Cours de la matière : Théorie des graphes
Pour les étudiants de la troisième année Licence Mathématiques Appliquée
Département de mathématiques
Centre Universitaire Abdelhafid Boussouf, Mila
Anné universitaire 2024/2025

Chapitre 5, Flots

Table des matières

Introduction

La recherche opérationnelle regroupe un ensemble de méthodes visant à optimiser la prise de décision dans des situations complexes. Elle permet de modéliser des problèmes issus du monde réel, d'identifier les approches de résolution les plus adaptées et d'exploiter les outils pertinents pour proposer des solutions efficaces. En tant que discipline appartenant à l'aide à la décision, elle fournit aux décideurs des techniques leur permettant d'opter pour les meilleures stratégies possibles. Comme toute théorie en constante évolution, la recherche opérationnelle ne cesse d'étendre son champ d'application à divers domaines. Certains problèmes de décision impliquent la prise en compte de plusieurs objectifs, nécessitant ainsi une analyse multicritère afin de sélectionner la solution la plus appropriée.

Parmi les outils fondamentaux de la recherche opérationnelle, la théorie des graphes joue un rôle central. En représentant des ensembles d'objets et leurs relations sous forme de graphes, elle permet d'aborder des problèmes d'optimisation majeurs tels que la recherche de chemins optimaux, l'ordonnancement des tâches, ainsi que l'étude des concepts de couverture et de domination dans les réseaux. Cette branche des mathématiques trouve des applications variées, notamment en logistique, en informatique, en télécommunications et dans le domaine des transports.

L'histoire de la théorie des graphes remonte au XVIII siècle avec les travaux de Leonhard Euler, notamment son célèbre problème des ponts de Königsberg : les habitants de la ville cherchaient à déterminer s'il était possible de traverser tous les ponts sans passer deux fois par le même et revenir au point de départ. D'autres problèmes historiques, tels que

la marche du cavalier sur l'échiquier ou le coloriage des cartes, ont également contribué à l'émergence de cette discipline.

Depuis son origine, la théorie des graphes a connu un développement considérable et s'est étendue à diverses disciplines telles que la chimie, la biologie et les sciences sociales. Dès le début du XX siècle, elle s'est imposée comme une branche à part entière des mathématiques, notamment grâce aux contributions de Konig, Menger, Cayley, Berge et Erdos.

De manière générale, un graphe permet de modéliser la structure et les connexions d'un système complexe en mettant en évidence les relations entre ses éléments. Il constitue un outil puissant pour représenter des réseaux de communication, des réseaux sociaux, des infrastructures routières, des interactions entre espèces en biologie, ou encore des circuits électriques. Aujourd'hui, la théorie des graphes demeure un domaine de recherche actif, mobilisant aussi bien des mathématiciens que des spécialistes de l'algorithmique, en raison de son importance dans la résolution de problèmes combinatoires et computationnels.

1

Cheminement

1.1 Graphe orienté

En donnant un sens aux arêtes d'un graphe, on obtient un digraphe (ou graphe orienté). Le mot « digraphe » est la contraction de l'expression anglaise « directed graph ».
Un digraphe fini G = (V,E) est défni par l'ensemble V = $\{v_1, v_2, ..., v_n\}$ dont les éléments sont appelés sommets, et par l'ensemble E = $\{e_1, e_2, ..., e_m\}$ dont les éléments sont appelés

arcs.
Un arc e de l'ensemble E est défini par une paire ordonnée de sommets. Lorsque e = (u,v), on dit que l'arc e va de u à v. On dit aussi que u est l'extrémité initiale et v l'extrémité finale de e.

1.1.1 Degré d'un sommet d'un digraphe

Soit v un sommet d'un graphe orienté.
On note $d^+(v)$ le degré extérieur du sommet v, c'est-à-dire le nombre d'arcs ayant v comme extrémité initiale.
On note $d^-(v)$ le degré intérieur du sommet v, c'est-à-dire le nombre d'arcs ayant v comme extrémité finale.
On définit le degré : $d(v) = d^+(v) + d^-(v)$

1.1.2 Chemins et circuits

Un chemin conduisant du sommet a au sommet b est une suite ayant pour éléments alternativement des sommets et des arcs, commençant et se terminant par un sommet, et telle que chaque arc est encadré à gauche par son sommet origine et à droite par son sommet destination. On ne peut donc pas prendre les arc à rebours. Sur le digraphe ci-après, on peut voir par exemple le chemin $(v_3, e_2, v_2, e_1, v_1)$. Par convention, tout chemin comporte au moins un arc.
On appelle distance entre deux sommets d'un digraphe la longueur du plus petit chemin les reliant. S'il n'existe pas de chemin entre les sommets x et y, on pose d(x,y) = ∞. Par exemple, sur le digraphe ci-dessous, $d(v_5, v_4) = 2$, $d(v_4, v_5) = \infty$, $d(v_3, v_1) = 1$.
Un circuit est un chemin dont les sommets de départ et de fin sont les mêmes. Le digraphe ci-dessus ne contient pas de circuit.
Les notions de chemins et de circuits sont analogues à celles des chaînes et des cycles pour les graphes non orientés. **Digraphe fortement connexe** Un digraphe est forte-

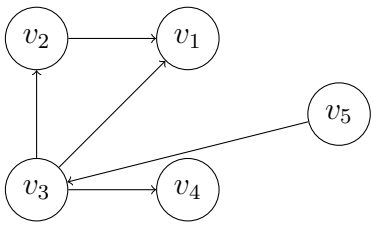


FIGURE 1.1 – G

ment connexe, si toute paire ordonnée (a,b) de sommets distincts du graphe est reliée par au moins un chemin. En d'autres termes, tout sommet est atteignable depuis tous les autres sommets par au moins un chemin.
On appelle composante fortement connexe tout sous-graphe induit maximal fortement connexe (maximal signifie qu'il n'y a pas de sous-graphe induit connexe plus grand contenant les sommets de la composante).

1.1.3 Représentations non graphiques des digraphes

Matrice d'adjacences On peut représenter un digraphe par une matrice d'adjacences. Une matrice $(n \times n)$ est un tableau de n lignes et m colonnes. (i, j) désigne l'intersection de la ligne i et de la colonne j.
Dans une matrice d'adjacences, les lignes et les colonnes représentent les sommets du graphe. Un $\ll 1 \gg$ à la position (i, j) signifie qu'un arc part de i pour rejoindre j.

Cette matrice a plusieurs caractéristiques :

1. Elle est carrée : il y a autant de lignes que de colonnes.
2. Il n'y a que des zéros sur la diagonale. Un $\ll 1 \gg$ sur la diagonale indiquerait une boucle.
3. Contrairement à celle d'un graphe non orienté, elle n'est pas symétrique.
4. Une fois que l'on fixe l'ordre des sommets, il existe une matrice d'adjacences unique pour chaque digraphe. Celle-ci n'est la matrice d'adjacences d'aucun autre digraphe.

1.2 Cheminement

On distingue deux grandes catégories de problèmes de cheminement :

— Les **problèmes de plus court chemin**, où l'on cherche un chemin minimisant une fonction de coût (ex. la distance).
— Les **problèmes de plus long chemin** (utiles en ordonnancement), souvent plus complexes car ils peuvent être NP-difficiles.

Les graphes considérés peuvent être orientés ou non, pondérés ou non. Nous nous concentrons ici sur les graphes orientés et pondérés.

2. Conditions d'existence des solutions

— Dans un graphe non pondéré ou à poids positifs, il existe toujours un plus court chemin entre deux sommets si une chaîne les relie.
— En présence de poids négatifs, il faut s'assurer qu'il n'existe pas de **cycle de poids négatif accessible**, ce qui rendrait la solution indéfinie (chemin de coût arbitrairement bas).

3. Algorithmes de résolution

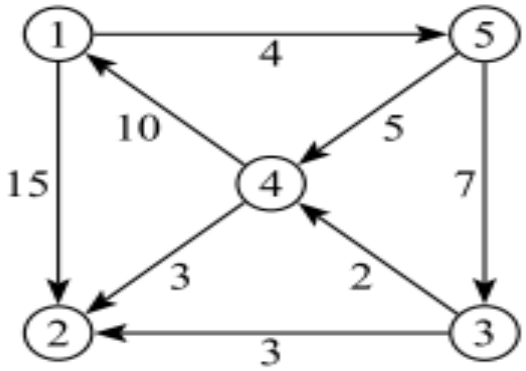
1.2.1 Algorithme de Dijkstra

Edgser Wybe Dijkstra (1930-2002) a proposé en 1959 un algorithme qui permet de calculer le plus court chemin entre un sommet particulier et tous les autres. Le résultat est une arborescence, c'est-à-dire un arbre avec un sommet particulier appelé racine. Numérotons les sommets du graphe G = (V,E) de 1 à n. Supposons que l'on s'intéresse aux chemins partant du sommet 1. On construit un vecteur $\lambda = (\lambda(1), \lambda(2), \lambda(3), ..., \lambda(n)$ ayant n composantes tel que $\lambda(j)$ soit égal à la longueur du plus court chemin allant de 1 au sommet j.
On initialise ce vecteur à c_{1j} , c'est-à-dire à la première ligne de la matrice des coûts du graphe, définie comme indiqué ci-dessous :

$$c_{ij} = \begin{cases} 0 & si \quad i = j, \\ \infty & si \quad i \neq j \quad et \quad (i,j) \quad n'appartient \quad a \quad E, \\ \delta(i,j) & si \quad i \neq j \quad et \quad (i,j) \in E. \end{cases}$$

où $\delta(i,j)$ ¿ 0 est le poids de l'arc (i,j).
On construit un autre vecteur p pour mémoriser le chemin pour aller du sommet 1 au sommet voulu. La valeur $p(i)$ donne le sommet qui précède i dans le chemin.
On considère ensuite deux ensembles de sommets, S initialisé à $\{1\}$ et T initialisé à $\{2, 3, ..., n\}$. À chaque pas de l'algorithme, on ajoute à S un sommet jusqu'à ce que S = V de telle sorte que le vecteur λ donne à chaque étape la longueur minimale des chemins de 1 aux sommets de S.
Résumé de l'algorithme de Dijkstra
On suppose que le sommet de départ (qui sera la racine de l'arborescence) est le sommet numéroté 1. Notons qu'on peut toujours renuméroter les sommets pour que ce soit le cas.
Initialisations
$\lambda(j) = c_1$ et $p(j)$ = NIL, pour $1 \leq j \leq$ n.
Pour $2 \leq j \leq$ n faire

Si $c_{1j} < \infty$ alors $p(j) = 1$.
S=$\{1\}$, T = $\{2, 3, ..., n\}$.
Itérations
Tant que T n'est pas vide faire
Choisir i dans T tel que $\lambda(i)$ est minimum
Retirer i de T et l'ajouter à S
Pour chaque successeur j de i, avec j dans T , faire
Si $\lambda(j) > \lambda(i) + \delta(i, j)$ alors $\lambda(j) = \lambda(i) + \delta(i, j)$
$p(j) = i$
Exerice
Appliquez l'algorithme de Dijkstra au graphe ci-dessous pour trouver tous les plus courts chemins en partant du sommet 5.



3.2 Algorithme de Dijkstra

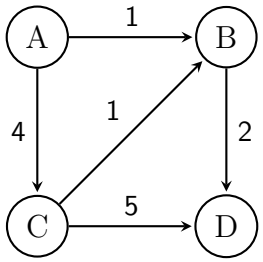
Dijkstra résout le problème du plus court chemin à partir d'une source, mais uniquement si tous les poids sont positifs.

Principe : utilisation d'un ensemble de sommets marqués (dont la distance est connue), et sélection du sommet non marqué le plus proche à chaque étape.

Complexité : $O(n^2)$ en version simple, $O(m + n \log n)$ avec file de priorité.

- Initialiser $d(s) = 0$, $d(v) = +\infty$ pour $v \neq s$; ensemble $S = \emptyset$.
- Tant que $S \neq V$:
  - Choisir $u \notin S$ tel que $d(u)$ soit minimal.
  - Ajouter u à S.
  - Pour chaque voisin v de u, faire la relaxation : si $d(u) + w(u, v) < d(v)$, alors $d(v) := d(u) + w(u, v)$

Exemple d'application :



En appliquant l'algorithme de Dijkstra depuis A, on obtient les plus courts chemins vers B, C, D en tenant compte uniquement de poids positifs.

3.1 Algorithme de Bellman-Ford

L'algorithme de Bellman-Ford permet de résoudre le problème du plus court chemin depuis une source vers tous les sommets, même en présence d'arêtes de poids négatif (mais pas de cycle négatif).

Principe : relaxation des arêtes $|V| - 1$ fois, puis test d'un cycle négatif.

Complexité : $O(nm)$ avec n sommets et m arêtes.

- Initialiser les distances : $d(s) = 0$, $d(v) = +\infty$ pour $v \neq s$
- Pour $i = 1$ à $n - 1$: pour chaque arête (u, v, w), si $d(u) + w < d(v)$, alors $d(v) := d(u) + w$
- Tester les cycles négatifs : si une amélioration est encore possible, alors il existe un cycle négatif.

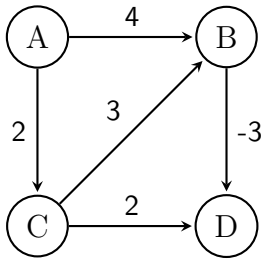
Algorithm 1: Algorithme de Bellman-Ford

Input: Un graphe orienté $G = (V, E)$ avec une fonction poids $w : E \to \mathbb{R}$, et une source $s \in V$

Output: Les distances minimales $d[v]$ depuis s vers chaque sommet $v \in V$, ou un message d'erreur si un cycle négatif est détecté

```
1  foreach v ∈ V do
2  |  d[v] ← +∞ ;                    // Initialiser les distances à l'infini
3  end
4  d[s] ← 0 ;                       // La distance de la source à elle-même est 0
5  for i ← 1 to |V| − 1 do
6  |  foreach (u, v) ∈ E do
7  |  |  if d[u] + w(u, v) < d[v] then
8  |  |  |  d[v] ← d[u] + w(u, v) ;      // Améliorer la distance si possible
9  |  |  end
10 |  end
11 end
12 foreach (u, v) ∈ E do
13 |  if d[u] + w(u, v) < d[v] then
14 |  |  erreur : cycle de poids négatif détecté;
15 |  |  return
16 |  end
17 end
18 return d[v] pour tout v ∈ V
```

Exemple d'application :



Dans ce graphe, Bellman-Ford appliqué depuis A permet de déterminer le plus court

chemin vers tous les sommets même avec l'arête (B, D) de poids négatif.