Métathéorèmes
Rapport pour le cours de Graphes

Table des matières

1 Introduction

1.1 Sujet

L'objectif était d'étudier 4 méta théorèmes, afin d'illustrer ou réfuter ce qu'ils avancent. Voici donc les quatres énoncés, qui seront étudiés dans cet ordre.

- les versions "arêtes" (resp. "arcs") sont plus faciles que les versions "sommets" (graphes non orienté, resp. orientés).
- les versions "non orienté" sont plus faciles que les versions "orienté" (graphes non orientés versus orientés).
- les versions "graphe biparti" sont aussi difficiles que les versions "graphe quelconque" (graphes non orientés).
- les versions "graphe sans circuit" sont plus faciles que les versions "graphe quelconque" (graphes orientés).

2 Métathéorème 1

2.1 Rappel de l'énoncé

Les versions "arêtes" (resp. "arcs") sont plus faciles que les versions "sommets" (graphes non orienté, resp. orientés).

2.2 Version sommets aussi dure

Considérons les problèmes de flot dans un graphe. La version arêtes est aussi dure que la version sommet. En effet, via une simple transformation, on peut passer d'un problème à l'autre, la réponse étant la même. On pose Pmax = capacité maximale de arêtes (ou des sommets).

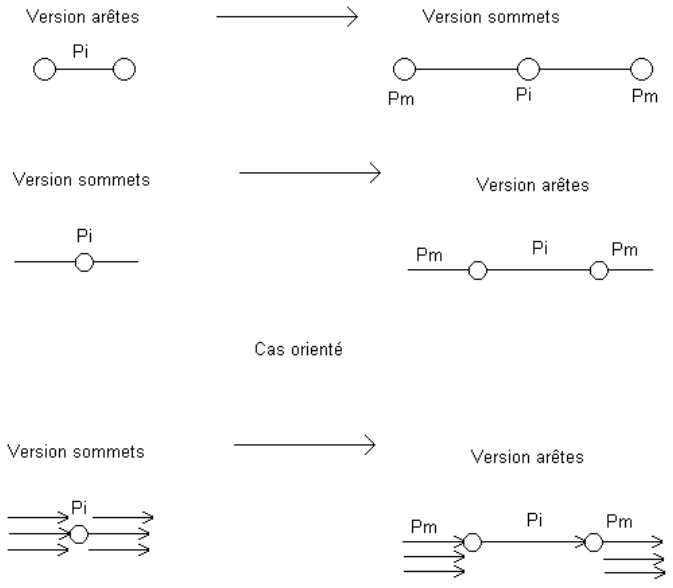

FIG. 1 – Conversion arêtes-sommets

À travers ce schéma (voir figure 1), on voit que on peut passer d'une instance de flots "arêtes" à une instance de flots "sommets" facilement, polynomialement, et avoir équivalence des résultats. De plus, cet exemple est aussi valable dans le cas orienté, puisqu'il suffit alors de conserver l'orientation des arêtes.

2.3 Les sommets ont des solutions plus simples

On va maintenant montrer un exemple dans lequel la solution "sur les sommets" est plus simple que celle sur les arêtes. Mais on ne parle pas de la complexité pour y arriver (dans un premier temps).
Il s'agit du problème de trouver un x-y séparateur (soit les arêtes, soit les sommets). Petit dessin :

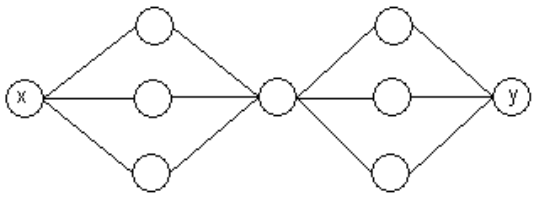

FIG. 2 – Pas si simple

Encore une fois, ceci est valable dans le cas orienté comme dans le cas non orienté. On voit que le x-y séparateur version sommet est un singleton, le sommet du milieu. Alors que sur les arêtes, il faut au minimum 3 arêtes (généralisable à n arêtes) pour déconnecter x de y.
On a même un résultat plus fort : arêtes>=sommets. En effet, étant donné un x-y séparateur version "arêtes", on peut obtenir un x-y séparateur version "sommets" de cardinal inférieur ou égal : il suffit de supprimer un sommet de l'un des deux bouts d'une arête du x-y séparateur.
Mais... dans le cas ou l'arête (x,y) appartient au graphe, il n'existe pas de x-y séparateur version "sommets" sans supprimer x ou y. Tandis que la version arêtes est toujours possible. On peut par exemple supprimer toutes les arêtes.

2.4 Les sommets sont plus durs

2.4.1 Pré-supposé

Le graphe est connexe/fortement connexe.

2.4.2 Non orienté

Voici un cas ou les arêtes sont plus simples que les sommets en terme de complexité pour trouver la solution au problème : Chemins (resp. cycle) Hamiltoniens (pour les sommets) versus chemins(resp. cycles) Eulériens (pour les arêtes).
Trouver un chemin (resp. cycle) Eulérien dans un graphe est simple : on compte le degré des sommets. Si on a 0 sommets de degré impair, alors il existe un cycle Eulérien (et

donc un chemin). Si on a 2 sommets de degré impair, alors on a un chemin, et pas de cycle. On ne s'interesse pas au fait de trouver le chemin ou le cycle (ce qui est simple, et même polynomial...)
Trouver un chemin (cycle) Hamiltonien dans un graphe est un problème NP-complet, il n'est pas nécessaire de le redémontrer.

2.4.3 Cas orienté

On peut démontrer que le chemin (ou cycle) Eulerien reste simple dans le cas orienté. Il existe un cycle eulérien si le degré entrant de chaque sommet est égal au degré sortant. S'il y a un cycle, il y a un chemin. Et enfin, il existe un chemin dans le cas où :

- On a un sommet tel que le degré sortant soit le degré entrant + 1.

- On a un sommet tel que le degré entrant soit le degré sortant +1.
- Tous les autres sommets ont le même degré entrant et sortant.

Enfin, le cas orienté des cycles/chemins Hamiltonien recouvre le cas non orienté (on mets des arcs dans les deux sens). Par conséquent, le problème des cycles/chemins Hamiltonien est au moins aussi dur dans le cas orienté que dans le cas non orienté.

2.4.4 Conclusion sur ce cas

Ainsi on a pu voir un cas ou la version sommet est NP-complète, tandis que la version arêtes (arcs) est dans P. A moins de démontrer que P=NP, on peut donc conclure que le métathéorème dit vrai dans ce cas.

2.5 Conclusion sur le Métathéorème 1

D'une part, la distinction "orienté" ou "non orienté" est inutile, au moins sur les exemples que j'ai pu voir. Passer du non orienté à l'orienté n'a fait que potentiellement accentuer la différence, et ce, dans le dernier cas.
D'autre part, on voit que selon le graphe et selon le problème, on peut avoir de tout (cf x-y séparateur).
Mais, on a pu montrer via un dernier exemple que la version "arêtes/arcs", dans P est potentiellement plus simple que la version sommets qui elle est NP-complète.

3 Métathéorème 2

3.1 Rappel de l'énoncé

Les versions "non orienté" sont plus faciles que les versions "orienté" (graphes non orientés versus orientés).

3.2 Cas non orienté ⇒ orienté

On peut considérer un graphe non orienté comme un graphe orienté particulier : pour chaque arête (x,y) appartenant au graphe non orienté, on ajoute (x,y) et (y,x) dans le graphe orienté. Ainsi "connexe" se tranforme en "fortement connexe".

3.3 Test de la connexité

On sait que la connexité est testable dans le cas non orienté en O(m+n). La question qui vient est "est-ce qu'on peut tester si un graphe est fortement connexe avec la même complexité ?". La réponse à cette question est "oui". Mais l'algorithme pour le cas orienté est beaucoup plus compliqué que pour le cas orienté. Plus loin vient un pseudo-code pour l'algorithme, mais avant, je vais résumer son idée générale. Il s'agit simplement de prendre un sommet S, et regarder les sommets atteignables. Puis, pour ceux qui sont atteignable : poursuivre l'augmentation de l'ensemble de ceux qui sont atteignables, et si jamais on tombe sur S et/ou sur un sommet marqué dans la CFC de S, ajouter le sommet courant dans la CFC, ainsi que tous les points du chemin de S au point courant.

```
bool traite[n] ; // pour savoir si on a déja traité un sommet
list<int> dependance[n] ; //dependance[i] :liste de sommets dans la CFC de 0 si i y est
bool T[n] ; //pour savoir si un sommet est dans la composante fortement connexe de 0
list<int> atraiter ; //sommets non traités, atteignables depuis 0.
traite[0]=true ;
T[0]=true ;
atraiter.push(0) ;
while( !atraiter.empty())
{
    encours=atraiter.front() ; //sommets que l'on traite
    pour tous les voisins v de encours
        if(T[v] and !T[encours]) //si le voisin est dans la CFC
        {
            T[encours]=true ;//alors le sommet est dans la CFC
            pour tout sommet s dans dependance[v]
                T[s]=true ;//et tous ceux qui dépendent du sommet aussi
        }
        else
        {
            dependance[v].pushfront(encours) ;//alors le sommets dépends de si le voisin
est dans la CFC
            pour tout somme s dans dependance[encours] n'appartenant pas a CFC (T[s]=false)
                dependance[v].pushfront(s) ;//et tout ceux qui dépendaient du sommet
aussi                 if( !traite[v])//si je n'ai pas déja marqué v comme atteignable
            {
                atraiter.pushback(v)//je l'ajoute dans la liste a traiter
                traite[v]=true ;// je le marque
```

```
            }
        }
}
```

On pourrait penser que cet algorithme travaille en $O(n^2)$ à cause des copies de dépendances. Ceci dit, on peut modifier un peu l'algorithme (je ne l'ai pas fait pour des raisons de clarté) afin d'éviter ces recopies. On n'ajouterai ainsi dans les dépendances que le sommet en cours de traitement. Et, pour tout sommet s du parcours de dépendance[v] verifier si les sommets de dependance[s] sont tous marqués dans T. si non, effectuer la descente récursive sur dependance[s] également.
Ainsi, on ne copie la dépendance d'un sommet que "degré sortant" fois. Soit en O(m). On se retrouve ainsi avec une complexité en O(m+n)

3.4 Intérêts de la transformation

Cette transformation permet de résoudre les problèmes de chemin/cycle Hamiltoniens du cas orienté dans le cas non orienté. Autrement dit : si on sait résoudre dans le cas orienté, on sait résoudre dans le cas non orienté. En effet, supposons qu'on ait trouvé un chemin Hamiltonien $x_1,x_2,...,x_n$ dans le graphe orienté (le cas cyclique se déduit du cas chemin avec $x_1=x_n$). Alors on sait que pour tout $i \neq j$, $x_i \neq x_j$. De plus, l'existence de l'arc (x_i,x_{i+1}) implique l'existence d'une arête (x_i,x_{i+1}). Donc le chemin $x_1,x_2,...,x_n$ est aussi possible dans le graphe non orienté, et il passe par tous les sommets une et une seule fois. Donc il s'agit d'un chemin Hamiltonien.

3.5 Problèmes d'expression

Si avec cette transformation, on peut passer d'un graphe non orienté à un graphe orienté, on ne peut pas vraiment faire l'inverse. En effet, considérons les problèmes de cycles/chemins Eulériens. Supposons qu'il existe une transformation arc⇒(arêtes+sommets) (cf figure suivante). Alors dans le cas non orienté, on pourra prendre le chemin dans les deux sens, ce qui n'est pas le cas pour le graphe orienté. Si transformation il y a, elle n'est donc pas si triviale que ca.
Les graphes orientés bénéficient donc d'une expressibilité plus importante. À titre d'exemple, beaucoup de problèmes d'ordonancement peuvent se traduire par des graphes orientés, mais n'ont pas de sens sur les graphes non orientés.

3.6 Est-si simple ?

Non, tout n'est pas si simple. En effet, certains problèmes d'ordonnancement se traduisant par des problèmes sur des graphes orientés sont NP-complets. Par exemple, le problème d'ordonnancer un graphe de tache sur P processeurs revient à trouver une p-partition dans le graphe sous certaines conditions. Or, le problème de chemin hamiltonien (ou du TSP) sont aussi NP-complets et sur des graphes non orientés. On peut donc trouver une équivalence entre certains problèmes sur des graphes orientés et d'autres problèmes sur des graphes non orientés.

3.7 Conclusion sur le métathéorème 2

On a donc vu que les problèmes de graphes orientés pouvaient se traduire d'une certaine facon en problèmes dans les graphes orientés. Les graphes orientés sont donc (selon les problèmes) aussi compliqués que les cas orientés. À travers le test de la connexité, on peut penser que algorithmiquement parlant, travailler sur un graphe non orienté est plus simple.
Dernière chose, on pourrait penser à tort que tous les problèmes se traduisant sur des graphes orientés uniquement ne sont pas expressible sur des graphes orientés. Pour le cas de problème NP-complets, on pourra s'y ramener.

4 Métathéorème 3

4.1 Rappel de l'énoncé

Les versions "graphe biparti" sont aussi difficiles que les versions "graphe quelconque" (graphes non orientés).

5 Biparti $\in$ Quelconque

Comme l'indique le titre de la section, les graphes bipartis sont aussi quelque part des graphes quelconques. Par conséquent, il est inutile de chercher à trouver des exemples ou les graphes bipartis sont plus durs que les graphes quelconques.
On peut même ajouter que la reconnaissance d'un graphe biparti est très simple (pas de cycle de longueur impaire), et que donc les graphes bipartis ne sont pas si "complexes" que ca.

5.1 C'est vrai !

Le problème de trouver le nombre chromatique dans un graphe quelconque est NP-complet, c'est bien connu. Or, sur un graphe biparti, qui osera dire qu'il est NP-complet ? Il est carrément constant... Voici un exemple très simple ou le cas biparti est beaucoup plus simple que le cas général.

5.2 Est-ce toujours le cas

Non, certains problèmes sont aussi compliqués sur les graphes bipartis que sur les graphes normaux. En effet, trouver un couplage maximal par exemple est aussi "simple" dans le cas général et dans le cas des graphes bipartis (il s'agit de trouver des chemins améliorants).

5.3 Les bipartis sont-ils si simples ?

Encore une fois, cela serait sous-estimer les graphes bipartis. Être biparti signifie simplement qu'on peut décomposer le graphe en deux sous ensembles indépendants. De ce fait, cela n'a aucune incidence sur le problème de cycle hamiltonien, qui reste NP-complet. On peut donc exprimer des problèmes sur les graphes bipartis.

5.4 Conclusion sur le métathéorème 3

Les graphes bipartis sont donc plus simples que les graphes quelconques, ne serait-ce que pour le problème de la coloration des graphes. On peut ajouter également que la reconnaissance d'un graphe biparti est faisable en O(n+m), ce qui indique quelque part que leur définition n'est pas si compliquée que ca.

6 Métathéorème 4

6.1 Rappel de l'énoncé

Les versions "graphe sans circuit" sont plus faciles que les versions "graphe quelconque" (graphes orientés).

6.2 Chemin de longueur maximale

6.2.1 Sans cycle

Le problème de trouver un chemin de longueur maximale dans un graphe (c'est à dire un chemin passant par une arête une et une seule fois) est un problème que l'on peut résoudre simplement dans un graphe orienté sans cycle. En effet, l'algo suivant résouds le problème :
int Longueur[n] ; //longueur du plus long chemin partant de i
Pour tout sommet s de degré sortant 0
    longueur[s]=0
    marquer s ;
Tant qu'il existe des sommets non marqués
    Pour chaque sommet s dont tous les voisins sont marqués
    longueur[s]=$\max_{v \in N(s)}$(longueur[v])+1
    marquer s
trouver le maximum de longueur[i] ;
retourner ce maximum ;
Cet algorithme ne fonctionne pas s'il existe un cycle, car pour tout sommet du cycle, à aucun moment tous ses voisins seront marqués, et donc il ne sera jamais marqué lui aussi.

6.2.2 avec cycle

Je n'ai pas réussi à trouver d'algorithme pour résoudre ce problème de facon simple, et je n'ai pas non plus reussi à trouver des articles traitant de ce problème (dans le cas avec cycle). Il existe des algorithmes pour trouver le plus court/long chemin entre deux points d'un graphe. On pourrait imaginer traiter le problème en utilisant l'algorithme pour toutes les paires de sommets, mais nous n'aurions aucun garantie. En effet, en faisant tourner ca sur le graphe suivant : On voit que le chemin le plus long est de longueur 8, tandis que calculer le chemin le plus long pour tout couple de sommet nous rendra le

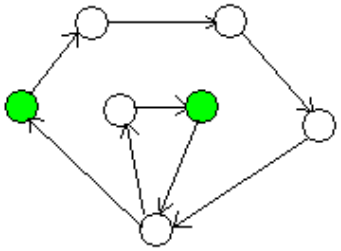

FIG. 3 – Problème

résultat 6 (sommets en vert).

J'ai tenté de réduire quelques problèmes NP-complets au problème du plus long chemin (en particulier celui de trouver un cycle hamiltonien, qui me semblait le plus proche, via une transformation arcs⇔sommets) mais je n'ai pas reussi. Ce qui m'a fait penser à ce problème, c'est le fait qu'on passe une seule fois par chaque arc, alors que le problème du cycle fait passer une unique fois par chaque sommet. Donc trouver un cycle serait revenu à chercher un chemin de longueur≥n.

7 Conclusion

On a pu voir à travers certains exemples que :

- Métathéorème 1 : Il n'est pas tout à fait valide selon le sens que l'on donne à "simple". Si on parle de classe de complexité des algorithmes, il est vrai. Si on parle en termes de solution, il est faux. Et dans certains problèmes, les sommets sont équivalents aux arêtes... Enfin, la distinction orienté vs non orienté n'est pas intervenue efficacement dans les exemples étudiés.
- Métathéorème 2 : Il est difficile de faire une conclusion sur ce théorème. Il y a une transformation intéressante qui, lorsqu'on sait résoudre un problème dans le cas non orienté permets de le résoudre dans le cas orienté. Cependant, le cas orienté est plus "expressif" et permets de faire plus de choses. Malgré tout, les problèmes NP-complets dans le cas orienté peuvent se transformer en problèmes NP-complets dans le cas non orienté par définition de la NP-complétude. Encore une fois, la définition de simple dépends de comment on l'interprète. En terme de complexité, les deux se valent. En terme de facilité de compréhension, le cas orienté est plus simple.
- Métathéorème 3 : Ce théorème est plutôt simple, vu que les graphes bipartis sont des restrictions des graphes quelconques. En terme de complexité, comme on l'a vu, etre biparti permets parfois de simplifier les choses (nombre chromatique).
- Métathéorème 4 : Je n'ai pas vraiment pu faire quelque chose de bien rigoureux pour ce théorème dans la mesure où je ne suis pas parvenu à établir la complexité exacte de la recherche d'un chemin élémentaire de longueur maximale dans le cas ou il y a des cycles. Une chose est sûre cependant, c'est qu'il est plus facile d'aborder

ce problème dans le cas sans cycle.

Ces problèmes étaient intéressants et m'ont permi de voir (cas 1) que parfois, ce qu'on intuite n'est pas toujours juste, et qu'il faut aussi se fixer des définitions claires sur certains mots ("simple") afin de pouvoir bien étudier les cas. Je me suis surtout penché sur les trois premiers métathéorèmes, tout en essayant malgré tout d'avoir des éléments de réponse sur le dernier.