**Master Sciences, Technologies, Santé**
**Mention Mathématiques - spécialité Enseignement des mathématiques**

**Algorithmique et graphes, thèmes du second degré**

# Éléments de Théorie des Graphes

## Coloration de graphes
## Chemins de moindre coût

**Éric SOPENA, eric.sopena@labri.fr**



## Quatrième partie

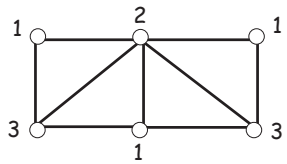
# Coloration de graphes



## k-coloration (1)

Une **k-coloration** (propre) d'un graphe (simple, sans boucles) $G$ est une application c qui associe à chaque sommet de $G$ une couleur de l'ensemble {1, …, k} de façon telle que les sommets voisins ont des couleurs distinctes :

$$\{u, v\} \in E(G) \Rightarrow c(u) \neq c(v)$$



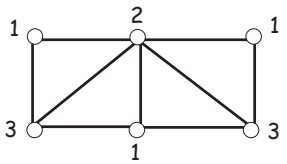



## k-coloration (2)

De façon équivalente, une k-coloration de $G$ est une "partition" de l'ensemble $V(G)$ des sommets de $G$ en k ensembles *stables* (ou *indépendants*) :

$$V(G) = V_1 \cup ... \cup V_k$$

($V_i$ = ensemble des sommets coloriés i)



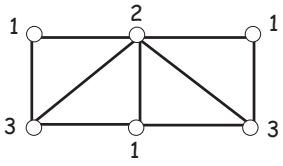



## Nombre chromatique

Un graphe $G$ est **k-coloriable** s'il peut être colorié avec k couleurs.

Le **nombre chromatique** d'un graphe $G$, noté $\chi(G)$, est le nombre minimum de couleurs nécessaire pour colorier $G$.

$\chi(G) = 3$




Un graphe $G$ est **k-chromatique** ssi $\chi(G) = k$.

$G$ k-chromatique ssi $G$ k-coloriable et non (k-1)-coloriable



## Algorithme de coloration First-Fit (1)

Un algorithme "simple" de coloration est l'algorithme *glouton* suivant :

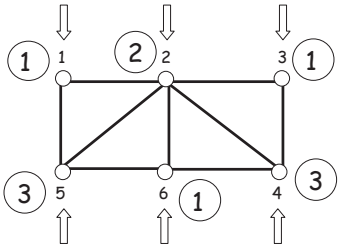
- ordonner les sommets de $G$ : $V(G) = \{ v_1, ..., v_n \}$
- traiter les sommets dans cet ordre, en affectant au sommet $v_i$ la plus petite couleur possible (i.e. distincte des couleurs de ses voisins déjà coloriés)

**Remarque.** Déterminer si un graphe est k-coloriable est un problème NP-complet pour $k \geq 3$.

**Algorithme de Welsh et Powell :** ordonner les sommets par degrés décroissants...

## Algorithme de coloration First-Fit (2)

**Exemple :**




**Question.** Quel est le nombre maximum de couleurs utilisées par cet algorithme ?



## Encadrement du nombre chromatique (1)

**Proposition.** Pour tout graphe $G$, $\chi(G) \leq \Delta(G) + 1$.

*Découle de l'algorithme First-Fit...*

**Théorème [Brooks, 1941].** Soit $G$ un graphe connexe de degré maximal $\Delta(G)$. Nous avons alors $\chi(G) = \Delta(G) + 1$ si et seulement si

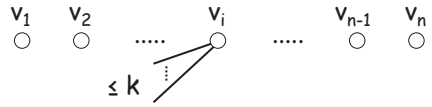
- $\Delta(G) = 2$ et $G$ est un cycle impair ou
- $\Delta(G) \geq 2$ et $G$ est le graphe complet à $\Delta(G) + 1$ sommets.

(Dans le cas contraire, nous avons $\chi(G) <= \Delta(G)$).



## Encadrement du nombre chromatique (2)

On peut essayer d'ordonner les sommets de $G$ de façon à minimiser le nombre d'arêtes "partant vers la gauche" à partir d'un sommet :



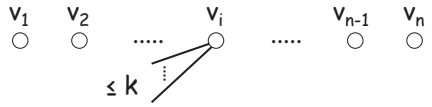

**Définition.** Un graphe $G$ est **k-dégénéré** si tout sous-graphe de $G$ contient un sommet de degré au plus k.

***Remarque***. *Les arbres sont 1-dégénérés...*



## Encadrement du nombre chromatique (3)

On peut essayer d'ordonner les sommets de $G$ de façon à minimiser le nombre d'arêtes "partant vers la gauche" à partir d'un sommet :




**Théorème (Halin, 1967).** Si $G$ est k-dégénéré alors $\chi(G) \leq k + 1$.



## Encadrement du nombre chromatique (4)

**Proposition.** Pour tout graphe $G$, $\chi(G) \geq \omega(G)$.

*$\omega(G)$ = taille maximale d'une clique dans $G$*

**Théorème [Zykov, 1949].** Pour tout k, il existe des graphes k-chromatiques sans triangles.

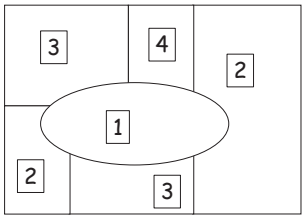
**Proposition.** Pour tout graphe $G$, $\chi(G) \geq |V(G)|/\alpha(G)$.

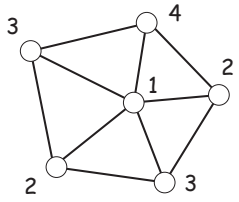
*$\alpha(G)$ = taille maximale d'un ensemble stable dans $G$*



## Le problème des quatre couleurs

**Francis Guthrie (1852).** Peut-on colorier toute carte à l'aide de quatre couleurs, de façon telle que les pays ayant une frontière commune aient des couleurs distinctes ?



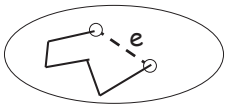



## Formule d'Euler (1)

**Formule d'Euler.** Pour tout graphe planaire connexe dessiné à n sommets, m arêtes et f faces, nous avons : **n - m + f = 2**

***Preuve.*** *Par récurrence sur m - n...*

- ***m - n = -1.*** *G est donc un arbre, f = 1 donc ok.*
- ***vrai jusqu'à m - n = k.*** *Soit G tel que m - n = k + 1 ≥ 0. Le graphe G contient donc un cycle. Soit e une arête appartenant à un cycle, et G' le graphe G-e.*




*Dans G', nous avons :*

$$n' - m' + f' = 2$$

*soit* $n - (m-1) + (f-1) = 2$

*d'où* $n - m + f = 2$ ***cqfd.***



## Formule d'Euler (2)

A l'aide de cette formule, nous pouvons notamment démontrer les propriétés suivantes :

**Proposition.** Tout graphe planaire à n sommets, n ≥ 3, possède au plus 3n - 6 arêtes.

***Preuve.***

- *Chaque face est bordée par au moins 3 arêtes, d'où f ≤ 2m/3.*
- *La formule d'Euler donne alors*

$$m + 2 = n + f \leq n + 2m/3$$

*soit* $m \leq 3n - 6$ ***cqfd.***



## Formule d'Euler (3)

**Proposition.** Tout graphe planaire contient un sommet de degré au plus 5.

***Preuve.***

*Raisonnons par l'absurde, et supposons que tous les sommets sont de degré au moins 6. Nous avons alors 2m ≥ 6n.*

*Nous savons par ailleurs que 2m ≥ 3f (toute face est bordée par au moins 3 arêtes). La formule d'Euler donne alors :*

$$2 = n - m + f \leq m/3 - m + 2m/3 = 0$$

*d'où la contradiction.* ***cqfd.***

Ainsi, tout graphe planaire est **5-dégénéré** et donc, d'après le théorème de Halin, **6-coloriable**...



## Coloration des graphes planaires

Il n'est pas trop difficile de démontrer que les graphes planaires sont 5-coloriables : preuve originale fausse des 4 couleurs due à Kempe (1879), reprise par Heawood (1890)...

Le théorème des quatre couleurs n'a été démontré qu'en 1976, à l'aide d'une preuve utilisant l'ordinateur...

**Théorème (Appel et Haken, 1976).** Tout graphe planaire est 4-coloriable.



### Cinquième partie

# Chemins de moindre coût

Algorithme de Ford-Bellman

Algorithme de Dijkstra



## Principes généraux

On considère un graphe **valué**, G = (S, A, C), avec C une fonction associant à chaque arc une valeur entière (coût).

Le **coût d'un chemin p** est alors la somme des coûts des arcs qui le composent (si tous les coûts valent 1, le coût du chemin p est la longueur de p).

Pour un tel graphe, on peut se poser les questions suivantes :

- Meilleur chemin (i.e. de coût minimal) reliant X à Y ?
- Meilleurs chemins reliant X aux autres sommets ?
- Meilleurs chemins reliant X à Y pour tous X et Y ?

## Algorithme de Ford – Bellman (1)

**Donnée :** un graphe valué G, un sommet X

**Résultat :** pour tout sommet Y, le coût d'un meilleur chemin de X à Y et un tel chemin.

**Principe.**

Chaque sommet Y est muni d'un poids L[Y] qui représente le coût du meilleur chemin de X à Y calculé jusqu'ici.

*(initialement, L[X] = 0 et L[Y] = $+\infty$ pour tout Y $\neq$ X).*

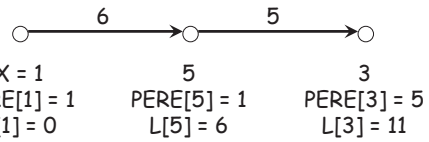
Chaque sommet Y est associé à un sommet PERE[Y] qui représente le sommet par lequel on arrive en Y par un meilleur chemin calculé jusqu'ici.

*(initialement, PERE[Y] = Y pour tout Y).*



## Algorithme de Ford – Bellman (2)

**Exemple :**




Ensuite, on parcourt les arcs pour faire des améliorations locales :




## Algorithme de Ford – Bellman (3)

**Extrait de l'algorithme :**

```
[...]
        MODIF = Vrai
        Tant que ( MODIF )
        faire   début
                MODIF = Faux
                Pour tout arc (Y, Z)
                faire   Si ( L[Z] > L[Y] + cout(Y,Z) )
                        Alors  début
                                MODIF = Vrai ; PERE [Z] = Y
                                L[Z] = L[Y] + cout(Y,Z)
                                fin
[...]
```



## Algorithme de Ford – Bellman (4)

**Questions :**

Comment peut-on parcourir les arcs ?

Cet algorithme termine-t-il toujours ?
Si non, à quelle(s) condition(s) termine-t-il ? Peut-on détecter qu'il ne va pas terminer ?



## Algo. de Ford-Bellman - Complexité

A chaque étape, on parcourt l'ensemble des arcs pour chercher des améliorations locales.

Le nombre maximum d'étapes est n car, à chaque étape, au moins un sommet prend ses valeurs définitives (L et PERE).

Ainsi, on parcourt au pire n fois l'ensemble des arcs.

Listes d'adjacence : $O(n(n + m))$

Matrice d'incidence : $O(n(n + n^2)) = O(n^3)$

*Cette complexité peut être améliorée...*

**n = nombre de sommets, m = nombre d'arcs**



## Algorithme de Dijkstra (1)

**Donnée :** un graphe valué **positivement** G, un sommet X

**Résultat :** pour tout sommet Y, le coût du meilleur chemin de X à Y et un tel chemin.

**Principe.**

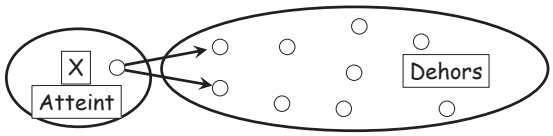
Comme pour l'algorithme de Ford-Bellman, on associe des valeurs L et PERE à chaque sommet

On associe de plus à chaque sommet un état pouvant prendre les valeurs Dehors, Atteint ou Dedans :

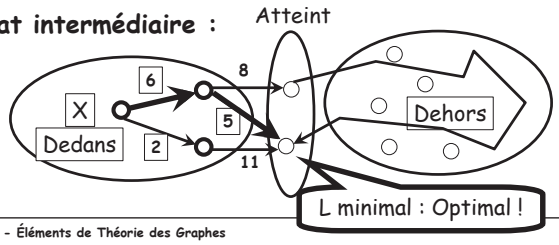
- Dehors : L et PERE n'ont pas encore été modifiés
- Atteint : L et PERE ont été modifiés, pas encore optimaux
- Dedans : L et PERE sont optimaux

## Algorithme de Dijkstra (2)

**État initial :**




**État intermédiaire :**




L minimal : Optimal !



## Algorithme de Dijkstra (3)

**Principe de l'algorithme :**

Soit Y le sommet dans l'état Atteint dont l'attribut L est minimal parmi tous les sommets dans l'état Atteint.

1. Y passe dans l'état Dedans
2. On met à jour les attributs L et PERE des successeurs de Y qui ne sont pas dans l'état Dedans et ils passent dans l'état Atteint.

**Questions.**

Valeur initiale des attributs ?

Différence(s) avec l'algorithme de Ford-Bellman ?