Résolution exacte et approchée

François Delbot

10 janvier 2020

Nous allons étudier de manière approfondie le problème du vertex cover :

- Comment le résoudre de manière exacte.
- Comment le résoudre de manière approchée.

Pourquoi ce problème ?

- Un problème simple à appréhender.
- De nombreux problèmes sont très fortement liés à celui-ci.

D'autres problèmes seront ensuite présentés de manière succincte.

Le problème du vertex cover

Un problème NP-difficile

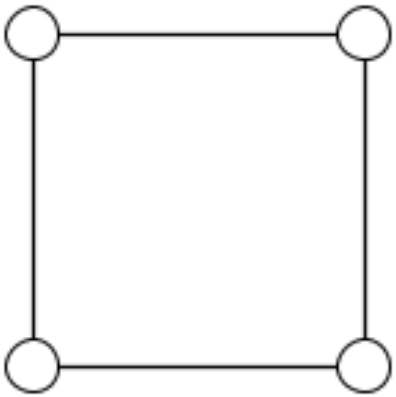
En 1972, un an après que Cook ait formalisé la notion de NP-complétude et prouvé que le problème SAT est NP-complet, Richard Karp a publié un article fondateur en théorie de la complexité, dans lequel il prouve la NP-complétude de 21 problèmes dont celui du Minimum Vertex Cover.

Definition (Minimum Vertex Cover)

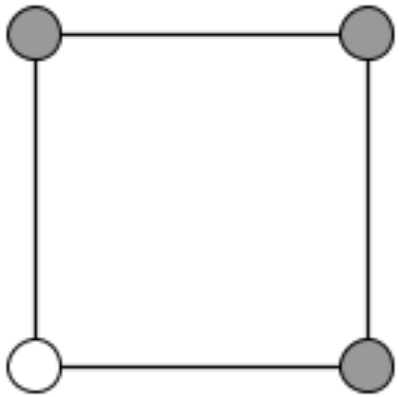
Étant donné un graphe $G = (V, E)$, avec V l'ensemble de ses sommets et E l'ensemble de ses arêtes, le problème du vertex cover consiste à trouver le plus petit ensemble possible $C \subseteq V$ tel que pour toute arête $(i, j) \in E$ on ait $i \in C$ ou $j \in C$ (ou bien les deux). On dit que C est une couverture minimale de G (un vertex cover de G).

Le problème du vertex cover

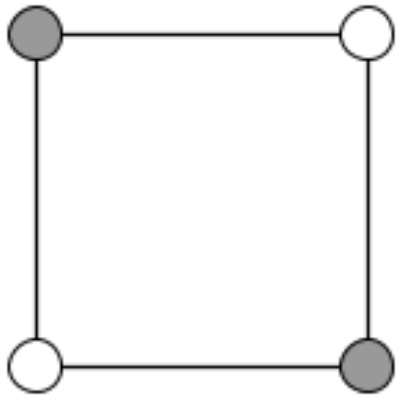
Exemple



Un graphe G



Une couverture des sommets de G



Une couverture optimale des sommets de G

Le problème du vertex cover

Un problème de décision ET un problème d'optimisation

1. Un problème de décision : Existe-t-il un sous-ensemble des sommets, tel que cet ensemble est un vertex cover et que la taille de cet ensemble soit de taille k ? (NP-complet)
2. Un problème d'optimisation : Trouver un sous-ensemble des sommets, tel que cet ensemble soit un vertex cover et qu'il soit de taille minimum. (NP-difficile)

Le problème du vertex cover

Un problème fondamental

La recherche d'un vertex cover intervient souvent dans la résolution d'autres problèmes :

1. Alignement de séquences multiples.
2. Résolution de conflits biologiques.
3. Surveillance des réseaux.

Le problème du vertex cover

Méthodes de résolution

Le fait que ce problème soit NP-difficile signifie qu'il est très improbable qu'on puisse trouver un algorithme le résolvant de manière exacte en un temps polynomial. De plus, ce problème reste NP-difficile même pour certaines classes de graphes particulières.

Nous devons donc choisir une méthode de résolution appropriée, suivant le contexte :

1. Résolution exacte.
   Solution optimale, mais temps de calcul important.
2. Résolution approchée.
   Solution dégradée, mais temps de calcul acceptable.

Le problème du vertex cover

Algorithmes de coupe et de branchement (Branch and Bound)

1. Le nombre de solutions est fini.
2. Il est possible d'énumérer toutes ces solutions.
3. Ensuite, il suffit de choisir celle qui optimise un critère voulu.
4. Mais un nombre gigantesque de solutions à énumérer.

Le problème du vertex cover

Algorithmes de coupe et de branchement (Branch and Bound)

Definition (Les méthodes de type branch and bound)

Les procédures par évaluation et séparation progressive consistent à énumérer ces solutions d'une manière intelligente, en exploitant la structure du problème et son contexte. En particulier, cette technique va éliminer des solutions partielles qui ne mènent pas à une solution optimale. Il devient donc possible d'effectuer une résolution exacte, même si dans certains cas le temps nécessaire reste prohibitif.

Le problème du vertex cover

Exemple de remarque permettant d'exploiter la nature du problème.

- **Question :** Existe-t-il une solution de taille au plus k pour le problème du vertex cover ?
- **Remarque :** Si il existe un sommet s de degré $k+1$, il est certain que ce sommet fera partie de la solution (si elle existe). Ne pas sélectionner le sommet s implique, nécessairement, de sélectionner son voisinage afin de couvrir toutes les arêtes.

Cette remarque nous permet d'éliminer directement toutes les solutions potentielles qui ne contiennent pas le sommet s.

Le problème du vertex cover

Exemple de remarque permettant d'exploiter la nature du problème.

Question

Est-ce une bonne idée d'ajouter un sommet u de degré 1, si on souhaite obtenir une solution optimale ?

Le problème du vertex cover

Exemple de remarque permettant d'exploiter la nature du problème.

Question

Est-ce une bonne idée d'ajouter un sommet u de degré 1, si on souhaite obtenir une solution optimale ?

1. Si v le voisin de u est de degré > 1 : aucun intérêt. Ajouter le sommet v à la solution permettra de couvrir l'arête (u, v), mais permettra aussi de couvrir les autres arêtes incidentes à v.
2. Si v le voisin est lui aussi de degré 1, alors il faut ajouter soit u soit v à la solution, mais pas les deux.

Exemple d'algorithme pour le problème du vertex cover

Exemple d'algorithme pour le problème du vertex cover

L'algorithme présenté ici est un algorithme « jouet ». De nombreuses améliorations sont possibles.

1. Nous allons créer un arbre binaire.
2. Chaque nœud de l'arbre va représenter une partition des sommets du graphe : ceux qui font partie de la solution, ceux qui n'en font pas partie et ceux pour lesquels une décision reste à prendre.
3. Ainsi, la racine de notre arbre contient un seul ensemble contenant l'ensemble des sommets du graphe.
4. A partir d'un nœud de notre arbre, on va choisir un sommet s du graphe pour lequel une décision reste à prendre.
5. Pour ce sommet, nous allons prendre les deux décisions possibles : l'inclure dans la solution ou non. Cela se matérialise par la création de deux nouveaux nœuds de notre arbre, l'un incluant s dans la solution courante, l'autre excluant s de la solution.
6. Il suffit ensuite de recommencer l'opération sur les deux nouveaux nœuds.

Exemple d'algorithme pour le problème du vertex cover

Exemple d'algorithme pour le problème du vertex cover

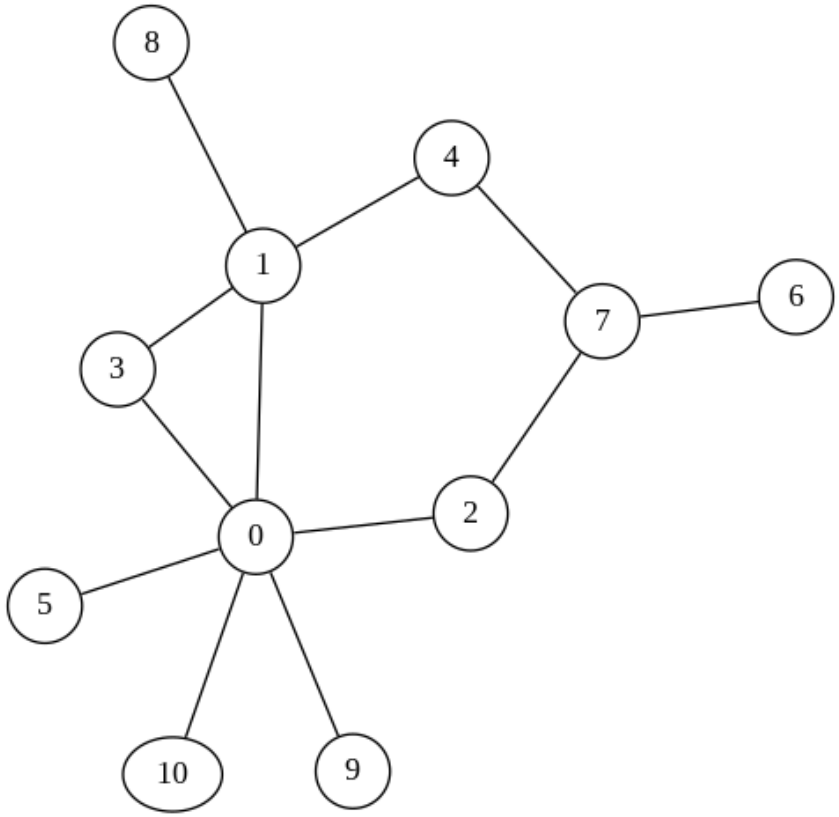
Conséquence d'une décision

Notez bien que ne pas inclure un sommet dans une solution implique nécessairement d'ajouter tous les sommets de son voisinage dans la solution. Dans le cas contraire, certaines arêtes ne seraient pas couvertes.

Exemple de déroulement de l'algorithme

Graphe de départ

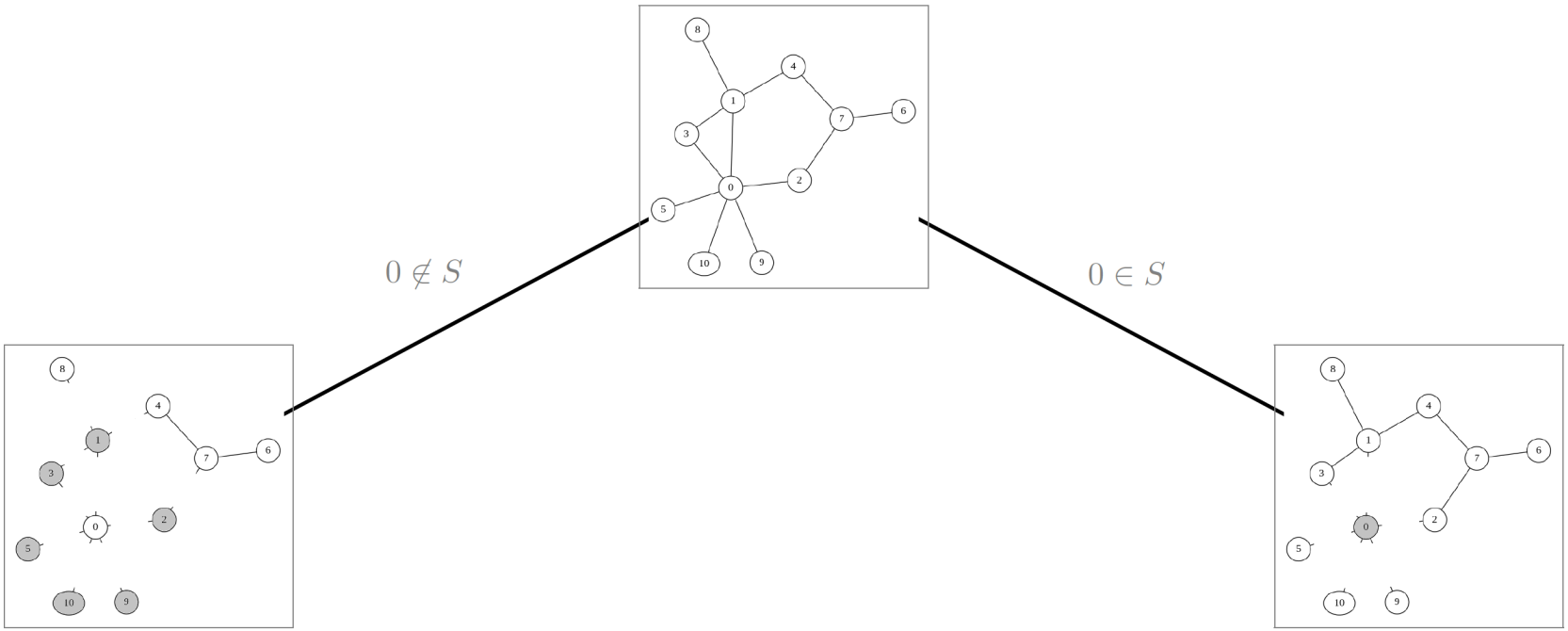
Nous allons appliquer cet algorithme sur le graphe suivant :



Résolution exacte

Exemple. Étape 1

Tout d'abord, effectuons un choix concernant le sommet 0 :



Résolution exacte

Exemple. Étape 1

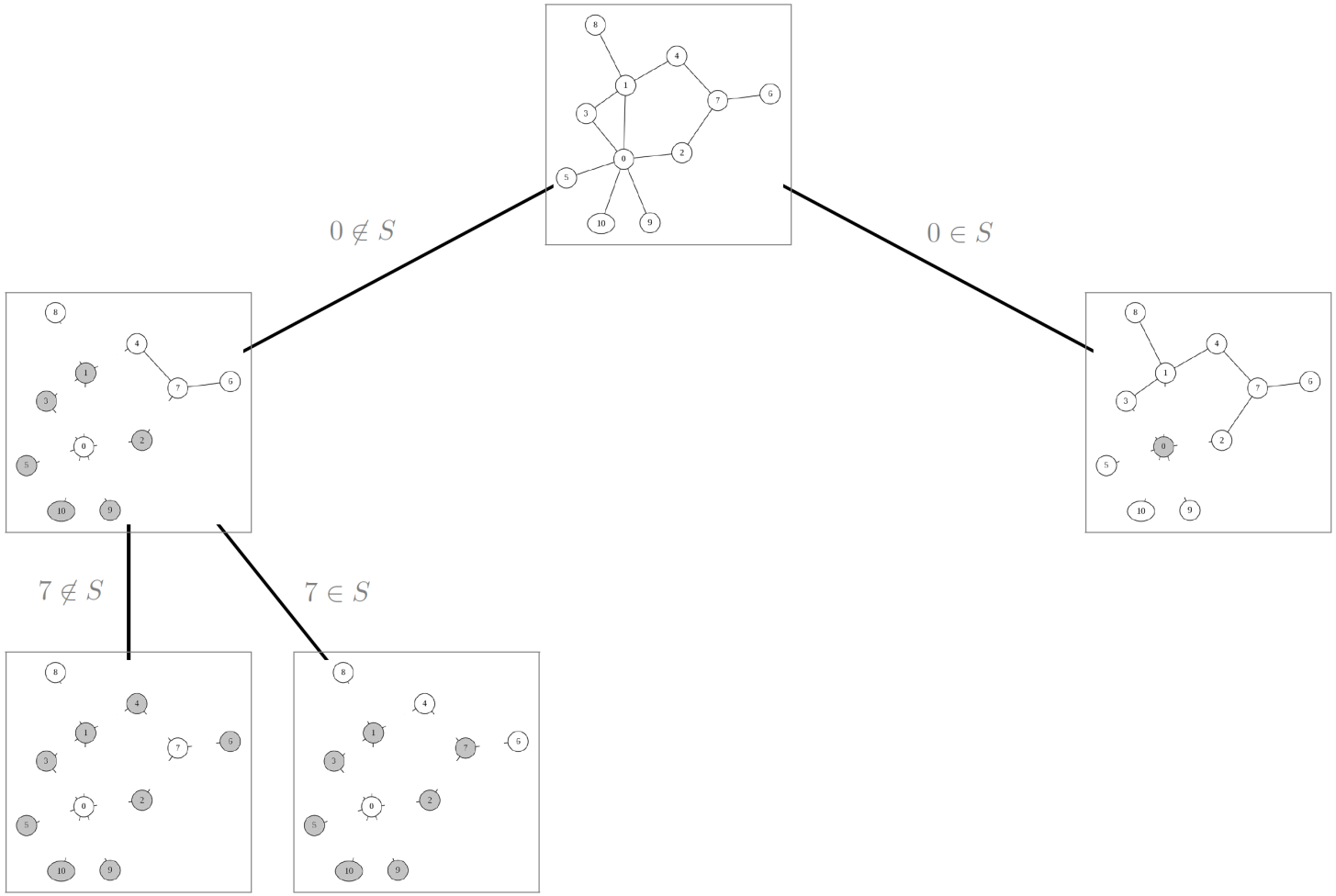
Tout d'abord, effectuons un choix concernant le sommet 0 :

- Soit 0 est inclus dans la solution. L'ensemble des arêtes incidentes à 0 sont couvertes.
- Soit 0 n'est pas inclus dans la solution. Pour couvrir les arêtes incidentes à 0 il est donc nécessaire de sélectionner les sommets $1, 2, 3, 5, 9$ et 10.

Pour chacun de ces choix, il est nécessaire de ré appliquer notre algorithme.

Résolution exacte

Exemple. Étape 2. Considérons le sommet 7 lorsque 0 n'est pas dans la solution.



Résolution exacte

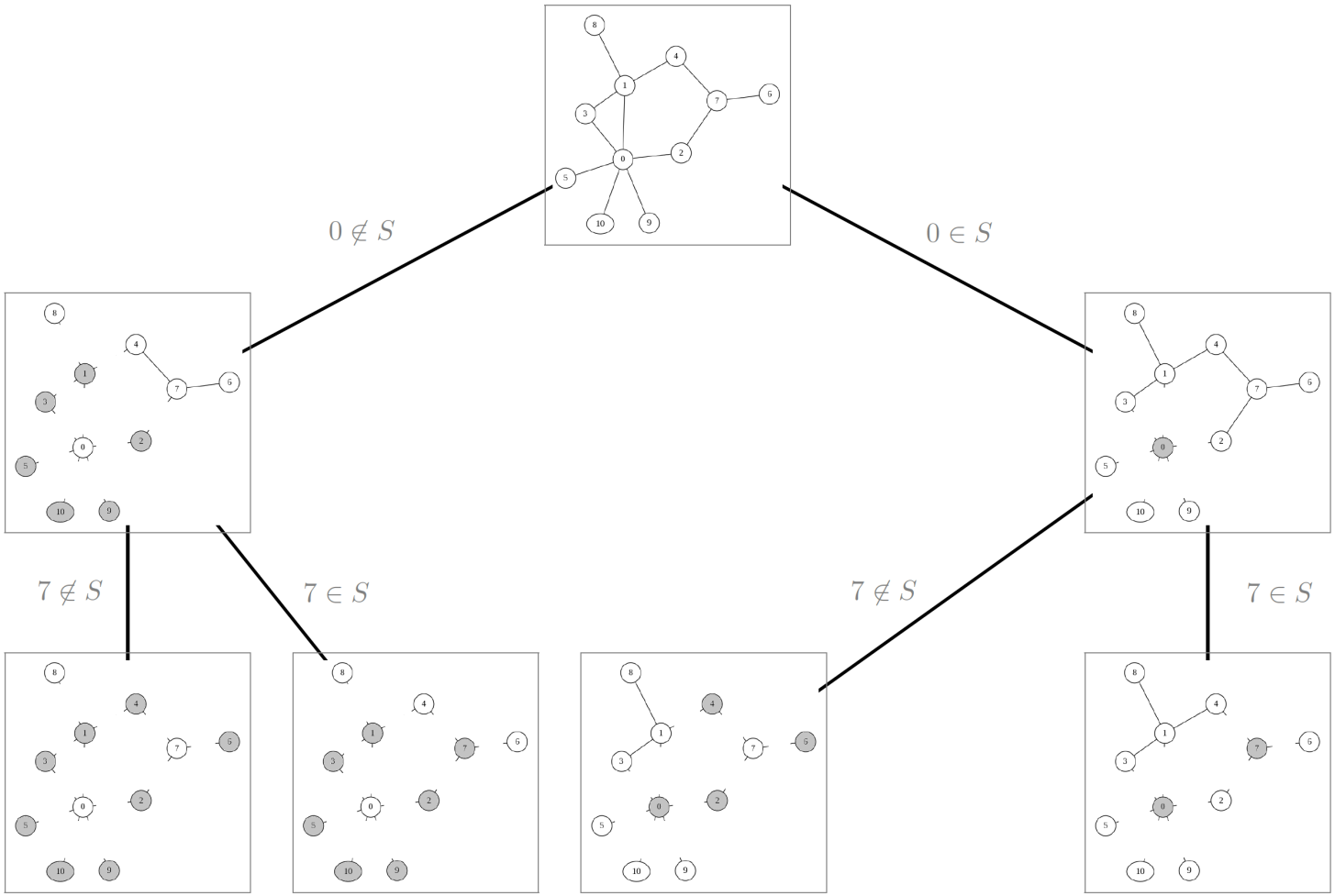
Exemple. Étape 2. Considérons le sommet 7 lorsque 0 n'est pas dans la solution.

Tout d'abord, effectuons un choix concernant le sommet 7 :

- Soit 7 est inclus dans la solution. L'ensemble des arêtes incidentes à 7 sont couvertes. Toutes les arêtes sont couvertes. Fin de l'algorithme.
- Soit 7 n'est pas inclus dans la solution. Pour couvrir les arêtes incidentes à 7 il est donc nécessaire de sélectionner les sommets 4 et 6. Toutes les arêtes sont couvertes. Fin de l'algorithme.

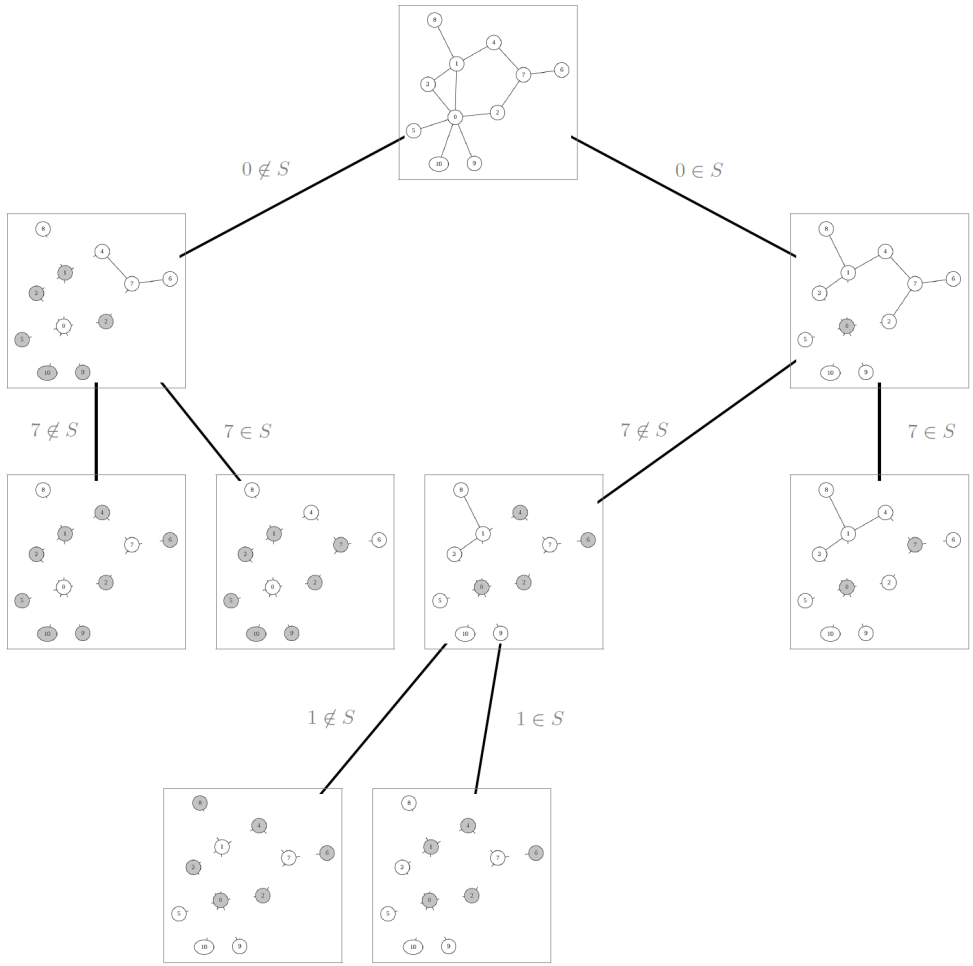
Résolution exacte

Exemple. Étape 3. Considérons le sommet 7 lorsque 0 est dans la solution.



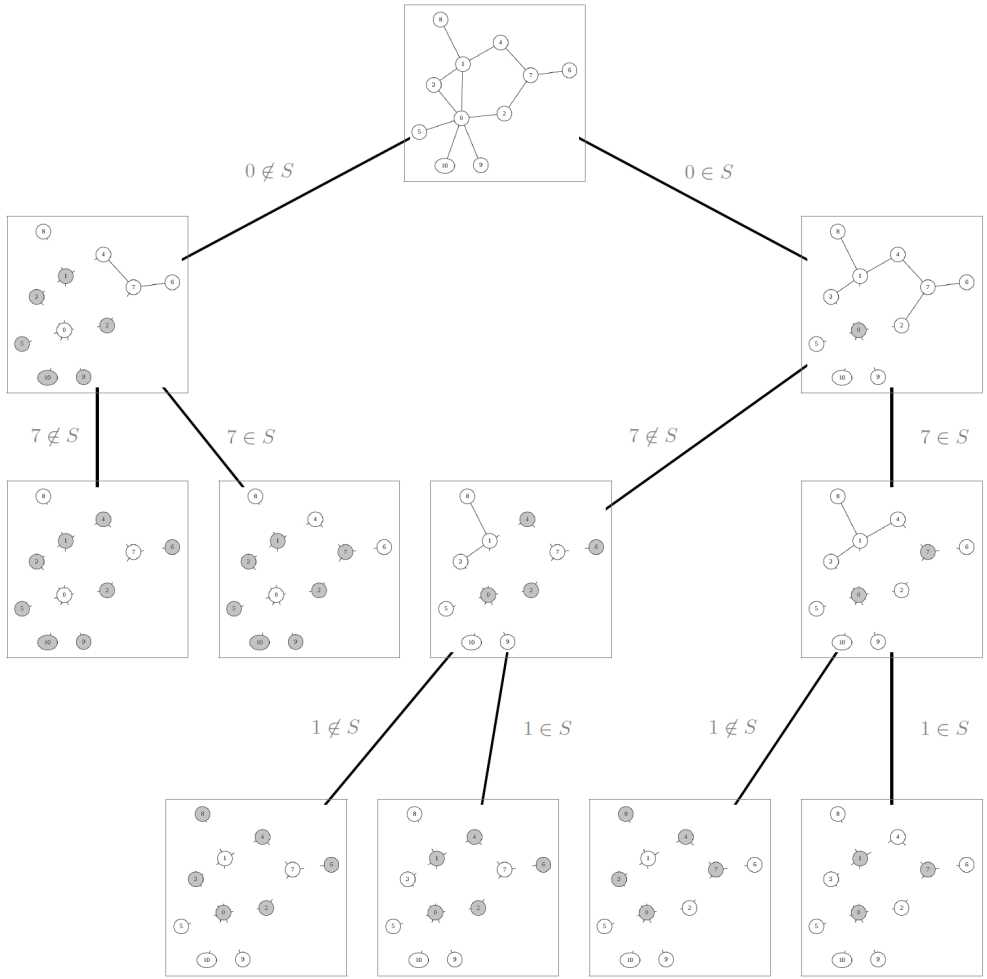
Résolution exacte

Exemple. Étape 4. Considérons le sommet 1 lorsque $0 \in S$ et $7 \notin S$.



Résolution exacte

Exemple. Étape 5. Considérons le sommet 1 lorsque $0 \in S$ et $7 \in S$.



Résolution exacte

Exemple. Déterminer la meilleure solution.

Nous venons d'énumérer toutes les solutions possibles. Cette méthode, plus efficace que l'énumération des $2^1 1$ sous-ensembles de sommets possibles reste exponentielle.

Solutions rencontrées

- $1, 2, 3, 4, 5, 6, 9, 10$: taille 8
- $1, 2, 3, 5, 7, 9, 10$: taille 7
- $0, 2, 3, 4, 6, 8$: taille 6
- $0, 1, 2, 4, 6$: taille 5
- $0, 3, 4, 7, 8$: taille 5
- $0, 1, 7$: taille 3

La solution de taille minimum est la suivante : $0, 1, 7$.

Résolution approchée

Puisqu'il est improbable que l'on puisse trouver un algorithme polynomial résolvant de façon exacte un problème NP-difficile, l'utilisation d'algorithmes polynomiaux retournant des solutions non optimales s'est généralisée. Cependant, il est souhaitable que la qualité des solutions retournées ne soit pas trop dégradée et on demande donc à ces algorithmes de donner une garantie sur leurs performances.

Résolution approchée

Definition (Algorithme $\alpha-$approché)

Pour un problème de minimisation, un algorithme $\alpha-$approché est un algorithme qui s'exécute (généralement) en un temps qui est polynomial en la taille de l'instance à traiter et qui retourne une solution dont on garantit que la valeur ne dépasse pas α fois la valeur de la solution optimale.

Autrement dit, si on note *Opt* la valeur de la solution optimale, et *Sol* la valeur de la solution retournée par l'algorithme, on a $Opt \leq Sol \leq \alpha Opt$.

A partir de cette définition, l'objectif devient donc de trouver, pour un problème donné, l'algorithme possédant le meilleur rapport d'approximation en pire cas possible.

Résolution approchée

L'algorithme Edge Deletion (ED)

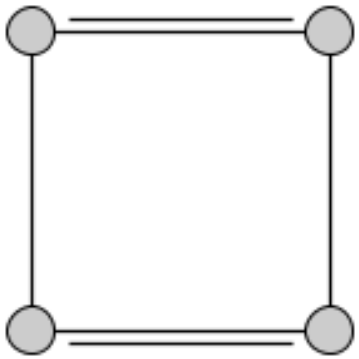
Les sommets d'un couplage maximal forment une couverture des sommets. Il retourne un vertex cover dont la taille est au plus $2 \cdot |OPT|$ (temps linéaire au nombre d'arêtes).

Algorithm 1: Edge Deletion

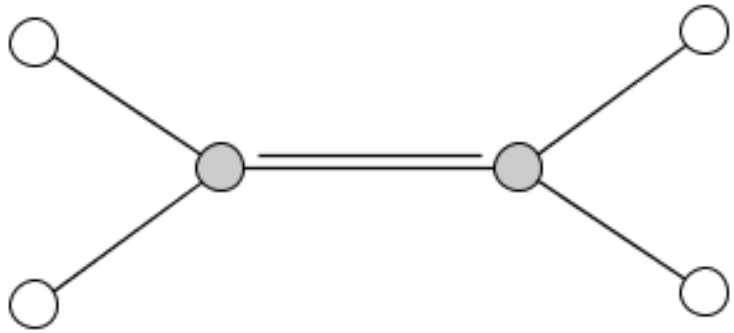
Data: Un graphe $G = (V, E)$
Result: Un vertex cover de G
$C \leftarrow \emptyset$;
while $E \neq \emptyset$ **do**
    sélectionner $uv \in E$;
    $C \leftarrow C \cup \{u, v\}$;
    $V \leftarrow V - \{u, v\}$;
end
return C;

Résolution approchée

L'algorithme Edge Deletion (ED)



Exemple de graphe pour lequel **Edge Deletion** atteint son rapport d'approximation en pire cas



Exemple de graphe pour lequel **Edge Deletion** peut retourner une solution optimale

Résolution approchée

Un algorithme non déterministe

Dans la boucle de l'algorithme **Edge Deletion**, le choix de l'arête à ajouter au couplage peut s'effectuer de plusieurs façons :

- Choix aléatoire uniforme d'une arête parmi l'ensemble des arêtes possibles.
- Choix aléatoire non-uniforme d'une arête parmi l'ensemble des arêtes possibles.
- Choix d'un sommet non isolé, puis choix de l'un de ses voisins.

Cela signifie que deux exécution du même algorithme peuvent retourner une solution différente, et même de taille différente.

Résolution approchée

L'algorithme Maximum Degree Greedy (MDG)

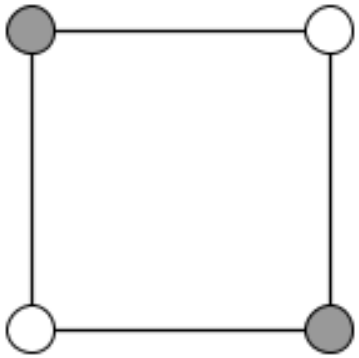
Cet algorithme va couvrir le maximum d'arêtes possibles à chaque étape en sélectionnant le sommet de degré maximum et en le retirant du graphe ainsi que toutes les arêtes qui lui sont incidentes.

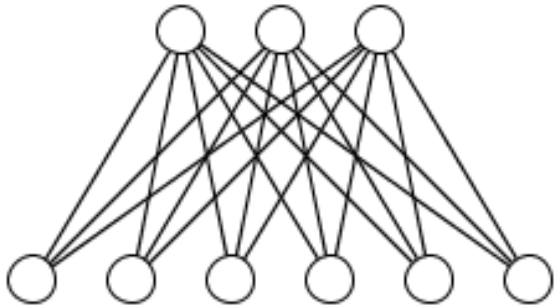
Algorithm 2: Maximum Degree Greedy

Data: Un graphe $G = (V, E)$
Result: Un vertex cover de G
$C \leftarrow \emptyset$;
while $E \neq \emptyset$ **do**
    sélectionner un sommet u de degré maximum;
    $V \leftarrow V - \{u\}$;
    $C \leftarrow C \cup \{u\}$;
end
return C;

Résolution approchée

L'algorithme Maximum Degree Greedy (MDG)

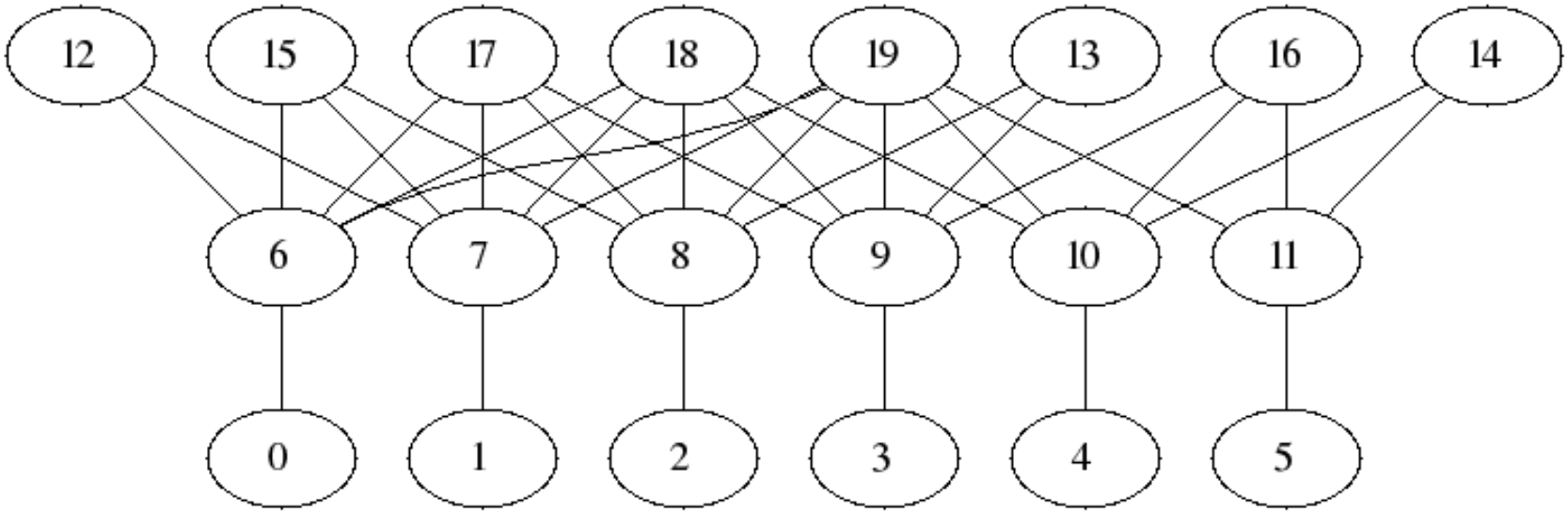




Exemples de graphes pour lesquels **MDG** peut retourner une solution optimale

Résolution approchée

L'algorithme Maximum Degree Greedy (MDG)



Exemple de graphe pour lequel **MDG** atteint son rapport d'approximation en pire cas.

Résolution approchée

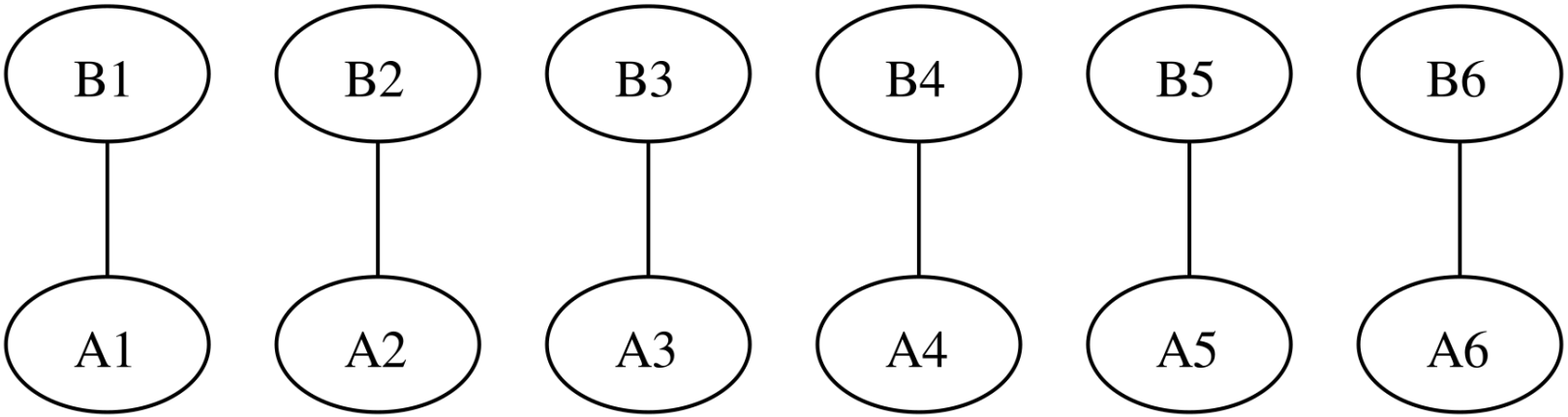
Construction des graphes anti-MDG

Nous allons voir comment construire de tels graphes, qui permettent d'atteindre le rapport d'approximation en pire cas pour l'algorithme MDG :

- On note k un entier qui représente la dimension du graphe Anti-MDG (dans notre exemple, on a $k = 6$).
- Ce graphe, que l'on nommera G, possède trois ensembles de sommets A, B et C. On note X_i le $i$ème sommet de l'ensemble X, avec $X \in \{A, B, C\}$.
- Au départ, les ensembles A et B contiennent exactement k sommets chacun et l'ensemble C ne contient aucun sommet.
- On crée un couplage entre les sommets de A et ceux de B : chaque sommet B_i est relié au sommet A_i, pour $i = 1, \ldots, k$.

Résolution approchée

Construction des graphes anti-MDG



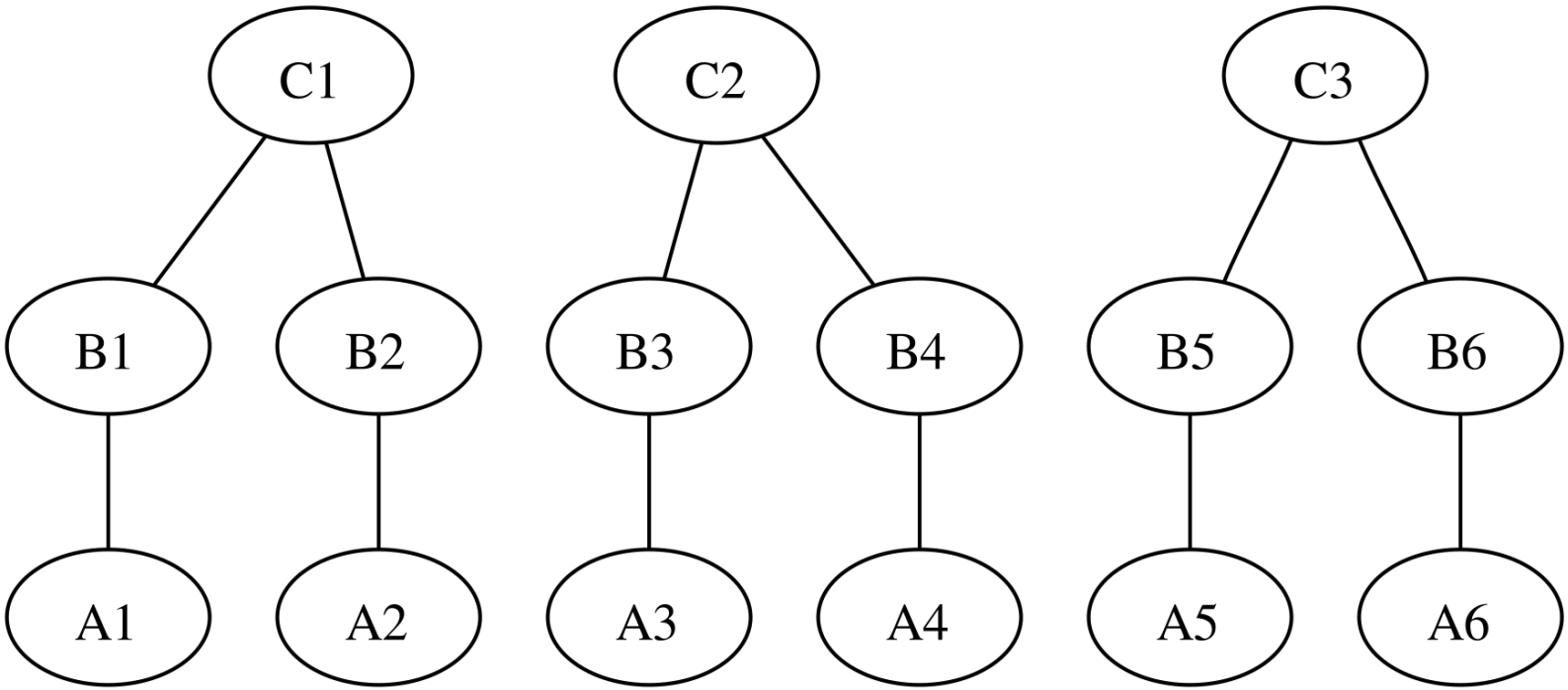
Résolution approchée

Construction des graphes anti-MDG

- Nous allons ensuite ajouter $\left\lfloor \frac{k}{2} \right\rfloor$ sommets de degré 2 dans C.
- Le premier sera relié aux deux premiers sommets de B, le deuxième aux deux suivants et ainsi de suite.
- Plus formellement, nous allons relier les sommets B_{2i+1} et B_{2i+2} au sommet C_{i+1}, pour i allant de 0 à $\left\lfloor \frac{k}{2} \right\rfloor - 1$

Résolution approchée

Construction des graphes anti-MDG



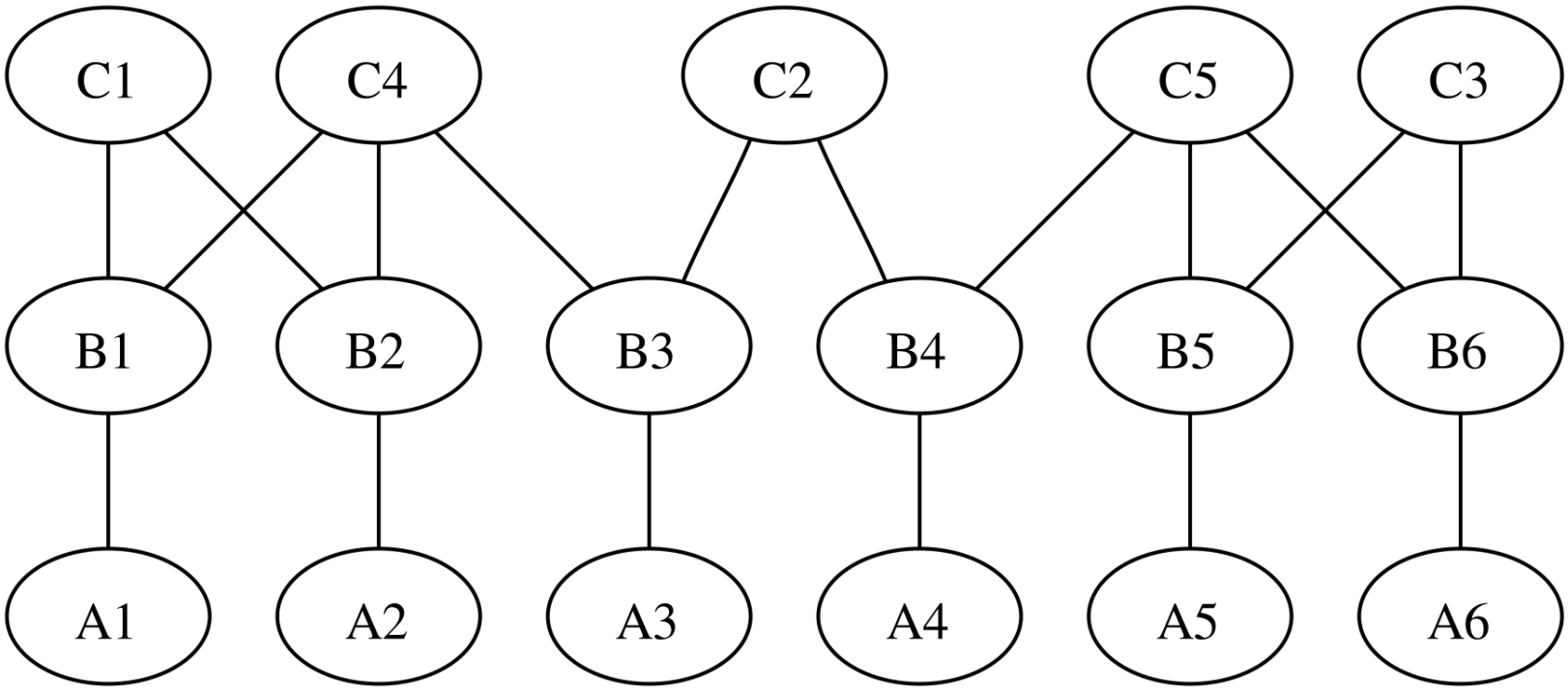
Résolution approchée

Construction des graphes anti-MDG

- Nous allons ajouter $\left\lfloor \frac{k}{3} \right\rfloor$ sommets de degré 3 dans C.
- Le premier sera relié aux trois premiers sommets de B, le deuxième aux trois suivants et ainsi de suite.
- Plus formellement, on relie les sommets B_{3i+1}, B_{3i+2} et B_{3i+3} au sommet C_{n+i+1}, pour i allant de 0 à $\left\lfloor \frac{k}{3} \right\rfloor - 1$, avec $n = \left\lfloor \frac{k}{2} \right\rfloor$ le nombre de sommets présents dans C avant l'ajout des sommets de degré 3.

Résolution approchée

Construction des graphes anti-MDG



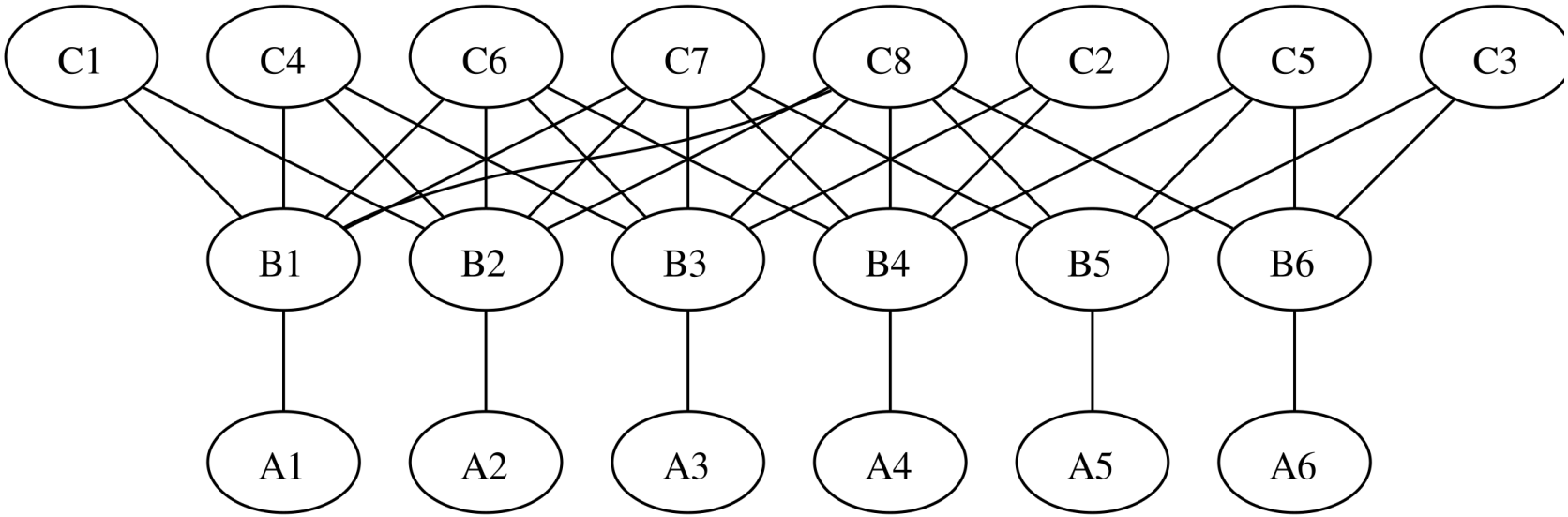
Résolution approchée

Construction des graphes anti-MDG

- On recommence une nouvelle fois l'opération mais en ajoutant cette fois-ci $\left\lfloor \frac{k}{4} \right\rfloor$ sommets de degré 4 dans C. Notons que dans notre exemple, nous n'ajoutons qu'un seul sommet.
- On poursuit de la même façon jusqu'à ce qu'on relie les k sommets de B à un unique sommet de C.

Résolution approchée

Construction des graphes anti-MDG



Résolution approchée

Construction des graphes anti-MDG

- Il existe une exécution de l'algorithme *MDG* qui sélectionne tous les sommets de l'ensemble C, puis sélectionne k sommets parmi ceux de l'ensemble $B \cup A$.
- Lors de la construction du graphe, de l'étape 2 jusqu'à la dernière étape, il existe un sommet de degré maximum dans l'ensemble C. MDG peut donc sélectionner le sommet de degré k de l'ensemble C, puis le sommet de degré $k-1$ et ainsi de suite, repassant par chacune des étapes de construction dans l'ordre inverse.
- Pour les étapes où nous avons ajouté plusieurs sommets, l'algorithme peut les sélectionner à la suite et dans un ordre quelconque.

Résolution approchée

Construction des graphes anti-MDG

Lorsqu'il n'y a plus de sommet dans l'ensemble C, MDG va devoir couvrir chaque arête du couplage de la première étape et va donc retourner une solution de taille
$\left(k + \left\lfloor \frac{k}{2} \right\rfloor + \left\lfloor \frac{k}{3} \right\rfloor + \left\lfloor \frac{k}{4} \right\rfloor + \ldots + 1\right) \approx H(k) \cdot k = H(\Delta) \cdot |OPT|$,
avec Δ le degré maximum du graphe et H la série harmonique.

Le problème de l'ensemble indépendant maximum(stable)

Definition (Maximum Independent Set (MIS))

Étant donné un graphe $G = (V, E)$, avec V l'ensemble de ses sommets et E l'ensemble de ses arêtes, le problème de l'ensemble indépendant de taille maximum consiste à trouver le plus grand ensemble possible $I \subseteq V$ tel que pour tout couple de sommets $(a, b) \in I^2$, l'arête $(a, b) \notin E$. On dit que I est un ensemble indépendant maximum de G.

Équivalence. Il est intéressant de remarquer que $V - I$ est un vertex cover optimal. Ainsi, résoudre le problème de l'ensemble indépendant maximum revient à rechercher un vertex cover optimal.

Le problème de la clique maximum

Definition (Clique maximum)

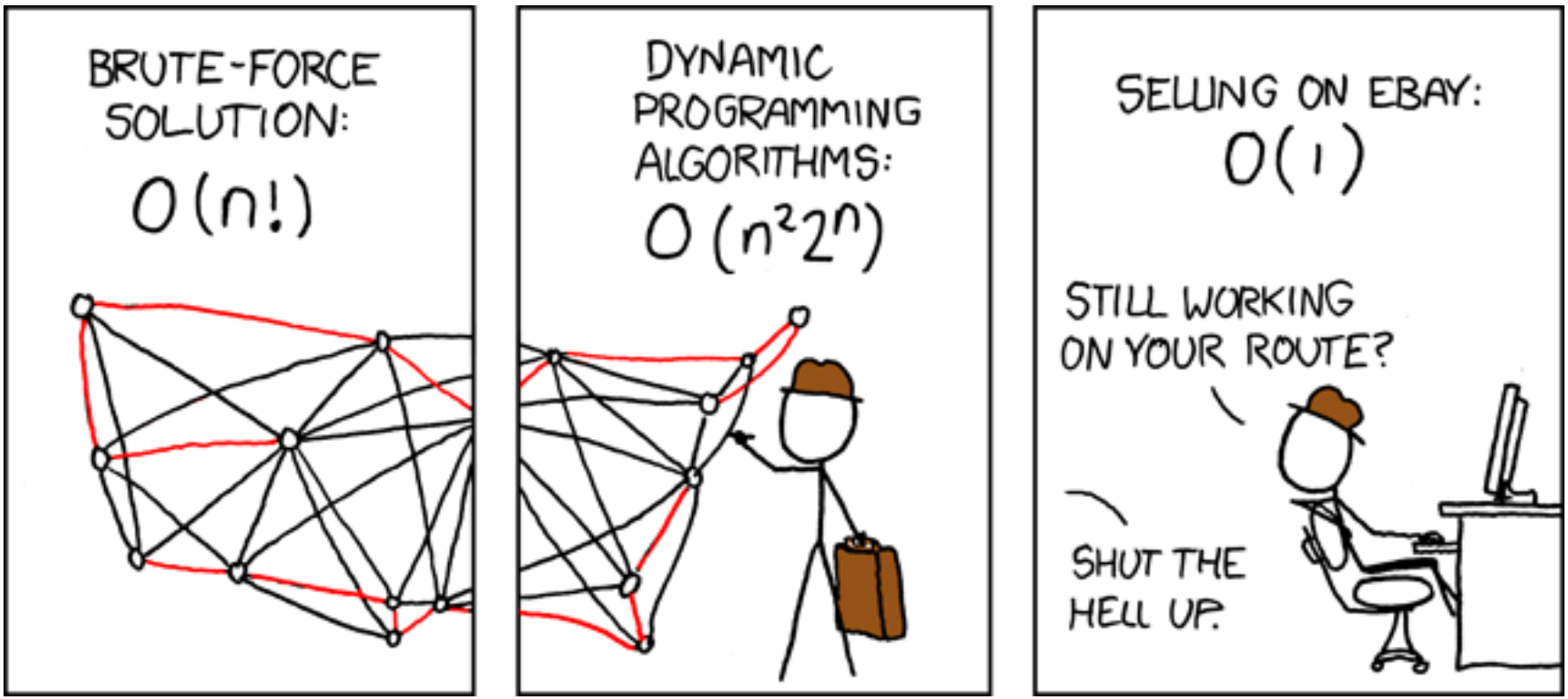
Étant donné un graphe $G = (V, E)$, avec V l'ensemble de ses sommets et E l'ensemble de ses arêtes, le problème de la clique maximum consiste à trouver le plus grand ensemble possible $C \subseteq V$ tel que pour tout couple de sommets $(a, b) \in C^2$, l'arête $(a, b) \in E$. On dit que C est une clique maximum de G.

Équivalence. Résoudre le problème de la clique maximum revient à rechercher un ensemble indépendant de taille maximum dans le graphe complémentaire.

Le problème du voyageur de commerce

Un commercial représentant un laboratoire pharmaceutique doit présenter une nouvelle molécule aux différents hôpitaux de sa région. Chaque hôpital se trouve dans une ville différente. Afin d'optimiser son trajet (et donc son temps, ou son argent ou autre), il souhaite ne passer qu'une et une seule fois par chacune des villes concernées et revenir à son point de départ tout en minimisant la distance totale parcourue. Trouver une solution à ce problème est une chose très difficile en général.

Le problème du voyageur de commerce



Le problème du voyageur de commerce

L'un des meilleurs algorithmes exact, proposé par Held et Karpet basé sur la programmation dynamique possède une complexité en $O(n^2 2^n)$. A titre d'information, 2^{100} est un nombre plus grand que le nombre de secondes qui nous séparent du Big Bang jusqu'à aujourd'hui.

Le problème des 4 couleurs

Étant donné un graphe, attribuer une couleur à chacun de ses sommets de telle sorte que deux sommets reliés par une arête soient de couleur différente. On cherche, bien entendu, à utiliser un nombre minimum de couleurs. Ce nombre minimum de couleurs est aussi appelé nombre chromatique.

Le problème des 4 couleurs

Definition (Graphe planaire)

Un graphe planaire est un graphe pouvant être dessiné sans qu'aucune arêtes n'en croise une autre.

Par exemple, la carte du monde peut-être représentée par un graphe planaire. Chaque pays est représenté par un sommet. Deux sommets sont voisins si et seulement si les deux pays partagent une frontière.

Le problème des 4 couleurs

Le théorème des 4 couleurs.

Ce théorème, très célèbre, indique que toute carte planaire peut être coloriée avec au plus 4 couleurs distinctes de telle sorte qu'aucune zone d'une couleur n'ait de frontière commune (différente d'un point) avec une autre zone de la même couleur.

Le problème des 4 couleurs

La première preuve informatisée.

Ce qui est intéressant avec la preuve de ce théorème, c'est qu'il s'agit d'une des premières preuves faisant intervenir un ordinateur. En effet, les mathématiques ont permis de limiter le nombre de cas à analyser, mais ce nombre (1478) restait trop important pour être traité par un humain. L'outil informatique a ainsi été utilisé pour traiter ces différents cas (plus de 1200 heures de calcul), et démontrer ainsi la validité du théorème. Cependant, cette preuve à divisé la communauté scientifique, car il aurait fallu prouver que le programme utilisé se comportait bien comme prévu. C'est chose faite depuis 2005 avec une version entièrement formalisée, formulée avec Coq.

Le problème des 4 couleurs