basique, qui contient des symétries. Un moyen d'accélérer cet algorithme est de les supprimer. Par exemple, lorsque les deux premiers sommets sont mis dans des groupes, lors de la première boucle, une symétrie est créée. Voici un exemple :
Prenons un graphe qui contient k sommets et travaillons pour une répartition en 3 groupes. La solution initiale est celle où tous les groupes sont vides : <>,<>,<> .
Soit (a, b) le premier arc étudié, alors l'algorithme ci-dessus donnera neuf sous-solutions :

- <ab>,<>,<>
- <>,<ab>,<>
- <>,<>,<ab>
- <a>,<b>,<>
- <a>,<>,<b>
- <b>,<a>,<>
- <>,<a>,<b>
- <b>,<>,<a>
- <>,<b>,<a>

Alors que certaines sont équivalentes, et reviennent aux deux solutions suivantes :

- <ab>,<>,<>
- <a>,<b>,<>

On peut donc améliorer l'algorithme en initialisant L non par la solution vide, mais une liste de solutions :deux solutions proposées juste ci-dessus, et ce pour chaque arc du graphe. Cette amélioration est codée sous le nom `"enumerationbis.cpp"`.

Voici les résultats obtenus pour les fichiers patati patata :
durée : 1.29381 Dans le groupe 1 se trouvent les sommets 1 Dans le groupe 2 se trouvent les sommets 2 5 4 Dans le groupe 3 se trouvent les sommets 3
Cette solution a pour valeur 5
L'amélioration est donc efficace puisque le temps a été divisé par plus de 5 sur l'instance à 5 sommets! Cependant, même si l'énumération est donc une méthode optimale, elle prend beaucoup de temps et il n'est pas raisonnable de l'utiliser pour les très grosses instances. En revanche, elle nous sera utile pour valider nos prochaines approches algorithmiques.

5 Algorithme de descente de gradient

Le Voisinage:
Dans notre problème, nous avons plusieurs combinaisons du nombre de sommets contenus dans un groupe, en fonction du critère d'équilibre de la partition. On choisit donc P ∩ D qui nous permet de modifier la structure, même si nous ne pouvons pas garandir que tout voisinage est réalisable. Il nous faut donc le vérifier avant de calculer la valeur de la fonction objective.
Le mouvement élémentaire:
On choisit l'un des sommets et on le déplace dans un autre groupe. Voici un exemple avec $P = 3, nb\_sommets = 5$:

- solution initiale <a b> <c d> <e>
- (a,2) <b> <acd> <e>
- (a,3) <b> <cd> <ae>
- (b,2) <a> <bcd> <e>
- (b,3) <a> <cd> <be>
- (c,1) <abc> <d> <e>
- (c,3) <ab> <d> <ce>
- (d,1) <abd> <c> <e>
- (d,3) <ab> <c> <de>
- (e,1) <abe> <cd> <>
- (e,2) <ab> <cde> <>

La taille de voisinage $| Voisinage | = nb\_sommets * (P - 1)$
Voici le pseudo-code de l'algorithme de descente de gradient:

$t \leftarrow 0$;
$S_t \leftarrow$ Solinitiale;
$f_{opt} \leftarrow f(s_t)$;
$S_{best} \leftarrow S_0$;
$S_{t+1} \leftarrow \emptyset$;
Fini $\leftarrow$ False;
while *Fini = False* **do**
  $V \leftarrow voisinage(S_t)$;
  $s_{t+1} = argbest f(s), s \in V(s_t)$;
  if $f(s_{t+1})$ *is better than* $f(s_t)$ **then**
    $f_{opt} \leftarrow f(s_{t+1})$;
    $S_{best} \leftarrow S_{t+1}$;
    $t \leftarrow t + 1$;
  else
    Fini $\leftarrow$ Vrai
  end
end

6 Résultats :

Après 1 essai avec $P = 3$:
CinqSommets:
Nous avons trouvé l'optimum global qui est 6 par l'algorithme d'énumération en 1,4s et un optimum local qui est 7 par l'algorithme de descente de gradient en 0,15s avec une erreur de 16%. Pour des graphes de si petite taille, nous pouvons obtenir l'optimum global dans un temps raisionnablel par l'algorithme d'énumération.
DixSommets:
Comme notre procédure d'énumération prend trop de temps pour traiter dix sommets et plus, nous n'avons pas pu obtenir de résultat final. Ceci est probablement dû au fait que nous traitons les arêtes plutôt que les sommets, et qu'il serait préférable de traiter les sommets directement pour ce type de graphe dense, et nous améliorerons cet algorithme dans le rapport final.
Cependant, nous avons obtenu un optimum local qui est 23 en 0.16s, par rapport aux cinq sommets, cela n'a guère changé.
MilleSommets:
Même en traitant un graphe de mille sommets, notre programme peut trouver un optimum local qui est 6646 en 0.67s. Nous pouvons dire qu'il s'agisse d'un graphe de cinq sommets ou d'un graphe de mille sommets, nous pouvons trouver un optimum local en moins d'une seconde par l'algorithme de descente de gradient, bien que nous ne disposions actuellement d'aucun autre algorithme pour en comparer la qualité.
Sachant que les résultats obtenus par cet algorithme dépendent fortement de la solution initiale, et comme son temps d'exécution est très court à chaque fois, nous pouvons obtenir une valeur moyenne en changeant plusieurs fois sa solution initiale stochastique, ce qui améliorera largement la qualité de notre solution.
Après 1000 essais avec $P = 3$:
CinqSommets:
La valeur moyenne après mille essais est de 6,635, ce qui représente une amélioration de 5%,

mais encore une erreur de 10%.
DixSommets et MilleSommets:
Les valeurs moyennes après mille essais sont respectivement de 21.92 et de 6671.25, ce qui représente respectivement une amélioration de 4% et de 0.3%.
On constate que le pourcentage d'amélioration n'est pas significatif, ce qui illustre les limites de cet algorithme.
Après 1000 essais avec $P = 3, P = 4, P = 5$:
En faisant varier la valeur de p, nous constatons que pour la méthode d'énumération le temps augmente beaucoup, en revanche la valeur de p a très peu d'effet sur le temps d'exécution de la méthode de descente du gradient.

6.1 Classification en 3 groupes :

6.1.1 Après 1 essai

```
centSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  1355  et temps moyen :  0.0163999 s.
   cinqCentSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  4983  et temps moyen :  0.159139 s.
   cinqSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode d'énumération explicite :  6  et temps moyen :  6.26014 s.
   Méthode d'énumération améliorée :  6  et temps moyen :  1.48817 s.
   Méthode de la descente de gradient :  7  et temps moyen :  0.159281 s.
   cinquanteSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  330  et temps moyen :  0.162155 s.
   dixSeptSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  84  et temps moyen :  0.162927 s.
   dixSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  23  et temps moyen :  0.163324 s.
   milleSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  6646  et temps moyen :  0.670291 s.
   quatreSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode d'énumération explicite :  3  et temps moyen :  6.29949 s.
   Méthode d'énumération améliorée :  3  et temps moyen :  1.49676 s.
   Méthode de la descente de gradient :  4  et temps moyen :  0.670344 s.
   quinzeSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  66  et temps moyen :  0.670837 s.
   trenteSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  276  et temps moyen :  0.672986 s.
   vingtCinqSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  206  et temps moyen :  0.674935 s.
   vingtDeuxSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  149  et temps moyen :  0.675981 s.
   vingtEtunSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  144  et temps moyen :  0.67708 s.
```

```
   vingtQuatreSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  175  et temps moyen :  0.678454 s.
   vingtSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  72  et temps moyen :  0.678972 s.
   vingtTroisSommets.txt
   Valeur optimale moyenne après 1 essais :
   Méthode de la descente de gradient :  160  et temps moyen :  0.680266 s.
```

6.1.2 Après 1 essai

```
centSommets.txt
   Valeur optimale moyenne après 1000 essais :
   Méthode de la descente de gradient :  1346.54  et temps moyen :  0.0160667 s.
   cinqCentSommets.txt
   Valeur optimale moyenne après 1000 essais :
   Méthode de la descente de gradient :  5010.45  et temps moyen :  0.160809 s.
   cinqSommets.txt
   Valeur optimale moyenne après 1000 essais :
   Méthode d'énumération explicite :  6  et temps  :  6.22821 s.
   Méthode d'énumération améliorée :  6  et temps  :  1.38075 s.
   Méthode de la descente de gradient :  6.635  et temps moyen :  0.160871 s.
   cinquanteSommets.txt
   Valeur optimale moyenne après 1000 essais :
   Méthode de la descente de gradient :  339.886  et temps moyen :  0.163899 s.
   dixSeptSommets.txt
   Valeur optimale moyenne après 1000 essais :
   Méthode de la descente de gradient :  83.987  et temps moyen :  0.164693 s.
   dixSommets.txt
   Valeur optimale moyenne après 1000 essais :
   Méthode de la descente de gradient :  21.92  et temps moyen :  0.164869 s.
   milleSommets.txt
   Valeur optimale moyenne après 1000 essais :
   Méthode de la descente de gradient :  6671.25  et temps moyen :  0.522888 s.
   quatreSommets.txt
   Valeur optimale moyenne après 1000 essais :
   Méthode d'énumération explicite :  3  et temps  :  0.0356483 s.
   Méthode d'énumération améliorée :  3  et temps  :  0.0091004 s.
   Méthode de la descente de gradient :  3.685  et temps moyen :  0.52295 s.
   quinzeSommets.txt
   Valeur optimale moyenne après 1000 essais :
   Méthode de la descente de gradient :  64.842  et temps moyen :  0.523426 s.
   trenteSommets.txt
   Valeur optimale moyenne après 1000 essais :
   Méthode de la descente de gradient :  271.798  et temps moyen :  0.525462 s.
   vingtCinqSommets.txt
```

```
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  205.342  et temps moyen :  0.52717 s.
vingtDeuxSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  148.721  et temps moyen :  0.528281 s.
vingtEtunSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  145.295  et temps moyen :  0.529324 s.
vingtQuatreSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  175.615  et temps moyen :  0.530622 s.
vingtSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  70.675  et temps moyen :  0.531441 s.
vingtTroisSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  159.177  et temps moyen :  0.532635 s.
```

6.2 Classification en 4 groupes :

Notons la solution pour l'instance à 4 sommets est triviale. `centSommets.txt`

```
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  1514.56  et temps moyen :  0.0198486 s.
cinqCentSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  5635.62  et temps moyen :  0.174544 s.
cinqSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode d'énumération explicite :  7  et temps  :  33.3713 s.
Méthode d'énumération améliorée :  7  et temps  :  4.2123 s.
Méthode de la descente de gradient :  7.245  et temps moyen :  0.174626 s.
cinquanteSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  382.349  et temps moyen :  0.178334 s.
dixSeptSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  94.302  et temps moyen :  0.17912 s.
dixSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  24.867  et temps moyen :  0.179353 s.
milleSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  7505.92  et temps moyen :  0.555528 s.
quatreSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode d'énumération explicite :  4  et temps  :  0.143286 s.
```

```
Méthode d'énumération améliorée :  4  et temps  :  0.0186362 s.
Méthode de la descente de gradient :  4  et temps moyen :  0.555591 s.
quinzeSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  72.878  et temps moyen :  0.556168 s.
trenteSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  305.52  et temps moyen :  0.559164 s.
vingtCinqSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  231.018  et temps moyen :  0.560976 s.
vingtDeuxSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  166.634  et temps moyen :  0.562509 s.
vingtEtunSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  162.652  et temps moyen :  0.563854 s.
vingtQuatreSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  197.61  et temps moyen :  0.565574 s.
vingtSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  79.135  et temps moyen :  0.566368 s.
vingtTroisSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  178.762  et temps moyen :  0.567867 s.
```

6.3 Classification en 5 groupes :

Notons qu'il n'y a pas de solution pour l'instance à 4 sommets et que la solution pour l'instance à 5 sommets est triviale. `centSommets.txt`

```
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  1615.61  et temps moyen :  0.0200221 s.
cinqCentSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  6011.21  et temps moyen :  0.193477 s.
cinqSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode d'énumération explicite :  8  et temps  :  149.175 s.
Méthode d'énumération améliorée :  8  et temps  :  14.7786 s.
Méthode de la descente de gradient :  8  et temps moyen :  0.193605 s.
cinquanteSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  409.078  et temps moyen :  0.199547 s.
dixSeptSommets.txt
Valeur optimale moyenne après 1000 essais :
```

```
Méthode de la descente de gradient :  100.531  et temps moyen :  0.200621 s.
dixSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  26.436  et temps moyen :  0.200955 s.
milleSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  8006.98  et temps moyen :  0.657076 s.
quatreSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode d'énumération explicite :  inf  et temps  :  0.466778 s.
Méthode d'énumération améliorée :  inf  et temps  :  0.0380505 s.
Méthode de la descente de gradient :  4  et temps moyen :  0.657149 s.
quinzeSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  77.81  et temps moyen :  0.65819 s.
trenteSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  326.429  et temps moyen :  0.661954 s.
vingtCinqSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  246.786  et temps moyen :  0.664681 s.
vingtDeuxSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  177.844  et temps moyen :  0.6667 s.
vingtEtunSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  173.498  et temps moyen :  0.668809 s.
vingtQuatreSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  211.162  et temps moyen :  0.671131 s.
vingtSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  84.412  et temps moyen :  0.672116 s.
vingtTroisSommets.txt
Valeur optimale moyenne après 1000 essais :
Méthode de la descente de gradient :  190.662  et temps moyen :  0.674209 s.
```