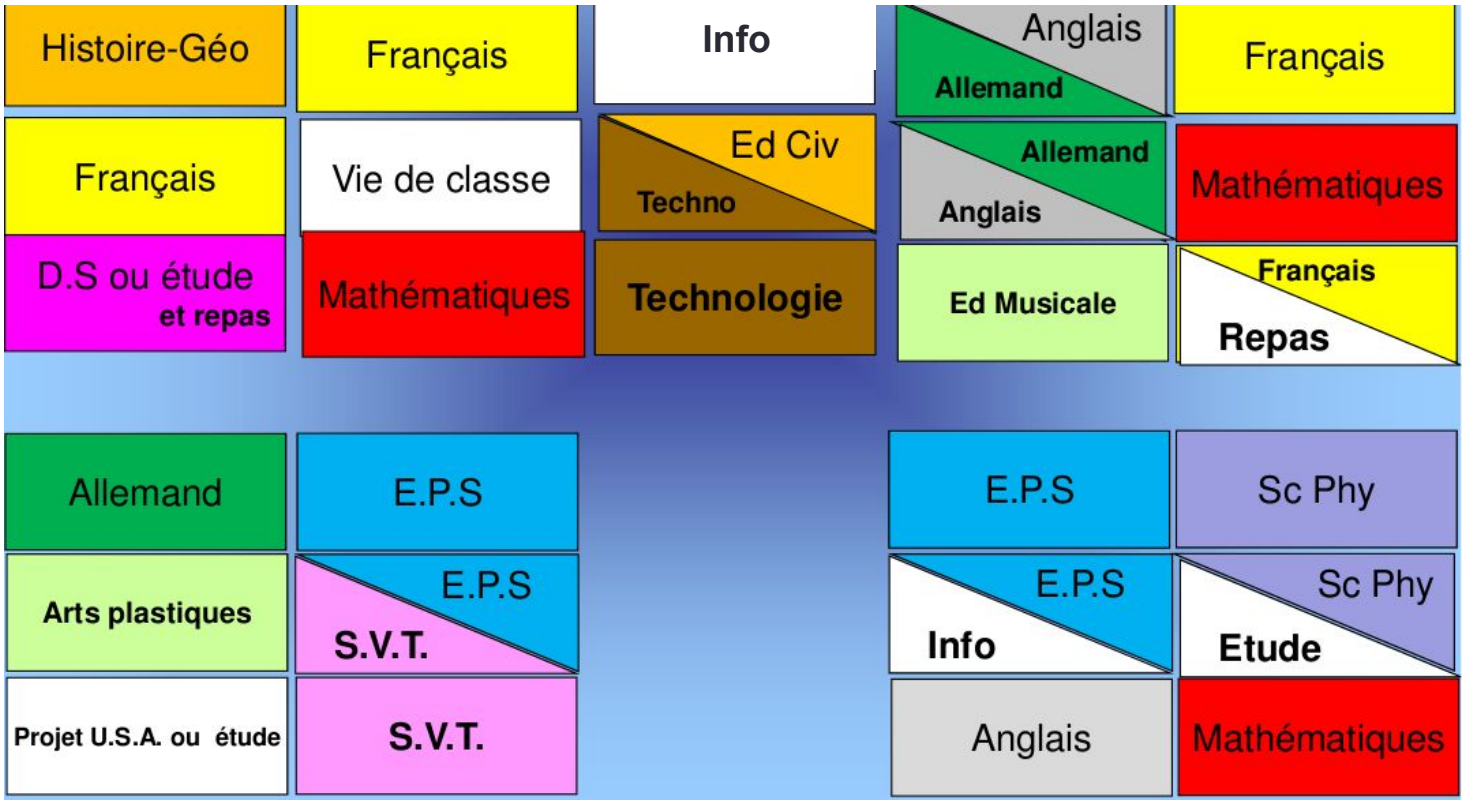
DU PROBLÈME CONCRET À SON MODÈLE ET SA RÉSOLUTION : EXEMPLE EN OPTIMISATION

L. JOURDAN

(Inspiré des supports de Alexandre Gondran
avec son autorisation)

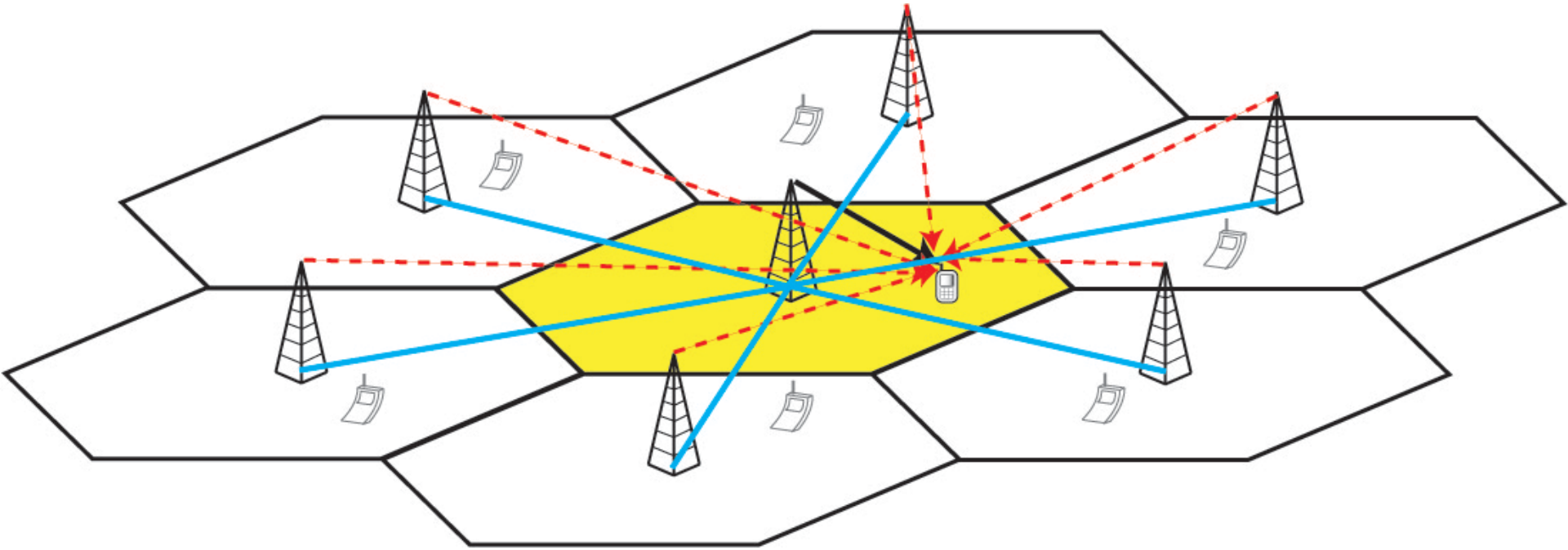
Problèmes d'emploi du temps

- Allocation de créneaux horaires à des événements : cours, examens, …
- Contraintes : enseignants, salle …



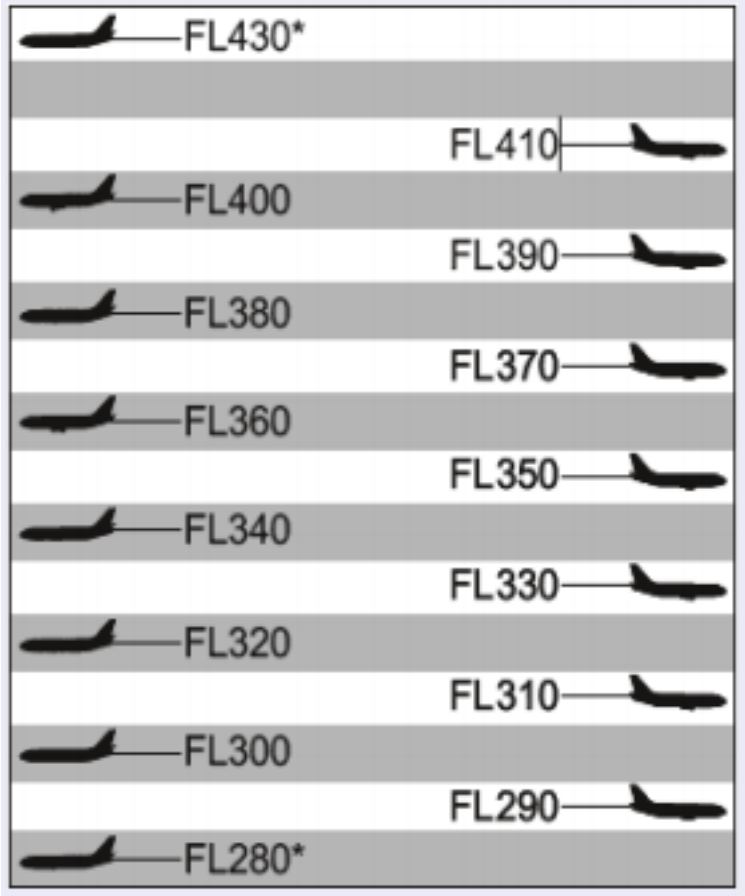
Allocation de fréquences dans les réseaux GSM

- Attribuer aux antennes relais des bandes de fréquences pour communiquer avec les usagers.



Allocation de niveaux de vol

- Attribuer un niveau de vol aux avions pour éviter les conflits aériens.

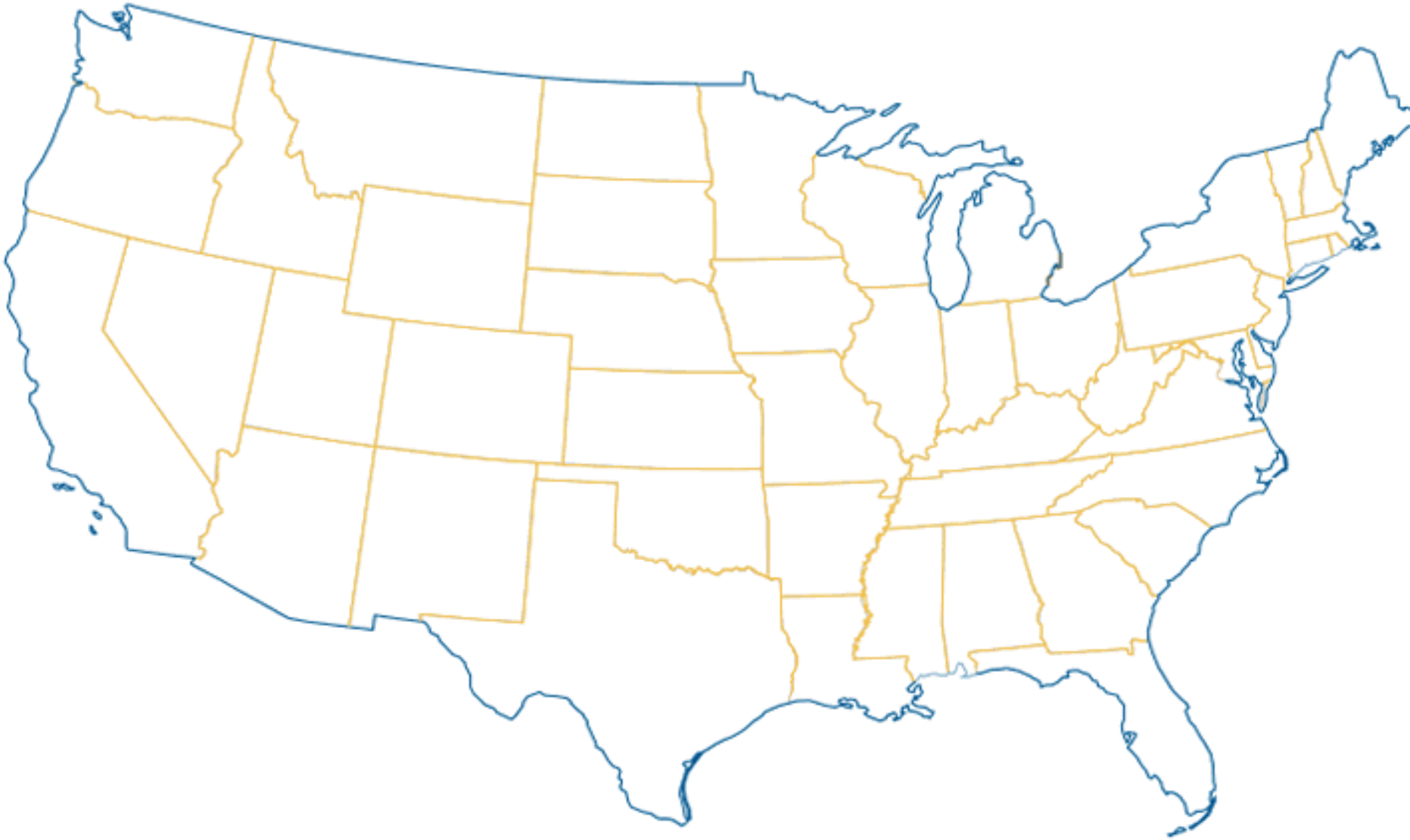


Sudoku, Carré Latin

- Compléter une grille de sudoku
- Le nombre de grilles solution d'un Sudoku est 6670090307520021007209360960 ≈ $6.67 \cdot 10^{21}$

			4			8	7	
	4	7		9	2		5	
2			6				3	
9	7		5			2		3
5		8		2	4	7		6
6		4			7		8	5
	9		3		8			7
		3	2	4		1	6	
	1	2					9	

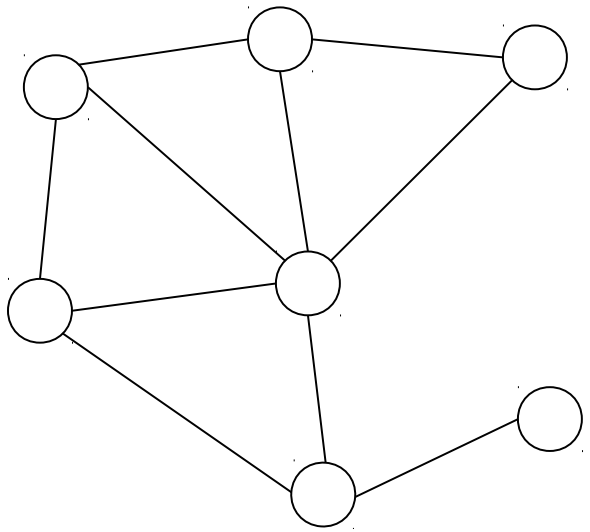
Et les cartes



Graphe

V : ensemble des sommets
E : ensemble des arêtes
$E \subseteq V \times V$ ensemble des arcs
E ensemble des arêtes, relation symetrique

Graphe G= (V,E)



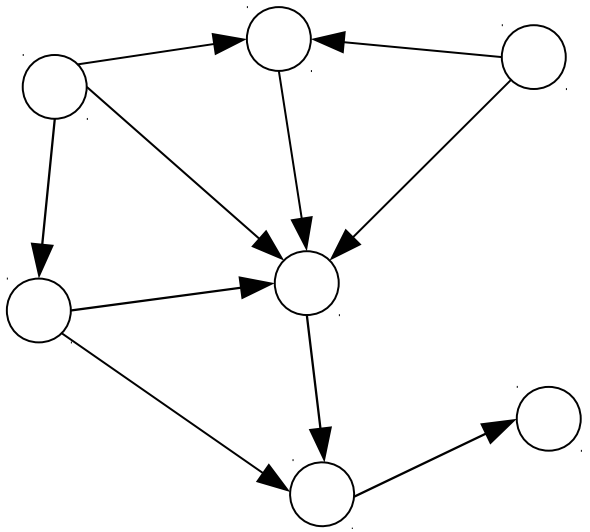
G

Le degré d'un sommet d'un graphe est le nombre d'arêtes reliant ce sommet (avec les boucles comptées deux fois)

Graphe orienté

V : ensemble des sommets
E : ensemble des arêtes
$E \subseteq V \times V$ ensemble des arcs

Graphe G= (V,E)



G

Utilisations des graphes en RO

- Modélisation, représentation de problèmes
  - circulation dans une ville
  - évolution des états d'un système
  - ...
- Résolution de problèmes
  - plus court chemin
  - flot maximal
  - …

Coloration de graphe

- Comment colorer un graphe de telle sorte que deux nœuds adjacents soient de couleur différente ?

(K-)coloration du graphe G=(V,E)

k-coloration de G

V : ensemble des sommets

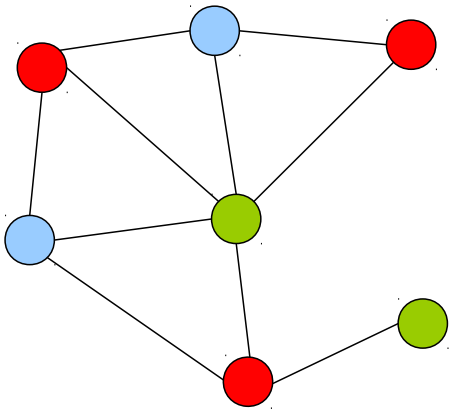
E : ensemble des arêtes

k : nombre de couleurs (fixé)

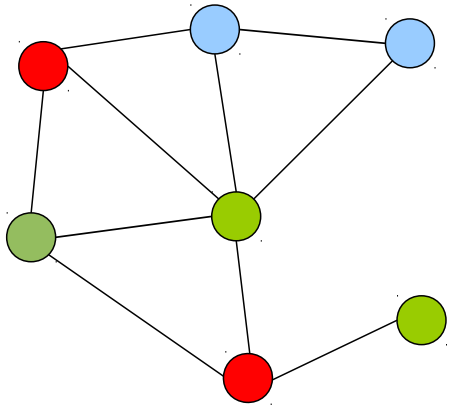
$c : V \rightarrow \{1, 2, \ldots k\}$

$v \longmapsto c(v)$

3-coloration légale



3-coloration NON légale



Définitions

- k-coloration légale : respect des contraintes : $c(v_i) \neq c(v_j) \quad \forall (v_i, v_j) \in E$
- G est k-coloriable s'il admet une k-coloration légale
- Le nombre chromatique $\chi(G)$ est le plus petit entier k tel que G est k – coloriable
- Une classe de couleur est un ensemble des sommets coloriés de la même couleur
- Un stable est un ensemble de sommets non adjacents
- une k -coloration légale = un partitionnement du graphe en k stables

k-coloration de G

V : ensemble des sommets

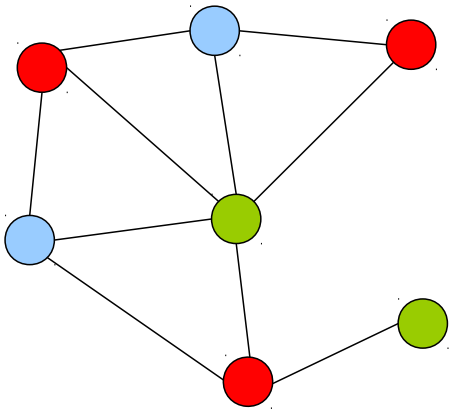
E : ensemble des arêtes

k : nombre de couleurs (fixé)

$c : V \rightarrow \{1, 2, \ldots k\}$

$v \longmapsto c(v)$

3-coloration légale



Définitions

Problème de coloration de graphe : trouver $\chi(G)$

→ NP-difficile

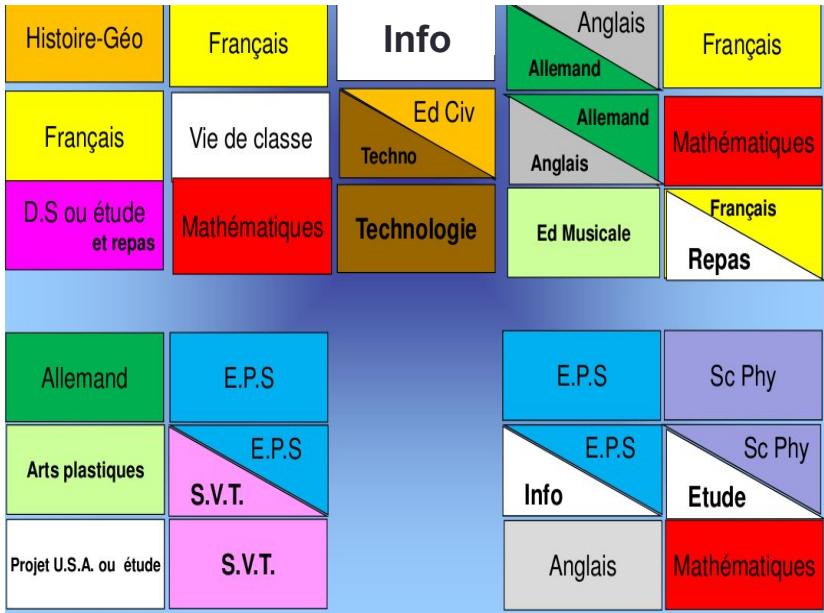
Problème de k -coloration : pour k donné, G est-il k -coloriable ?

→ NP_complet (pour $k > 2$)

Emplois du temps

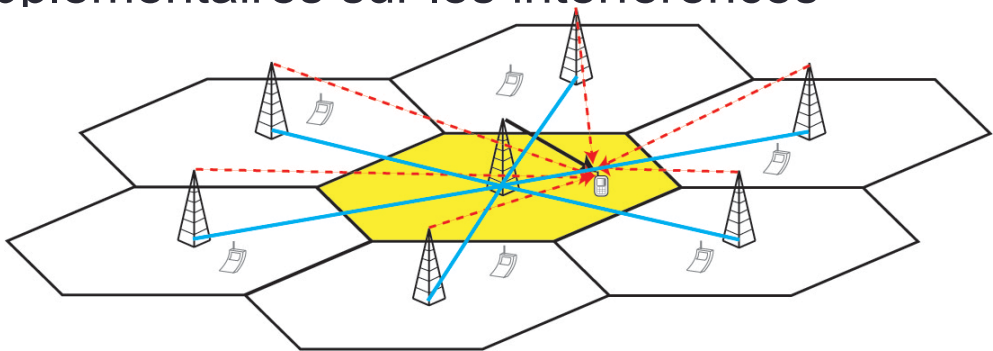
Allocation de créneaux horaires à des événements : cours, examens...

- Modèle sous forme de graphe et coloration de graphe
  - Sommets : les événements
  - Arêtes : les contraintes; deux événements ne peuvent se dérouler simultanément
  - Couleurs : les créneaux horaires
  - Objectif : Minimiser la durée totale des événements



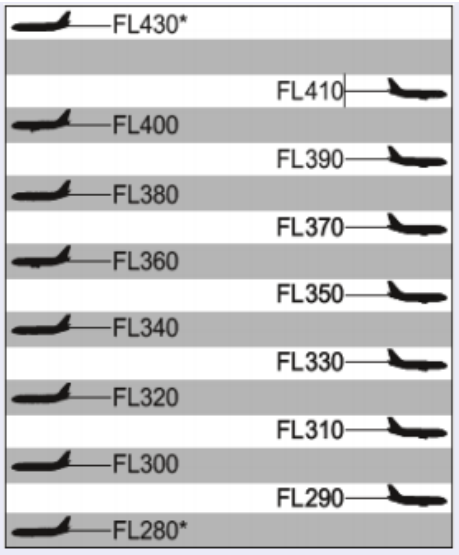
Allocation de fréquences dans les réseaux GSM

- Attribuer aux antennes relais des bandes de fréquences pour communiquer avec les usagers.
- Modèle :
  - Sommets : les antennes relais
  - Arêtes : entre deux antennes trop proches géographiquement l'une de l'autre (niveau d'interférence trop important)
  - Couleurs : les canaux de fréquences radio
  - Objectif : Minimiser le nombre de fréquences utilisées ou pour un nombre de fréquences donné minimiser les interférences
  - + Beaucoup de contraintes supplémentaires sur les interférences et la qualité de service



Allocation de niveaux de vol

- Attribuer un niveau de vol aux avions pour éviter les conflits aériens.
- Modèle :
  - Sommets : les avions
  - Arêtes : entre deux avions en conflits (ne respectant pas les distances de sécurité)
  - Couleurs : les niveaux de vol
  - Objectif : Minimiser le nombre de niveaux de vol utilisés



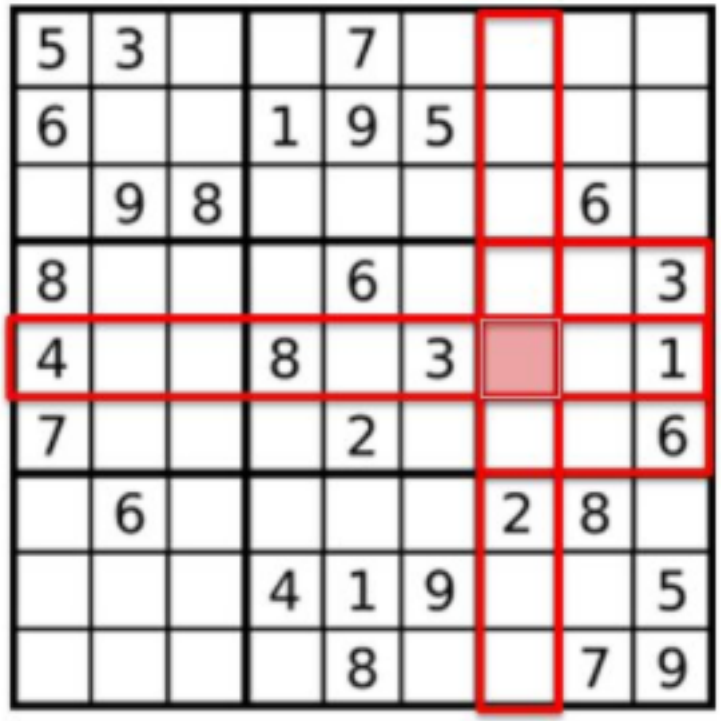
Sudoku

Compléter une grille de sudoku

Modèle

- Sommets : les cases de la grille
- Arêtes : entre deux cases de la même ligne, même colonne et même carré
- Couleurs : les numéros

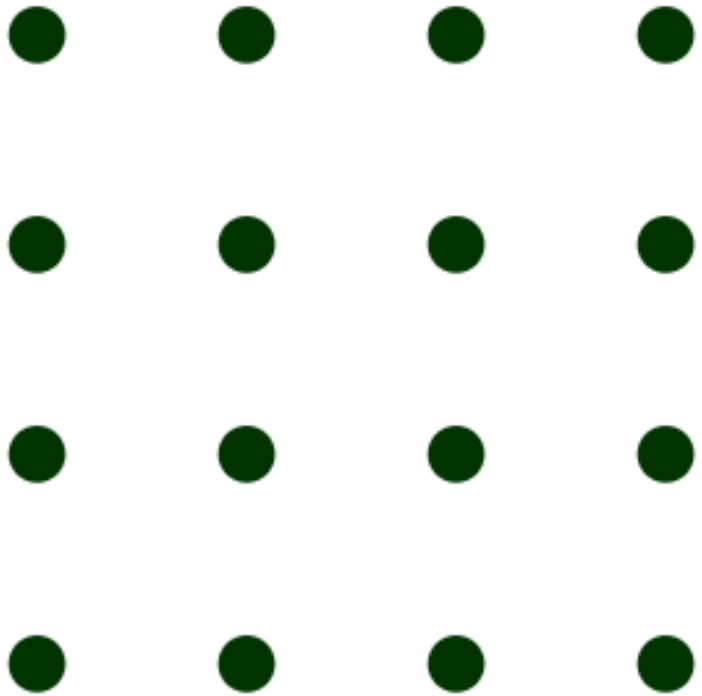
- Existence d'une solution à partir d'une solution partielle (certains somment sont déjà coloriés)
- C'est un 9 coloriage de graphe
- Le graphe a 81 sommets, un pour chaque case de la grille.



Sudoku

1	3	4	2
4	2	1	3
2	4	3	1
3	1	2	4

4 X 4 Sudoku

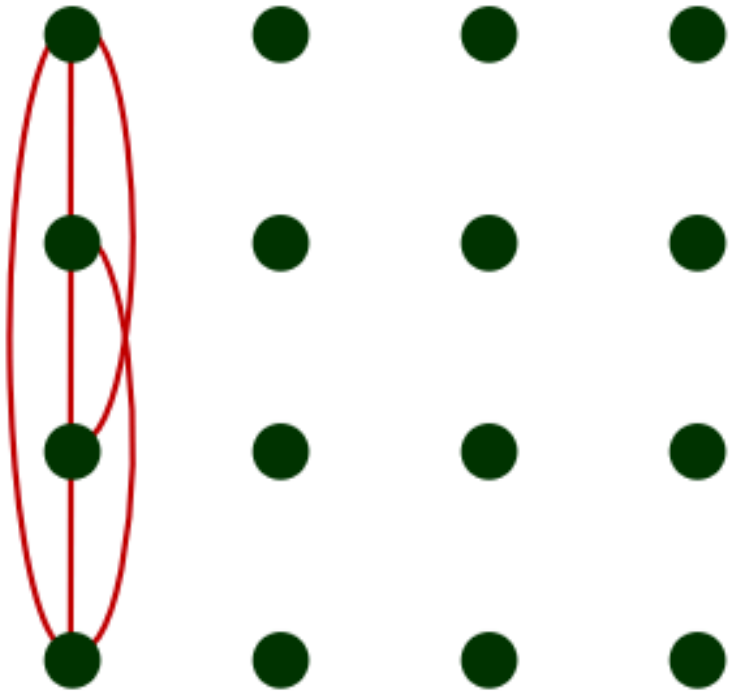


Column constraints

Sudoku

1	3	4	2
4	2	1	3
2	4	3	1
3	1	2	4

4 X 4 Sudoku

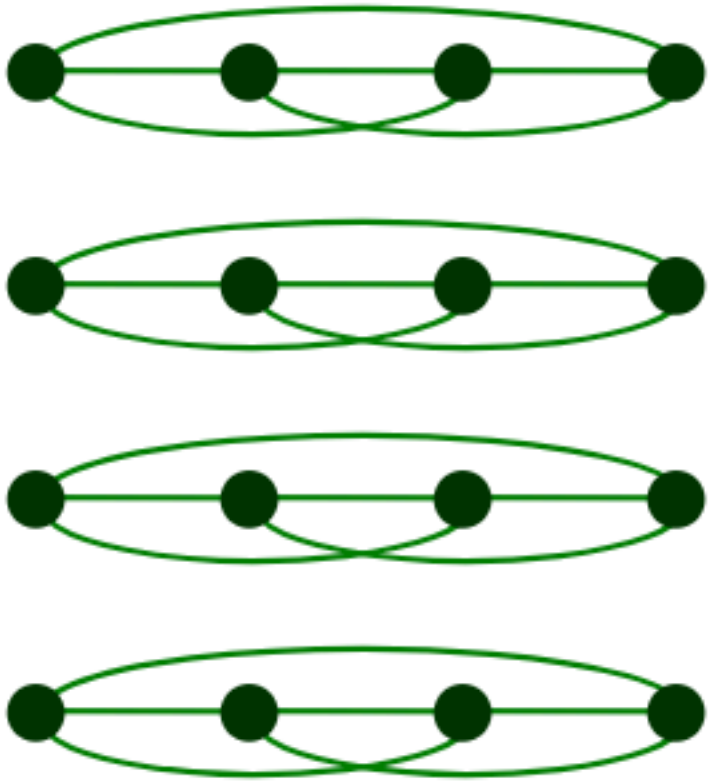


Column constraints

Sudoku

1	3	4	2
4	2	1	3
2	4	3	1
3	1	2	4

4 X 4 Sudoku

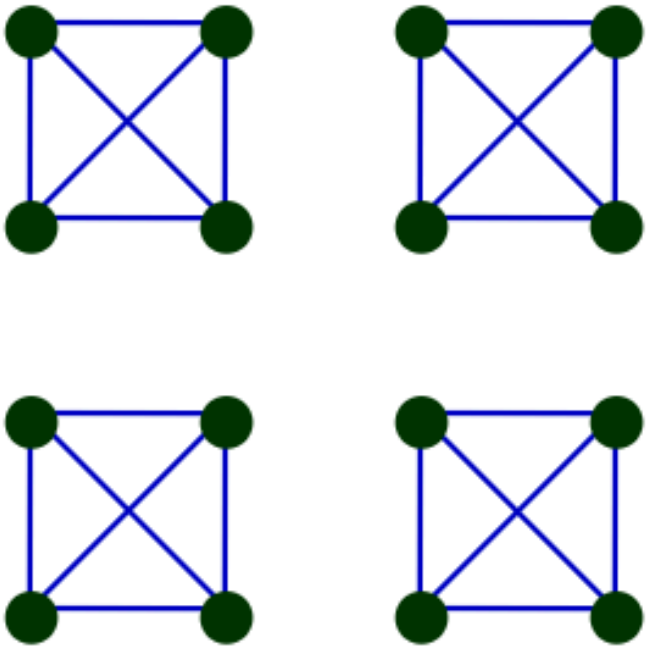


Row constraints

Sudoku

1	3	4	2
4	2	1	3
2	4	3	1
3	1	2	4

4 X 4 Sudoku

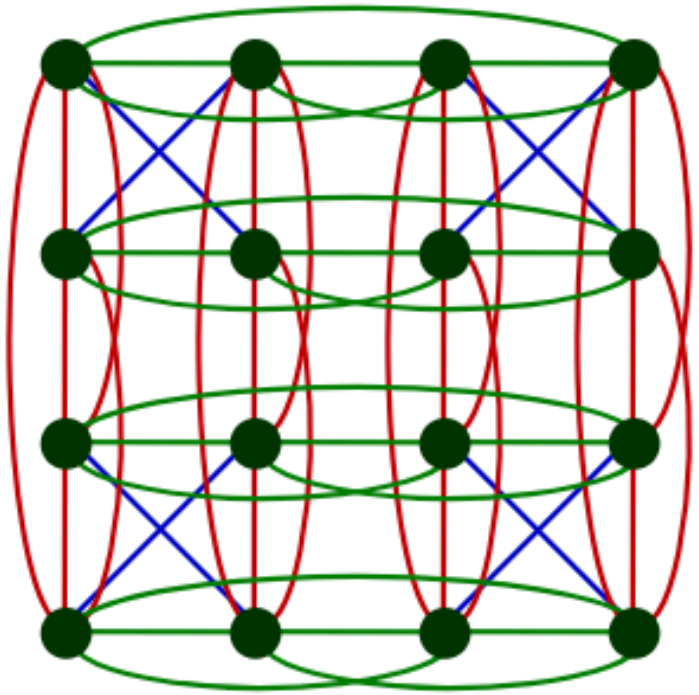


Block constraints

Sudoku

1	3	4	2
4	2	1	3
2	4	3	1
3	1	2	4

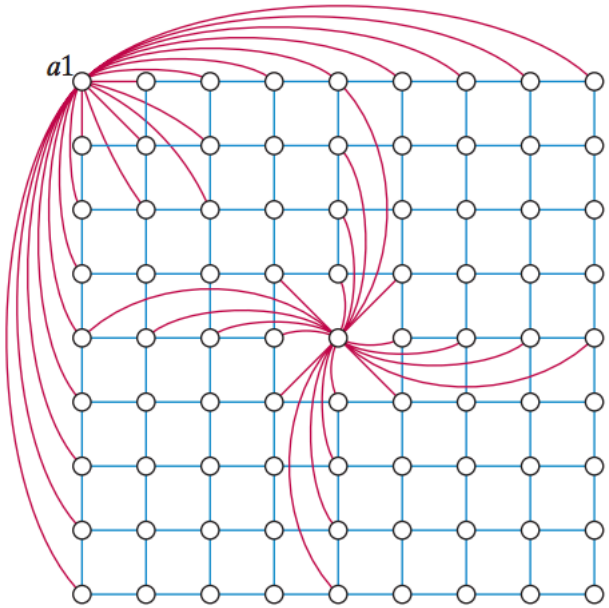
4 X 4 Sudoku
16 nodes



All constraints
56 edges

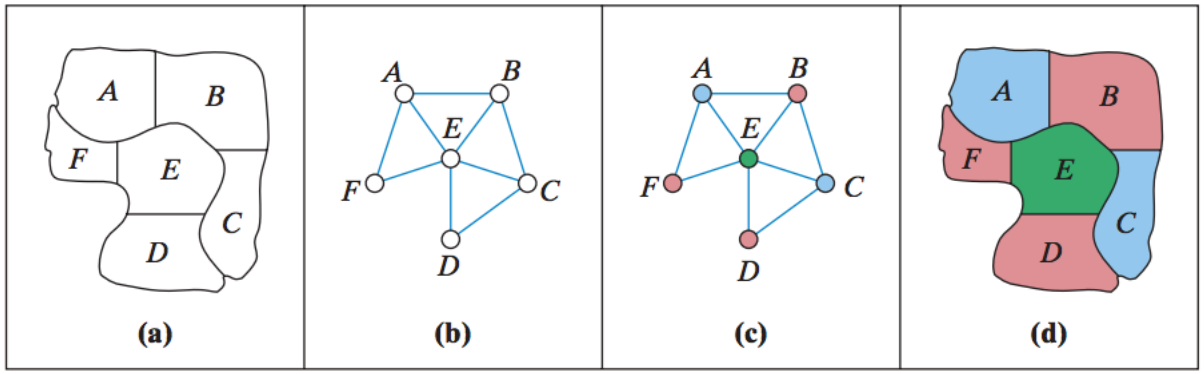
9 x 9 Sudoku: 81 nodes, 810 edges

Sudoku

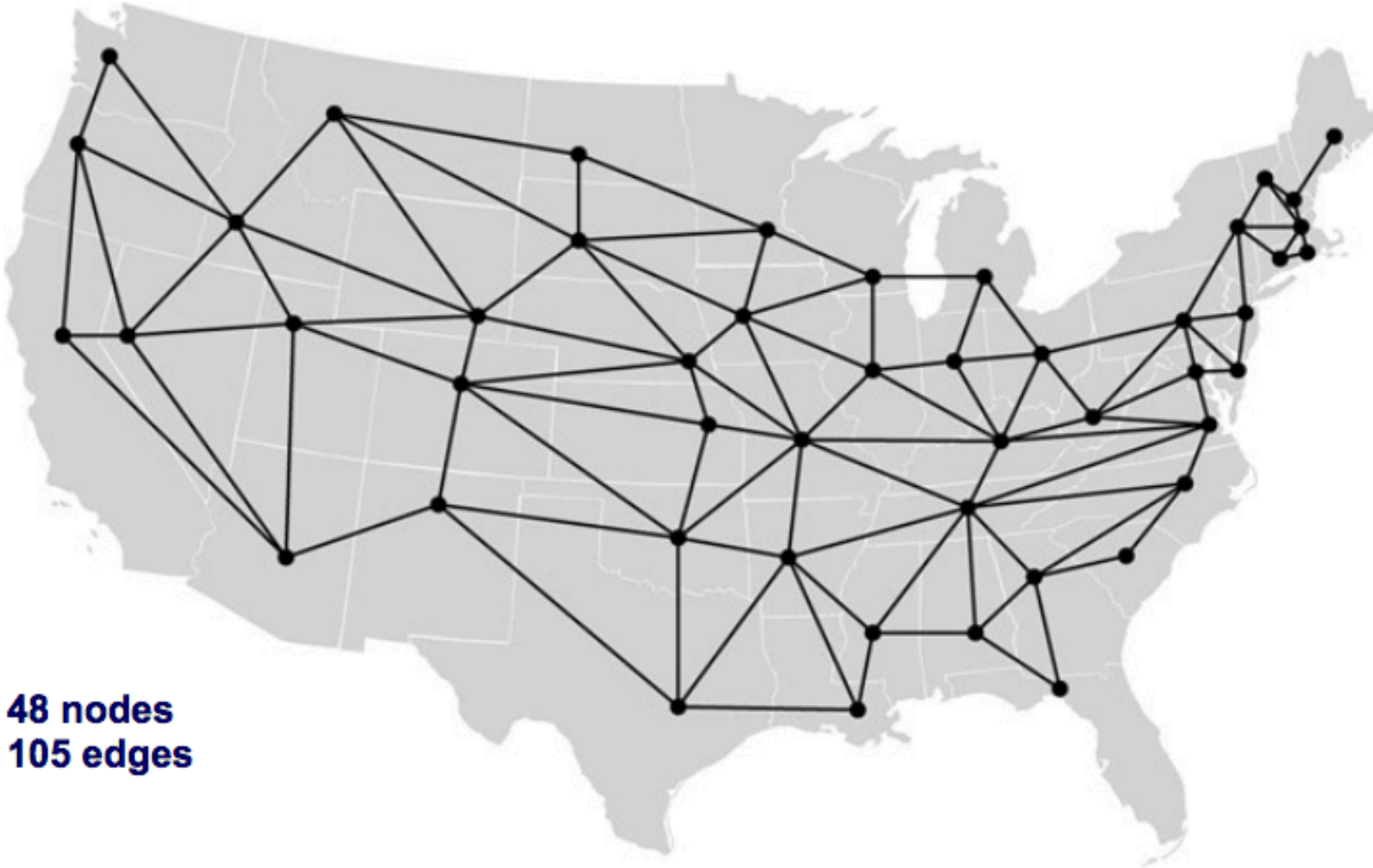


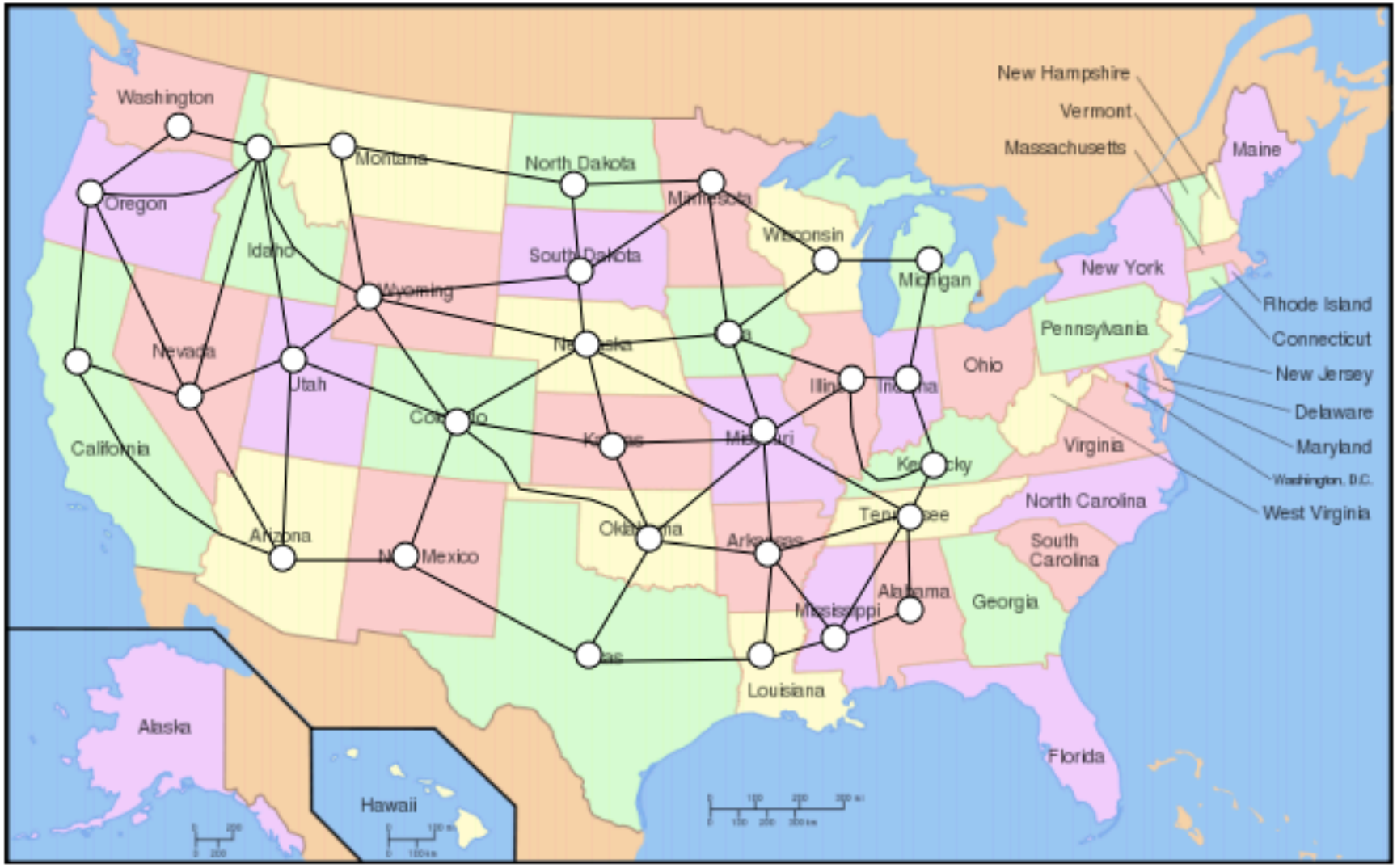
Cell number:	1	2	3	4	5	6	7	8	9
Vertex color:	●	●	●	●	●	●	●	●	●

Coloriage de carte



Coloriage de carte



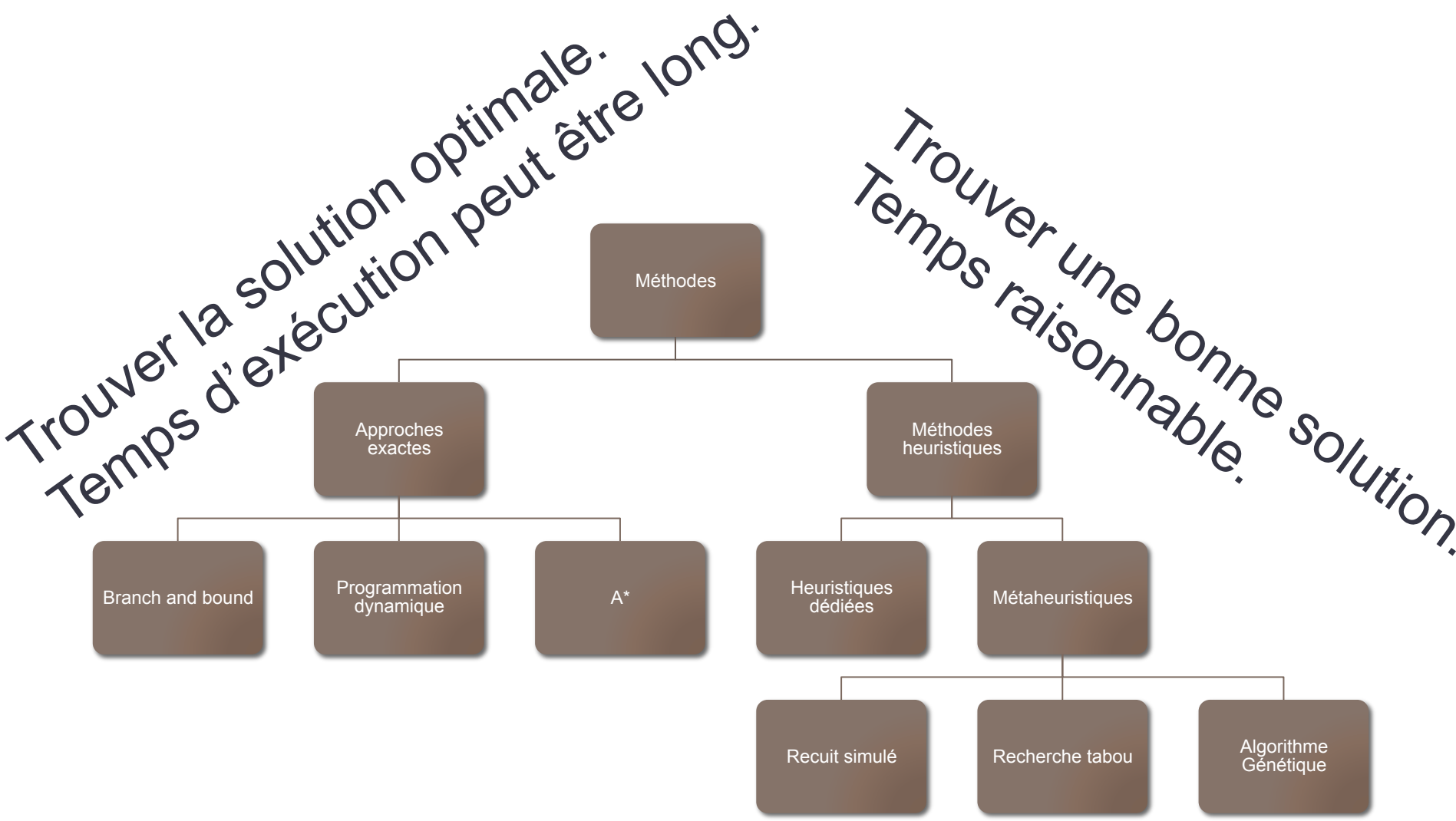

New Hampshire
Vermont
Massachusetts
Maine
Washington
Montana
North Dakota
Minnesota
Oregon
Idaho
Wisconsin
South Dakota
Michigan
New York
Wyoming
Rhode Island
Pennsylvania
Connecticut
Nevada
Utah
Ohio
New Jersey
Delaware
Maryland
Virginia
California
Washington, D.C.
West Virginia
North Carolina
Arizona
Mexico
South Carolina
Georgia
Alaska
Louisiana
Florida
Hawaii

Méthodes de résolution - Principales approches

Quatre principales approches

1. Méthodes constructives : Méthodes gloutonnes, évaluations et séparations, recherches arborescentes, programmation par contraintes...
2. Recherches locales ou par voisinage : descentes, recuits simulés, méthodes tabou, min-conflit...
3. Approches d'évolution ou à population : algorithme génétique, stratégies d'évolution, programmation évolutive...
4. Hybridation : combinaison évolution + RL, PPC + RL...

Méthodes de résolution



Un algorithme glouton de coloration de graphe

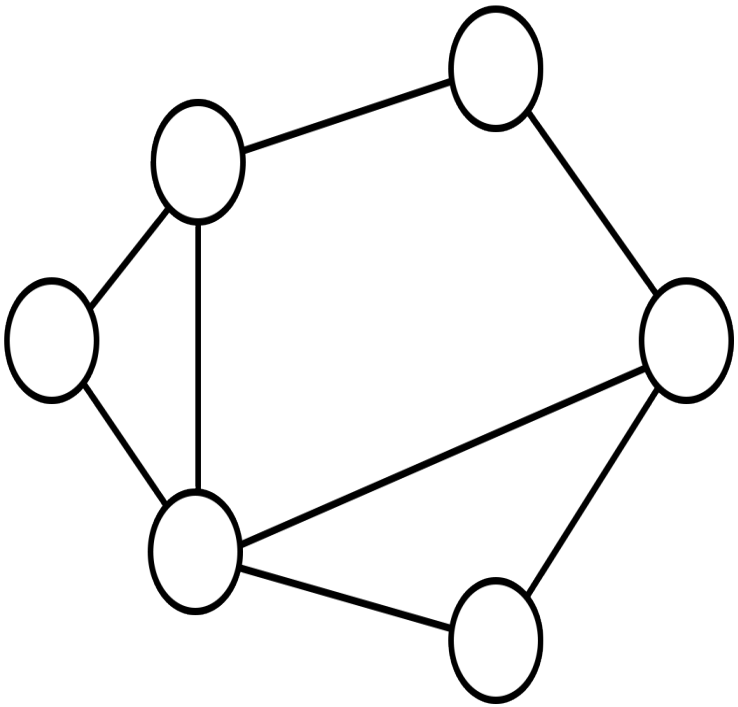
Algorithme glouton : avance étape par étape, choisit une solution optimale localement, sans souci d'optimalité globale.

1. Définir un ordre quelconque $x_1, x_2, \ldots, x_n$ des sommets de G
2. Pour i de 1 à n, faire colorier x_i avec la plus petite couleur non utilisée par un de ses voisins déjà colorié.

- Cet algorithme colorie tout graphe G en au plus $\Delta(G) + 1$ couleurs, où $\Delta(G)$ est le degré maximum des sommets de G.

Coloration de graphe : algorithme glouton

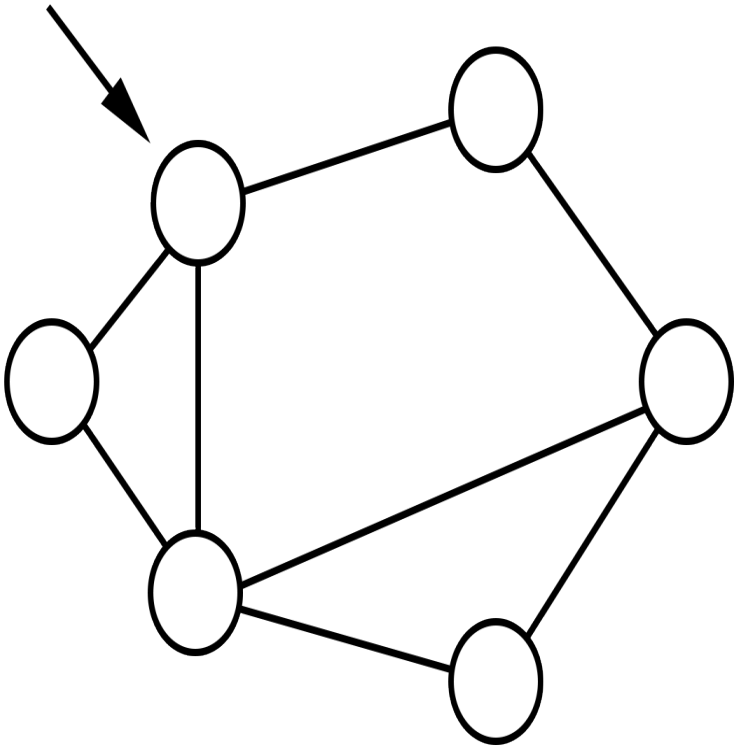
Exemple :



Couleurs utilisées :

Coloration de graphe : algorithme glouton

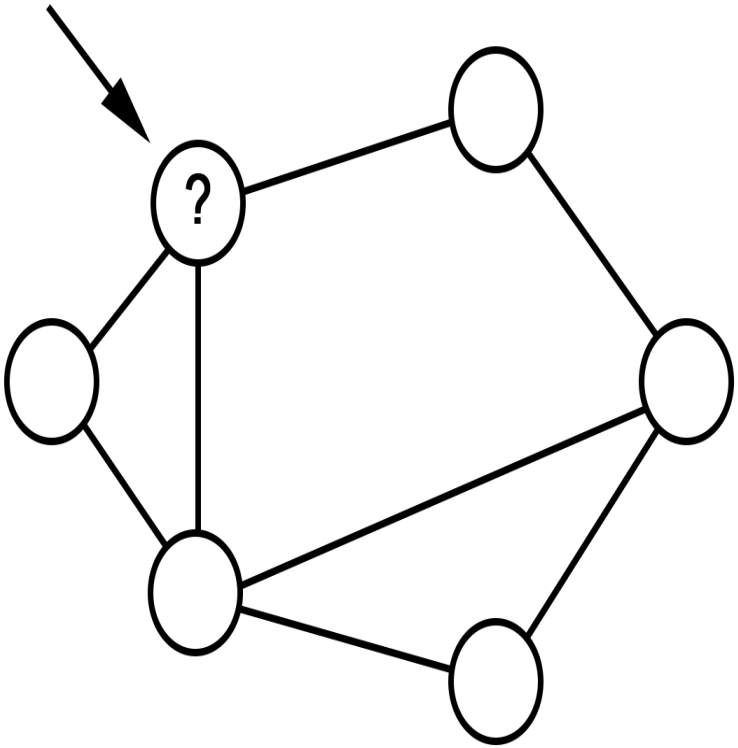
Exemple :



Couleurs utilisées :

Coloration de graphe : algorithme glouton

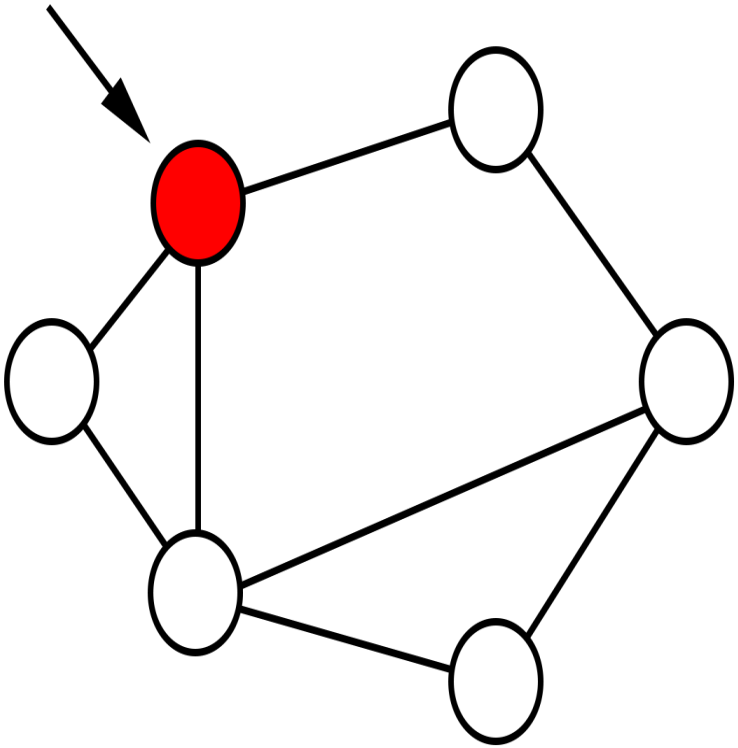
Exemple :



Couleurs utilisées :

Coloration de graphe : algorithme glouton

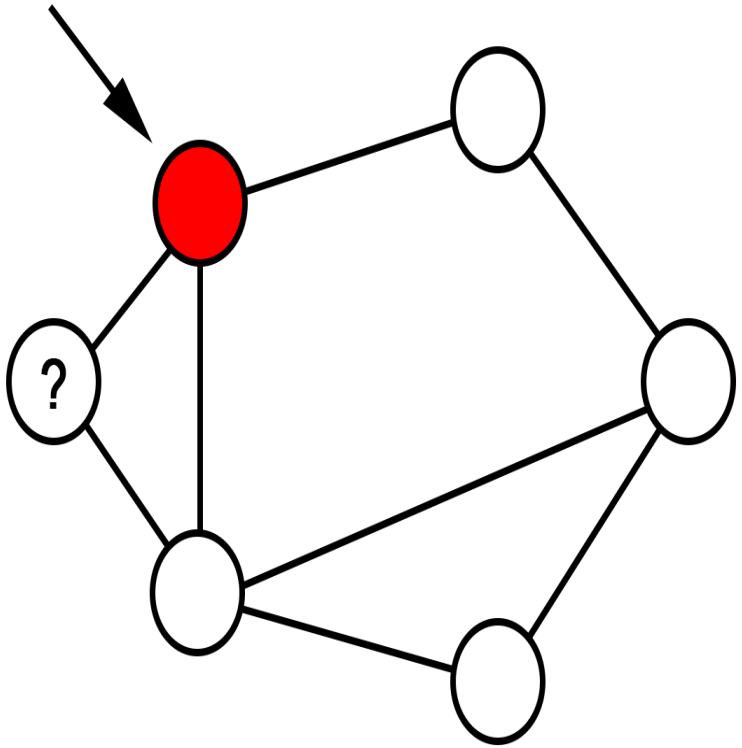
Exemple :



Couleurs utilisées : 0

Coloration de graphe : algorithme glouton

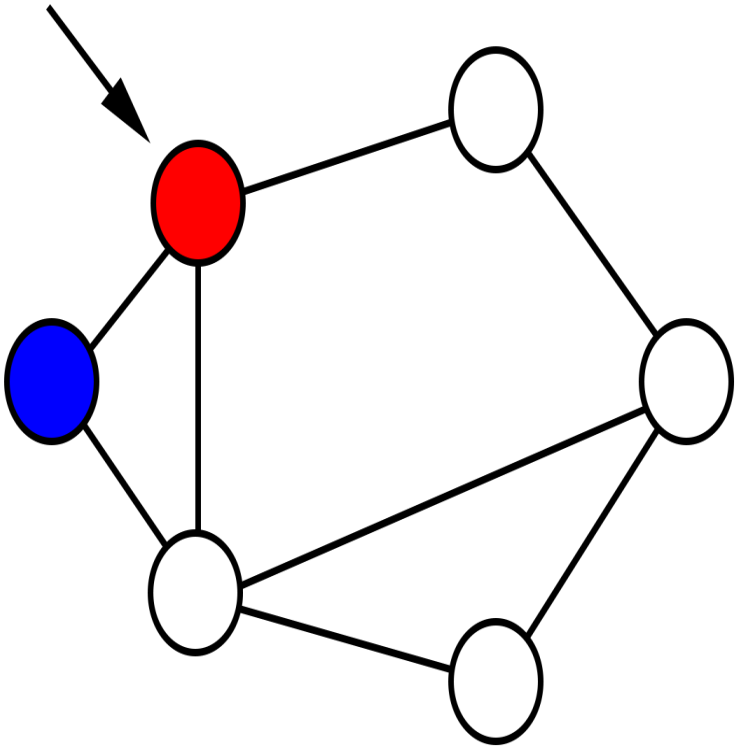
Exemple :



Couleurs utilisées : 0

Coloration de graphe : algorithme glouton

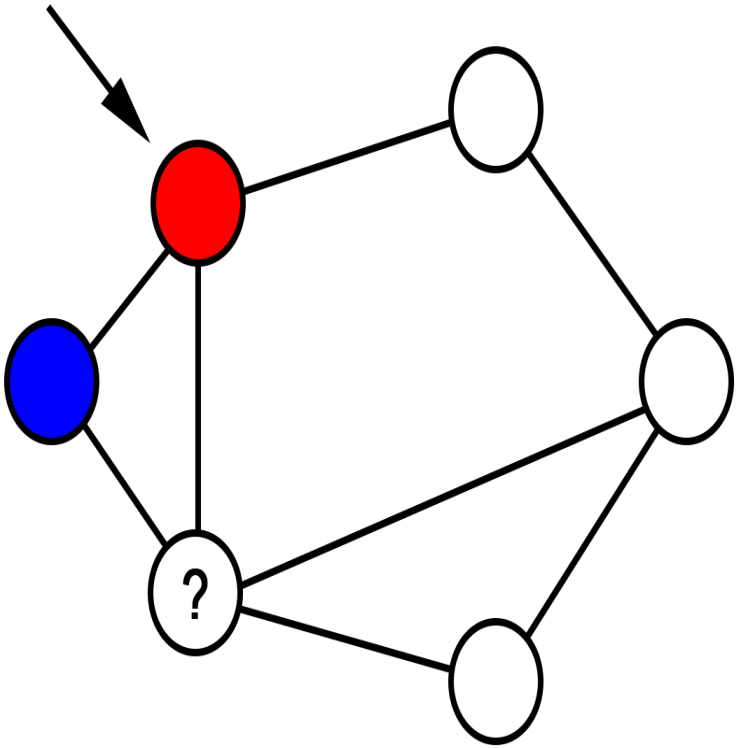
Exemple :



Couleurs utilisées : 0, 1

Coloration de graphe : algorithme glouton

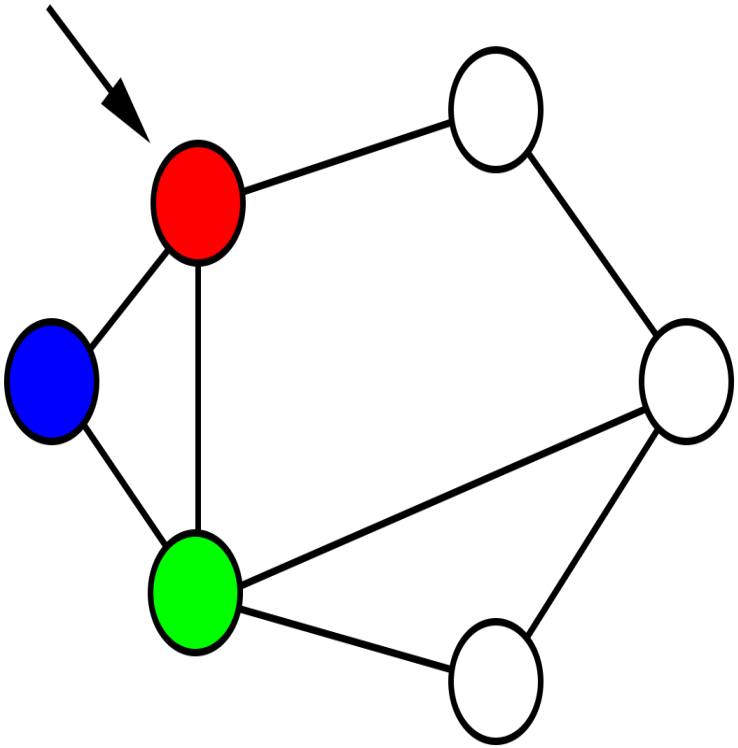
Exemple :



Couleurs utilisées : 0, 1

Coloration de graphe : algorithme glouton

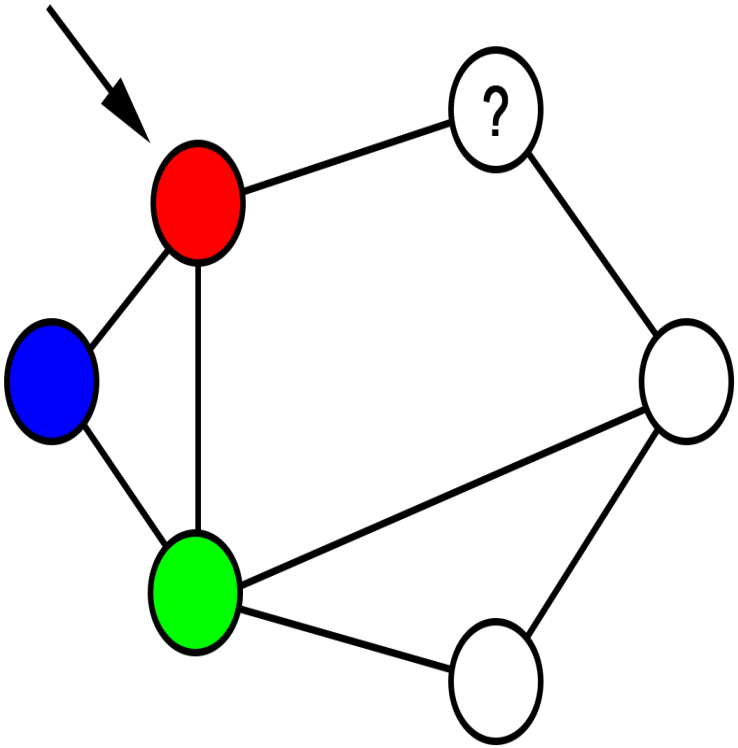
Exemple :



Couleurs utilisées : 0, 1, 2

Coloration de graphe : algorithme glouton

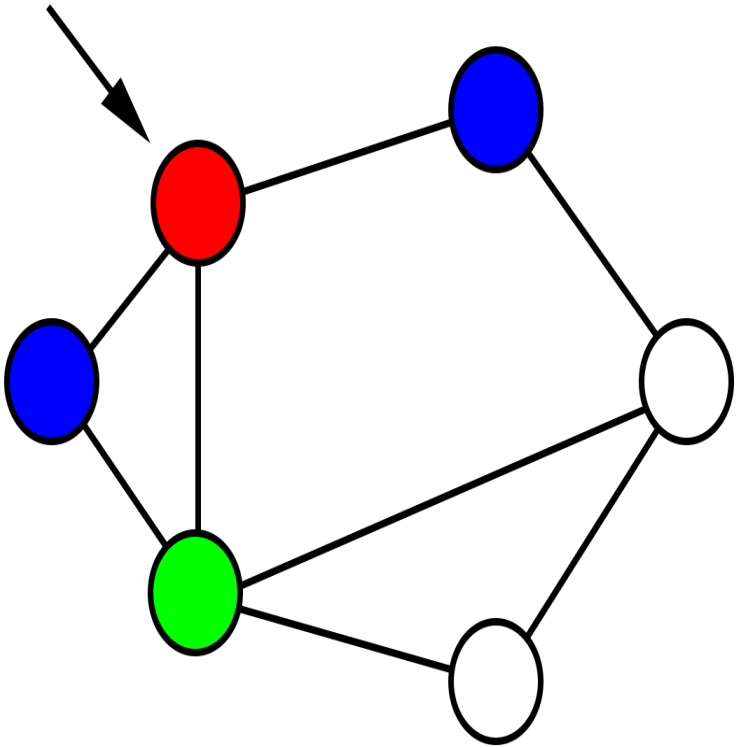
Exemple :



Couleurs utilisées : 0, 1, 2

Coloration de graphe : algorithme glouton

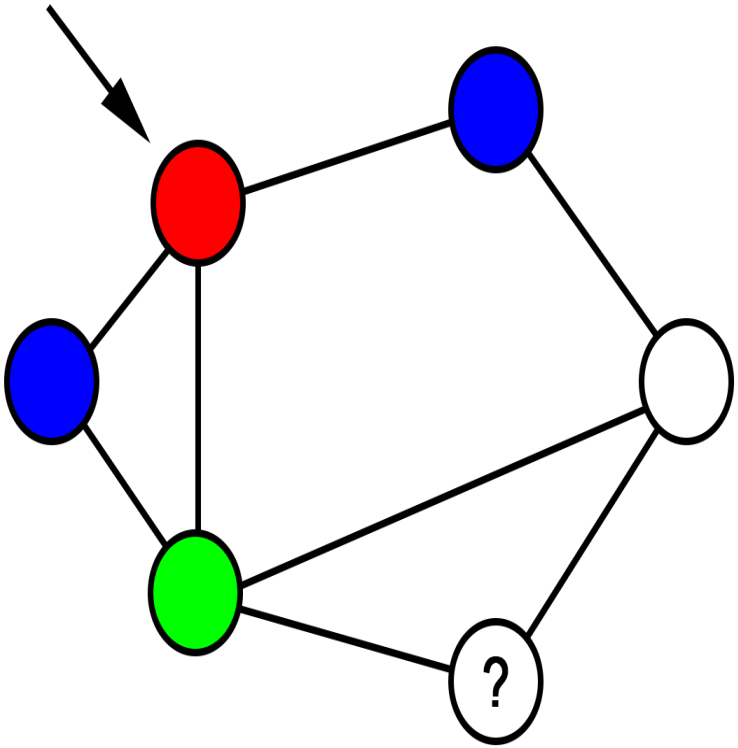
Exemple :



Couleurs utilisées : 0, 1, 2

Coloration de graphe : algorithme glouton

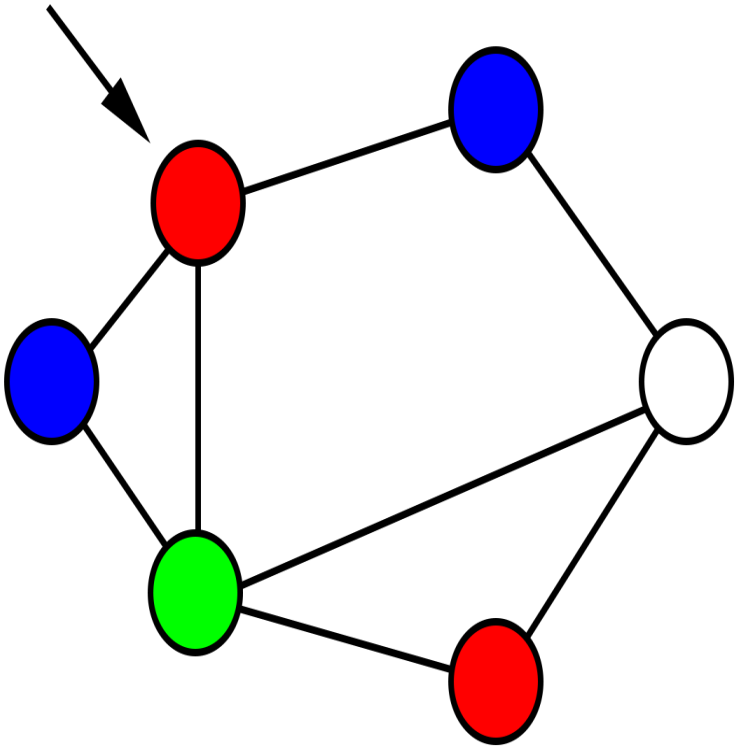
Exemple :



Couleurs utilisées : 0, 1, 2

Coloration de graphe : algorithme glouton

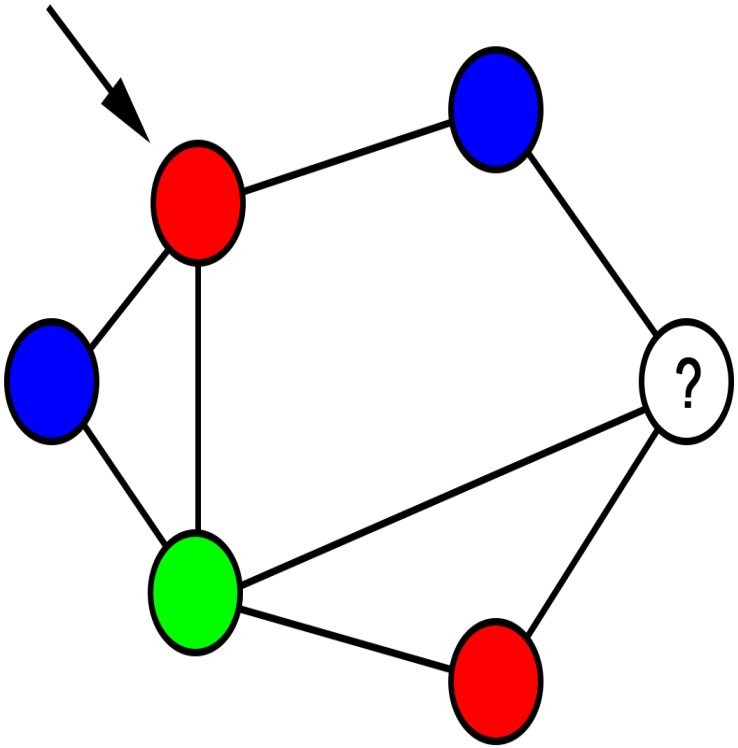
Exemple :



Couleurs utilisées : 0, 1, 2

Coloration de graphe : algorithme glouton

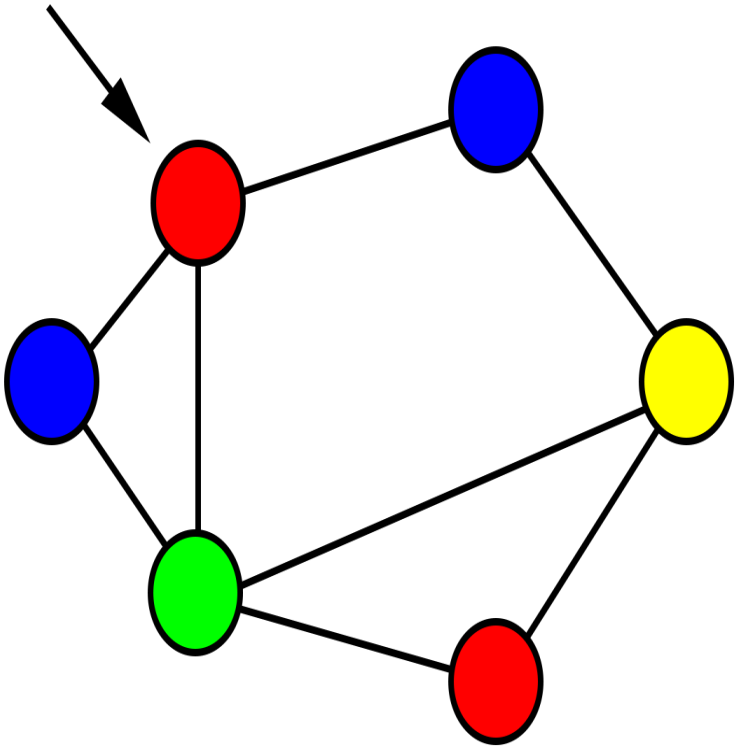
Exemple :



Couleurs utilisées : 0, 1, 2

Coloration de graphe : algorithme glouton

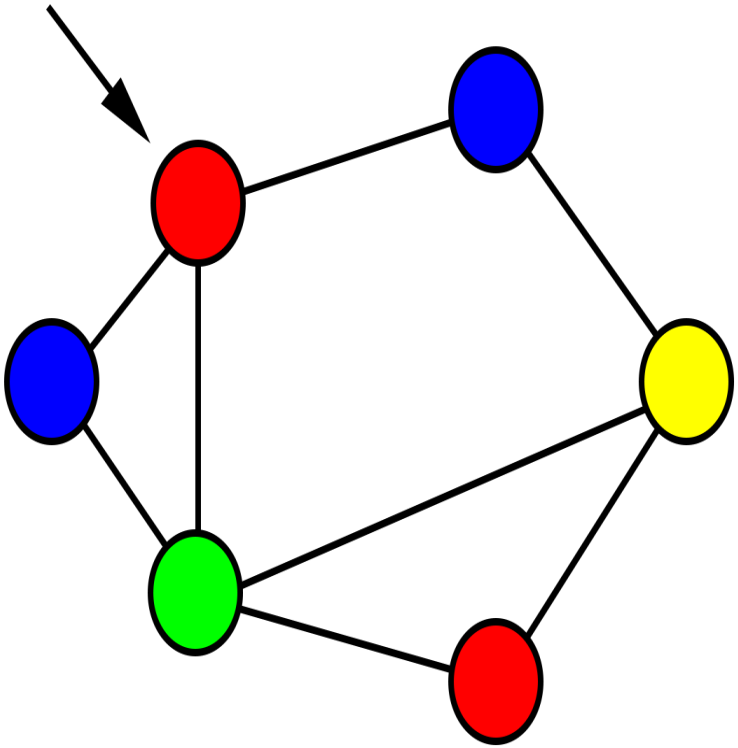
Exemple :



Couleurs utilisées : 0, 1, 2, 3

Coloration de graphe : algorithme glouton

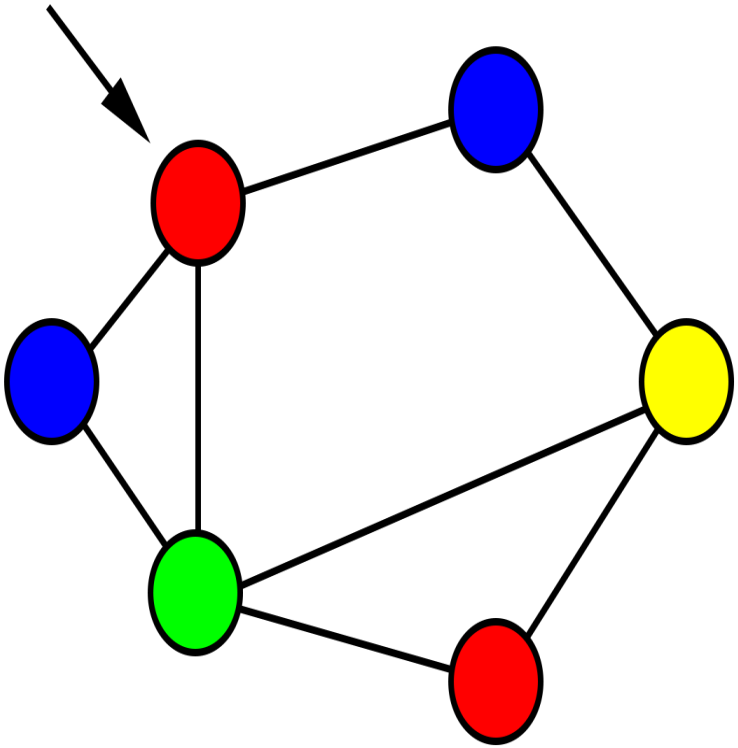
Exemple :



Couleurs utilisées : 0, 1, 2, 3 → 4 couleurs

Coloration de graphe : algorithme glouton

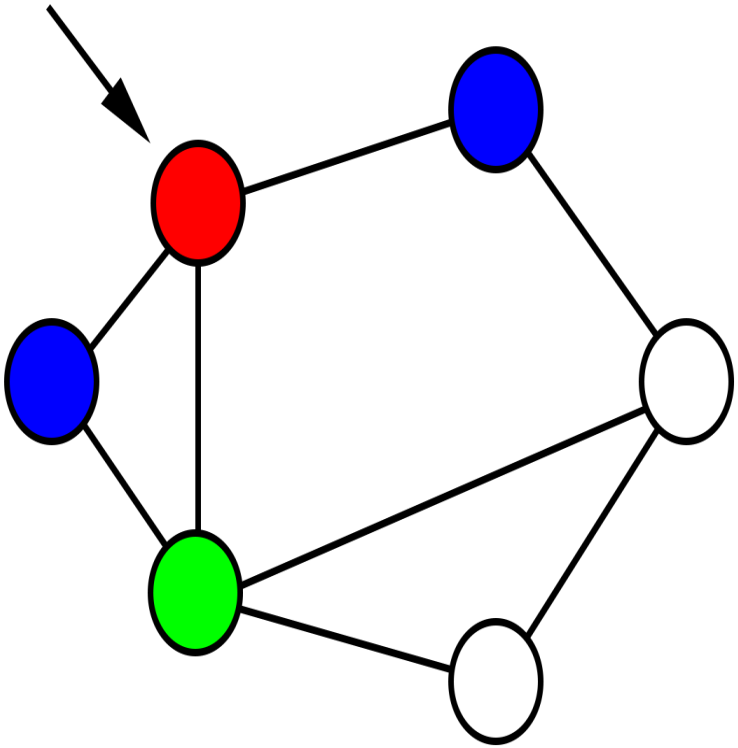
Exemple :



Et pourtant…

Coloration de graphe : algorithme glouton

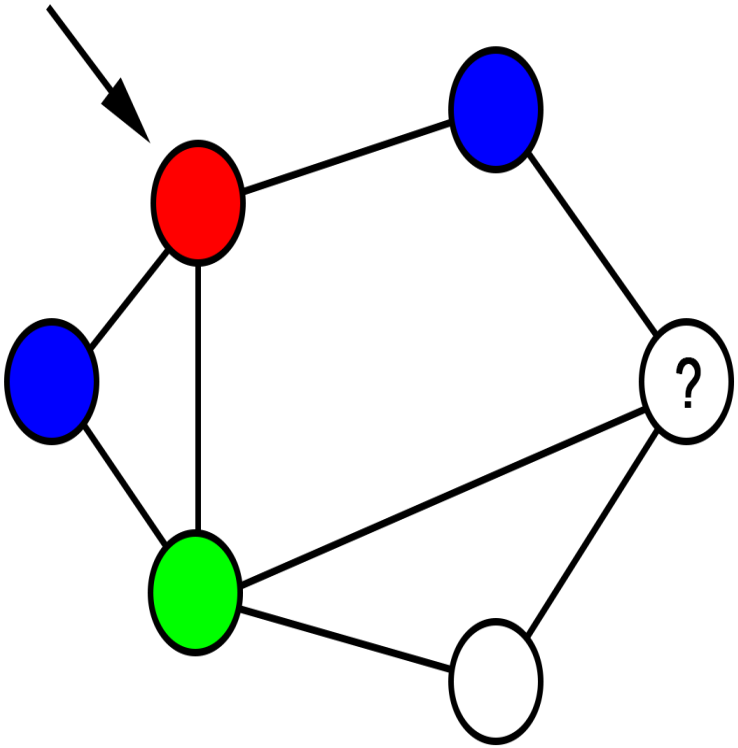
Exemple :



Couleurs utilisées : 0, 1, 2

Coloration de graphe : algorithme glouton

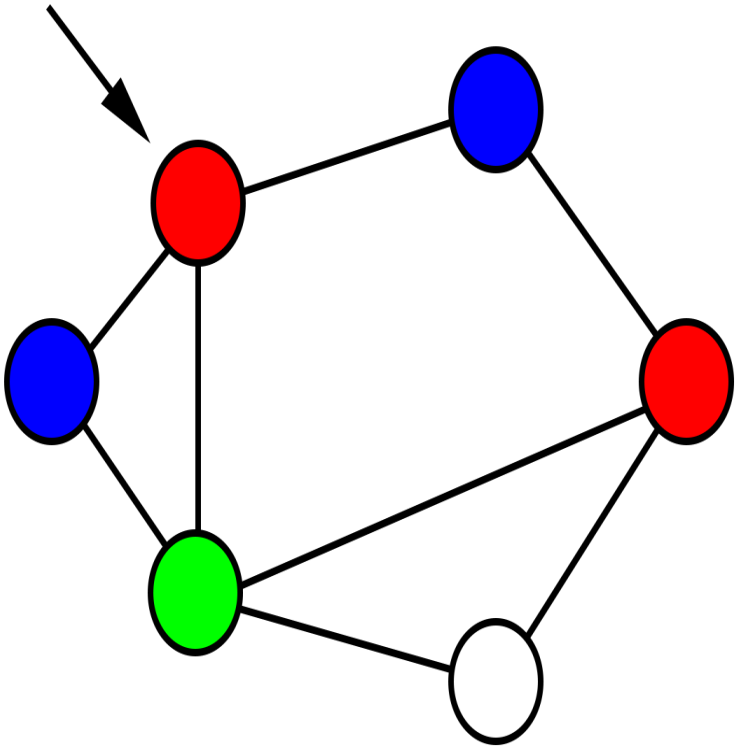
Exemple :



Couleurs utilisées : 0, 1, 2

Coloration de graphe : algorithme glouton

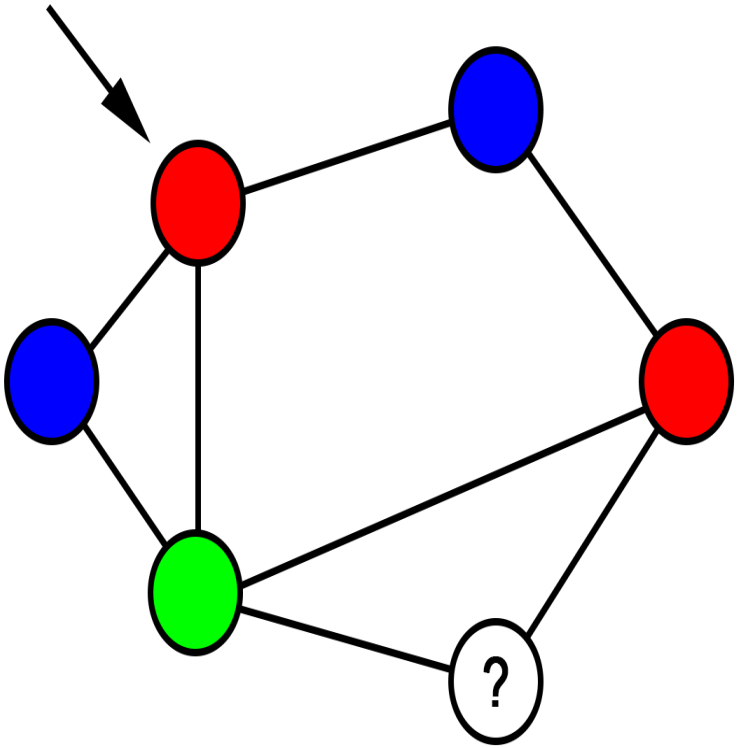
Exemple :



Couleurs utilisées : 0, 1, 2

Coloration de graphe : algorithme glouton

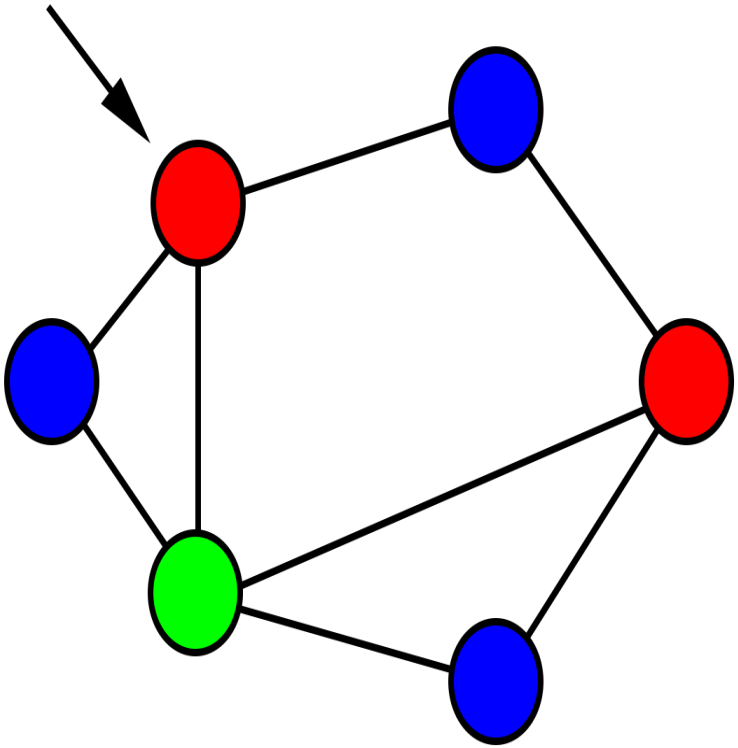
Exemple :



Couleurs utilisées : 0, 1, 2

Coloration de graphe : algorithme glouton

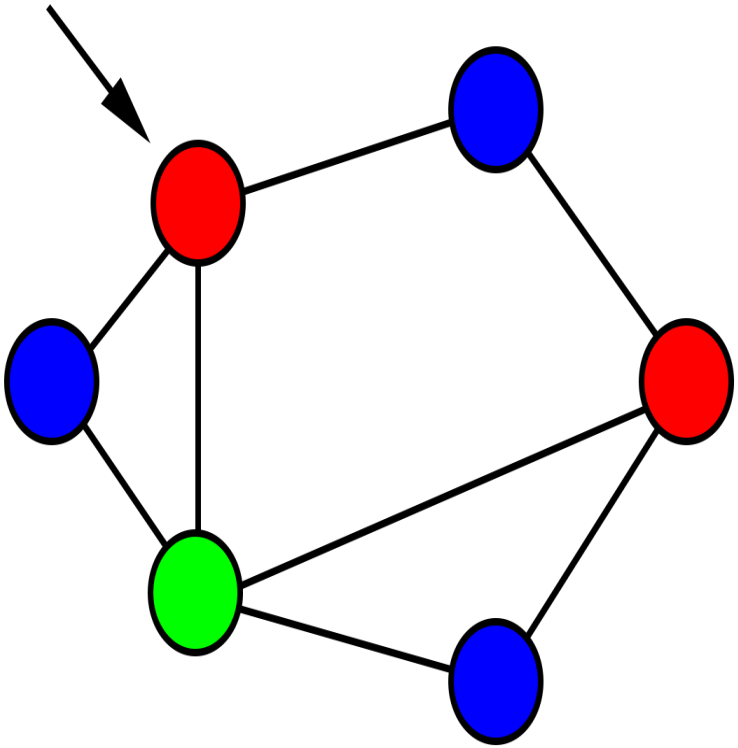
Exemple :



Couleurs utilisées : 0, 1, 2

Coloration de graphe : algorithme glouton

Exemple :



Couleurs utilisées : 0, 1, 2 → 3 couleurs

Coloration de graphe : algorithme glouton

- Problème : le nombre de couleurs dépend du nœud de départ, de la manière de parcourir le graphe…

- En pratique, il vaut mieux commencer par les sommets de plus gros degrés → Algortime de Welsh et Powell.
- Complexité : $\Theta(|V|+|E|)$

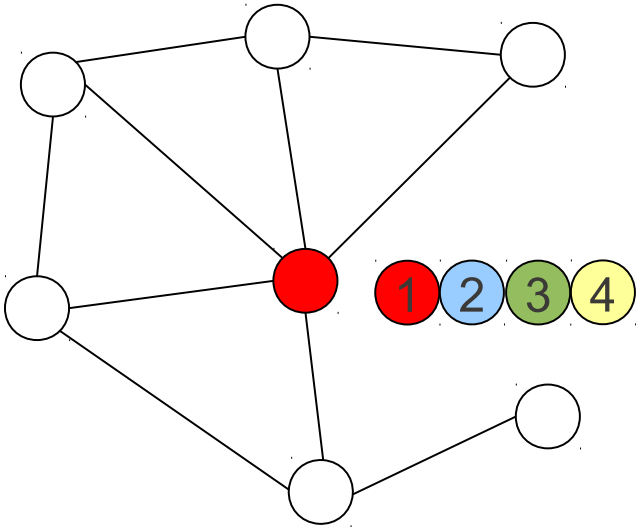
DSATUR [Brélaz 79]

DSAT(v) = degré(v) si aucun voisin de v n'est colorié sinon
DSAT(v) = le nombre de couleurs différentes utilisées

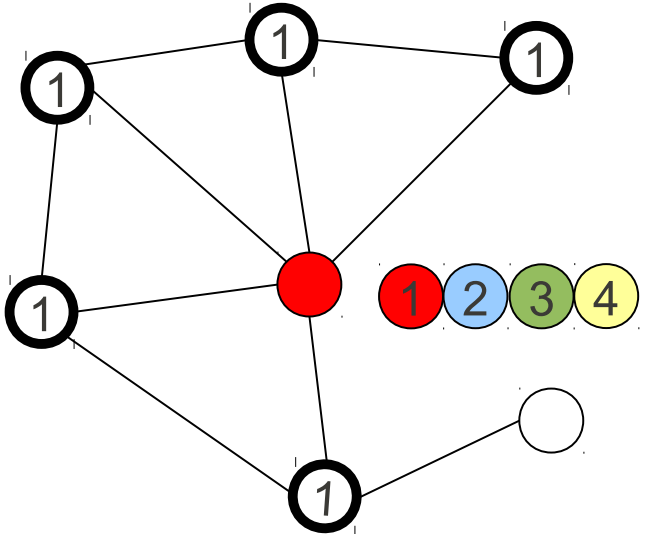
par le voisinage de v

1. Ordonner les sommets par ordre décroissant de degrés.
2. Colorer un sommet de degré maximum avec la couleur 1.
3. Choisir un sommet avec DSAT maximum. En cas d'égalité, choisir un sommet de degré maximal.
4. Colorer ce sommet avec la plus petite couleur possible
5. Si tous les sommets sont colorés alors stop. Sinon aller en 3.

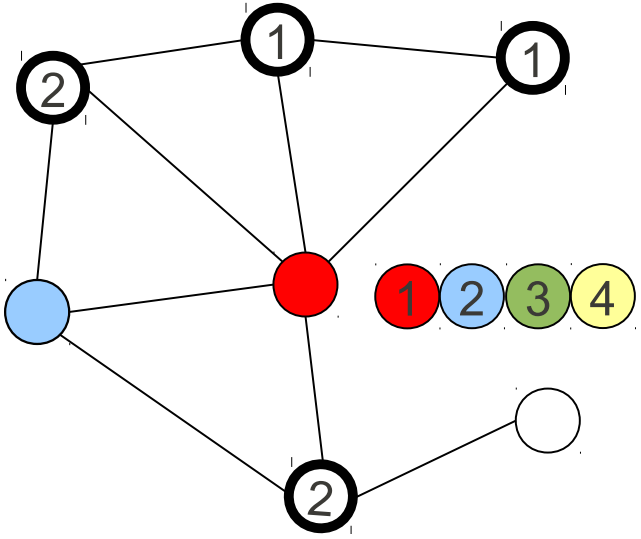
DSATUR



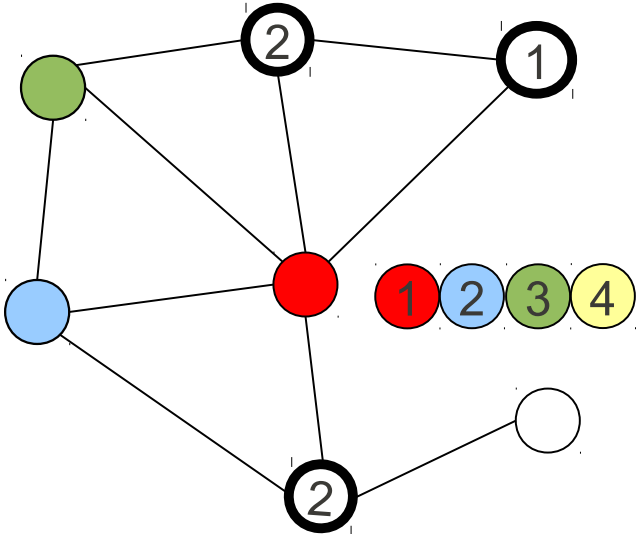
DSATUR



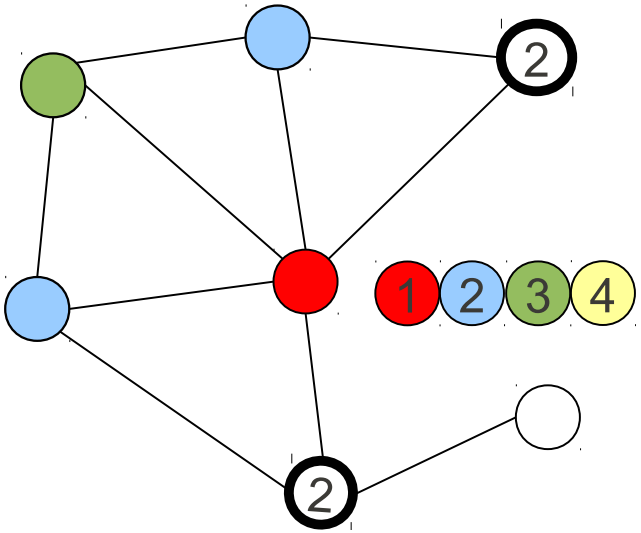
DSATUR



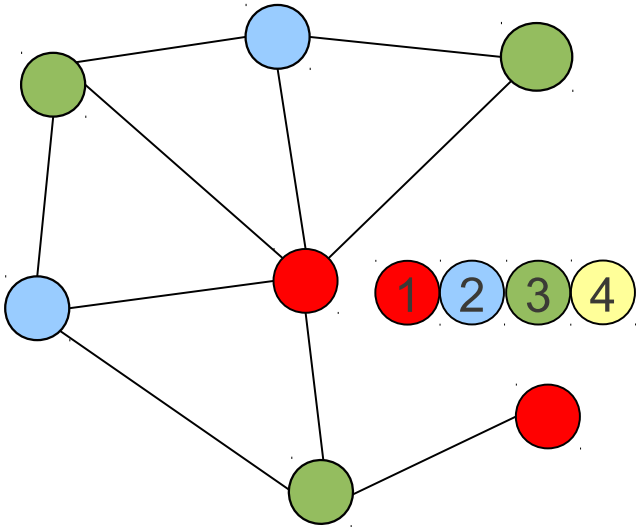
DSATUR



DSATUR



DSATUR

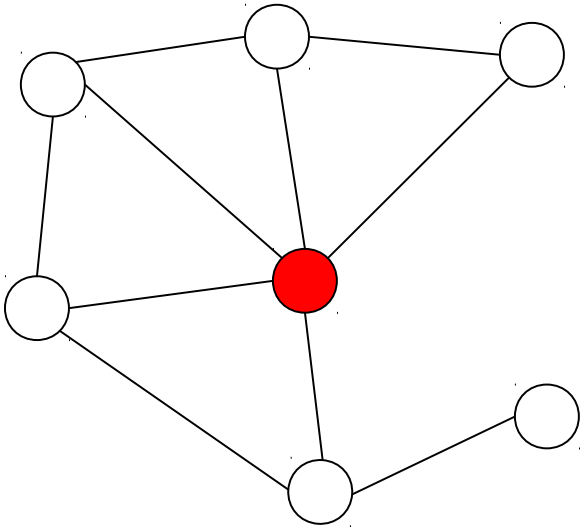


RLF (Recursive-Large-First) [Leighton 79]

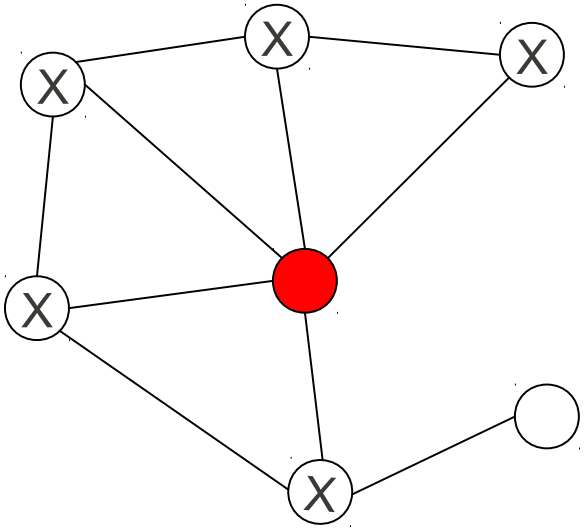
- vc(x, y) = |E(G)| − |E(G \ x, y)| : nombre de voisins communs à x et à y
- 1. choisir un sommet x de degré maximum
- 2. choisir un sommet y non voisin de x tel que vc(x, y) est maximum et contracter (fusionner) y en x
- 3. répéter 2. jusqu'à que x soit voisin de tous les autres sommets
- 4. enlever le sommet x et reprendre à l'étape 1.

⇒ tous les sommets contractés en x feront parti de la même classe de couleur que x (auront la même couleur)

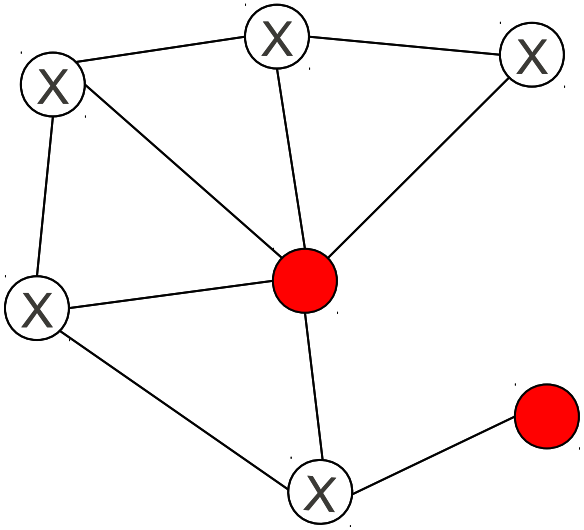
RLF (Recursive-Large-First)



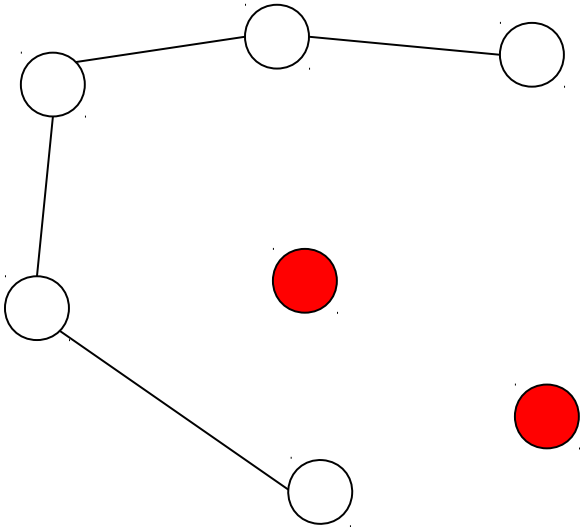
RLF (Recursive-Large-First)



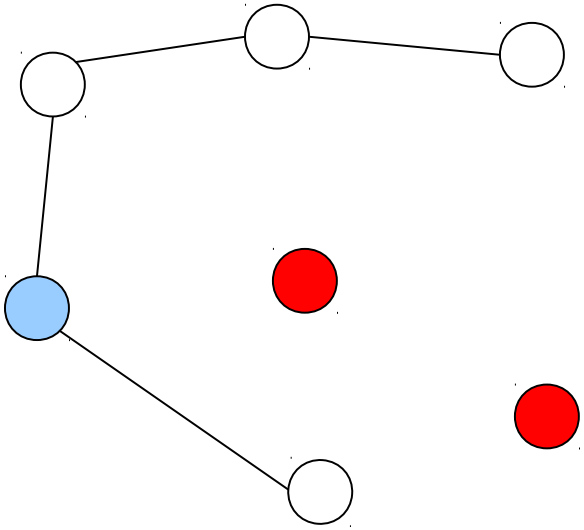
RLF (Recursive-Large-First)



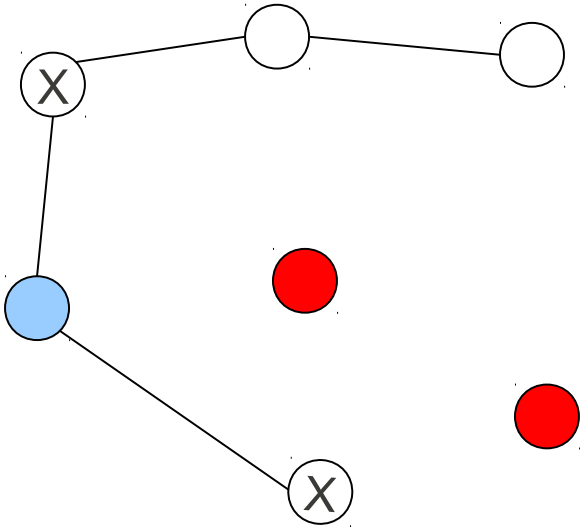
RLF (Recursive-Large-First)



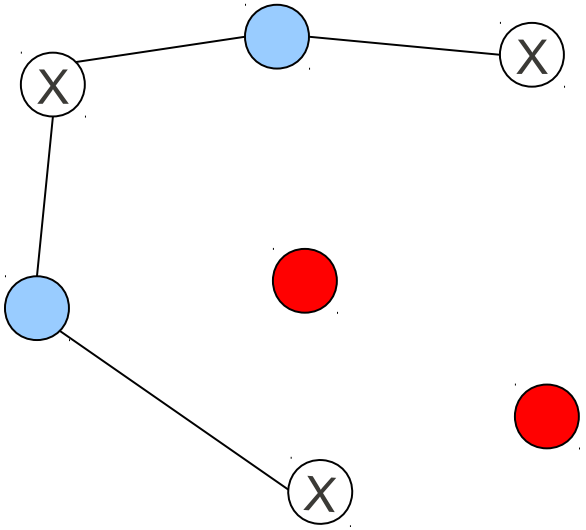
RLF (Recursive-Large-First)



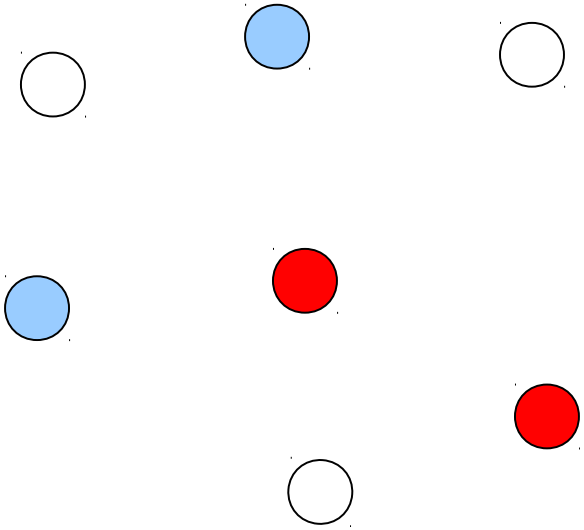
RLF (Recursive-Large-First)



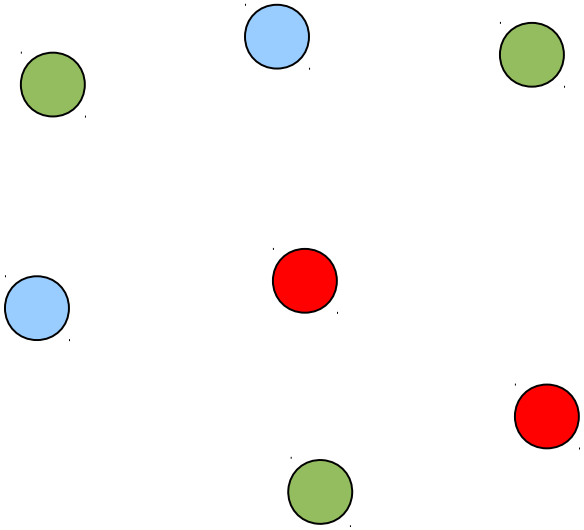
RLF (Recursive-Large-First)



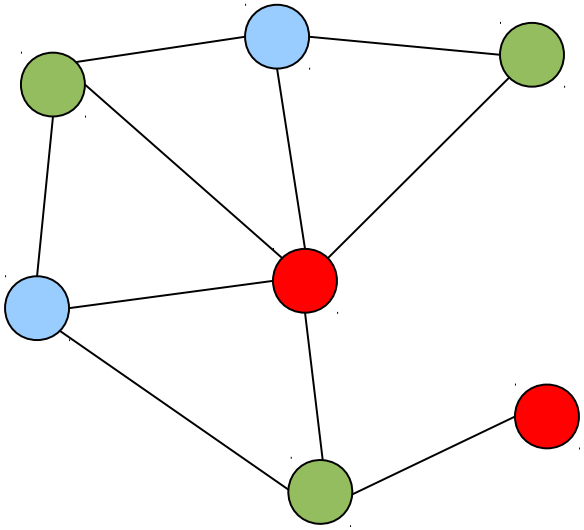
RLF (Recursive-Large-First)



RLF (Recursive-Large-First)

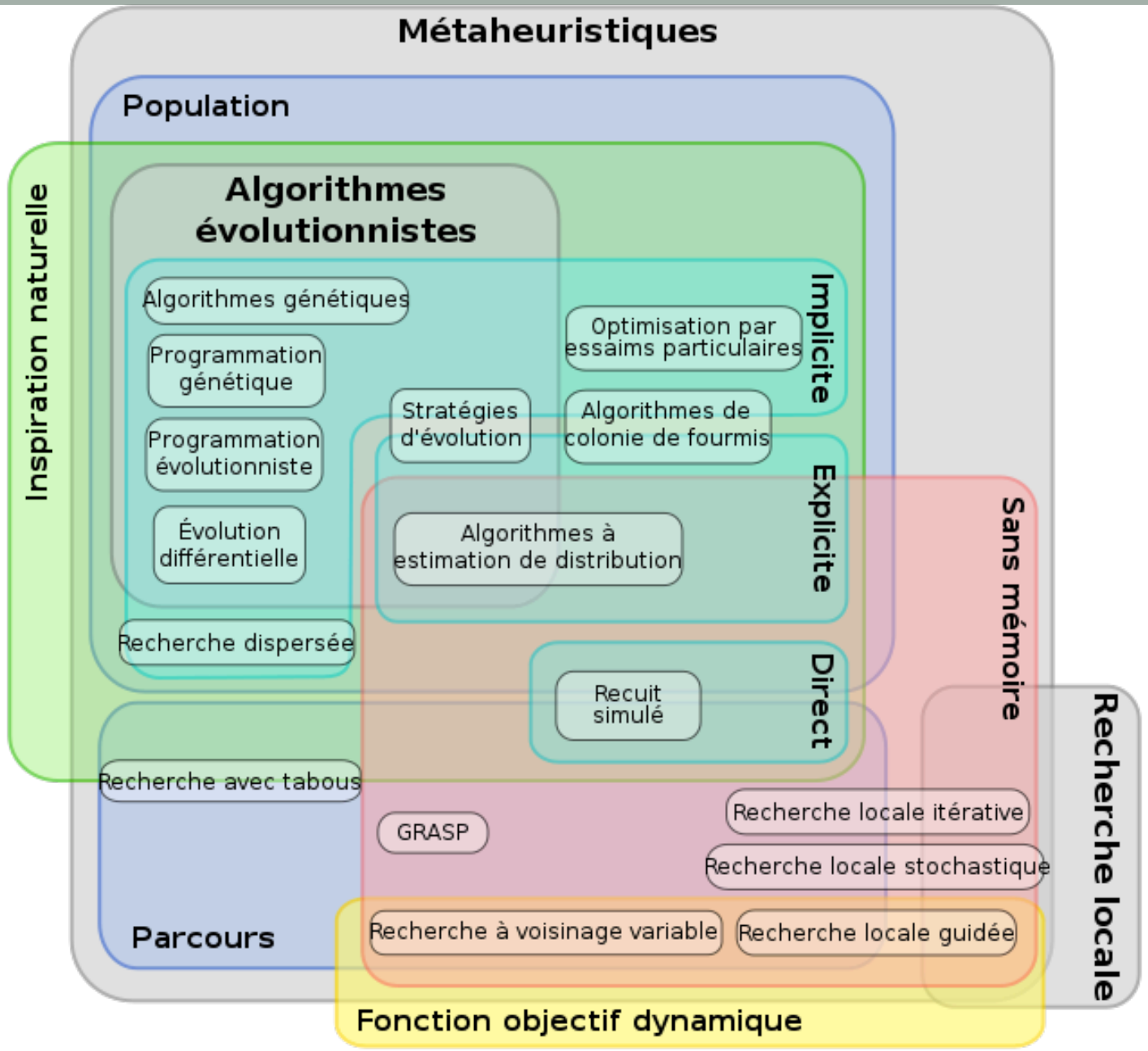


RLF (Recursive-Large-First)

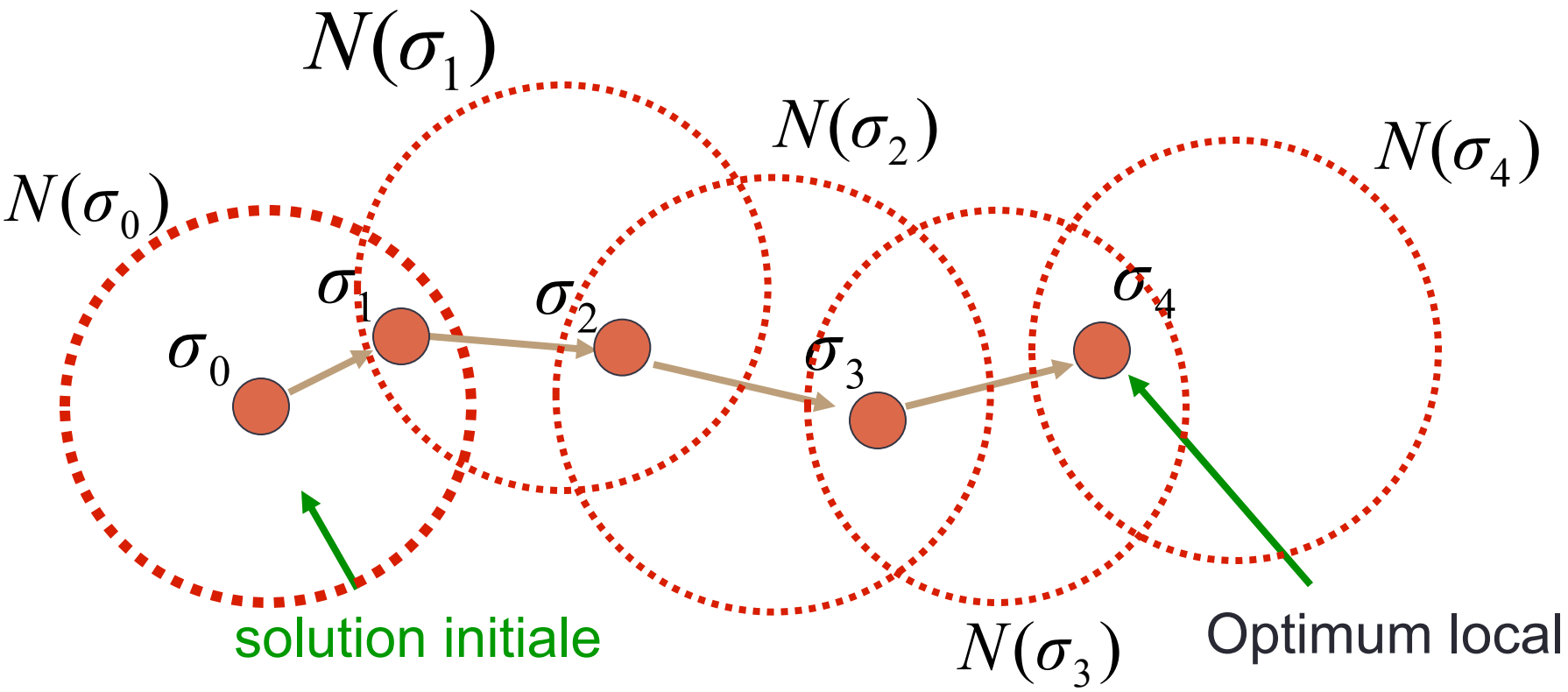


Heuristiques et Métaheuristiques

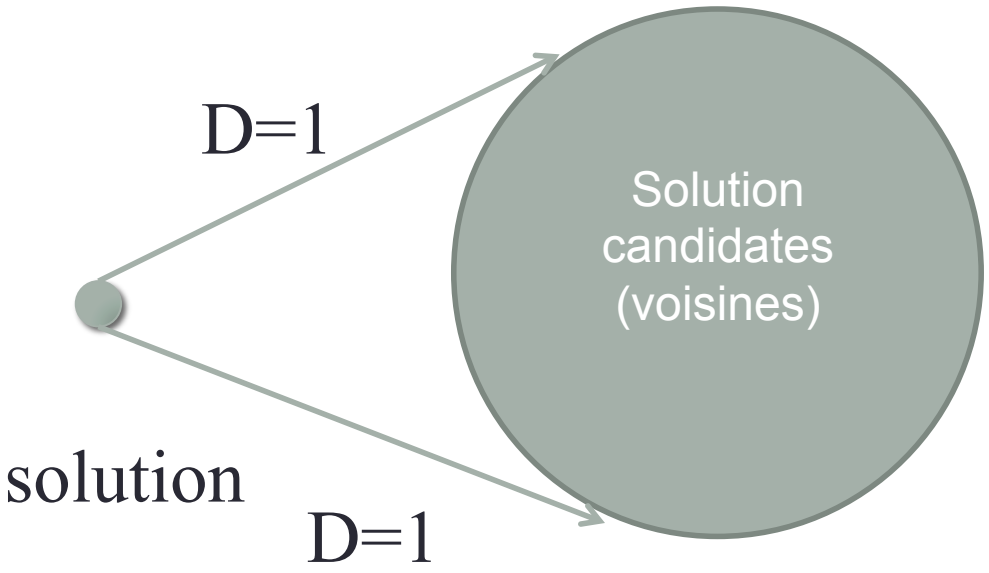
- Heuristique = algorithme qui fournit rapidement (en temps polynomial) une solution réalisable, mais pas nécessairement optimale, pour un problème d'optimisation difficile.
- Métaheuristiques = famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficile pour lesquels on ne connaît pas de méthode classique plus efficace. Ces méthodes utilisent cependant un haut niveau d'abstraction, leur permettant d'être adaptées à une large gamme de problèmes différents. Les métaheuristiques les plus connues sont la recherche avec tabous, le recuit simulé, les algorithmes génétiques et les colonies de fourmis.


Métaheuristiques
Population
Inspiration naturelle
Algorithmes évolutionnistes
Algorithmes génétiques
Programmation génétique
Programmation évolutionniste
Évolution différentielle
Implicite
Optimisation par essaims particulaires
Stratégies d'évolution
Algorithmes de colonie de fourmis
Explicite
Algorithmes à estimation de distribution
Recherche dispersée
Direct
Recuit simulé
Sans mémoire
Recherche locale
Recherche avec tabous
GRASP
Recherche locale itérative
Recherche locale stochastique
Parcours
Recherche à voisinage variable
Recherche locale guidée
Fonction objectif dynamique

Recherche locale



Principe : comment et quand choisir ?

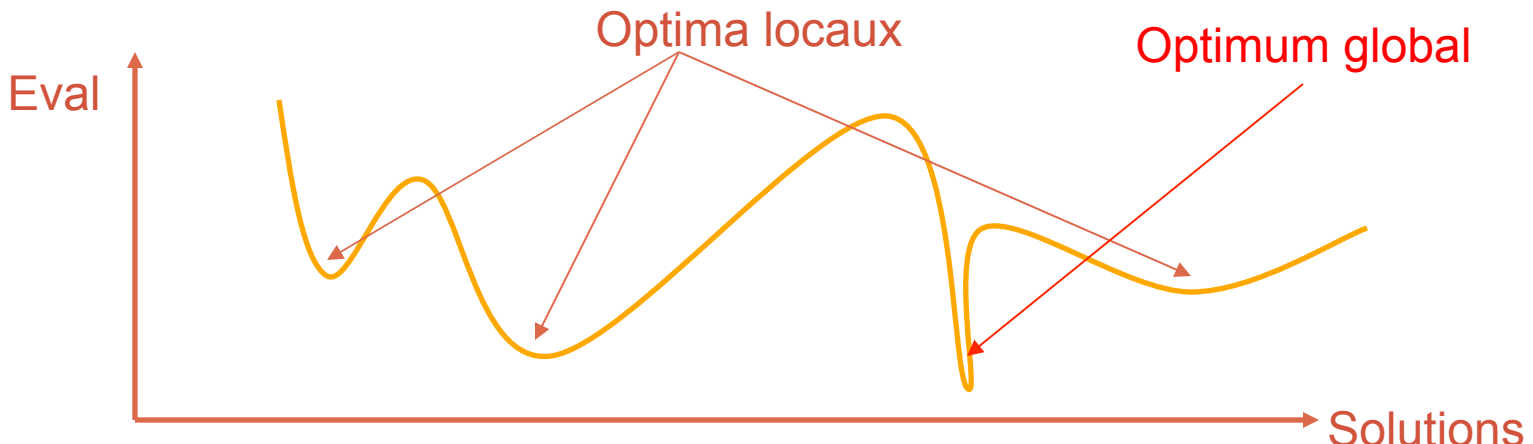


Stratégies

FIRST IMPROVEMENT	BEST IMPROVEMENT
Choisit le premier voisin améliorant la qualité de la solution.	Evalue la qualité de tous les voisins et choisit le meilleur.

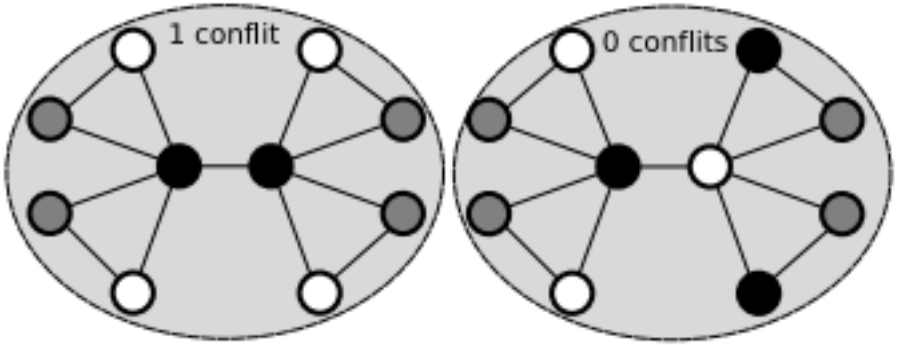
Recherche locale

- Facile à appliquer (représentation, fonction d'évaluation, voisinage).
- Converge uniquement vers des optima locaux
- L'optimum local trouvé dépend de la solution initiale (HC itéré)
- Pas de démarche générale pour borner l'erreur relative / optimum global
- Impossible d'avoir une borne supérieure sur le temps d'exécution



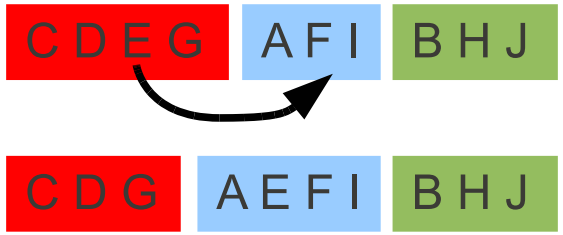
Recherches locales ou à voisinage pour la coloration de graphe

- Chaque coloration = un point dans un espace de recherche
- Les points de l'espace sont évalués par la fonction objectif:
  - Le nombre de conflits = le nombre d'arêtes avec les deux extrémités de la même couleur



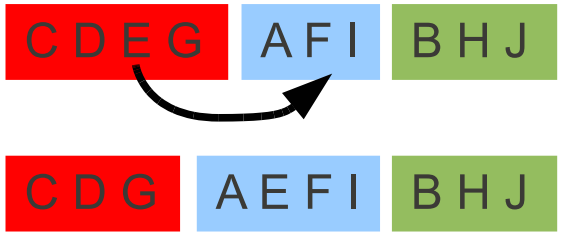
Recherches locales ou à voisinage pour la coloration de graphe

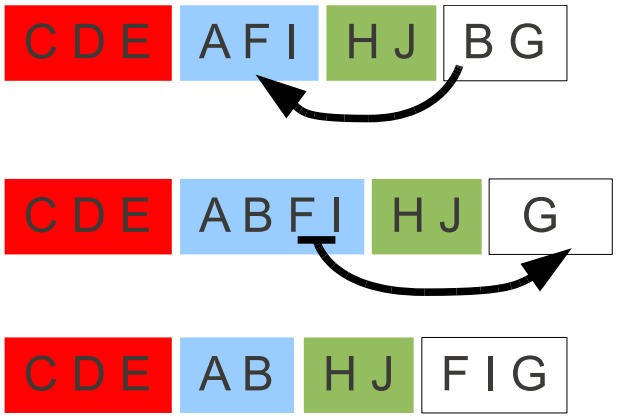
- Démarre d'un coloriage aléatoire
- La recherche locale passe d'une coloration à l'autre en changeant une couleur en cherchant à « optimiser » (ici réduite le nombre de conflits)
- Exemple : 1-move



Recherches locales ou à voisinage pour la coloration de graphe

Search space	k not fixed		k fixed	
	legal colorings	colorings	k-colorings	partial legal k-colorings
Objective function	$-\sum_{i=1}^{k} \lvert V_i \rvert^2$	$\sum_{i=1}^{k} \lvert V_i \rvert (2\lvert E_i \rvert - \lvert V_i \rvert)$	$\sum_{i=1}^{k} \lvert E_i \rvert$	$\sum_{v \in V_{k+1}} d(v)$
Neighborhood structure	Kempe chain interchanges	1-moves	critical 1-moves	i-swaps Kempe chain interchanges
Authors	Morgenstern and Shapiro (1988) Johnson et al. (1991)	Johnson et al. (1991)	Chams et al. (1987) Hertz and de Werra (1987) Johnson et al. (1991)	Morgenstern (1996) Bloechliger and Zufferey (2003)
Search strategy	legal	penalty	k-fixed penalty	k-fixed partial legal

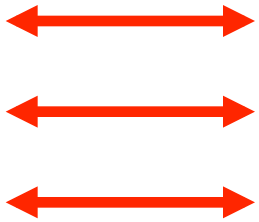




Algorithmes évolutionnaires
Métaphore

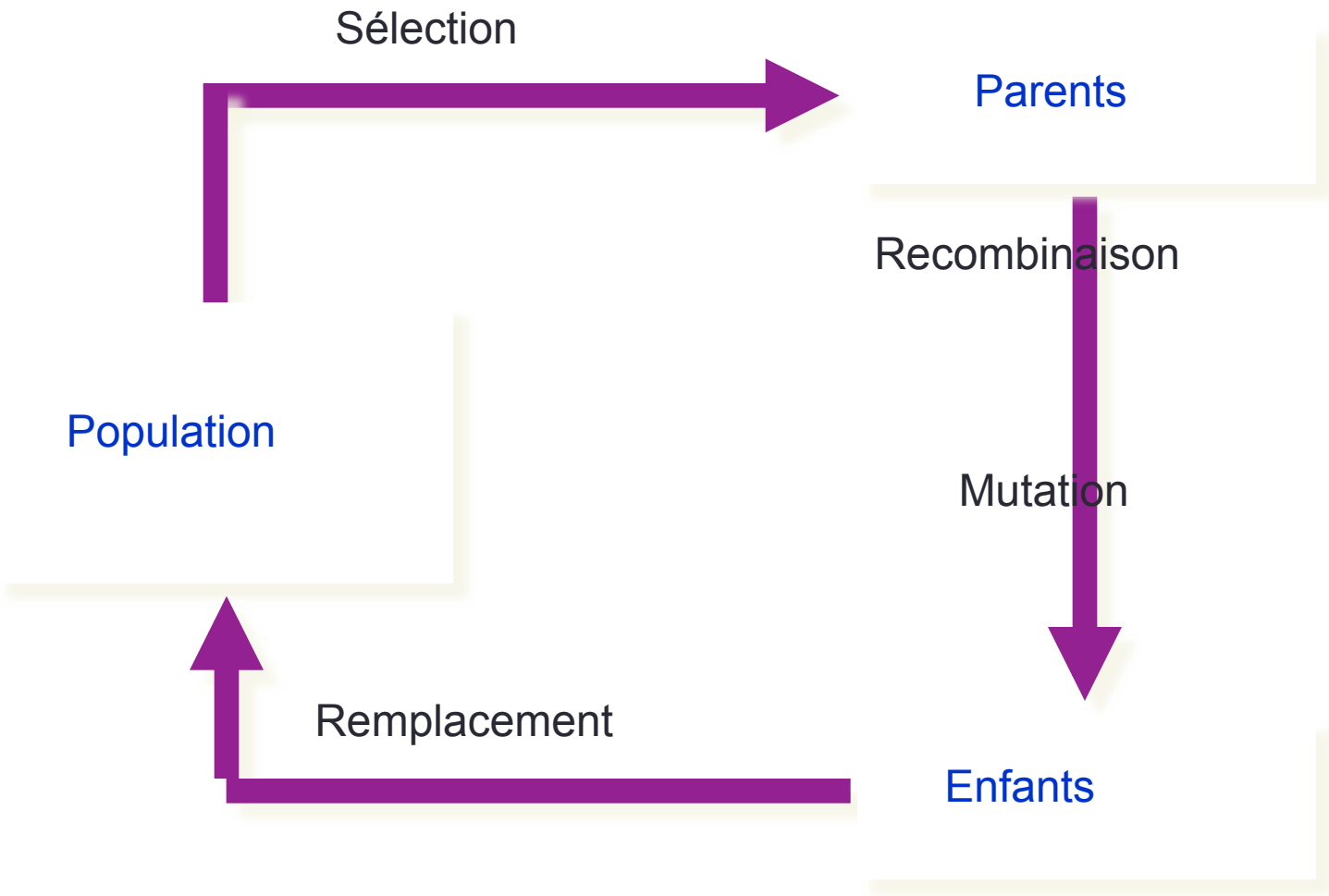
- EVOLUTION
  - Individu
  - Adaptation
  - Environnement

- PROBLEME D' OPTIMISATION
  - Solution Candidate
    - Qualité
    - Problème



- Basés sur l'évolution d'une population de solutions
- Méthodes de recherche locale et constructives basées sur une solution

Algorithmes évolutionnaires
Cycle de l'évolution



Algorithmes évolutionnaires

```
Procédure Algorithme évolutionnaire
    génération t =0 ;
    Initialiser la population P(t) ;
    Evaluer la population P(t) ;
    Répéter
        Sélection P(t+1) à partir de P(t) ;
        Reproduire P(t+1) ;
        Evaluer P(t+1) ;
        Remplacer P(t+1) ;
        t=t+1 ;
    Jusqu'à critère d'arrêt ;
```

Algorithmes évolutionnaires : Etapes à suivre

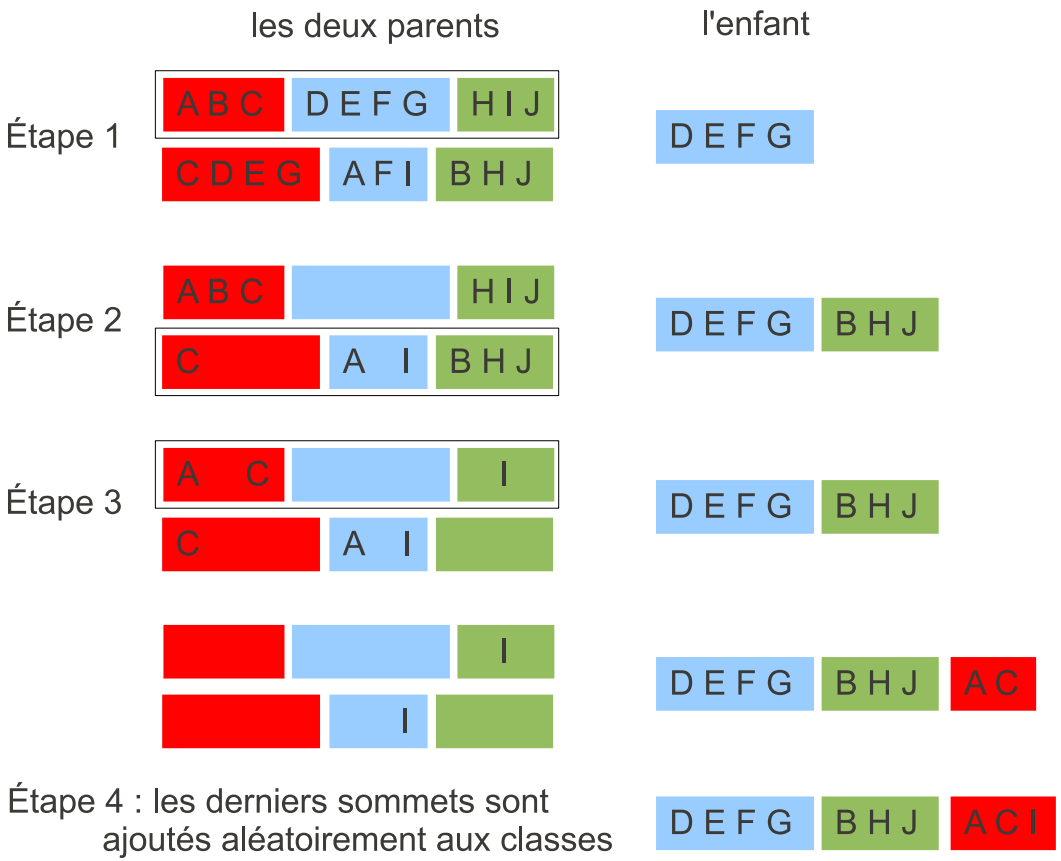
- Dans la conception d'un algorithme évolutionnaire, il y a plusieurs étapes à réaliser :
  - Construire une représentation
  - Comment initialiser la population
  - Comment évaluer un individu
  - Opérateur(s) de mutation adapté
  - Opérateur(s) de recombinaison adapté
  - Comment gérer la population
  - Comment sélectionner les parents
  - Comment remplacer les individus
  - Critère d'arrêt

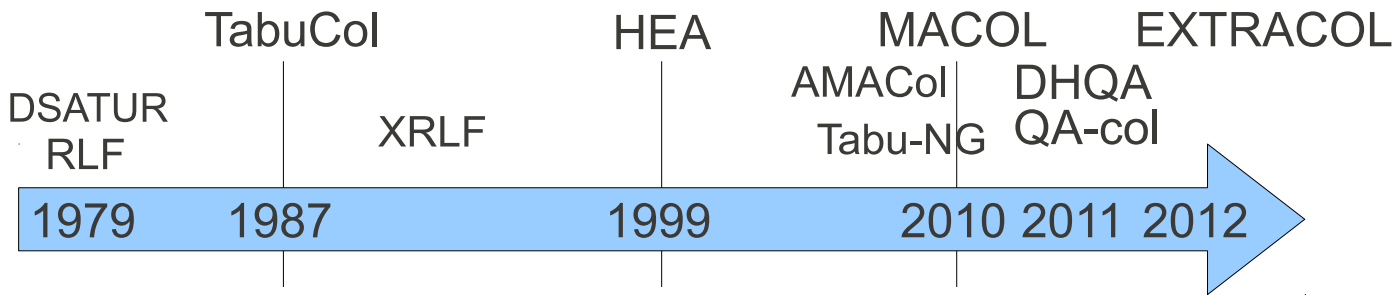
Algorithmes évolutionnaires pour la coloration de graphes

- Croisements fondés sur l'ordre de coloration des sommets [Davis 91]
- Croisements fondés sur l'affectation de couleurs : uniforme... [Costa et al. 95] → mauvais résultats
- Croisements fondés sur les classes de couleurs : GPX - Greedy Pertition Crossover [Galinier et Hao 99]

Greedy Pertition Crossover dans HEA [Galinier et Hao 99]

- Stratégie de pénalisation à k fixe
- Idée : les classes de grand cardinal doivent être transmises à l'enfant.




TabuCol
HEA
MACOL
EXTRACOL
AMACol
DHQA
DSATUR
RLF
XRLF
Tabu-NG
QA-col
1979
1987
1999
2010
2011
2012

Conclusion

- De nombreux problèmes se réduisent en une même classe de problèmes d'optimisation
  - Problème de coloration de graphe
  - Problème de satisfaction de constrainte
  - Problème de packing
  - …
- Différents types d'algorithmes pour les résoudre
  - Méthodes exactes
  - Algorithmes gloutons
  - Métaheuristiques
  - …