Stratégies de Recherches Intelligentes

ou comment résoudre efficacement les problèmes difficiles.

Dr Jean-Philippe Préaux,

CReA / Université de Provence.

Comment résoudre un problème NP-difficile ?

- On a vu en optimisation combinatoire plusieurs classes de problèmes NP-difficiles, dont :
  - Coloration de graphe,
  - Problème du sac à dos,
  - Problème du voyageur de commerce,
  - Problème de satisfaisabilité des clauses.

Comment résoudre un problème NP-difficile ?

- On a vu en optimisation combinatoire plusieurs classes de problèmes NP-difficiles, dont :
  - Coloration de graphe,
  - Problème du sac à dos,
  - Problème du voyageur de commerce,
  - Problème de satisfaisabilité des clauses.
- On a vu pourquoi (P=NP?), sauf à être éxagéremment optimiste (pessimiste?) on ne pouvait espérer en trouver une solution vraiment efficace, *i.e.* un algorithme de complexité polynomiale.

Comment résoudre un problème NP-difficile ?

- Cependant, la technologie moderne impose tous les jours de résoudre aussi efficacement que possible de tels problèmes avec des tailles de données gigantesques :
  - Réseaux modernes de communication,
  - Gestion du trafic aérien,
  - Génie logiciel, Intelligence artificielle,
  - Organisation logistique,
  - Optimisation des coûts, des bénéfices, des stocks, dans l'industrie, l'économie,
  - etc...
- Comment alors doit-on procéder ?... C'est le sujet de ce cours.

Comment résoudre un problème NP-difficile ?

On distingue deux types de méthodes :

Comment résoudre un problème NP-difficile ?

On distingue deux types de méthodes :

- Les méthodes exactes, qui retournent une solution optimale -de complexité exponentielle en théorie- mais qui, en moyenne, s'avèrent efficientes.

Comment résoudre un problème NP-difficile ?

On distingue deux types de méthodes :

- Les méthodes exactes, qui retournent une solution optimale -de complexité exponentielle en théorie- mais qui, en moyenne, s'avèrent efficientes.
- Les méthodes approchées ou ***heuristiques***, qui construisent en un temps acceptable une solution raisonnable sans aucune garantie d'optimalité.

Les méthodes exactes

- **Méthodes arborescentes** : (ou branch & bound. ex: algo. de Little pour le PVC).

  Elles cherchent ‘intelligemment’ parmi l’ensemble des solutions à déterminer une solution optimale sans analyser tous les cas.

Les méthodes exactes

- **Méthodes arborescentes** : (ou branch & bound. ex: algo. de Little pour le PVC).

  Elles cherchent ‘intelligemment’ parmi l’ensemble des solutions à déterminer une solution optimale sans analyser tous les cas.
- **Programmation dynamique** :

  Elles déterminent inductivement une solution en ramenant un problème sur N données à un problème sur N-1 données.

Les méthodes exactes

- **Méthodes arborescentes** : (ou branch & bound. ex: algo. de Little pour le PVC).

  Elles cherchent ‘intelligemment’ parmi l’ensemble des solutions à déterminer une solution optimale sans analyser tous les cas.
- **Programmation dynamique** :

  Elles déterminent inductivement une solution en ramenant un problème sur N données à un problème sur N-1 données.
- La **programmation linéaire en nombres entiers** (PLNE)

  Elle développe principalement 2 types d’approche dont l’une n’est autre qu’une méthode arborescente.

Les heuristiques

- **<u>Méthodes constructives :</u>**

  Elles construisent une solution par une suite de choix partiels et définitifs, c'est à dire sans retour arrière. Lorsque à chaque itération elles prennent le choix le plus avantageux, on parle de méthodes *gloutonnes.*

Les heuristiques

- **<u>Méthodes constructives :</u>**

  Elles construisent une solution par une suite de choix partiels et définitifs, c'est à dire sans retour arrière. Lorsque à chaque itération elles prennent le choix le plus avantageux, on parle de méthodes *gloutonnes.*

- **<u>Recherches locales :</u>**

  On part d'une solution initiale et par transformations successives on construit une suite de solutions de coûts décroissants.

Les heuristiques

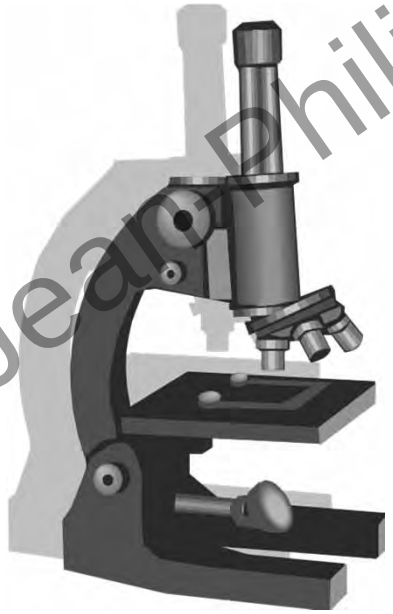
- **<u>Méthodes constructives :</u>**

  Elles construisent une solution par une suite de choix partiels et définitifs, c'est à dire sans retour arrière. Lorsque à chaque itération elles prennent le choix le plus avantageux, on parle de méthodes *gloutonnes.*
- **<u>Recherches locales :</u>**

  On part d'une solution initiale et par transformations successives on construit une suite de solutions de coûts décroissants.
- **<u>Métaheuristiques :</u>**

  Il s'agit d'une méthode de recherche locale ou l'on s'autorise d'augmenter temporairement la fonction économique pour éviter de rester piégé en un minimum local. En général elles convergent en probabilité.

Evaluation des Heuristiques



Evaluation des Heuristiques

- Le problème de l'évaluation des heuristiques est crucial : en effet n'ayant aucune garantie d'optimalité on peut dans une implémentation être proche ou très éloigné d'un optimum.

Evaluation des Heuristiques

- Le problème de l'évaluation des heuristiques est crucial : en effet n'ayant aucune garantie d'optimalité on peut dans une implémentation être proche ou très éloigné d'un optimum.
- Pour une donnée *D* on note *H(D)* le coût d'une heuristique *H* et *Opt(D)* le coût optimal. On appelle *performance relative* le quotient :

qui est ≥1 pour un problème de minimisation et ≤1 pour un problème de maximisation.

Evaluation des Heuristiques

- Le problème de l'évaluation des heuristiques est crucial : en effet n'ayant aucune garantie d'optimalité on peut dans une implémentation être proche ou très éloigné d'un optimum.
- Pour une donnée *D* on note *H(D)* le coût d'une heuristique *H* et *Opt(D)* le coût optimal. On appelle *performance relative* le quotient :

qui est ≥1 pour un problème de minimisation et ≤1 pour un problème de maximisation.

- Elle peut être parfois évaluée à priori ou être imprévisible et constatée à postériori.

Evaluation à priori

- On appelle *performance relative au pire* d'une heuristique *H* :

$$P_H^\infty = \max_{D} \{R_H(D)\}$$

c'est à dire sa plus mauvaise performance relative sur l'ensemble des données.

- Il s'agit d'une garantie de performance obtenue mathématiquement par l'analyse de l'heuristique *H.*
- C'est difficile à obtenir et l'on n'en connaît que pour quelques heuristiques.

Exemples de performances relatives au pire

- on connaît une heuristique de PRP 1.5 pour le PVC (Δ-PVC) lorsque les coûts vérifient l'inégalité triangulaire $C_{ij} \leq C_{ik}+C_{kj}$ et sont symétriques $C_{ij}=C_{ji}$.

Exemples de performances relatives au pire

- on connaît une heuristique de PRP 1.5 pour le PVC (Δ-PVC) lorsque les coûts vérifient l'inégalité triangulaire $C_{ij} \leq C_{ik}+C_{kj}$ et sont symétriques $C_{ij}=C_{ji}$.

  $\Rightarrow$ Le résultat de cet heuristique n'est jamais à plus de 50% de l'optimum.

Exemples de performances relatives au pire

- on connaît une heuristique de PRP 1.5 pour le PVC (Δ-PVC) lorsque les coûts vérifient l'inégalité triangulaire $C_{ij} \leq C_{ik}+C_{kj}$ et sont symétriques $C_{ij}=C_{ji}$.

  $\Rightarrow$ Le résultat de cet heuristique n'est jamais à plus de 50% de l'optimum.
- On ne connaît pas d'heuristique pour le problème de coloration de graphe avec une PRP meilleure que O(N/log N), où N=card(*D*), en particulier bornée.

- Il est bien rare de pouvoir déterminer la PRP. En voici cependant un exemple :

- Il est bien rare de pouvoir déterminer la PRP. En voici cependant un exemple :
- Problème du transversal minimal : dans un graphe non orienté trouver un ensemble de sommets *T*, de cardinal minimal tel que toute arête en soit incidente.

- Il est bien rare de pouvoir déterminer la PRP. En voici cependant un exemple :
- Problème du transversal minimal : dans un graphe non orienté trouver un ensemble de sommets *T*, de cardinal minimal tel que toute arête en soit incidente.

Heuristique :

T est vide

Tant qu'il reste des arêtes

choisir une arête, ajouter ses extrémités à *T*

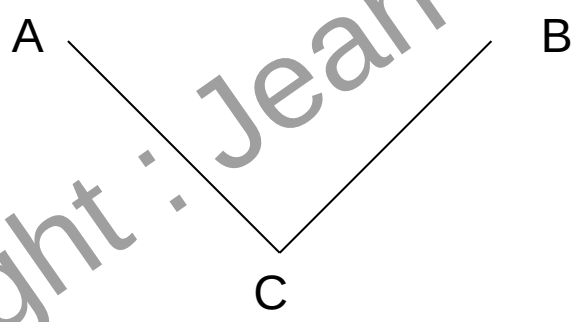
Les supprimer du graphe, ainsi que toute arête incidente.

Fin Tant que

- Le résultat obtenu est clairement un transversal.

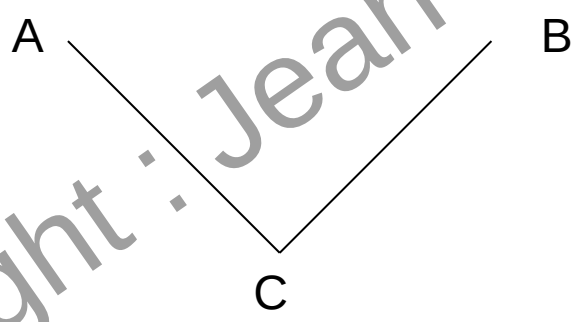
- Tout transversal du graphe, y compris un transversal minimal, doit contenir au moins un sommet de chaque arête choisie dans l'heuristique. Aussi aucun ne peut contenir moins de la moitié des sommets de *T.* Aussi $P_H < 2$.

- Tout transversal du graphe, y compris un transversal minimal, doit contenir au moins un sommet de chaque arête choisie dans l'heuristique. Aussi aucun ne peut contenir moins de la moitié des sommets de *T.* Aussi $P_H < 2$.
- Voici un exemple où $R_H > 2$:



Choisir [A,C]. On obtient un transversal {A,C} de cardinal 2, alors que celui minimal, {C}, est de cardinal 1.

- Tout transversal du graphe, y compris un transversal minimal, doit contenir au moins un sommet de chaque arête choisie dans l'heuristique. Aussi aucun ne peut contenir moins de la moitié des sommets de *T.* Aussi $P_n < 2$.
- Voici un exemple où $R_n > 2$:



Choisir [A,C]. On obtient un transversal {A,C} de cardinal 2, alors que celui minimal, {C}, est de cardinal 1. Ainsi, $P_n \geq 2$.

Evaluation à posteriori

- Le plus souvent, on ne sait pas évaluer la performance relative au pire. On ne peut alors qu'évaluer la performance relative sur les résultats obtenus après exécution de l'heuristique, et encore, à condition de connaître l'optimum.

Evaluation à posteriori

- Le plus souvent, on ne sait pas évaluer la performance relative au pire. On ne peut alors qu'évaluer la performance relative sur les résultats obtenus après exécution de l'heuristique, et encore, à condition de connaître l'optimum.
- Mais en général le calcul de l'optimum ne peut se faire en temps raisonnable.
- On peut obtenir une évaluation moins fine si l'on dispose d'une évaluation de l'optimum : par défaut (minorant) pour un minimum ou par excès (majorant) pour un maximum.

Evaluation à posteriori

- Le plus souvent, on ne sait pas évaluer la performance relative au pire. On ne peut alors qu'évaluer la performance relative sur les résultats obtenus après exécution de l'heuristique, et encore, à condition de connaître l'optimum.
- Mais en général le calcul de l'optimum ne peut se faire en temps raisonnable.
- On peut obtenir une évaluation moins fine si l'on dispose d'une évaluation de l'optimum : par défaut (minorant) pour un minimum ou par excès (majorant) pour un maximum.
- Exemple : Dans la recherche d'un minimum, pour une donnée *D*, en notant *B(D)* une évaluation par défaut de *Opt(D)* :

$$1 \leq R_H(D) = \frac{H(D)}{Opt(D)} \leq \frac{H(D)}{B(D)}$$

- En particulier si *H(D)/B(D)*=1, on est sûr d'avoir atteint un optimum.

Exemples de Bornes min *B(D)* pour le PVC

Soit D la matrice n×n des coûts :

- borne inf naïve :

  la somme des n plus petits éléments de D.

Exemples de Bornes min *B(D)* pour le PVC

Soit D la matrice n×n des coûts :

- borne inf naïve :

  la somme des n plus petits éléments de D.

- borne inf moins naïve :

  la somme des minima de chaque ligne.

Exemples de Bornes min *B(D)* pour le PVC

Soit D la matrice n×n des coûts :

- borne inf naïve :

  la somme des n plus petits éléments de D.

- borne inf moins naïve :

  la somme des minima de chaque ligne.

- borne inf de Little :

  la somme des nombres soustraits aux lignes & colonnes dans la ‘réduction de la matrice’ (cf. algo de Little).

Exemples de Bornes min *B(D)* pour le PVC

- Si D est symétrique (graphe non orienté) :
- Si dans un circuit hamiltonien on supprime une arête on obtient un arbre recouvrant (i.e. contenant chaque sommet) de coût inférieur.
- borne inf : coût minimal d'un arbre recouvrant.

Exemples de Bornes min *B(D)* pour le PVC

- Si D est symétrique (graphe non orienté) :
- Si dans un circuit hamiltonien on supprime une arête on obtient un arbre recouvrant (i.e. contenant chaque sommet) de coût inférieur.
- borne inf : coût minimal d'un arbre recouvrant.
- C'est un problème facile (en $O(N^2)$), dont voici un algorithme :

Initialement l'arbre T est réduit à un sommet

Pour k variant de 1 jusqu'à N-1

Chercher l'arête [i , j] de coût min t.q. $i \in T$ et $j \notin T$

Ajouter [i , j] à T

Fin Pour

- **Exemple :**

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

- **Exemple :**
- Borne naïve :

  $B_1(D)=6$.

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

- **Exemple :**
- Borne naïve :

  $B_1(D)=6$.
- Borne moins naïve :

  $B_2(D)=7$.

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

- **Exemple :**
- Borne naïve :

  $B_1(D)=6$.
- Borne moins naïve :

  $B_2(D)=7$.
- Borne de Little :

  $B_3(D)=9$.

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

- **Exemple :**
- Borne naïve :

  $B_1(D)=6$.
- Borne moins naïve :

  $B_2(D)=7$.
- Borne de Little :

  $B_3(D)=9$.
- Coût minimal d'un arbre recouvrant :

  $B_4(D)=7$ (arbre : [A,B],[B,E],[E,C],[C,D]).

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

- **Exemple :**
- Borne naïve :

  $B_1(D)=6$.
- Borne moins naïve :

  $B_2(D)=7$.
- Borne de Little :

  $B_3(D)=9$.
- Coût minimal d'un arbre recouvrant :

  $B_4(D)=7$ (arbre : [A,B],[B,E],[E,C],[C,D]).
- **Heuristique du plus proche voisin :**

  Partir du sommet A et construire un cycle en reliant le dernier sommet visité au plus proche sommet libre.

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

- **Exemple :**
- Borne naïve :

  $B_1(D)=6$.
- Borne moins naïve :

  $B_2(D)=7$.
- Borne de Little :

  $B_3(D)=9$.
- Coût minimal d'un arbre recouvrant :

  $B_4(D)=7$ (arbre : [A,B],[B,E],[E,C],[C,D]).
- **Heuristique du plus proche voisin :**

  Partir du sommet A et construire un cycle en reliant le dernier sommet visité au plus proche sommet libre. On obtient A-B-E-C-D-A de coût 3+1+2+1+5=12.

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

- **Exemple :**
- Borne naïve :

  $B_1(D)=6$.
- Borne moins naïve :

  $B_2(D)=7$.
- Borne de Little :

  $B_3(D)=9$.

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

- Coût minimal d'un arbre recouvrant :

  $B_4(D)=7$ (arbre : [A,B],[B,E],[E,C],[C,D]).
- **Heuristique du plus proche voisin :**

  Partir du sommet A et construire un cycle en reliant le dernier sommet visité au plus proche sommet libre. On obtient A-B-E-C-D-A de coût 3+1+2+1+5=12. La performance relative au pire sur C est au plus 12/9=1.33, soit une erreur d'au plus 33%. En fait le coût optimal est 12 (appliquer l'algo de Little). Par chance c'est optimal...

Evaluation statistique(à posteriori)

- On est souvent contraint à cette extrémité pour les problèmes sur de grande taille de données.

Evaluation statistique(à posteriori)

- On est souvent contraint à cette extrémité pour les problèmes sur de grande taille de données.
- On compare diverses méthodes sur une série de problèmes générés aléatoirement (ex : pour le PVC sur N sommets attribuer aléatoirement les coûts à l'aide d'une loi de probabilité uniforme sur $[0,C_{max}]$, TSPLIB) aux méthodes existantes.

Evaluation statistique(à posteriori)

- On est souvent contraint à cette extrémité pour les problèmes sur de grande taille de données.
- On compare diverses méthodes sur une série de problèmes générés aléatoirement (ex : pour le PVC sur N sommets attribuer aléatoirement les coûts à l'aide d'une loi de probabilité uniforme sur $[0,C_{max}]$, TSPLIB) aux méthodes existantes.
- On compare leur résultat, leur temps d'exécution minimaux, maximaux, moyens, les écarts types, à un résultat optimal (si c'est possible) ou à une borne inférieure.

Evaluation statistique(à posteriori)

- On est souvent contraint à cette extrémité pour les problèmes sur de grande taille de données.
- On compare diverses méthodes sur une série de problèmes générés aléatoirement (ex : pour le PVC sur N sommets attribuer aléatoirement les coûts à l'aide d'une loi de probabilité uniforme sur $[0,C_{max}]$, TSPLIB) aux méthodes existantes.
- On compare leur résultat, leur temps d'exécution minimaux, maximaux, moyens, les écarts types, à un résultat optimal (si c'est possible) ou à une borne inférieure.
- Cela présente par ailleurs un avantage : même si l'on connaît la PRP, elle peut n'être atteint que dans des cas pathologiques. Une performance 'en moyenne' présente un grand intérêt.

Evaluation statistique(à posteriori)

- On est souvent contraint à cette extrémité pour les problèmes sur de grande taille de données.
- On compare diverses méthodes sur une série de problèmes générés aléatoirement (ex : pour le PVC sur N sommets attribuer aléatoirement les coûts à l'aide d'une loi de probabilité uniforme sur $[0,C_{max}]$, TSPLIB) aux méthodes existantes.
- On compare leur résultat, leur temps d'exécution minimaux, maximaux, moyens, les écarts types, à un résultat optimal (si c'est possible) ou à une borne inférieure.
- Cela présente par ailleurs un avantage : même si l'on connaît la PRP, elle peut n'être atteint que dans des cas pathologiques. Une performance 'en moyenne' présente un grand intérêt.
- Toute amélioration substantielle est intéressante, comme par exemple la meilleure heuristique 'en moyenne ' selon la taille des données.

Heuristiques gloutonnes

Problème du sac à dos

- n objets disponibles.
- L'objet i a un poids $p_i>0$ et une valeur $v_i>0$.
- Emporter un sous-ensemble d'objets de valeur maximale dans un sac de capacité P.

Problème du sac à dos

- n objets disponibles.
- L'objet i a un poids $p_i>0$ et une valeur $v_i>0$.
- Emporter un sous-ensemble d'objets de valeur maximale dans un sac de capacité P.
- Dans le problème du sac à dos 0-1, chaque objet est en unique exemplaire.
- $\sum_i p_i > P$ sinon le problème est trivial.

Problème du sac à dos

- n objets disponibles.
- L'objet i a un poids $p_i>0$ et une valeur $v_i>0$.
- Emporter un sous-ensemble d'objets de valeur maximale dans un sac de capacité P.
- Dans le problème du sac à dos en nombres entiers chaque objet est en une infinité d'exemplaire.

Heuristique gloutonne

Pour le sac à dos en 0-1 :

- Numéroter les objets i par v_i/p_i décroissants.
- Poids=0 ; valeur=0
- Pour i variant de 1 jusqu'à n
  - Si Poids + $p_i \leq P$ alors
    - Poids=Poids+p_i
    - Valeur=Valeur+v_i
  - Fin Si
- Fin pour

Heuristique gloutonne

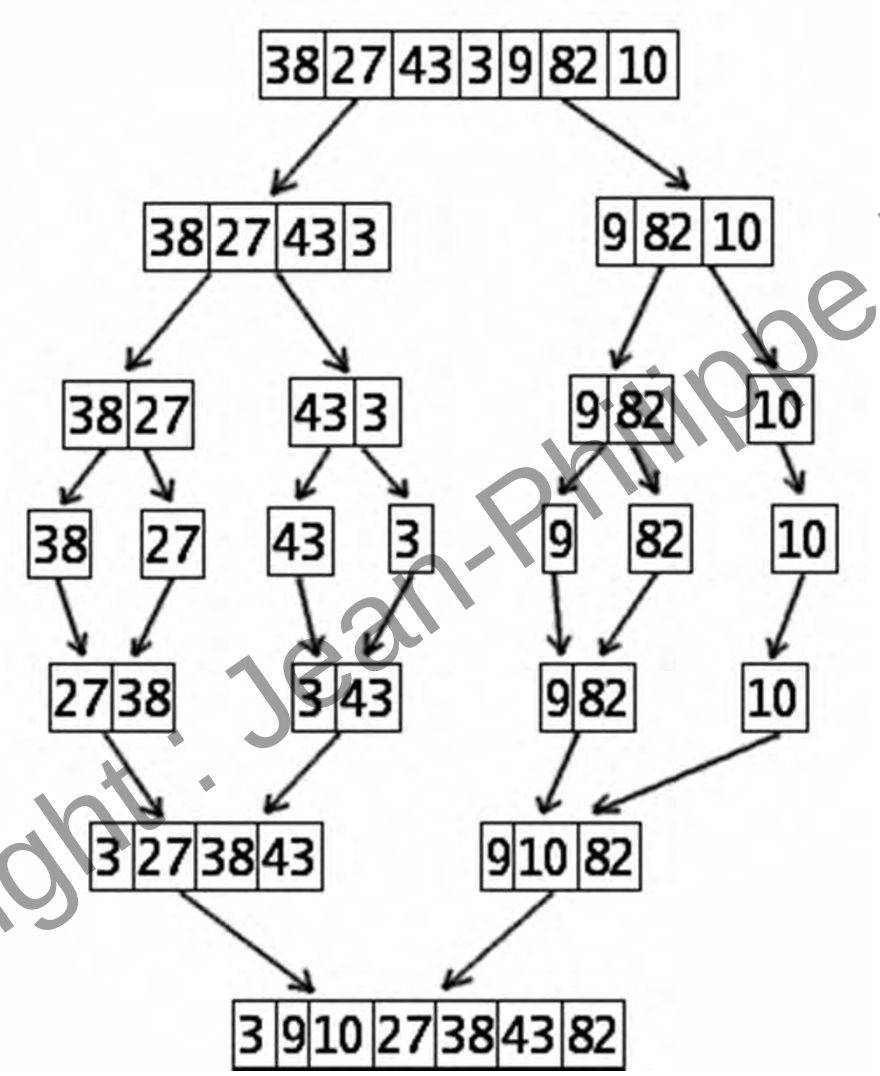
<u>Pour le sac à dos en nombres entiers :</u>

- Numéroter les objets i par v_i/p_i décroissants.
- Poids=0 ; valeur=0
- Pour i variant de 1 jusqu'à n
  - Tant que Poids + $p_i \leq P$
    - Poids=Poids+p_i
    - Valeur=Valeur+v_i
  - Fin Tant que
- Fin pour

- Une heuristique gloutonne consistant à prendre d’abord les objets les plus chers, ou les plus lourds n’est pas très bonne : elle ignore une partie des données.

- Une heuristique gloutonne consistant à prendre d'abord les objets les plus chers, ou les plus lourds n'est pas très bonne : elle ignore une partie des données.
- La complexité de l'heuristique est en $O(n^2)$ avec un algorithme de tri naïf (tri à bulle, tri par insertion,...). En O(n log n) avec un algorithme de tri efficace (tri fusion, tri par tas...)

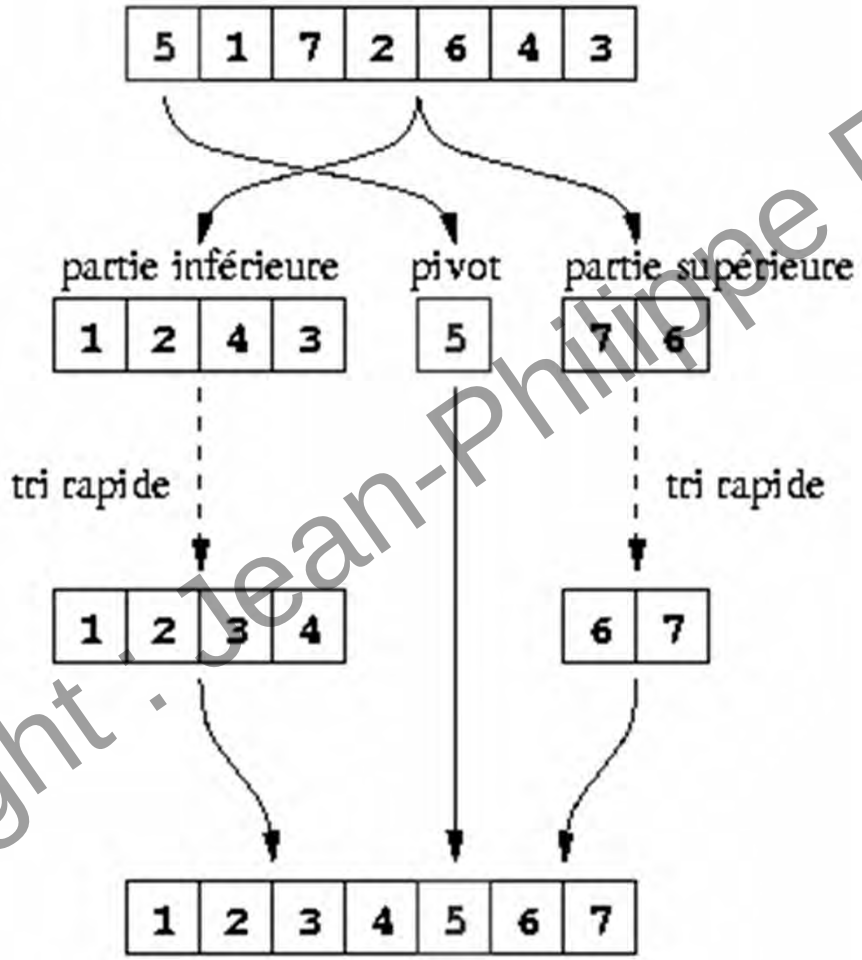
TRI FUSION



Complexité : O(n log n)

Principe fondamental : DIVISER POUR REGNER !

TRI RAPIDE



Complexité $O(n^2)$, en moyenne $O(n \log n)$.

Complexité $O(n \log n)$ si l'on choisit bien le pivot (introsort...)

- Pour le sac à dos en nombre entiers l'heuristique n'est pas mauvaise, de PRP 0,5.

- Pour le sac à dos en nombre entiers l'heuristique n'est pas mauvaise, de PRP 0,5.
- Exemple : n=2, $p_1=v_1=k+1$, $p_2=v_2=k$, P=2k. L'algorithme (peut) trouve(r) k+1 tandis que l'optimum est 2k. Ainsi P_H peut-être aussi proche que l'on souhaite de 0.5 ($k\rightarrow\infty$).

- Pour le sac à dos en nombre entiers l'heuristique n'est pas mauvaise, de PRP 0,5.
- Exemple : n=2, $p_1=v_1=k+1$, $p_2=v_2=k$, P=2k. L'algorithme (peut) trouve(r) k+1 tandis que l'optimum est 2k. Ainsi P_H peut-être aussi proche que l'on souhaite de 0.5 ($k\rightarrow\infty$).
- Avec $M=P/\max(p_i)$, on montre que $P_H=M/(M+1)$: plus les objets sont légers meilleure est la performance de l'heuristique.

- L’heuristique est mauvaise dans le cas du problème du sac à dos en 0-1.
- Exemple : n=2, $p_1=v_1=1$, $p_2=v_2=P$.

- L’heuristique est mauvaise dans le cas du problème du sac à dos en 0-1.
- Exemple : n=2, $p_1=v_1=1$, $p_2=v_2=P$.
- Elle (peut) trouve(r) 1 alors que l’optimum est P.
- Ainsi $P_H \rightarrow 0$ quand $P \rightarrow \infty$.

- L’heuristique est mauvaise dans le cas du problème du sac à dos en 0-1.
- Exemple : n=2, $p_1=v_1=1$, $p_2=v_2=P$.
- Elle (peut) trouve(r) 1 alors que l’optimum est P.
- Ainsi $P_H \rightarrow 0$ quand $P \rightarrow \infty$.
- Le comportement moyen est bon en moyenne.
- Sans utilité pour les problèmes d’empilement ($p_i=v_i$)

Problème de mise en boîte

- Ranger n objets de tailles a_i, i=1,..,n, dans un nombre minimal de boîtes de capacité b.

Problème de mise en boîte

- Ranger n objets de tailles a_i, i=1,..,n, dans un nombre minimal de boîtes de capacité b.
- First-Fit (FF) : place à l'itération i l'objet i dans la première boîte qui convienne.

Problème de mise en boîte

- Ranger n objets de tailles a_i, i=1,..,n, dans un nombre minimal de boîtes de capacité b.
- First-Fit (FF) : place à l'itération i l'objet i dans la première boîte qui convienne.
- Best-Fit (BF) : place à l'itération i l'objet i dans la boîte qu'il remplit le mieux.

Problème de mise en boîte

- Ranger n objets de tailles a_i, i=1,..,n, dans un nombre minimal de boîtes de capacité b.
- First-Fit (FF) : place à l'itération i l'objet i dans la première boîte qui convienne.
- Best-Fit (BF) : place à l'itération i l'objet i dans la boîte qu'il remplit le mieux.

Intuitivement il vaut mieux placer d'abord les objets les plus gros.

Problème de mise en boîte

- Ranger n objets de tailles a_i, i=1,..,n, dans un nombre minimal de boîtes de capacité b.
- First-Fit (FF) : place à l'itération i l'objet i dans la première boîte qui convienne.
- Best-Fit (BF) : place à l'itération i l'objet i dans la boîte qu'il remplit le mieux.

Intuitivement il vaut mieux placer d'abord les objets les plus gros.

- Avec un tri préalable par ordre décroissant sur la taille des objets on obtient :
- First-Fit-Decreasing (FFD).
- Best-Fit-decreasing (BFD).

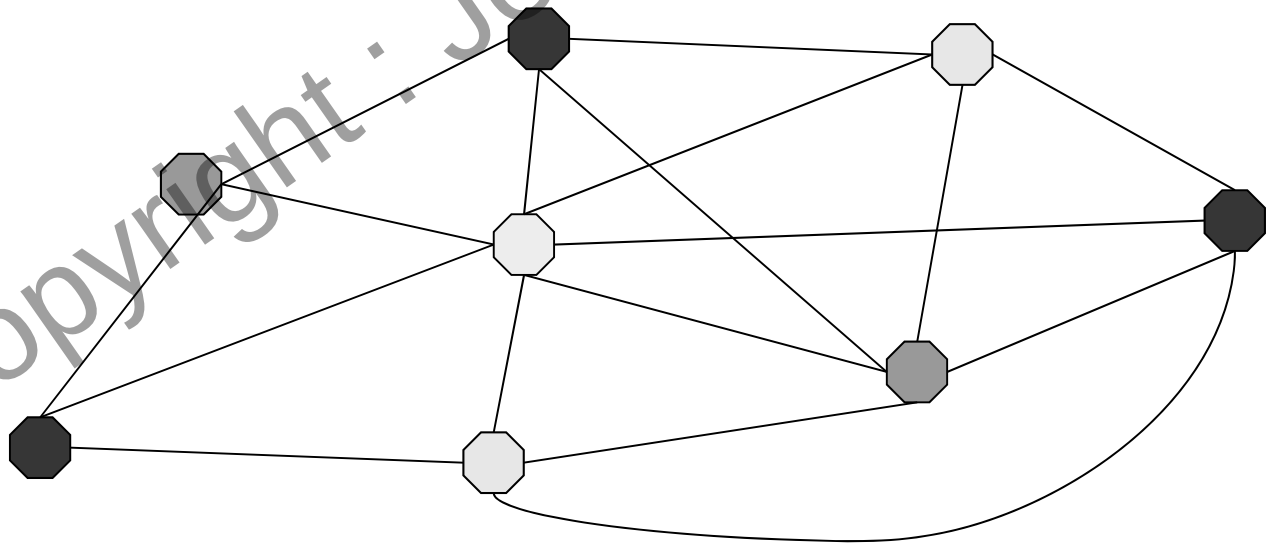
- FF et BF ont une PRP de 2.
- FFD et BFD ont une PRP de 1.5.
- BFD est meilleure que FFD en moyenne.

- FF et BF ont une PRP de 2.
- FFD et BFD ont une PRP de 1.5.
- BFD est meilleure que FFD en moyenne.
- On a prouvé des résultats + fins :
- $FF(d) \leq 17/10\ OPT(d)+2$ (tend vers 70%)
- $FFD(d) \leq 11/9\ OPT(d)+4$ (tend vers 22.2%)

- FF et BF ont une PRP de 2.
- FFD et BFD ont une PRP de 1.5.
- BFD est meilleure que FFD en moyenne.
- On a prouvé des résultats + fins :
- $FF(d) \leq 17/10\ OPT(d)+2$ (tend vers 70%)
- $FFD(d) \leq 11/9\ OPT(d)+4$ (tend vers 22.2%)
- Ca marche mieux quand la taille diminue :
- Si $a_i \leq B/2$ la PRP tend vers 18.3%.
- Si $B/3<a_i<B$ c'est un problème de couplage, polynomial. Dans ce cas FFD donne toujours l'optimum.

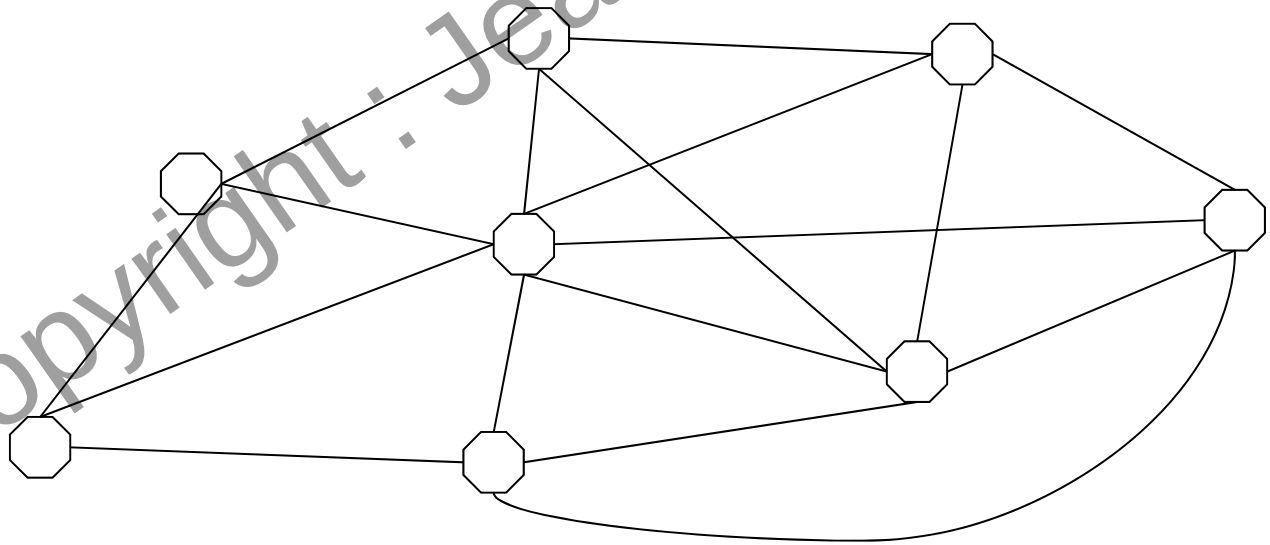
Heuristiques séquentielles pour la coloration de graphes

- Problème : Dans un graphe non orienté, colorier chaque sommet de sorte que deux sommets adjacents soient de couleur différente, avec un nombre minimal de couleurs.

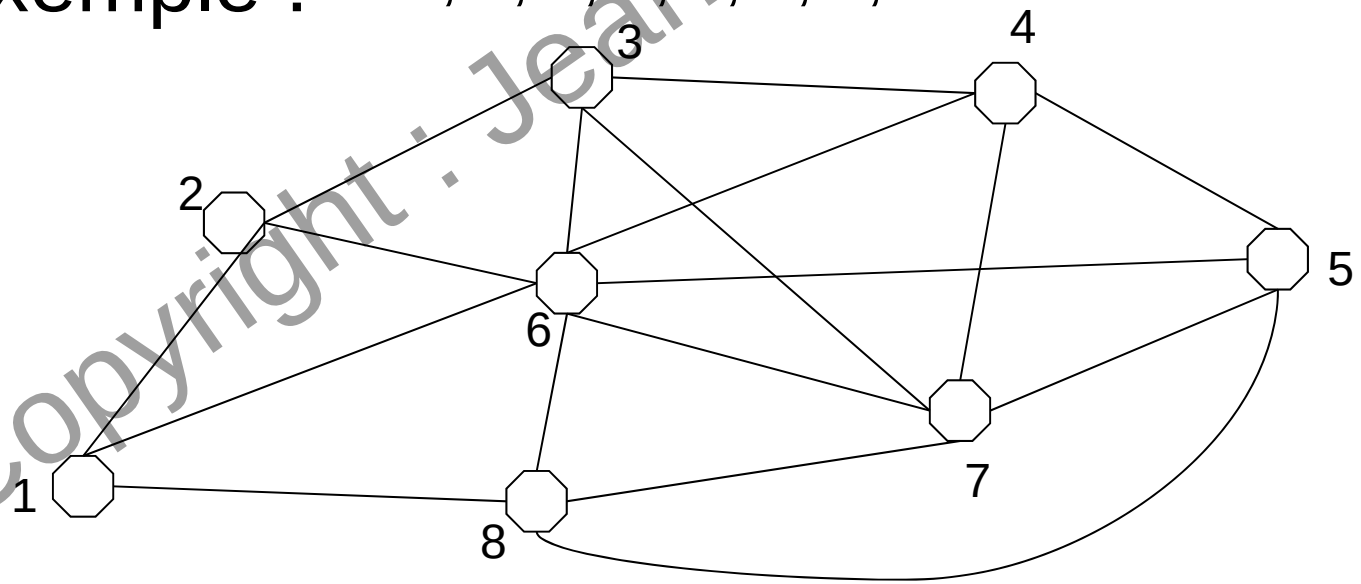


- **<u>Heuristique séquentielle :</u>**
  - Se donner n couleurs indicées 1,2,..,n.
  - Ordonner les sommets : $S_1,S_2,...,S_n$.
  - Colorier successivement les sommets par la plus petite couleur non utilisée par ses voisins.

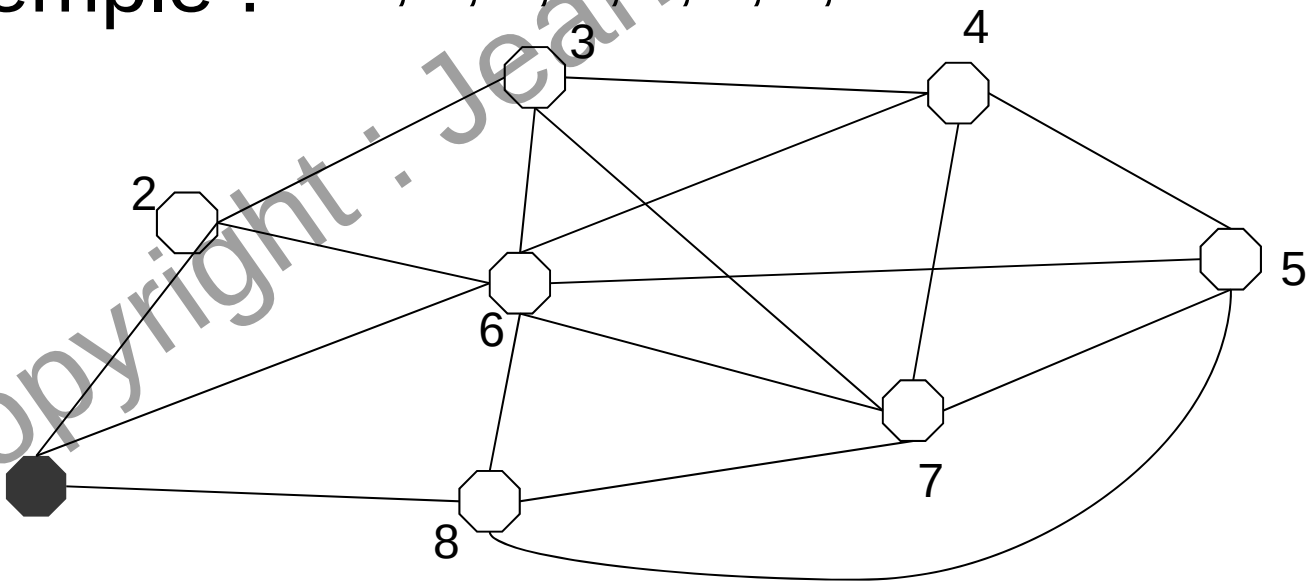
- **<u>Heuristique séquentielle :</u>**
  - Se donner n couleurs indicées 1,2,..,n.
  - Ordonner les sommets : $S_1,S_2,...,S_n$.
  - Colorier successivement les sommets par la plus petite couleur non utilisée par ses voisins.
- C'est l'heuristique <u>First Fit Sequential (FFS)</u>
- Exemple :



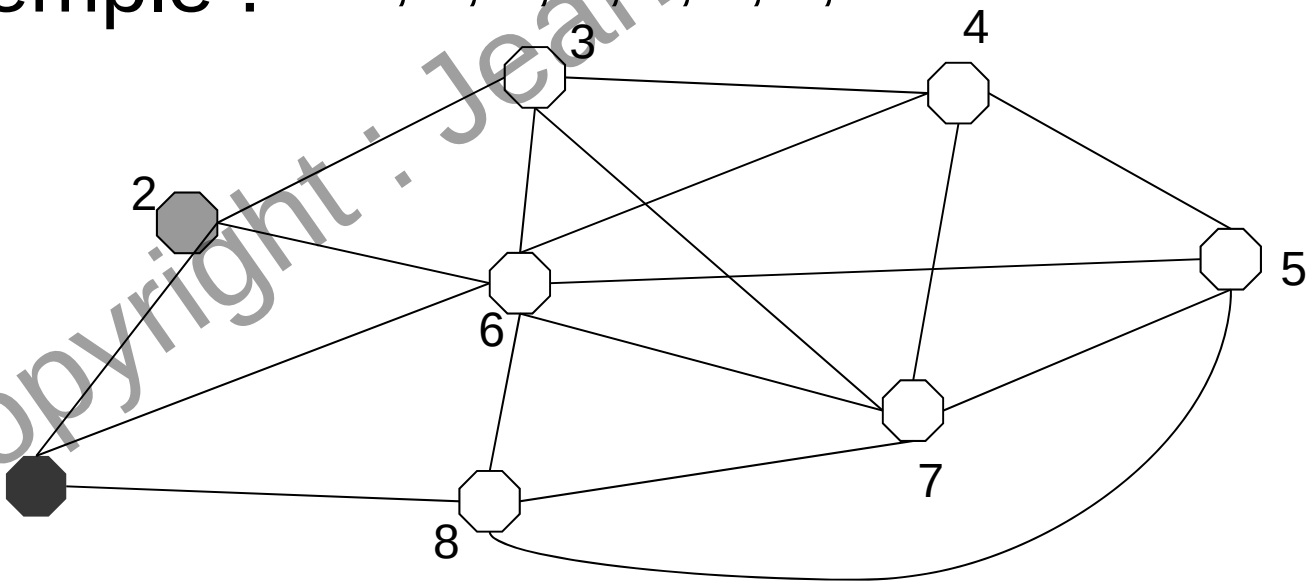
- **<u>Heuristique séquentielle :</u>**
  - Se donner n couleurs indicées 1,2,..,n.
  - Ordonner les sommets : $S_1, S_2, \ldots, S_n$.
  - Colorier successivement les sommets par la plus petite couleur non utilisée par ses voisins.
- C'est l'heuristique <u>First Fit Sequential (FFS)</u>
- Exemple : 1, 2, 3, 4, 5, 6, 7, 8



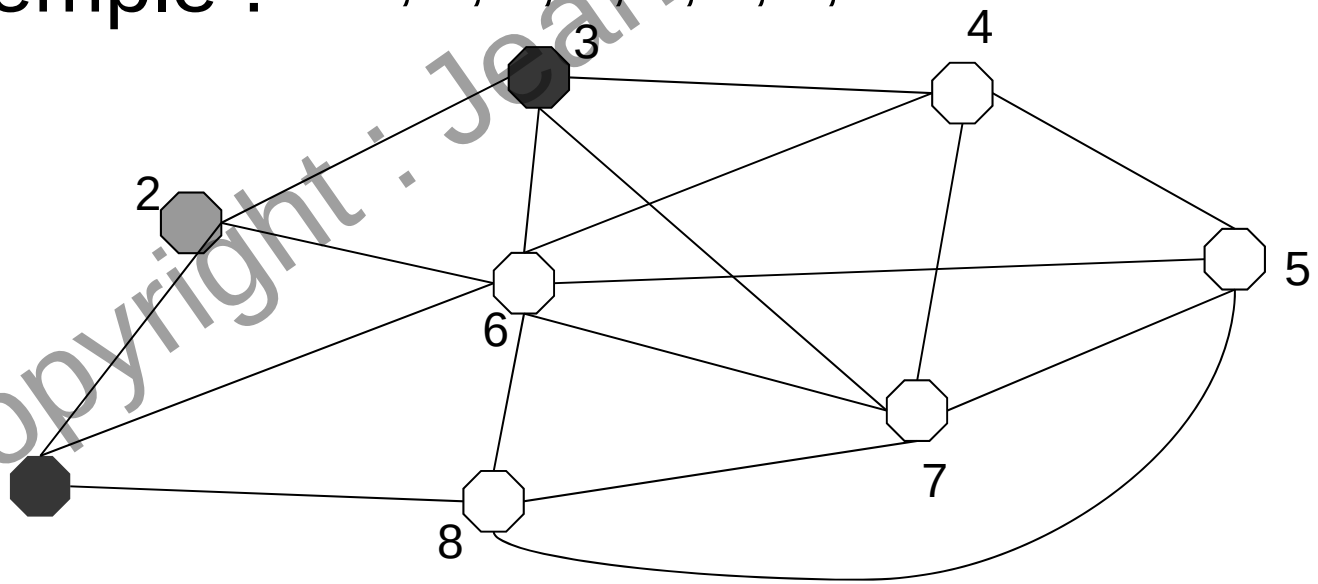
- **<u>Heuristique séquentielle :</u>**
  - Se donner n couleurs indicées 1,2,..,n.
  - Ordonner les sommets : $S_1, S_2, \ldots, S_n$.
  - Colorier successivement les sommets par la plus petite couleur non utilisée par ses voisins.
- C'est l'heuristique <u>First Fit Sequential (FFS)</u>
- Exemple : 1, 2, 3, 4, 5, 6, 7, 8



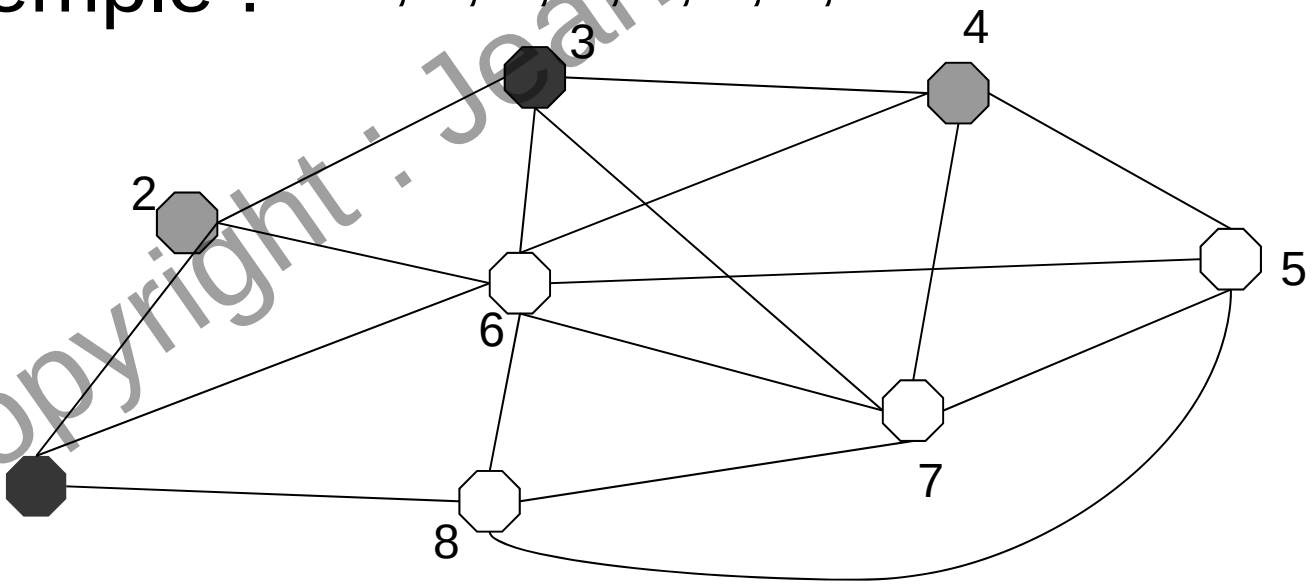
- **<u>Heuristique séquentielle :</u>**
  - Se donner n couleurs indicées 1,2,..,n.
  - Ordonner les sommets : $S_1, S_2, \ldots, S_n$.
  - Colorier successivement les sommets par la plus petite couleur non utilisée par ses voisins.
- C'est l'heuristique <u>First Fit Sequential (FFS)</u>
- Exemple : 1, 2, 3, 4, 5, 6, 7, 8



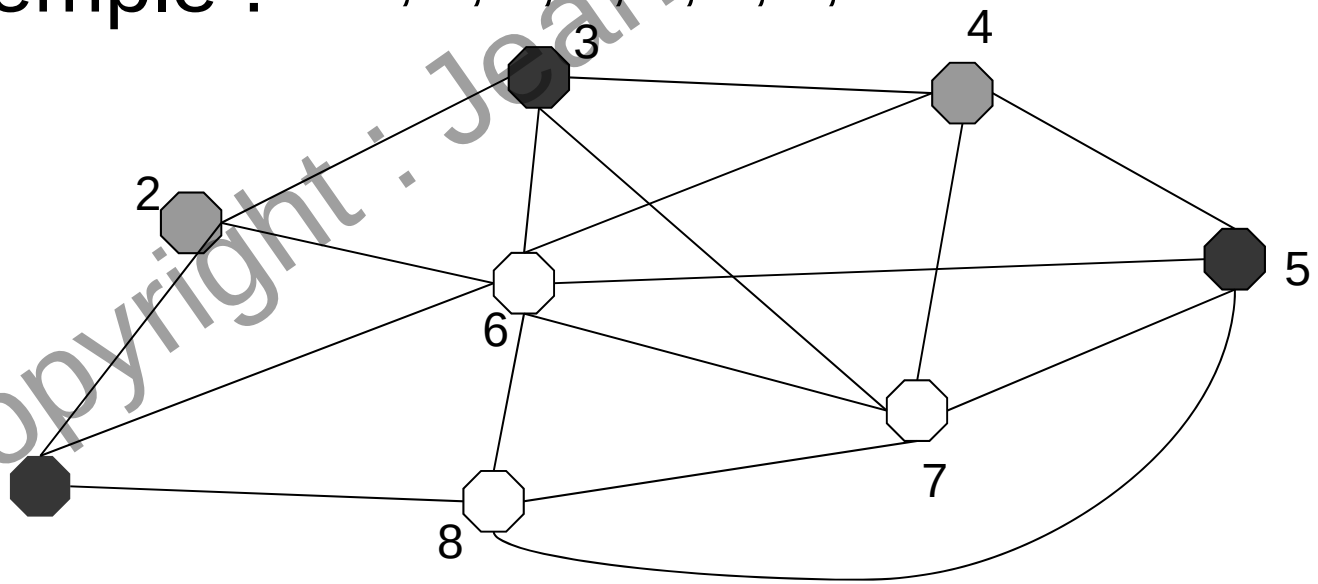
- **<u>Heuristique séquentielle :</u>**
  - Se donner n couleurs indicées 1,2,..,n.
  - Ordonner les sommets : $S_1, S_2, \ldots, S_n$.
  - Colorier successivement les sommets par la plus petite couleur non utilisée par ses voisins.
- C'est l'heuristique <u>First Fit Sequential (FFS)</u>
- Exemple : 1, 2, 3, 4, 5, 6, 7, 8



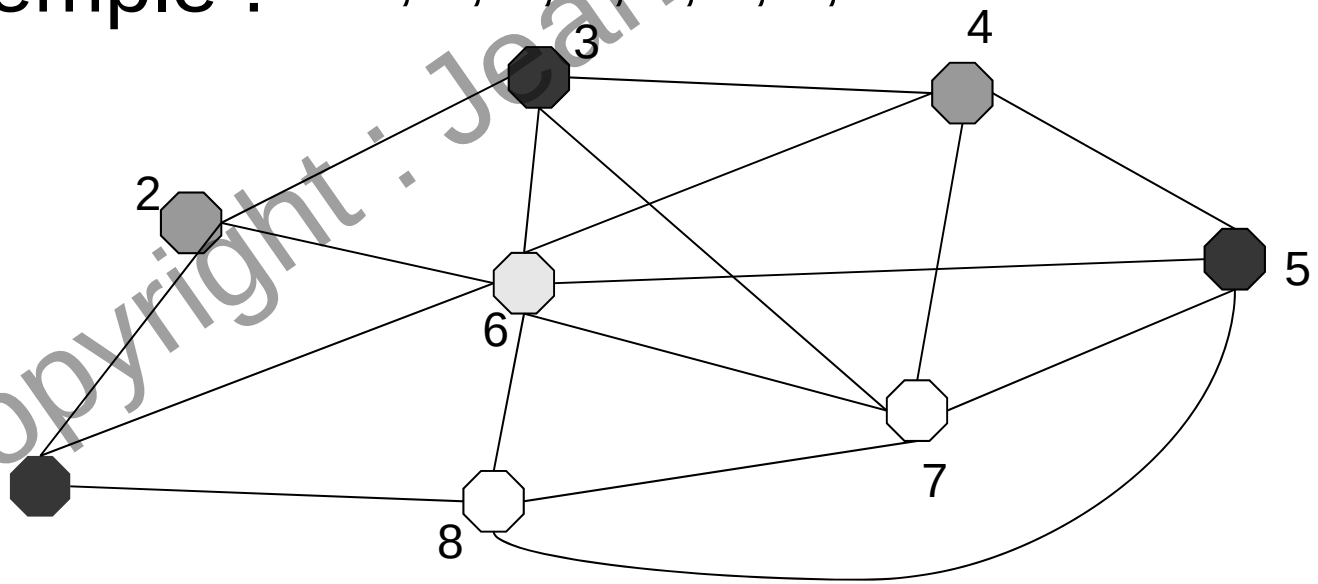
- **<u>Heuristique séquentielle :</u>**
  - Se donner n couleurs indicées 1,2,..,n.
  - Ordonner les sommets : $S_1, S_2, \ldots, S_n$.
  - Colorier successivement les sommets par la plus petite couleur non utilisée par ses voisins.
- C'est l'heuristique <u>First Fit Sequential (FFS)</u>
- Exemple : 1, 2, 3, 4, 5, 6, 7, 8



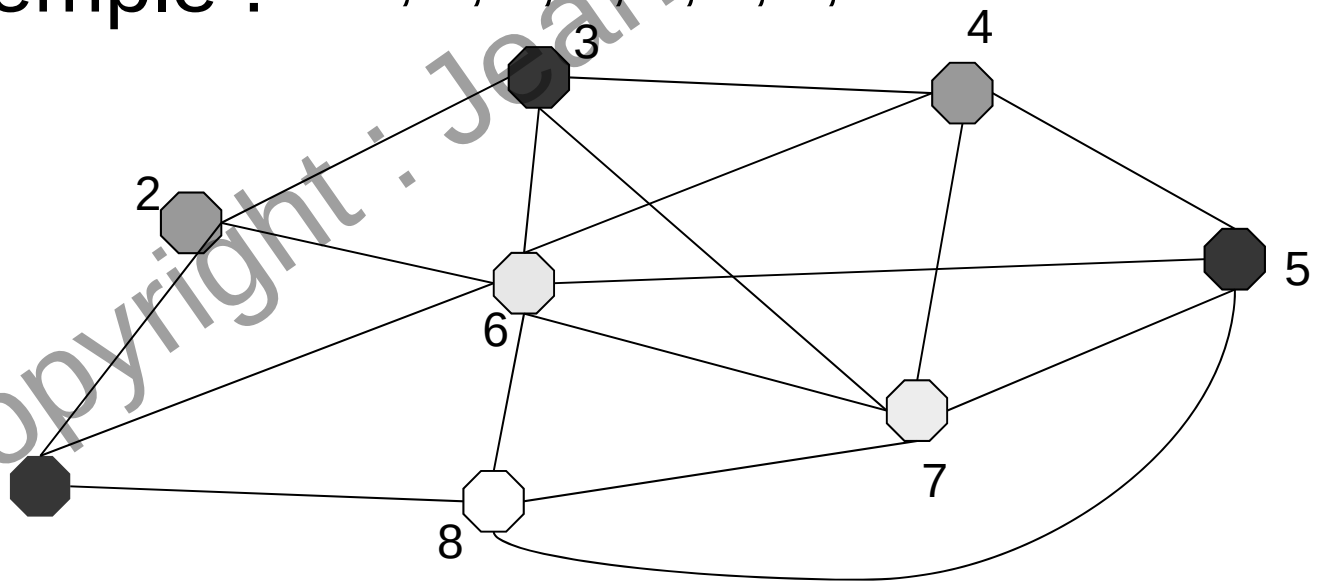
- **<u>Heuristique séquentielle :</u>**
  - Se donner n couleurs indicées 1,2,..,n.
  - Ordonner les sommets : $S_1, S_2, \ldots, S_n$.
  - Colorier successivement les sommets par la plus petite couleur non utilisée par ses voisins.
- C'est l'heuristique <u>First Fit Sequential (FFS)</u>
- Exemple : 1, 2, 3, 4, 5, 6, 7, 8



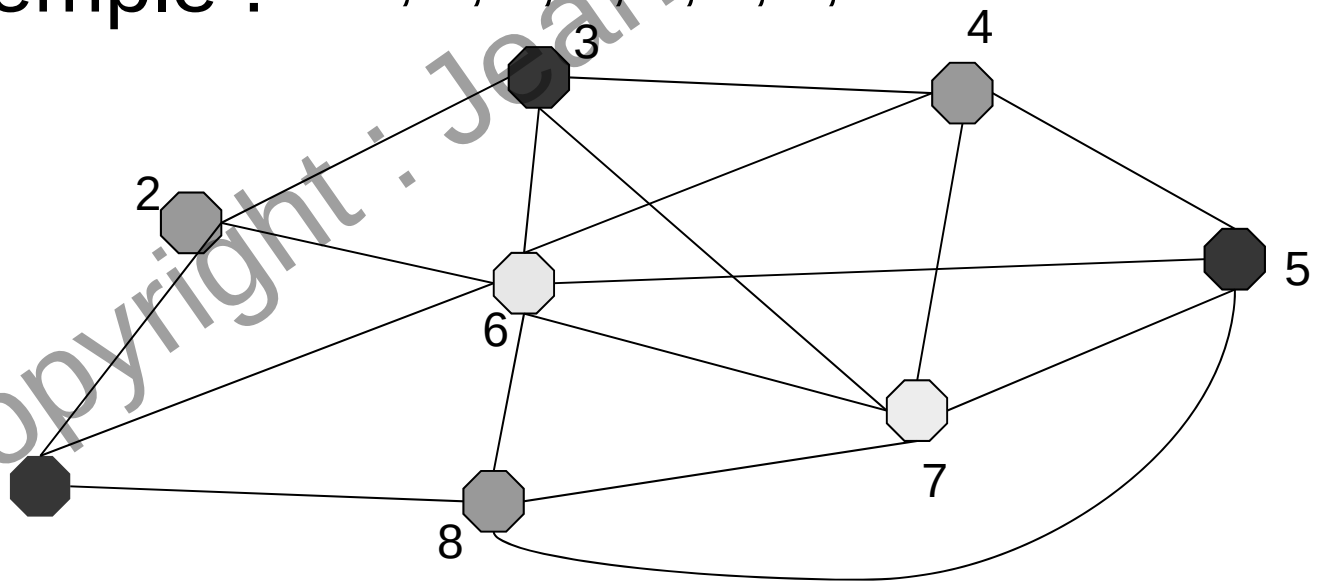
- **<u>Heuristique séquentielle :</u>**
  - Se donner n couleurs indicées 1,2,..,n.
  - Ordonner les sommets : $S_1, S_2, \ldots, S_n$.
  - Colorier successivement les sommets par la plus petite couleur non utilisée par ses voisins.
- C'est l'heuristique <u>First Fit Sequential (FFS)</u>
- Exemple : 1, 2, 3, 4, 5, 6, 7, 8



- **<u>Heuristique séquentielle :</u>**
  - Se donner n couleurs indicées 1,2,..,n.
  - Ordonner les sommets : $S_1, S_2, \ldots, S_n$.
  - Colorier successivement les sommets par la plus petite couleur non utilisée par ses voisins.
- C'est l'heuristique <u>First Fit Sequential (FFS)</u>
- Exemple : 1, 2, 3, 4, 5, 6, 7, 8

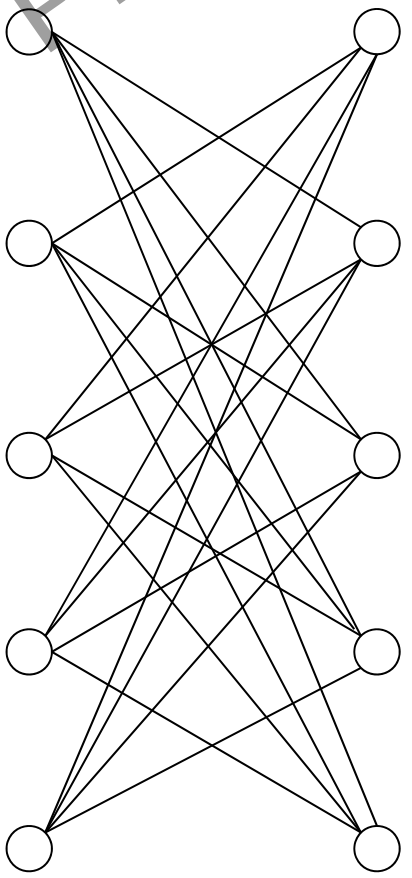


- **<u>Heuristique séquentielle :</u>**
  - Se donner n couleurs indicées 1,2,..,n.
  - Ordonner les sommets : $S_1,S_2,...,S_n$.
  - Colorier successivement les sommets par la plus petite couleur non utilisée par ses voisins.
- C'est l'heuristique <u>First Fit Sequential (FFS)</u>
- Exemple : 1, 2, 3, 4, 5, 6, 7, 8



- Cette heuristique peut être très mauvaise :

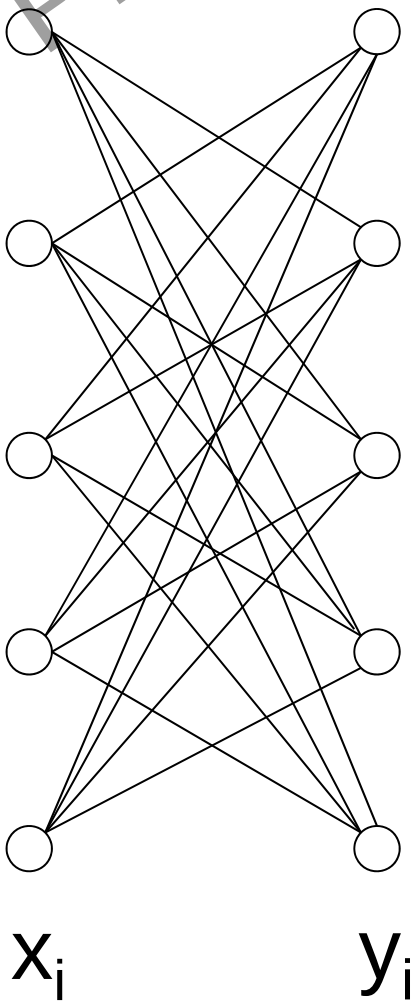
- Cette heuristique peut être très mauvaise :
- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.



- Cette heuristique peut être très mauvaise :

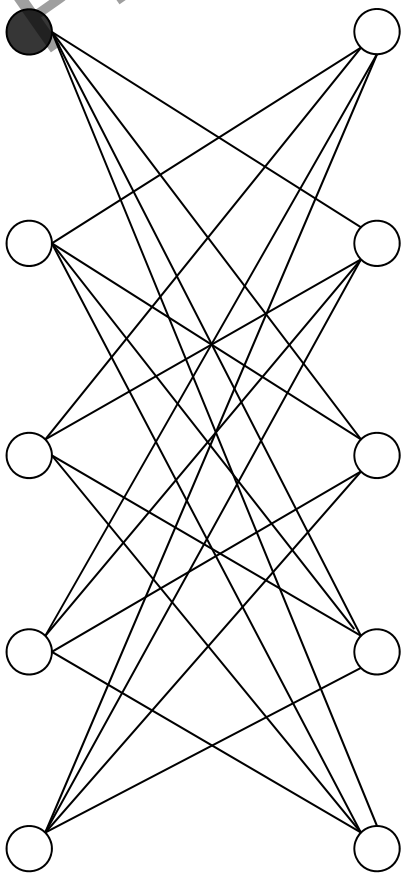
- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.

- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum



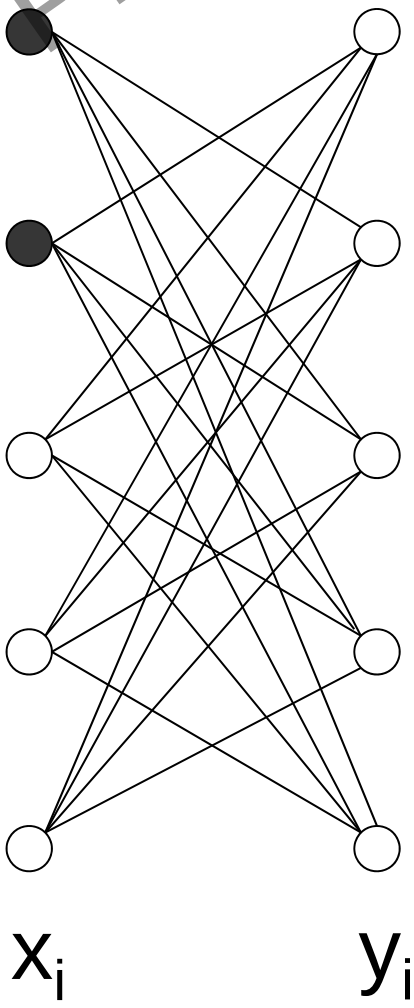
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum



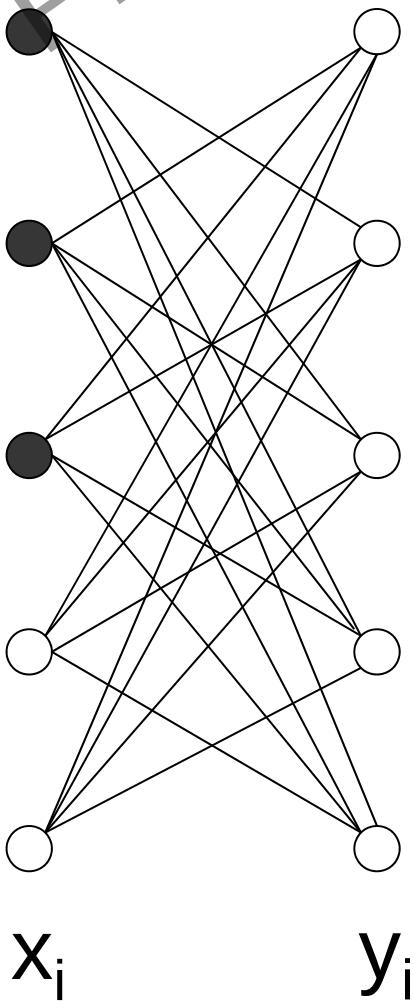
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum



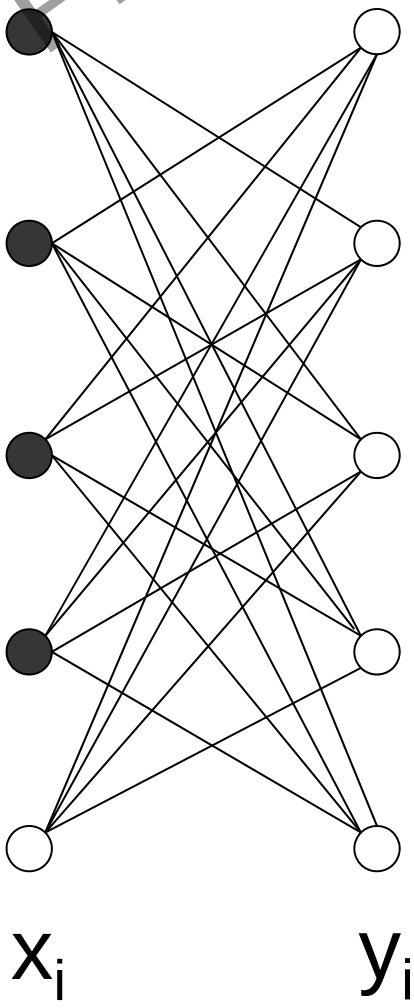
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum



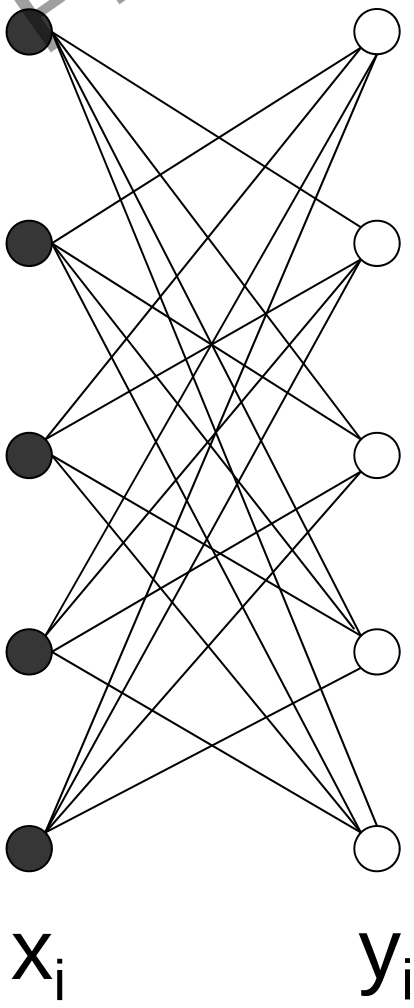
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum



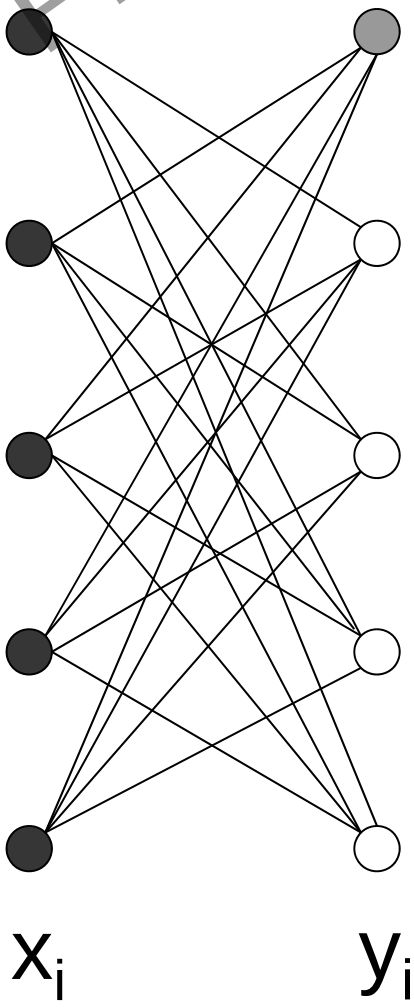
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum



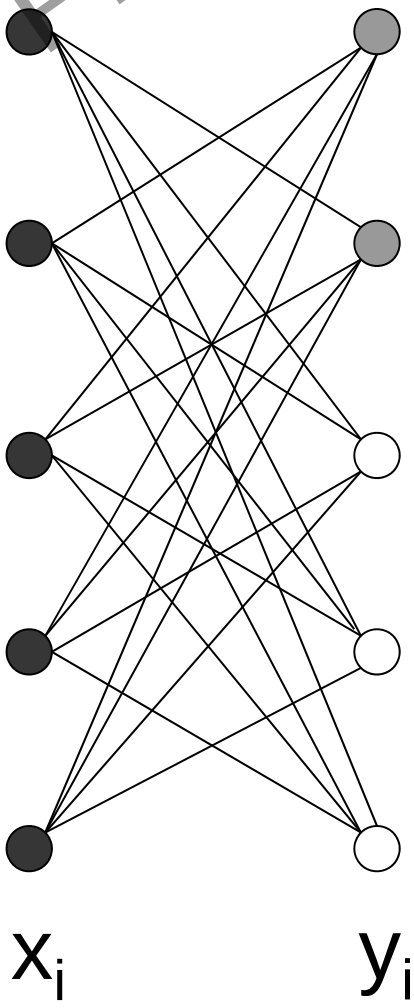
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum



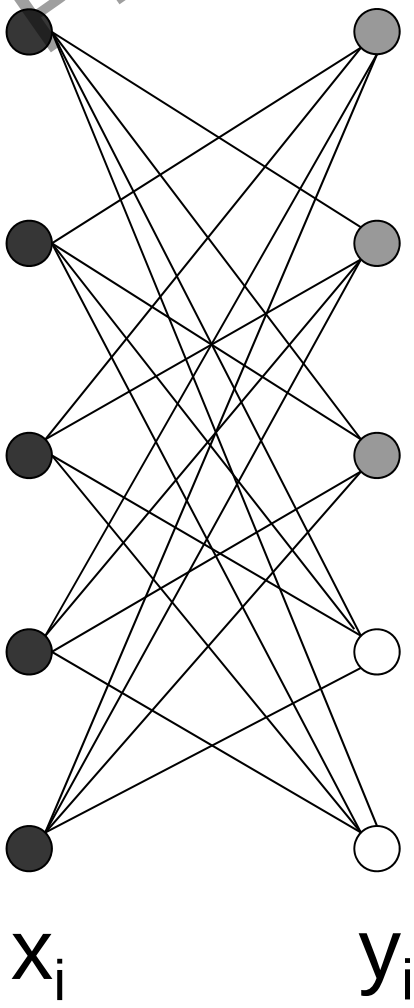
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum



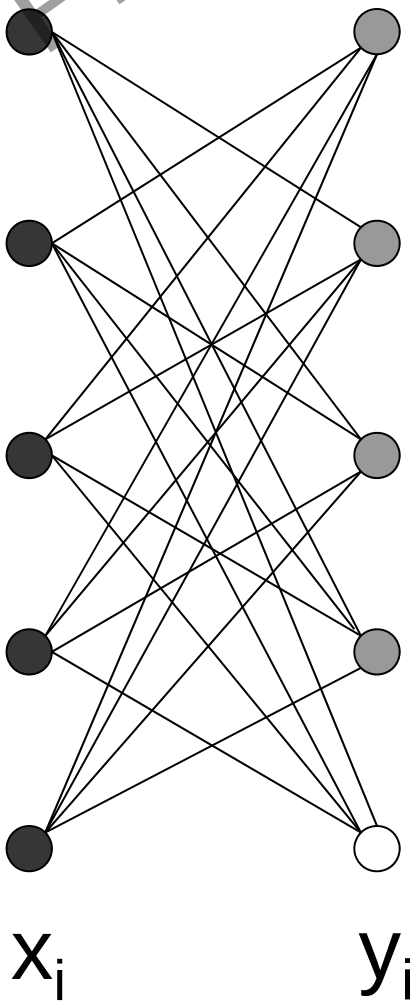
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum



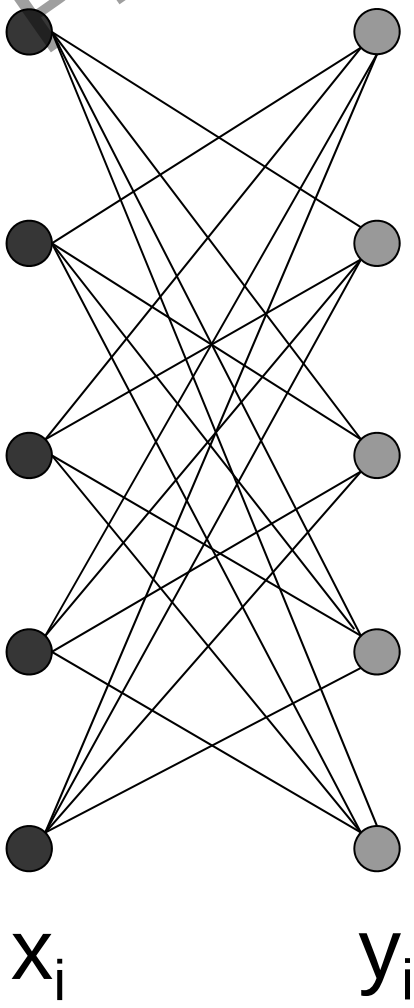
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum



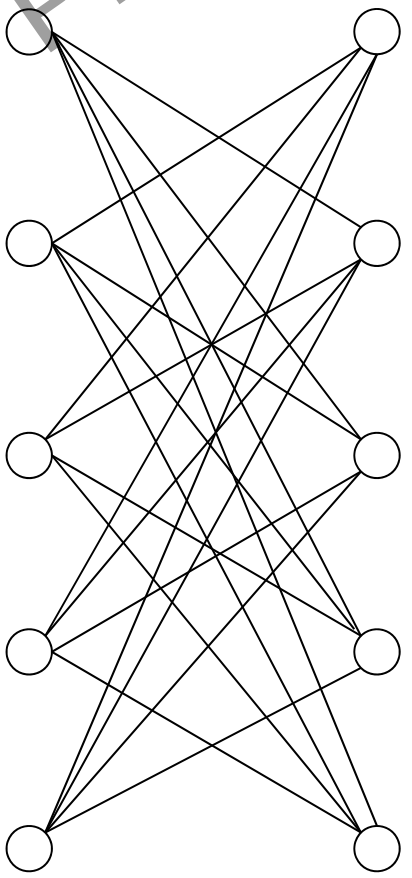
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum



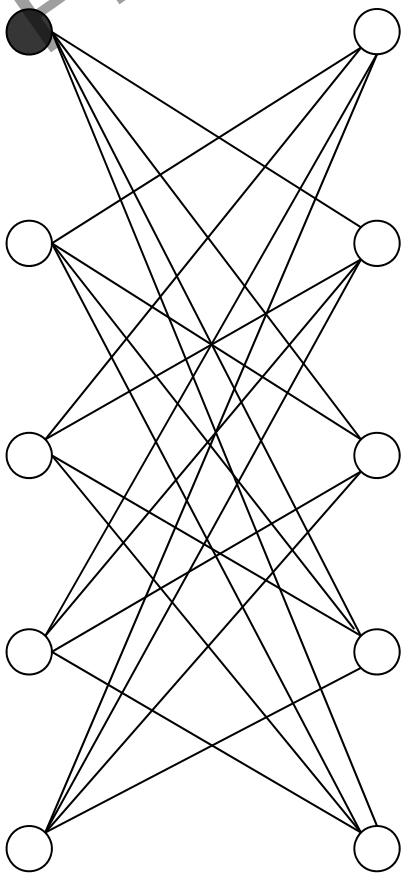
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs



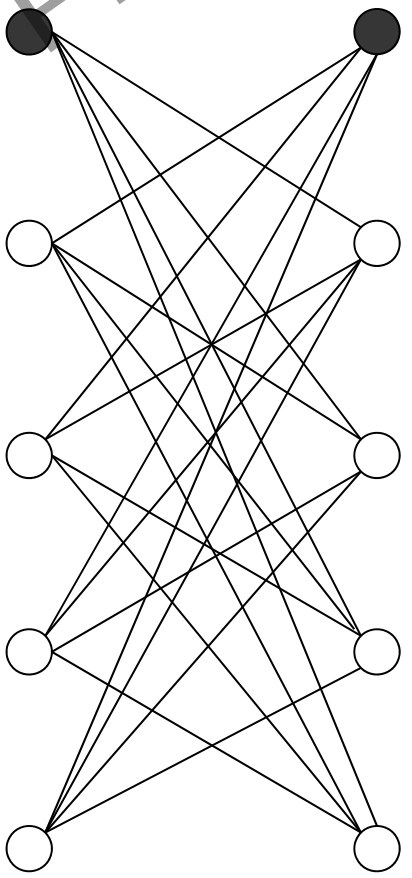
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs



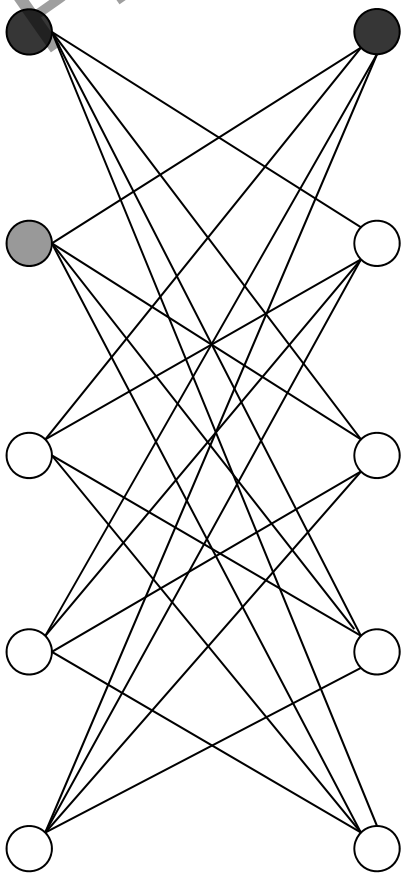
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs



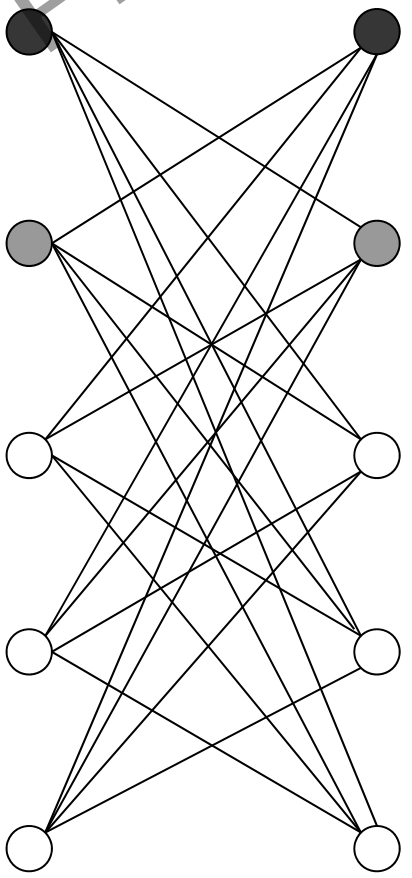
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs



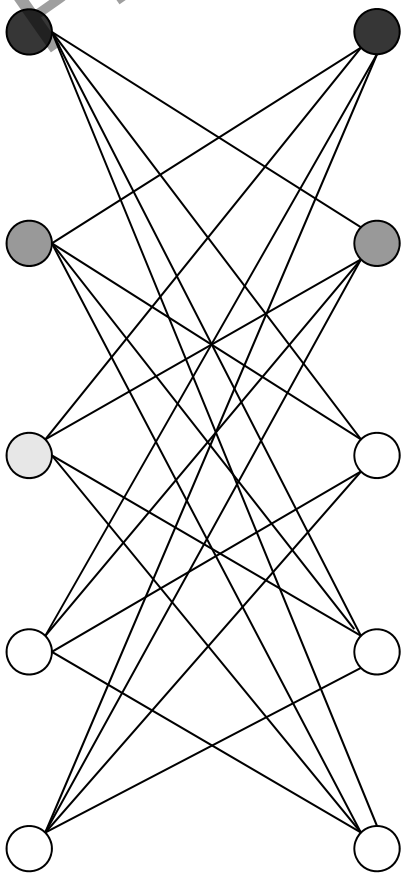
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs



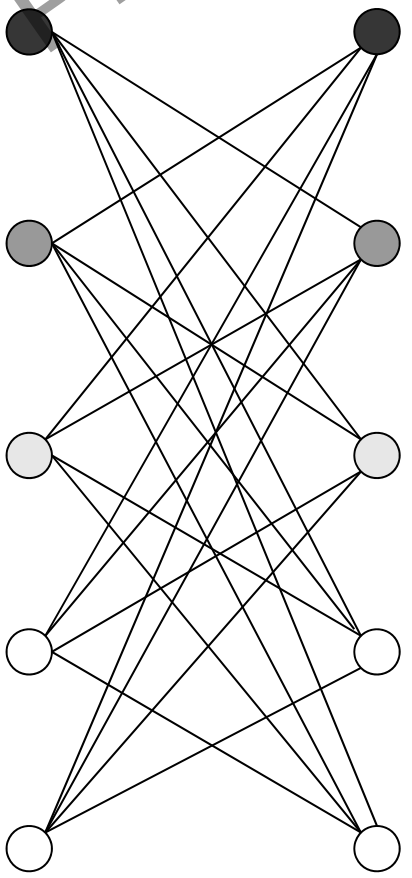
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs



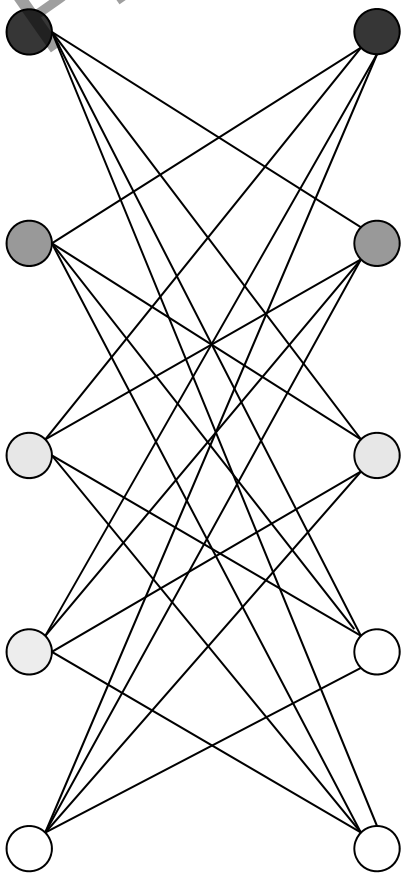
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs



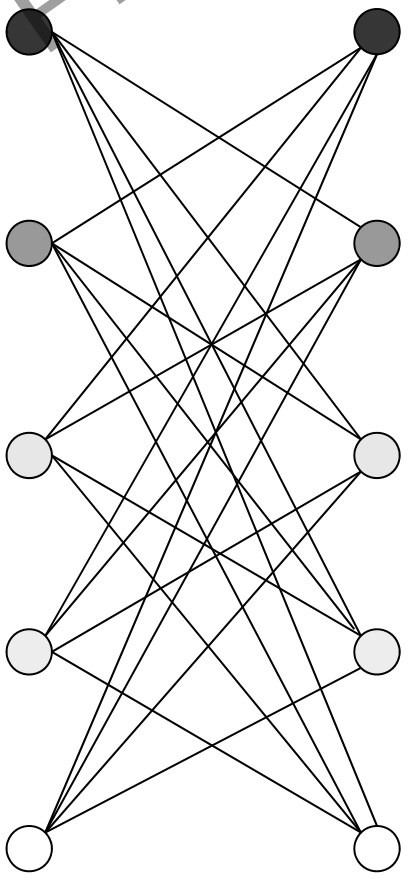
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs



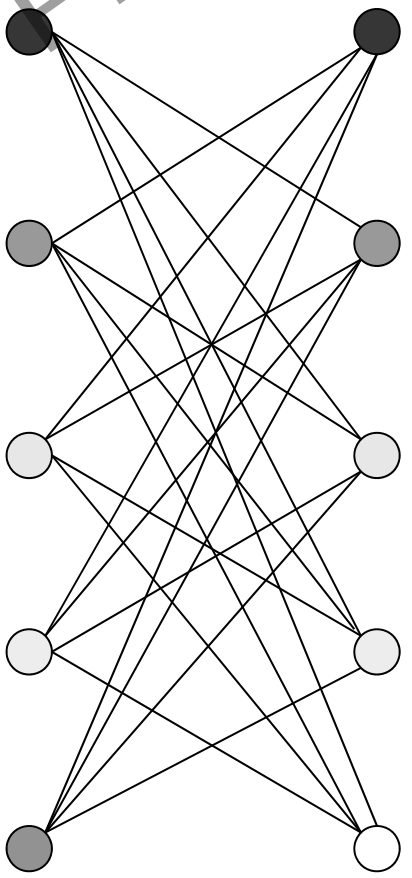
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs



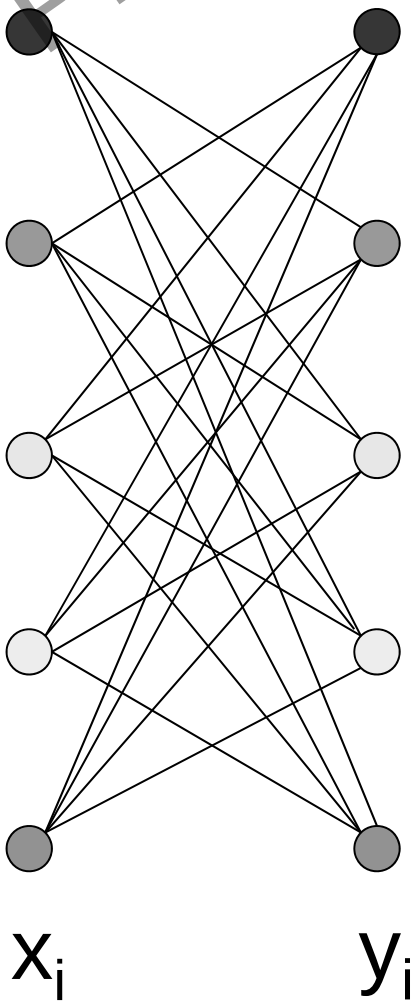
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs



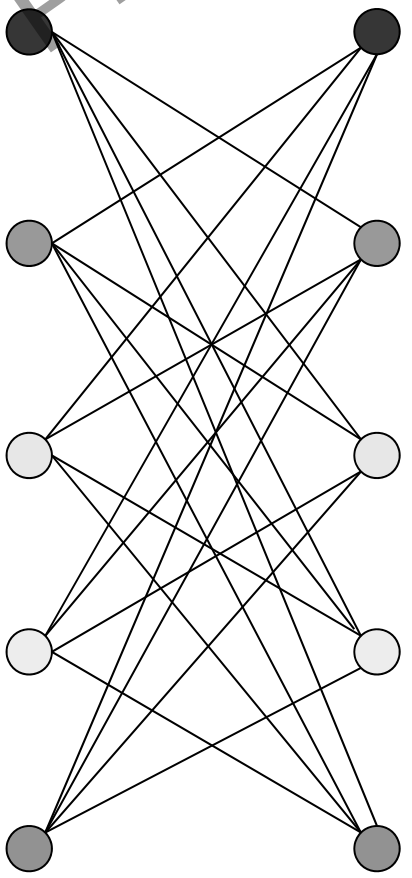
- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs



- Cette heuristique peut être très mauvaise :

- Soit G un graphe biparti de sommets $x_1,...,x_n$ et $y_1,...,y_n$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.
- Avec l'ordre $x_1,...,x_n,y_1,...,y_n$ l'heuristique trouve l'optimum
- Avec l'ordre $x_1,y_1,...,x_n,y_n$ elle utilise n/2 couleurs
- Soit une PRP d'au moins n/4.



- **Cette heuristique a cependant 2 propriétés intéressantes :**

- **Cette heuristique a cependant 2 propriétés intéressantes :**

  Soit d_{max} le degré maximal d'un sommet

- **Elle utilise au plus $d_{max}+1$ couleurs.**

- **Cette heuristique a cependant 2 propriétés intéressantes :**

  Soit d_{max} le degré maximal d'un sommet

- **Elle utilise au plus $d_{max}+1$ couleurs.**

  En effet il faut une couleur pour le premier sommet, puis chaque sommet aura au plus d_{max} voisins.

- **Cette heuristique a cependant 2 propriétés intéressantes :**

  Soit d_{max} le degré maximal d'un sommet

- **Elle utilise au plus $d_{max}+1$ couleurs.**

  En effet il faut une couleur pour le premier sommet, puis chaque sommet aura au plus d_{max} voisins.

- **Il existe un ordre sur les sommets pour lequel elle trouve l'optimum.**

- **Cette heuristique a cependant 2 propriétés intéressantes :**

  Soit d_{max} le degré maximal d'un sommet

- **Elle utilise au plus $d_{max}+1$ couleurs.**

  En effet il faut une couleur pour le premier sommet, puis chaque sommet aura au plus d_{max} voisins.

- **Il existe un ordre sur les sommets pour lequel elle trouve l'optimum.**

  En effet soit une coloration optimale. Ordonner les sommets en regroupant ceux de même couleur.

- **Cette heuristique a cependant 2 propriétés intéressantes :**

  Soit d_{max} le degré maximal d'un sommet

- **Elle utilise au plus $d_{max}+1$ couleurs.**

  En effet il faut une couleur pour le premier sommet, puis chaque sommet aura au plus d_{max} voisins.

- **Il existe un ordre sur les sommets pour lequel elle trouve l'optimum.**

  En effet soit une coloration optimale. Ordonner les sommets en regroupant ceux de même couleur.

  Cependant il y a n! ordre possibles...

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$ et ordonnons les sommets par $S_1,S_2,...,S_n$.

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$ et ordonnons les sommets par $S_1,S_2,...,S_n$. Notons d_i le degré de S_i.

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$ et ordonnons les sommets par $S_1,S_2,...,S_n$. Notons d_i le degré de S_i.
- S_i admet une couleur $\leq i$

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$ et ordonnons les sommets par $S_1,S_2,...,S_n$. Notons d_i le degré de S_i.
- S_i admet une couleur $\leq i$ et $\leq d_i+1$.

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$ et ordonnons les sommets par $S_1,S_2,...,S_n$. Notons d_i le degré de S_i.
- S_i admet une couleur $\leq i$ et $\leq d_i+1$. Donc au plus min $\{i,d_i+1\}$.

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$ et ordonnons les sommets par $S_1,S_2,...,S_n$. Notons d_i le degré de S_i.
- S_i admet une couleur $\leq i$ et $\leq d_i+1$. Donc au plus min $\{i,d_i+1\}$.
- Il faut donc au plus $B_2=max_i$ $\{min\ \{i,d_i+1\}\}$ couleurs.

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$ et ordonnons les sommets par $S_1,S_2,...,S_n$. Notons d_i le degré de S_i.
- S_i admet une couleur $\leq i$ et $\leq d_i+1$. Donc au plus min $\{i,d_i+1\}$.
- Il faut donc au plus $B_2=\max_i \{\min \{i,d_i+1\}\}$ couleurs.
- B_2 dépend de l'ordre considéré.

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$ et ordonnons les sommets par $S_1,S_2,...,S_n$. Notons d_i le degré de S_i.
- S_i admet une couleur $\leq i$ et $\leq d_i+1$. Donc au plus $\min \{i,d_i+1\}$.
- Il faut donc au plus $B_2=\max_i \{\min \{i,d_i+1\}\}$ couleurs.
- B_2 dépend de l'ordre considéré.
- $B_2 \leq B_1$

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$ et ordonnons les sommets par $S_1,S_2,...,S_n$. Notons d_i le degré de S_i.
- S_i admet une couleur $\leq i$ et $\leq d_i+1$. Donc au plus min $\{i,d_i+1\}$.
- Il faut donc au plus $B_2=\max_i \{\min \{i,d_i+1\}\}$ couleurs.
- B_2 dépend de l'ordre considéré.
- $B_2 \leq B_1$, et $B_2=B_1$ si le sommet de plus grand degré d_{max} est coloré en dernier.

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$ et ordonnons les sommets par $S_1,S_2,...,S_n$. Notons d_i le degré de S_i.
- S_i admet une couleur $\leq i$ et $\leq d_i+1$. Donc au plus min $\{i,d_i+1\}$.
- Il faut donc au plus $B_2=\max_i \{\min \{i,d_i+1\}\}$ couleurs.
- B_2 dépend de l'ordre considéré.
- $B_2 \leq B_1$, et $B_2=B_1$ si le sommet de plus grand degré d_{max} est coloré en dernier.
- L'heuristique Largest First Sequential (LFS) consiste à ordonner les sommets dans l'ordre décroissant des degrés.

Construction de bons ordres pour les heuristiques séquentielles I

- Notons $B_1=d_{max}+1$ et ordonnons les sommets par $S_1,S_2,...,S_n$. Notons d_i le degré de S_i.
- S_i admet une couleur $\leq i$ et $\leq d_i+1$. Donc au plus min $\{i,d_i+1\}$.
- Il faut donc au plus $B_2=\max_i \{\min \{i,d_i+1\}\}$ couleurs.
- B_2 dépend de l'ordre considéré.
- $B_2 \leq B_1$, et $B_2=B_1$ si le sommet de plus grand degré d_{max} est coloré en dernier.
- L'heuristique Largest First Sequential (LFS) consiste à ordonner les sommets dans l'ordre décroissant des degrés.
- On montre que cet ordre minimise B_2 pour tout graphe.

Construction de bons ordres pour les heuristiques séquentielles II

- Notons D_i le degré de S_i dans le sous-graphe engendré par S_1, S_2, ..., S_i.

Construction de bons ordres pour les heuristiques séquentielles II

- Notons D_i le degré de S_i dans le sous-graphe engendré par S_1, S_2, ..., S_i.
- La couleur de S_i est au plus $1+D_i$. Il faut donc au plus $B_3=1+\max_i\{D_i\}$ couleurs. (B_3 dépend de l'ordre.)

Construction de bons ordres pour les heuristiques séquentielles II

- Notons D_i le degré de S_i dans le sous-graphe engendré par $S_1, S_2, ..., S_i$.
- La couleur de S_i est au plus $1+D_i$. Il faut donc au plus $B_3=1+\max_i\{D_i\}$ couleurs. (B_3 dépend de l'ordre.)
- $B_3 \leq B_2$ (car $D_i \leq d_i$ et $D_i \leq i-1$).

Construction de bons ordres pour les heuristiques séquentielles II

- Notons D_i le degré de S_i dans le sous-graphe engendré par $S_1, S_2, ..., S_i$.
- La couleur de S_i est au plus $1+D_i$. Il faut donc au plus $B_3=1+\max_i\{D_i\}$ couleurs. (B_3 dépend de l'ordre.)
- $B_3 \leq B_2$ (car $D_i \leq d_i$ et $D_i \leq i-1$).
- Heuristique Smallest Last Sequential (SLS) :
- Ordre :
  - S_n est le sommet de degré minimal.
  - Pour i = n-1, n-2,..., 1, S_i est le sommet de degré minimal dans le sous-graphe obtenu en supprimant $S_{i+1},..., S_n$ (et leurs arêtes incidentes).

Construction de bons ordres pour les heuristiques séquentielles II

- Notons D_i le degré de S_i dans le sous-graphe engendré par S_1, S_2, ..., S_i.
- La couleur de S_i est au plus $1+D_i$. Il faut donc au plus $B_3=1+\max_i\{D_i\}$ couleurs. (B_3 dépend de l'ordre.)
- $B_3 \leq B_2$ (car $D_i \leq d_i$ et $D_i \leq i-1$).
- Heuristique Smallest Last Sequential (SLS) :
- Ordre :
  - S_n est le sommet de degré minimal.
  - Pour i = n-1, n-2,..., 1, S_i est le sommet de degré minimal dans le sous-graphe obtenu en supprimant S_{i+1},..., S_n (et leurs arêtes incidentes).
- On montre que SLS minimise B_3 sur tout graphe.

Construction de bons ordres pour les heuristiques séquentielles II

- Notons D_i le degré de S_i dans le sous-graphe engendré par $S_1, S_2, ..., S_i$.
- La couleur de S_i est au plus $1+D_i$. Il faut donc au plus $B_3=1+\max_i\{D_i\}$ couleurs. (B_3 dépend de l'ordre.)
- $B_3 \leq B_2$ (car $D_i \leq d_i$ et $D_i \leq i-1$).
- Heuristique Smallest Last Sequential (SLS) :
- Ordre :
  - S_n est le sommet de degré minimal.
  - Pour i = n-1, n-2,..., 1, S_i est le sommet de degré minimal dans le sous-graphe obtenu en supprimant $S_{i+1},..., S_n$ (et leurs arêtes incidentes).
- On montre que SLS minimise B_3 sur tout graphe.
- SLS colore tout graphe planaire en au plus 6 couleurs (optimum ≤ 4)

Construction de bons ordres pour les heuristiques séquentielles III

- L'heuristique Degré de Saturation (DS) construit un ordre dynamiquement :

Construction de bons ordres pour les heuristiques séquentielles III

- L'heuristique Degré de Saturation (DS) construit un ordre dynamiquement :
- A l'itération i le degré de saturation d'un sommet S, $DS_i(S)$ est le nombre de couleurs déja utilisées par les voisins de S.

Construction de bons ordres pour les heuristiques séquentielles III

- L'heuristique Degré de Saturation (DS) construit un ordre dynamiquement :
- A l'itération i le degré de saturation d'un sommet S, $DS_i(S)$ est le nombre de couleurs déja utilisées par les voisins de S.
- Colorer le sommet de degré maximal avec la couleur 1.

Construction de bons ordres pour les heuristiques séquentielles III

- L'heuristique Degré de Saturation (DS) construit un ordre dynamiquement :
- A l'itération i le degré de saturation d'un sommet S, $DS_i(S)$ est le nombre de couleurs déja utilisées par les voisins de S.
- Colorer le sommet de degré maximal avec la couleur 1.
- Aux étapes suivantes prendre le sommet libre de DS maximal (si plusieurs prendre parmi eux celui de degré maximal) et le colorer avec la plus petite couleur possible.

Construction de bons ordres pour les heuristiques séquentielles III

- L'heuristique Degré de Saturation (DS) construit un ordre dynamiquement :
- A l'itération i le degré de saturation d'un sommet S, $DS_i(S)$ est le nombre de couleurs déja utilisées par les voisins de S.
- Colorer le sommet de degré maximal avec la couleur 1.
- Aux étapes suivantes prendre le sommet libre de DS maximal (si plusieurs prendre parmi eux celui de degré maximal) et le colorer avec la plus petite couleur possible.
- Le nombre de couleur utilisées est au plus :

$$B_4 = 1 + \max \{DS(S_i), 1 \leq i \leq n\}.$$

Construction de bons ordres pour les heuristiques séquentielles III

- L'heuristique Degré de Saturation (DS) construit un ordre dynamiquement :
- A l'itération i le degré de saturation d'un sommet S, $DS_i(S)$ est le nombre de couleurs déja utilisées par les voisins de S.
- Colorer le sommet de degré maximal avec la couleur 1.
- Aux étapes suivantes prendre le sommet libre de DS maximal (si plusieurs prendre parmi eux celui de degré maximal) et le colorer avec la plus petite couleur possible.
- Le nombre de couleur utilisées est au plus :

$$B_4 = 1 + \max \{DS(S_i), 1 \leq i \leq n\}.$$

- $B_4 \leq B_3 \leq B_2 \leq B1$.

Construction de bons ordres pour les heuristiques séquentielles III

- L'heuristique Degré de Saturation (DS) construit un ordre dynamiquement :
- A l'itération i le degré de saturation d'un sommet S, $DS_i(S)$ est le nombre de couleurs déja utilisées par les voisins de S.
- Colorer le sommet de degré maximal avec la couleur 1.
- Aux étapes suivantes prendre le sommet libre de DS maximal (si plusieurs prendre parmi eux celui de degré maximal) et le colorer avec la plus petite couleur possible.
- Le nombre de couleur utilisées est au plus :

$$B_4 = 1 + \max \{DS(S_i), 1 \leq i \leq n\}.$$

- $B_4 \leq B_3 \leq B_2 \leq B1$.
- L'heuristique DS est très bonne en moyenne.

Construction de bons ordres pour les heuristiques séquentielles III

- L'heuristique Degré de Saturation (DS) construit un ordre dynamiquement :
- A l'itération i le degré de saturation d'un sommet S, $DS_i(S)$ est le nombre de couleurs déja utilisées par les voisins de S.
- Colorer le sommet de degré maximal avec la couleur 1.
- Aux étapes suivantes prendre le sommet libre de DS maximal (si plusieurs prendre parmi eux celui de degré maximal) et le colorer avec la plus petite couleur possible.
- Le nombre de couleur utilisées est au plus :

$$B_4 = 1 + \max \{DS(S_i), 1 \leq i \leq n\}.$$

- $B_4 \leq B_3 \leq B_2 \leq B1$.
- L'heuristique DS est très bonne en moyenne.
- Toutes ces Heuristiques sont de complexité linéaire et de PRP d'ordre O(n). La meilleure connue (Johnson) a PRP n/log n.

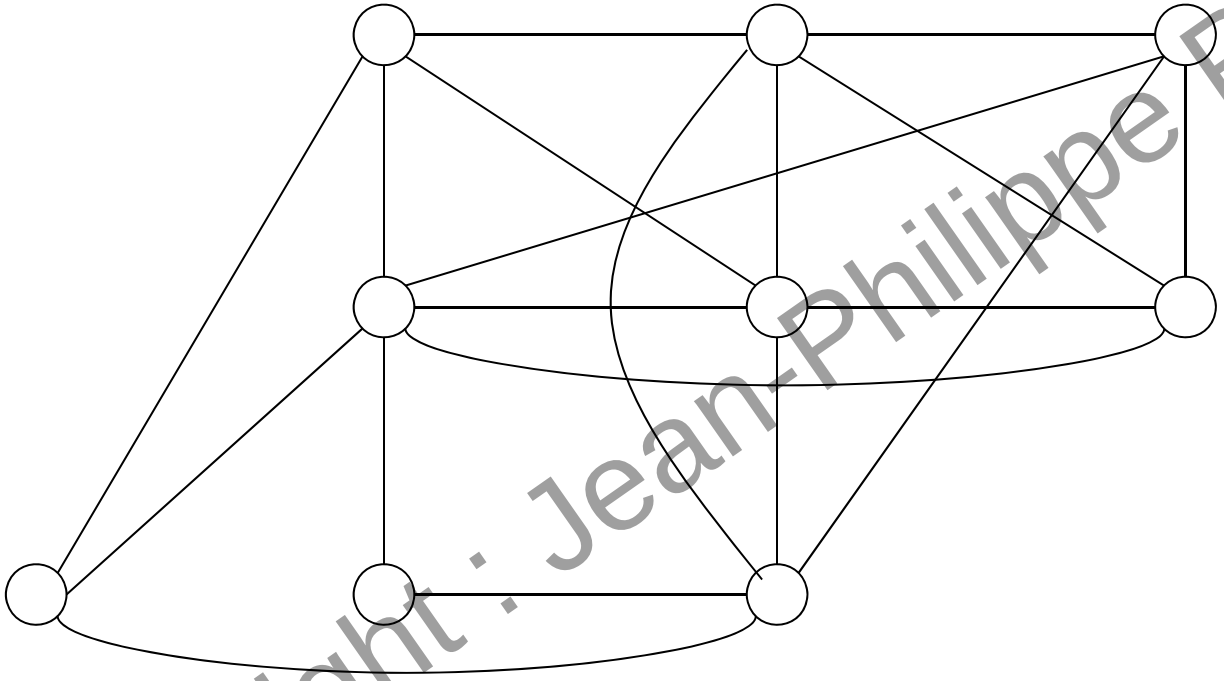
Construction de bons ordres pour les heuristiques séquentielles III

- L'heuristique Degré de Saturation (DS) construit un ordre dynamiquement :
- A l'itération i le degré de saturation d'un sommet S, $DS_i(S)$ est le nombre de couleurs déja utilisées par les voisins de S.
- Colorer le sommet de degré maximal avec la couleur 1.
- Aux étapes suivantes prendre le sommet libre de DS maximal (si plusieurs prendre parmi eux celui de degré maximal) et le colorer avec la plus petite couleur possible.
- Le nombre de couleur utilisées est au plus :

$$B_4 = 1 + \max \{DS(S_i), 1 \leq i \leq n\}.$$

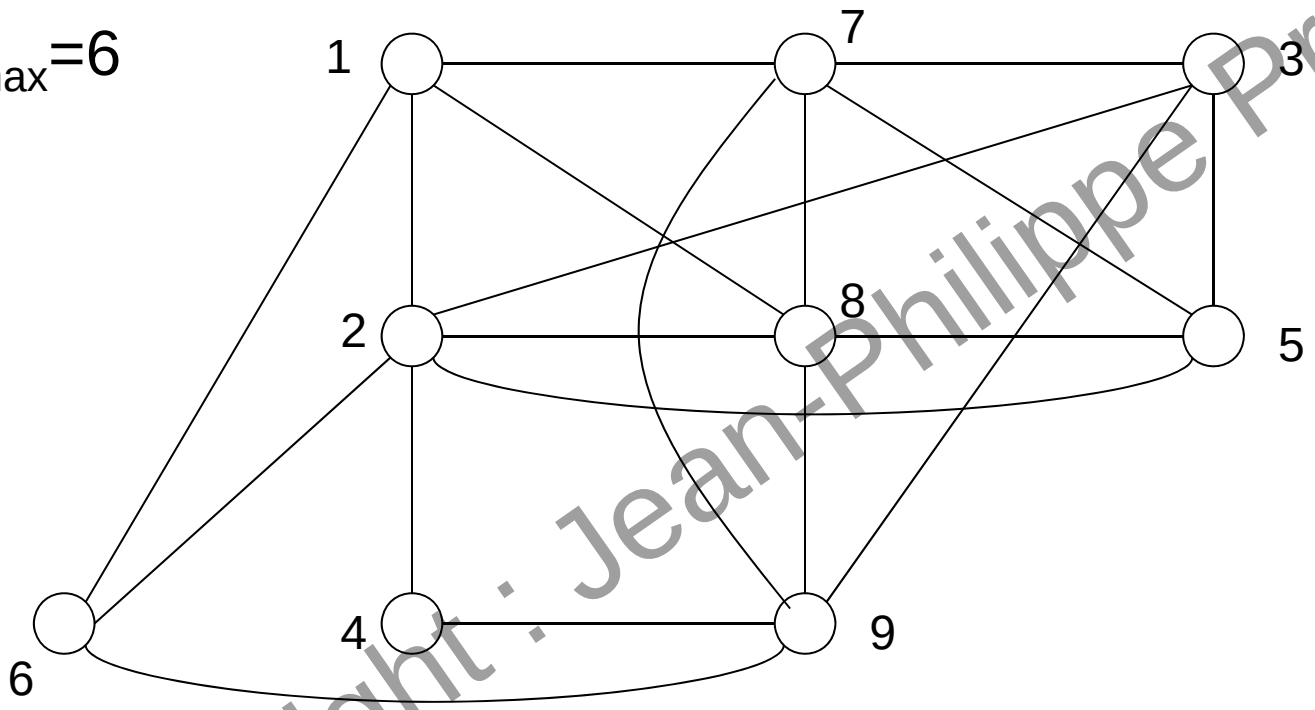
- $B_4 \leq B_3 \leq B_2 \leq B1$.
- L'heuristique DS est très bonne en moyenne.
- Toutes ces Heuristiques sont de complexité linéaire et de PRP d'ordre O(n). La meilleure connue (Johnson) a PRP n/log n.
- On ne connaît pas de bonne heuristique pour la coloration.

- Exemple :



- Exemple : FFS

d_{max}=6

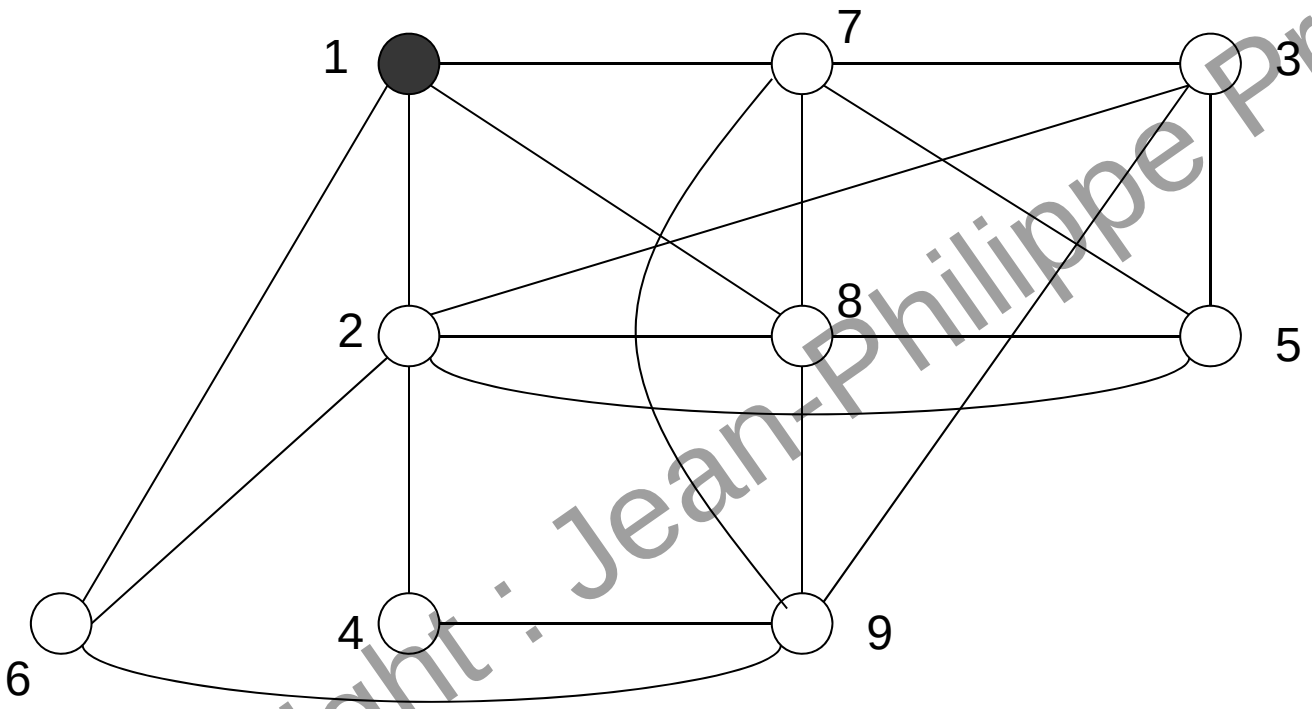


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS

1 2 3 4 5 6 7

- Exemple : FFS

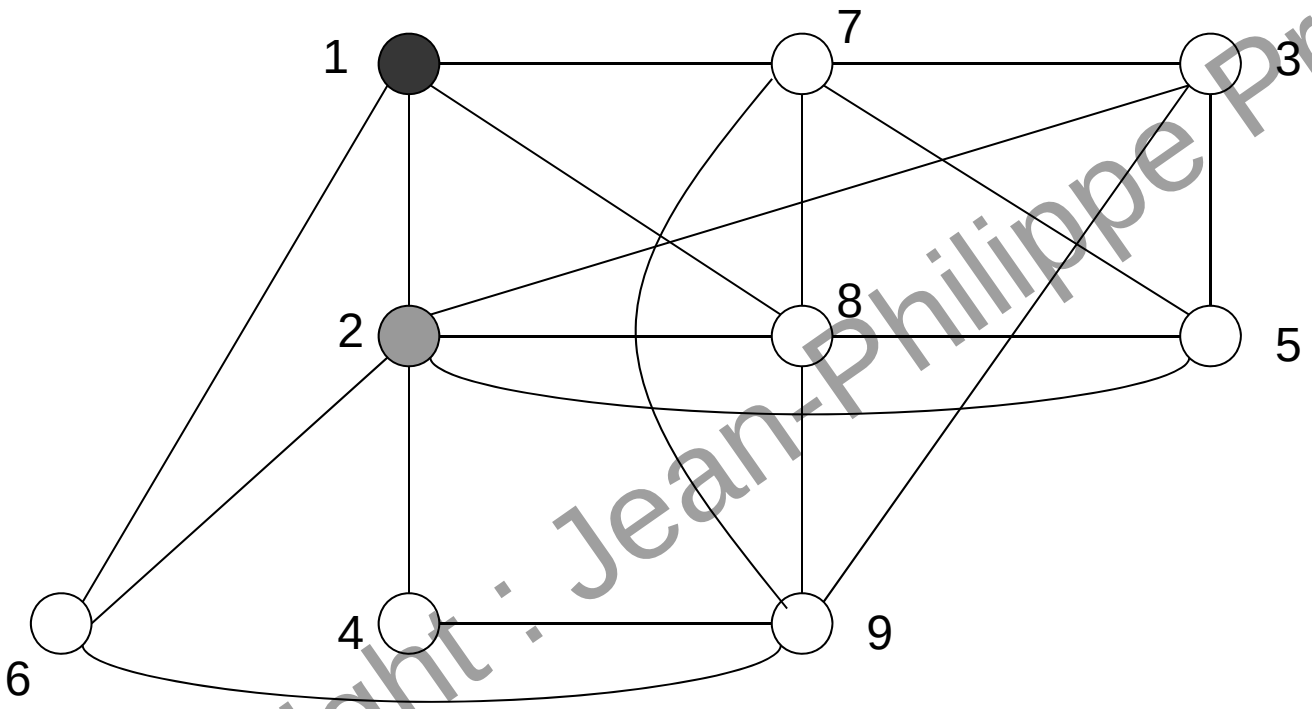


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS

1 2 3 4 5 6 7

- Exemple : FFS

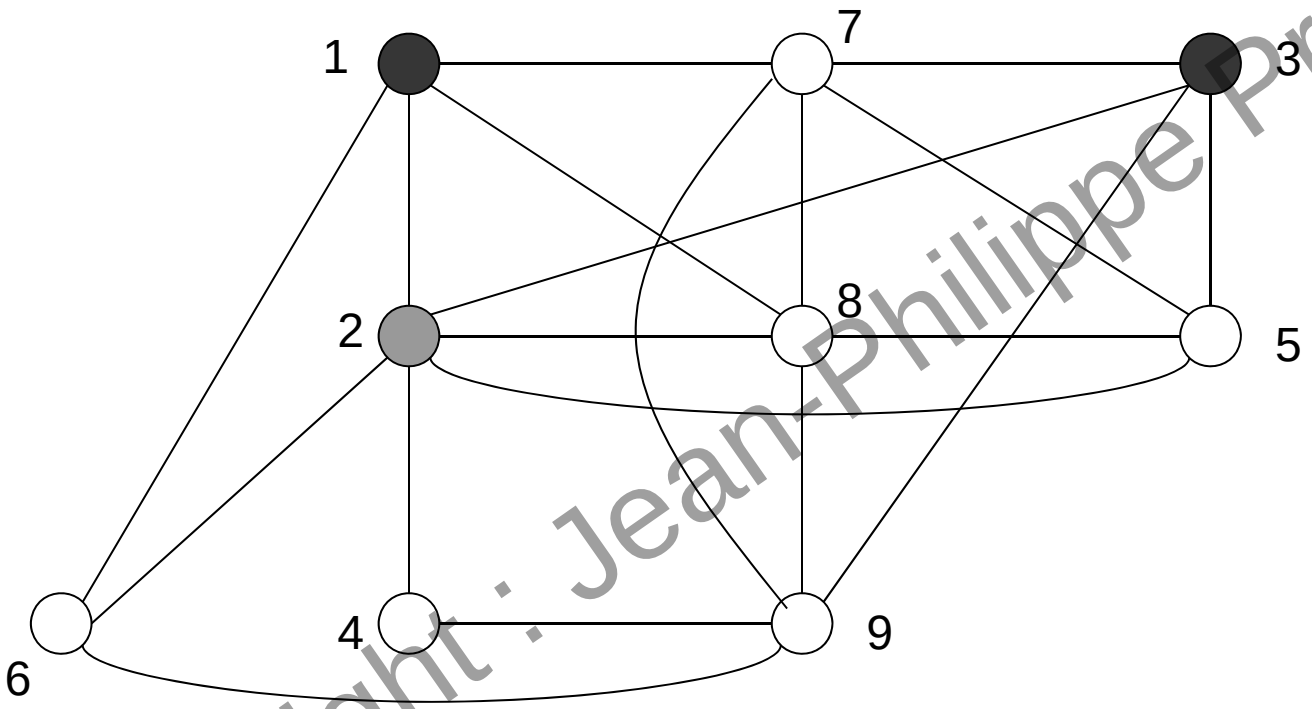


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS

1 2 3 4 5 6 7

- Exemple : FFS

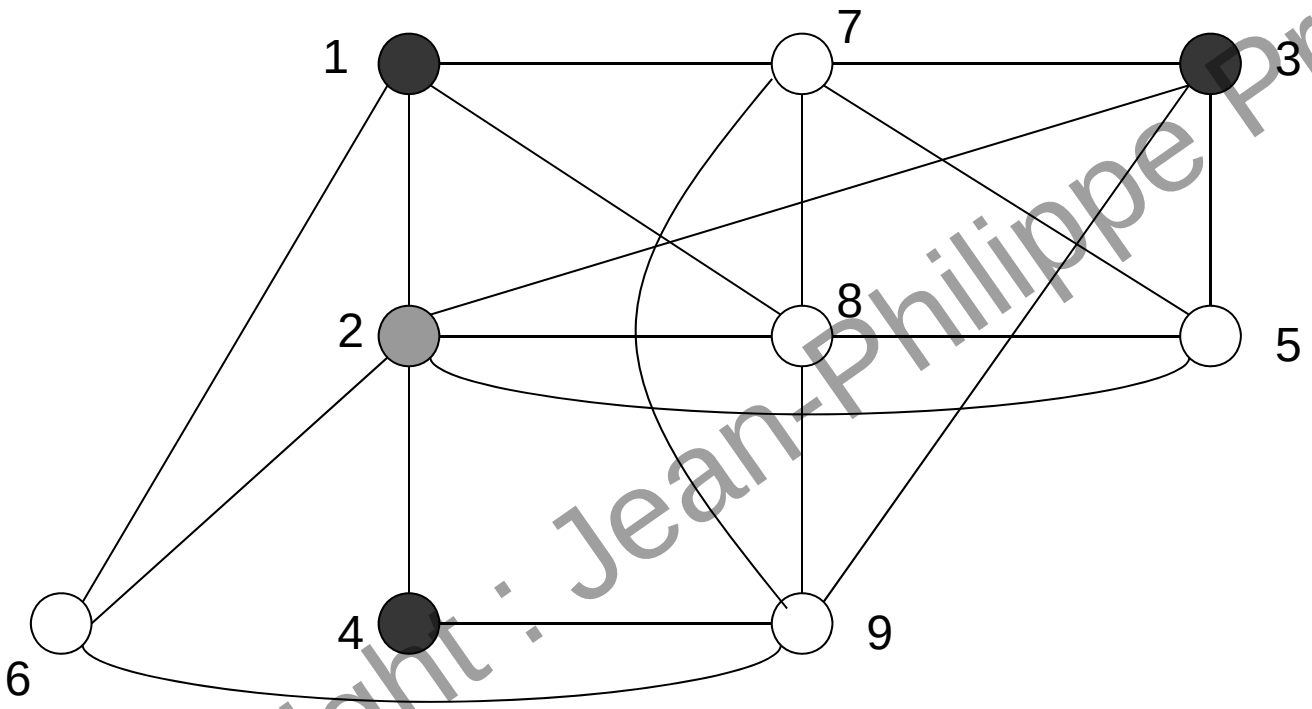


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS

1 2 3 4 5 6 7

- Exemple : FFS

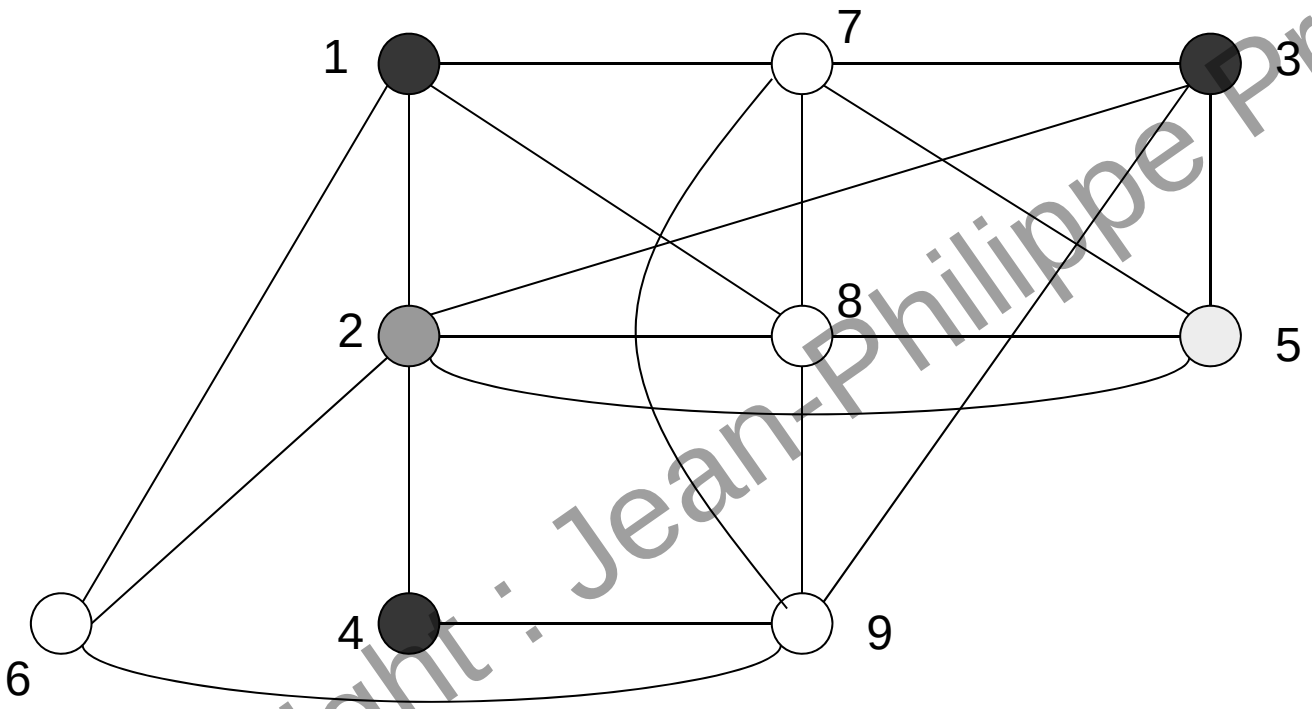


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS

1 2 3 4 5 6 7

- Exemple : FFS

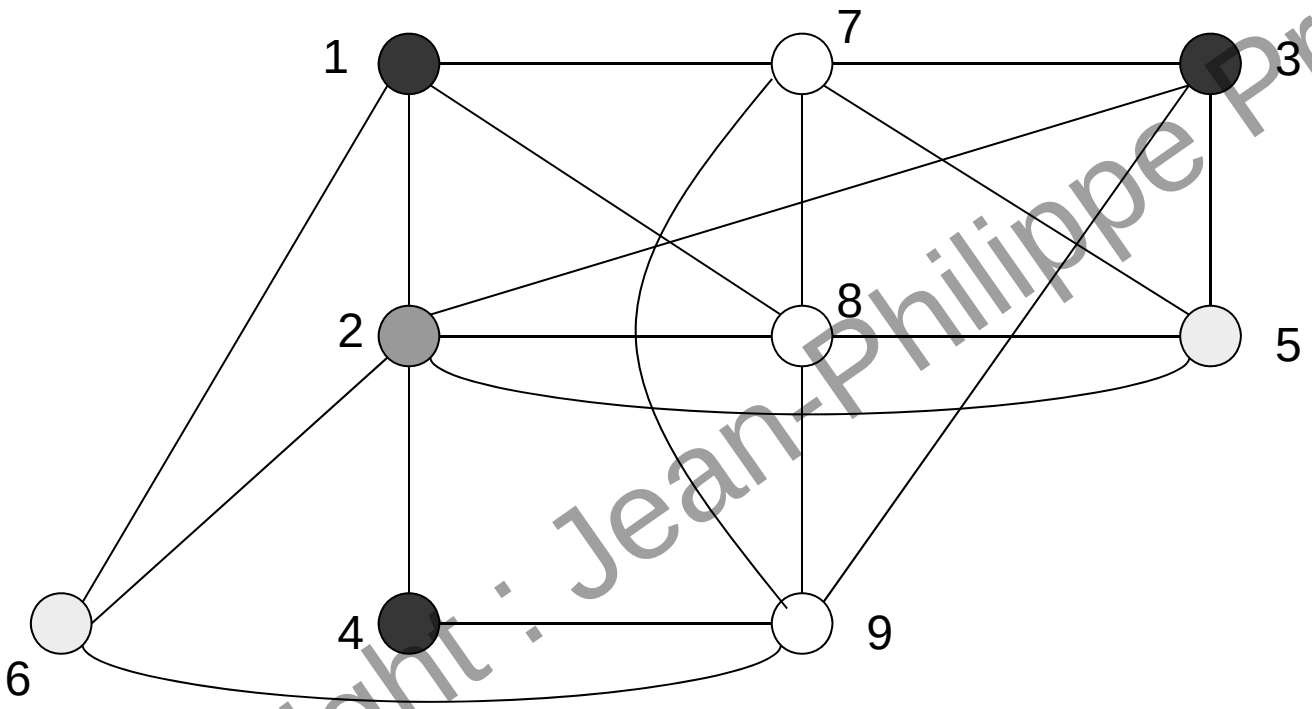


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS

1 2 3 4 5 6 7

- Exemple : FFS

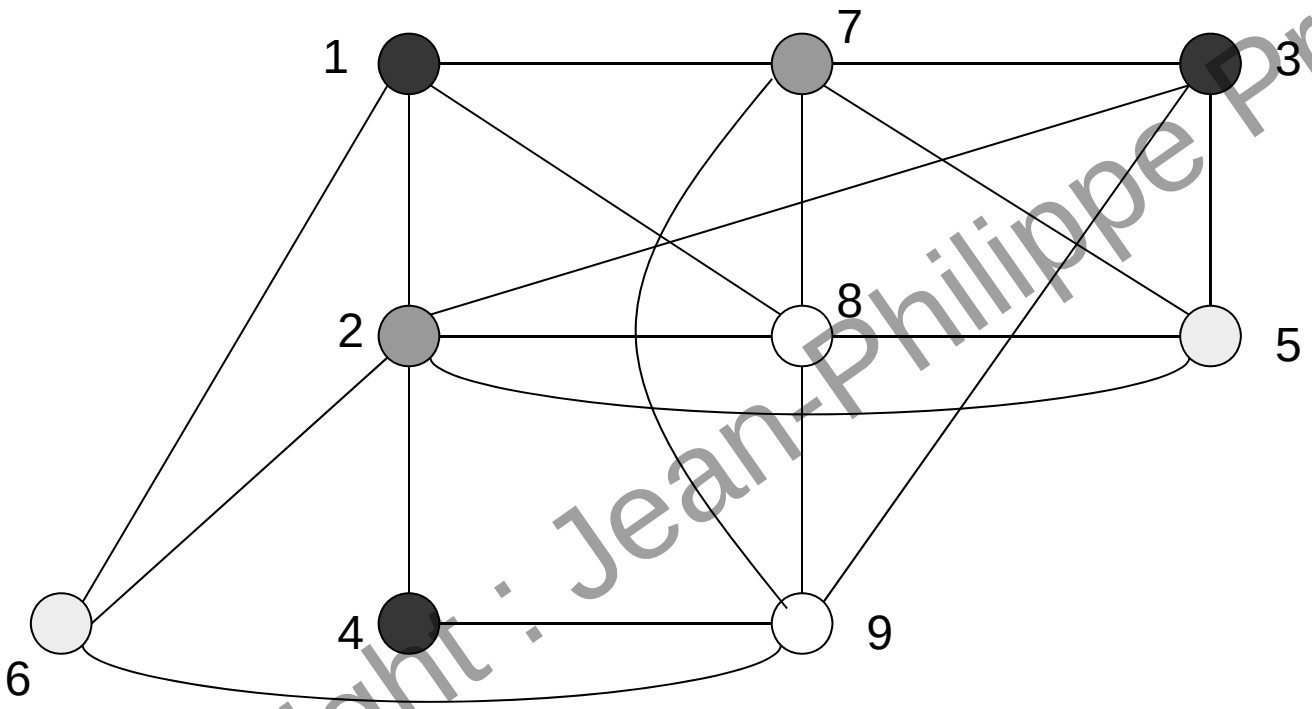


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS

1 2 3 4 5 6 7

- Exemple : FFS

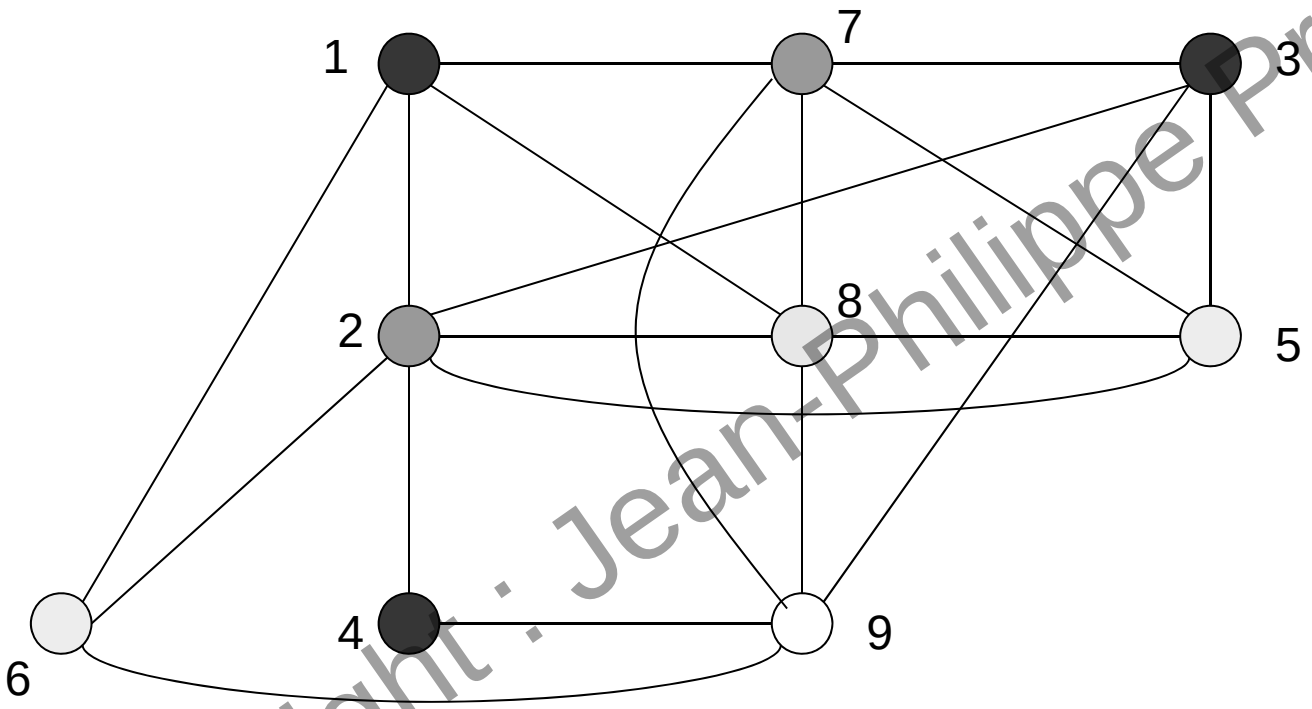


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS

1 2 3 4 5 6 7

- Exemple : FFS

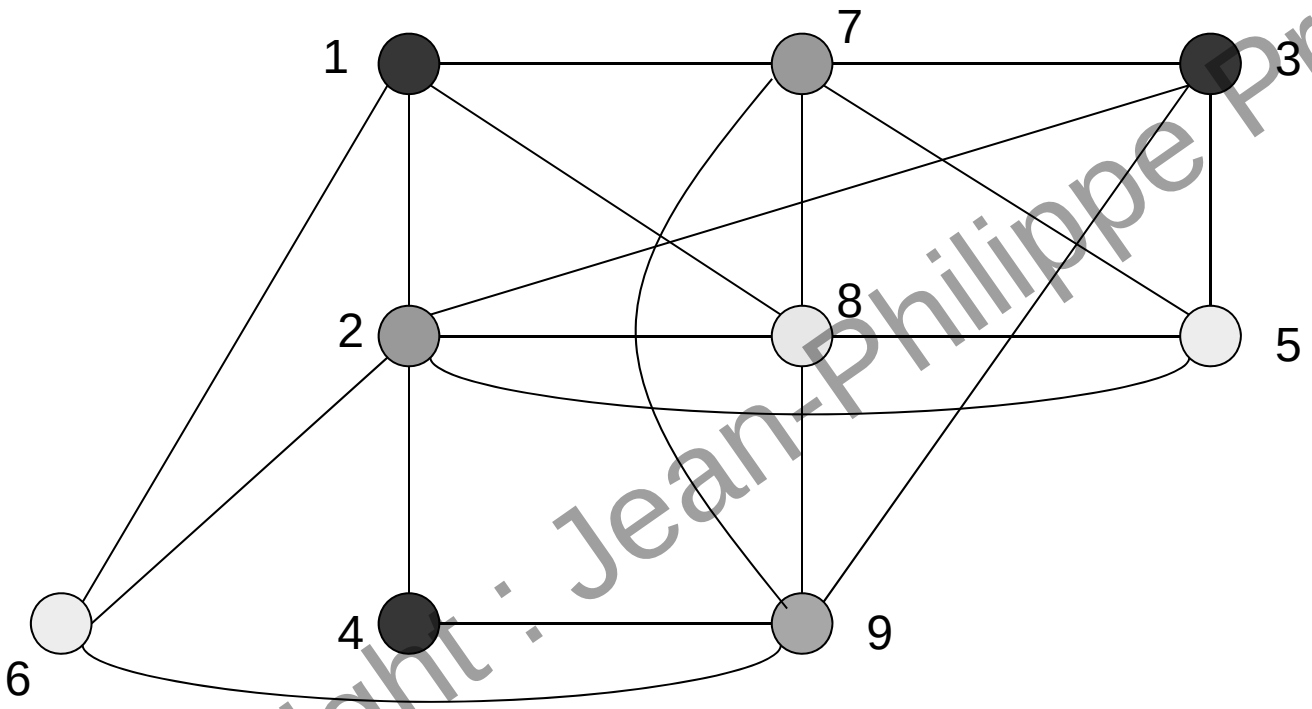


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS

1 2 3 4 5 6 7

- Exemple : FFS

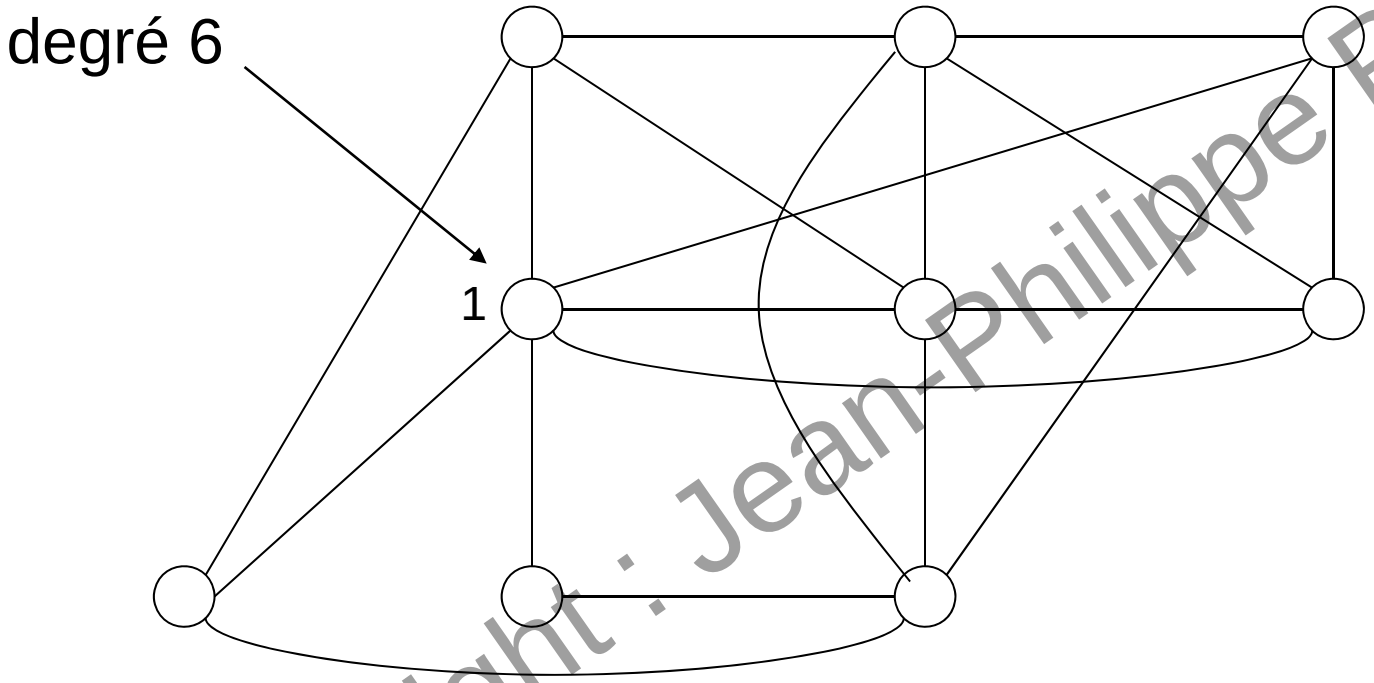


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

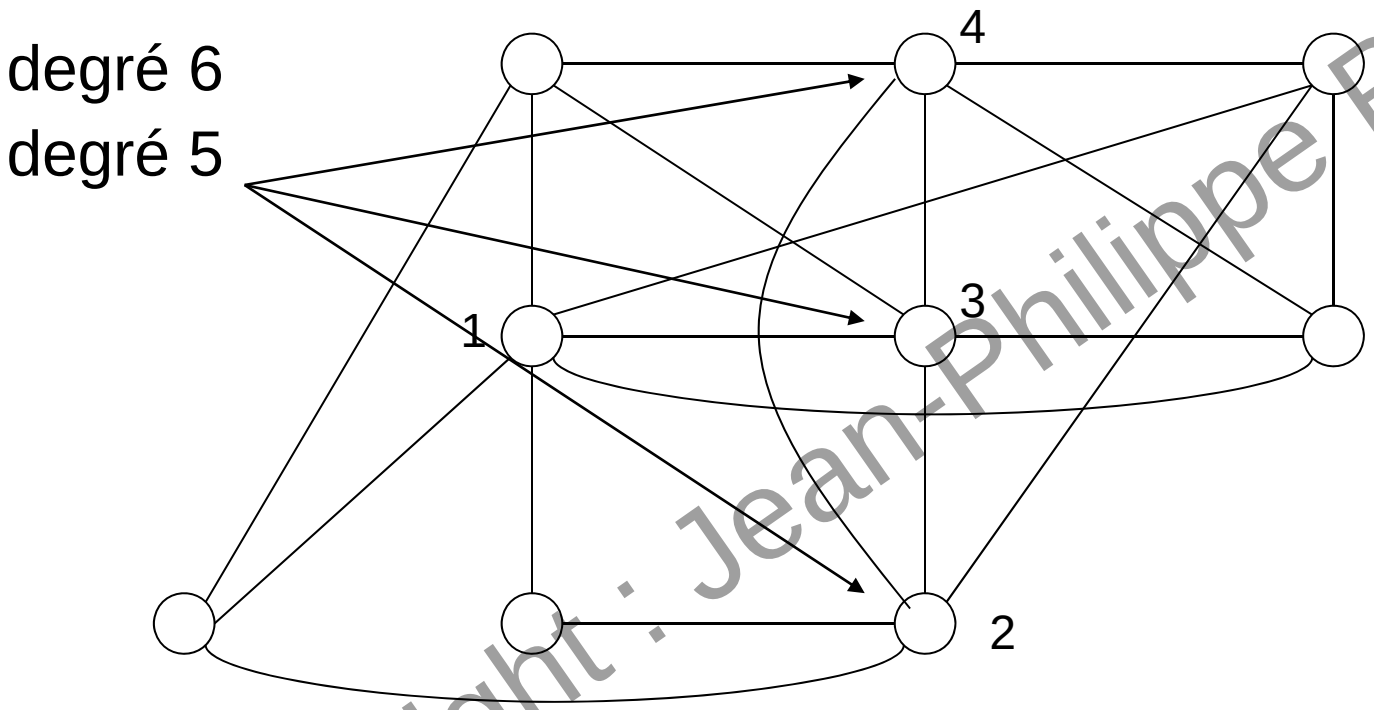


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

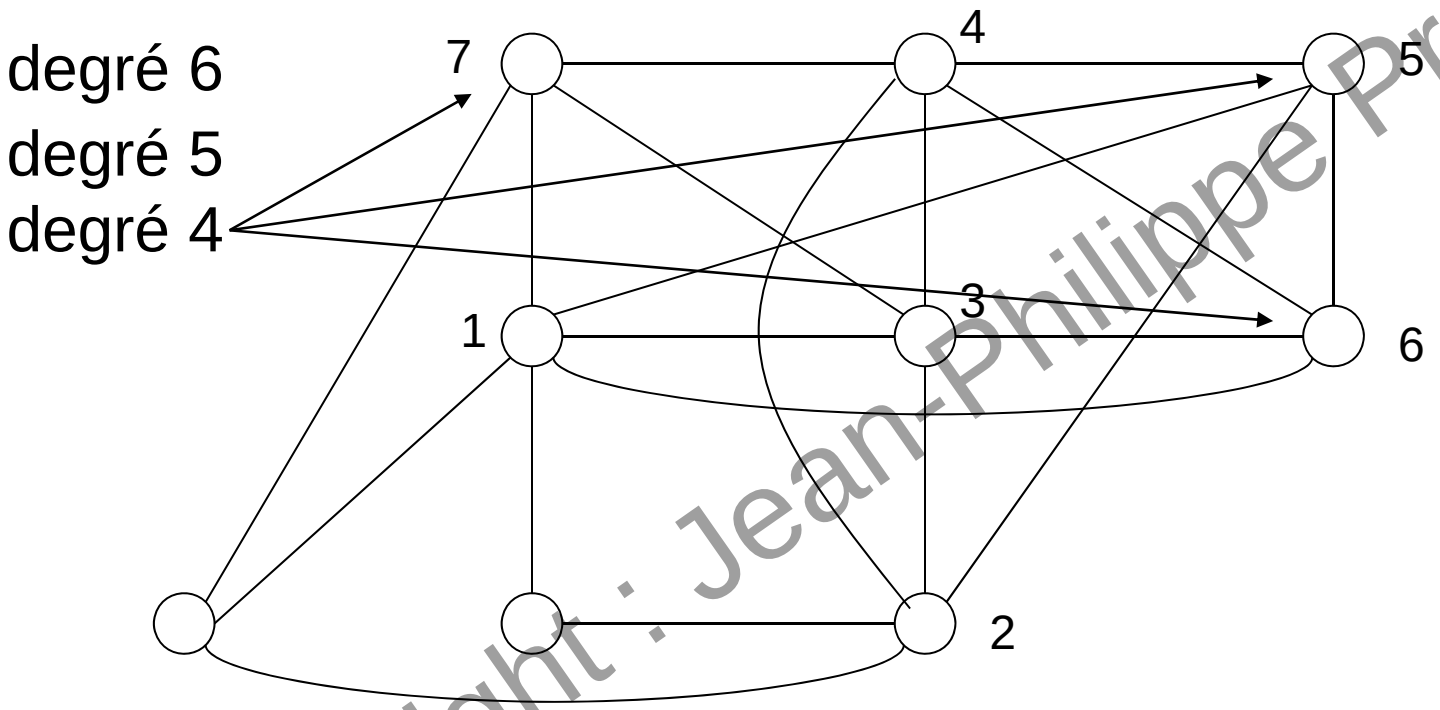


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

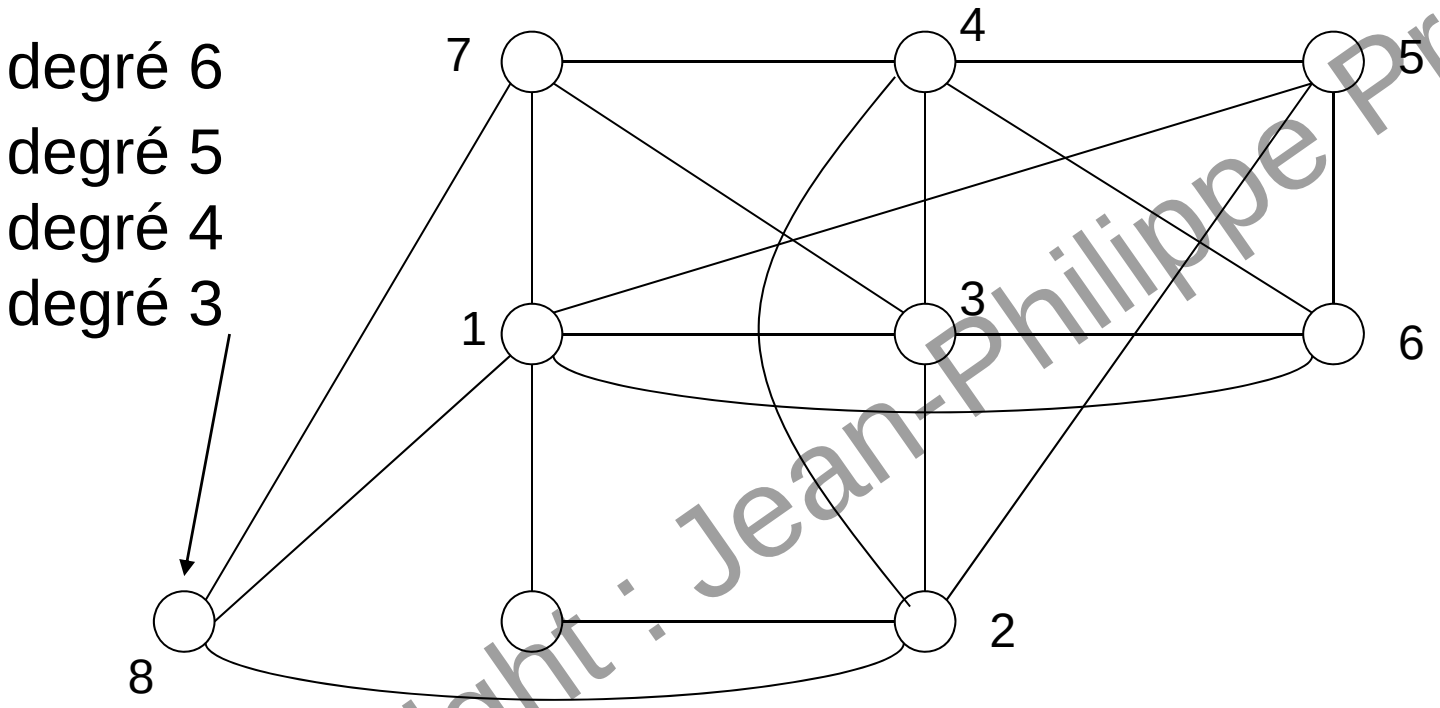


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

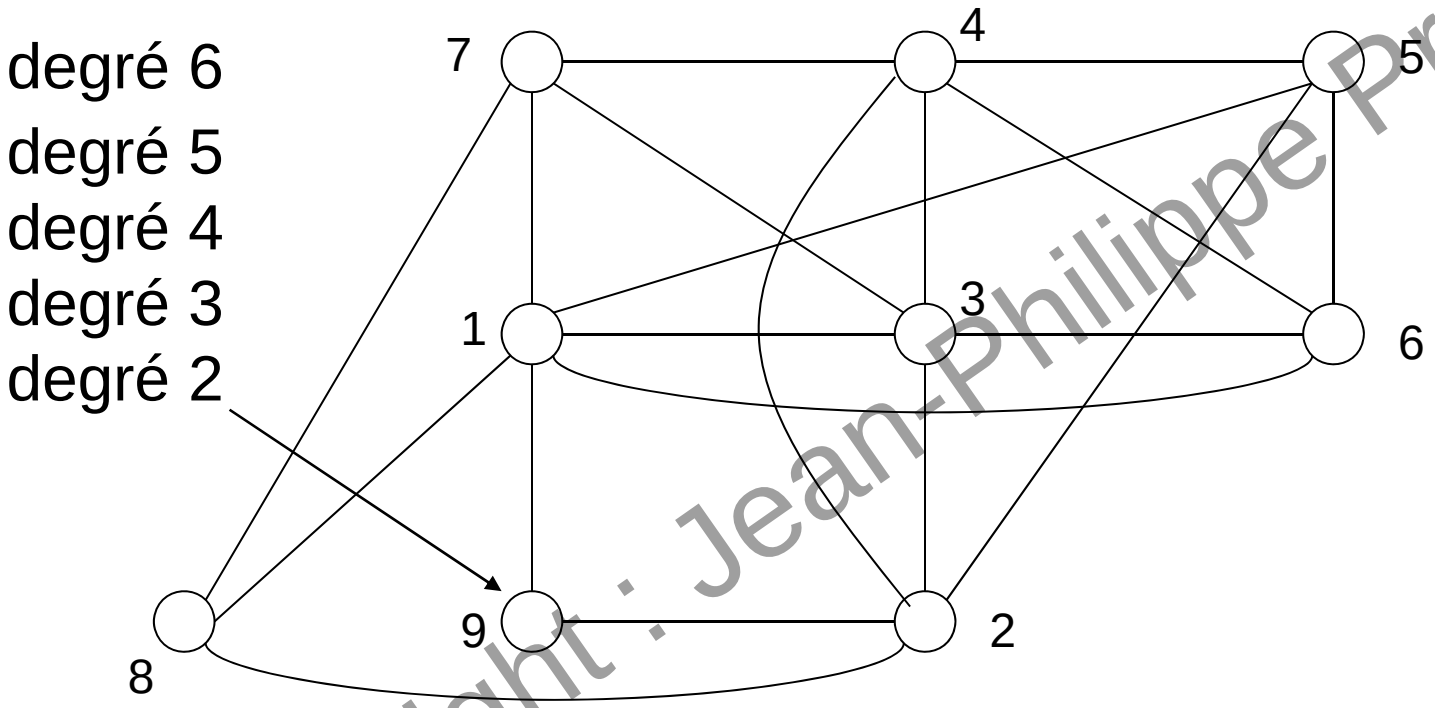


B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS



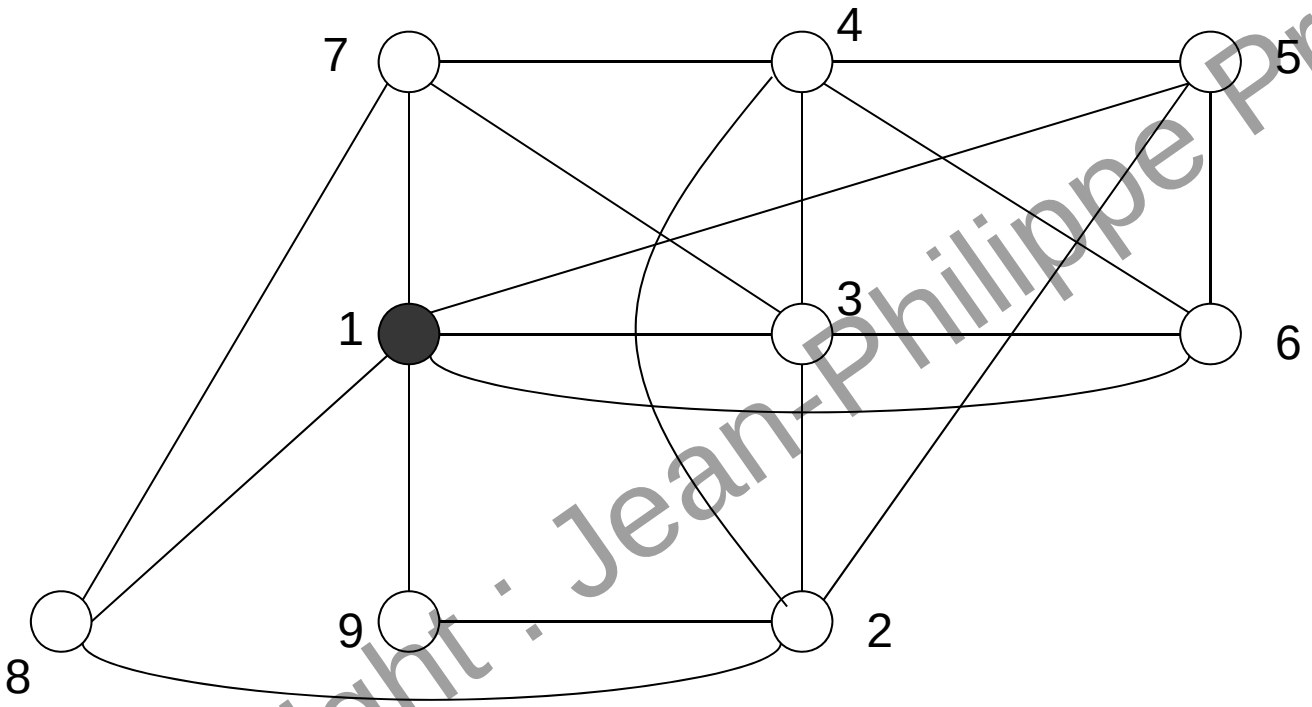
B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

Calcul de B_2 : $\min(1,1+d_1)=1$
max=1



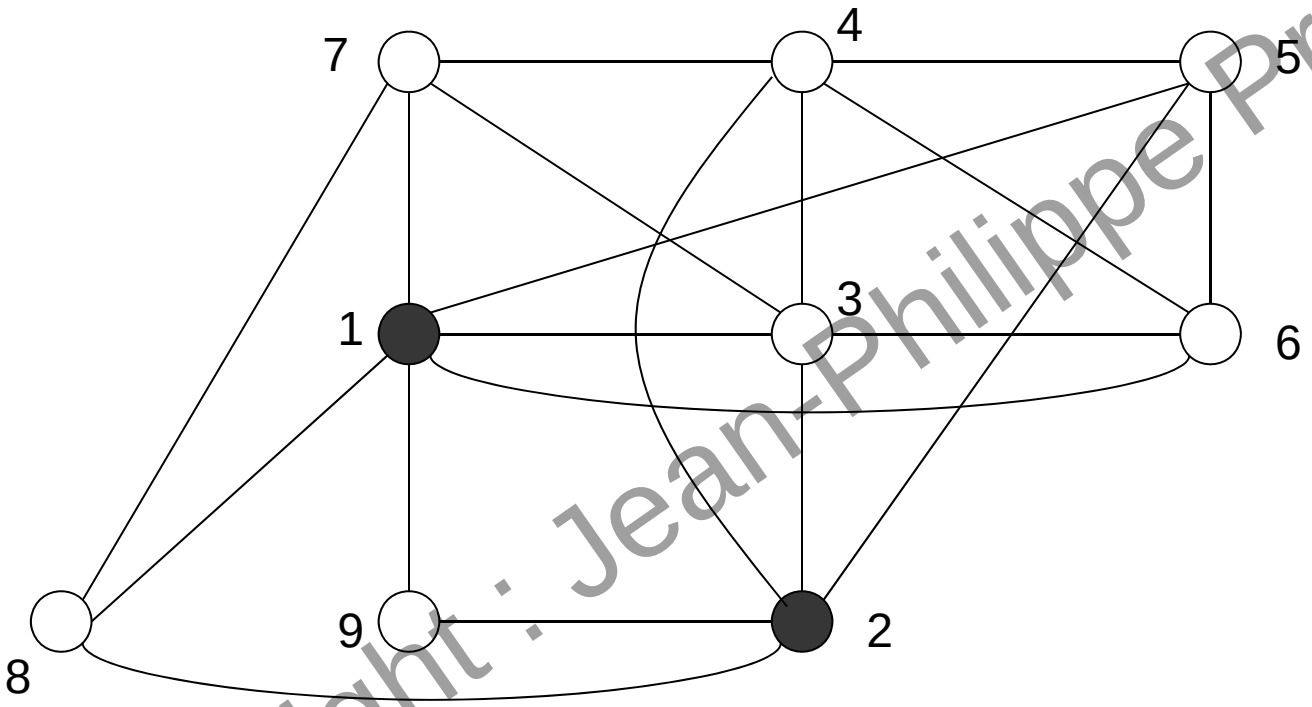
B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

Calcul de B_2 : $\min(2,1+d_2)=2$
max=2



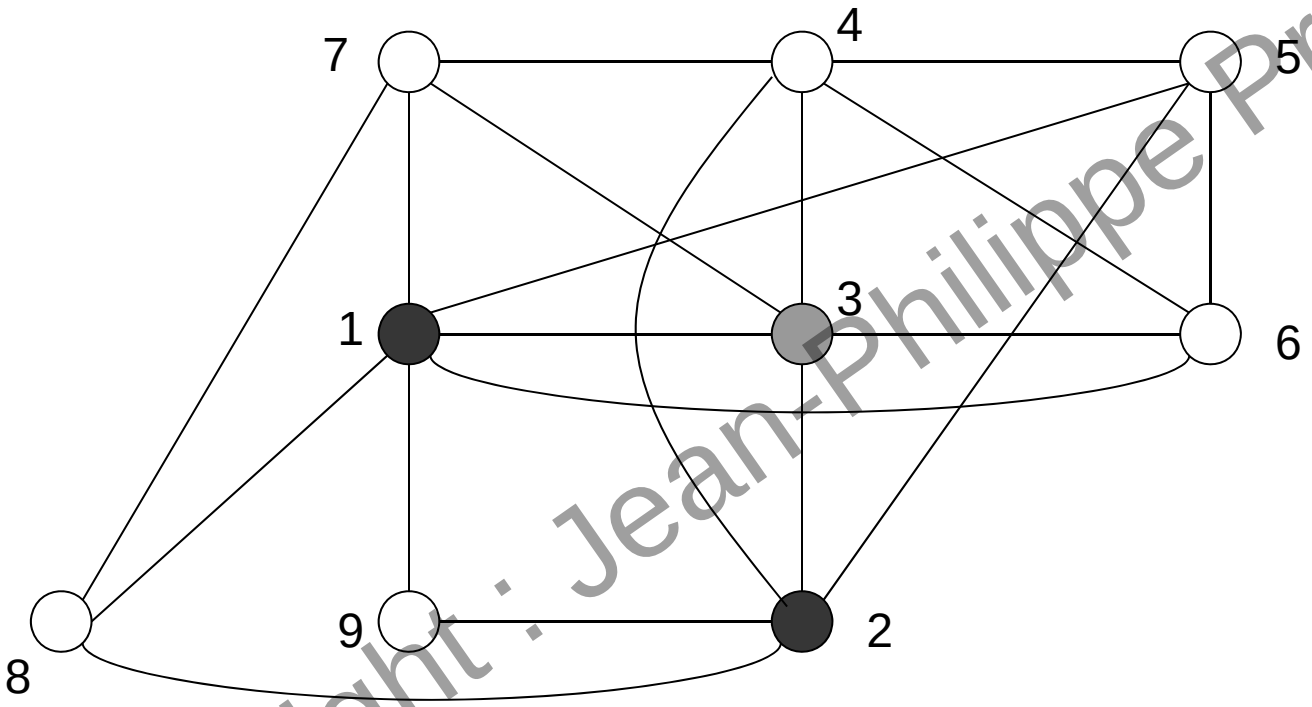
B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

Calcul de B_2 : $\min(3,1+d_3)=3$
max=3



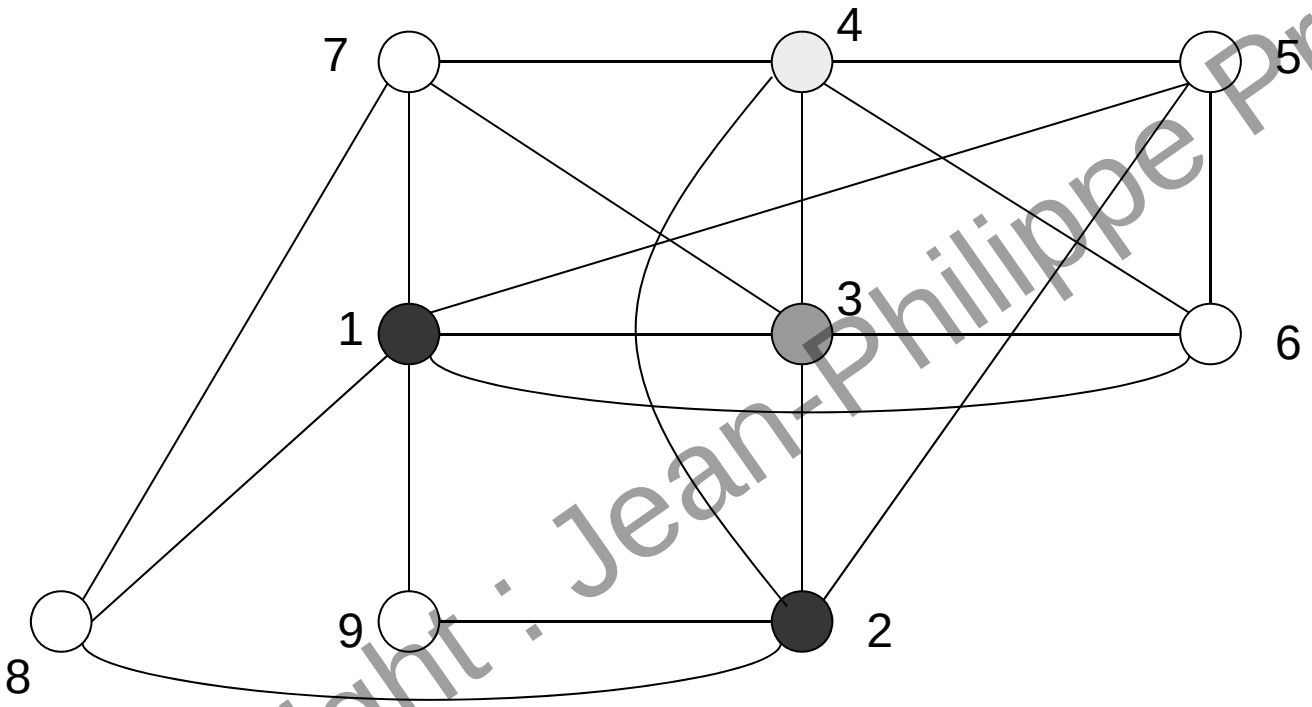
B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

Calcul de B_2 : $\min(4,1+d_4)=4$
max=4



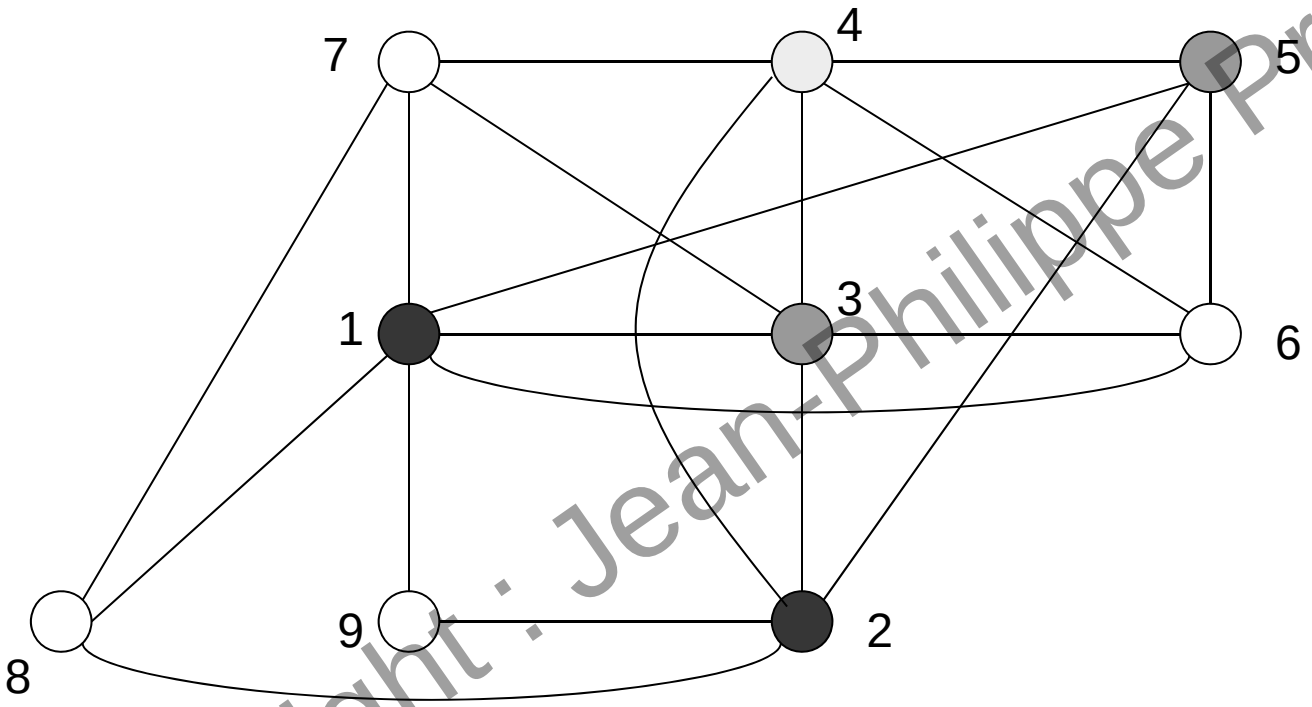
B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

Calcul de B_2 : $\min(5,1+d_5)=5$
max=5



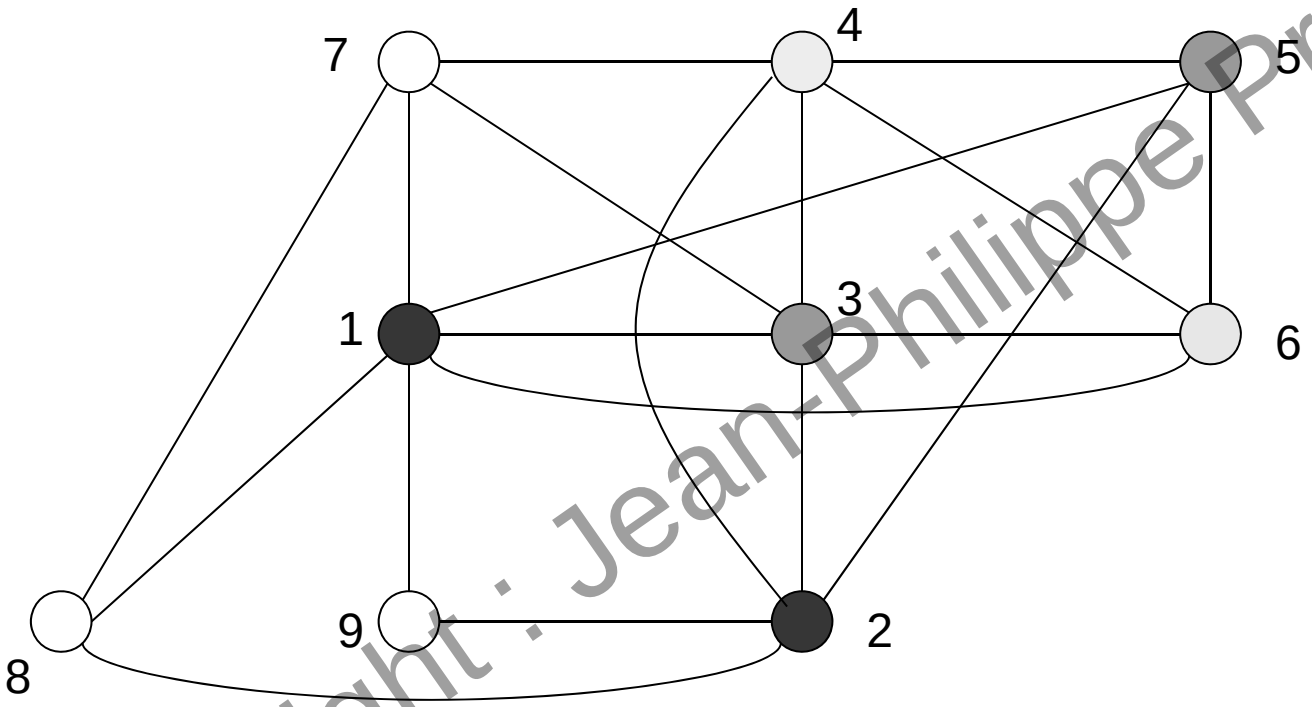
B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

Calcul de B_2 : $\min(6,1+d_6)=5$
max=5



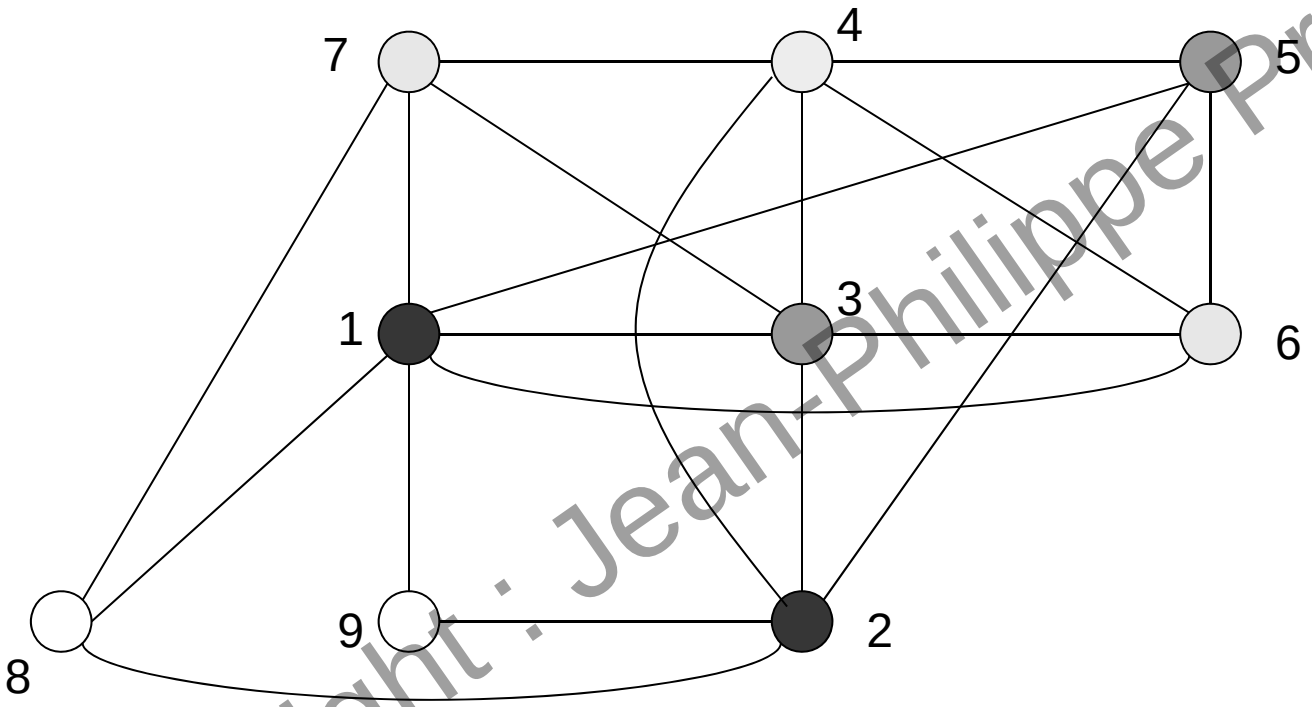
B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

Calcul de B_2 : $\min(7,1+d_7)=5$
max=5



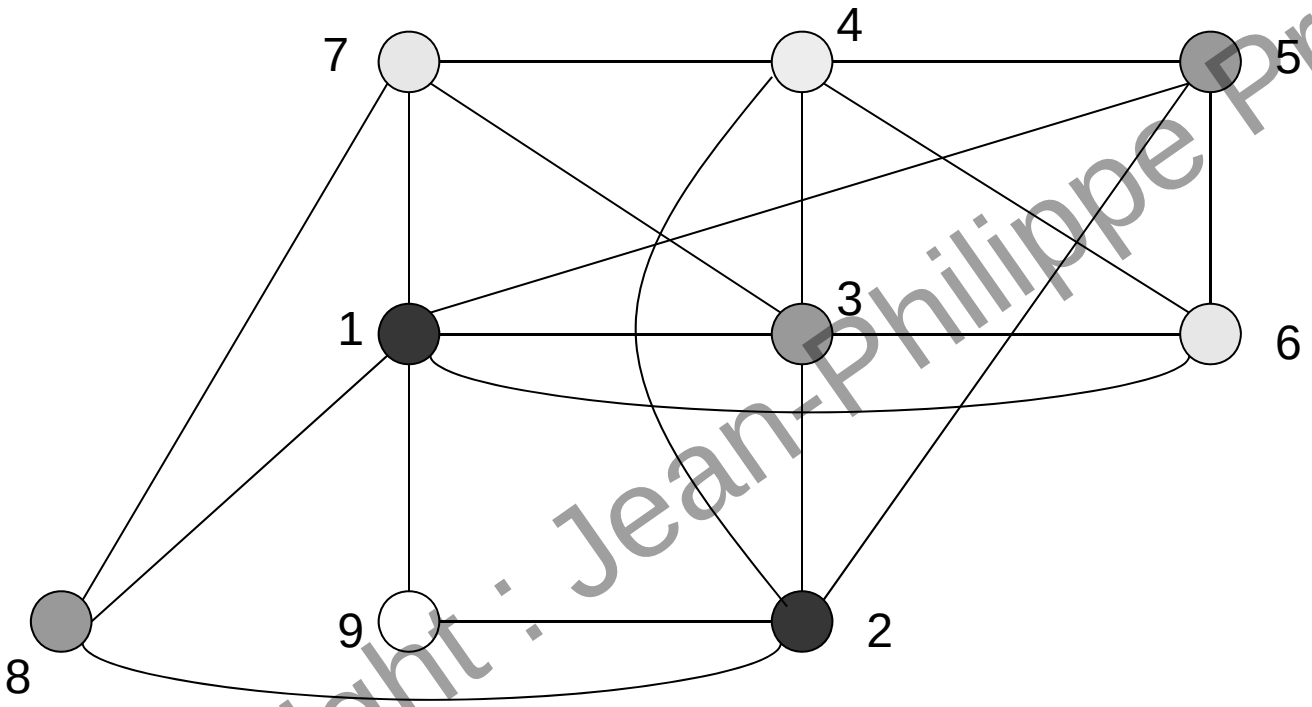
B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

Calcul de B_2 : $\min(8,1+d_8)=4$
max=5



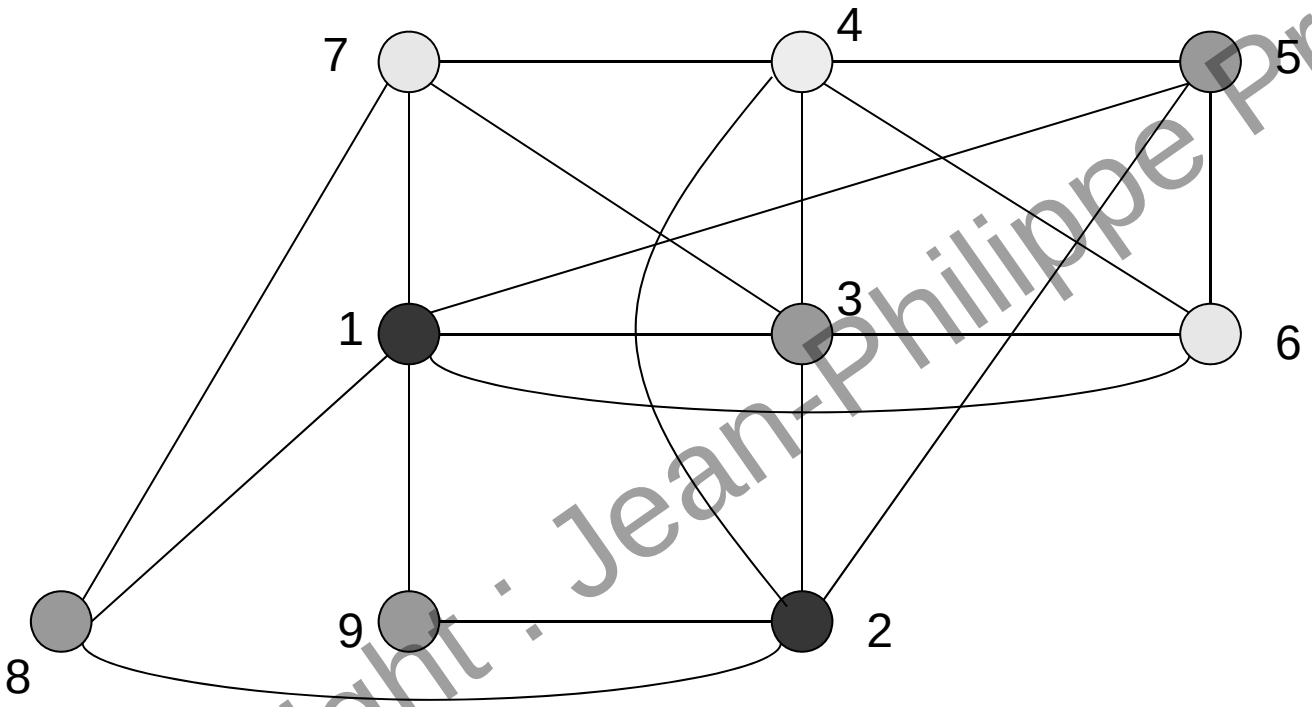
B_1	B_2	B_3	B_4
7			

FFS	LFS	SLS	DS
5			

1 2 3 4 5 6 7

- Exemple : LFS

Calcul de B_2 : $\min(9,1+d_9)=3$
max=5



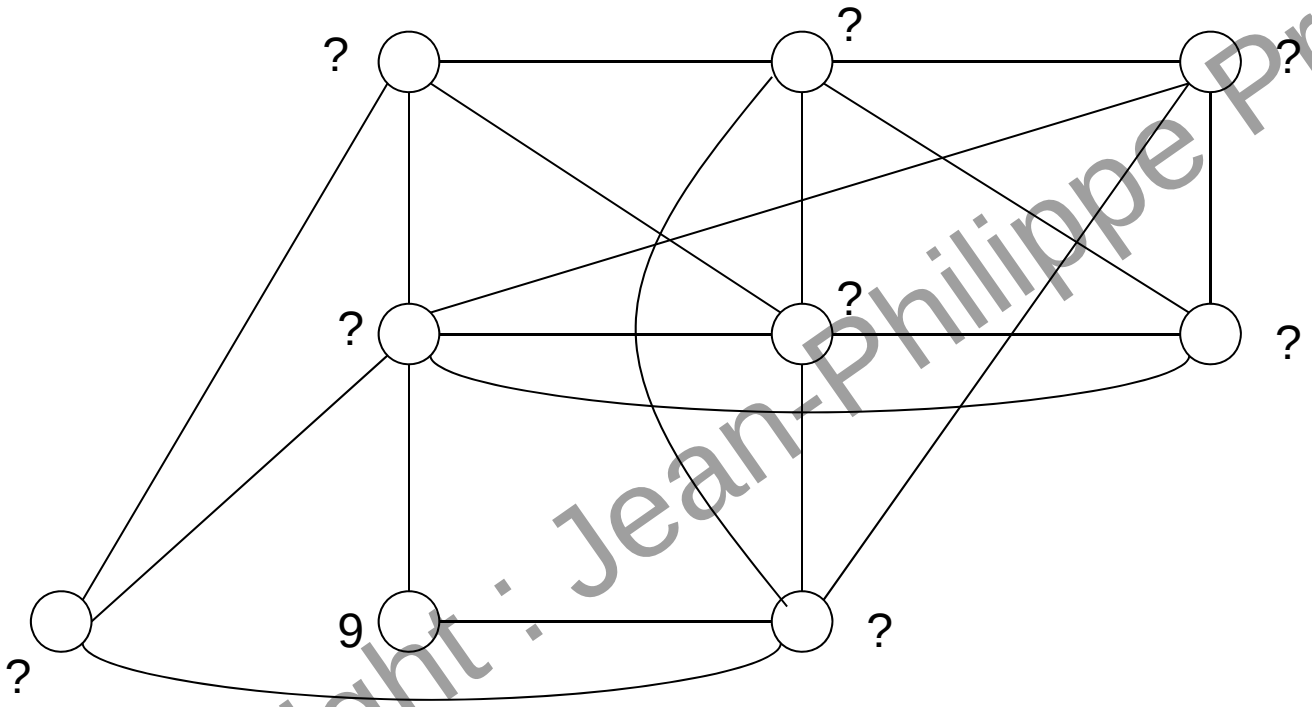
B_1	B_2	B_3	B_4
7	5		

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

Calcul de B_3 : $1+D_9=3$ max=3



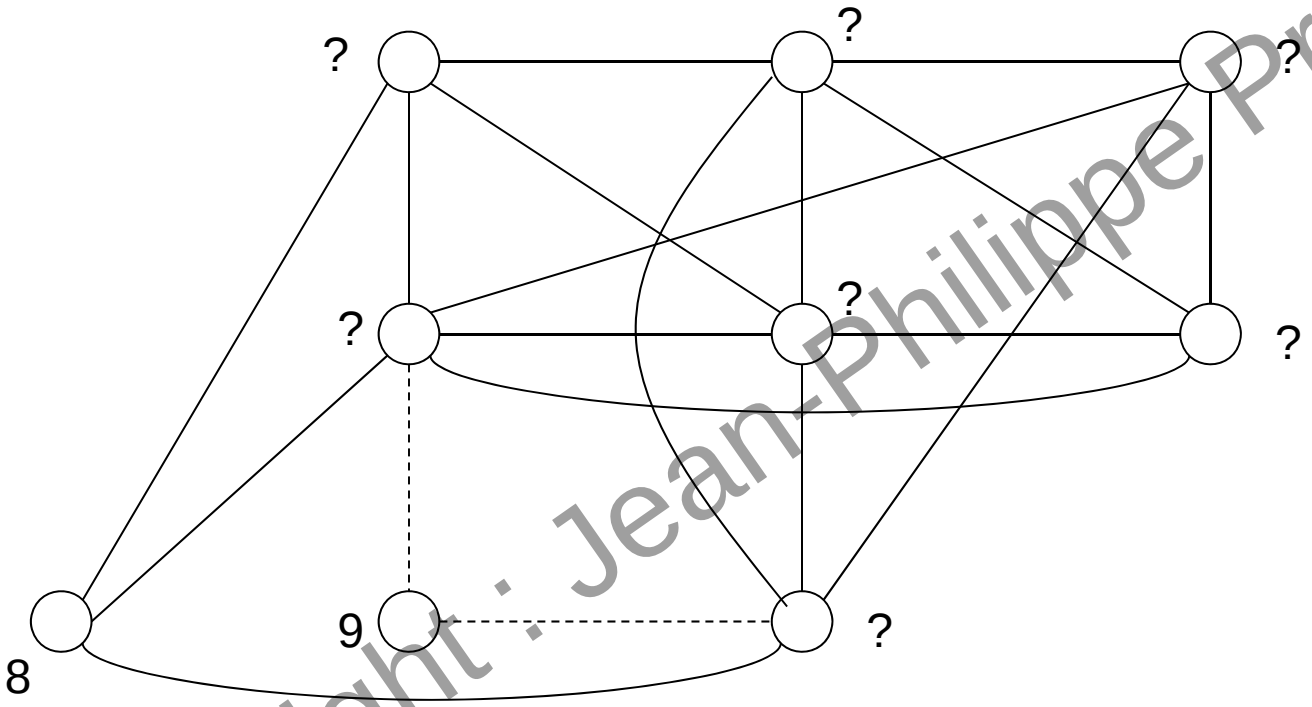
B_1	B_2	B_3	B_4
7	5		

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

Calcul de B_3 : $1+D_8=4$
max=4



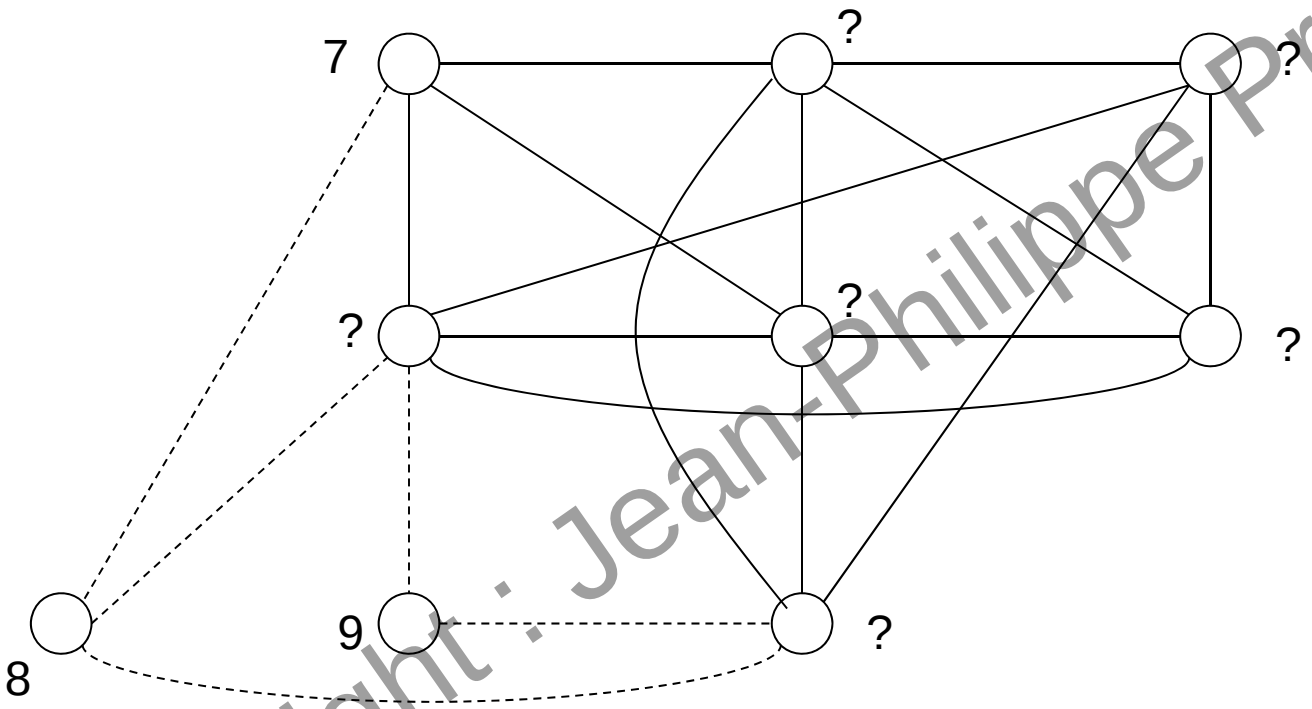
B_1	B_2	B_3	B_4
7	5		

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

Calcul de B_3 : $1+D_7=4$
max=4



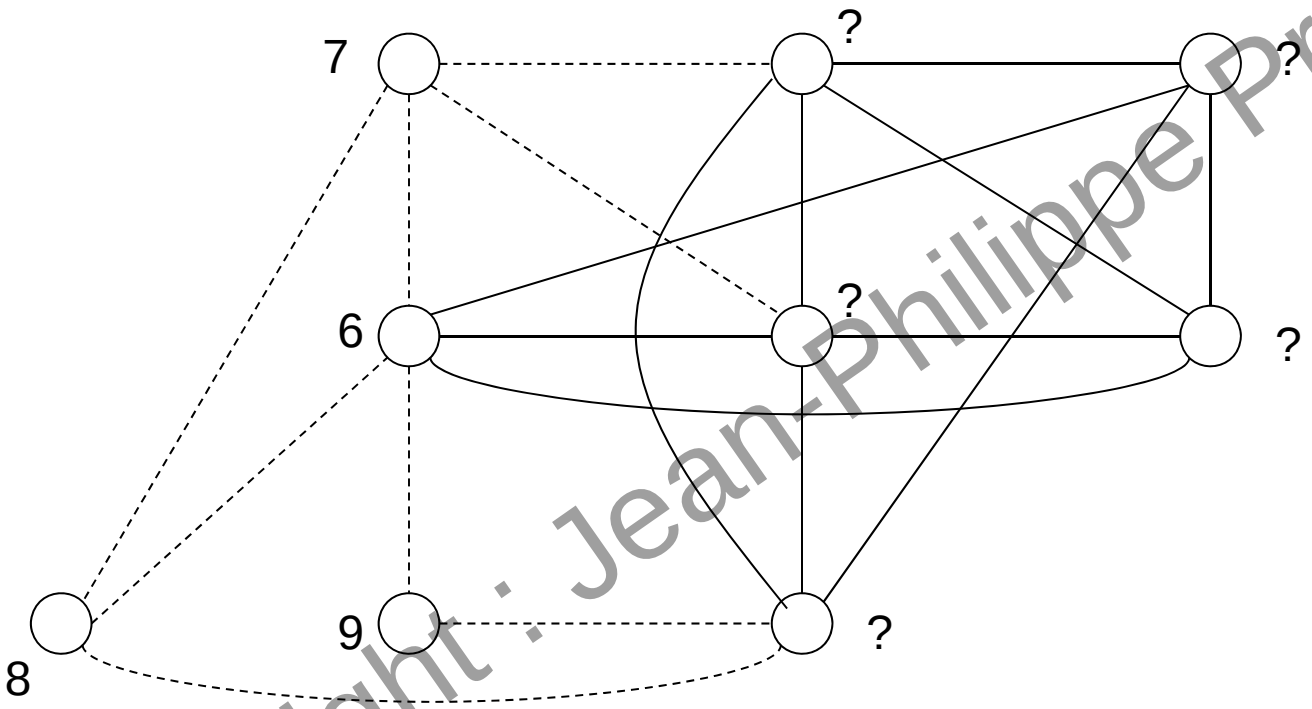
B_1	B_2	B_3	B_4
7	5		

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

Calcul de B_3 : $1+D_6=4$
max=4



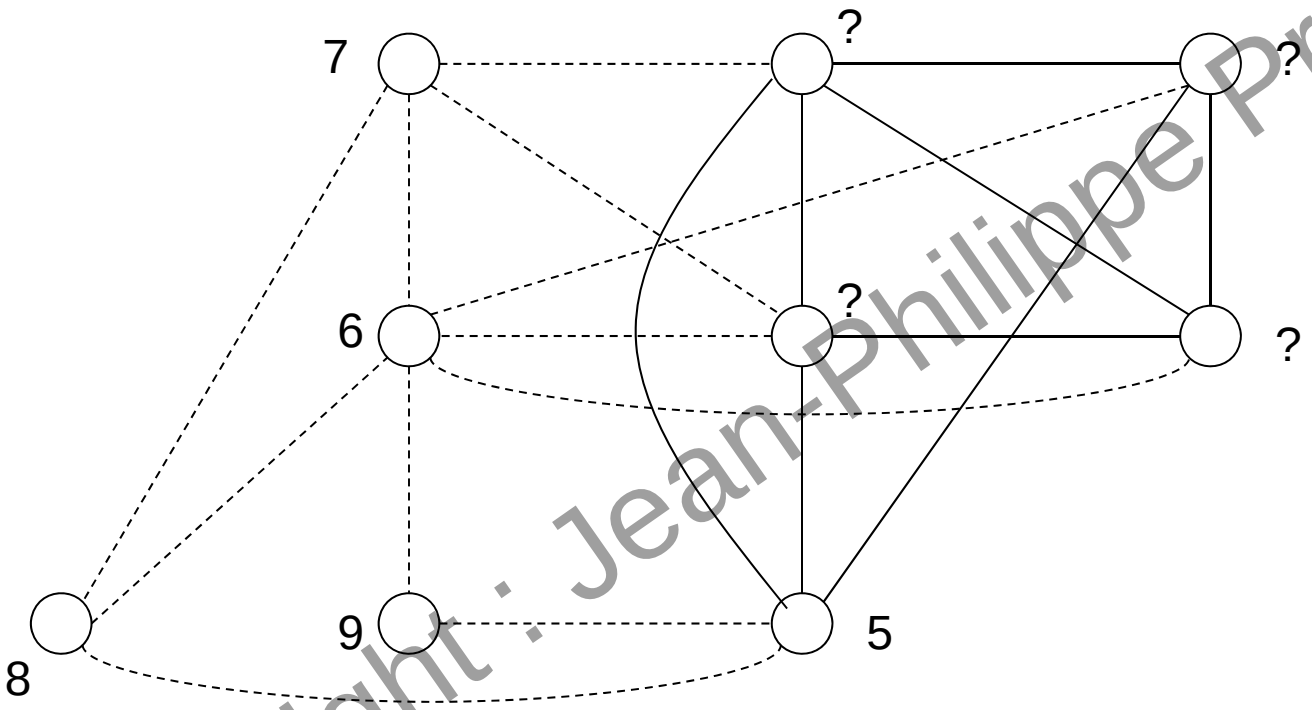
B_1	B_2	B_3	B_4
7	5		

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

Calcul de B_3 : $1+D_5=4$
max=4



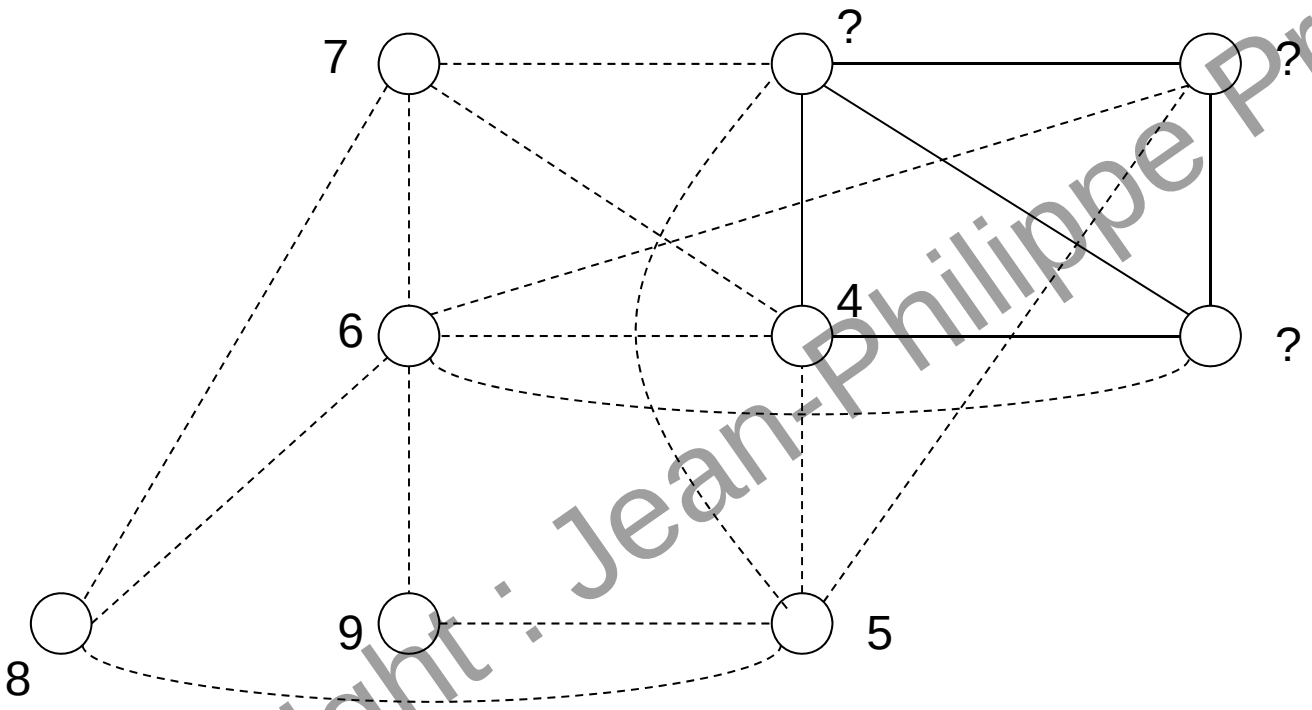
B_1	B_2	B_3	B_4
7	5		

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

Calcul de B_3 : $1+D_4=3$ max=4



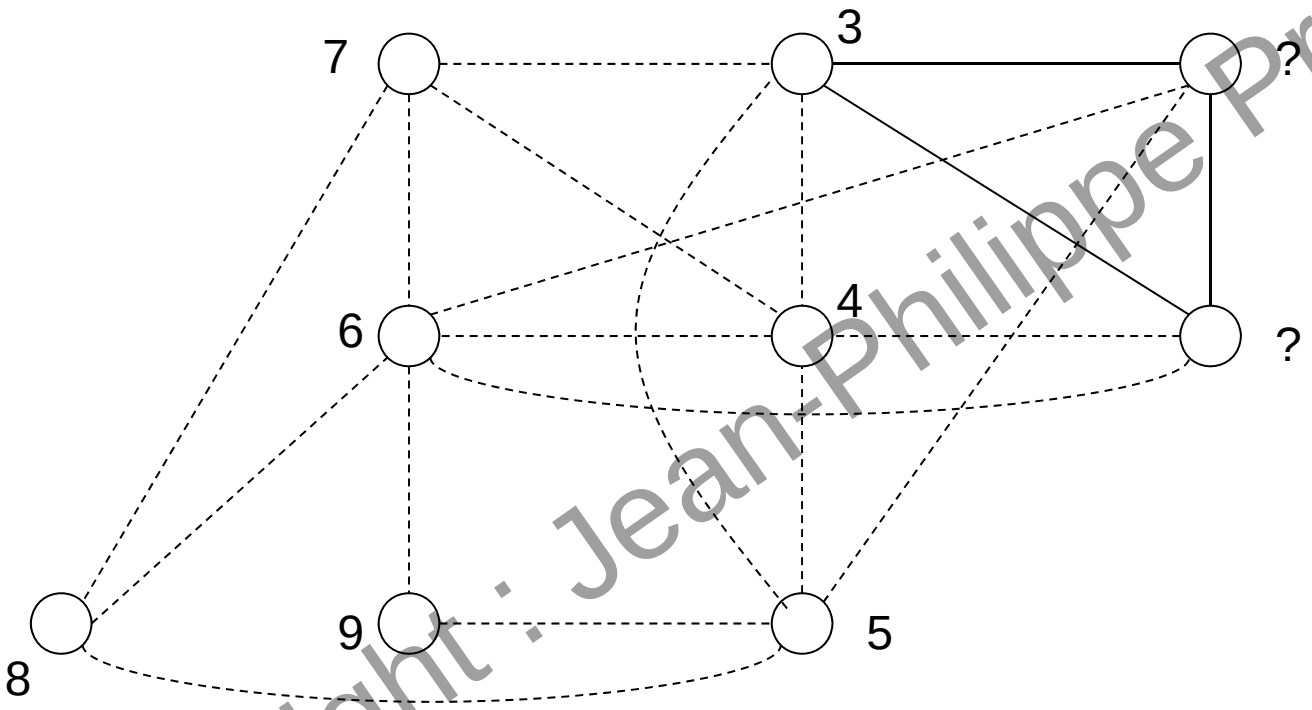
B_1	B_2	B_3	B_4
7	5		

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

Calcul de B_3 : $1+D_3=3$
max=4



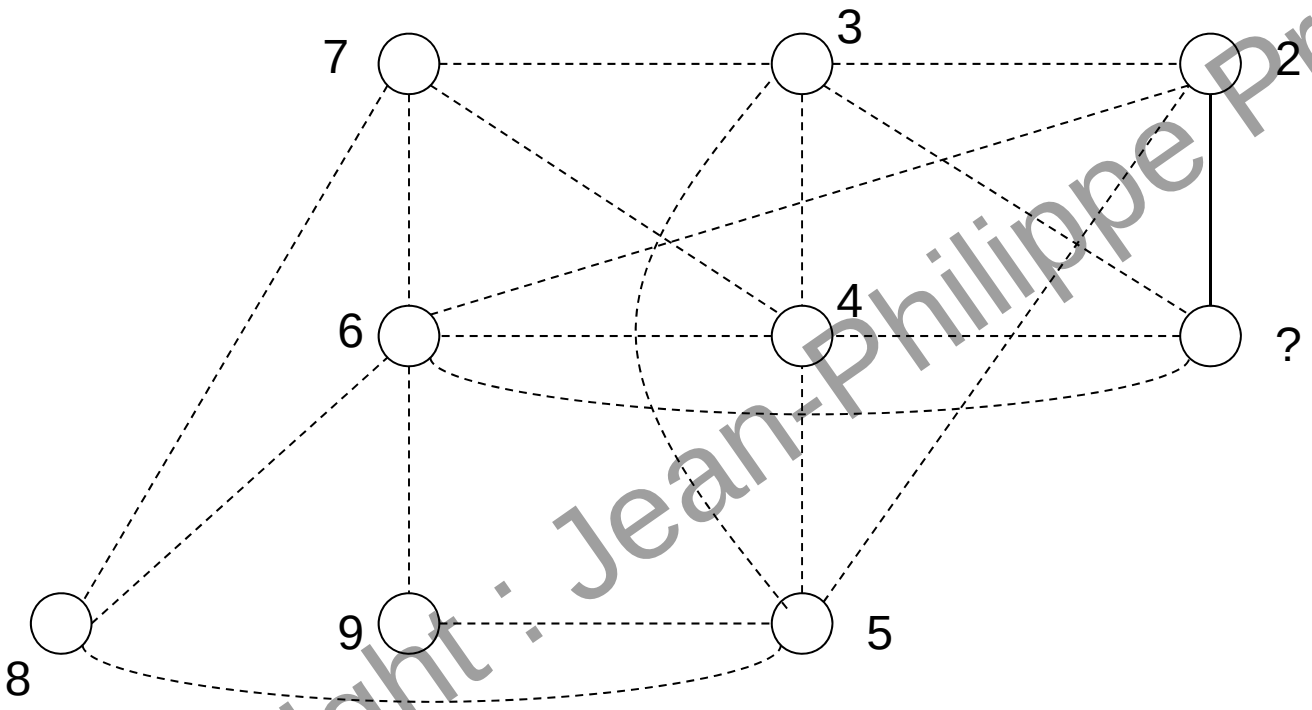
B_1	B_2	B_3	B_4
7	5		

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

Calcul de B_3 : $1+D_2=2$
max=4



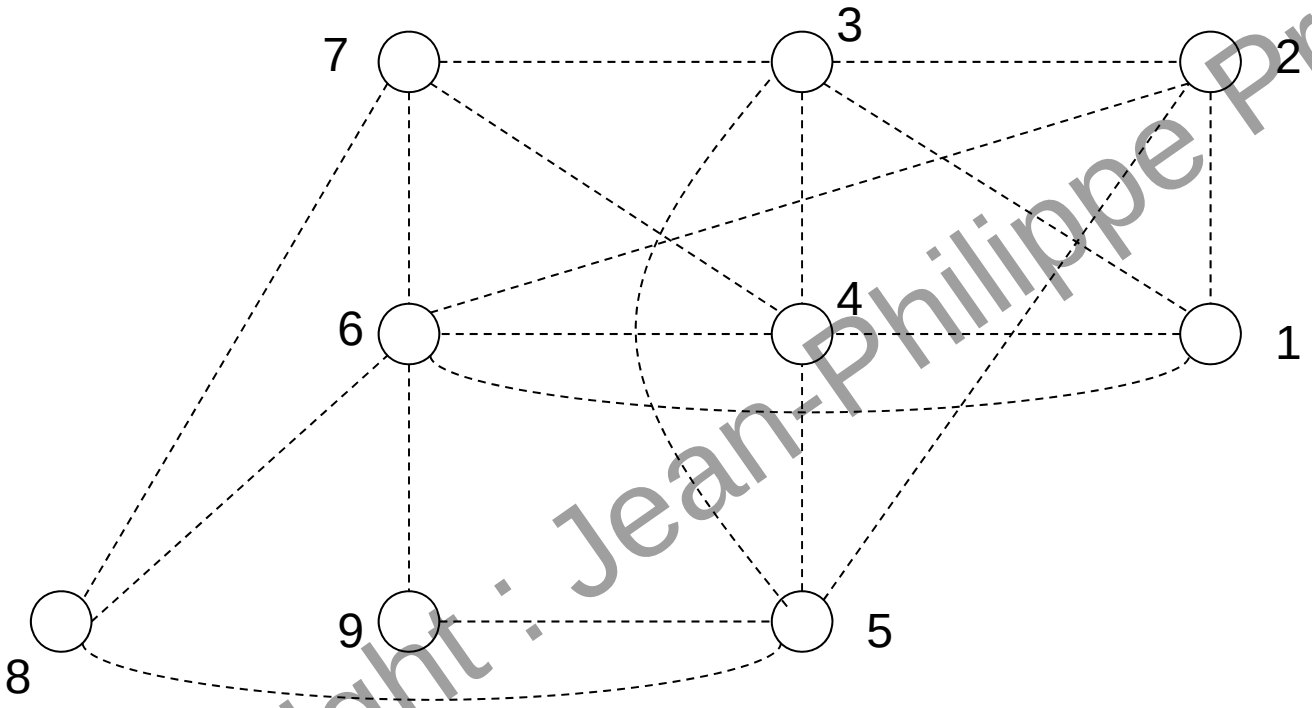
B_1	B_2	B_3	B_4
7	5		

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

Calcul de B_3 : $1+D_1=1$
max=4

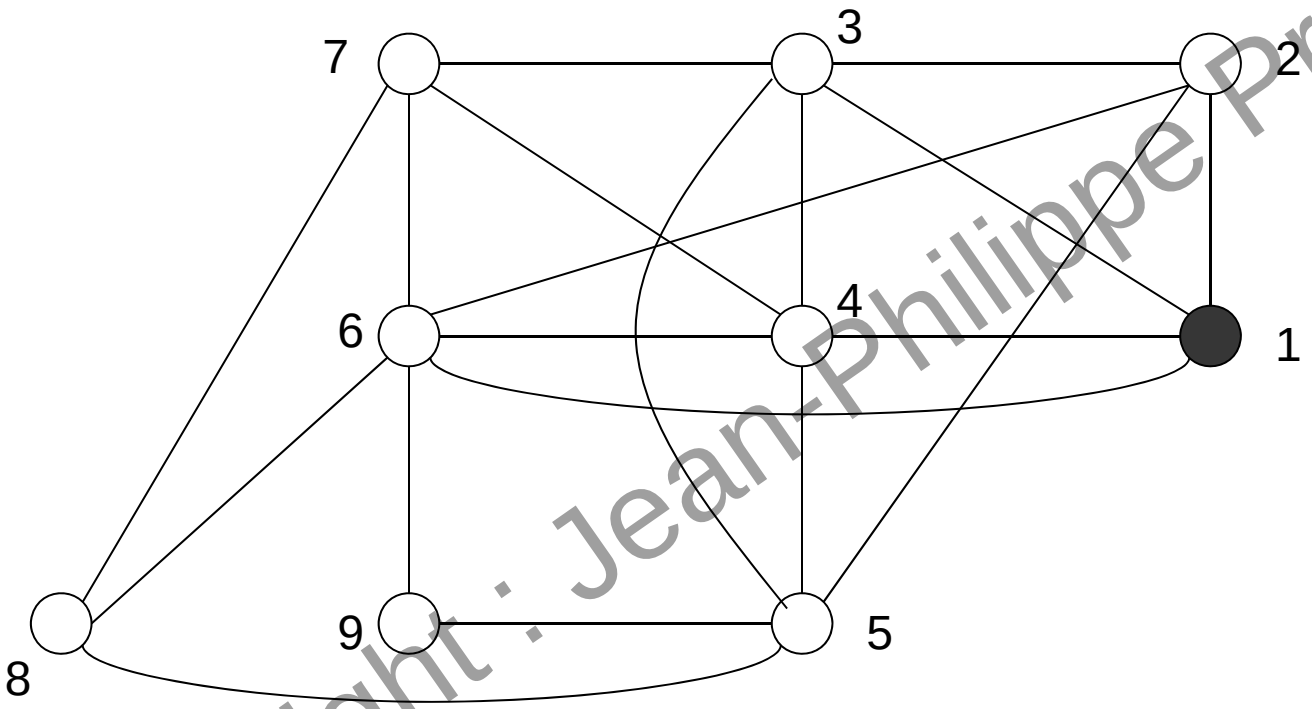


B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

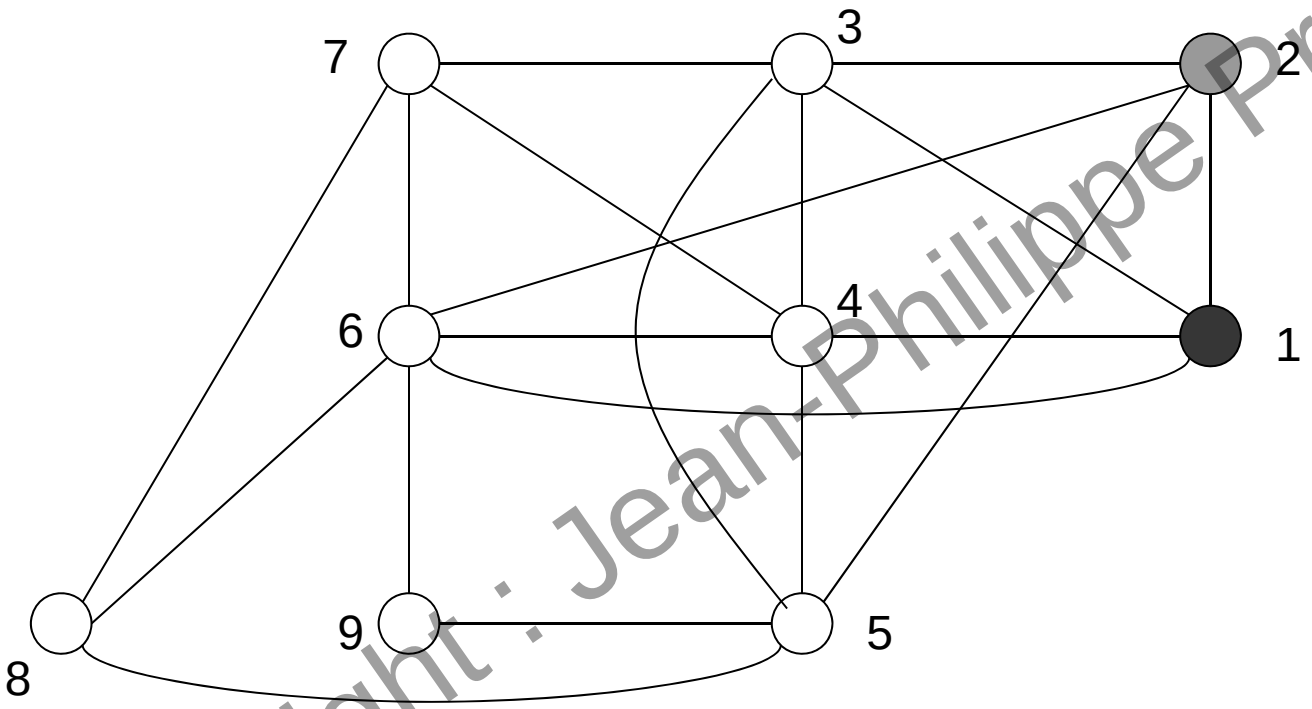


B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

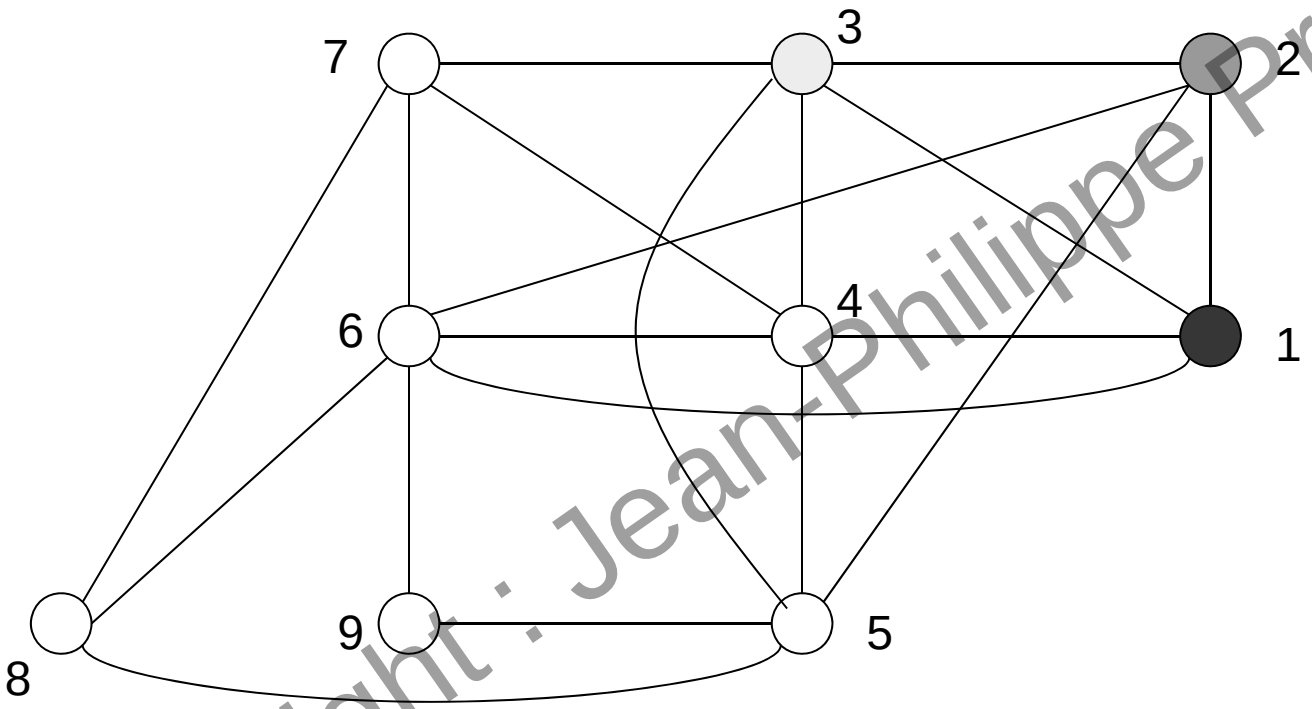


B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

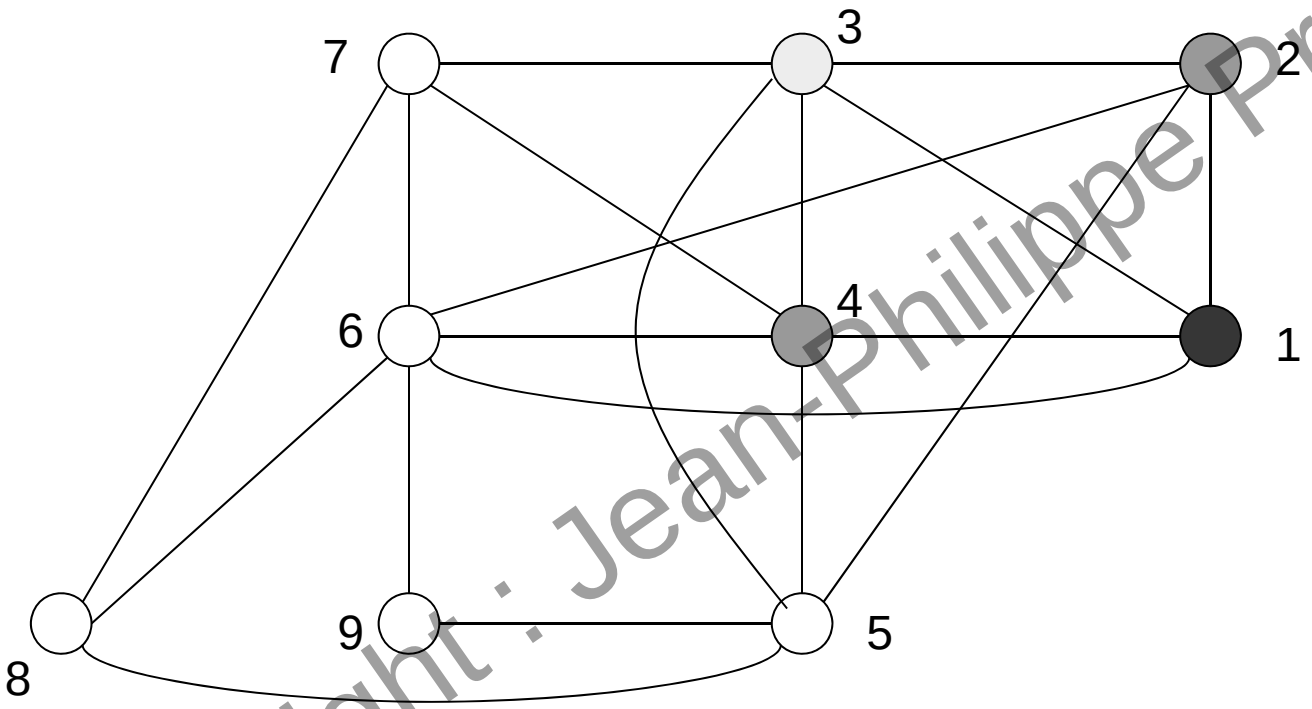


B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

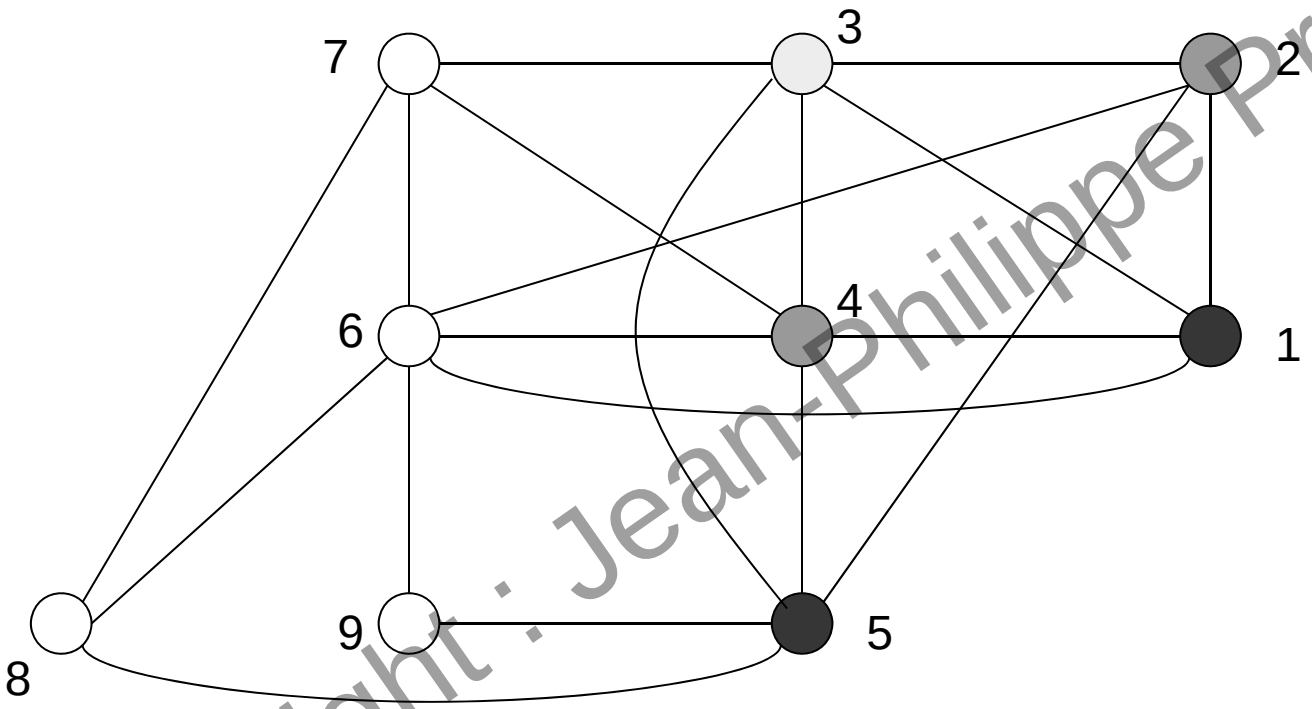


B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

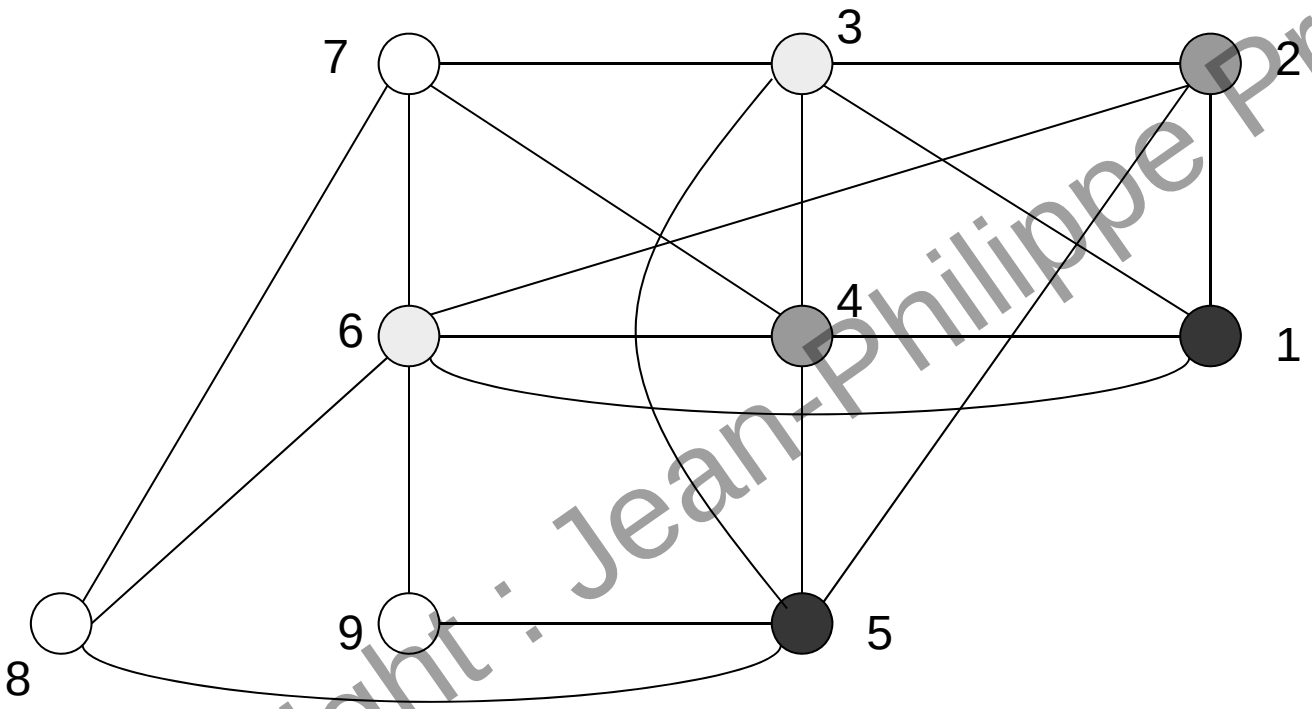


B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

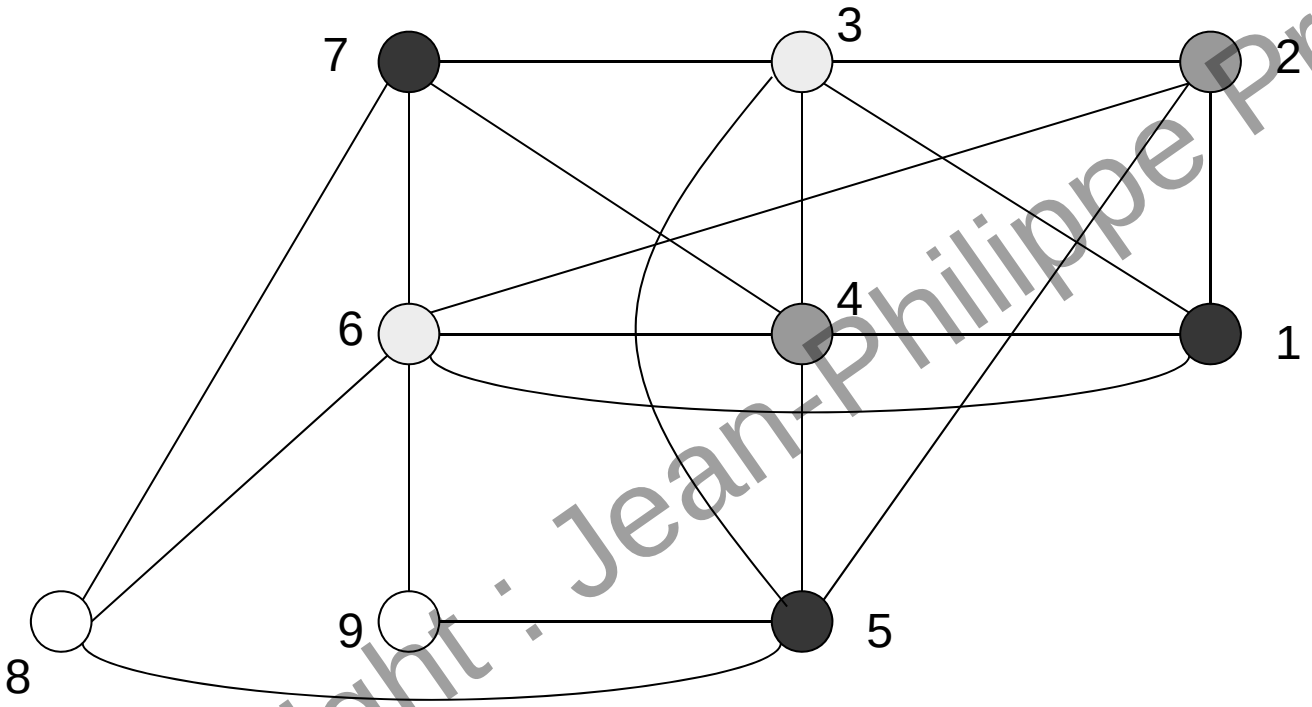


B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

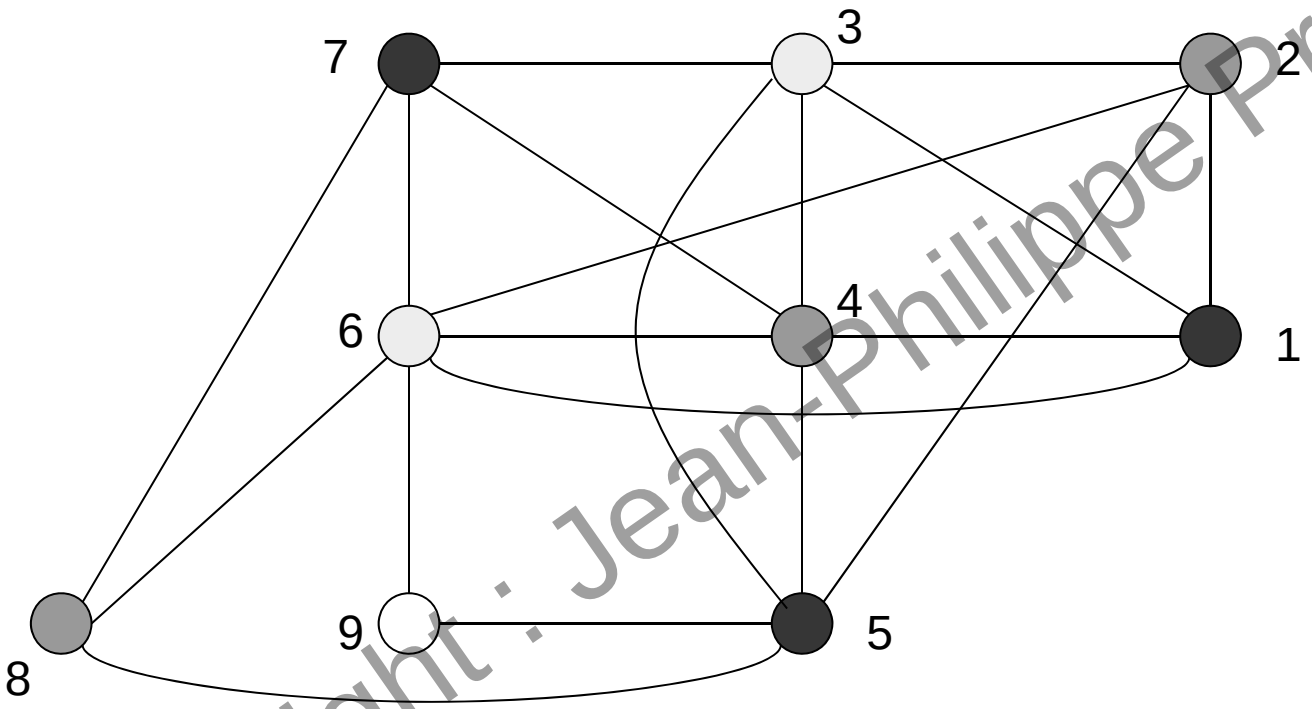


B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

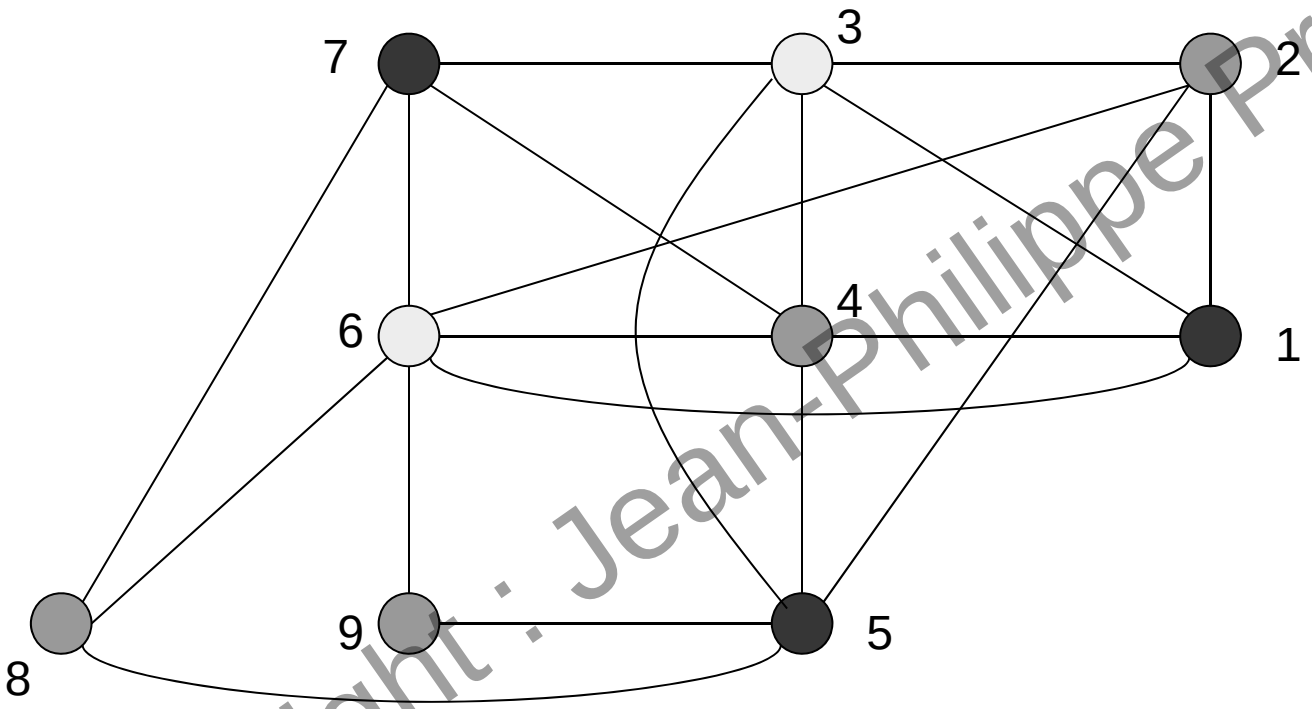


B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4		

1 2 3 4 5 6 7

- Exemple : SLS

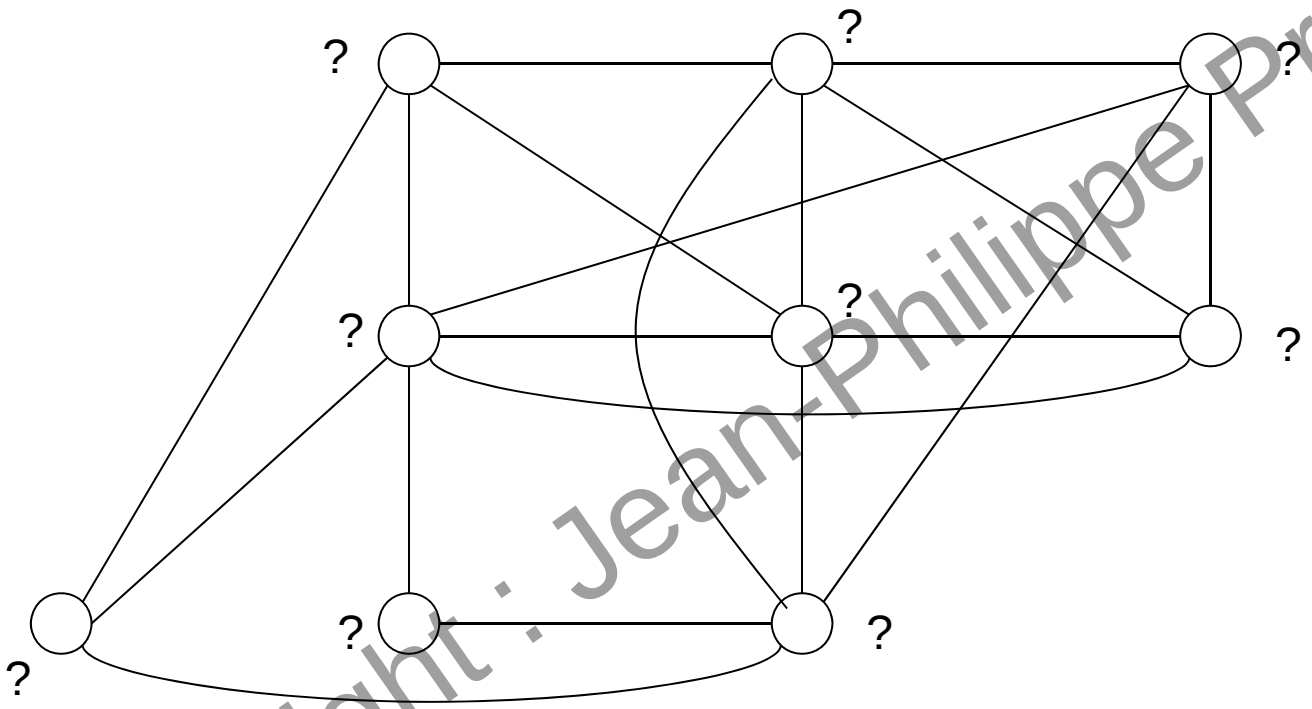


B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4	3	

1 2 3 4 5 6 7

- Exemple : DS



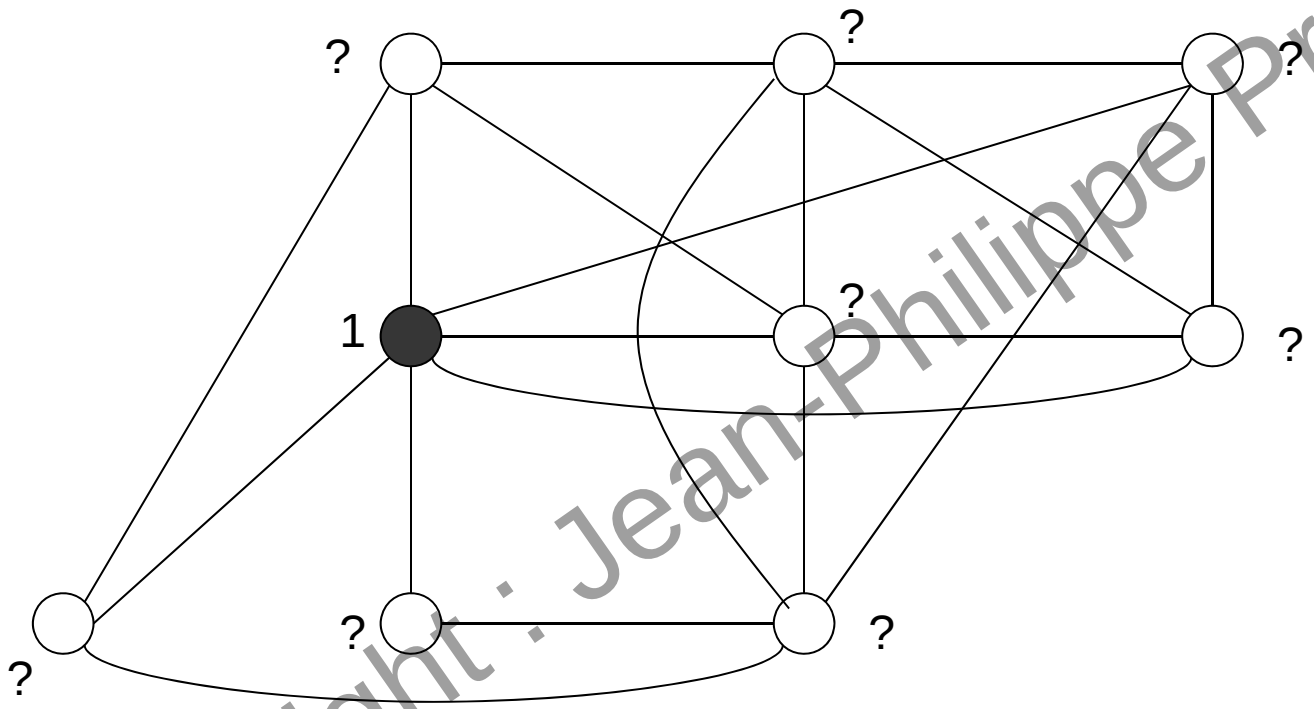
B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4	3	

1 2 3 4 5 6 7

- Exemple : DS

Calcul de B_4 : $1+DS_1=1$
max=1



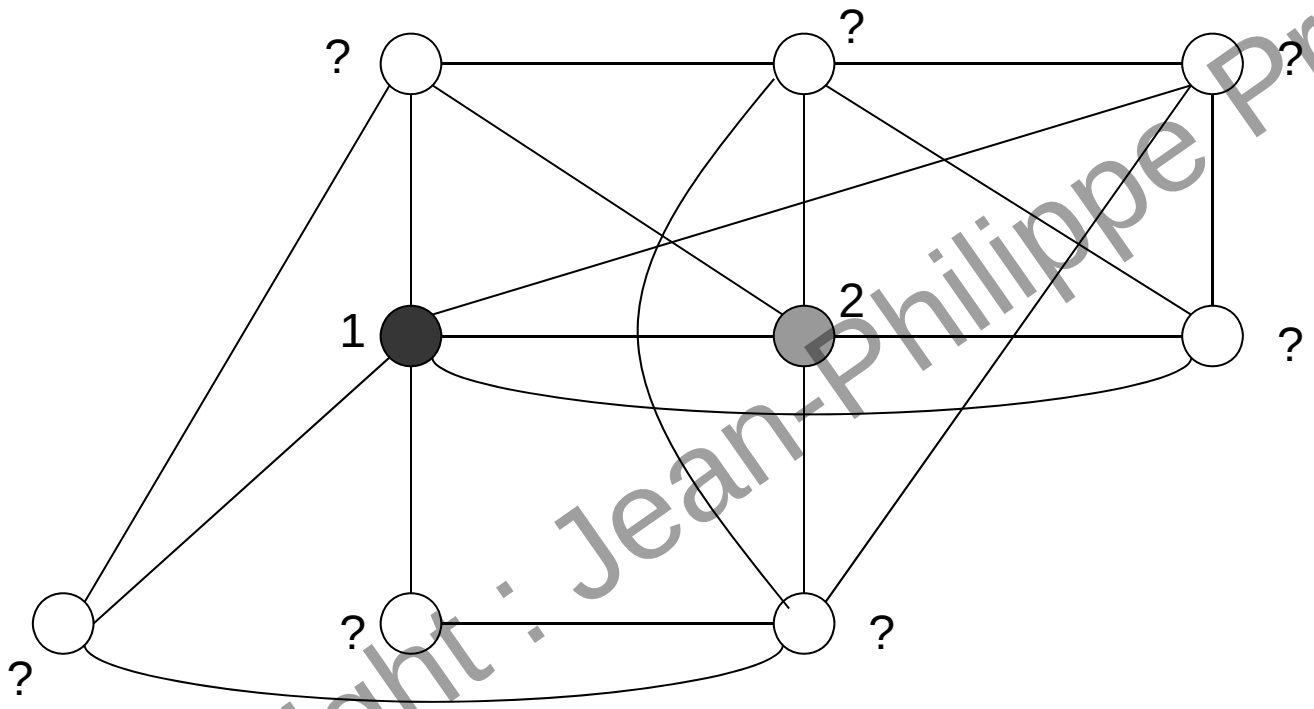
B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4	3	

1 2 3 4 5 6 7

- Exemple : DS

Calcul de B_4 : $1+DS_2=2$
max=2



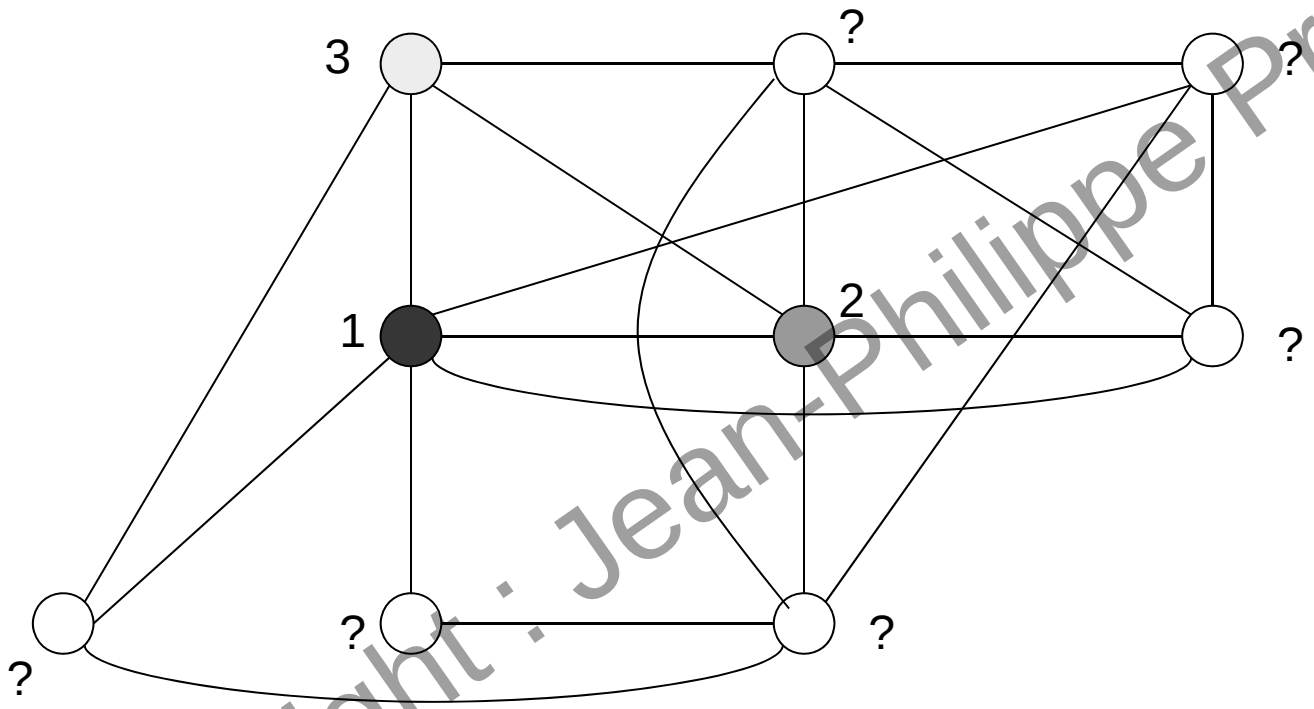
B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4	3	

1 2 3 4 5 6 7

- Exemple : DS

Calcul de B_4 : $1+DS_3=3$
max=3



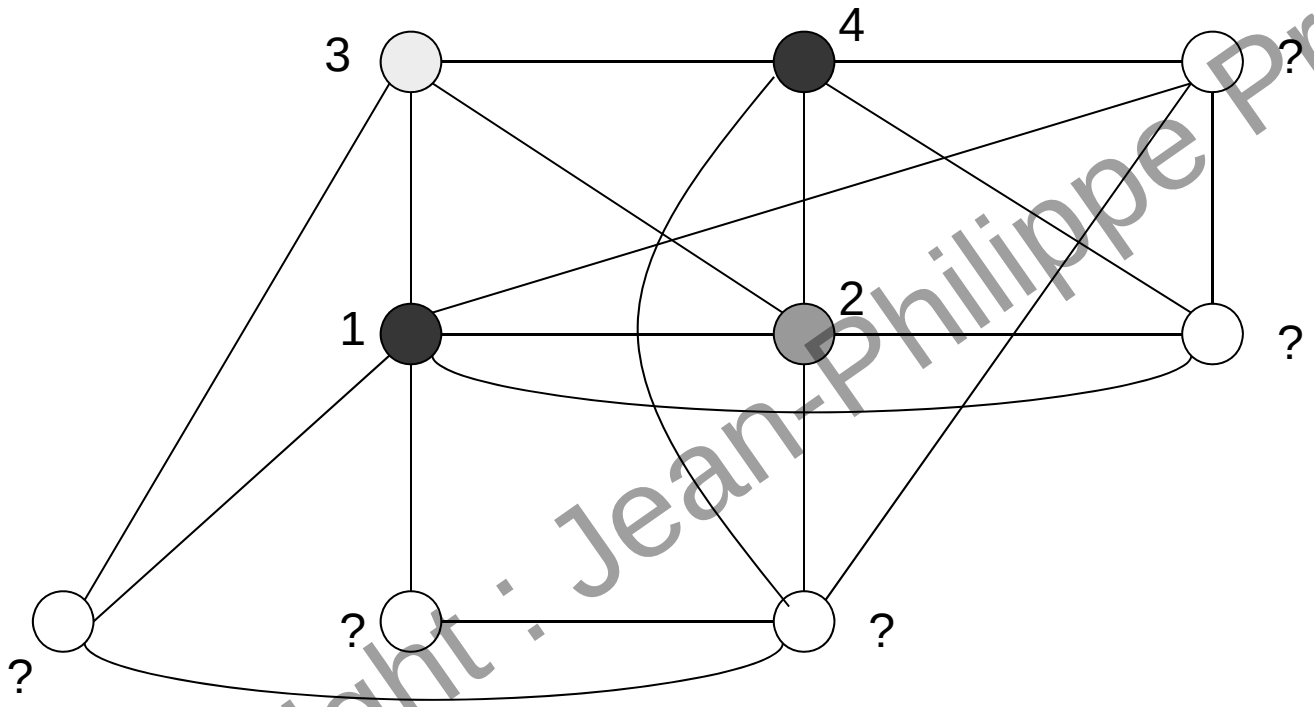
B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4	3	

1 2 3 4 5 6 7

- Exemple : DS

Calcul de B_4 : $1+DS_4=3$
max=3



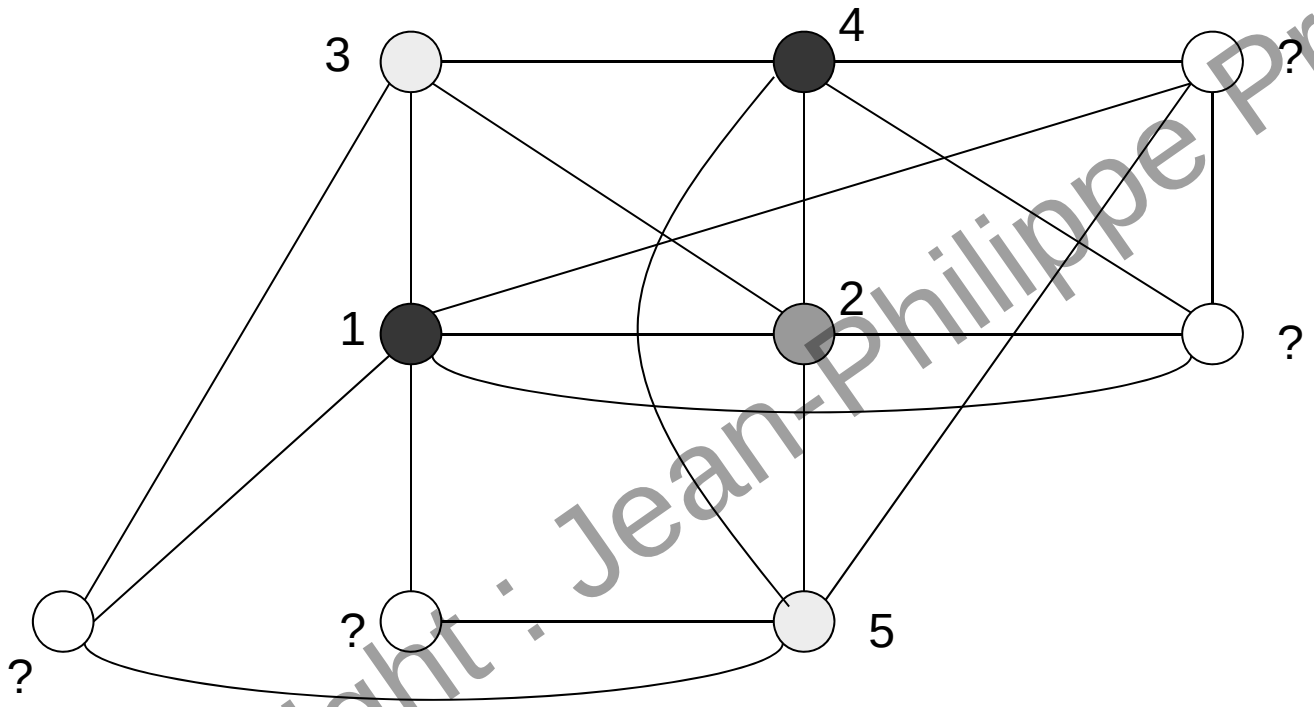
B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4	3	

1 2 3 4 5 6 7

- Exemple : DS

Calcul de B_4 : $1+DS_5=3$
max=3



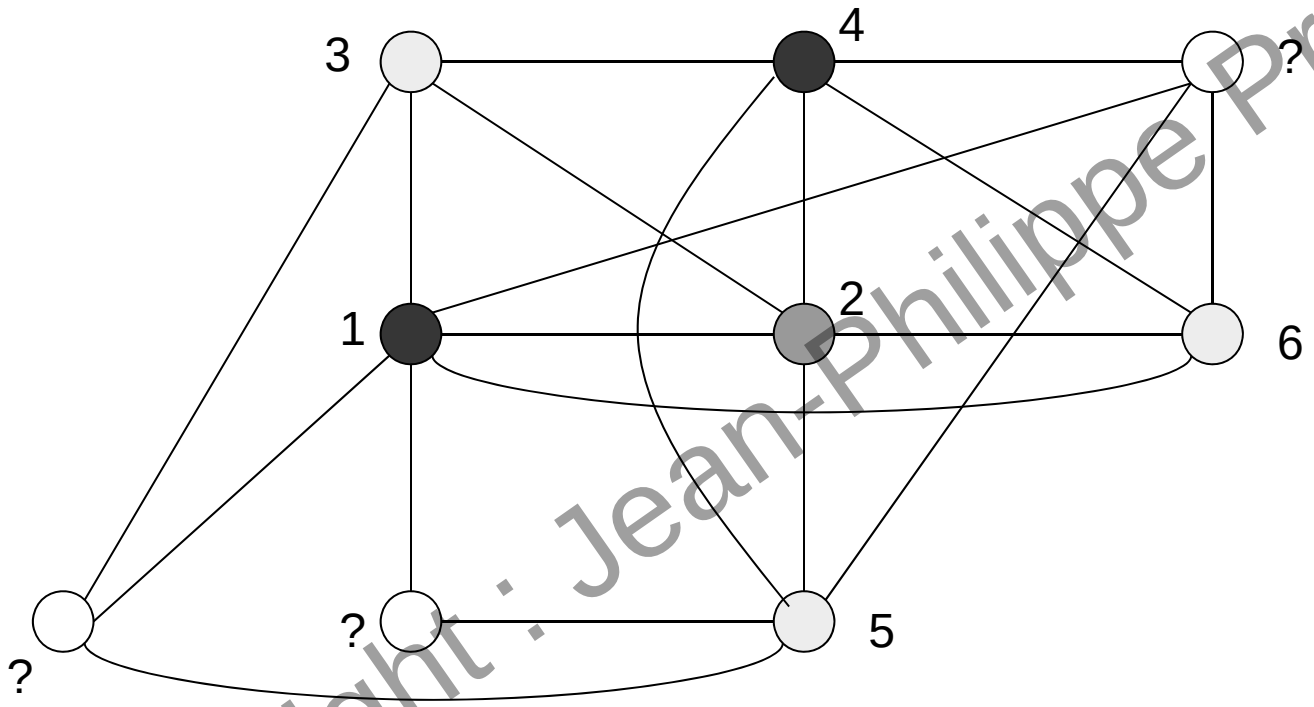
B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4	3	

1 2 3 4 5 6 7

- Exemple : DS

Calcul de B_4 : $1+DS_6=3$
max=3



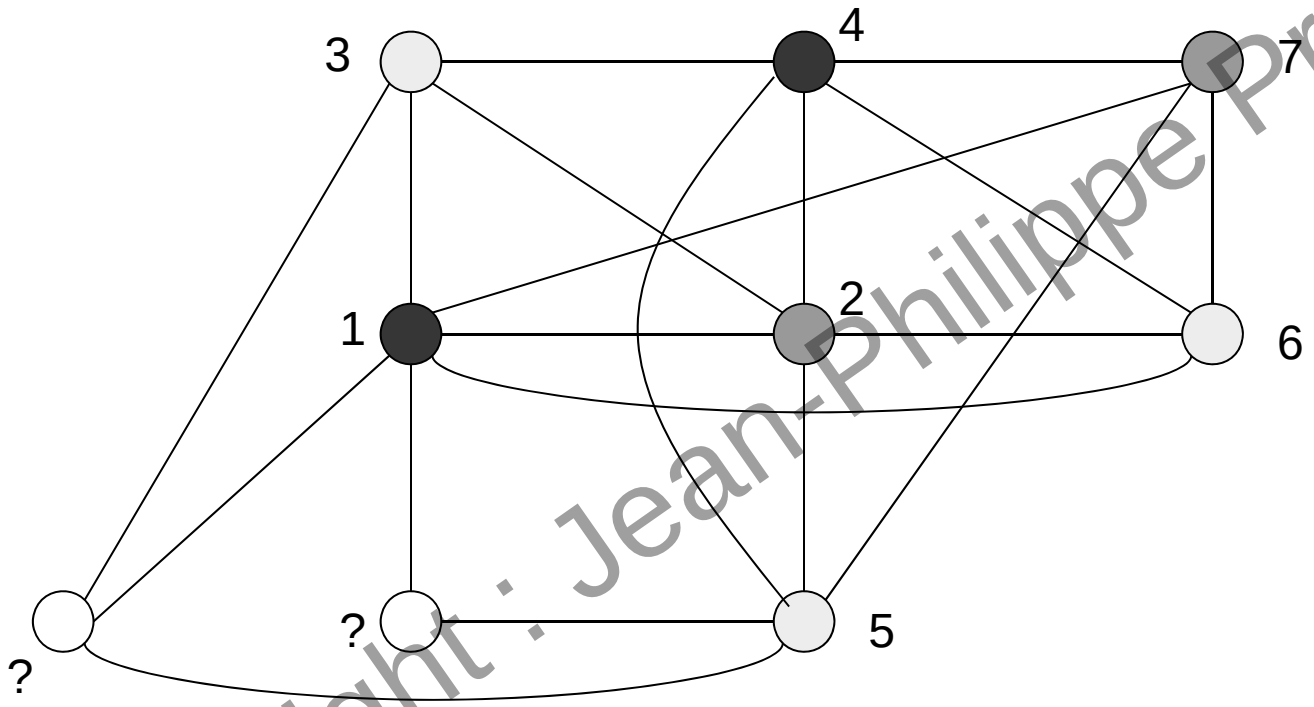
B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4	3	

1 2 3 4 5 6 7

- Exemple : DS

Calcul de B_4 : $1+DS_7=3$
max=3



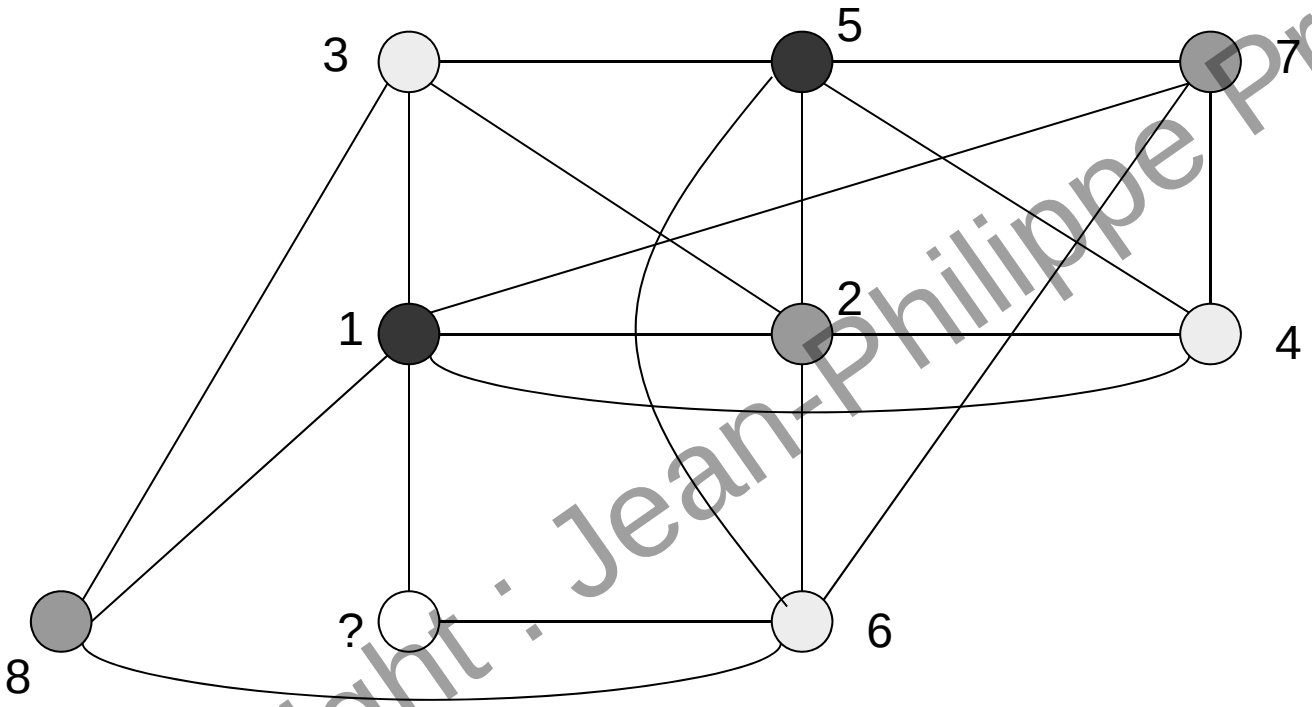
B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4	3	

1 2 3 4 5 6 7

- Exemple : DS

Calcul de B_4 : $1+DS_8=3$
max=3



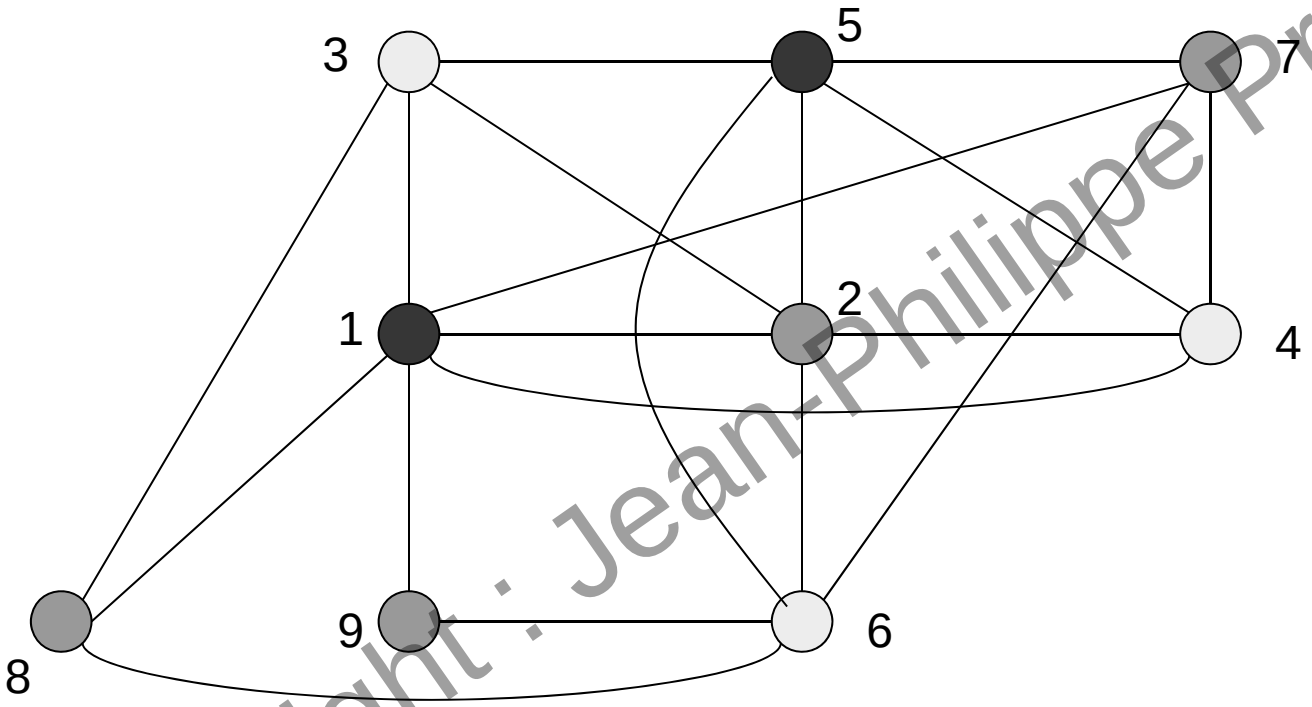
B_1	B_2	B_3	B_4
7	5	4	

FFS	LFS	SLS	DS
5	4	3	

1 2 3 4 5 6 7

- Exemple : DS

Calcul de B_4 : $1+DS_9=3$
max=3



B_1	B_2	B_3	B_4
7	5	4	3

FFS	LFS	SLS	DS
5	4	3	3

1 2 3 4 5 6 7

Heuristiques gloutonnes pour le PVC

- Nous avons déja vu l'heuristique du **plus proche voisin** (**PPV**), la plus naïve. Elle s'implémente en $O(N^2)$.

Heuristiques gloutonnes pour le PVC

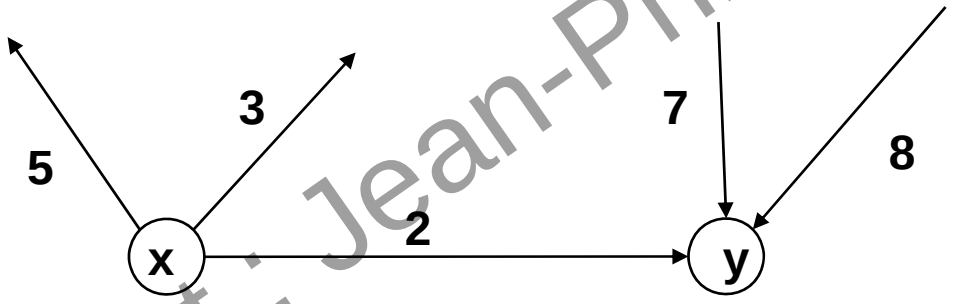
- Nous avons déja vu l'heuristique du **plus proche voisin** (**PPV**), la plus naïve. Elle s'implémente en $O(N^2)$.
- Pour le Δ-PVC sa performance relative au pire est $\log_2 N$. C'est franchement mauvais : pour 1024 sommets elle peut trouver un coût 10 fois supérieur au coût optimal...
- Sa performance moyenne est meilleure.

Heuristiques gloutonnes pour le PVC

- Une approche moins naïve consiste à ajouter à chaque étape non pas l'arête de coût minimal au départ de x, mais celle de **regret maximal**. C'est une **méthode à pénalité**.

Heuristiques gloutonnes pour le PVC

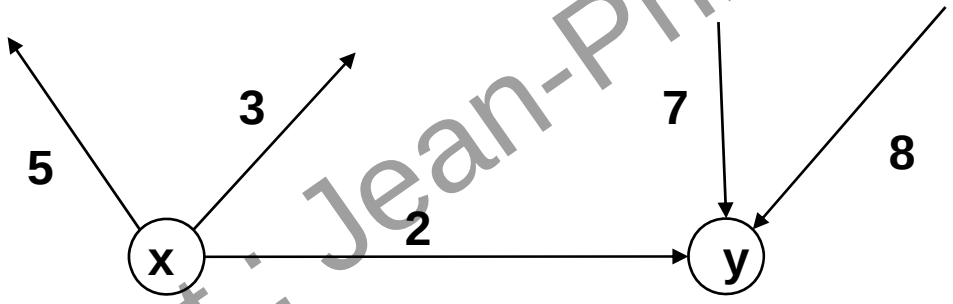
- Une approche moins naïve consiste à ajouter à chaque étape non pas l'arête de coût minimal au départ de x, mais celle de **regret maximal**. C'est une **méthode à pénalité**.



- [x,y] a pour coût 2 et pour regret (ou pénalité) 3+7=10.

Heuristiques gloutonnes pour le PVC

- Une approche moins naïve consiste à ajouter à chaque étape non pas l'arête de coût minimal au départ de x, mais celle de **regret maximal**. C'est une **méthode à pénalité**.



- [x,y] a pour coût 2 et pour regret (ou pénalité) 3+7=10.
- Les méthodes par pénalité sont meilleures en moyenne car moins locales, mais en $O(n^3)$...

Méthode à pénalité pour le PVC

- Reprenons l'exemple précédent :

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

Méthode à pénalité pour le PVC

- Reprenons l'exemple précédent :

	A	B	C	D	E
A	-	3(5)	4(4)	5(4)	4(4)
B	3(5)	-	2(2)	2(2)	1(4)
C	4(4)	2(2)	-	1(4)	2(2)
D	5(3)	2(2)	1(4)	-	3(2)
E	4(4)	1(4)	2(3)	3(2)	-

Méthode à pénalité pour le PVC

- Reprenons l'exemple précédent :

	A	B	C	D	E
A	-	**3(5)**	4(4)	5(4)	4(4)
B	3(5)	-	2(2)	2(2)	1(4)
C	4(4)	2(2)	-	1(4)	2(2)
D	5(3)	2(2)	1(4)	-	3(2)
E	4(4)	1(4)	2(3)	3(2)	-

- A-B
- coût=3

Méthode à pénalité pour le PVC

- Reprenons l'exemple précédent :

	A	B	C	D	E
A	-	-	-	-	-
B	-	-	2(2)	2(2)	**1(4)**
C	4	-	-	1	2
D	5	-	1	-	3
E	4	-	2	3	-

- A-B-E
- coût=3+1

Méthode à pénalité pour le PVC

- Reprenons l'exemple précédent :

	A	B	C	D	E
A	-	**-**	-	**-**	-
B	-	-	-	-	**-**
C	4	-	-	1	-
D	5	-	1	-	-
E	-	-	**2(4)**	3(3)	-

- A-B-E-C
- coût=3+1+2

Méthode à pénalité pour le PVC

- Reprenons l'exemple précédent :

	A	B	C	D	E
A	-	-	-	-	-
B	-	-	-	-	-
C	-	-	-	1	-
D	5	-	-	-	-
E	-	-	-	-	-

- A-B-E-C-D-A : cycle hamiltonien
- coût=3+1+2+1+5=12 (optimal ici...)

Méthodes par insertion pour le PVC

- On part d’un cycle U trivial sur un sommet arbitraire.

Méthodes par insertion pour le PVC

- On part d'un cycle U trivial sur un sommet arbitraire.
- A chaque itération on choisit un sommet libre k, et on cherche la position d'insertion entre deux sommets consécutifs i,j de U, qui minimise l'augmentation de coût :

$$\Delta C = C_{ik} + C_{kj} - C_{ij}$$

Méthodes par insertion pour le PVC

- On part d'un cycle U trivial sur un sommet arbitraire.
- A chaque itération on choisit un sommet libre k, et on cherche la position d'insertion entre deux sommets consécutifs i,j de U, qui minimise l'augmentation de coût :

$$\Delta C = C_{ik} + C_{kj} - C_{ij}$$

- On change U en un cycle ayant un sommet supplémentaire en supprimant l'arête [i,j] et en ajoutant [i,k] et [k,j].

Méthodes par insertion pour le PVC

- On part d'un cycle U trivial sur un sommet arbitraire.
- A chaque itération on choisit un sommet libre k, et on cherche la position d'insertion entre deux sommets consécutifs i,j de U, qui minimise l'augmentation de coût :

$$\Delta C = C_{ik} + C_{kj} - C_{ij}$$

- On change U en un cycle ayant un sommet supplémentaire en supprimant l'arête [i,j] et en ajoutant [i,k] et [k,j].
- On poursuit tant qu'il reste un sommet libre.

Méthodes par insertion pour le PVC

Selon le choix du sommet libre on a différentes heuristiques :

Méthodes par insertion pour le PVC

Selon le choix du sommet libre on a différentes heuristiques :

- Plus proche insertion (PPI) :

  Le sommet choisi est le plus proche de U (où dist(x,U)=min $\{C_{xy}$: y sommet de U$\}$.)

Méthodes par insertion pour le PVC

Selon le choix du sommet libre on a différentes heuristiques :

- Plus proche insertion (PPI) :

  Le sommet choisi est le plus proche de U (où dist(x,U)=min $\{C_{xy}$: y sommet de U$\}$.)

- Plus lointaine insertion (PLI) :

  Le sommet choisi est le plus éloigné de U

Méthodes par insertion pour le PVC

Selon le choix du sommet libre on a différentes heuristiques :

- Plus proche insertion (PPI) :

  Le sommet choisi est le plus proche de U (où dist(x,U)=min $\{C_{xy}$: y sommet de U$\}$.)
- Plus lointaine insertion (PLI) :

  Le sommet choisi est le plus éloigné de U
- Meilleure insertion (MI) :

  Le sommet choisi est celui qui donne la plus faible augmentation de coût.

PPI

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

PLI

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

MI

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

PPI

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

A

PLI

	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

A

MI

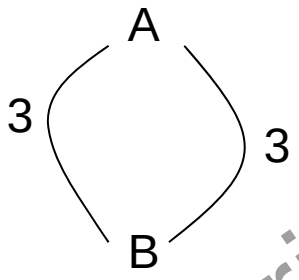
	A	B	C	D	E
A	-	3	4	5	4
B	3	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

A

On choisit (arbitrairement ici...) A comme sommet initial.

PPI

	A	**B**	C	D	E
A	-	**3**	4	5	4
B	**3**	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

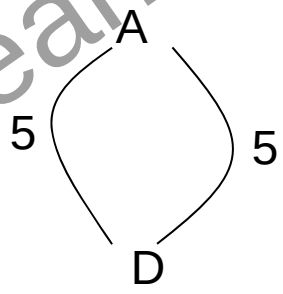


Le plus proche est B

(distance=3) Δ=6

PLI

	A	B	C	**D**	E
A	-	3	4	**5**	4
B	3	-	2	2	1
C	4	2	-	1	2
D	**5**	2	1	-	3
E	4	1	2	3	-

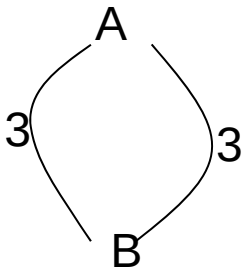


Le plus éloigné est D

(distance=5) Δ=10

MI

	A	**B**	C	D	E
A	-	**3**	4	5	4
B	**3**	-	2	2	1
C	4	2	-	1	2
D	5	2	1	-	3
E	4	1	2	3	-

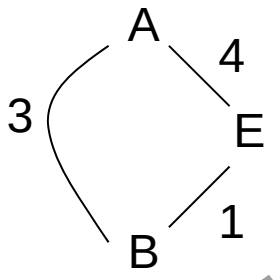


Le meilleur est B

Δ=6

PPI

	A	*B*	C	D	**E**
A	-	-	4	5	**4**
B	-	-	2	2	**1**
C	4	2	-	1	2
D	5	2	1	-	3
E	**4**	**1**	2	3	-

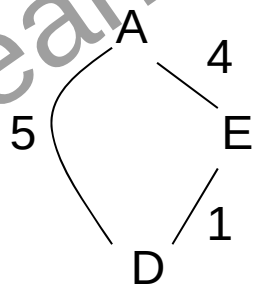


Le plus proche est E

(distance 1) Δ=2

PLI

	A	B	C	*D*	**E**
A	-	3	4	-	**4**
B	3	-	2	2	1
C	4	2	-	1	2
D	-	2	1	-	**3**
E	**4**	1	2	**3**	-

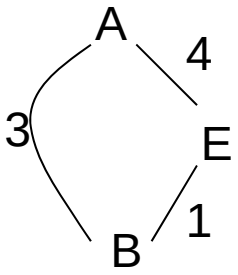


Le plus éloigné est D

(distance 3) Δ=0

MI

	A	B	C	D	**E**
A	-	-	4	5	**4**
B	-	-	2	2	**1**
C	4	2	-	1	2
D	5	2	1	-	3
E	**4**	**1**	2	3	-

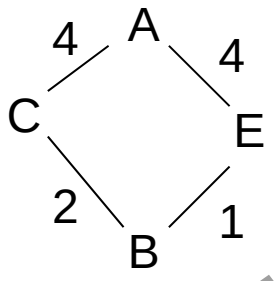


Le meilleur est E

(Δ=2)

PPI

	A	*B*	**C**	D	*E*
A	-	-	**4**	5	-
B	-	-	**2**	2	-
C	**4**	**2**	-	1	**2**
D	5	2	1	-	3
E	-	-	**2**	3	-

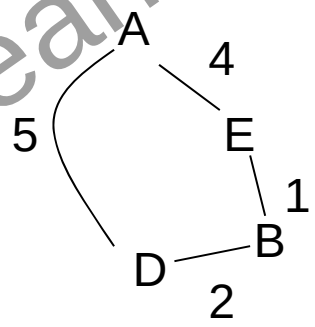


Le plus proche est C

(et D, distance 2) Δ=3

PLI

	A	**B**	C	*D*	*E*
A	-	**3**	4	-	-
B	**3**	-	2	**2**	**1**
C	4	2	-	1	2
D	-	**2**	1	-	-
E	-	**1**	2	-	-

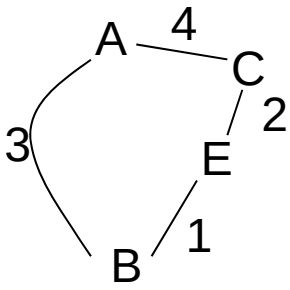


Le plus éloigné est B

(et C, distance 1) Δ=2

MI

	A	B	**C**	D	*E*
A	-	-	**4**	5	-
B	-	-	2	2	-
C	**4**	**2**	-	1	**2**
D	5	2	1	-	3
E	-	-	**2**	3	-

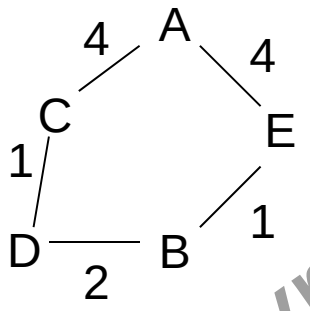


Le meilleur est C

(Δ=2)

PPI

	A	B	C	D	E
A	-	-	-	5	-
B	-	-	-	2	-
C	-	-	-	1	-
D	5	2	1	-	3
E	-	-	-	3	-

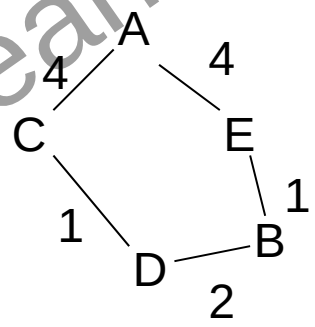


Il ne reste que D

Δ=1

PLI

	A	B	C	D	E
A	-	-	4	-	-
B	-	-	2	-	-
C	4	2	-	1	2
D	-	-	1	-	-
E	-	-	2	-	-

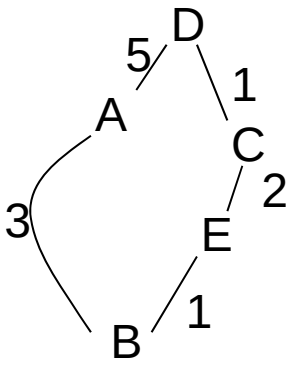


Il ne reste que C

Δ=0

MI

	A	B	C	D	E
A	-	-	-	5	-
B	-	-	-	2	-
C	-	-	-	1	-
D	5	2	1	-	3
E	-	-	-	3	-

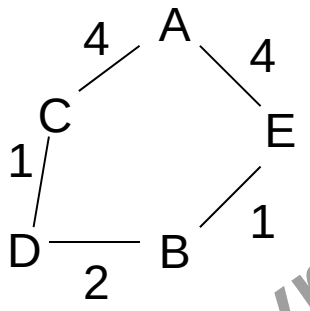


Il ne reste que D

Δ=2

PPI

	A	*B*	*C*	D	*E*
A	-	-	-	5	-
B	-	-	-	2	-
C	-	-	-	1	-
D	5	2	1	-	3
E	-	-	-	3	-

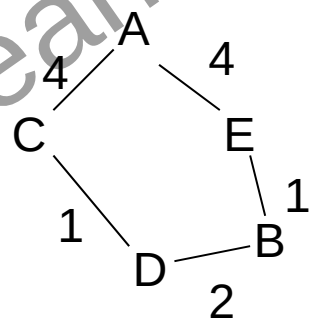


A-E-B-D-C-A, coût 12

cycle hamiltonien

PLI

	A	*B*	C	*D*	*E*
A	-	-	4	-	-
B	-	-	2	-	-
C	4	2	-	1	2
D	-	-	1	-	-
E	-	-	2	-	-

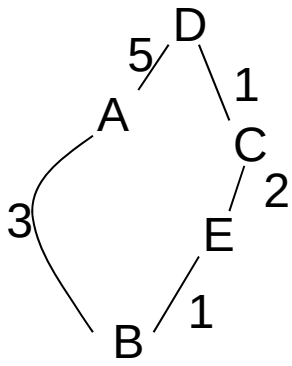


A-E-B-D-C-A, coût 12

cycle hamiltonien

MI

	A	B	*C*	D	*E*
A	-	-	-	5	-
B	-	-	-	2	-
C	-	-	-	1	-
D	5	2	1	-	3
E	-	-	-	3	-



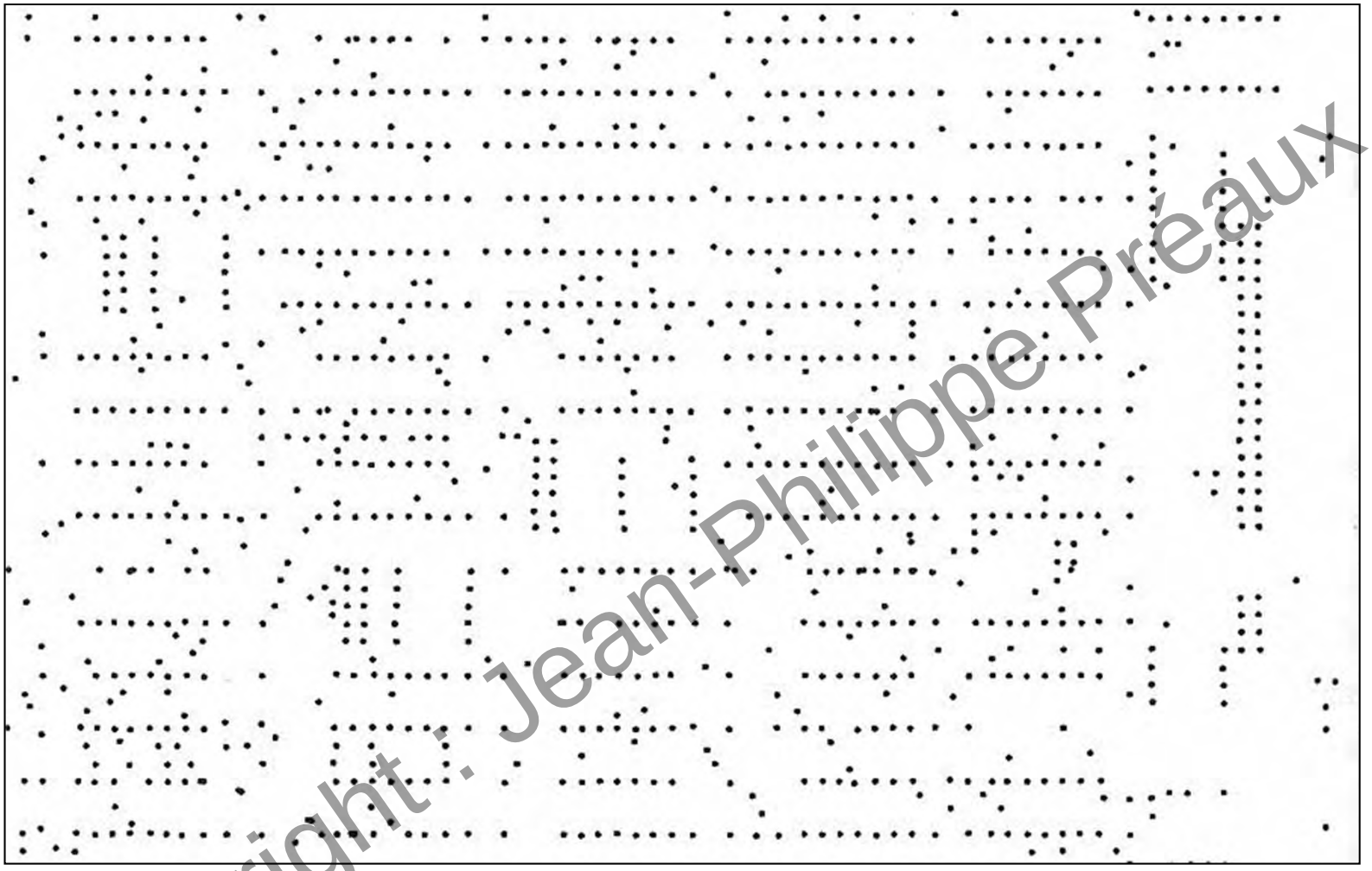
A-D-C-E-B-A, coût 12

cycle hamiltonien

- PLI et PPI s'implémentent en $O(n^2)$; MI en $O(n^3)$. On prouve (à priori) que leur PRP pour le Δ-PVC est au pire 2.

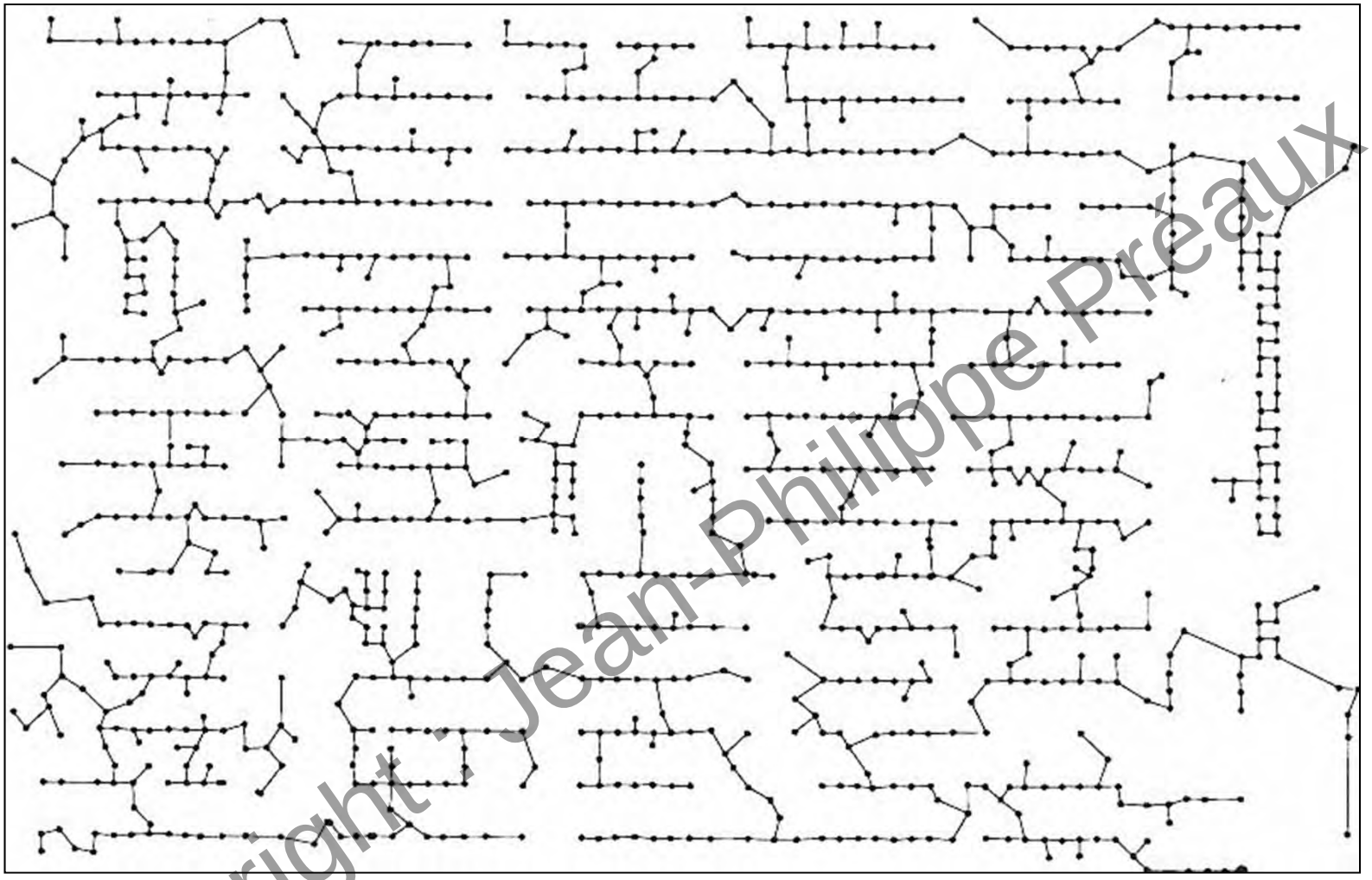
- PLI et PPI s'implémentent en $O(n^2)$; MI en $O(n^3)$. On prouve (à priori) que leur PRP pour le Δ-PVC est au pire 2.
- Leur performance est cependant meilleure en moyenne. En moyenne MI est meilleure que PPI qui est meilleure que PLI, dont la performance moyenne (statistique) est 1.16 (testée sur TSPLIB).

- PLI et PPI s'implémentent en $O(n^2)$; MI en $O(n^3)$. On prouve (à priori) que leur PRP pour le Δ-PVC est au pire 2.
- Leur performance est cependant meilleure en moyenne. En moyenne MI est meilleure que PPI qui est meilleure que PLI, dont la performance moyenne (statistique) est 1.16 (testée sur TSPLIB).
- Pour des PVC euclidiens, étonnement (?..) la meilleure est PLI. Ce qui montre la necessité d'évaluer les heuristiques...



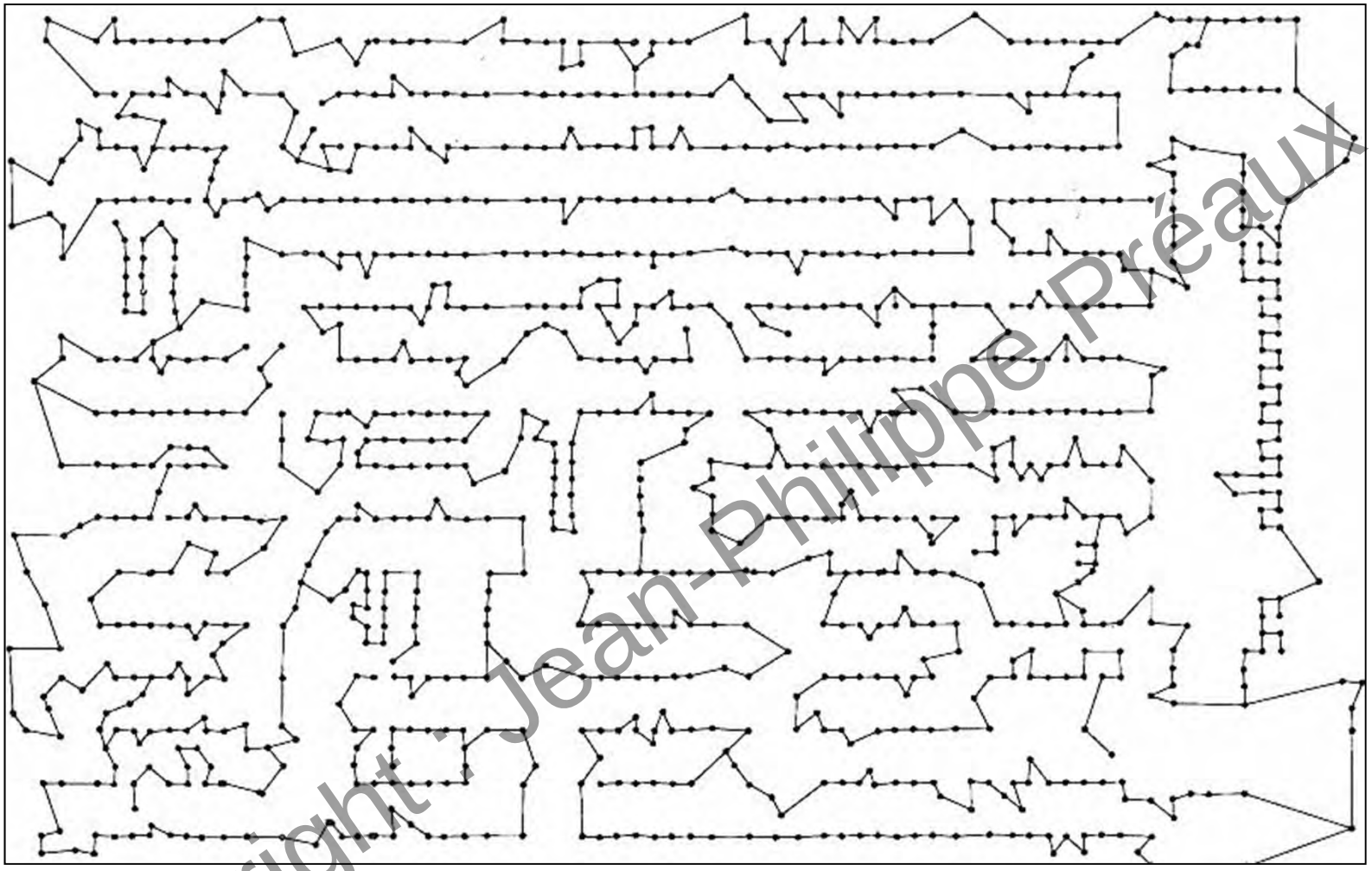
Un PVC euclidien de 1173 sommets.

(Le record (août 1994) est de 7397 sommets.)



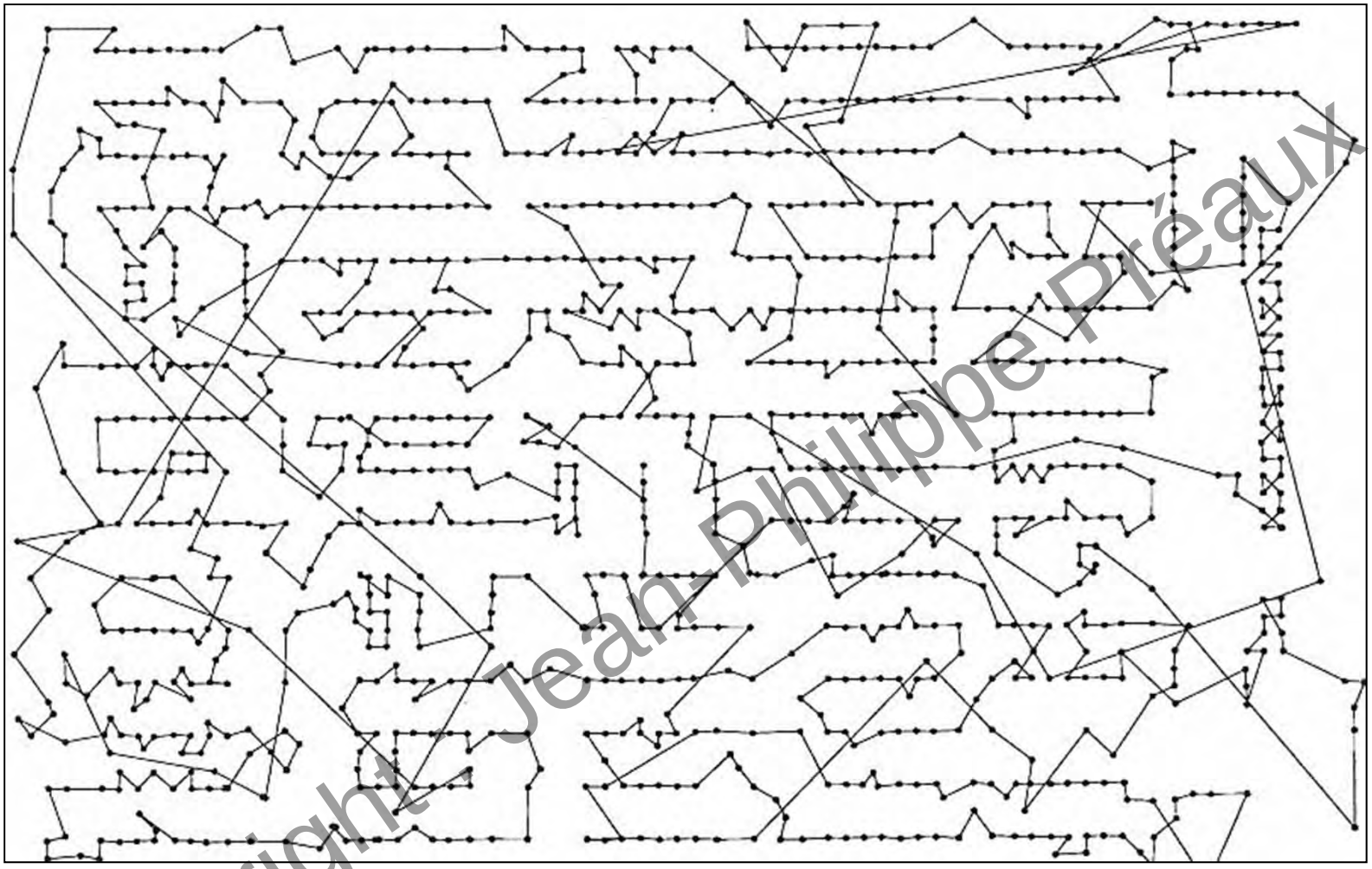
Borne minimale : arbre recouvrant de coût minimal

coût = 51488



Borne minimale plus fine : borne de Held-Karp

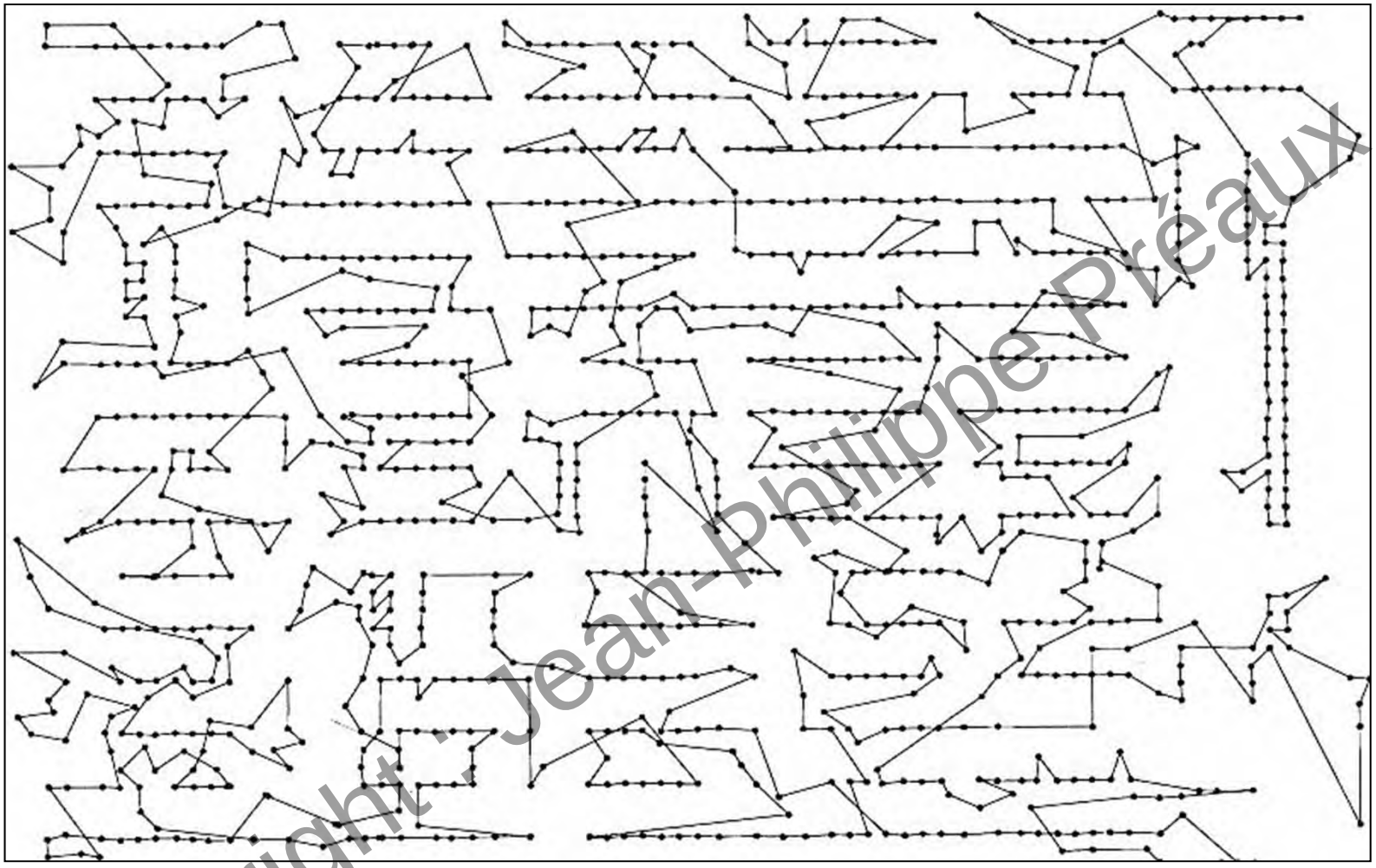
coût = 56349



Heuristique : Plus Proche Voisin

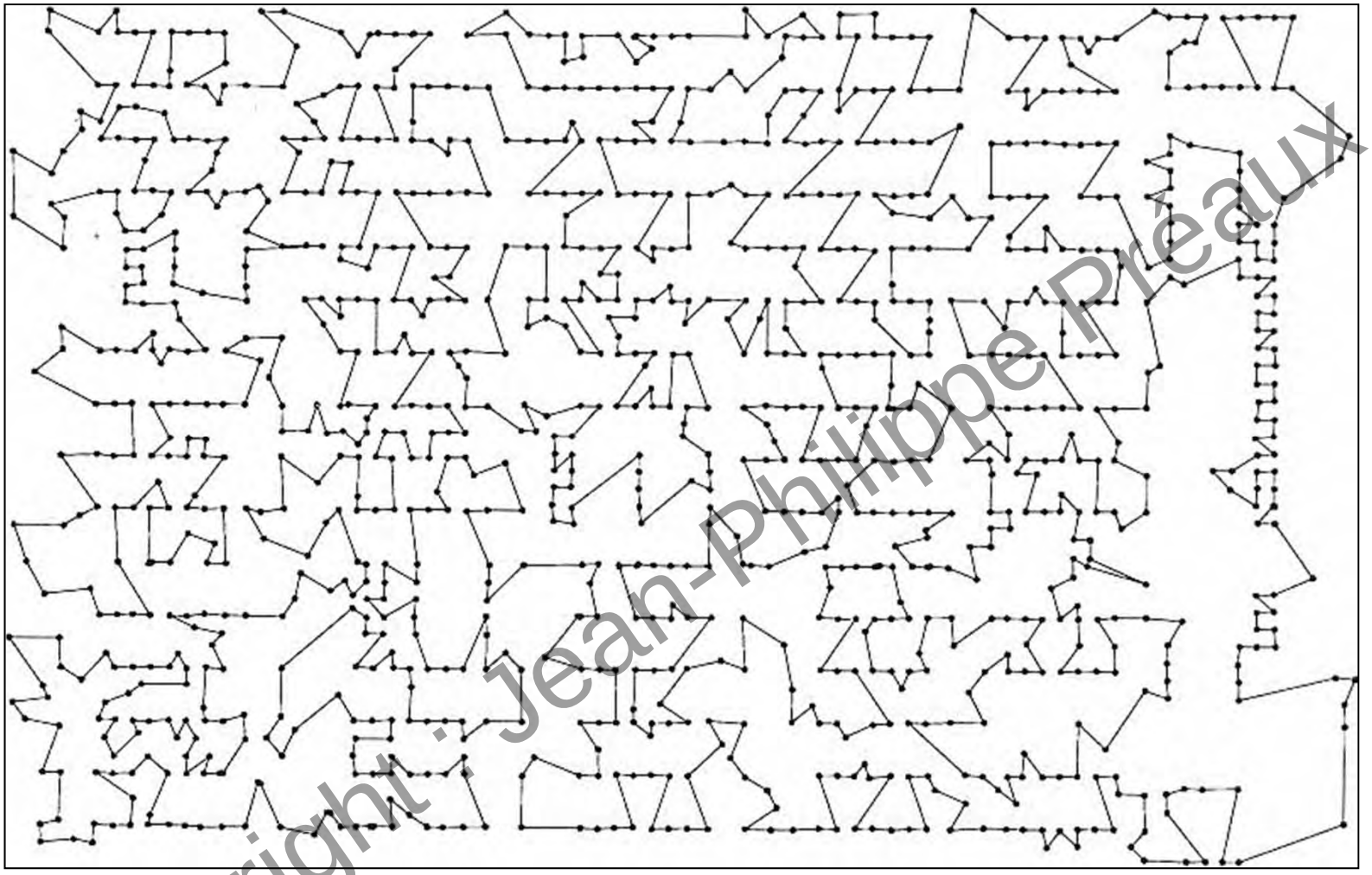
coût = 67822 ; erreur au plus 20%

Remarquer les nombreux « longs trajets »



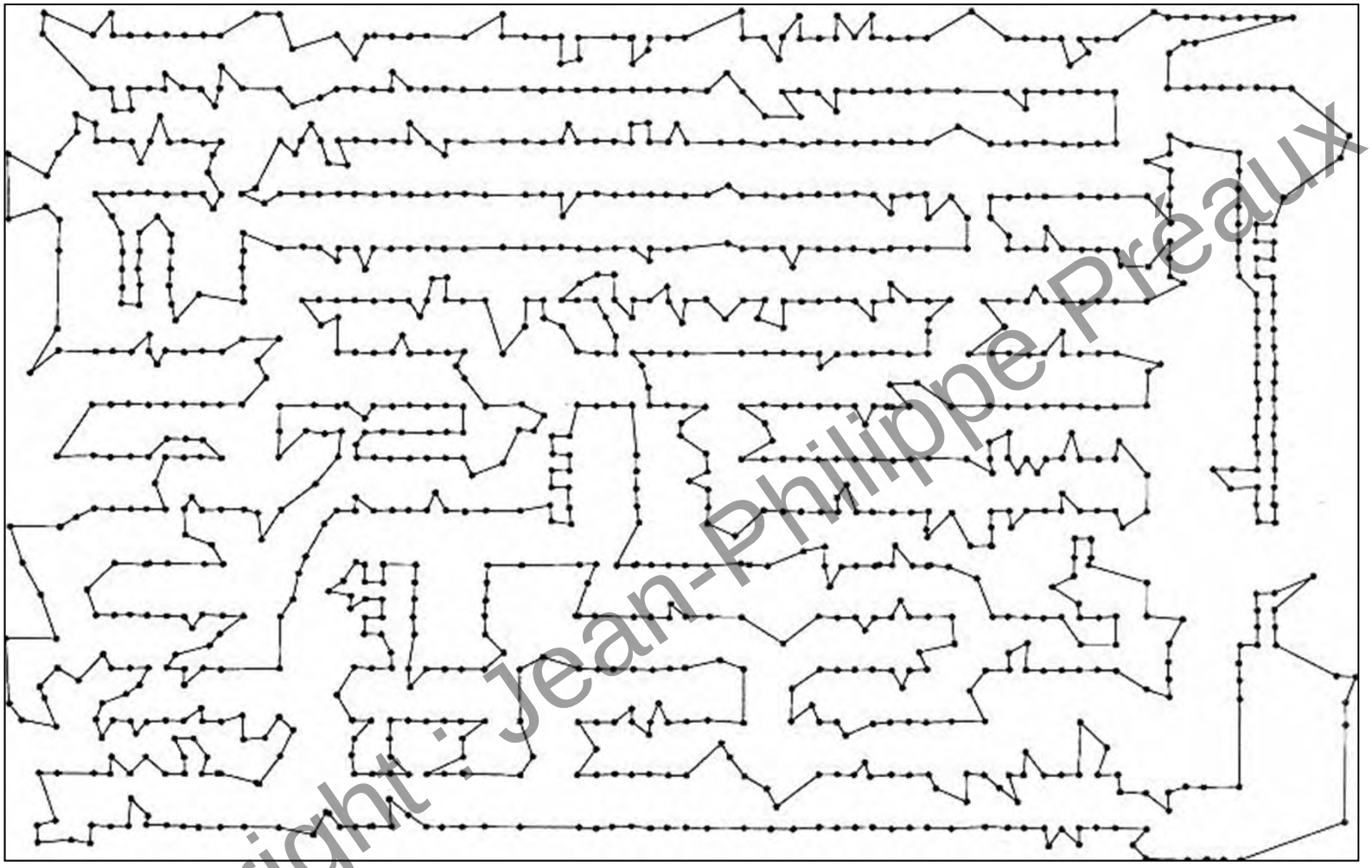
Heuristique : Plus Proche Insertion

coût = 72337 ; erreur au plus 28%



Heuristique : Plus Lointaine Insertion

coût = 65980 ; erreur au plus 17%



Meilleure heuristique connue : Lin-Kernighan
(Recherche locale, d'erreur 1% sur TSPLIB)

coût = 56892 ; erreur au plus 1%

Heuristique de Christofide

C’est l’heuristique de meilleure **PRP=1,5** connue pour le Δ- PVC !

Heuristique de Christofide

- Soit G graphe complet valué pour le Δ-PVC.

Heuristique de Christofide

- Soit G graphe complet valué pour le Δ-PVC.
- Déterminer un arbre recouvrant T de coût minimal.

Heuristique de Christofide

- Soit G graphe complet valué pour le Δ-PVC.
- Déterminer un arbre recouvrant T de coût minimal.
- Soit I les sommets de T de degré impair. Déterminer dans G un couplage C parfait de I de coût minimal.

Heuristique de Christofide

- Soit G graphe complet valué pour le Δ-PVC.
- Déterminer un arbre recouvrant T de coût minimal.
- Soit I les sommets de T de degré impair. Déterminer dans G un couplage C parfait de I de coût minimal.
- Construire un graphe $T_1 \supset T$, avec $T_1 = T \int C$ (ajouter les arêtes de C qui ne sont pas dans T, et dupliquer celles qui sont déjà dans T). Tous les sommets de T_1 sont de degré pair.

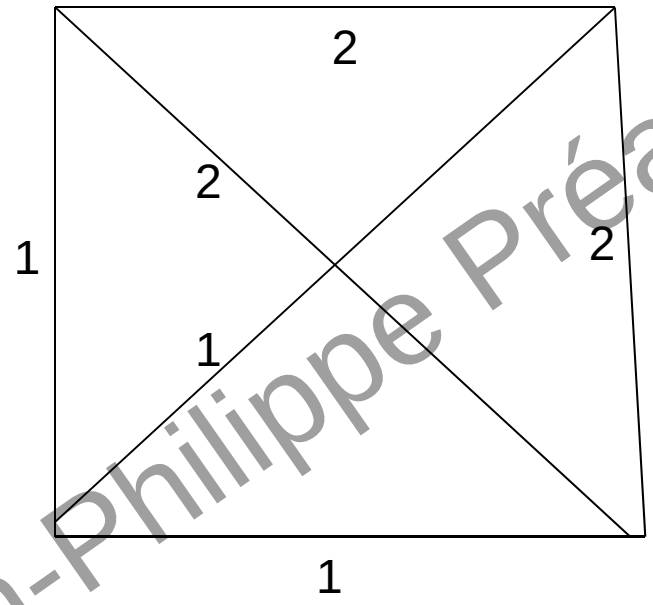
Heuristique de Christofide

- Soit G graphe complet valué pour le Δ-PVC.
- Déterminer un arbre recouvrant T de coût minimal.
- Soit I les sommets de T de degré impair. Déterminer dans G un couplage C parfait de I de coût minimal.
- Construire un graphe $T_1 \supset T$, avec $T_1 = T \int C$ (ajouter les arêtes de C qui ne sont pas dans T, et dupliquer celles qui sont déjà dans T). Tous les sommets de T_1 sont de degré pair.
- Pour chaque sommet u de degré ≥ 4 dans T_1 supprimer des arêtes [v,u] et [u,w] de T_1 et ajouter [v,w] à T_1 en gardant le graphe connexe, (et tant qu'à faire, de surcoût minimal). (« shortcut »)

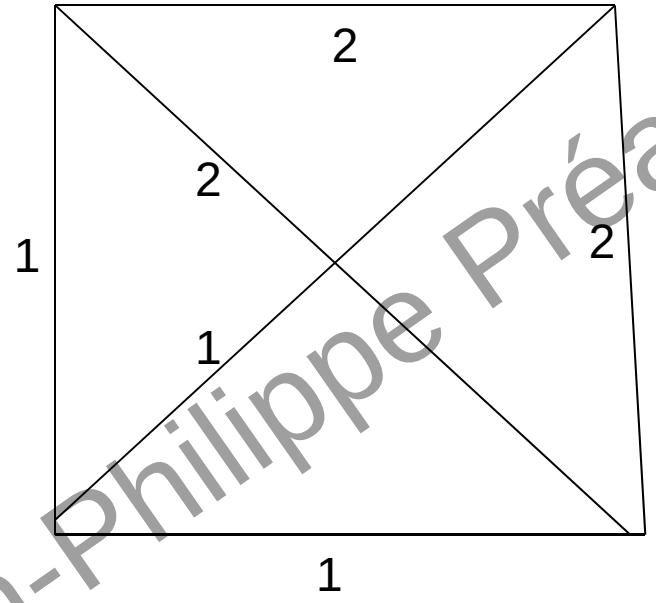
Heuristique de Christofide

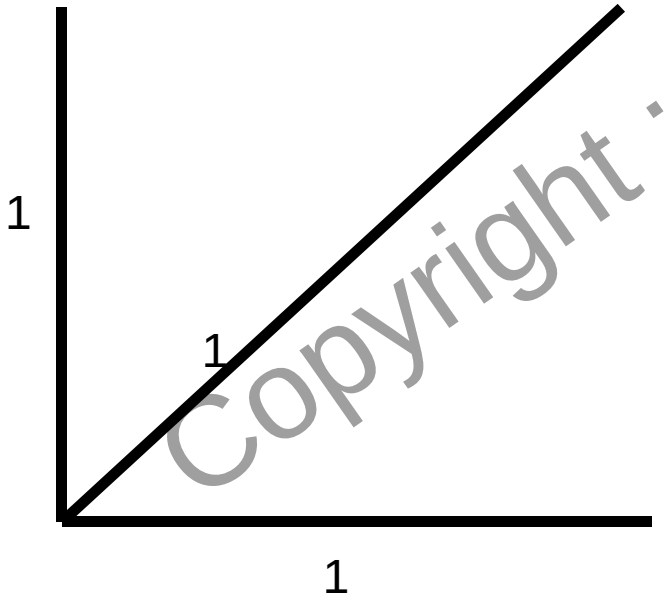
- Soit G graphe complet valué pour le Δ-PVC.
- Déterminer un arbre recouvrant T de coût minimal.
- Soit I les sommets de T de degré impair. Déterminer dans G un couplage C parfait de I de coût minimal.
- Construire un graphe $T_1 \supset T$, avec $T_1 = T \int C$ (ajouter les arêtes de C qui ne sont pas dans T, et dupliquer celles qui sont déjà dans T). Tous les sommets de T_1 sont de degré pair.
- Pour chaque sommet u de degré ≥ 4 dans T_1 supprimer des arêtes [v,u] et [u,w] de T_1 et ajouter [v,w] à T_1 en gardant le graphe connexe, (et tant qu'à faire, de surcoût minimal). (« shortcut »)
- Dès qu'il n'y a plus de sommet de degré ≥ 4, on a construit un cycle hamiltonien.

- **Exemple** ; graphe G :



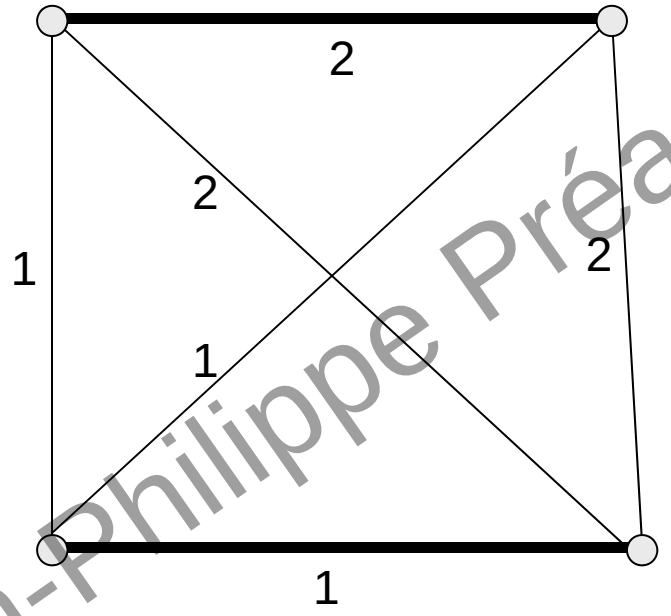
- **Exemple** ; graphe G :

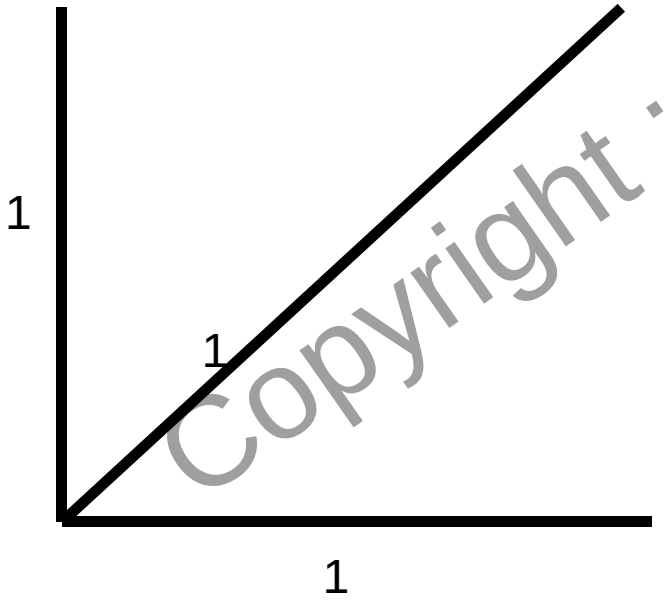




Un arbre T recouvrant de coût minimal 3.

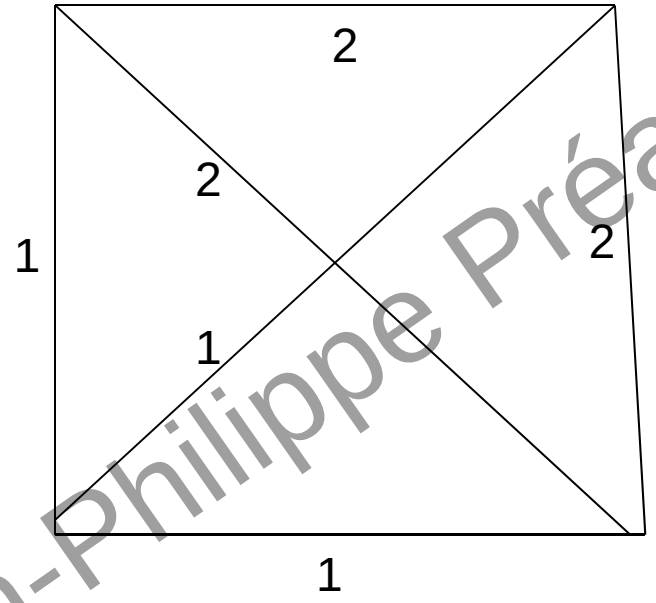
- **Exemple** ; graphe G :

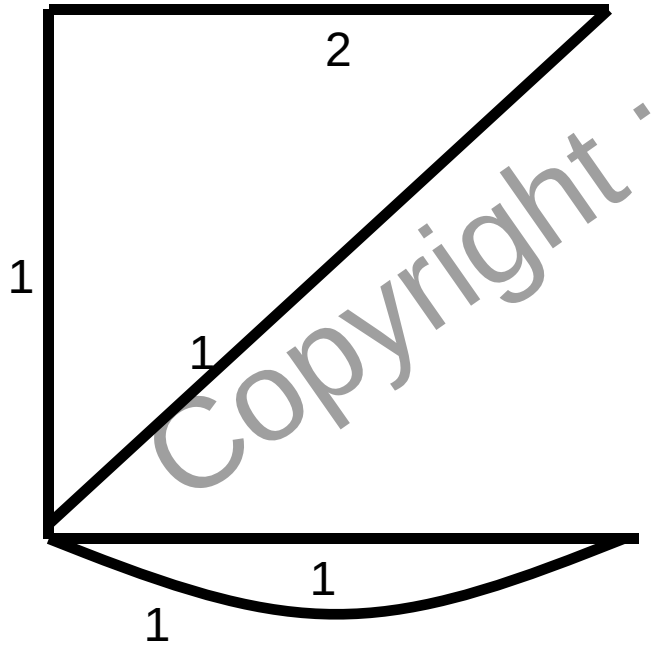




Un couplage parfait de coût minimal des sommets impairs de T.

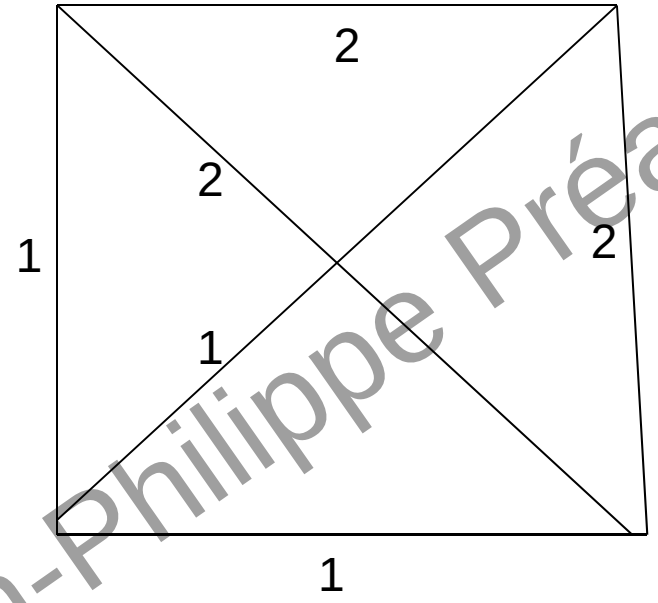
- **Exemple** ; graphe G :

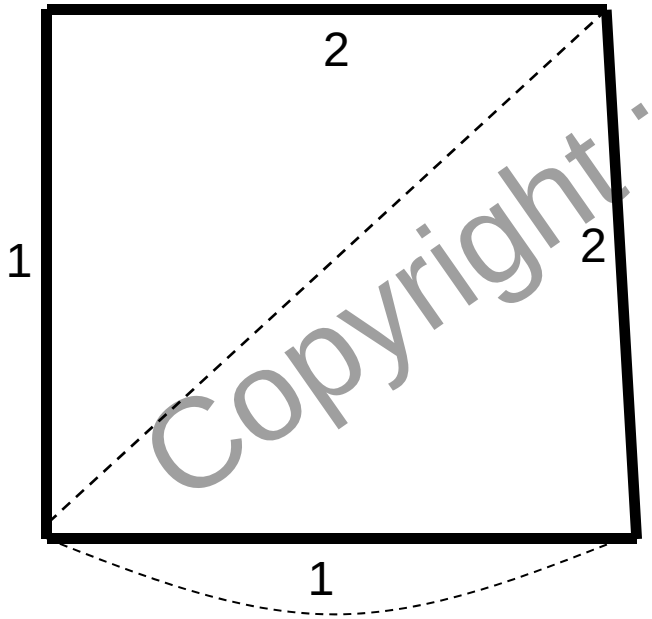




Un surgraphe de T contenant un cycle eulérien de coût minimal 6. Il contient un sommet de degré >2.

- **Exemple** ; graphe G :





Tous les sommets sont de degré 2 après « shortcut »: on obtient un cycle hamiltonien de coût 6.

- **<u>Théorème :</u>** Pour le Δ-PVC, la PRP de l'heuristique de Christofide est ≤ 1.5.

- **Théorème :** Pour le Δ-PVC, la PRP de l'heuristique de Christofide est ≤ 1.5.

Preuve : Soient O une solution au PVC, et H le cycle obtenu.

- **Théorème :** Pour le Δ-PVC, la PRP de l'heuristique de Christofide est ≤ 1.5.

Preuve : Soient O une solution au PVC, et H le cycle obtenu.
L'inégalité triangulaire $\Rightarrow c(H) \leq c(T)+c(C)$.

- **Théorème :** Pour le Δ-PVC, la PRP de l'heuristique de Christofide est ≤ 1.5.

Preuve : Soient O une solution au PVC, et H le cycle obtenu.

L'inégalité triangulaire $\Rightarrow c(H) \leq c(T)+c(C)$.

Or $c(T) \leq c(O)$ (car supprimer une arête de O donne un arbre recouvrant). Il suffit donc de montrer que :

$$c(C) \leq 0.5\ c(O)$$

- **Théorème :** Pour le Δ-PVC, la PRP de l'heuristique de Christofide est ≤ 1.5.

Preuve : Soient O une solution au PVC, et H le cycle obtenu.

L'inégalité triangulaire $\Rightarrow c(H) \leq c(T)+c(C)$.

Or $c(T) \leq c(O)$ (car supprimer une arête de O donne un arbre recouvrant). Il suffit donc de montrer que :

$$c(C) \leq 0.5\ c(O)$$

Soit un circuit U dans G reliant les sommets I impairs de T, par des arêtes et dans l'ordre où ils apparaissent dans O.

$$U=(i_1,i_2)\ (i_2,i_3)...(i_{p-1},i_p)$$

- **Théorème :** Pour le Δ-PVC, la PRP de l'heuristique de Christofide est ≤ 1.5.

Preuve : Soient O une solution au PVC, et H le cycle obtenu.

L'inégalité triangulaire $\Rightarrow c(H) \leq c(T)+c(C)$.

Or $c(T) \leq c(O)$ (car supprimer une arête de O donne un arbre recouvrant). Il suffit donc de montrer que :

$$c(C) \leq 0.5\ c(O)$$

Soit un circuit U dans G reliant les sommets I impairs de T, par des arêtes et dans l'ordre où ils apparaissent dans O.

$$U=(i_1,i_2)\ (i_2,i_3)...(i_{p-1},i_p)$$

L'inégalité triangulaire $\Rightarrow$ chaque arête est géodésique

$\Rightarrow c(U) \leq c(O)$.

- **Théorème :** Pour le Δ-PVC, la PRP de l'heuristique de Christofide est ≤ 1.5.

Preuve : Soient O une solution au PVC, et H le cycle obtenu.

L'inégalité triangulaire $\Rightarrow c(H) \leq c(T)+c(C)$.

Or $c(T) \leq c(O)$ (car supprimer une arête de O donne un arbre recouvrant). Il suffit donc de montrer que :

$$c(C) \leq 0.5\ c(O)$$

Soit un circuit U dans G reliant les sommets I impairs de T, par des arêtes et dans l'ordre où ils apparaissent dans O.

$$U=(i_1,i_2)\ (i_2,i_3)...(i_{p-1},i_p)$$

L'inégalité triangulaire $\Rightarrow$ chaque arête est géodésique

$\Rightarrow c(U) \leq c(O)$. Or U donne deux couplage parfaits de U : (i_1,i_2) ; (i_3,i_4) ; ... et (i_2,i_3) ; (i_4,i_5) ; ...

- **<u>Théorème :</u>** Pour le Δ-PVC, la PRP de l'heuristique de Christofide est ≤ 1.5.

<u>Preuve :</u> Soient O une solution au PVC, et H le cycle obtenu.

L'inégalité triangulaire $\Rightarrow c(H) \leq c(T)+c(C)$.

Or $c(T) \leq c(O)$ (car supprimer une arête de O donne un arbre recouvrant). Il suffit donc de montrer que :

$$c(C) \leq 0.5\ c(O)$$

Soit un circuit U dans G reliant les sommets I impairs de T, par des arêtes et dans l'ordre où ils apparaissent dans O.

$$U=(i_1,i_2)\ (i_2,i_3)...(i_{p-1},i_p)$$

L'inégalité triangulaire $\Rightarrow$ chaque arête est géodésique

$\Rightarrow c(U) \leq c(O)$. Or U donne deux couplage parfaits de U : (i_1,i_2) ; (i_3,i_4) ; ... et (i_2,i_3) ; (i_4,i_5) ; ...

Donc l'un des deux est de coût au plus : 0.5 c(0).

Donc $c(C) \leq 0.5\ c(O)$ puisque c'est un couplage optimal. **cqfd**

- **<u>Théorème :</u>** Pour le Δ-PVC, la PRP de l'heuristique de Christofide est ≤ 1.5.

  Sa complexité est en $O(n^3)$.

<u>Preuve :</u> Soient O une solution au PVC, et H le cycle obtenu.

L'inégalité triangulaire $\Rightarrow c(H) \leq c(T)+c(C)$.

Or $c(T) \leq c(O)$ (car supprimer une arête de O donne un arbre recouvrant). Il suffit donc de montrer que :

$$c(C) \leq 0.5\ c(O)$$

Soit un circuit U dans G reliant les sommets I impairs de T, par des arêtes et dans l'ordre où ils apparaissent dans O.

$$U=(i_1,i_2)\ (i_2,i_3)...(i_{p-1},i_p)$$

L'inégalité triangulaire $\Rightarrow$ chaque arête est géodésique

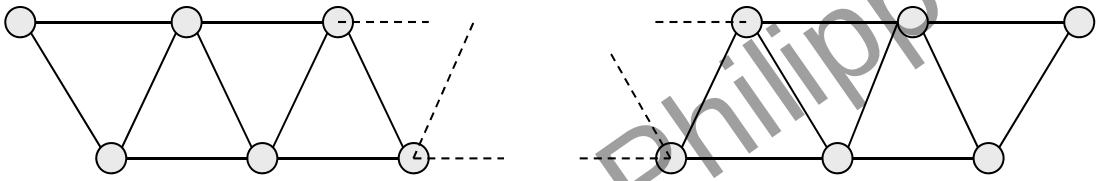
$\Rightarrow c(U) \leq c(O)$. Or U donne deux couplage parfaits de U : (i_1,i_2) ; (i_3,i_4) ; ... et (i_2,i_3) ; (i_4,i_5) ; ...

Donc l'un des deux est de coût au plus : 0.5 c(0).

Donc $c(C) \leq 0.5\ c(O)$ puisque c'est un couplage optimal. **cqfd**

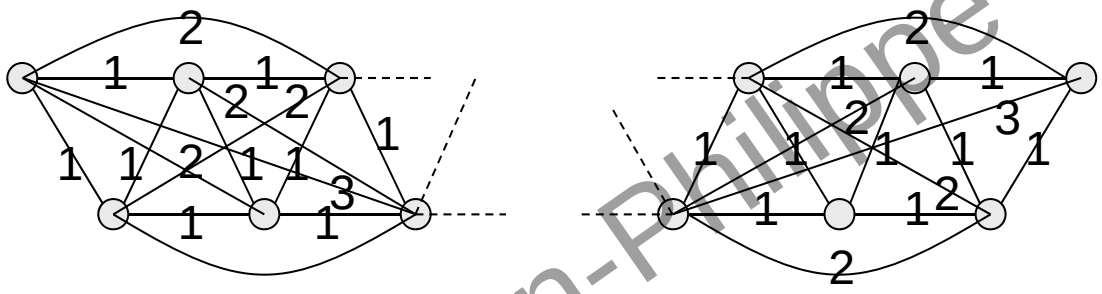
PRP(Christofide) = 1,5

- Considérer un graphe complet à n sommets dont les distances sont celles dans le graphe suivant :



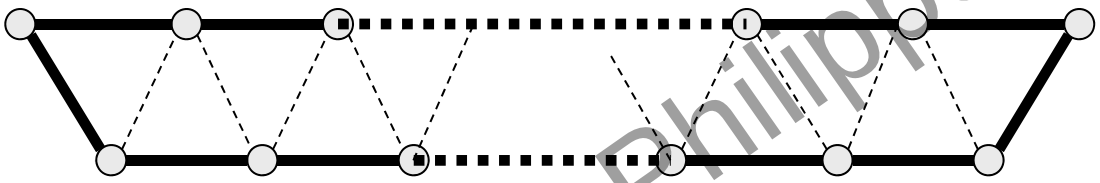
PRP(Christofide) = 1,5

- Considérer un graphe complet à n sommets dont les distances sont celles dans le graphe suivant :



PRP(Christofide) = 1,5

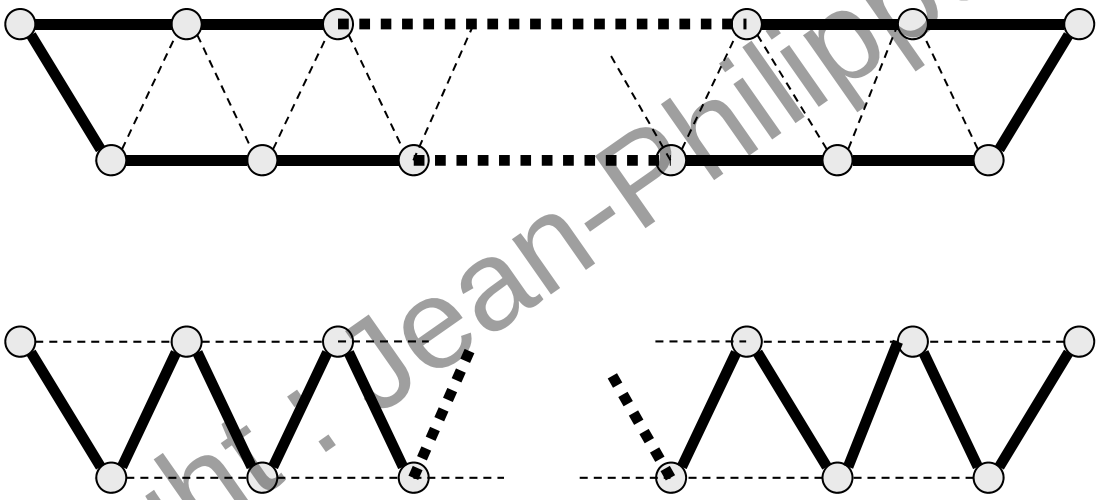
- Considérer un graphe complet à n sommets dont les distances sont celles dans le graphe suivant :



- Un cycle hamiltonien optimal a pour coût = n.

PRP(Christofide) = 1,5

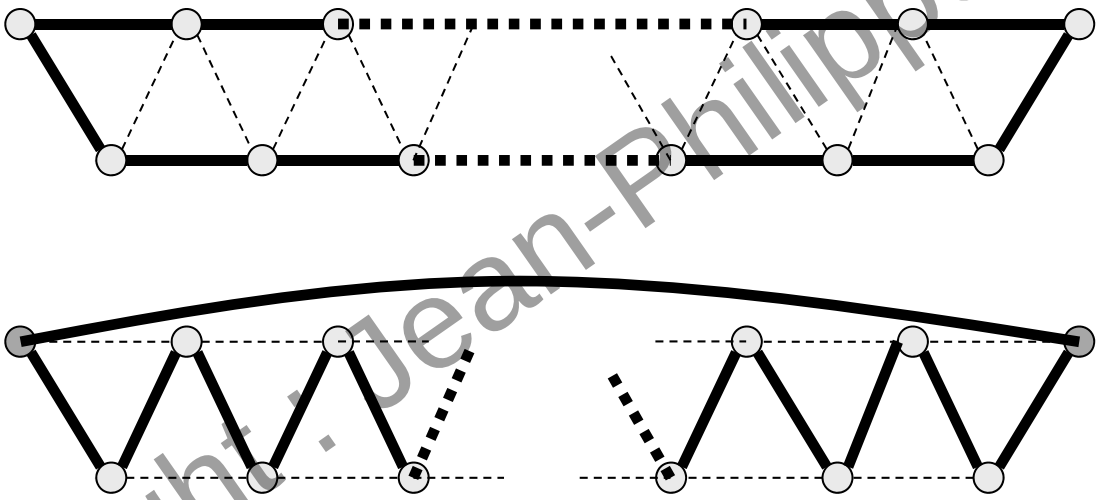
- Considérer un graphe complet à n sommets dont les distances sont celles dans le graphe suivant :



- Un cycle hamiltonien optimal a pour coût = n.
- Un arbre recouvrant de coût minimal = n-1.

PRP(Christofide) = 1,5

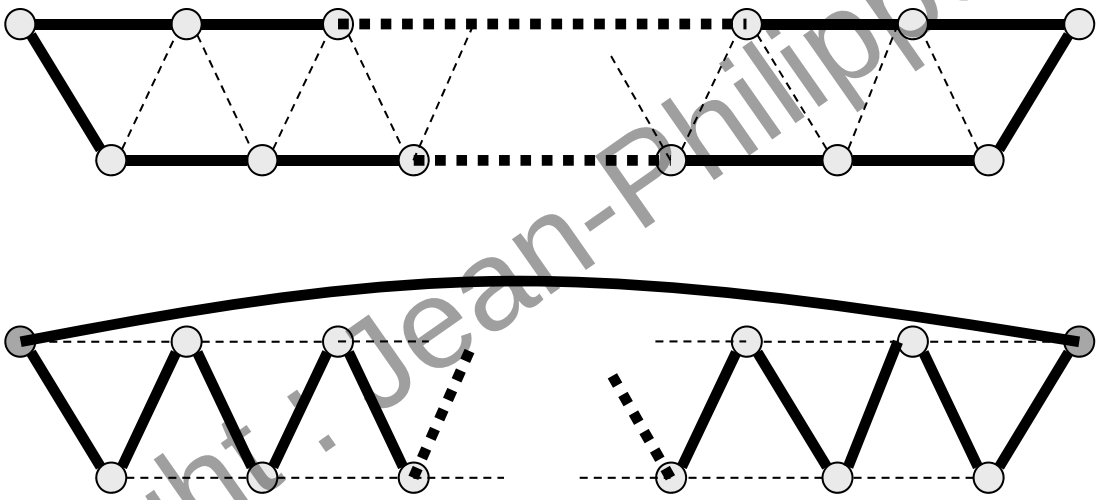
- Considérer un graphe complet à n sommets dont les distances sont celles dans le graphe suivant :



- Un cycle hamiltonien optimal a pour coût = n.
- Un arbre recouvrant de coût minimal = n-1.
- Christofide construit un cycle de coût n-1+(n-1)/2.

PRP(Christofide) = 1,5

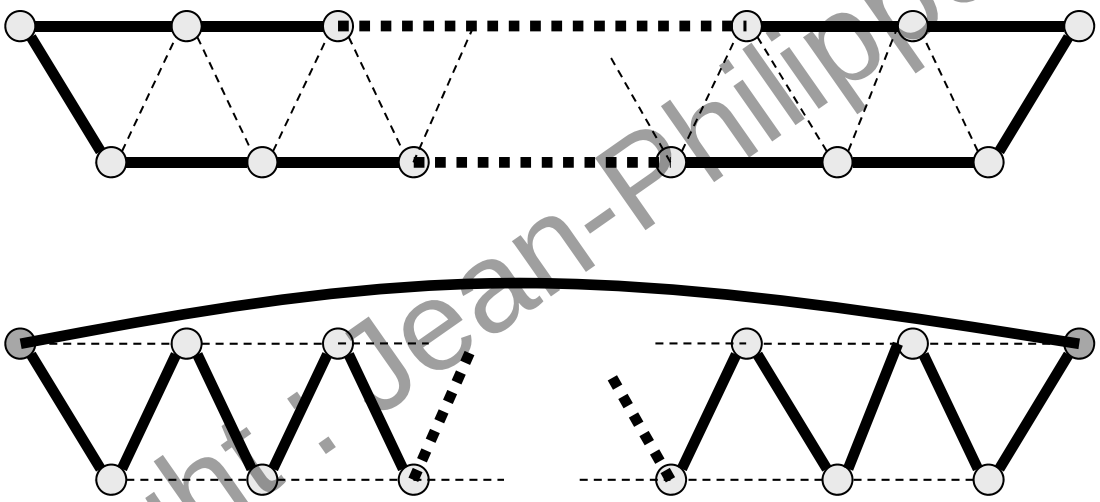
- Considérer un graphe complet à n sommets dont les distances sont celles dans le graphe suivant :



- Un cycle hamiltonien optimal a pour coût = n.
- Un arbre recouvrant de coût minimal = n-1.
- Christofide construit un cycle de coût n-1+(n-1)/2.
- Soit une R_C=(3n-3)/2n

PRP(Christofide) = 1,5

- Considérer un graphe complet à n sommets dont les distances sont celles dans le graphe suivant :



- Un cycle hamiltonien optimal a pour coût = n.
- Un arbre recouvrant de coût minimal = n-1.
- Christofide construit un cycle de coût n-1+(n-1)/2.
- Soit une $R_C=(3n-3)/2n \rightarrow 3/2 \Rightarrow P_C=1,5$.

Heuristiques de Recherches Locales

Recherches locales

- Pour un problème donné soit S l'ensemble des solutions réalisables (les cycles hamiltoniens pour le PVC, les colorations réalisables pour une coloration, etc..).

Recherches locales

- Pour un problème donné soit S l'ensemble des solutions réalisables (les cycles hamiltoniens pour le PVC, les colorations réalisables pour une coloration, etc..).
- On se donne un procédé pour construire pour s $\in$ S un voisinage V(s) dans S de s.

Recherches locales

- Pour un problème donné soit S l'ensemble des solutions réalisables (les cycles hamiltoniens pour le PVC, les colorations réalisables pour une coloration, etc..).
- On se donne un procédé pour construire pour s $\in$ S un voisinage V(s) dans S de s.
- On cherche dans V(s) une solution s' de meilleur coût que s.

Recherches locales

- Pour un problème donné soit S l'ensemble des solutions réalisables (les cycles hamiltoniens pour le PVC, les colorations réalisables pour une coloration, etc..).
- On se donne un procédé pour construire pour s $\in$ S un voisinage V(s) dans S de s.
- On cherche dans V(s) une solution s' de meilleur coût que s.
- On poursuit l'algorithme avec s' en construisant V(s').

Recherches locales

- Pour un problème donné soit S l'ensemble des solutions réalisables (les cycles hamiltoniens pour le PVC, les colorations réalisables pour une coloration, etc..).
- On se donne un procédé pour construire pour s $\in$ S un voisinage V(s) dans S de s.
- On cherche dans V(s) une solution s' de meilleur coût que s.
- On poursuit l'algorithme avec s' en construisant V(s').
- On continue tant que l'on améliore le coût et que l'on ne dépasse pas une limite préétablie de durée d'implémentation .

Recherches locales

- La solution initiale peut être construite au hasard, ou mieux, construite à l'aide d'une heuristique gloutonne.

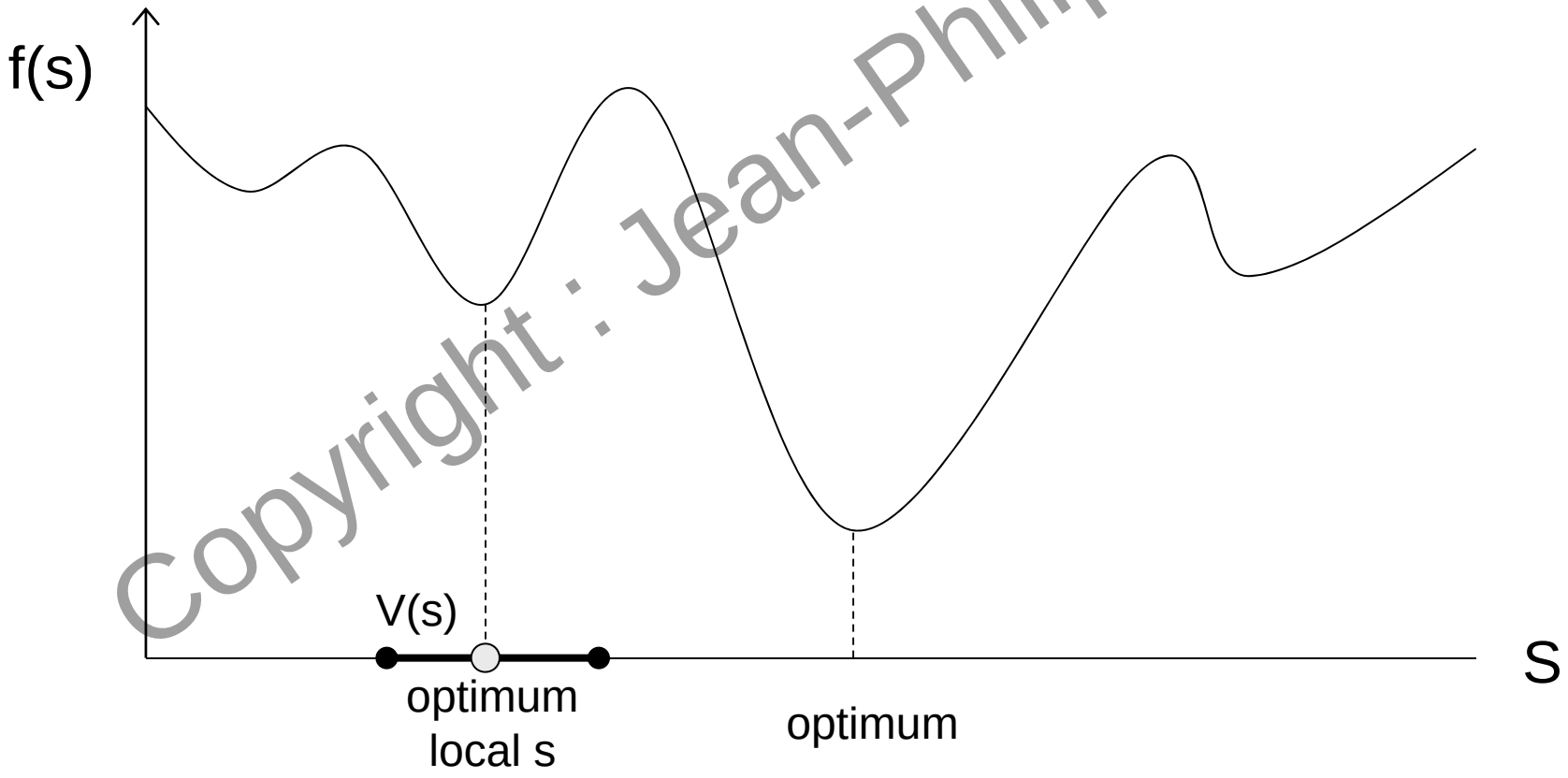
Recherches locales

- La solution initiale peut être construite au hasard, ou mieux, construite à l'aide d'une heuristique gloutonne.
- En général on se donne le voisinage implicitement, comme obtenu à l'aide de petites transformations, et on ne l'explore pas intégralement : on va sur la première amélioration trouvée.

Recherches locales

- La solution initiale peut être construite au hasard, ou mieux, construite à l'aide d'une heuristique gloutonne.
- En général on se donne le voisinage implicitement, comme obtenu à l'aide de petites transformations, et on ne l'explore pas intégralement : on va sur la première amélioration trouvée.
- Il est important que V(s) soit suffisamment petit pour que la recherche soit rapide et suffisamment grand pour avoir une chance d'améliorer le coût. Sur n éléments V(s) doit comporter au plus $O(n^2)$, $O(n^3)$ éléments.

- Lorsque la recherche s'arrête sans trouver de solution améliorante, on n'est pas nécessairement sur un optimum, mais sur un optimum local au sens de la topologie définie sur S par le choix des voisinages :



- Exemple : problème du sac à dos 0-1:
- Une solution réalisable s est un sous-ensemble de l'ensemble des objets avec $\sum_s p_k \leq P$.

- Exemple : problème du sac à dos 0-1:
- Une solution réalisable s est un sous-ensemble de l'ensemble des objets avec $\sum_s p_k \leq P$.
- V(s) : ensemble des solutions obtenues à partir de s en remplaçant dans s un objet quelconque de poids p_i, par un objet de poids au plus $P + p_i - \sum_s p_k$.

- Exemple : problème du sac à dos 0-1:
- Une solution réalisable s est un sous-ensemble de l'ensemble des objets avec $\sum_s p_k \leq P$.
- V(s) : ensemble des solutions obtenues à partir de s en remplaçant dans s un objet quelconque de poids p_i, par un objet de poids au plus $P + p_i - \sum_s p_k$.
- Si la valeur de cet objet est > v_i on a amélioré l'objectif.

- Exemple : problème du sac à dos 0-1:
- Une solution réalisable s est un sous-ensemble de l'ensemble des objets avec $\sum_s p_k \leq P$.
- V(s) : ensemble des solutions obtenues à partir de s en remplaçant dans s un objet quelconque de poids p_i, par un objet de poids au plus $P + p_i - \sum_s p_k$.
- Si la valeur de cet objet est > v_i on a amélioré l'objectif.
- Avec n=2, $p_1=v_1=1$, $p_2=v_2=P$. Si s={objet$_1$}, alors V(s)={objet$_1$,objet$_2$}, et on améliore l'objectif en remplaçant l'objet$_1$ par l'objet$_2$.

- **Exemple :**
- Problème de coloration : une solution est une coloration réalisable.
- Il est délicat de déformer une coloration en la gardant réalisable.

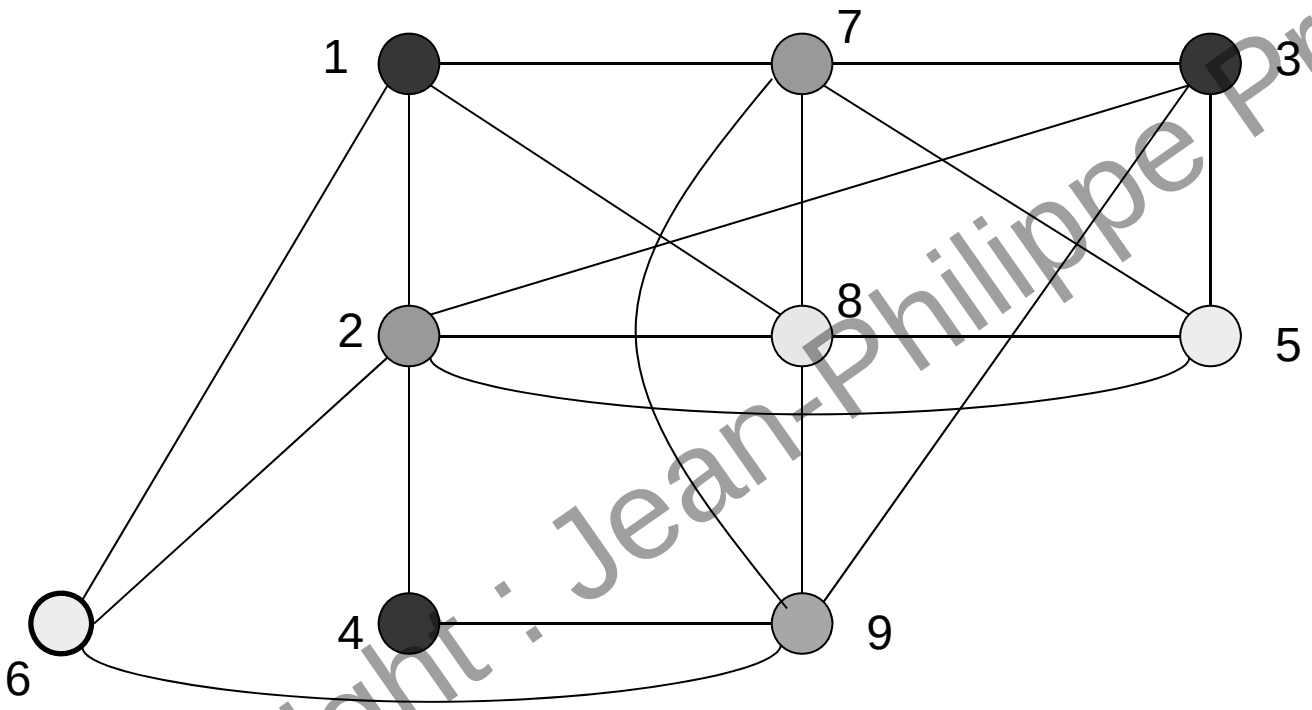
- **Exemple :**
- Problème de coloration : une solution est une coloration réalisable.
- Il est délicat de déformer une coloration en la gardant réalisable.
- On peut par contre modifier facilement un ordre : pour s une coloration réalisable obtenu par une des heuristiques séquentielles grâce à un ordre des sommets :

- **Exemple :**
- Problème de coloration : une solution est une coloration réalisable.
- Il est délicat de déformer une coloration en la gardant réalisable.
- On peut par contre modifier facilement un ordre : pour s une coloration réalisable obtenu par une des heuristiques séquentielles grâce à un ordre des sommets :
- Définir V(s) comme les colorations obtenues en perturbant l'ordre par une permutation de 2 sommets ; il y en a $O(n^2)$:

$$s_1,\ldots,s_i,\ldots,s_k,\ldots,s_n$$

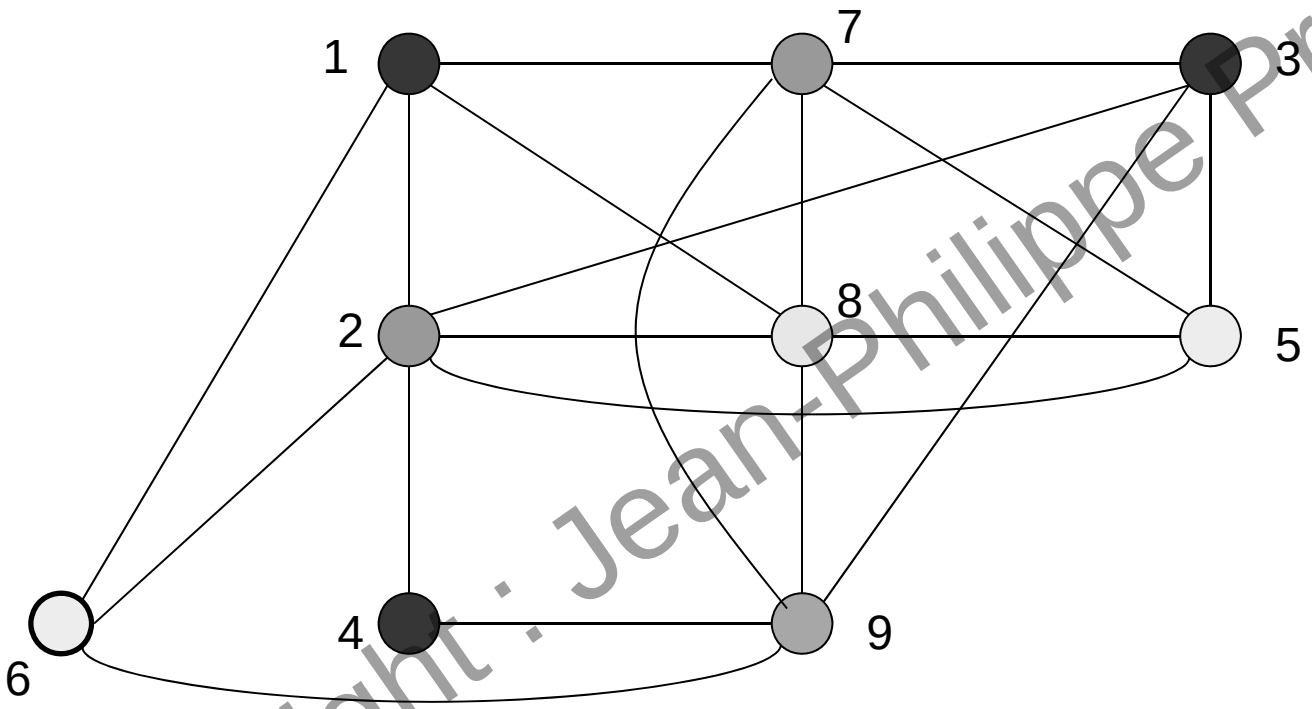
$$s_1,\ldots,s_k,\ldots,s_i,\ldots,s_n$$

- Avec FFS : 5 couleurs



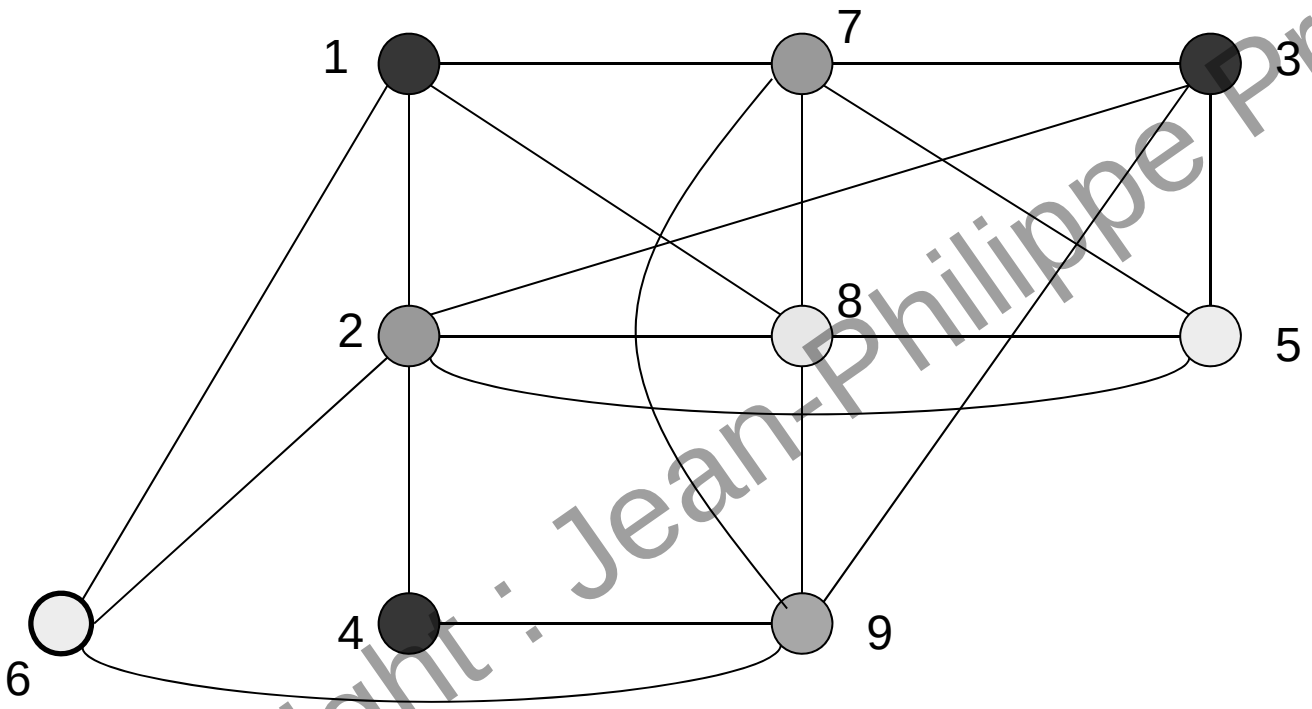
1 2 3 4 5 6 7

- s = coloration suivante avec 5 couleurs obtenue par FFS sur l'ordre donné :



1 2 3 4 5 6 7

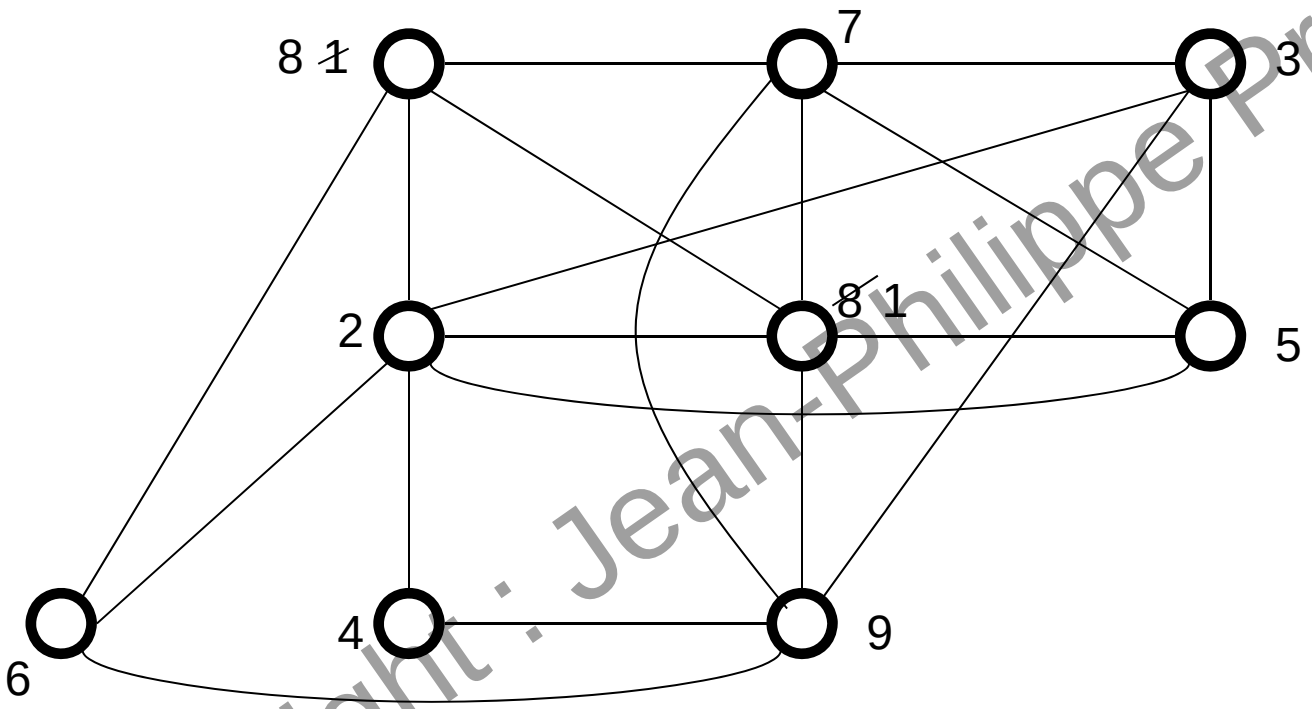
- s = coloration suivante avec 5 couleurs :



<u>Voisinage de s</u> : Coloration obtenue par FFS après permutation de la position de 2 sommets dans l'ordre donné.

1 2 3 4 5 6 7

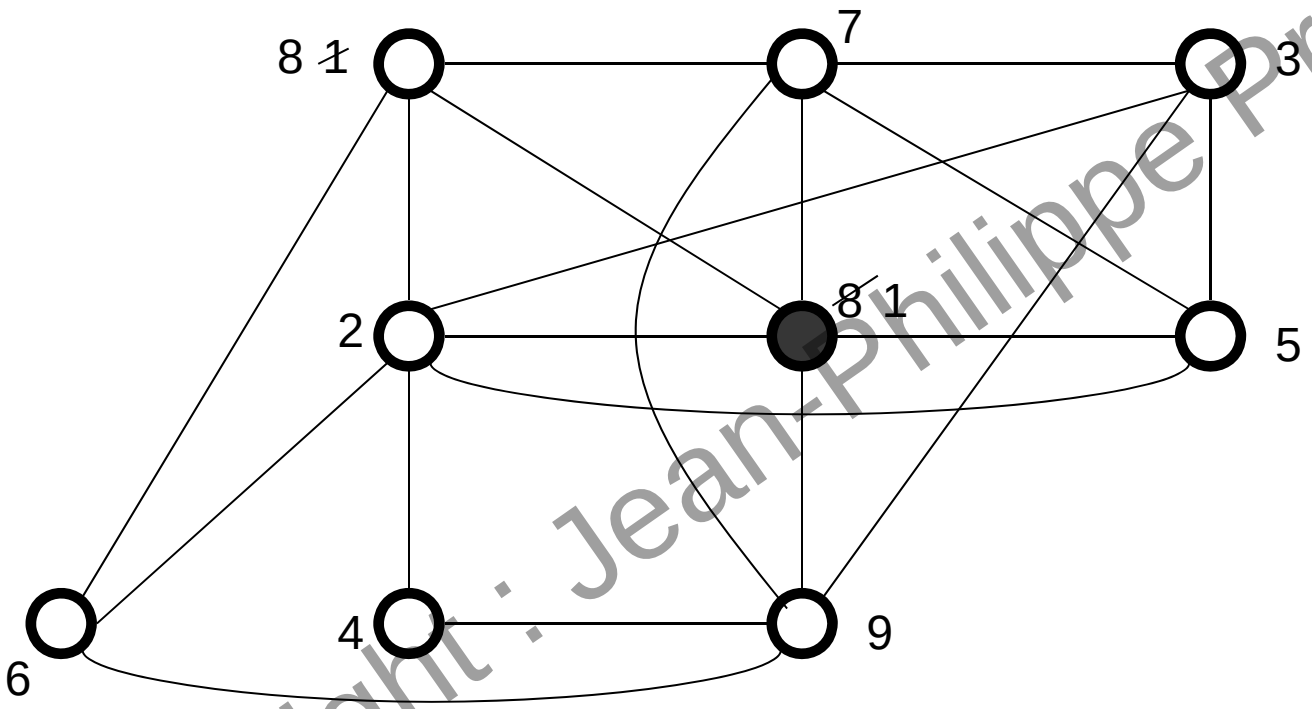
- s = coloration suivante avec 5 couleurs :



s' : Coloration dans V(s) obtenue par FFS après permutation des sommets 1 et 8.

1 2 3 4 5 6 7

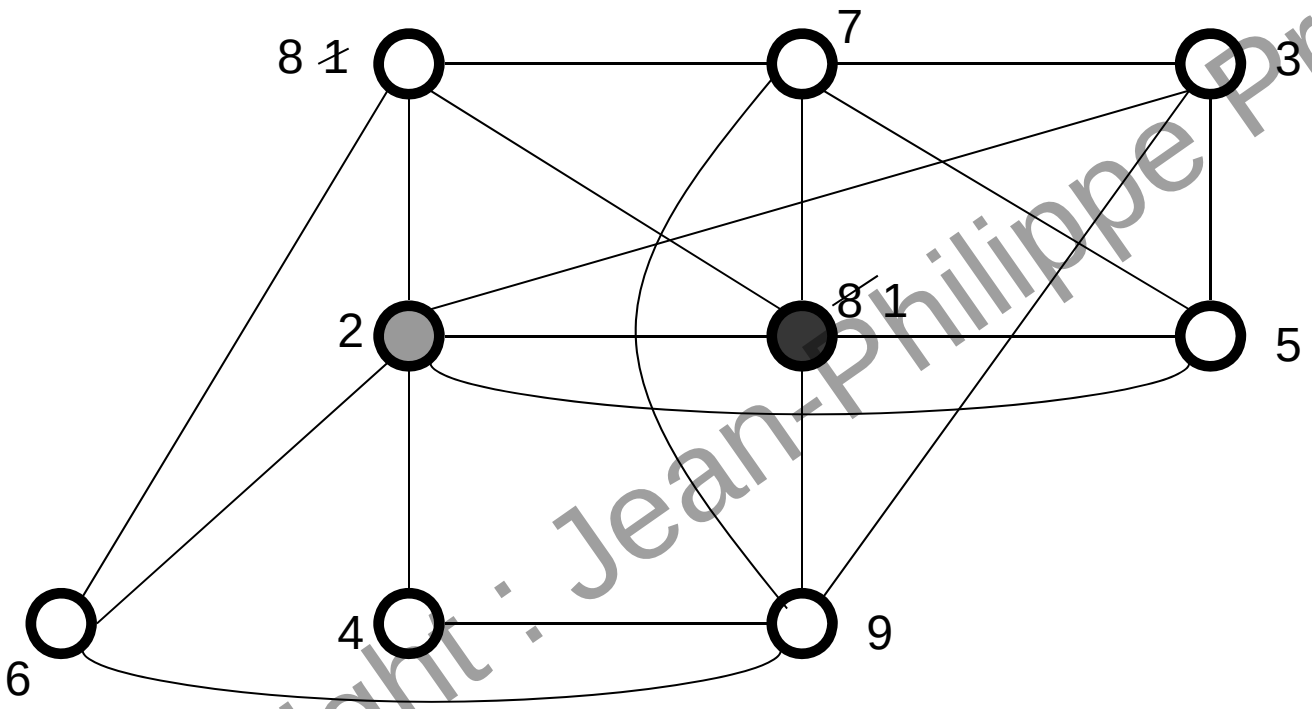
- s = coloration suivante avec 5 couleurs :



s' : Coloration dans V(s) obtenue par FFS après permutation des sommets 1 et 8.

1 2 3 4 5 6 7

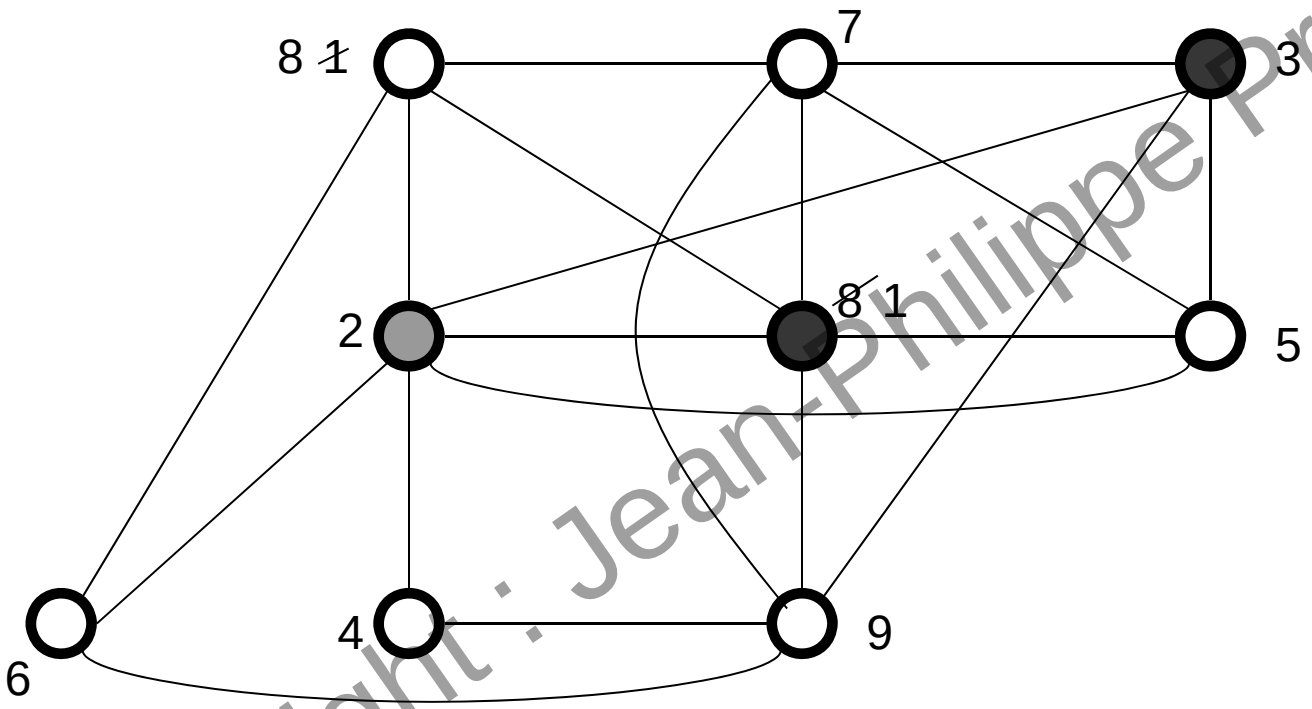
- s = coloration suivante avec 5 couleurs :



s' : Coloration dans V(s) obtenue par FFS après permutation des sommets 1 et 8.

1 2 3 4 5 6 7

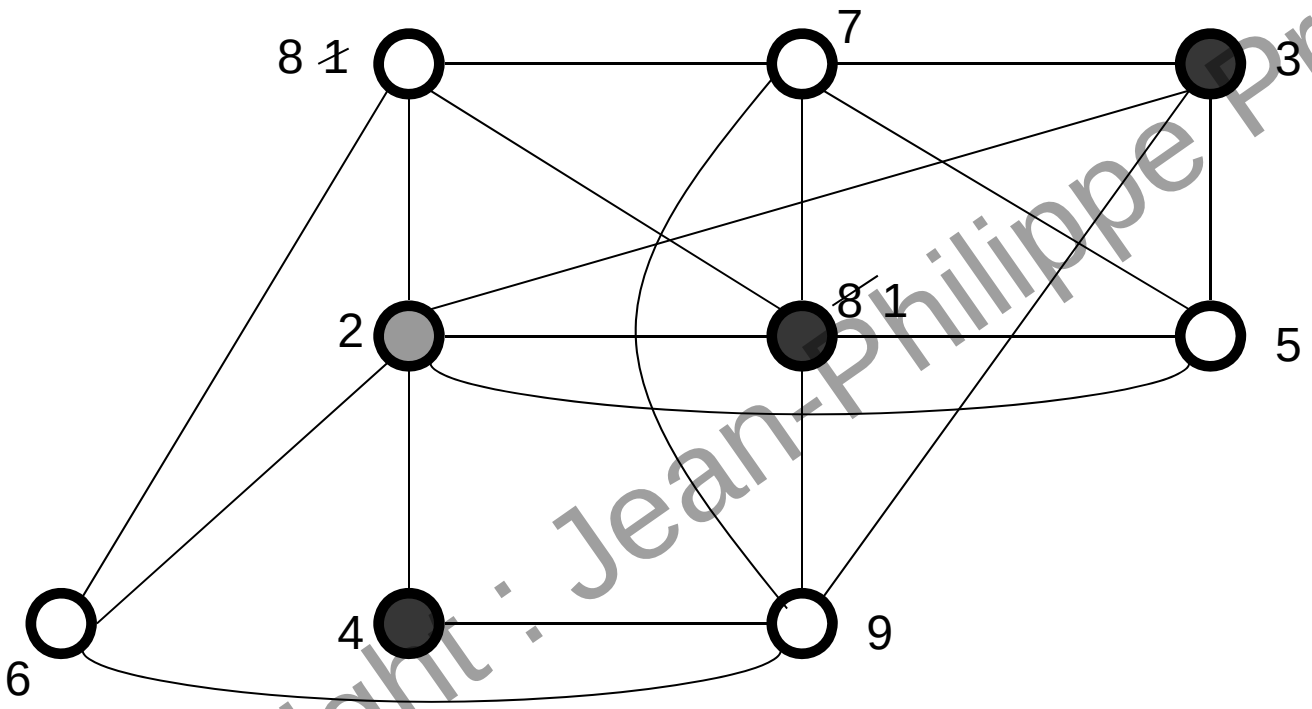
- s = coloration suivante avec 5 couleurs :



s' : Coloration dans V(s) obtenue par FFS après permutation des sommets 1 et 8.

1 2 3 4 5 6 7

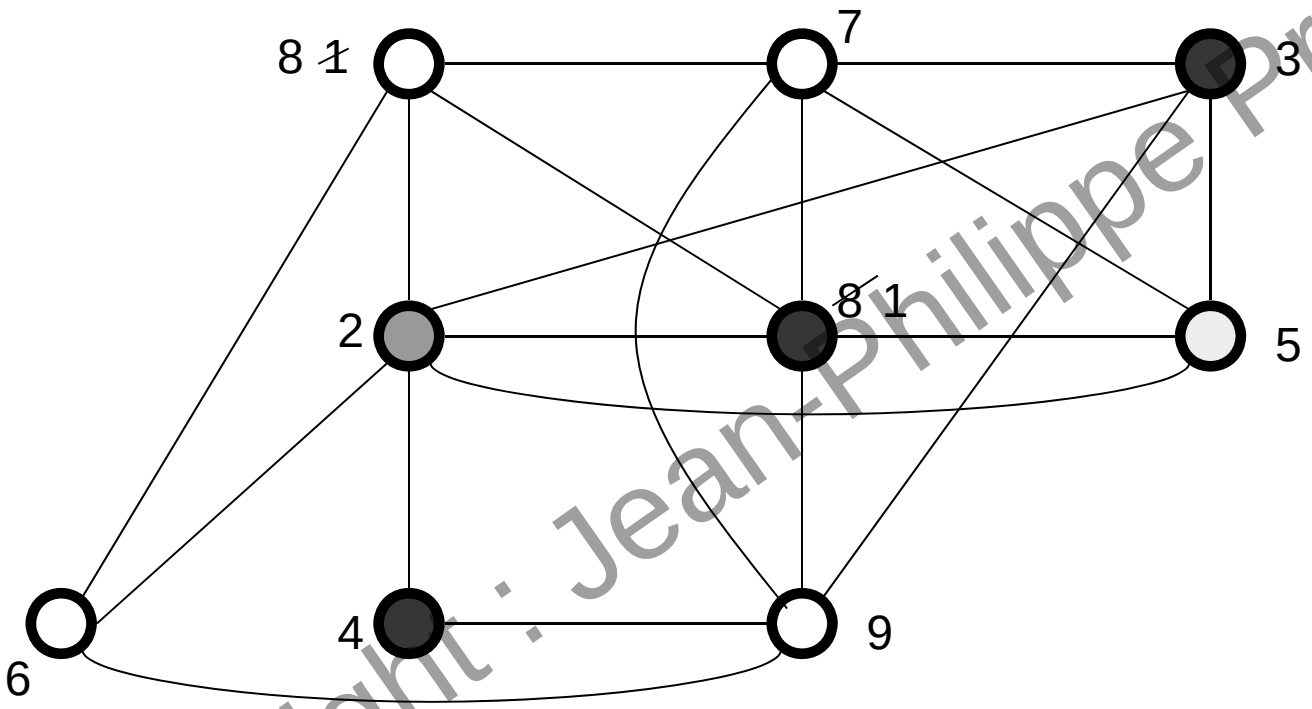
- s = coloration suivante avec 5 couleurs :



s' : Coloration dans V(s) obtenue par FFS après permutation des sommets 1 et 8.

1 2 3 4 5 6 7

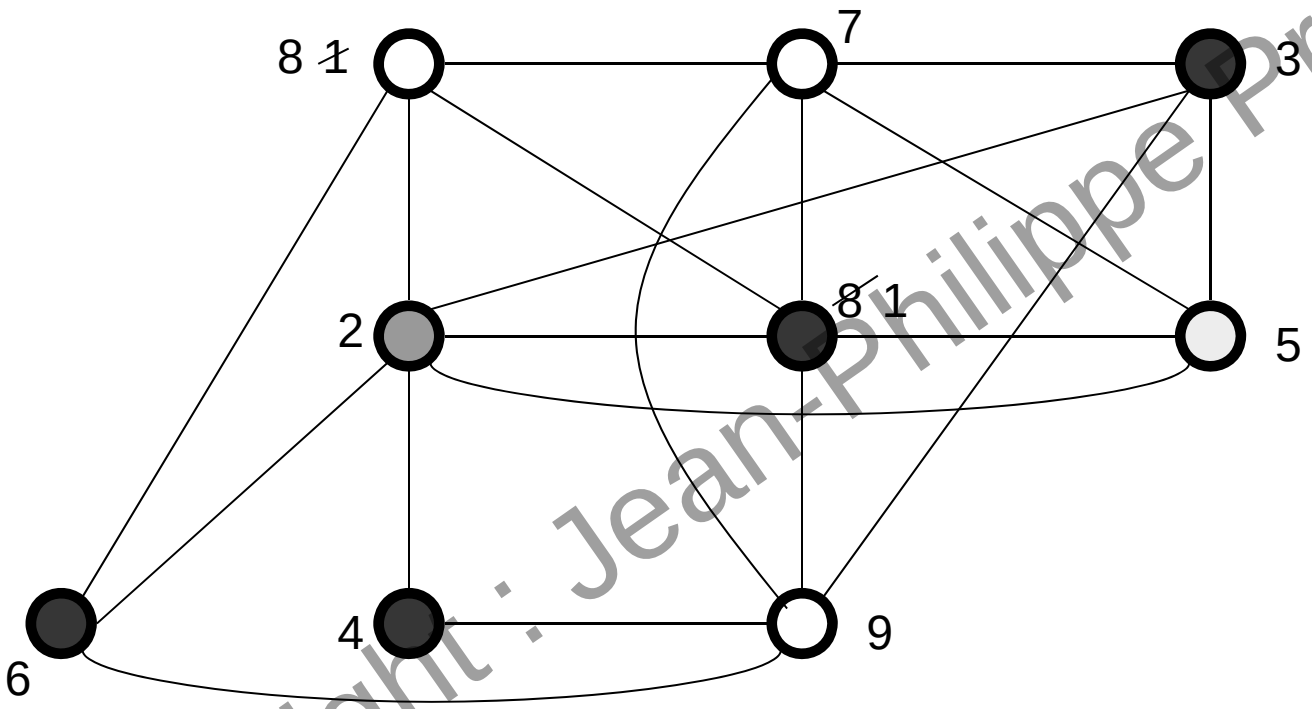
- s = coloration suivante avec 5 couleurs :



s’ : Coloration dans V(s) obtenue par FFS après permutation des sommets 1 et 8.

1 2 3 4 5 6 7

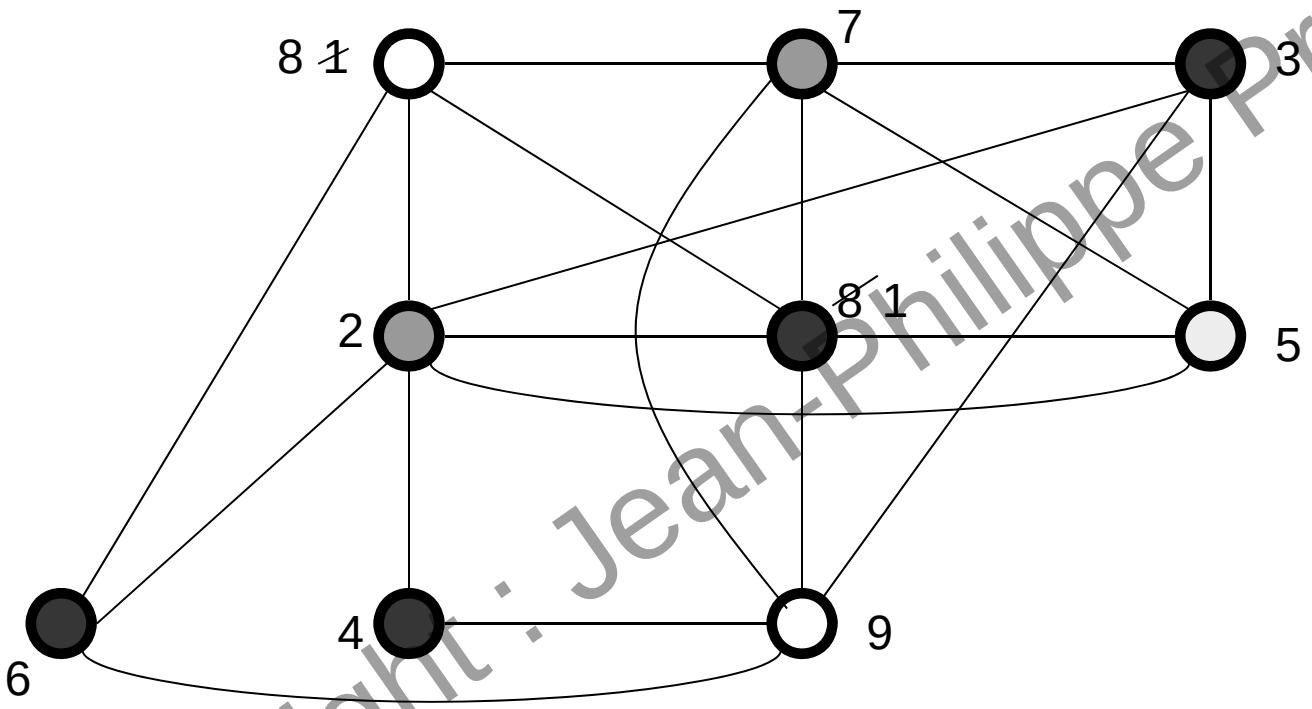
- s = coloration suivante avec 5 couleurs :



s' : Coloration dans V(s) obtenue par FFS après permutation des sommets 1 et 8.

1 2 3 4 5 6 7

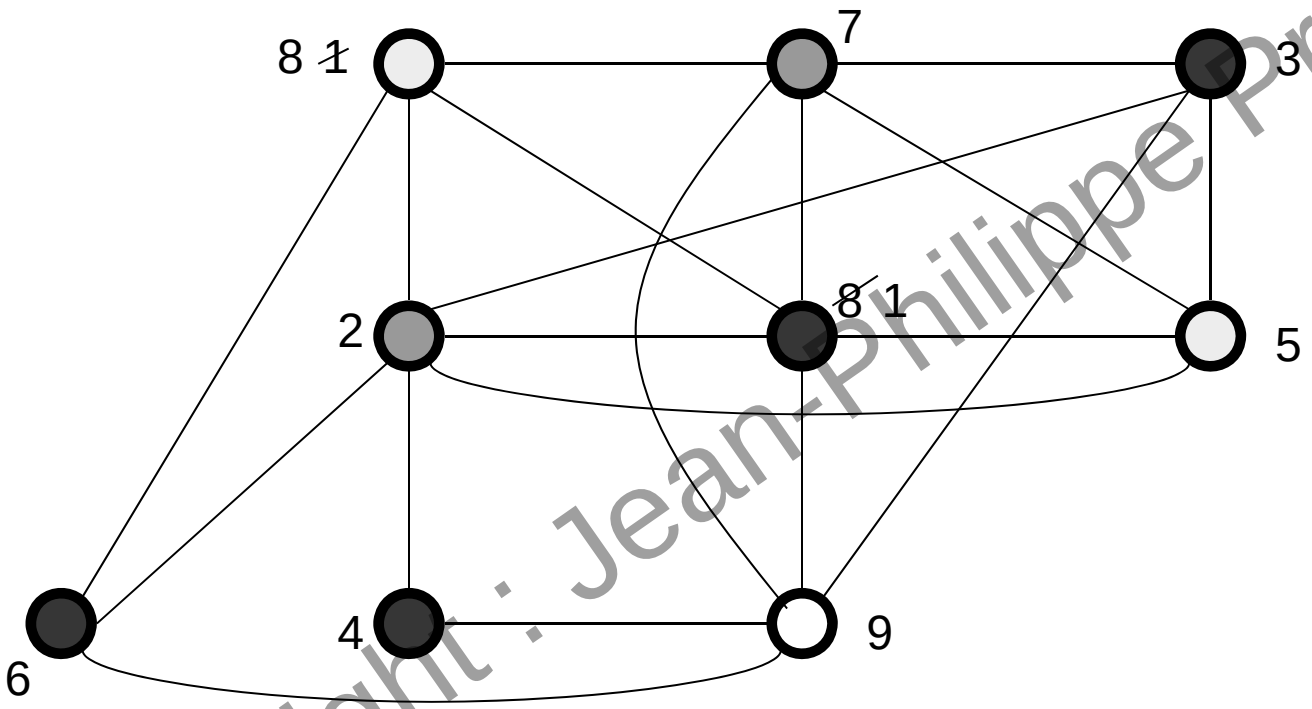
- s = coloration suivante avec 5 couleurs :



s' : Coloration dans V(s) obtenue par FFS après permutation des sommets 1 et 8.

1 2 3 4 5 6 7

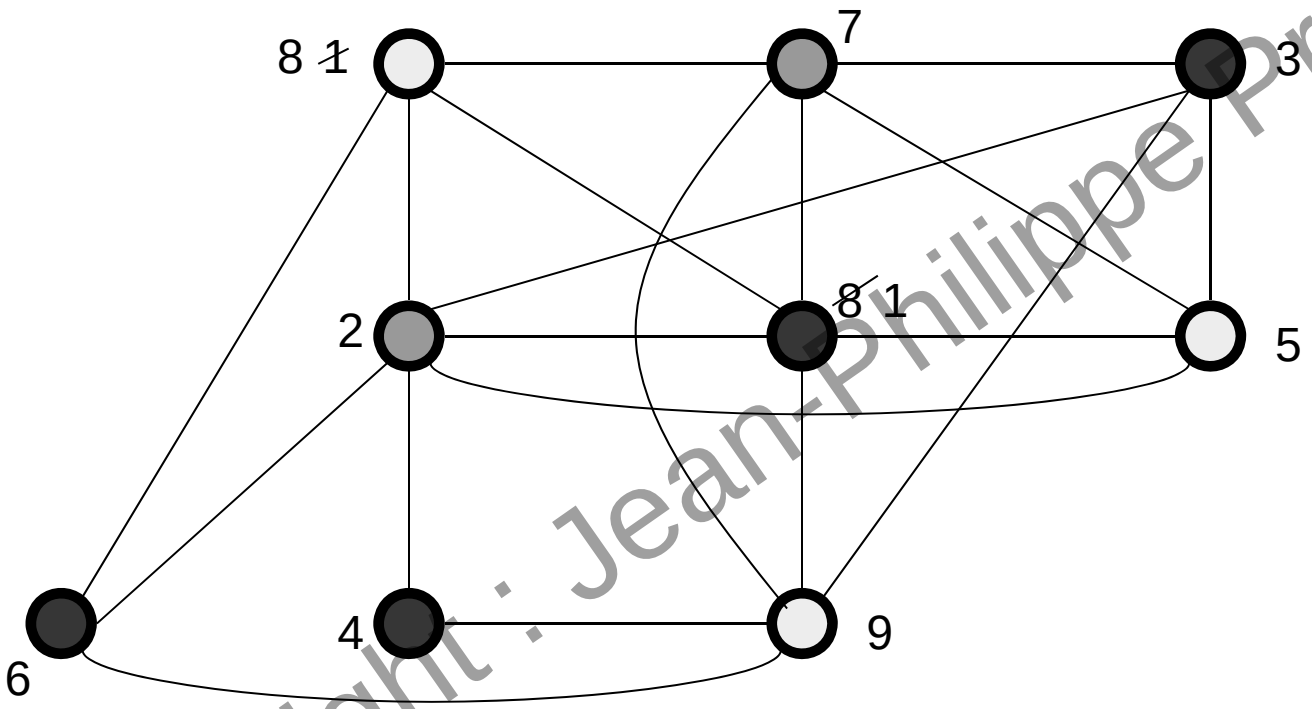
- s = coloration suivante avec 5 couleurs :



s’ : Coloration dans V(s) obtenue par FFS après permutation des sommets 1 et 8.

1 2 3 4 5 6 7

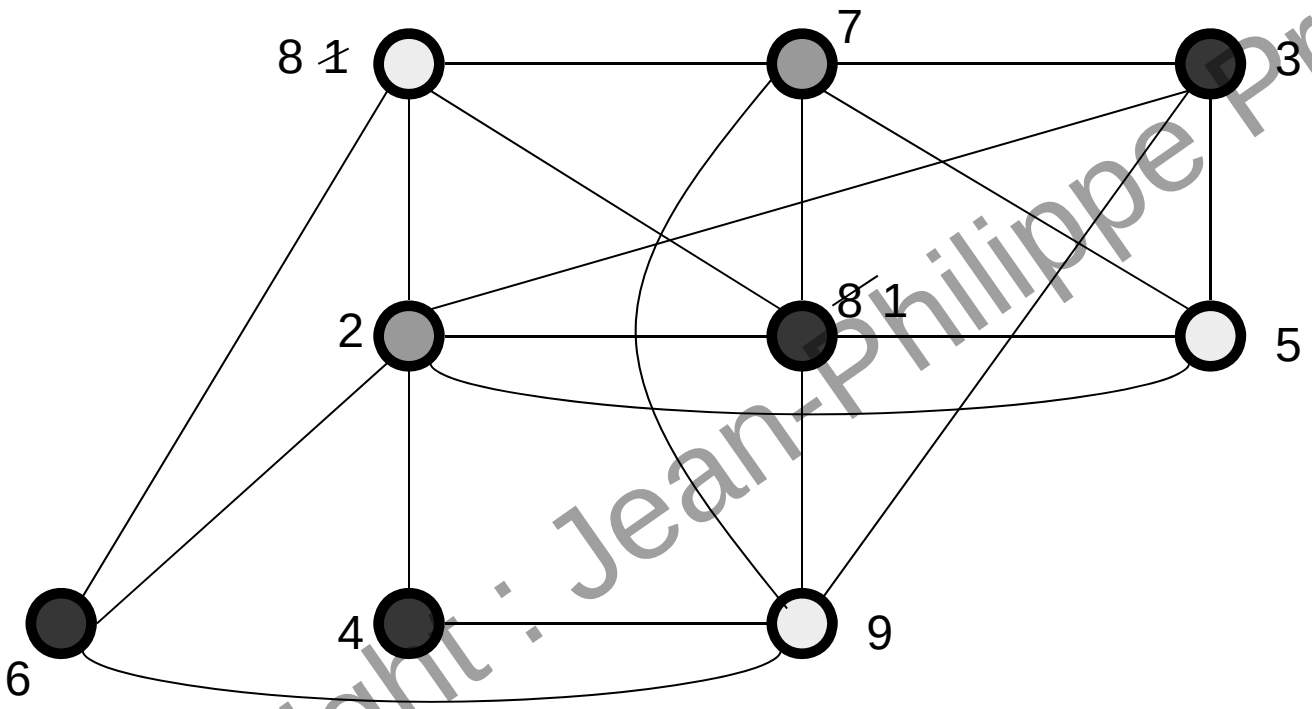
- s = coloration suivante avec 5 couleurs :



s' : Coloration dans V(s) obtenue par FFS après permutation des sommets 1 et 8.

1 2 3 4 5 6 7

- s = coloration suivante avec 5 couleurs :



s' : Coloration dans V(s) obtenue par FFS après permutation des sommets 1 et 8. Avec 3 couleurs !

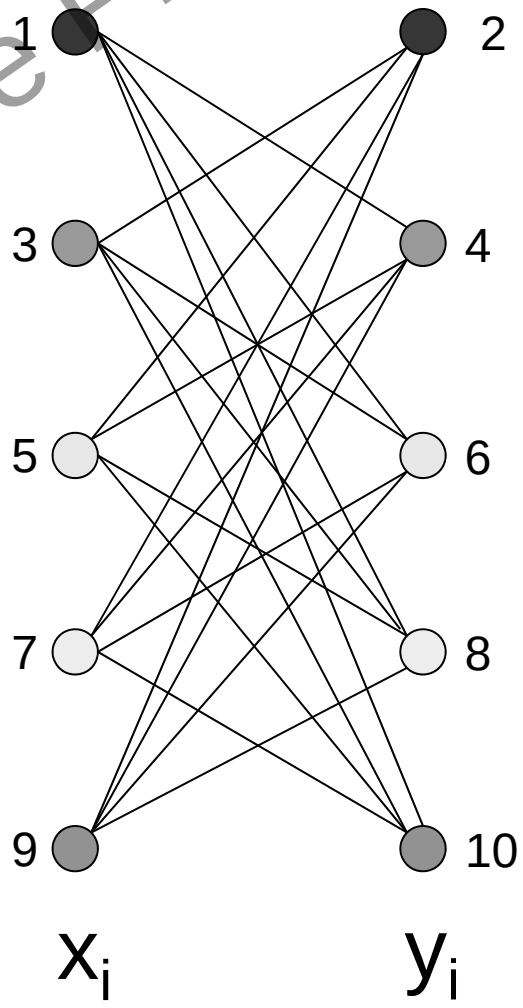
1 2 3 4 5 6 7

- Autre exemple de coloration :

- Reprenons le graphe biparti de sommets $x_1,...,x_5$ et $y_1,...,y_5$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.

- Et l'ordre $x_1,y_1,...,x_5,y_5$ qui produit 5 couleurs avec FFS.

<u>ordre :</u> $x_1y_1x_2y_2x_3y_3x_4y_4x_5y_5$



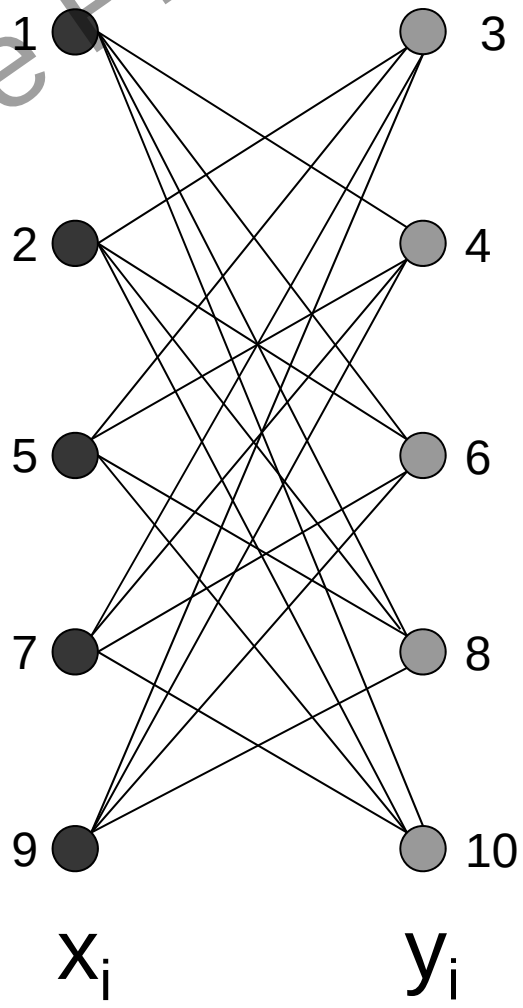
- Autre exemple de coloration :

- Reprenons le graphe biparti de sommets $x_1,...,x_5$ et $y_1,...,y_5$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.

- Et l'ordre $x_1,y_1,...,x_5,y_5$ qui produit 5 couleurs avec FFS.

<u>ordre :</u> $x_1y_1x_2y_2x_3y_3x_4y_4x_5y_5$

$x_1x_2y_1y_2x_3y_3x_4y_4x_5y_5$



- Autre exemple de coloration :

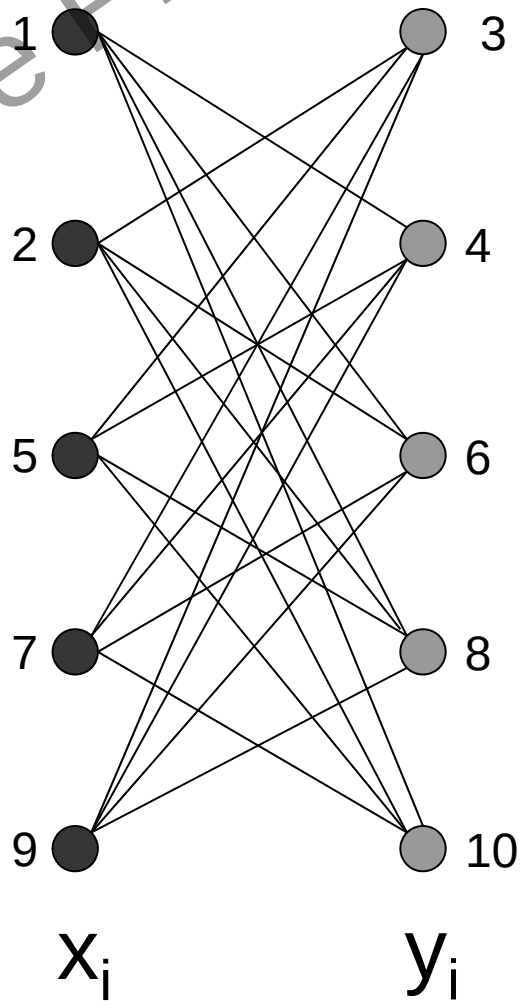
- Reprenons le graphe biparti de sommets $x_1,...,x_5$ et $y_1,...,y_5$, et d'arêtes $[x_a,y_b]$ où $a \neq b$.

- Et l'ordre $x_1,y_1,...,x_5,y_5$ qui produit 5 couleurs avec FFS.

<u>ordre :</u> $x_1y_1x_2y_2x_3y_3x_4y_4x_5y_5$

$x_1x_2y_1y_2x_3y_3x_4y_4x_5y_5$

- Ici tout changement qui 'casse' le préfixe $x_1(y_a...y_b)y_1$ donne un résultat optimal, et autrement n'est pas améliorant...



Comparaison des Heuristiques de coloration

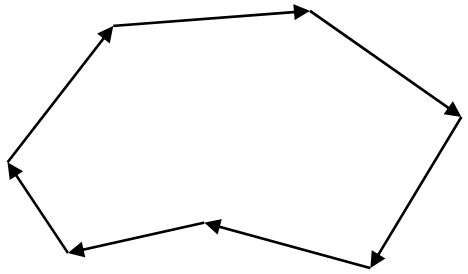
- Implémentation sur une suite aléatoire de graphes à 30 sommets avec un pentium 1Ghz.

Méthode	# couleurs moyen	# optima	Erreur moyenne	Durée moyenne
FFS	9.18	0	27%	<1ms
LFS	8.44	7	17,5%	<1ms
SLS	8.40	7	17%	<1ms
DS	7.92	17	10%	<1ms
R.Loc.	7.64	28	6%	3ms

Recherche locale pour le PVC symétrique

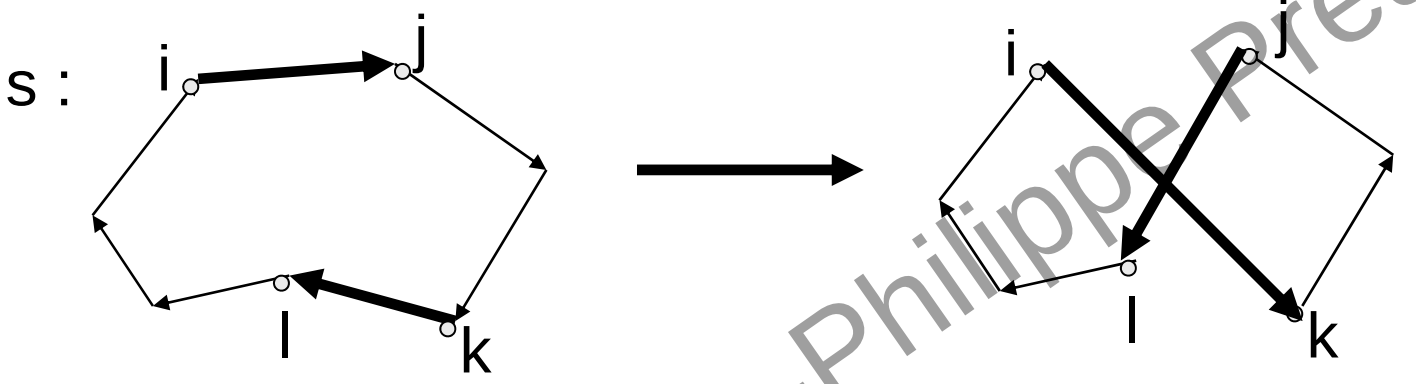
- Méthode 2-opt : on part d'un cycle hamiltonien s :

s :



Recherche locale pour le PVC <u>symétrique</u>

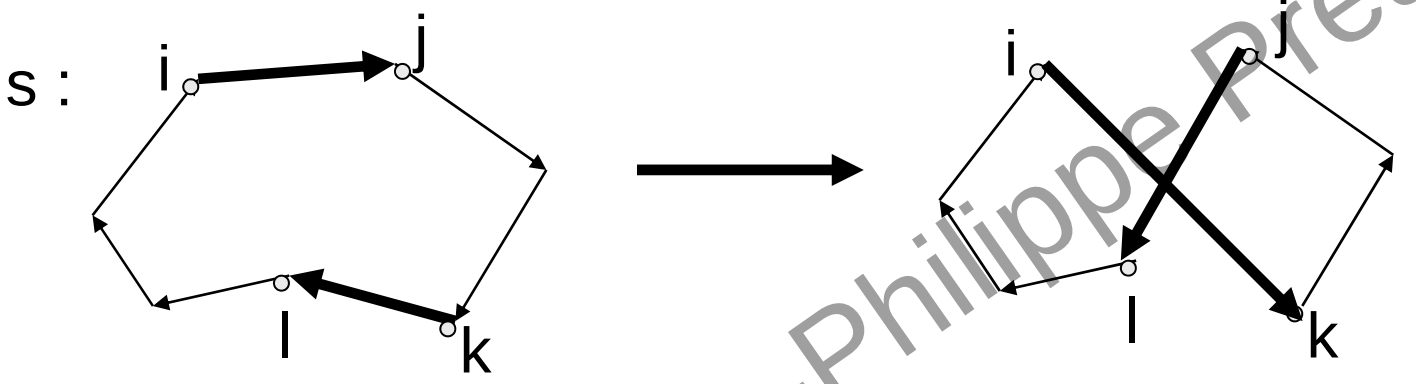
- <u>Méthode 2-opt :</u> on part d'un cycle hamiltonien s :



- Un élément de V(s) s'obtient en considérant deux arêtes non consécutives de s, [i,j] et [k,l], que l'on remplace par les arêtes [i,k] et [j,l] de façon à obtenir un nouveau cycle hamiltonien s'.

Recherche locale pour le PVC <u>symétrique</u>

- <u>Méthode 2-opt :</u> on part d'un cycle hamiltonien s :

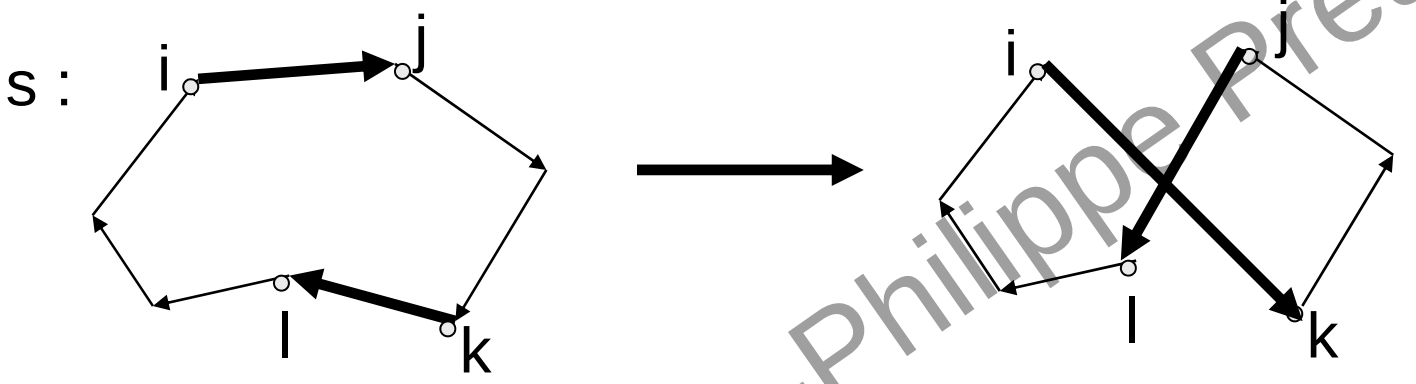


- Un élément de V(s) s'obtient en considérant deux arêtes non consécutives de s, [i,j] et [k,l], que l'on remplace par les arêtes [i,k] et [j,l] de façon à obtenir un nouveau cycle hamiltonien s'.
- le cycle **s'** est meilleur que **s** si et seulement si :

$$c_{ik}+c_{jl} < c_{ij}+c_{kl}$$

Recherche locale pour le PVC <u>symétrique</u>

- <u>Méthode 2-opt :</u> on part d'un cycle hamiltonien s :



- Un élément de V(s) s'obtient en considérant deux arêtes non consécutives de s, [i,j] et [k,l], que l'on remplace par les arêtes [i,k] et [j,l] de façon à obtenir un nouveau cycle hamiltonien s'.
- le cycle **s'** est meilleur que **s** si et seulement si :

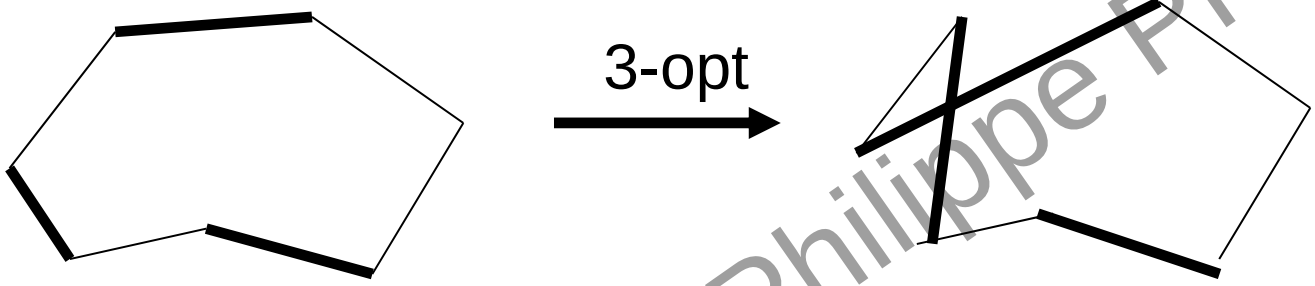
$$c_{ik}+c_{jl} < c_{ij}+c_{kl}$$

- V(s) contient $O(n^2)$ éléments.

Recherche locale pour le PVC <u>symétrique</u>

- <u>Méthode k-opt :</u> on part d'un cycle hamiltonien s :

s :

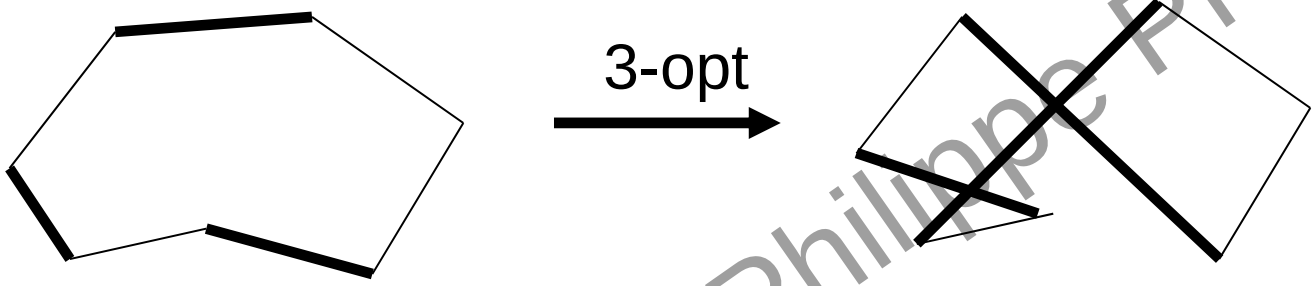


- Un élément de V(s) s'obtient en considérant k arêtes de s, que l'on remplace de façon à obtenir un nouveau cycle hamiltonien s' (c'est à dire connexe).

Recherche locale pour le PVC <u>symétrique</u>

- <u>Méthode k-opt :</u> on part d'un cycle hamiltonien s :

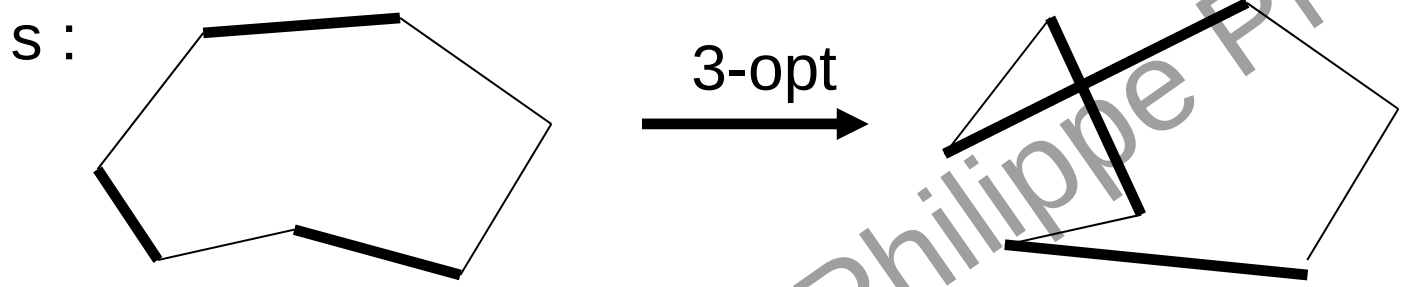
s :



- Un élément de V(s) s'obtient en considérant k arêtes de s, que l'on remplace de façon à obtenir un nouveau cycle hamiltonien s' (c'est à dire connexe).

Recherche locale pour le PVC <u>symétrique</u>

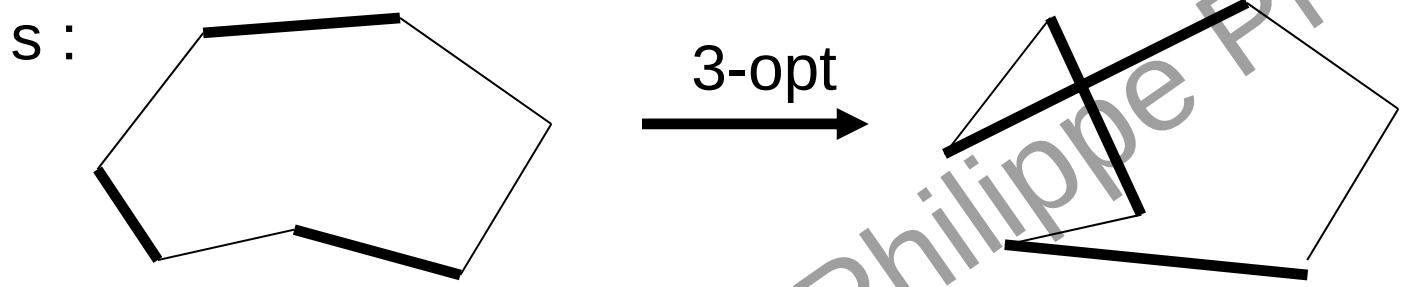
- <u>Méthode k-opt :</u> on part d'un cycle hamiltonien s :



- Un élément de V(s) s'obtient en considérant k arêtes de s, que l'on remplace de façon à obtenir un nouveau cycle hamiltonien s' (c'est à dire connexe).

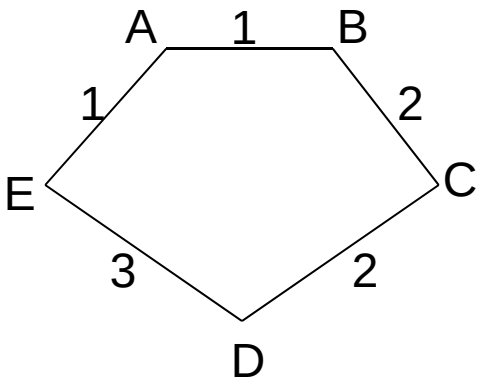
Recherche locale pour le PVC <u>symétrique</u>

- <u>Méthode k-opt :</u> on part d'un cycle hamiltonien *s* :



- Un élément de V(s) s'obtient en considérant k arêtes de s, que l'on remplace de façon à obtenir un nouveau cycle hamiltonien s'.
- V(s) contient $O(n^k)$ éléments.
- En moyenne 3-opt marche un peu mieux que 2-opt. Par contre 4-opt n'améliore que très peu 3-opt pour un coût de calcul très élevé (n-opt est optimal et non efficient). Aussi on n'utilise que 2-opt et 3-opt.
- Meilleures en moyenne que les méthodes gloutonnes.

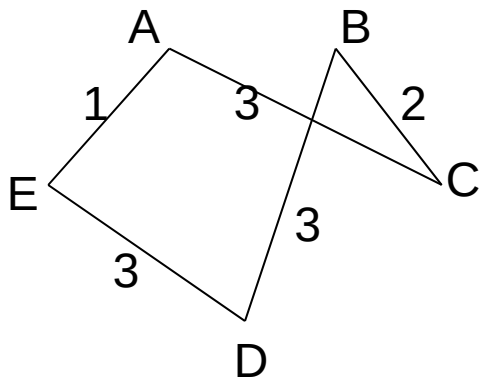
- Exemple : V(s) par 2-opt.



	A	B	C	D	E
A	-	1	3	2	1
B		-	2	2	1
C			-	2	2
D				-	3
E					-

- cycle s=A-B-C-D-E-A, c=9.

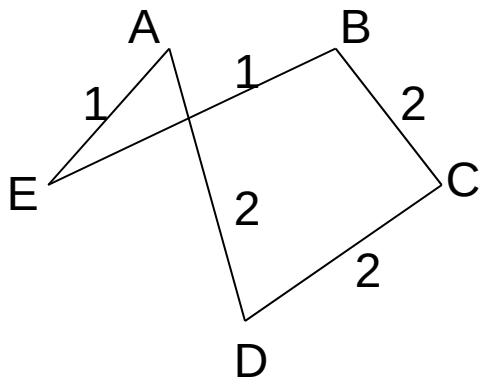
- Exemple : V(s) par 2-opt.



	A	B	C	D	E
A	-	***1***	**3**	2	1
B		-	2	***2***	1
C			-	***2***	2
D				-	3
E					-

- cycle s=A-B-C-D-E-A, c=9.
- A-C-B-D-E-A, c=12.

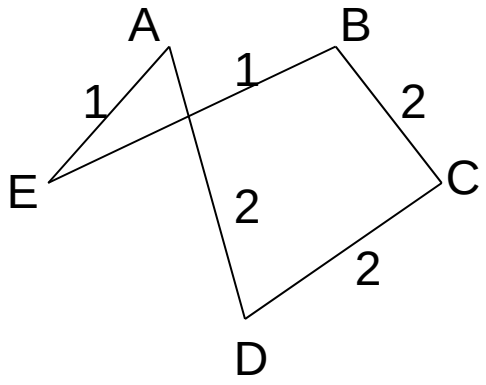
- Exemple : V(s) par 2-opt.



	A	B	C	D	E
A	-	***1***	3	**2**	1
B		-	2	2	**1**
C			-	2	2
D				-	***3***
E					-

- cycle s=A-B-C-D-E-A, c=9.
- A-C-B-D-E-A, c=12.
- A-D-C-B-E-A, c=8.

- Exemple : V(s) par 2-opt.



	A	B	C	D	E
A	-	***1***	3	**2**	1
B		-	2	2	**1**
C			-	2	2
D				-	***3***
E					-

- cycle s=A-B-C-D-E-A, c=9.
- A-C-B-D-E-A, c=12.
- A-D-C-B-E-A, c=8.
- Puis on repart de s'=A-D-C-B-E-A...

Heuristique de Lin-Kernighan pour le PVC

- On considère le PVC symétrique.
- Une des meilleures heuristiques est due à Lin-Kernighan (erreur <1% sur TSPLIB).

Heuristique de Lin-Kernighan pour le PVC

- On considère le PVC symétrique.
- Une des meilleures heuristiques est due à Lin-Kernighan (erreur <1% sur TSPLIB).
- C’est une recherche locale, par une suite de mouvements k-opt, améliorée :

Heuristique de Lin-Kernighan pour le PVC

- On considère le PVC symétrique.
- Une des meilleures heuristiques est due à Lin-Kernighan (erreur <1% sur TSPLIB).
- C’est une recherche locale, par une suite de mouvements k-opt, améliorée :
  - On effectue certains mouvements k-opt pour k variable.
  - On ne ‘bifurque’ pas dès que l’on trouve une solution améliorante, mais on les cherche toutes.
  - On mémorise les meilleurs cycles trouvés .

Heuristique de Lin-Kernighan pour le PVC

- On considère le PVC symétrique.
- Une des meilleures heuristiques est due à Lin-Kernighan (erreur <1% sur TSPLIB).
- C’est une recherche locale, par une suite de mouvements k-opt, améliorée :
  - On effectue certains mouvements k-opt pour k variable.
  - On ne ‘bifurque’ pas dès que l’on trouve une solution améliorante, mais on les cherche toutes.
  - On mémorise les meilleurs cycles trouvés .
- C’est une méthode générique qui admet plusieurs variantes.

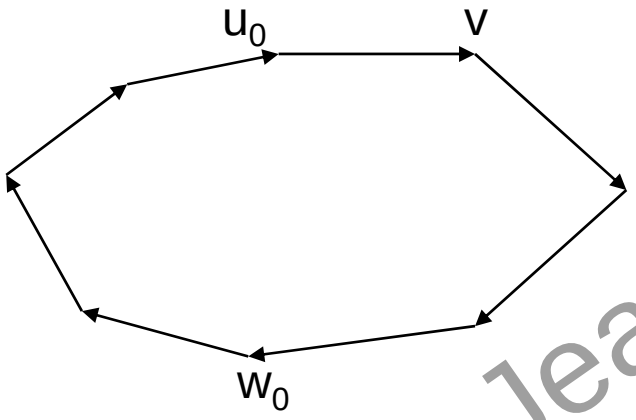
Heuristique de Lin-Kernighan

- Pour le PVC symétrique :
- Partir d'un cycle hamiltonien C.
- Pour chaque sommet u et arête [u,v] de C
  - Faire un « scan de [u,v] »

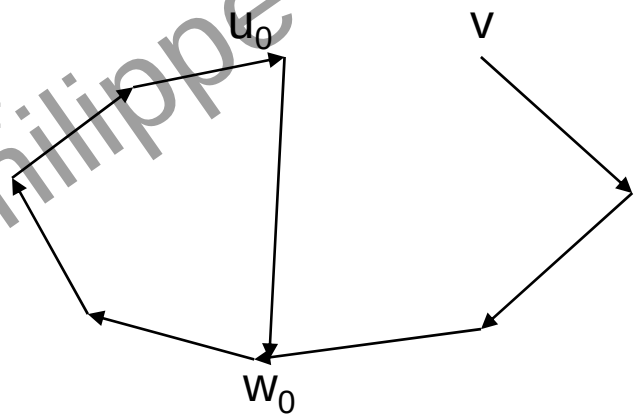
    (construit des cycles par k-opt sur C).
  - Changer C par le meilleur cycle obtenu.
- Fin boucle pour
- Retourner C.

- **<u>Scan de [u_0,v] :</u>**
- Chercher w_0 sommet de C avec c([u_0,w_0]) < c([u_0,v]).
- Si aucun : fin du scan de [u_0,v].
- Sinon : changer dans C, [u_0,v] par [u_0,w_0], et i=0 :

C :

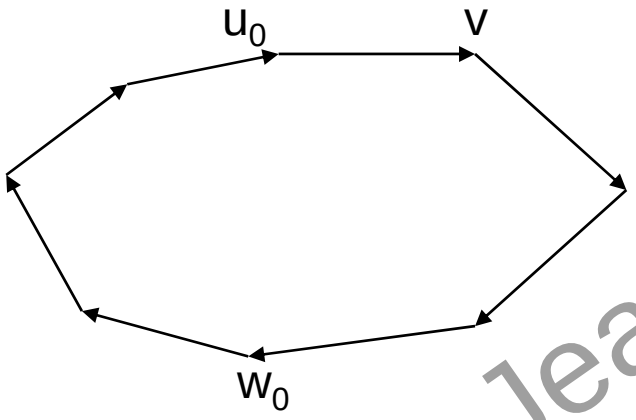


C_0 :

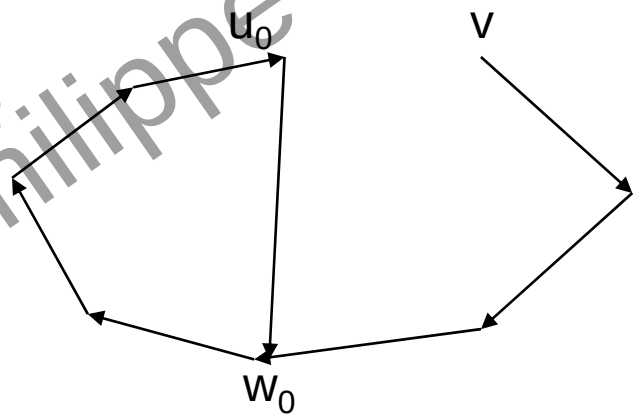


- **<u>Scan de [u_0,v] :</u>**
- Chercher w_0 sommet de C avec c([u_0,w_0]) < c([u_0,v]).
- Si aucun : fin du scan de [u_0,v].
- Sinon : changer dans C, [u_0,v] par [u_0,w_0], et i=0 :

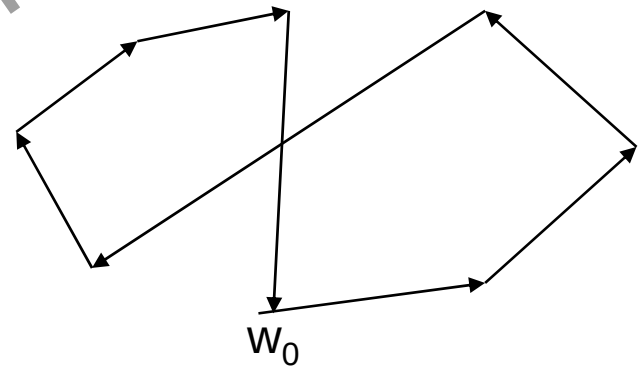
C :



C_0 :



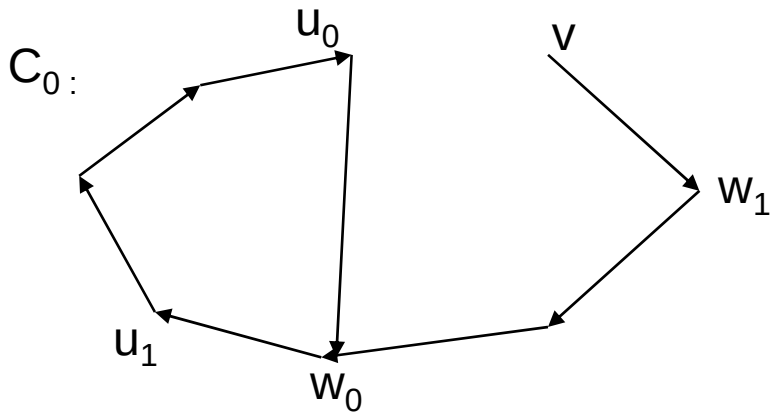
- « Refermer » en un circuit (2-opt).

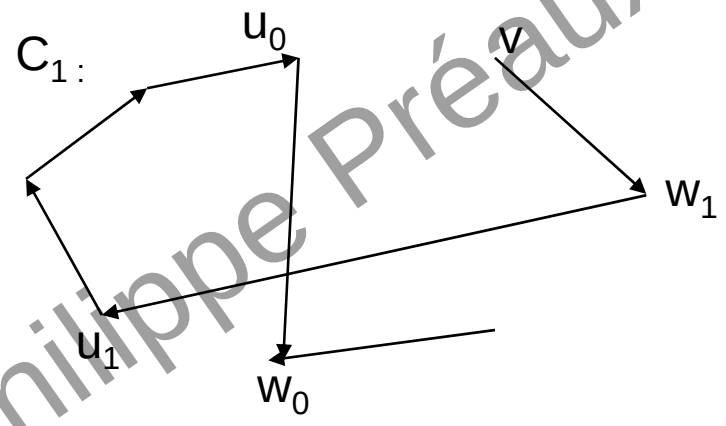


: « fermeture » de C_0

- Si coût < c(C) le stocker. Passer à l'étape suivante

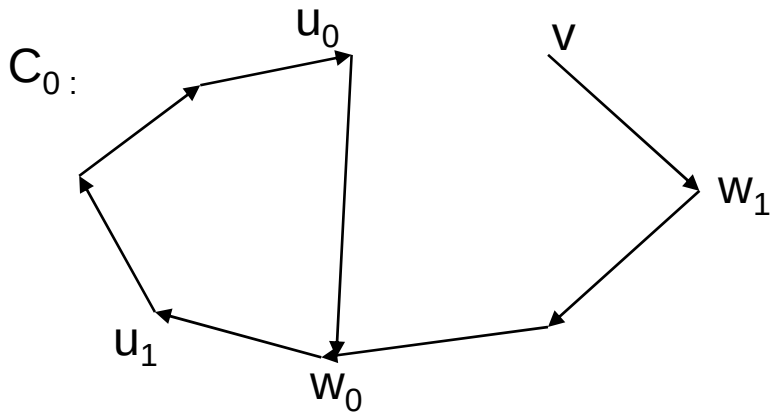
- Soit u_1 le successeur de w_0 dans C_0. Chercher w_1 tel que $c(w_1,u_1) < c(w_0,u_1)$.

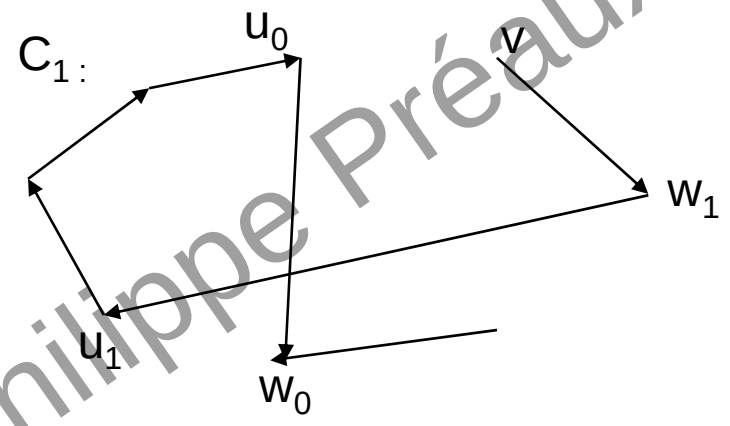




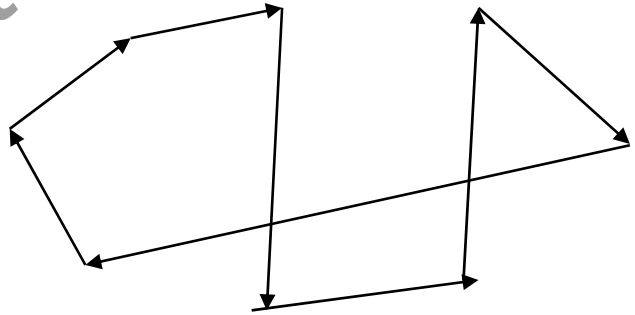
- Si il n'y en a pas, ou si $[w_0,u_1]$ a été ajouté durant ce scan : fin du scan.

- Soit u_1 le successeur de w_0 dans C_0. Chercher w_1 tel que $c(w_1,u_1) < c(w_0,u_1)$.





- Si il n'y en a pas, ou si $[w_0,u_1]$ a été ajouté durant ce scan : fin du scan. Sinon remplacer $[w_0,u_1]$ et l'arête issue de w_1 par $[w_0,w_1]$ dans C_0. On obtient C_1. Et i=i+1.



- Le refermer (3-opt) et le stocker si le coût est meilleur.

- Tant que le scan de $[u_0,v]$ se poursuit :
  - Soit u_{i+1} le successeur de w_i dans C_i. Chercher w_{i+1} tel que $c(w_i,w_{i+1}) < c(w_i,u_{i+1})$.
  - Si il n'y en a pas, ou si $[w_i,u_{i+1}]$ a été ajouté durant ce scan : fin du scan.
  - Sinon remplacer $[w_i,u_{i+1}]$ et l'arête issue de w_i par $[w_i,w_{i+1}]$ dans C_i. On obtient C_{i+1}.
  - i=i+1.
  - Refermer (i+2-opt) ; stocker si meilleur coût.
- Fin Tant que.