Méthodes d’optimisation

BUT Info 2e année

Florent Foucaud
Dipayan Chakraborty, Malika More, Adrien Wohrer

IUT CLERMONT AUVERGNE
Aurillac - Clermont-Ferrand - Le Puy-en-Velay
Montluçon - Moulins - Vichy

2022-2023

Heuristiques

Rappel : on ne peut pas toujours résoudre efficacement n’importe quel problème d’optimisation.

Comment faire ?

Heuristiques

Rappel : on ne peut pas toujours résoudre efficacement n'importe quel problème d'optimisation.

Comment faire ?

Définition (Algorithme heuristique)

Algorithme qui résout un problème donné (en temps efficace), mais qui ne calcule pas forcément la solution optimale.

Heuristiques

Rappel : on ne peut pas toujours résoudre efficacement n'importe quel problème d'optimisation.

Comment faire ?

Définition (Algorithme heuristique)

Algorithme qui résout un problème donné (en temps efficace), mais qui ne calcule pas forcément la solution optimale.

Définition (Algorithme d'approximation)

Algorithme heuristique avec une garantie sur la taille de la solution calculée :

- au plus un facteur multiplicatif constant par rapport à l'optimal (si problème de minimisation) ;
- au moins un facteur multiplicatif constant par rapport à l'optimal (si problème de maximisation).

Heuristiques

Rappel : on ne peut pas toujours résoudre efficacement n'importe quel problème d'optimisation.

Comment faire ?

Définition (Algorithme heuristique)

Algorithme qui résout un problème donné (en temps efficace), mais qui ne calcule pas forcément la solution optimale.

Définition (Algorithme d'approximation)

Algorithme heuristique avec une garantie sur la taille de la solution calculée :

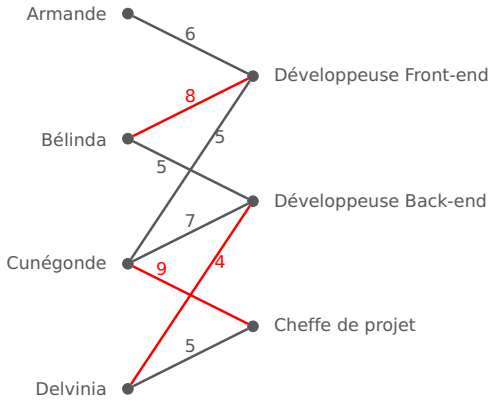
- au plus un facteur multiplicatif constant par rapport à l'optimal (si problème de minimisation) ;
- au moins un facteur multiplicatif constant par rapport à l'optimal (si problème de maximisation).

Exemple :
solution de taille au plus 2 fois l'optimum pour un problème de minimisation
→ On parle d'algorithme d'approximation à facteur 2

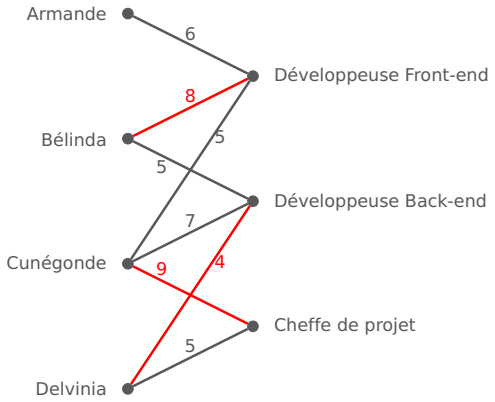
Couplage dans un graphe biparti

Problème : sélectionner un ensemble d'arêtes d'un graphe biparti $G = (V, E)$, de telle façon que chaque sommet soit touché au plus une fois.

Objectif : maximiser le poids total du couplage



Couplage dans un graphe biparti



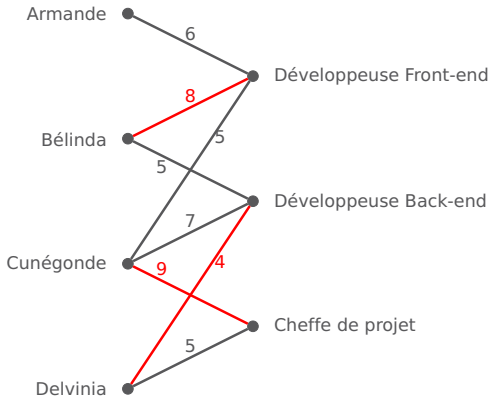
Problème : sélectionner un ensemble d'arêtes d'un graphe biparti $G = (V, E)$, de telle façon que chaque sommet soit touché au plus une fois.

Objectif : maximiser le poids total du couplage

On écrit le PLNE suivant ($x_e = 1$ si l'arête e est sélectionnée, sinon $x_e = 0$) :

$$\begin{aligned}
&\text{maximiser :} && \sum_{e \in E} p_e x_e \\
&\text{tel que :} && \sum_{e=uv \in E} x_e \leq 1 \quad \forall v \in V \\
& && x_e \leq 1 \quad \forall e \in E \\
& && x_e \geq 0 \quad \forall e \in E \\
& && x_e \in \mathbb{N} \quad \forall e \in E
\end{aligned}$$

Couplage dans un graphe biparti



Problème : sélectionner un ensemble d'arêtes d'un graphe biparti $G = (V, E)$, de telle façon que chaque sommet soit touché au plus une fois.

Objectif : maximiser le poids total du couplage

On écrit le PLNE suivant ($x_e = 1$ si l'arête e est sélectionnée, sinon $x_e = 0$) :

$$\text{maximiser :} \quad \sum_{e \in E} p_e x_e$$

$$\text{tel que :} \quad \sum_{e=uv \in E} x_e \leq 1 \quad \forall v \in V$$

$$x_e \leq 1 \quad \forall e \in E$$

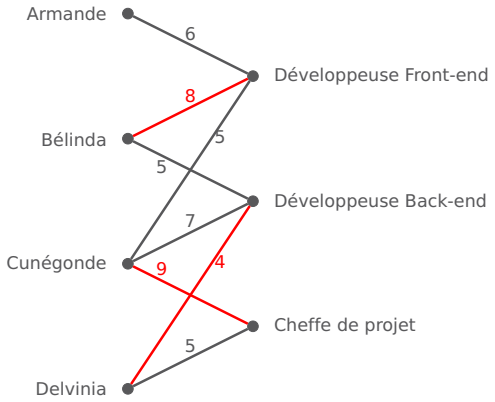
$$x_e \geq 0 \quad \forall e \in E$$

$$x_e \in \mathbb{N} \quad \forall e \in E$$

Relaxation linéaire d'un PLNE :

PL avec les mêmes contraintes et même objectif, mais valeurs non-entières

Couplage dans un graphe biparti



Problème : sélectionner un ensemble d'arêtes d'un graphe biparti $G = (V, E)$, de telle façon que chaque sommet soit touché au plus une fois.

Objectif : maximiser le poids total du couplage

On écrit le PLNE suivant ($x_e = 1$ si l'arête e est sélectionnée, sinon $x_e = 0$) :

$$\text{maximiser : } \sum_{e \in E} p_e x_e$$

$$\text{tel que : } \sum_{e=uv \in E} x_e \leq 1 \quad \forall v \in V$$

$$x_e \leq 1 \quad \forall e \in E$$

$$x_e \geq 0 \quad \forall e \in E$$

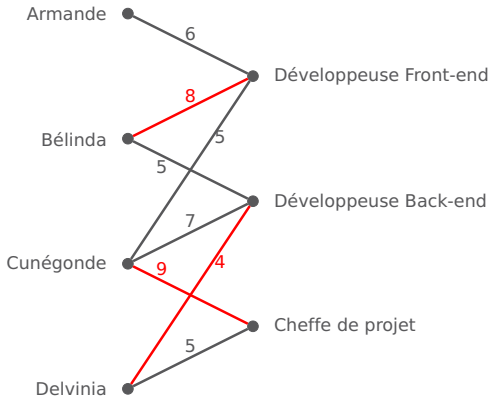
$$x_e \in \mathbb{N} \quad \forall e \in E$$

Relaxation linéaire d'un PLNE :
PL avec les mêmes contraintes et même objectif, mais valeurs non-entières

Théorème

Pour un graphe biparti, les sommets du polytope associé à la relaxation linéaire du PLNE du couplage ont tous des coordonnées entières.

Couplage dans un graphe biparti



Problème : sélectionner un ensemble d'arêtes d'un graphe biparti $G = (V, E)$, de telle façon que chaque sommet soit touché au plus une fois.

Objectif : maximiser le poids total du couplage

On écrit le PLNE suivant ($x_e = 1$ si l'arête e est sélectionnée, sinon $x_e = 0$) :

$$\text{maximiser :} \quad \sum_{e \in E} p_e x_e$$

$$\text{tel que :} \quad \sum_{e=uv \in E} x_e \leq 1 \quad \forall v \in V$$

$$x_e \leq 1 \quad \forall e \in E$$

$$x_e \geq 0 \quad \forall e \in E$$

$$x_e \in \mathbb{N} \quad \forall e \in E$$

Relaxation linéaire d'un PLNE :
PL avec les mêmes contraintes et même objectif, mais valeurs non-entières

Théorème

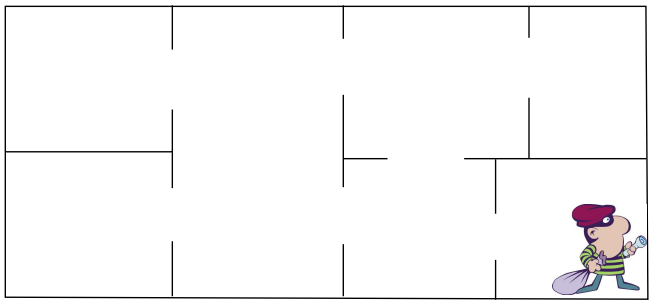
Pour un graphe biparti, les sommets du polytope associé à la relaxation linéaire du PLNE du couplage ont tous des coordonnées entières.

Rappel : on peut trouver la solution optimale d'un PL en temps polynomial
→ on obtient "gratuitement" la solution optimale de ce PLNE en temps polynomial !

Surveillance d’un bâtiment : couverture par sommets

Problème : couvrir les couloirs d’un bâtiment avec des détecteurs de mouvement.

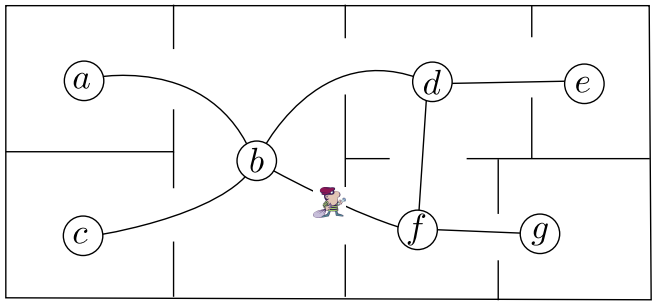
Objectif : minimiser le nombre de détecteurs



Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement.

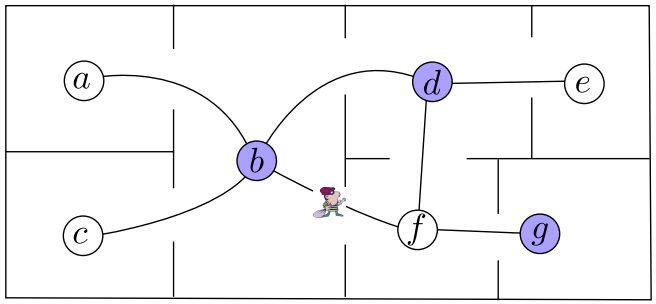
Objectif : minimiser le nombre de détecteurs



Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement.

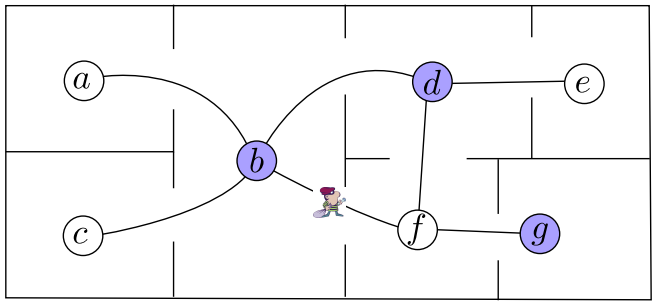
Objectif : minimiser le nombre de détecteurs



Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement.

Objectif : minimiser le nombre de détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$.

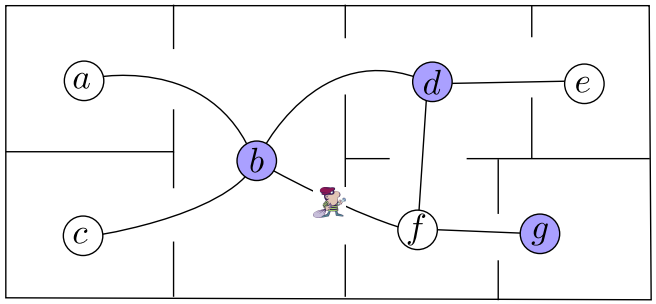
On écrit le PL en nombres entiers (PLNE) suivant :

Une variable x_v pour chaque sommet v : $x_v = 1$ si on a un détecteur sur v, 0 sinon.

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement.

Objectif : minimiser le nombre de détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$.

On écrit le PL en nombres entiers (PLNE) suivant :

Une variable x_v pour chaque sommet v : $x_v = 1$ si on a un détecteur sur v, 0 sinon.

minimiser : $x_a + x_b + x_c + x_d + x_e + x_f + x_g$

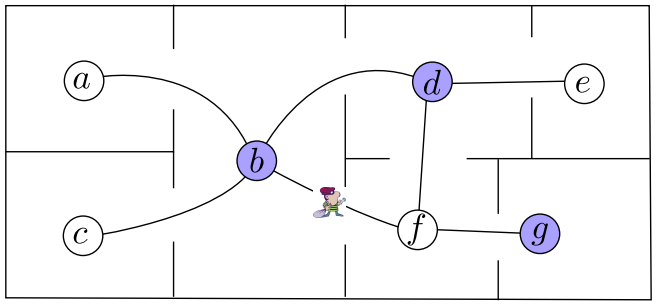
tel que :

$$
\begin{array}{rcl}
x_a + x_b & \geq & 1 \\
x_b + x_c & \geq & 1 \\
x_b + x_d & \geq & 1 \\
x_b + x_f & \geq & 1 \\
x_d + x_f & \geq & 1 \\
x_d + x_e & \geq & 1 \\
x_f + x_g & \geq & 1 \\
x_a, \ldots, x_g & \leq & 1 \\
x_a, \ldots, x_g & \geq & 0 \\
x_a, \ldots, x_g & \in & \mathbb{N}
\end{array}
$$

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement.

Objectif : minimiser le nombre de détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$.

On écrit le PL en nombres entiers (PLNE) suivant :

Une variable x_v pour chaque sommet v : $x_v = 1$ si on a un détecteur sur v, 0 sinon.

$$\begin{aligned} \text{minimiser :} \quad & \sum_{v \in V} x_v \\ \text{tel que :} \quad & x_u + x_v \geq 1 \quad \forall uv \in E \\ & x_v \leq 1 \quad \forall v \in V \\ & x_v \geq 0 \quad \forall v \in V \\ & x_v \in \mathbb{N} \quad \forall v \in V \end{aligned}$$

Relaxation linéaire et algorithme d’approximation

Méthode 1 : résoudre le PLNE exactement par la méthode “brancher et borner”

Relaxation linéaire et algorithme d'approximation

Méthode 1 : résoudre le PLNE exactement par la méthode "brancher et borner"

Méthode 2 : résoudre la relaxation linéaire (PL normal, pas de nombres entiers)

$$\begin{aligned} \text{minimiser :} \quad & \sum_{v \in V} x_v \\ \text{tel que :} \quad & x_u + x_v \geq 1 \quad \forall uv \in E \\ & x_v \leq 1 \quad \forall v \in V \\ & x_v \geq 0 \quad \forall v \in V \end{aligned}$$

Relaxation linéaire et algorithme d'approximation

Méthode 1 : résoudre le PLNE exactement par la méthode "brancher et borner"

Méthode 2 : résoudre la relaxation linéaire (PL normal, pas de nombres entiers)

$$
\begin{array}{lllll}
\text{minimiser :} & \sum_{v \in V} x_v & & & \\
\text{tel que :} & x_u + x_v & \geq & 1 & \forall uv \in E \\
& x_v & \leq & 1 & \forall v \in V \\
& x_v & \geq & 0 & \forall v \in V
\end{array}
$$

On construit une solution S en "arrondissant" :

- si $x_v \geq 0.5$, on prend v dans la solution
- si $x_v < 0.5$, on ne prend pas v

Relaxation linéaire et algorithme d'approximation

Méthode 1 : résoudre le PLNE exactement par la méthode "brancher et borner"

Méthode 2 : résoudre la relaxation linéaire (PL normal, pas de nombres entiers)

$$
\begin{array}{lllll}
\text{minimiser :} & \displaystyle\sum_{v \in V} x_v & & & \\
\text{tel que :} & x_u + x_v & \geq & 1 & \forall uv \in E \\
& x_v & \leq & 1 & \forall v \in V \\
& x_v & \geq & 0 & \forall v \in V
\end{array}
$$

On construit une solution S en "arrondissant" :

- si $x_v \geq 0.5$, on prend v dans la solution
- si $x_v < 0.5$, on ne prend pas v

Théorème

L'ensemble S obtenu est une solution valide, de taille au plus 2 fois l'optimum.

(On dit qu'il y a un écart d'intégralité d'au plus 2.)

Relaxation linéaire et algorithme d'approximation

Méthode 1 : résoudre le PLNE exactement par la méthode "brancher et borner"

Méthode 2 : résoudre la relaxation linéaire (PL normal, pas de nombres entiers)

$$\begin{array}{lll} \text{minimiser :} & \sum_{v \in V} x_v & \\ \text{tel que :} & x_u + x_v \geq 1 & \forall uv \in E \\ & x_v \leq 1 & \forall v \in V \\ & x_v \geq 0 & \forall v \in V \end{array}$$

On construit une solution S en "arrondissant" :

- si $x_v \geq 0.5$, on prend v dans la solution
- si $x_v < 0.5$, on ne prend pas v

Théorème

L'ensemble S obtenu est une solution valide, de taille au plus 2 fois l'optimum.

(On dit qu'il y a un écart d'intégralité d'au plus 2.)

Démonstration :
1) L'ensemble est bien valide, car pour toute arête uv, soit $x_u \geq 0.5$, soit $x_v \geq 0.5$.

Relaxation linéaire et algorithme d'approximation

Méthode 1 : résoudre le PLNE exactement par la méthode "brancher et borner"

Méthode 2 : résoudre la relaxation linéaire (PL normal, pas de nombres entiers)

$$\begin{array}{lll} \text{minimiser :} & \sum_{v \in V} x_v & \\ \text{tel que :} & x_u + x_v \geq 1 & \forall uv \in E \\ & x_v \leq 1 & \forall v \in V \\ & x_v \geq 0 & \forall v \in V \end{array}$$

On construit une solution S en "arrondissant" :

- si $x_v \geq 0.5$, on prend v dans la solution
- si $x_v < 0.5$, on ne prend pas v

Théorème

L'ensemble S obtenu est une solution valide, de taille au plus 2 fois l'optimum.

(On dit qu'il y a un écart d'intégralité d'au plus 2.)

Démonstration :

1) L'ensemble est bien valide, car pour toute arête uv, soit $x_u \geq 0.5$, soit $x_v \geq 0.5$.

2) Taille de S : $\leq$ nombre de sommets v tel que $x_v \geq 0.5$.

Relaxation linéaire et algorithme d'approximation

Méthode 1 : résoudre le PLNE exactement par la méthode "brancher et borner"

Méthode 2 : résoudre la relaxation linéaire (PL normal, pas de nombres entiers)

$$\begin{array}{lll} \text{minimiser :} & \sum_{v \in V} x_v & \\ \text{tel que :} & x_u + x_v \geq 1 & \forall uv \in E \\ & x_v \leq 1 & \forall v \in V \\ & x_v \geq 0 & \forall v \in V \end{array}$$

On construit une solution S en "arrondissant" :

- si $x_v \geq 0.5$, on prend v dans la solution
- si $x_v < 0.5$, on ne prend pas v

Théorème

L'ensemble S obtenu est une solution valide, de taille au plus 2 fois l'optimum.

(On dit qu'il y a un écart d'intégralité d'au plus 2.)

Démonstration :

1) L'ensemble est bien valide, car pour toute arête uv, soit $x_u \geq 0.5$, soit $x_v \geq 0.5$.

2) Taille de S : $\leq$ nombre de sommets v tel que $x_v \geq 0.5$.
$\leq$ 2 fois la somme des x_v

Relaxation linéaire et algorithme d'approximation

Méthode 1 : résoudre le PLNE exactement par la méthode "brancher et borner"

Méthode 2 : résoudre la relaxation linéaire (PL normal, pas de nombres entiers)

$$\begin{array}{lllll} \text{minimiser :} & \sum_{v \in V} x_v & & & \\ \text{tel que :} & x_u + x_v & \geq & 1 & \forall uv \in E \\ & x_v & \leq & 1 & \forall v \in V \\ & x_v & \geq & 0 & \forall v \in V \end{array}$$

On construit une solution S en "arrondissant" :

- si $x_v \geq 0.5$, on prend v dans la solution
- si $x_v < 0.5$, on ne prend pas v

Théorème

L'ensemble S obtenu est une solution valide, de taille au plus 2 fois l'optimum.

(On dit qu'il y a un écart d'intégralité d'au plus 2.)

Démonstration :

1) L'ensemble est bien valide, car pour toute arête uv, soit $x_u \geq 0.5$, soit $x_v \geq 0.5$.

2) Taille de S : $\leq$ nombre de sommets v tel que $x_v \geq 0.5$.
$\leq$ 2 fois la somme des x_v
$\leq$ 2 fois la valeur optimale de la relaxation linéaire

Relaxation linéaire et algorithme d'approximation

Méthode 1 : résoudre le PLNE exactement par la méthode "brancher et borner"

Méthode 2 : résoudre la relaxation linéaire (PL normal, pas de nombres entiers)

$$\begin{array}{lllll} \text{minimiser :} & \sum_{v \in V} x_v & & & \\ \text{tel que :} & x_u + x_v & \geq & 1 & \forall uv \in E \\ & x_v & \leq & 1 & \forall v \in V \\ & x_v & \geq & 0 & \forall v \in V \end{array}$$

On construit une solution S en "arrondissant" :

- si $x_v \geq 0.5$, on prend v dans la solution
- si $x_v < 0.5$, on ne prend pas v

Théorème

L'ensemble S obtenu est une solution valide, de taille au plus 2 fois l'optimum.

(On dit qu'il y a un écart d'intégralité d'au plus 2.)

Démonstration :

1) L'ensemble est bien valide, car pour toute arête uv, soit $x_u \geq 0.5$, soit $x_v \geq 0.5$.

2) Taille de S : $\leq$ nombre de sommets v tel que $x_v \geq 0.5$.
$\leq$ 2 fois la somme des x_v
$\leq$ 2 fois la valeur optimale de la relaxation linéaire
$\leq$ 2 fois l'optimum du PLNE (car il ne peut pas faire mieux que le PL).

Relaxation linéaire et algorithme d’approximation

Pour résumer :

- On écrit le PLNE
- On résout sa relaxation linéaire
  (par un algorithme générique pour les PL en temps efficace)
- On “arrondit” les variables obtenues
- Cela donne une solution du PLNE initial
- La taille de cette solution est au plus 2 fois l’optimum

Relaxation linéaire et algorithme d'approximation

Pour résumer :

- On écrit le PLNE
- On résout sa relaxation linéaire
  (par un algorithme générique pour les PL en temps efficace)
- On "arrondit" les variables obtenues
- Cela donne une solution du PLNE initial
- La taille de cette solution est au plus 2 fois l'optimum

Cela donne un algorithme d'approximation à facteur 2 (en temps efficace) pour le problème de couverture par sommets.

Relaxation linéaire et algorithme d'approximation

Pour résumer :

- On écrit le PLNE
- On résout sa relaxation linéaire
  (par un algorithme générique pour les PL en temps efficace)
- On "arrondit" les variables obtenues
- Cela donne une solution du PLNE initial
- La taille de cette solution est au plus 2 fois l'optimum

Cela donne un algorithme d'approximation à facteur 2 (en temps efficace) pour le problème de couverture par sommets.

Attention, cela ne marche pas pour tous les problèmes !

Algorithme glouton

Définition (algorithme glouton / greedy algorithm)

C'est un type d'algorithme qui construit une solution pas à pas, en prenant la meilleure décision à chaque étape

→ (optimum local mais pas forcémeent global).

Le Glouton ou Carcajou, un animal du grand nord

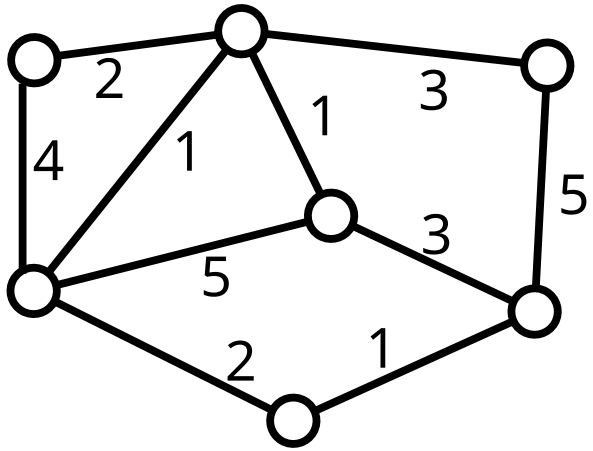
("Wolverine" en anglais)

Exemple : arbre couvrant de poids minimum

(vu au cours de Graphes en BUT1)

Problème : Construire un arbre couvrant T d'un graphe G (T touche tous les sommets)

Objectif : minimiser le poids total des arêtes de l'arbre

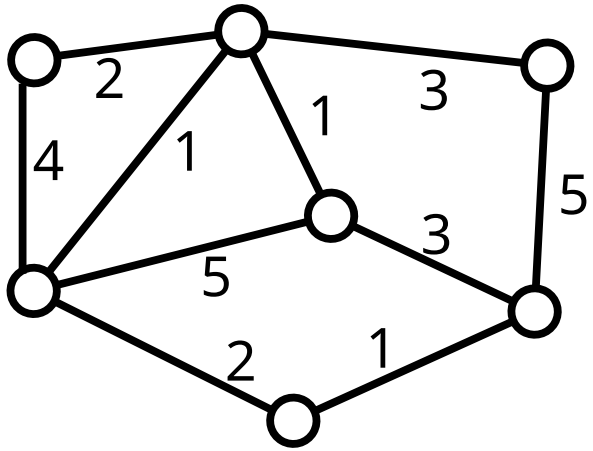


Exemple : arbre couvrant de poids minimum

(vu au cours de Graphes en BUT1)

Problème : Construire un arbre couvrant T d'un graphe G (T touche tous les sommets)

Objectif : minimiser le poids total des arêtes de l'arbre

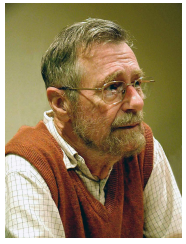


Algorithme de Jarník-Prim :

- Choisir un sommet v_0 de départ ; $T = (\{v_0\}, \{\})$
- Tant que tous les sommets de G ne sont pas dans l'arbre T
  - Choisir un sommet v dans $V(G) \setminus V(T)$ qui a le plus petit coût de rattachement à T via l'arête e
  - $T = T \cup \{v, e\}$
- Renvoyer T

Vojtěch Jarník
(1897-1970)

Robert C. Prim
(1921-)



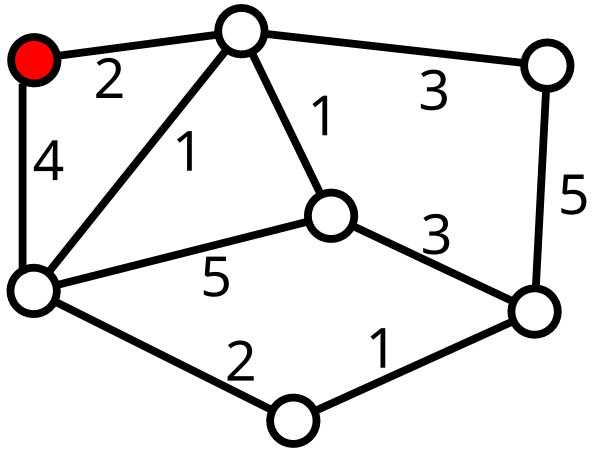
Edsger W. Dijkstra
(1930-2002)

Exemple : arbre couvrant de poids minimum

(vu au cours de Graphes en BUT1)

Problème : Construire un arbre couvrant T d'un graphe G (T touche tous les sommets)

Objectif : minimiser le poids total des arêtes de l'arbre

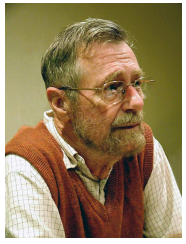


Algorithme de Jarník-Prim :

- Choisir un sommet v_0 de départ ; $T = (\{v_0\}, \{\})$
- Tant que tous les sommets de G ne sont pas dans l'arbre T
  - Choisir un sommet v dans $V(G) \setminus V(T)$ qui a le plus petit coût de rattachement à T via l'arête e
  - $T = T \cup \{v, e\}$
- Renvoyer T

Vojtěch Jarník
(1897-1970)

Robert C. Prim
(1921-)



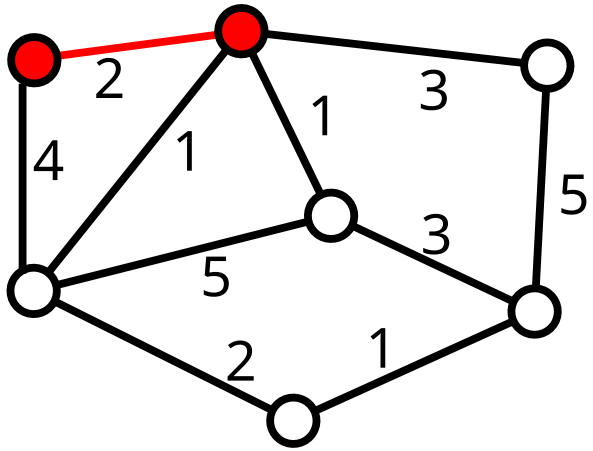
Edsger W. Dijkstra
(1930-2002)

Exemple : arbre couvrant de poids minimum

(vu au cours de Graphes en BUT1)

Problème : Construire un arbre couvrant T d'un graphe G (T touche tous les sommets)

Objectif : minimiser le poids total des arêtes de l'arbre

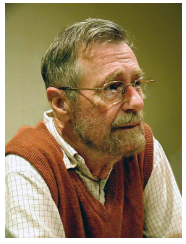


Algorithme de Jarník-Prim :

- Choisir un sommet v_0 de départ ; $T = (\{v_0\}, \{\})$
- Tant que tous les sommets de G ne sont pas dans l'arbre T
  - Choisir un sommet v dans $V(G) \setminus V(T)$ qui a le plus petit coût de rattachement à T via l'arête e
  - $T = T \cup \{v, e\}$
- Renvoyer T

Vojtěch Jarník
(1897-1970)

Robert C. Prim
(1921-)



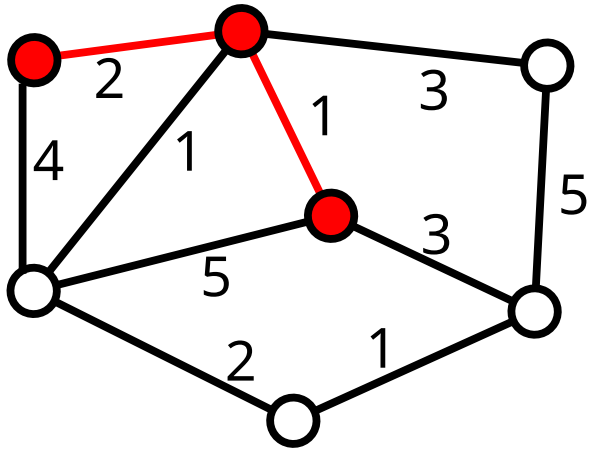
Edsger W. Dijkstra
(1930-2002)

Exemple : arbre couvrant de poids minimum

(vu au cours de Graphes en BUT1)

Problème : Construire un arbre couvrant T d'un graphe G (T touche tous les sommets)

Objectif : minimiser le poids total des arêtes de l'arbre

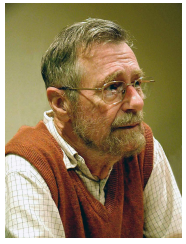


Algorithme de Jarník-Prim :

- Choisir un sommet v_0 de départ ; $T = (\{v_0\}, \{\})$
- Tant que tous les sommets de G ne sont pas dans l'arbre T
  - Choisir un sommet v dans $V(G) \setminus V(T)$ qui a le plus petit coût de rattachement à T via l'arête e
  - $T = T \cup \{v, e\}$
- Renvoyer T

Vojtěch Jarník
(1897-1970)

Robert C. Prim
(1921-)



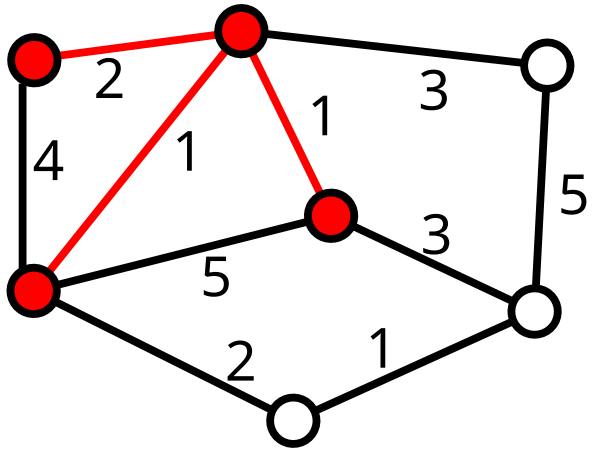
Edsger W. Dijkstra
(1930-2002)

Exemple : arbre couvrant de poids minimum

(vu au cours de Graphes en BUT1)

Problème : Construire un arbre couvrant T d'un graphe G (T touche tous les sommets)

Objectif : minimiser le poids total des arêtes de l'arbre

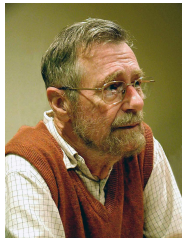


Algorithme de Jarník-Prim :

- Choisir un sommet v_0 de départ ; $T = (\{v_0\}, \{\})$
- Tant que tous les sommets de G ne sont pas dans l'arbre T
  - Choisir un sommet v dans $V(G) \setminus V(T)$ qui a le plus petit coût de rattachement à T via l'arête e
  - $T = T \cup \{v, e\}$
- Renvoyer T

Vojtěch Jarník
(1897-1970)

Robert C. Prim
(1921-)



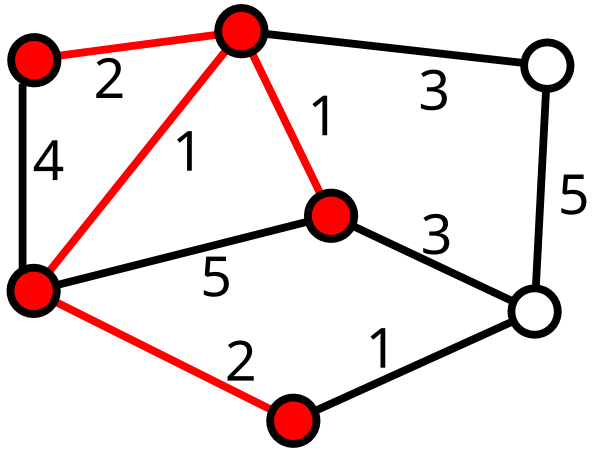
Edsger W. Dijkstra
(1930-2002)

Exemple : arbre couvrant de poids minimum

(vu au cours de Graphes en BUT1)

Problème : Construire un arbre couvrant T d'un graphe G (T touche tous les sommets)

Objectif : minimiser le poids total des arêtes de l'arbre

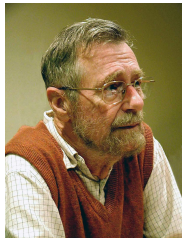


Algorithme de Jarník-Prim :

- Choisir un sommet v_0 de départ ; $T = (\{v_0\}, \{\})$
- Tant que tous les sommets de G ne sont pas dans l'arbre T
  - Choisir un sommet v dans $V(G) \setminus V(T)$ qui a le plus petit coût de rattachement à T via l'arête e
  - $T = T \cup \{v, e\}$
- Renvoyer T

Vojtěch Jarník
(1897-1970)

Robert C. Prim
(1921-)



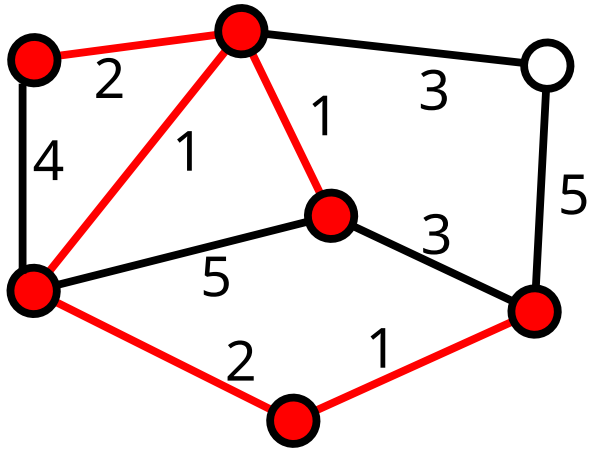
Edsger W. Dijkstra
(1930-2002)

Exemple : arbre couvrant de poids minimum

(vu au cours de Graphes en BUT1)

Problème : Construire un arbre couvrant T d'un graphe G (T touche tous les sommets)

Objectif : minimiser le poids total des arêtes de l'arbre



Algorithme de Jarník-Prim :

- Choisir un sommet v_0 de départ ; $T = (\{v_0\}, \{\})$
- Tant que tous les sommets de G ne sont pas dans l'arbre T
  - Choisir un sommet v dans $V(G) \setminus V(T)$ qui a le plus petit coût de rattachement à T via l'arête e
  - $T = T \cup \{v, e\}$
- Renvoyer T

Vojtěch Jarník
(1897-1970)

Robert C. Prim
(1921-)

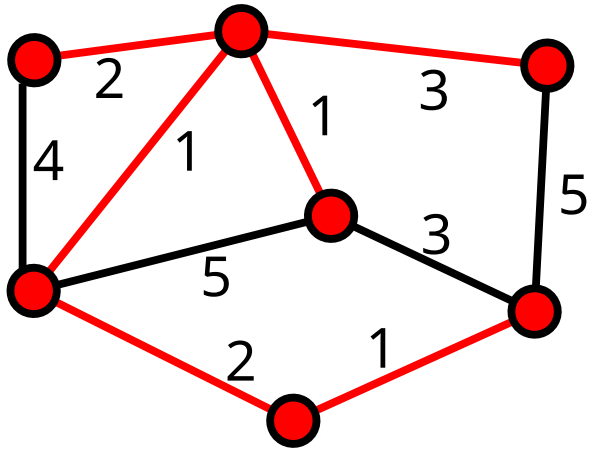
Edsger W. Dijkstra
(1930-2002)

Exemple : arbre couvrant de poids minimum

(vu au cours de Graphes en BUT1)

Problème : Construire un arbre couvrant T d'un graphe G (T touche tous les sommets)

Objectif : minimiser le poids total des arêtes de l'arbre

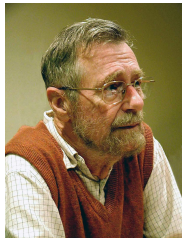


Algorithme de Jarník-Prim :

- Choisir un sommet v_0 de départ ; $T = (\{v_0\}, \{\})$
- Tant que tous les sommets de G ne sont pas dans l'arbre T
  - Choisir un sommet v dans $V(G) \setminus V(T)$ qui a le plus petit coût de rattachement à T via l'arête e
  - $T = T \cup \{v, e\}$
- Renvoyer T

Vojtěch Jarník
(1897-1970)

Robert C. Prim
(1921-)

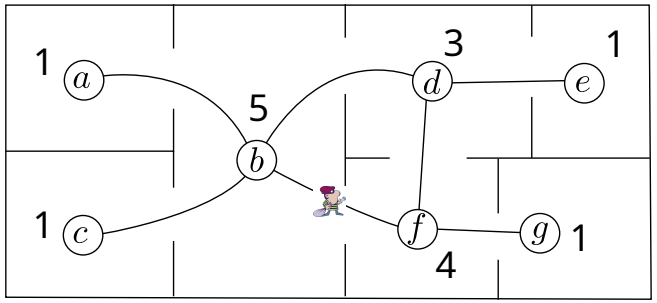


Edsger W. Dijkstra
(1930-2002)

Surveillance d’un bâtiment : couverture par sommets

Problème : couvrir les couloirs d’un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

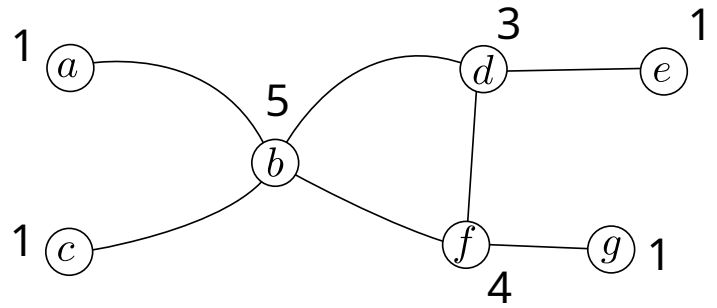
Objectif : minimiser le coût total de la pose des détecteurs



Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs

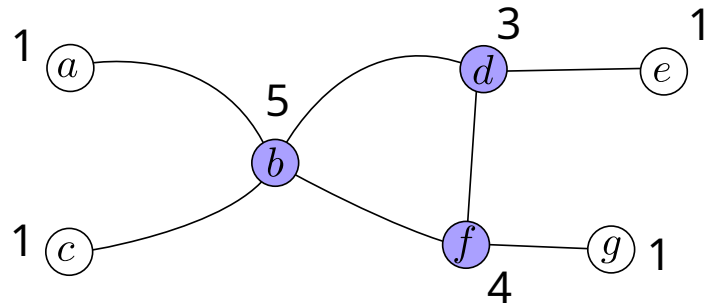


Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs

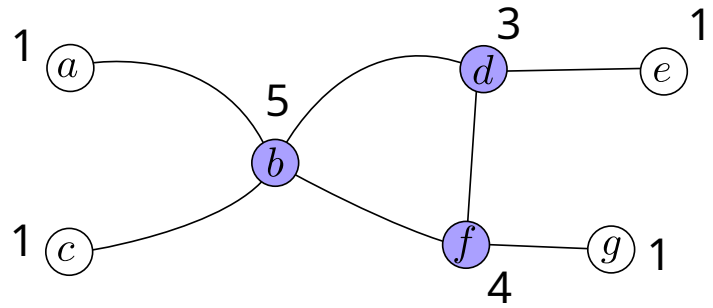


Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

On écrit le PL en nombres entiers (PLNE) suivant :

Une variable x_v pour chaque sommet v : $x_v = 1$ si on a un détecteur sur v, 0 sinon. Chaque sommet v a une valeur $c(v)$ (le coût de la pose d'un détecteur sur v).

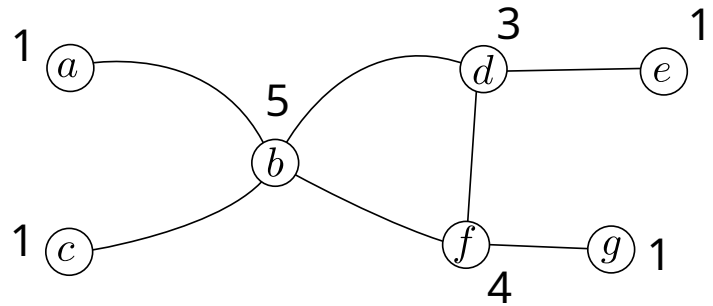
$$\begin{array}{lllll} \text{minimiser :} & \sum_{v \in V} x_v c(v) & & & \\ \text{tel que :} & x_u + x_v & \geq & 1 & \forall uv \in E \\ & x_v & \leq & 1 & \forall v \in V \\ & x_v & \geq & 0 & \forall v \in V \\ & x_v & \in & \mathbb{N} & \forall v \in V \end{array}$$

Rappel : On obtient une solution approchée (au pire 2 fois moins bonne que l'optimum) en passant par la relaxation linéaire de ce PL.

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

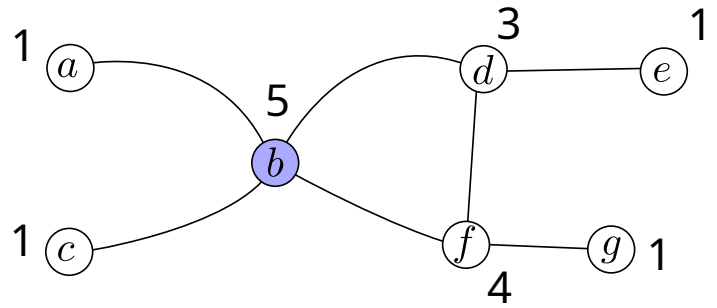
Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

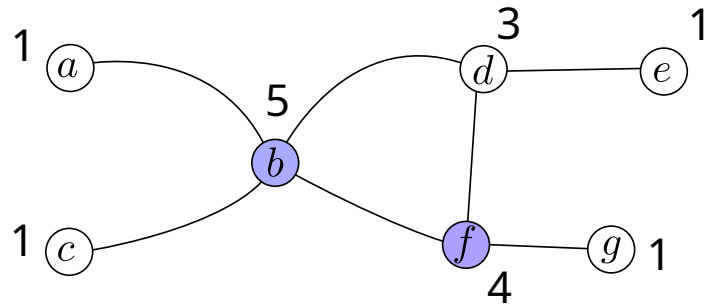
Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

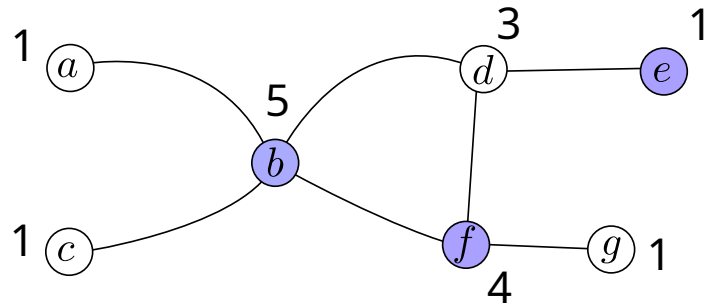
Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

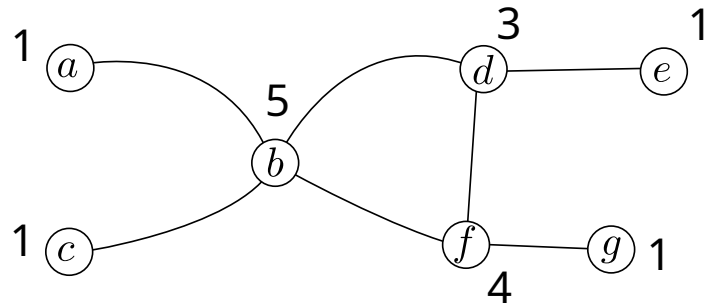
Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

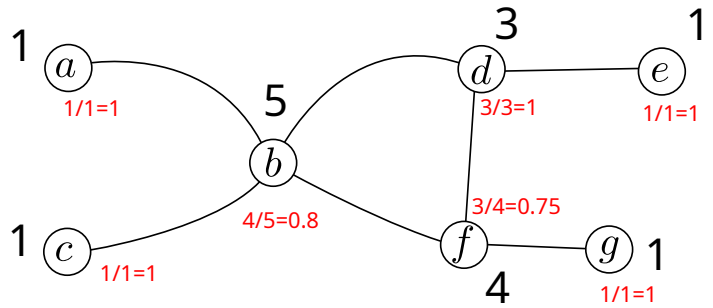
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

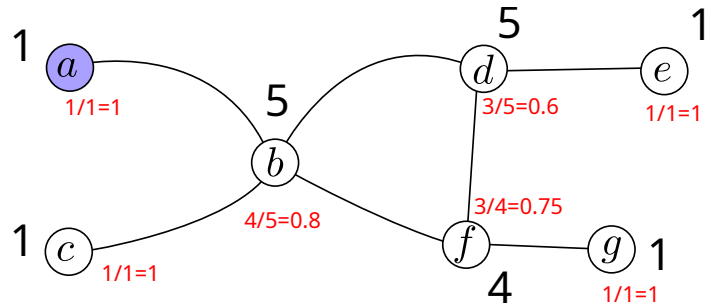
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

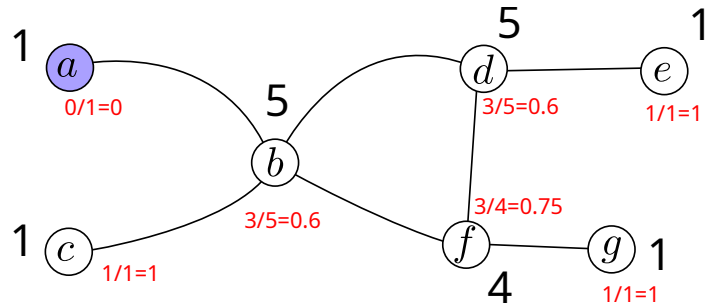
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

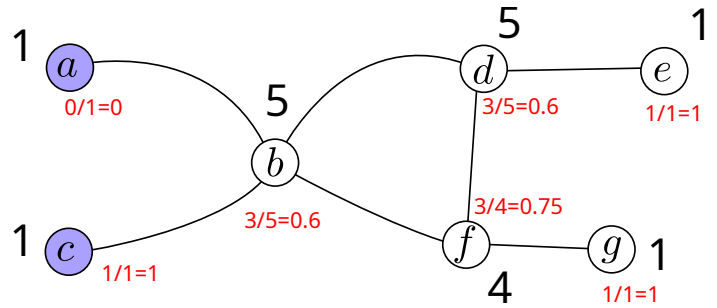
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

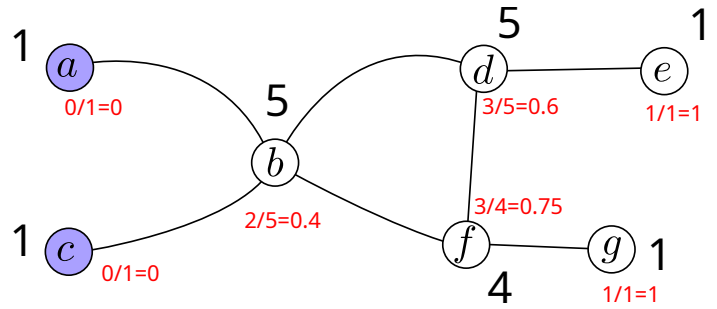
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

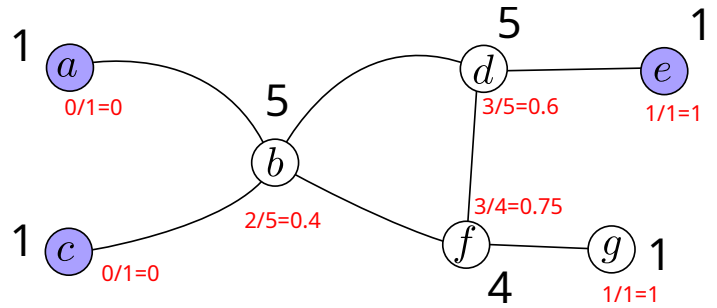
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

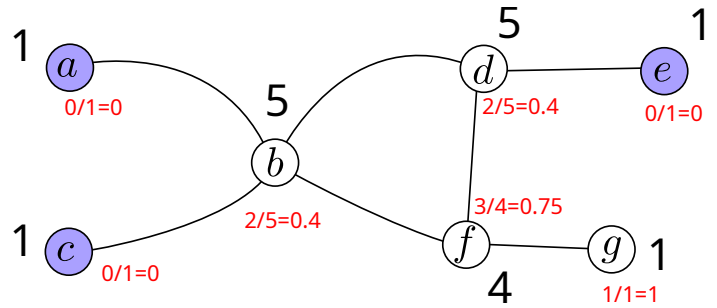
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

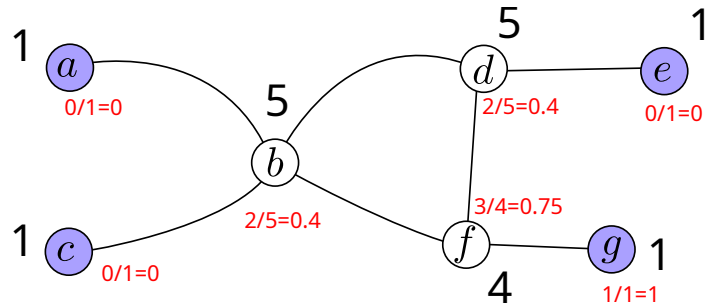
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

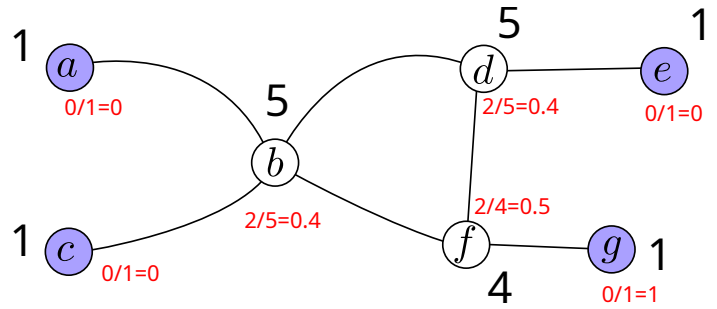
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

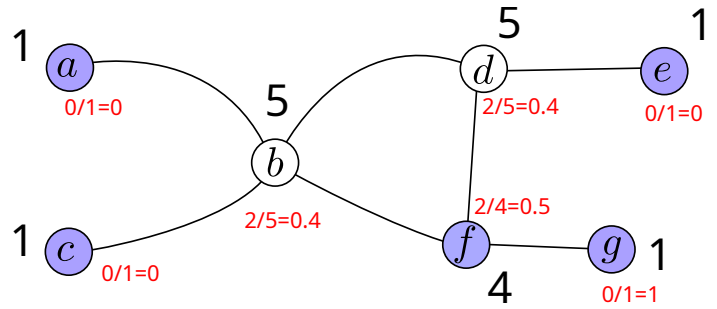
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

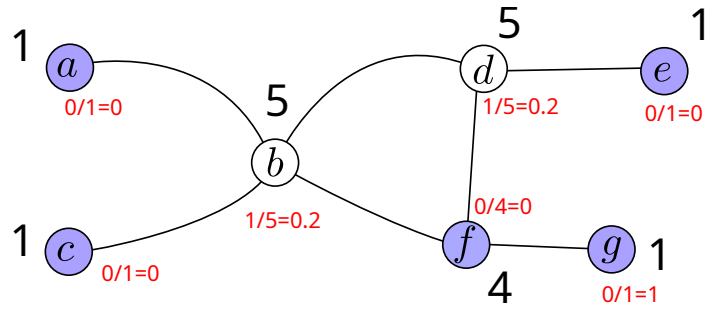
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

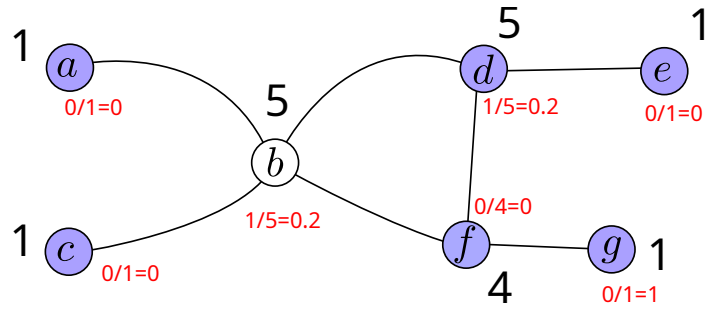
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

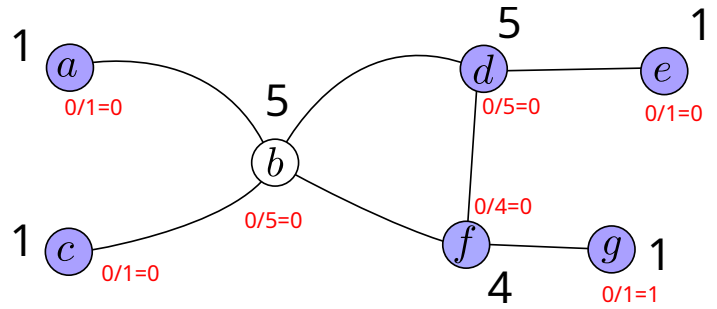
Algorithme glouton n°2 :

- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Surveillance d'un bâtiment : couverture par sommets

Problème : couvrir les couloirs d'un bâtiment avec des détecteurs de mouvement. Pour un sommet v, poser un détecteur coûte $c(v)$.

Objectif : minimiser le coût total de la pose des détecteurs



Le bâtiment est un graphe non-orienté $G = (V, E)$ et pondéré.

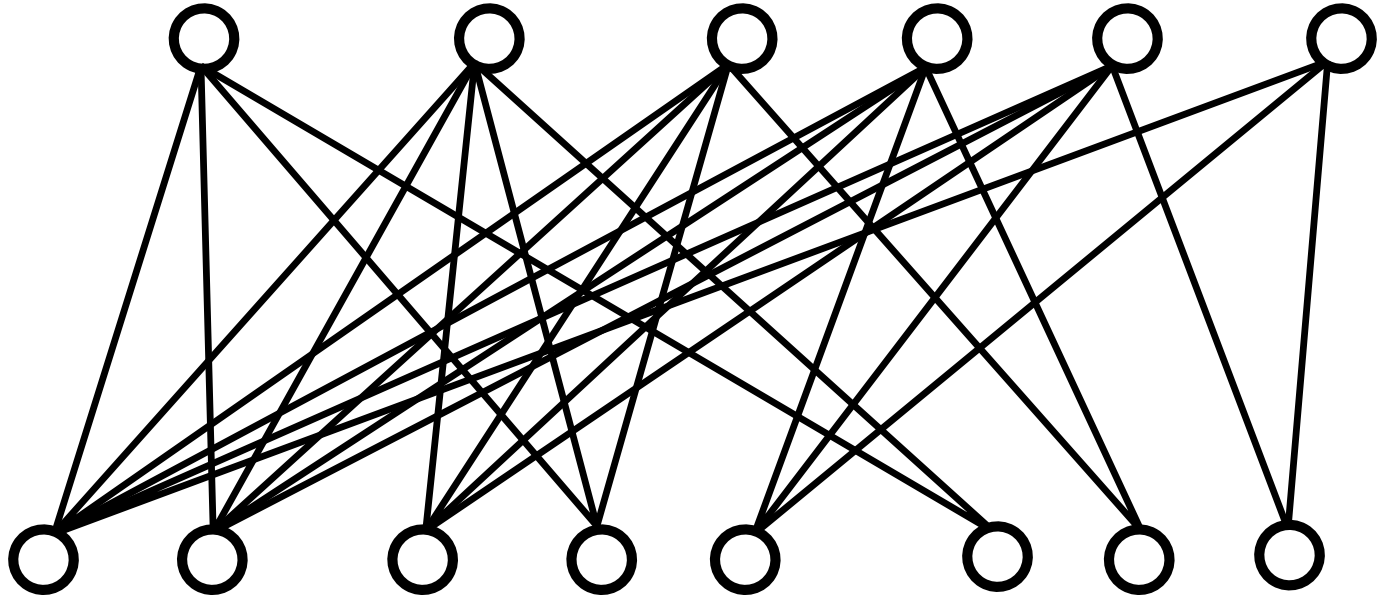
Algorithme glouton n°1 :

- $S = \emptyset$ (*ensemble vide*)
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir un sommet v qui touche le plus grand nombre d'arêtes non couvertes par S
  - $S = S \cup \{v\}$
- Renvoyer S

Algorithme glouton n°2 :

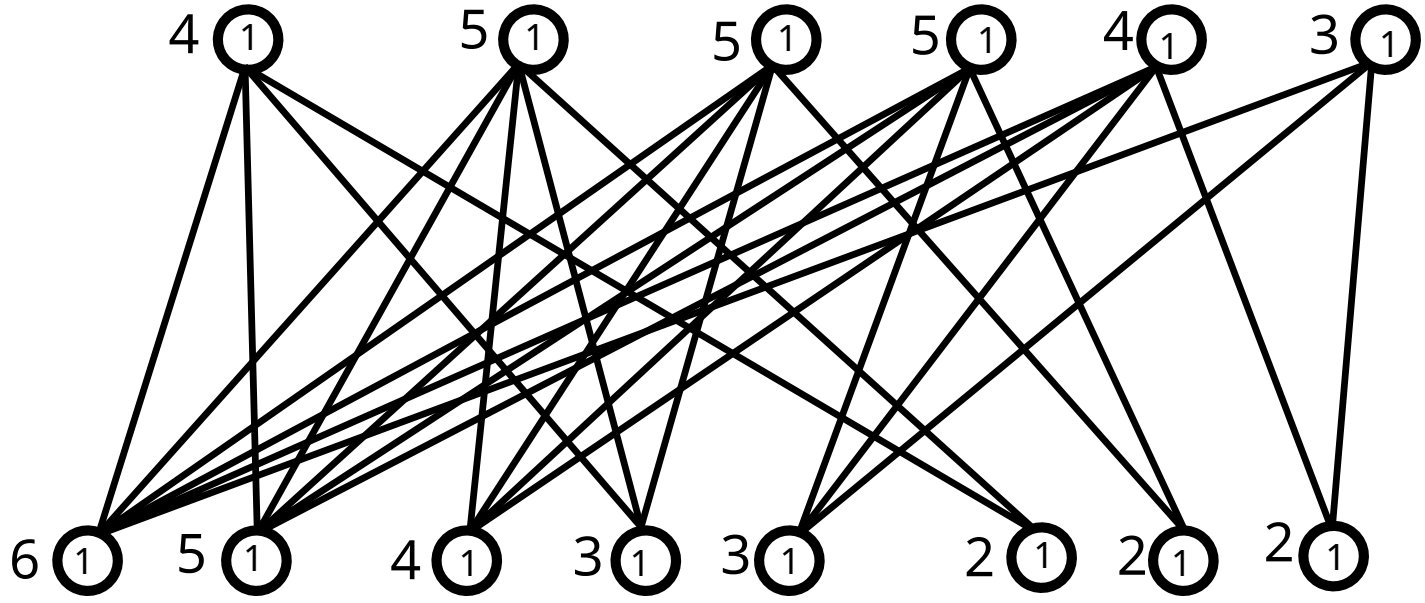
- $S = \emptyset$
- Tant que toutes les arêtes ne sont pas couvertes :
  - Choisir v maximisant le ratio "nombre d'arêtes nouvellement couvertes" / $c(v)$
  - $S = S \cup \{v\}$
- Renvoyer S

Aucun des deux algos gloutons n’est bon



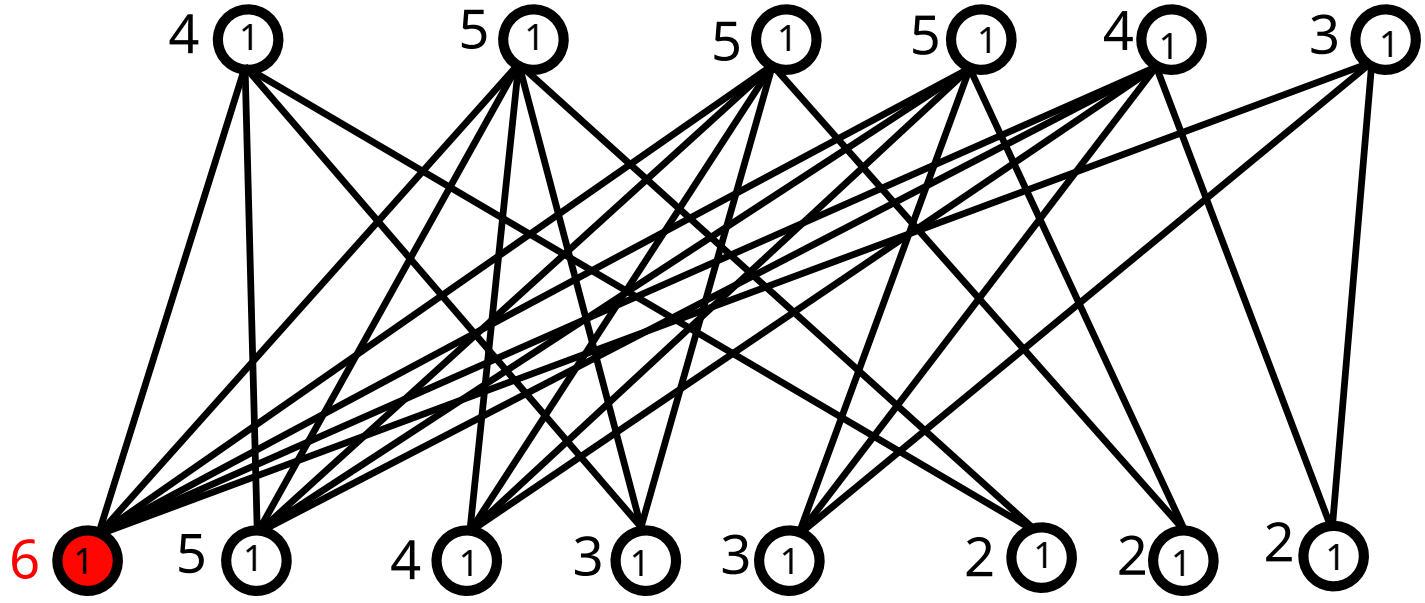
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



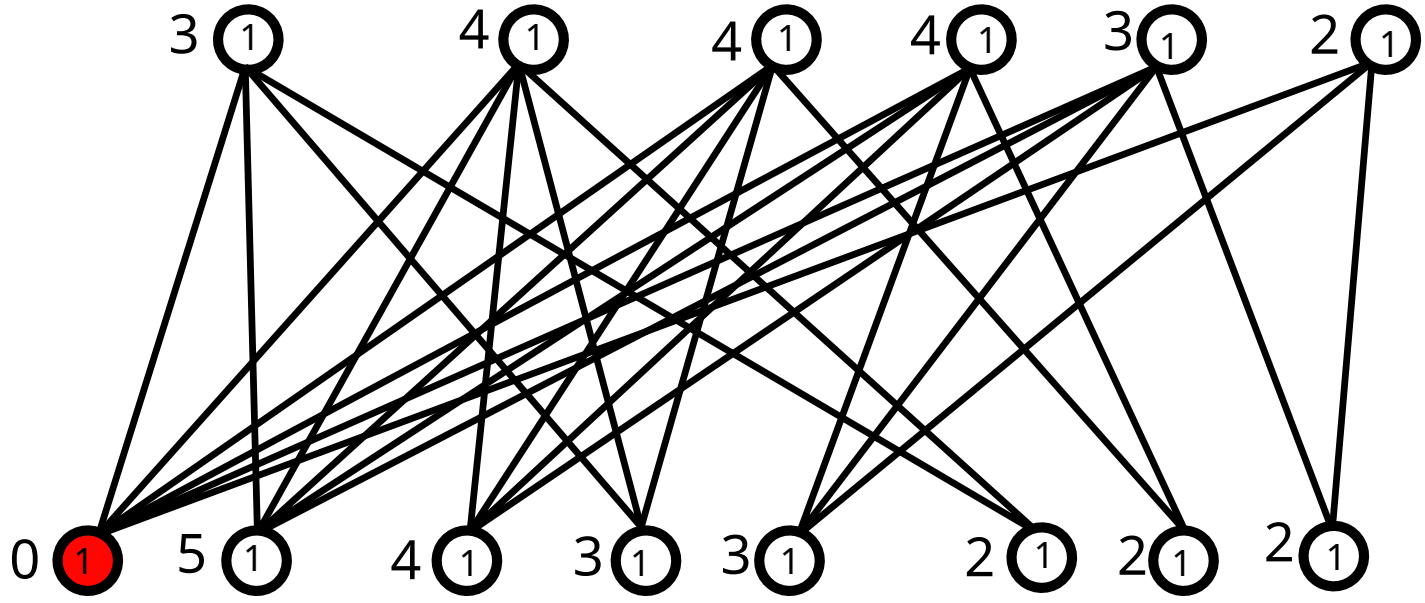
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



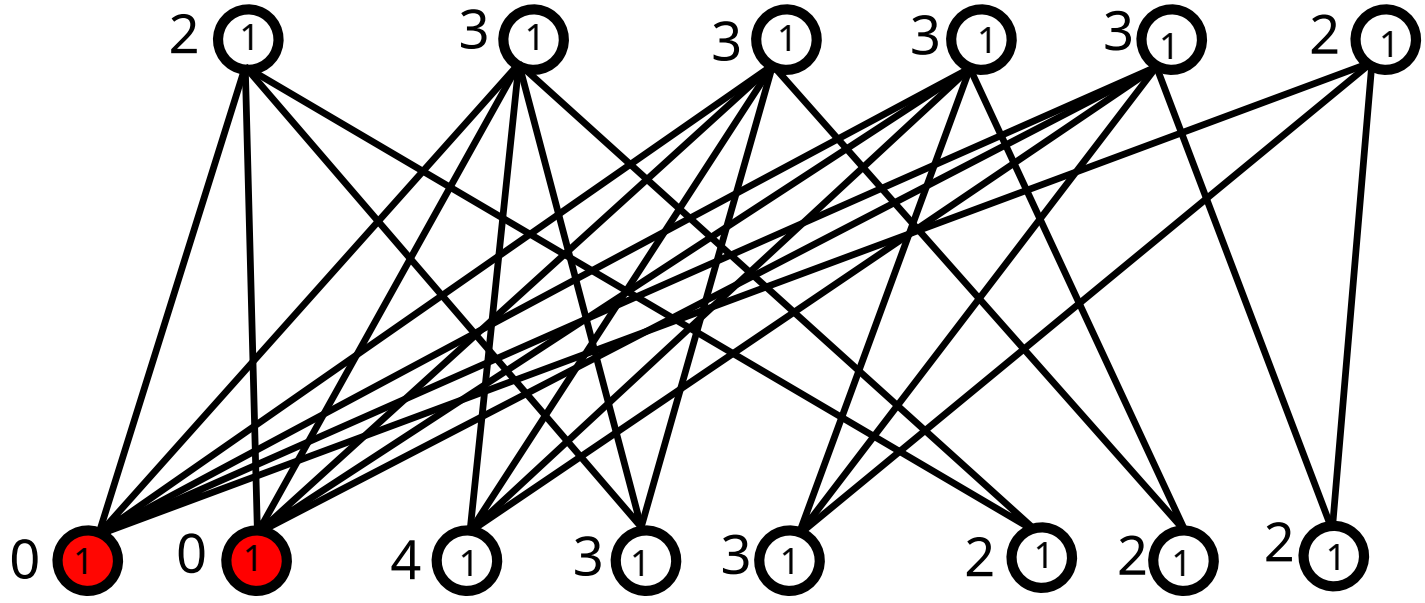
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



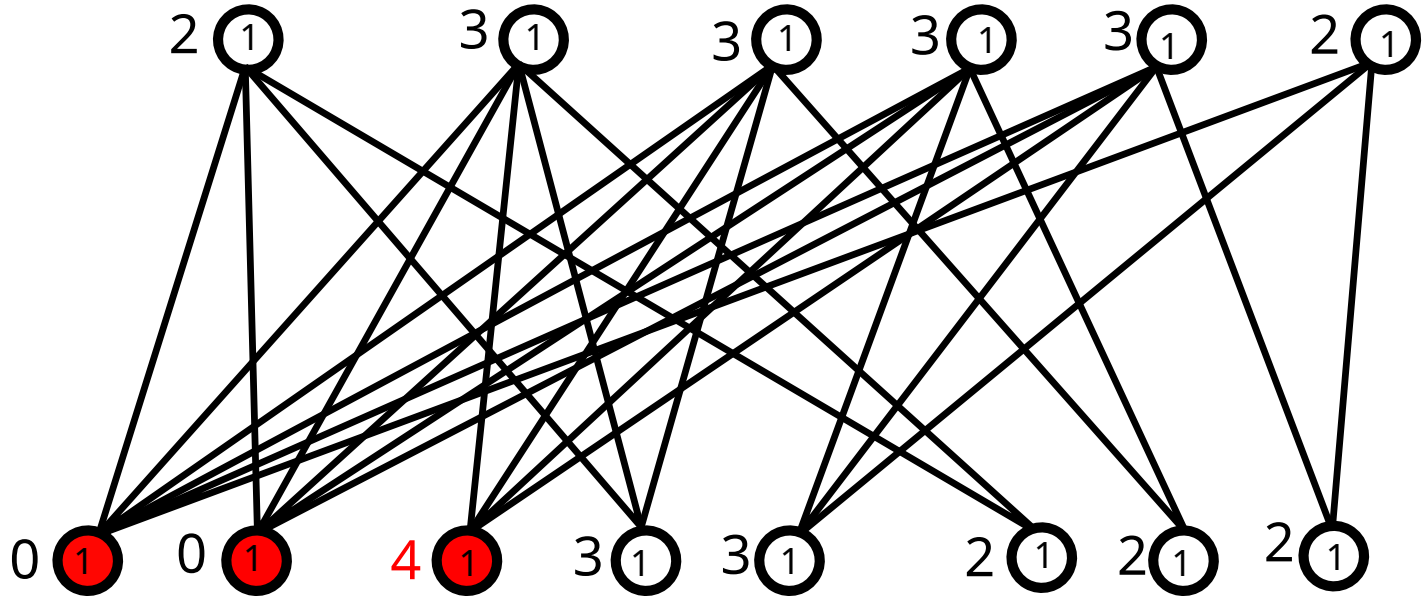
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



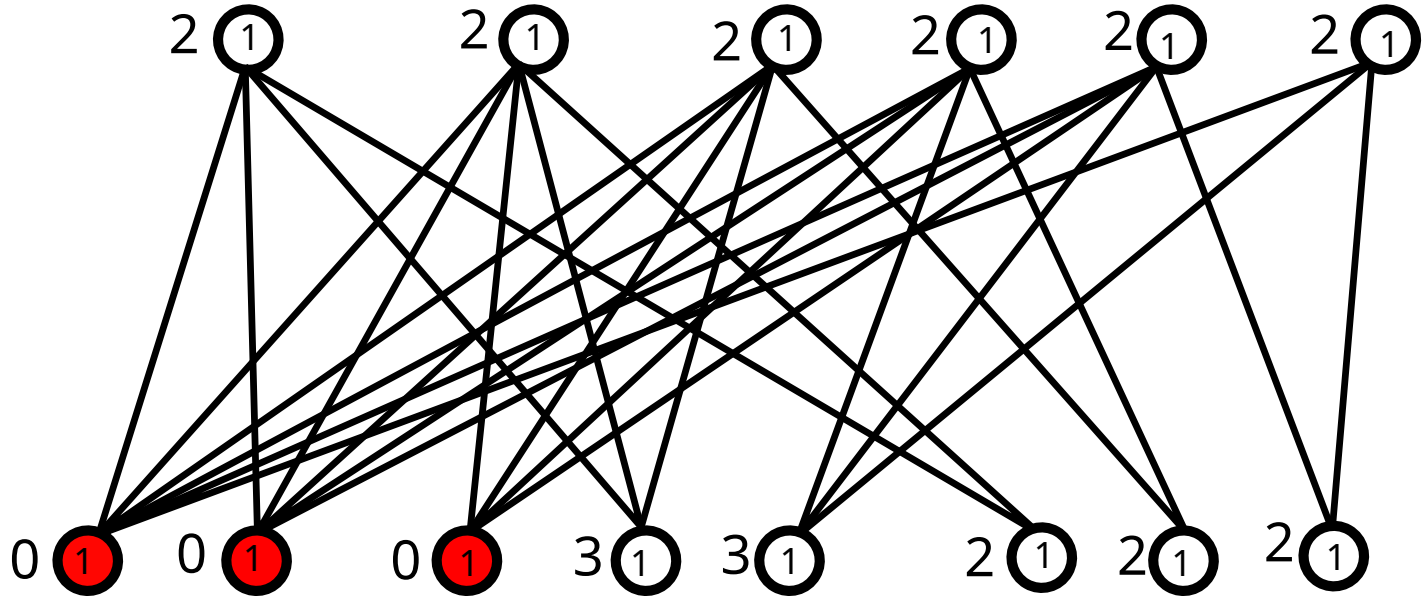
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



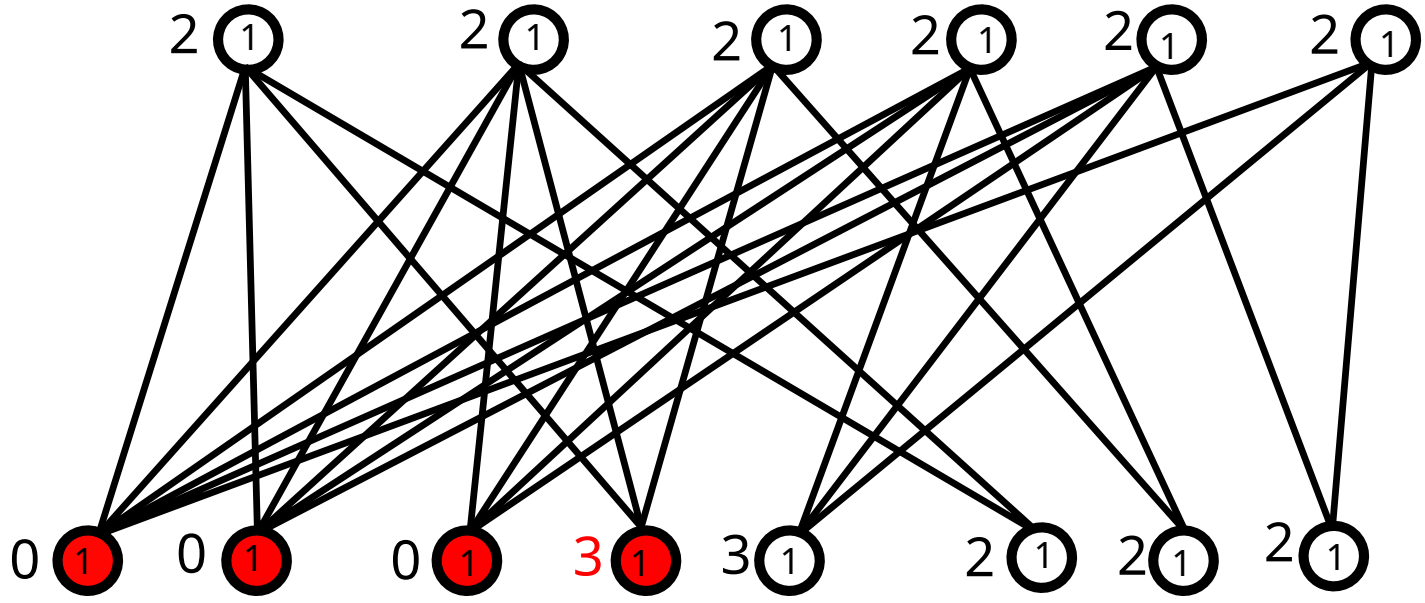
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



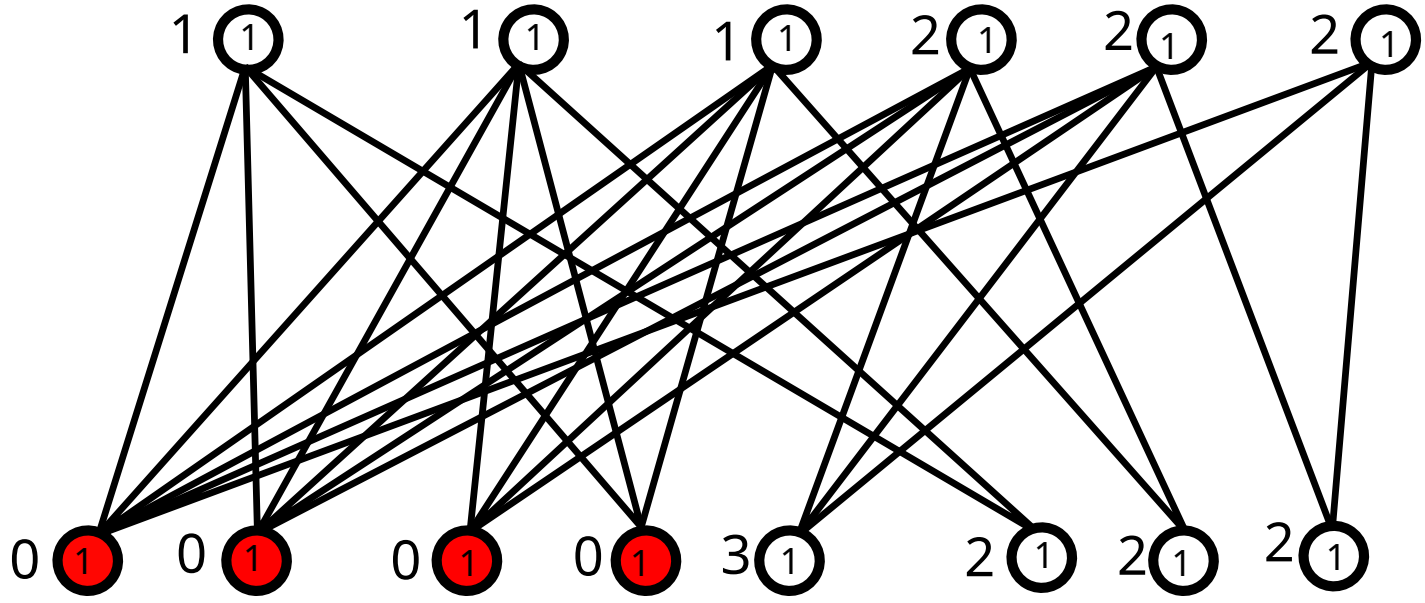
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



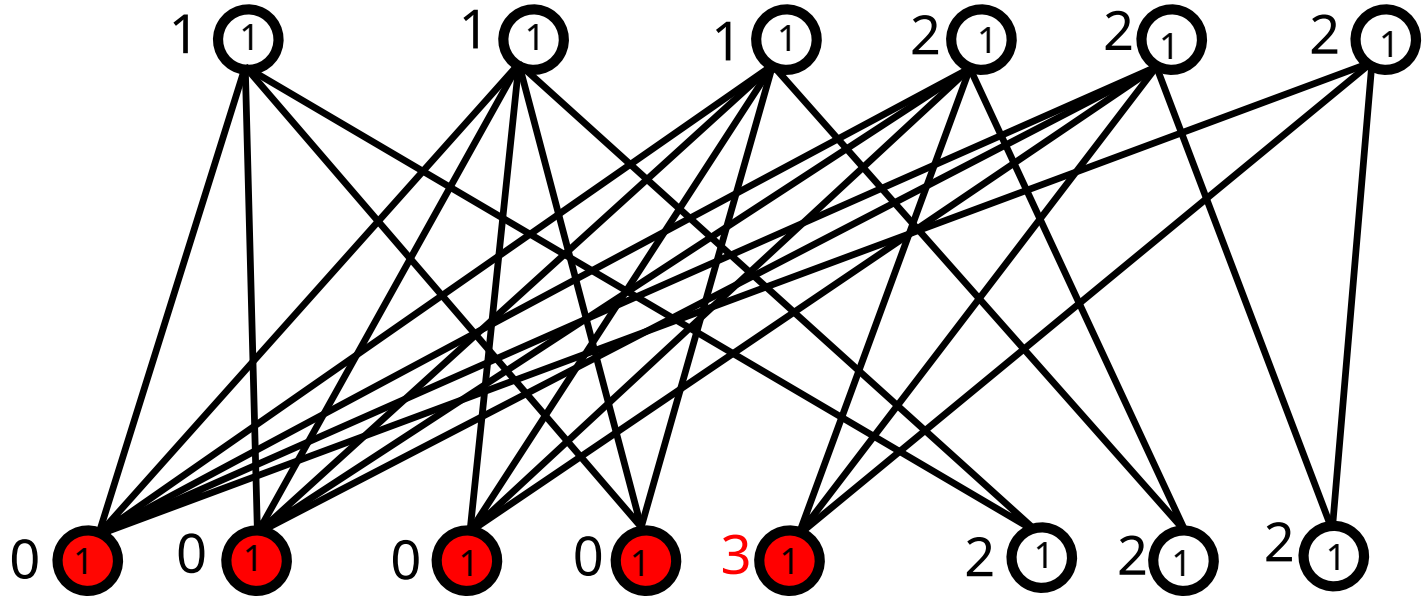
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



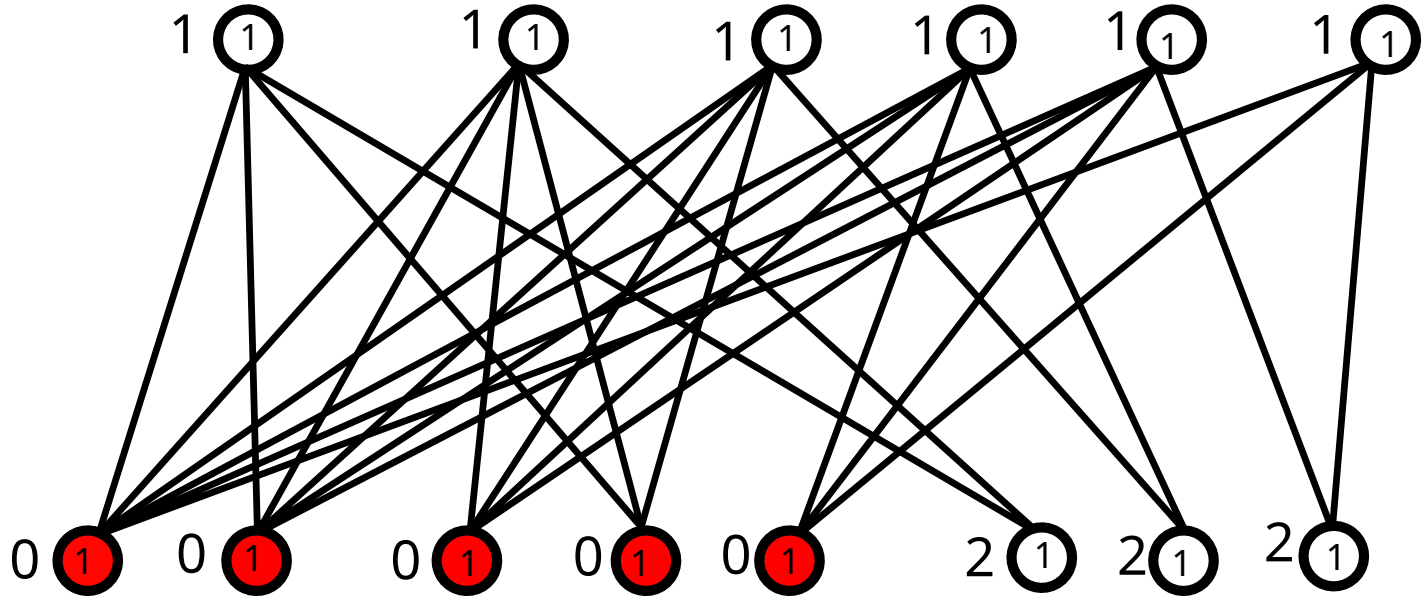
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



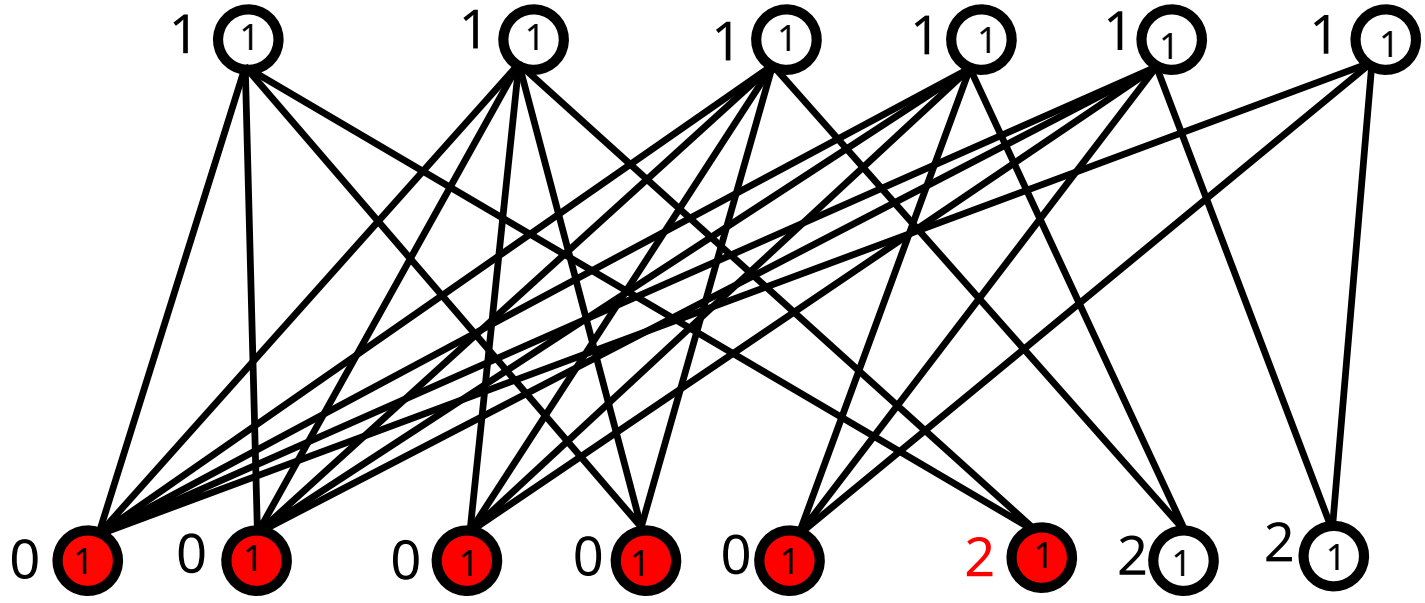
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



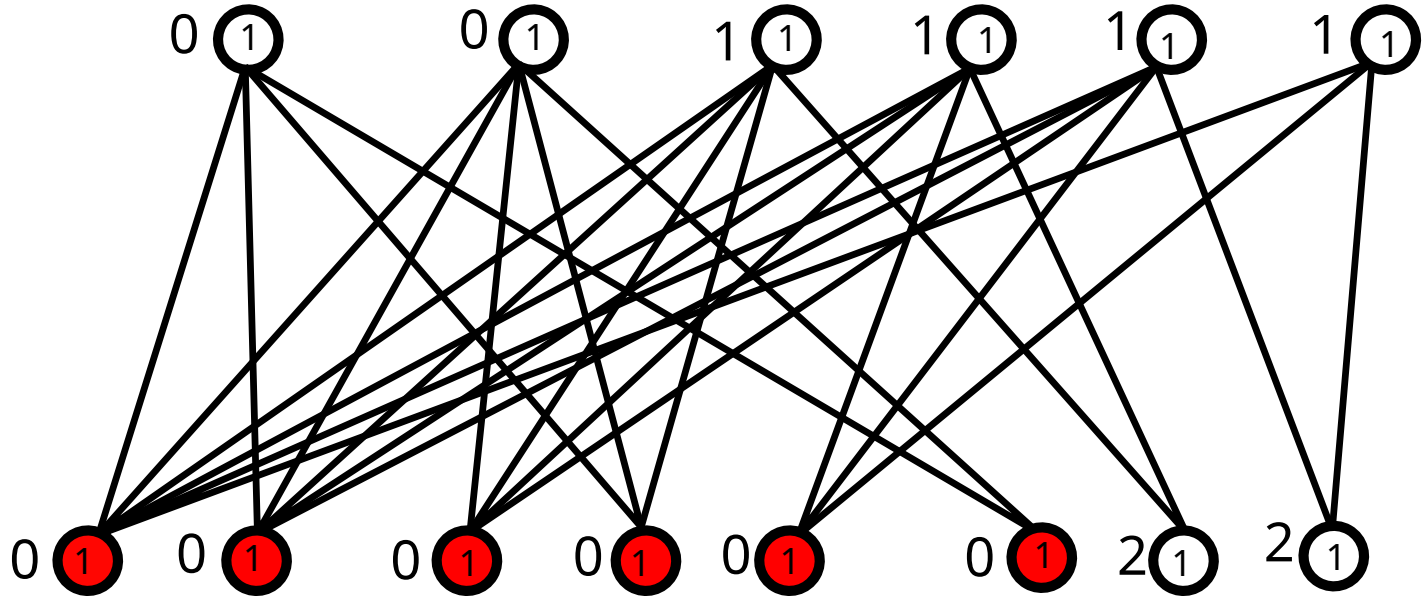
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



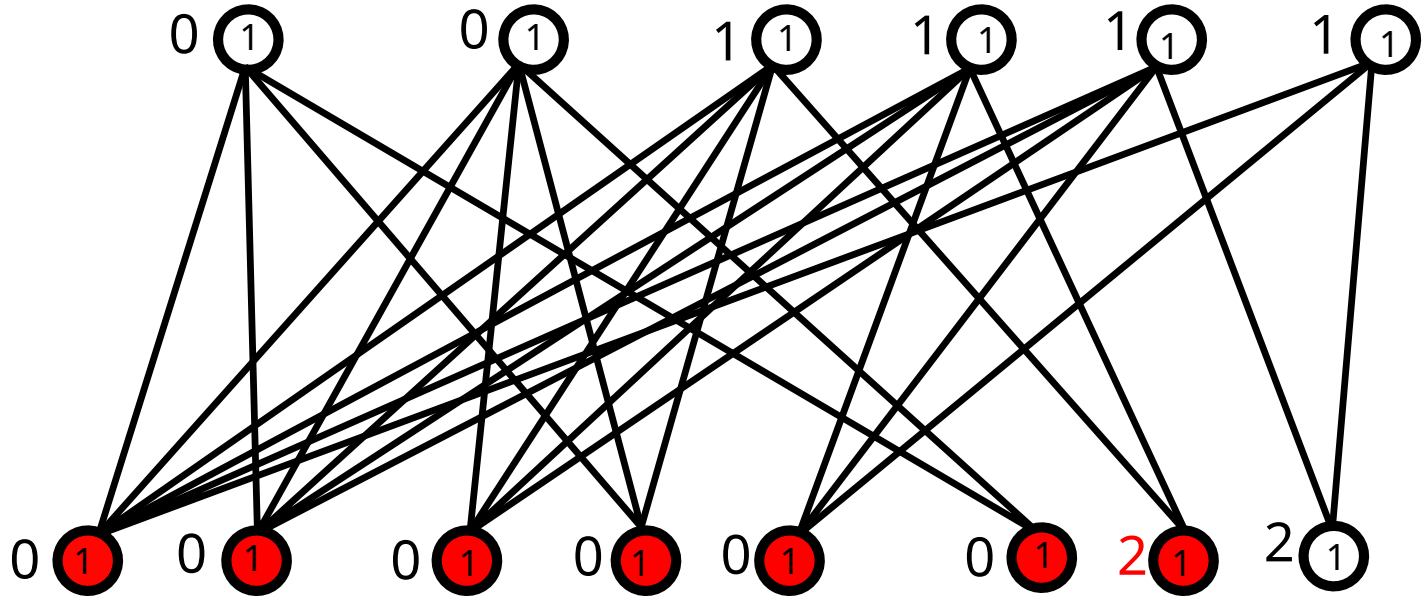
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



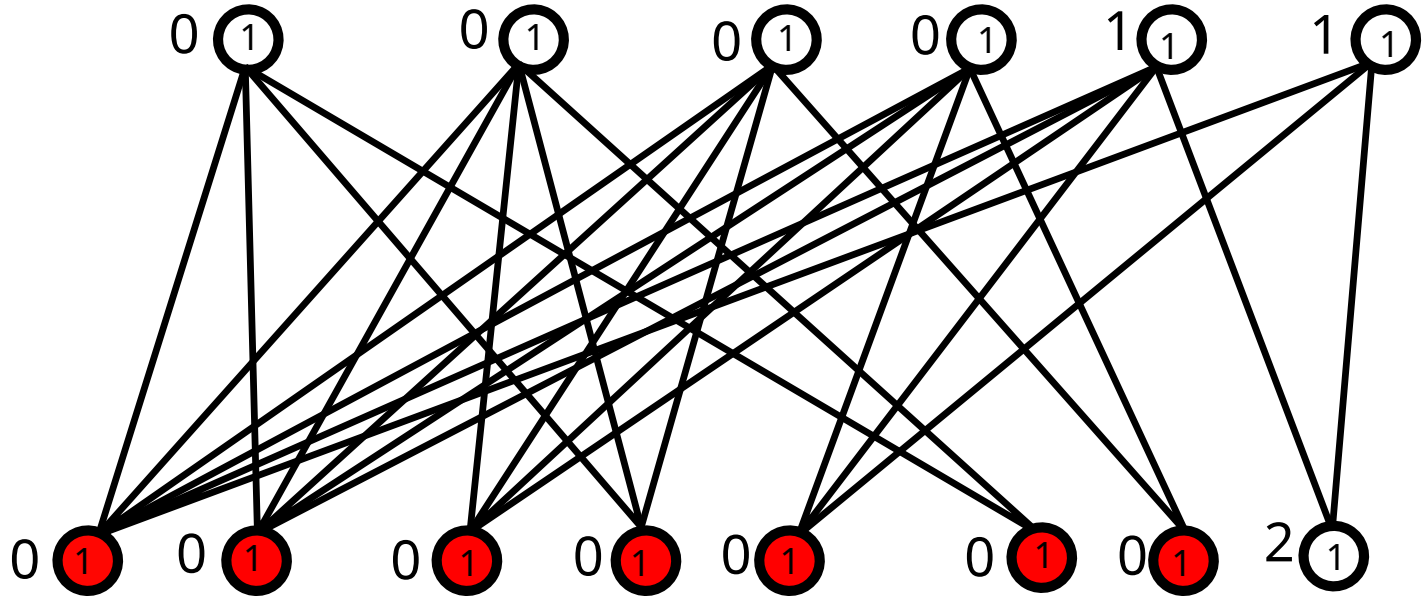
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



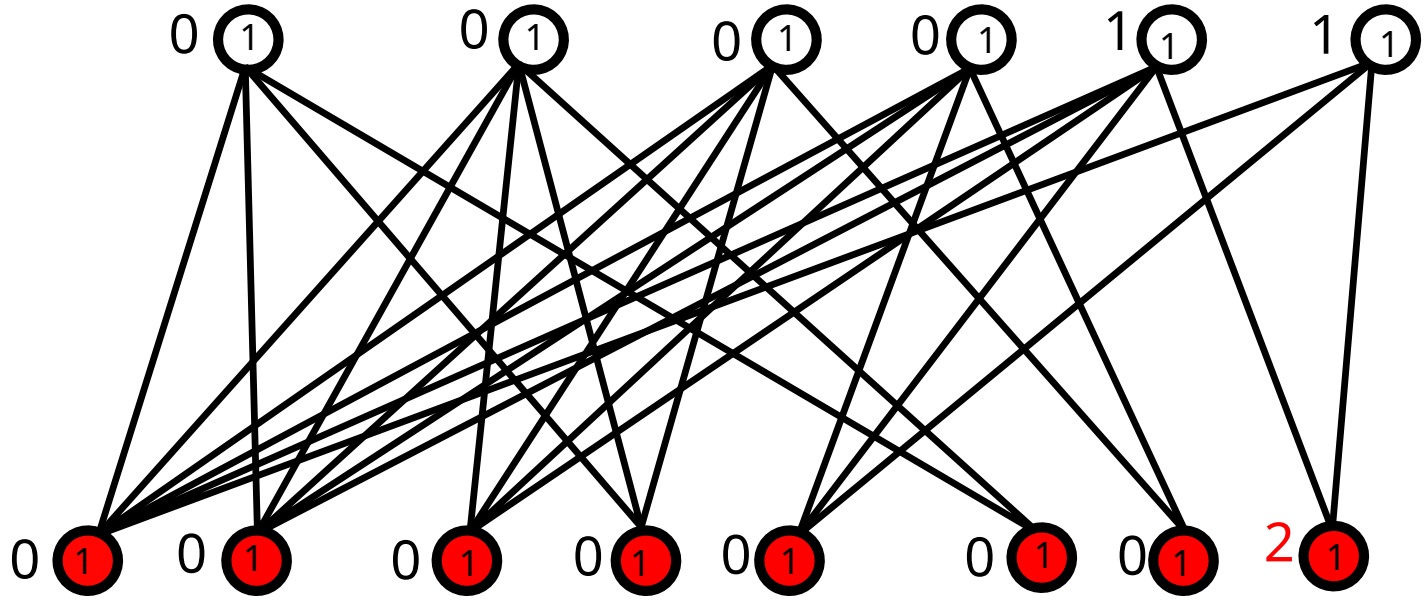
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



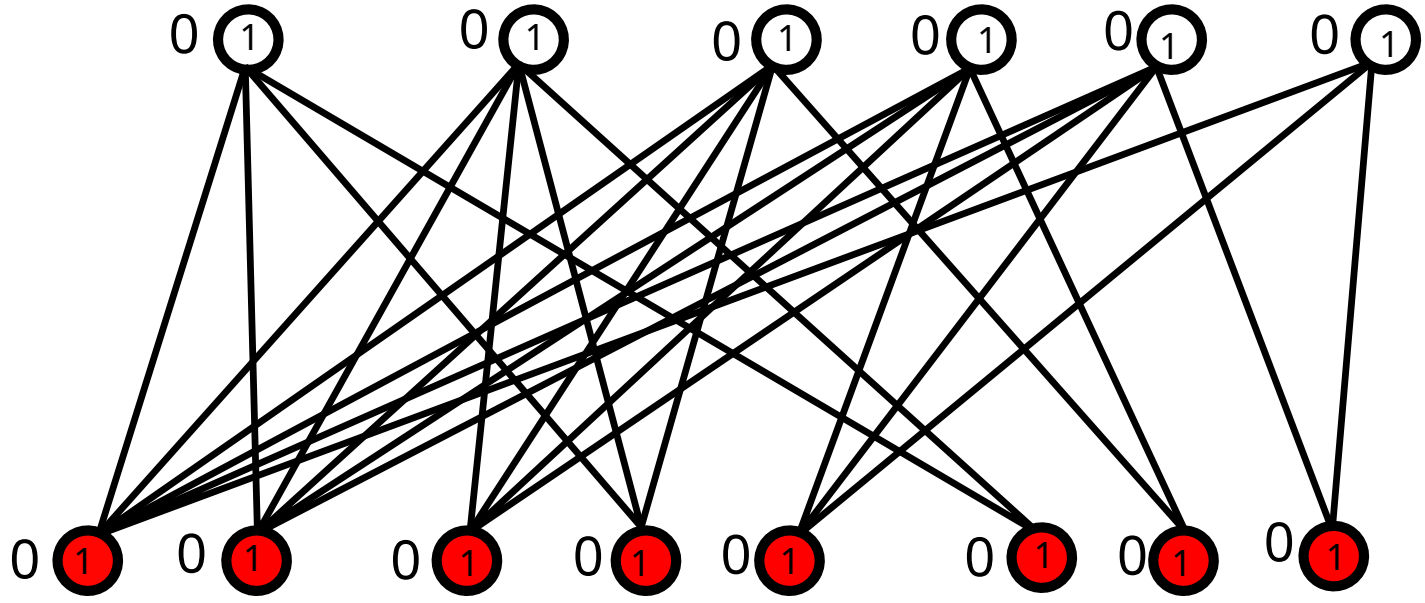
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



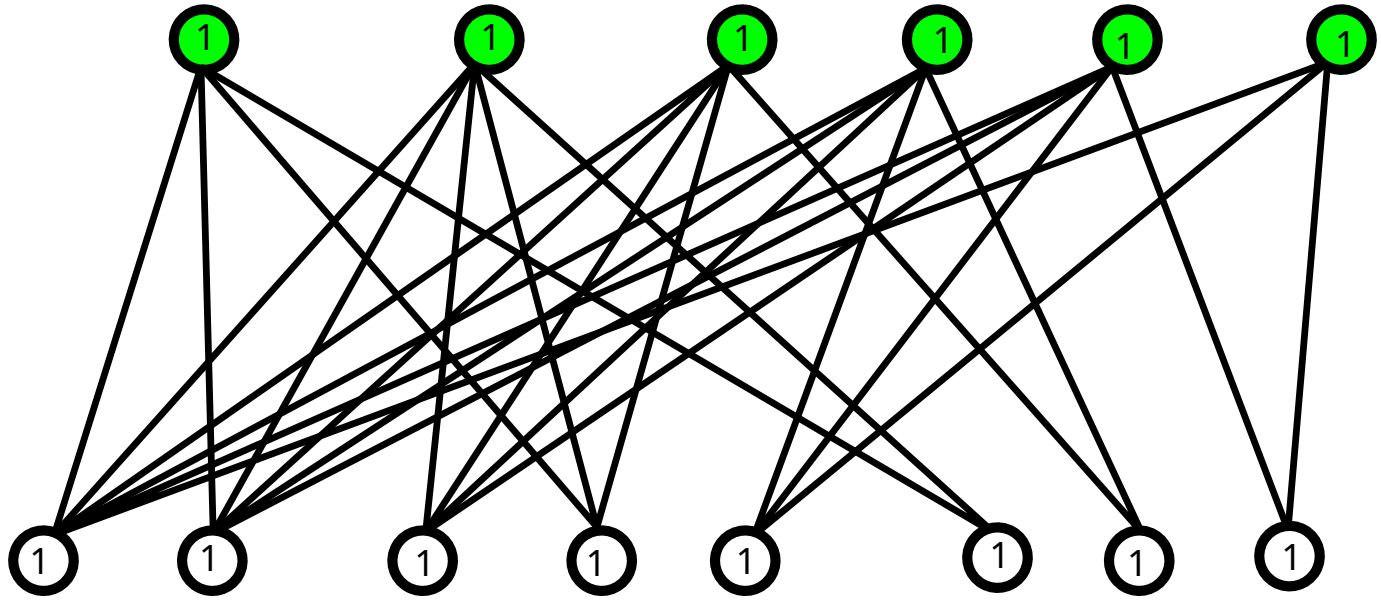
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



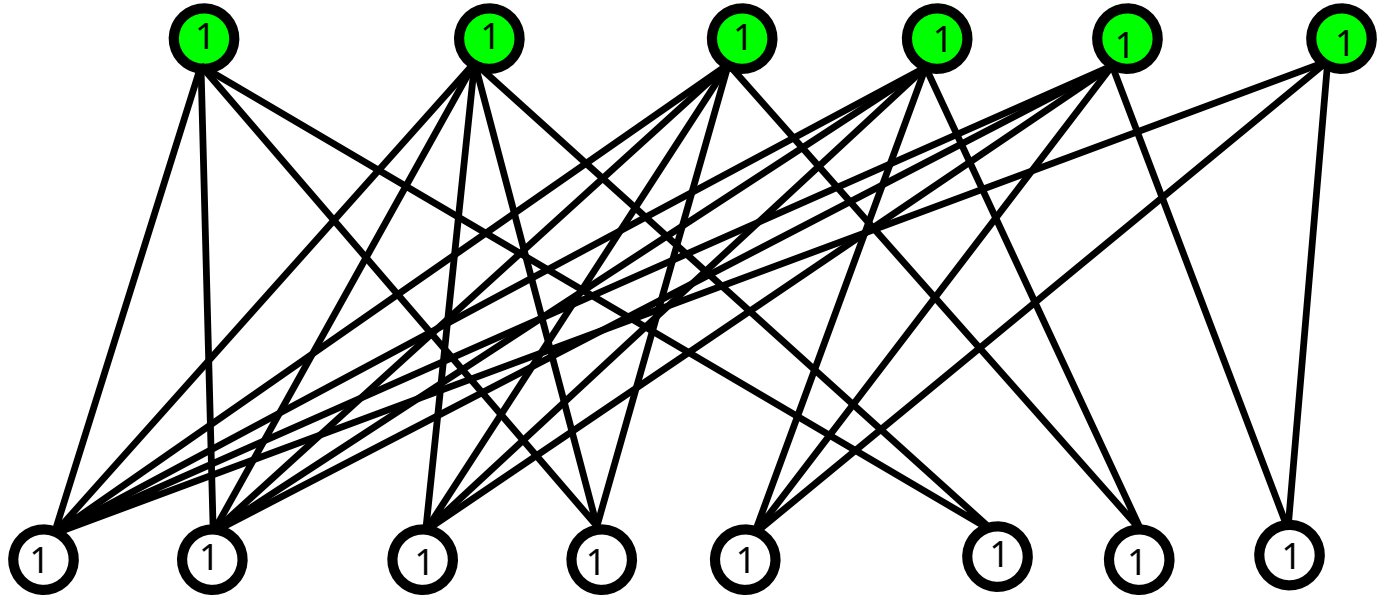
(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



(tous les sommets ont le même poids : 1)

Aucun des deux algos gloutons n’est bon



(tous les sommets ont le même poids : 1)

Les algorithmes gloutons du transparent précédent n’ont

aucune garantie de performances...

Conclusion

- Notions d’algorithme heuristique et d’algorithme d’approximation
- Parfois, la relaxation linéaire donne toujours des solutions entières
- Technique de l’arrondi de la relaxation linéaire : peut donner un algorithme d’approximation
- Algorithmes gloutons : efficaces en temps, calculent parfois une solution optimale, mais pas toujours → algorithmes heuristiques