



ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

Recherche Opérationnelle R.O.

Pr. Abdessamad Kamouss

Cycle Ingénieur
ENSAM Casablanca

Contenu du module

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- Résolution des PL
- Dualité
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- Parcours eulériens et hamiltoniens des graphes
- Problème de plus court chemin
- Problèmes de transport et de flots



Introduction à la recherche opérationnelle

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- Résolution des PL
- Dualité
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- Parcours eulériens et hamiltoniens des graphes
- Problème de plus court chemin
- Problèmes de transport et de flots

Définitions de la Recherche Opérationnelle

Définition (Cambridge Dictionary)

Operational research UK (US operations research) The systematic study of how best to **solve problems in business and industry**.

Définition (Wikipedia)

Operational research is the use of **mathematical models**, statistics and **algorithms** to aid in **decision-making**.

Définition (ROADEF)

Recherche Opérationnelle est une **approche scientifique** pour la résolution de problèmes de **gestion de systèmes complexes**.

Recherche Opérationnelle - autres définitions

Définition

La **recherche opérationnelle** est une discipline qui utilise des méthodes **mathématiques** et **algorithmiques** pour analyser des problèmes complexes et optimiser des processus décisionnels.

Elle vise à aider les organisations à prendre des décisions efficaces en modélisant des situations réelles, en évaluant différentes options et en minimisant ou maximisant des critères spécifiques, comme le coût, le temps ou la ressource. Ses applications couvrent divers domaines, tels que la logistique, la finance, la production et la gestion des opérations.

Un premier problème

Achat de billets d'avion

Un homme d'affaires doit effectuer 5 voyages entre Casa (**CAS**) et Paris (**PAR**) selon les conditions suivantes :

- Il doit partir le lundi de CAS à PAR et revenir le mercredi de PAR à CAS.
- Un billet aller-retour : 400U.
- Réduction de 20 % si un weekend est inclus.
- Aller simple : 75 % du prix aller-retour.

Question

Comment acheter les billets pour les 5 semaines (à prix minimum) ?

Un premier problème

Evaluation des alternatives

Alternatives

- Acheter 5 CAS-PAR-CAS normaux.

$$5 \times 400 = 2000$$

- Acheter un CAS-PAR, 4 PAR-CAS-PAR comprenant un weekend et un PAR-CAS.

$$0.75 \times 400 + 4 \times 0.8 \times 400 + 0.75 \times 400 = 1880$$

- Acheter un CAS-PAR-CAS pour le lundi de la première semaine et le mercredi de la dernière semaine, et 4 PAR-CAS-PAR comprenant un weekend pour les autres voyages.

$$5 \times 0.8 \times 400 = 1600$$

La troisième alternative est la meilleure.

Décision binaire

Voyage sans redondance (Problème de voyageur de commerce)

Un étudiant projette de visiter les campus de trois universités au cours d'un voyage unique, débutant et finissant à l'aéroport de **P**.

- Les trois établissements sont dans les villes de **A**, **B**, et **C**.
- L'étudiant ne veut visiter chaque ville qu'une seule fois.
- On veux maintenir le trajet total le plus court possible.
- Les distances entre les aéroports de ces villes sont données dans la table ci-après :

Décision binaire

Ville	P	A	B	C
P	-	30	38	73
A	35	-	18	53
B	32	19	-	51
C	79	50	59	-

Table – Les distances de vol inter-aéroports.

Question

L'objectif ici est de trouver une modélisation mathématique capable de représenter ce problème et ses différentes contraintes tout en permettant d'évaluer et minimiser la distance globale de ce voyage.

Décision binaire : modélisation

- Puisque n'importe quel trajet consiste en une série de petits déplacements entre deux villes, on numérote les villes comme suit : 1 pour **P**, 2 pour **A**, 3 pour **B** et 4 pour **C**. Ainsi, nous aurons une variable $x_{1,2}$ égale à 1 si l'étudiant voyage de **P** à **A** au cours de son parcours total et 0 sinon.
- Puisqu'il n'y a pas de voyage d'une ville vers cette même ville, nous avons les contraintes : $x_{i,i} = 0$, $i = 1, \dots, 4$
- Chaque ville ne devant être visitée qu'une seule fois, elle ne peut apparaître qu'une seule fois comme ville d'arrivé. Donc pour j fixé avec $j = 1, \dots, 4$,

$$x_{1,j} + x_{2,j} + x_{3,j} + x_{4,j} = 1$$

d'où :

$$\sum_{i=1}^4 x_{i,j} = 1, \quad j = 1, \dots, 4.$$

Décision binaire : modélisation

- Puisque la même ville ne peut pas être une source d'un trajet plus qu'une fois, alors on pose la contrainte suivante :

$$\sum_{j=1}^4 x_{i,j} = 1, \quad i = 1, \dots, 4.$$

- Afin d'obtenir un véritable trajet ayant même origine et départ, nous devons rejeter les affectations qui décrivent des groupes déconnectés de petits déplacements comme $x_{1,2} = x_{2,1} = 1$ ou $x_{3,4} = x_{4,3} = 1$, avec toutes les autres variables égales à 0. Nous pouvons forcer ceci avec les contraintes :

$$x_{i,j} + x_{j,i} \leq 1, \quad i = 1, \dots, 4, \quad j = 1, \dots, 4.$$

- On peut décrire la distance totale associé à n'importe quel parcours autorisé par :

$$\sum_{i=1}^4 \sum_{j=1}^4 x_{i,j} a_{i,j}.$$

Décision binaire

Voyage sans redondance

Donc la modélisation du problème sera comme suit :

$$\min \sum_{i=1}^4 \sum_{j=1}^4 x_{i,j} a_{i,j}$$

- $x_{i,j} \in \{0, 1\}, i = 1, \dots, 4, j = 1, \dots, 4.$
- $x_{i,i} = 0, i = 1, \dots, 4.$
- $\sum_{i=1}^4 x_{i,j} = 1, j = 1, \dots, 4.$
- $\sum_{j=1}^4 x_{i,j} = 1, i = 1, \dots, 4.$
- $x_{i,j} + x_{j,i} \leq 1, i = 1, \dots, 4, j = 1, \dots, 4.$

R.O : Applications

Problème du voyageur de commerce

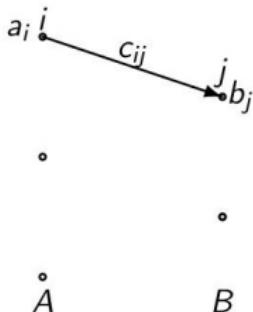
- Un voyageur de commerce, basé à Casablanca doit visiter plusieurs clients situés sur différentes villes au Maroc.
- Il souhaite effectuer la **tournée** la plus courte possible



R.O : Applications

Problème de transport

- de marchandises.
- des entrepôts vers les clients
- coûts de transport, distance sur les arcs
- trouver le meilleur plan de distribution

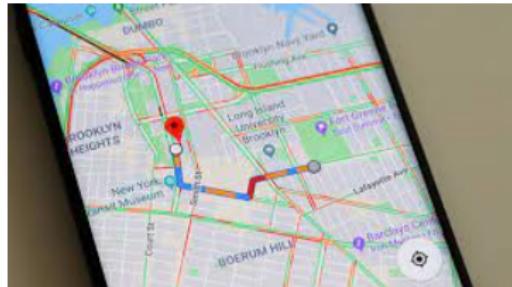


$$\begin{aligned} \min \sum c_{ij}x_{ij} \\ \sum_{j \in B} x_{ij} &\leq a_i \\ \sum_{i \in A} x_{ij} &\geq b_j \\ x_{ij} &\geq 0 \end{aligned}$$

R.O : Applications

Plus court chemin

- entre deux villes
- entre deux pays
- entre un fournisseur et ses clients
- ...



R.O : Applications

Mariages stables

- consistant à trouver par exemple, étant donnés n hommes, n femmes et leurs listes de préférences, une façon stable de les mettre en couple.
- Une situation est dite instable s'il existe au moins un homme et une femme qui préféreraient se mettre en couple plutôt que de rester avec leurs partenaires actuels

R.O : Applications

Applications du problème de mariage stable

En général, le problème de mariage stable est souvent utilisé dans des situations nécessitant une répartition de biens rares ou hétérogènes :

- Affectation des élèves à des écoles d'ingénieurs
- Affectation des étudiants à des spécialités
- Association des travailleurs à des postes clés
- Attribution des médecins internes à des hôpitaux
- Dons d'organes
- ...

R.O : Applications

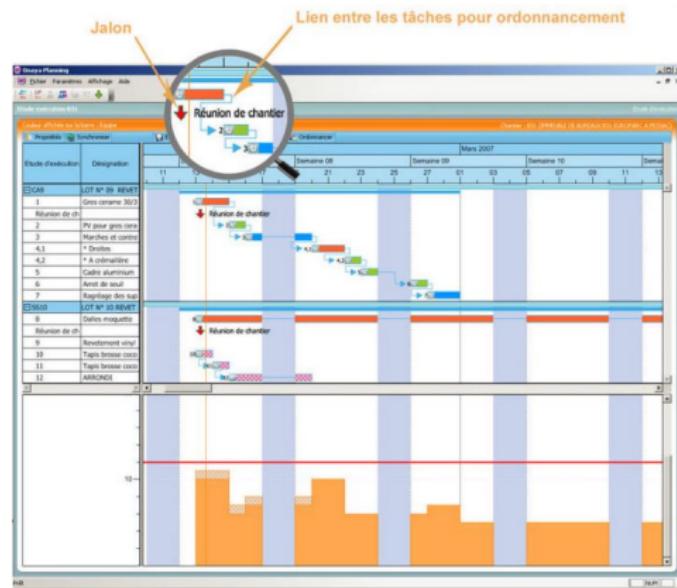
Challenges ROADEF

- 2022 Problème d'optimisation de chargement de camion 3D
Planification de la maintenance des pannes basée sur l'exploitation du réseau
- 2020 Problème de stock de coupe
- 2018 Problème d'acheminement des stocks pour la distribution de gaz
- 2016 Les trains ne disparaissent pas
- 2012 Réaffectation de machines
 - Un problème de gestion d'énergie de grande taille comportant des contraintes diversifiées
- 2010 Gestion des perturbations dans le domaine aérien
 - Planification des techniciens et des interventions pour les télécommunications
- 2009 Ordonnancement de véhicules pour une chaîne de montage automobile
- 2005 Gestion des prises de vue réalisées par un satellite d'observation
- 2003 Allocation de fréquences avec polarisation
- 2001

R.O : Applications en ingénierie

A. Planifier et ordonner

Ordonnancement des chantiers



R.O : Applications en ingénierie

A. Planifier et ordonner

Ordonnancement d'atelier

Ordonnancer
les passages
sur les
machines



R.O : Applications en ingénierie

A. Planifier et ordonner

Emplois du temps

Planifier n cours en le minimum de temps, certains cours ne pouvant avoir lieu en parallèle (partage des ressources: classe ou prof).

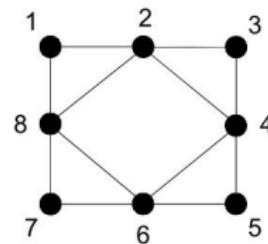
conflits entre les cours



graphe d'exclusion mutuelle

Un exemple

***Un cours =
un prof+une classe***



R.O : Applications en ingénierie

A. Planifier et ordonner

Planification des centres d'appels

Charges salariales = 70% des coûts de l'entreprise.

- 6 millions de clients
- 2500 téléconseillers de clientèle (TC)
- 7 sites, 33 activités
- 70000 appels par jour
- Coût annuel > 100 M€

R.O : Applications en ingénierie

A. Planifier et ordonner

Planification des centres d'appels

■ Données

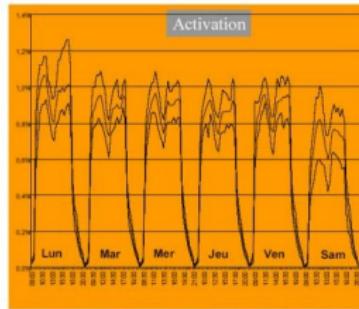
- ❑ courbe de demande
- ❑ contrats des TC (droits)

■ Objectif

- ❑ affecter au mieux les jours de congé aux TC

■ Contraintes

- ❑ répondre à la demande
- ❑ respecter les contrats



R.O : Applications en ingénierie

B. Stocker et Gérer

Gestion de la production, des stocks et de la maintenance

- Suivi de production
- Respect des délais
- Gain de temps
- Respect du client
- Meilleure compétitivité
- Organisation du travail
- résistance aux aléas
- ...

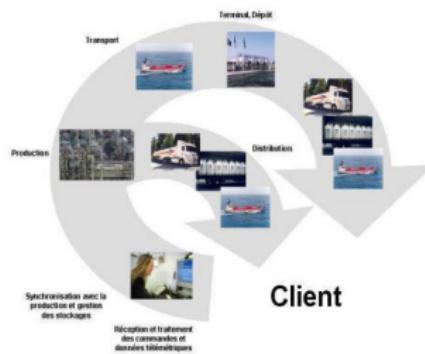


R.O : Applications en ingénierie

C. Transporter

Transport, logistique

- Optimisation des tournées de véhicules, distribution
- Relations fournisseurs / clients
- Organisation des centres logistique.



R.O : Applications en ingénierie

C. Transporter

Transport, logistique

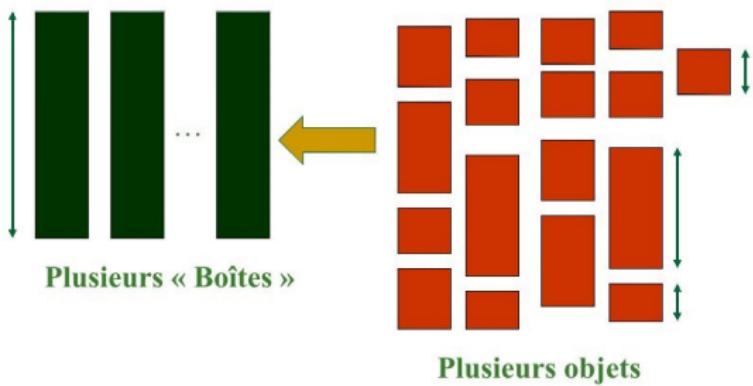
- Le ramassage scolaire



R.O : Applications en ingénierie

D. Emballer et ranger

Emballage

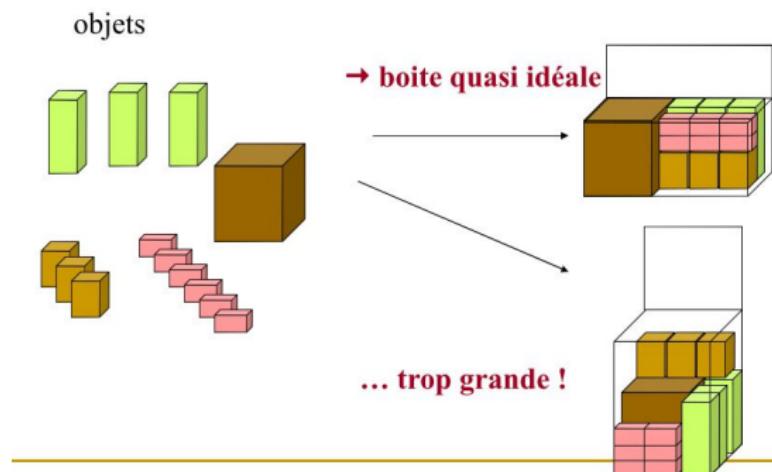


Comment mettre les objets dans les Boîtes en utilisant le moins possible de Boîtes ?

R.O : Applications en ingénierie

D. Emballer et ranger

Emballage



R.O : Applications en ingénierie

D. Emballer et ranger

Emballage

- Déterminer la boîte idéale pour placer les objets (celle qui peut accueillir les objets et qui minimise la place perdue).
- Problèmes de chargement de bateaux (2 degrés de liberté, objets similaires) avec des conteneurs.

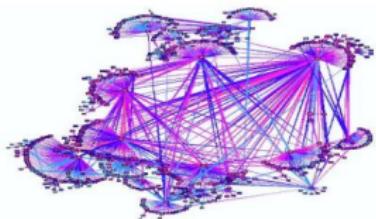


R.O : Applications en ingénierie

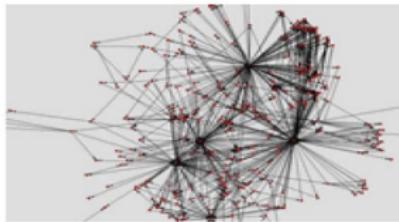
E. Router et Relier

Router et sécuriser

Réseau de mobilophonie



Web



R.O : Applications en ingénierie

E. Router et Relier

Calcul d'itinéraires

- en fonction du traffic :

- Données actualisées toutes les 5 minutes
- Informations nécessaires en temps réel
- Temps disponible pour calculer un itinéraire :
 $1/10^{\text{ème}}$ de seconde



R.O : Conclusion

La R.O consiste à :

- Faire le mieux : coût min, meilleur profit, plus court parcours, plus rapide chemin, sécurité maximum, ...
- Avec les ressources disponibles : ressources humaines, matière première, temps machines, moyen de transport, mémoire, ...



Recherche Opérationnelle R.O.

Partie 2: Programmation Linéaire P.L

Pr. Abdessamad Kamouss

Cycle Ingénieur
ENSAM Casablanca

Contenu du module

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- Résolution des PL
- Dualité
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- Parcours eulériens et hamiltoniens des graphes
- Problème de plus court chemin
- Problèmes de transport et de flots

Contenu du module

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- Résolution des PL
- Dualité
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- Parcours eulériens et hamiltoniens des graphes
- Problème de plus court chemin
- Problèmes de transport et de flots

Programmation linéaire

La programmation linéaire est une branche de la recherche opérationnelle dont l'objectif est de **minimiser** ou **maximiser** une fonction numérique multilinéaire (dite **fonction objectif** ou **fonction économique**) à plusieurs variables, sachant que ces dernières sont liées moyennant **des équations ou des inéquations linéaires** dites **contraintes**.

De nombreux problèmes concrets provenant de domaines aussi divers que l'industrie lourde, le raffinage, les transports, l'agriculture, la gestion, la production, l'informatique, l'ingénierie... peuvent être modélisés comme des programmes linéaires, la résolution de ces modèles ayant permis l'obtention de gains substantiels.

Problème de production

Exemple (Production de voiture)

Un fabricant de voitures propose 2 modèles à la vente, des grosses voitures et des petites voitures. La grosse voiture est vendue à 16000€ tandis que les petites voitures à 10000 €. Son problème vient de l'approvisionnement limité en deux matières premières, le caoutchouc et l'acier.

La construction d'une petite voiture nécessite l'emploi d'une unité de caoutchouc et d'une unité d'acier, tandis que celle d'une grosse voiture nécessite une unité de caoutchouc mais deux unités d'acier.

Matière	Caoutchouc	Acier
Disponibilités	400 unités	600 unités

Question

Le fabricant souhaite savoir combien de petites et de grosses voitures doit-il produire afin de maximiser son chiffre d'affaires en prenant en compte son stock actuel.

Problème de production

Modélisation

Variables de décision :

- x : nombre de grosses voitures produites
- y : nombre de petites voitures produites

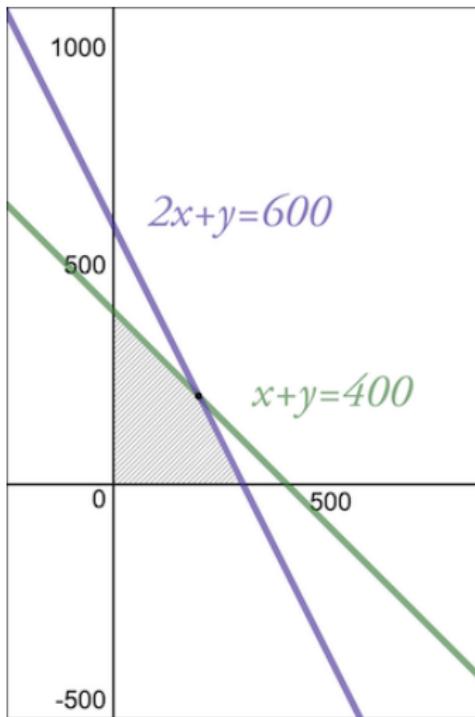
Fonction objectif :

$$[\text{Max}] \quad z = 16000x + 10000y$$

Contraintes :

- $x + y \leq 400$ (caoutchouc)
- $2x + y \leq 600$ (acier)
- $x \geq 0$ et $y \geq 0$.

Problème de production



Solution optimale $x^* = 200$, $y^* = 200$ et $z^* = 5.200.000\text{€}$.

Programme linéaire - Définitions

Définition

Un programme linéaire est une modélisation mathématique en R.O dans lequel les **variables sont des réels** qui doivent satisfaire un ensemble **d'équations et/ ou d'inéquations linéaires** (dites contraintes) et la valeur d'une **fonction linéaire** de ces variables (appelée fonction objectif) qu'on souhaite rendre minimum ou maximum.

Ingrédients principaux :

- Alternatives - variables de décision - inconnues du problème - variables d'activité.
- Restrictions - contraintes.
- Fonction à optimiser - fonction objectif - fonction économique.

Programme linéaire - Modélisation

Forme générale d'un programme linéaire :

$$\left\{ \begin{array}{l} \text{Maximiser ou Minimiser } z(x_1, x_2, \dots, x_n) = \sum_{j=1}^n c_j x_j \\ \text{Sous les contraintes} \quad \left\{ \begin{array}{l} \sum_{i=1}^n a_{1i} x_i \leq, =, \geq b_1 \\ \sum_{i=1}^n a_{2i} x_i \leq, =, \geq b_2 \\ \dots \dots \dots \\ \sum_{i=1}^n a_{mi} x_i \leq, =, \geq b_m \\ x_1, x_2, \dots, x_n \in \mathbb{R} \end{array} \right. \end{array} \right.$$

Programme linéaire - Modélisation

- second membre $b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$
- n var. de décision $X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$

- matrice de format $m \times n$

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & & \ddots & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

- coût (ou profit) $c = (c_1, c_2 \dots c_n)$

Représentation matricielle

$$\max z = cx$$

$$s.c. \quad Ax \quad \left\{ \begin{array}{l} \leq \\ \geq \\ = \end{array} \right\} b$$

Programme linéaire - Forme Canonique

Forme canonique d'un PL :

Un programme linéaire est dit **canonique** s'il est écrit sous la forme suivante :

$$(PC) \left\{ \begin{array}{l} [Max] \quad z = c^T x \\ Ax \leq b \\ x \geq 0 \end{array} \right.$$

Remarques

Deux propriétés caractérisent la forme canonique.

- Toutes les variables sont astreintes à être positives ou nulles (non négatives).
- Toutes les contraintes sont des inéquations.

Programme linéaire - Forme Standard

Forme standard d'un PL :

Un programme linéaire est dit **standard** s'il est écrit sous la forme suivante :

$$(PC) \left\{ \begin{array}{l} [Max] \quad z = c^T x \\ Ax = b \\ x \geq 0 \end{array} \right.$$

Remarques

Deux propriétés caractérisent la forme canonique.

- Toutes les variables sont astreintes à être positives ou nulles.
- Toutes les autres contraintes sont des équations.

Programme linéaire - Passage entre Formes

Et-il possible de passer d'une forme à une autre forme d'un programme linéaire quelconque ?

Théorème

- *Tout programme linéaire sous **forme standard** peut être écrit sous **forme canonique**, et*
- *tout programme linéaire sous **forme canonique** peut être écrit sous **forme standard**.*

Programme linéaire - Passage entre Formes

- équation → inéquation

$$ax = b \iff \begin{cases} ax \leq b \\ ax \geq b \end{cases}$$

- inéquation → équation : ajouter une variable d'écart ou d'excédent

$$\begin{aligned} ax \leq b &\iff ax + e = b, \quad e \geq 0 \\ ax \geq b &\iff ax - e = b, \quad e \geq 0 \end{aligned}$$

- variable non contrainte → variables positives

$$x \leq 0 \iff \begin{cases} x = e_1 - e_2 \\ e_1, e_2 \geq 0 \end{cases}$$

- $\max \leftrightarrow \min : *$ $\min f(x) = -\max(-f(x))$

Algorithme pour construire le programme linéaire

Afin de transformer un problème en un programme linéaire, on suit l'algorithme suivant :

Algorithme de construction de programme linéaire

- ① Identifier les variables de décision nécessaires ;
- ② Identifier les contraintes du problème et les exprimer en fonction des variables d'activités ;
- ③ Identifier la fonction objectif ;
- ④ Ecrire le programme linéaire et spécifier si le critère de sélection est à maximiser ou à minimiser.

Exemple d'un PL d'optimisation de production

Exemple (Problème de production)

Une usine fabrique deux produits P_1 et P_2 . Chacun de ces produits demande pour son usinage, des heures de fabrications unitaires sur les machines A, B, C, D, E comme indiqué dans le tableau suivant :

	A	B	C	D	E
P_1	0	1,5	2	3	3
P_2	3	4	3	2	0
Disponibilité de chaque machine	39h	60h	57h	70h	57h

Les marges brutes de chaque produit sont respectivement :

$$M_1 = 1700 \text{ UM} \quad M_2 = 3200 \text{ UM}$$

Question

Ecrire le programme linéaire qui détermine les nombres de produits de type P_1 et de type P_2 à fabriquer pour maximiser sa marge brute.

Exemple d'un PL d'optimisation de production

- Les variables : x_1 quantité à fabriquer de P_1 et x_2 quantité à fabriquer de P_2
- L'objectif :

$$\text{Maximiser } z = 1700x_1 + 3200x_2$$

- Les contraintes suivantes

$$3x_2 \leq 39$$

$$1,5x_1 + 4x_2 \leq 60$$

$$2x_1 + 3x_2 \leq 57$$

$$3x_1 + 2x_2 \leq 70$$

$$3x_1 \leq 57$$

$$x_1 \geq 0, \quad x_2 \geq 0$$

Exemple d'un PL d'optimisation de production

Le PL correspondant

$$[Max] z = 1700x_1 + 3200x_2$$

$$\begin{cases} 3x_2 \leq 39 \\ 1,5x_1 + 4x_2 \leq 60 \\ 2x_1 + 3x_2 \leq 57 \\ 3x_1 + 2x_2 \leq 70 \\ 3x_1 \leq 57 \\ x_1 \geq 0, \quad x_2 \geq 0 \end{cases}$$

Exemple d'un PL d'optimisation de transport

Exemple (Problème de transport)

Une entreprise de construction d'automobiles possède trois usines situées respectivement à Tanger, Kénitra et Agadir. Un certain métal nécessaire à la construction des véhicules est disponible aux stocks de Casa et El Jadida. Les quantités de ce métal nécessaires aux usines sont 400, 300 et 200 tonnes respectivement pour les usines de Tanger, Kénitra et Agadir chaque semaine, tandis que les quantités disponibles sont 550 et 350 tonnes par semaines respectivement à Casa et à El Jadida. Les coûts de transport unitaires sont comme suit :

	Tanger	Kénitra	Agadir
Casa	5	6	3
El Jadida	3	5	4

Exemple d'un PL d'optimisation de transport

Exemple (Suite)

Ce tableau signifie que pour envoyer x tonnes de Casa à Kénitra, par exemple, il en coûte $6x$ UM. Le problème consiste à déterminer un **plan de transport optimal**, c'est-à-dire à trouver quels sont les poids de métal à envoyer de chaque stock à chaque usine de sorte que :

- (i) Les demandes soient satisfaites (chaque usine reçoit au moins la quantité de métal qui lui est nécessaire).
- (ii) Les quantités demandées à chaque stock n'excèdent pas les quantités disponibles.
- (iii) Les quantités envoyées sont positives ou nulles.
- (iv) Le coût total du transport est rendu minimum compte tenu des contraintes ci-dessus.

Exemple d'un PL d'optimisation de transport

Le PL correspondant

Affectant au stock de Casa l'indice 1, au stock d'El Jadida l'indice 2 et aux trois usines les indices 1,2 et 3 respectivement pour Tanger, Kénitra et Agadir, on conviendra que x_{ij} représentera le nombre de tonnes de métal qui sont acheminées chaque semaine depuis le stock d'indice i vers l'usine d'indice j . Le programme linéaire s'écrit alors :

$$[Min] z = 5x_{11} + 6x_{12} + 3x_{13} + 3x_{21} + 5x_{22} + 4x_{23}$$

$$\left\{ \begin{array}{l} x_{11} + x_{21} \geq 400 \\ x_{12} + x_{22} \geq 300 \\ x_{13} + x_{23} \geq 200 \\ x_{11} + x_{12} + x_{13} \leq 550 \\ x_{21} + x_{22} + x_{23} \leq 350 \\ x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23} \geq 0 \end{array} \right.$$

Contenu du module

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- **Résolution des PL**
- Dualité
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- Parcours eulériens et hamiltoniens des graphes
- Problème de plus court chemin
- Problèmes de transport et de flots

Résolution d'un programme linéaire

Méthodes de résolution de programme linéaire

Les méthodes suivantes sont les plus utilisées :

- Méthode graphique
- Méthode algébrique
- Méthode Simplexe
- Fonction Solveur
- Python en utilisant PuLP

Définition (**Solution admissible**)

Une solution **admissible** est un ensemble de valeurs données aux variables qui satisfait toutes les contraintes.

Définition (**Solution optimale**)

Une **solution optimale** est une solution admissible qui optimise la fonction objectif.

Description de la méthode

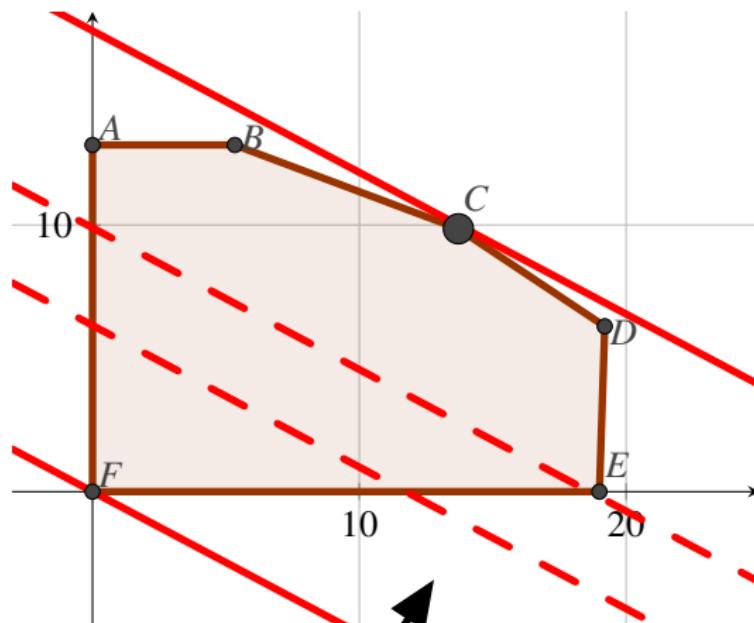
- ① On trace les droites représentant les contraintes. Chaque inéquation est satisfaite dans un demi-plan limité par la droite d'équation correspondante.
- ② Après avoir hachuré les parties du plan ne respectant pas les contraintes, il reste une zone non hachurée qui représente l'ensemble des solutions possibles (ensemble admissible).
- ③ Une solution optimale du programme linéaire est située en un sommet du domaine de l'ensemble des solutions possibles.

Résolution de PL - Méthode Graphique

Résolution du PL du problème de l'optimisation de la production

$$\left\{ \begin{array}{l} [\text{Max}] z = 1700x_1 + 3200x_2 \\ 3x_2 \leq 39 \\ 1,5x_1 + 4x_2 \leq 60 \\ 2x_1 + 3x_2 \leq 57 \\ 3x_1 + 2x_2 \leq 70 \\ 3x_1 \leq 57 \\ x_1 \geq 0, x_2 \geq 0 \end{array} \right.$$

Résolution de PL - Méthode Graphique

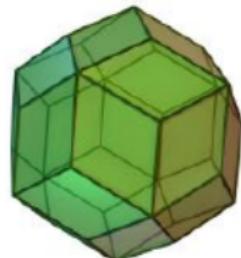


L'optimum se trouve au point **C** ($x_1 = \frac{96}{7}$ et $x_2 = \frac{69}{7}$). Donc

$$z_{max} = \frac{384000}{7}.$$

Résolution de PL - Méthode Graphique

- L'ensemble des solutions réalisables est toujours un polyèdre (intersection de demi-espaces)
- Les lignes de niveau $\{f = \text{constante}\}$ de la fonction-objectif $z = f(x_1, x_2, \dots, x_n)$ sont des hyperplans affines ($n = 2 \Rightarrow$ droite, $n = 3 \Rightarrow$ plan...)



Optimum atteint au bord

L'optimum de la fonction-objectif, s'il existe, est atteint en un ou plusieurs **sommets** du polyèdre.

Justification mathématique :

Les dérivées partielles de $f(X) = c.X$ ne s'annulent jamais, et le domaine $\{X / \sum_{j=1}^n a_{ij}x_j \leq b_i, i \in \{1, \dots, m\}\}$ est un compact.

Donc l'optimum est atteint au bord.

Résolution de PL - Méthode Simplexe

Solution de base

Considérons un PL dans sa forme standard qui sera noté (**PLS**) :

$$\left\{ \begin{array}{l} \text{Maximiser } z(x_1, x_2, \dots, x_n) = \sum_{j=1}^n c_j x_j \\ \text{Sous les contraintes} \quad \left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \pm e_1 = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \pm e_2 = b_2 \\ \cdots \cdots \cdots \\ \cdots \cdots \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \pm e_m = b_m \\ x_1, x_2, \dots, x_n \in \mathbb{R}^+ \end{array} \right. \end{array} \right.$$

- n désigne le nombre de variables de décision x_1, x_2, \dots, x_n .
- m le nombre des contraintes = le nombre de variables d'écart ou d'excédent e_1, e_2, \dots, e_m .
- On obtient donc un système de m équations linéaires à $n' = n + m$ inconnues ($m < n'$) : infinité de solutions possible.

Résolution de PL - Méthode Simplexe

Définition (Solutions de base)

Une solution $(x_1, x_2, \dots, x_n, e_1, e_2, \dots, e_m)$ vérifiant les m contraintes de **(PLS)** est dite **solution de base** de **(PLS)** si au moins $(n' - m)$ de ses variables sont égales à 0.

Les variables fixées à zéro sont appelées variables **hors base** et les autres variables **en base**.

Remarque

Une solution de base est dites **dégénérée** lorsque certaines variables de base sont nulles.

Définition (Solutions de base admissible)

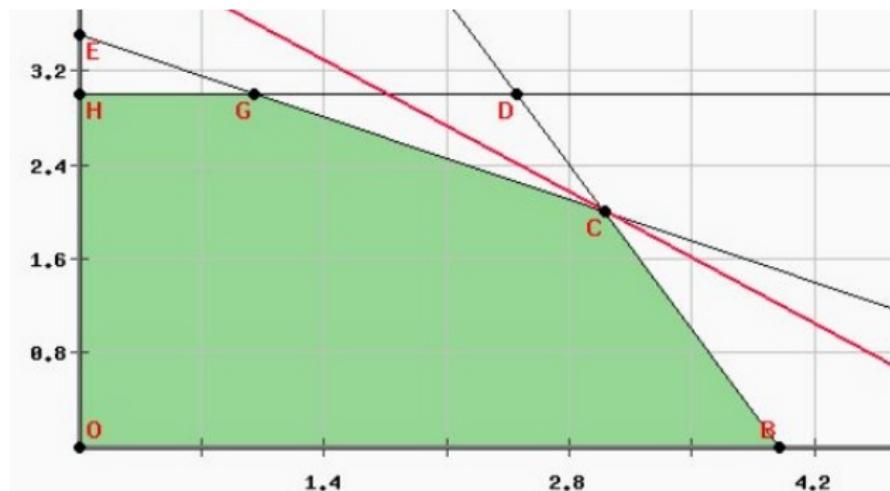
Une solution de base dont tout les coordonnées sont **non négatives** est dite **solution de base admissible** de **(PLS)**.

Résolution de PL - Méthode Simplexe

PLS - Bases et points extrêmes

Considérons l'exemple suivant :

$$(PL) \begin{cases} 2x + y \leq 8 \\ x + 2y \leq 7 \\ y \leq 3 \\ x, y \geq 0 \end{cases} \Rightarrow (PLS) \begin{cases} 2x + y + e_1 = 8 \\ x + 2y + e_2 = 7 \\ y + e_3 = 3 \\ x, y, e_1, e_2, e_3 \geq 0 \end{cases}$$



Résolution de PL - Méthode Simplexe

PLS - Bases et points extrêmes

x	y	e ₁	e ₂	e ₃	sol de base	admiss.	pt extrême
0	0	8	7	3	✓	✓	(0,0)
0	8	0	-9	-5	✓	✗	
0	3.5	4.5	0	-0.5	✓	✗	
0	3	5	1	0	✓	✓	(0,3)
4	0	0	3	3	✓	✓	(4,0)
7	0	-6	0	3	✓	✗	
0			0		✗	✗	
3	2	0	0	1	✓	✓	(3,2)
2.5	3	0	-1.5	0	✓	✗	
1	3	3	0	0	✓	✓	(1,3)

PLS - Bases et points extrêmes

Base voisine et pivotage

Bases voisines

Deux sommets voisins correspondent à deux bases B et B' telles qu'on remplace une variable de B pour obtenir B'

⇒ passer à un sommet voisin = changer de base (base voisine).

C'est le principe du pivotage.

Résolution de PL - Méthode Simplexe

PLS - Simplexe

Algorithme du simplexe

- Dantzig, 1947.
- Algorithme itératif de résolution de problème de programmation linéaire.

Principe

A partir d'un sommet, chercher un sommet voisin qui améliore la fonction objectif.

Propriété du problème

Soit X_0 un sommet non optimum.

Alors il existe X , un sommet voisin de X_0 , tel que $f(X) > f(X_0)$.

⇒ On peut procéder d'une manière itérative jusqu'à l'obtention d'une **solution optimale**

Résolution de PL - Méthode Simplexe : Exemple

- Transformer le (PL) sous sa forme standard (PLS) .

$$\left\{ \begin{array}{ll} \text{Maximizer } z & = 4x + 5y \\ 2x + y & \leq 8 \\ x + 2y & \leq 7 \\ y & \leq 3 \\ x, y & \geq 0 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{ll} \text{Maximizer } z & = 4x + 5y \\ 2x + y + e_1 & = 8 \\ x + 2y + e_2 & = 7 \\ y + e_3 & = 3 \\ x, y, e_1, e_2, e_3 & \geq 0 \end{array} \right.$$

- Trouver une solution de base admissible initiale. Par exemple $\{e_1, e_2, e_3\}$

$$\left\{ \begin{array}{l} 2x + y + e_1 = 8 \\ x + 2y + e_2 = 7 \\ y + e_3 = 3 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} e_1 = 8 - 2x - y \\ e_2 = 7 - x - 2y \\ e_3 = 3 - y \end{array} \right.$$

On obtient donc $x = y = 0$ et $(e_1, e_2, e_3) = (8, 7, 3)$.

Donc $\{e_1, e_2, e_3\}$: variables de base et $\{x, y\}$ variables hors base.

La valeur de la fonction objectif est $z = 0$.

Résolution de PL - Méthode Simplexe : Exemple

- Déterminer la solution de base admissible voisine (une variable entrante et une variable sortante) qui permet d'augmenter la valeur de la fonction objectif z :

Regardons bien : $z = 4x + 5y$

On peut faire augmenter z en faisant entrer x ou y dans la base. On prends y vu qu'elle a le plus grand coefficient positif.

Quelle est la valeur maximale que pourra prendre y ?

$$\begin{cases} e_1 = 8 - 2x - y \geq 0 \Rightarrow y \leq 8 \\ e_2 = 7 - x - 2y \geq 0 \Rightarrow y \leq 3.5 \\ e_3 = 3 - y \geq 0 \Rightarrow y \leq 3 \end{cases}$$

Le max de y est 3 , pour $y = 3$, on obtenons $e_1 = 5 - x$, $e_2 = 1 - x$ et $e_3 = 0$.

Nouvelle base candidate :

$$\{e_1, e_2, e_3\} \cup \{\textcolor{blue}{y}\} \setminus \{\textcolor{red}{e}_3\} = \{e_1, e_2, \textcolor{blue}{y}\}$$

Résolution de PL - Méthode Simplexe : Exemple

$$\begin{cases} e_1 = 8 - 2x - y \\ e_2 = 7 - x - 2y \\ e_3 = 3 - y \end{cases} \Leftrightarrow \begin{cases} e_1 = 5 - 2x + e_3 \\ e_2 = 1 - x + 2e_3 \\ y = 3 - e_3 \end{cases}$$

- On exprime z en fonction des variables hors base :

$$z = 4x + 5y$$

$$z = 15 + 4x - 5e_3$$

⇒ Solution de base associée : $x = e_3 = 0$

$$\begin{cases} e_1 = 5 - 2x + e_3 = 5 \\ e_2 = 1 - x + 2e_3 = 1 \\ y = 3 - e_3 = 3 \end{cases}$$

Donc $(e_1, e_2, y) = (5, 1, 3)$ et la fonction objectif prends la valeur $\mathbf{z = 15}$.

Résolution de PL - Méthode Simplexe : Exemple

$$z = 15 + 4x - 5e_3$$

Augmenter encore z ? Faire entrer x

Quelle est la valeur maximale que pourra avoir x ?

$$\begin{cases} e_1 = 5 - 2x + e_3 \geq 0 \Rightarrow x \leq 2.5 \\ e_2 = 1 - x + 2e_3 \geq 0 \Rightarrow x \leq 1 \\ y = 3 - e_3 \geq 0 \Rightarrow \text{pas de contrainte} \end{cases}$$

Le max de x est 1, et e_2 peut sortir de la base.

Nouvelle base candidate : $\{e_1, x, y\}$

$$\begin{cases} e_1 = 3 + 2e_2 - 3e_3 \\ x = 1 - e_2 + 2e_3 \\ y = 3 - e_3 \\ z = 19 - 4e_2 + 3e_3 \end{cases}$$

Résolution de PL - Méthode Simplexe : Exemple

$$z = 19 - 4e_2 + 3e_3$$

Augmenter encore z ? Faire entrer e_3

Quelle est la valeur maximale que pourra avoir e_3 ?

$$\begin{cases} e_1 = 3 + 2e_2 - 3e_3 \geq 0 \Rightarrow e_3 \leq 1 \\ x = 1 - e_2 + 2e_3 \geq 0 \Rightarrow \text{pas de contrainte} \\ y = 3 - e_3 \geq 0 \Rightarrow e_3 \leq 3 \end{cases}$$

Le max de e_3 est 1, et e_1 peut sortir de la base.

Nouvelle base candidate : $\{e_3, x, y\}$

$$\begin{cases} e_3 = 1 + 2/3e_2 - 1/3e_1 \\ x = 3 + 1/3e_2 + 2/3e_1 \\ y = 2 - 2/3e_2 + 1/3e_1 \\ z = 22 - 2e_2 - e_1 \end{cases}$$

Résolution de PL - Méthode Simplexe : Exemple

Finalement, on a :

$$z = 22 - 2e_2 - e_1$$

donc $z^* \leq 22$.

Or la solution de base $x = 3, y = 2$ et $e_3 = 1$ permet d'obtenir la valeur optimale de la fonction objectif $\mathbf{z}^* = 22$.

⇒ Donc on a trouvé l'optimum

Contenu du module

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- Résolution des PL
- **Dualité**
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- Parcours eulériens et hamiltoniens des graphes
- Problème de plus court chemin
- Problèmes de transport et de flots

Définition (Dualité)

- La dualité fait référence à la relation entre un problème d'optimisation appelé "**problème primal**" et un autre problème associé appelé "**problème dual**".
- Chaque problème est formulé de manière complémentaire, et les solutions des deux problèmes sont liées.
- Cela permet :
 - d'obtenir des informations supplémentaires
 - de simplifier la résolution du problème primal.
 - avoir une autre vision du problème.
 - réaliser un équilibre sur un marché.
 - ...

Problème Dual - Vision économique

Exemple (Vision du Primal)

Une société **META** fabrique, deux articles P_1 et P_2 qu'elle vend à des grossistes aux prix respectifs de 320 et 550 UM.

La fabrication de ces deux produits P_1 et P_2 nécessite l'utilisation de trois machines différentes M_1 , M_2 et M_3 pendant des temps exprimés en heures dans le tableau suivant :

	M_1	M_2	M_3	Prix
P_1	50	30	20	320
P_2	10	20	15	550
Disponibilités	600	500	300	-

But

Maximiser le profit obtenu par la fabrication et la vente des produits P_1 et P_2 .

Problème Primal

Exemple (Vision du Primal)

On considère les variables suivantes :

- x_1 la quantité produite de P_1 .
- x_2 la quantité produite de P_2 .

On peut modéliser le problème **PRIMAL** de la manière suivante :

$$[Max] z = 320x_1 + 550x_2$$

$$\text{sc} \quad \begin{cases} 50x_1 + 10x_2 \leq 600 \\ 30x_1 + 20x_2 \leq 500 \\ 20x_1 + 15x_2 \leq 300 \\ x_1 \geq 0, x_2 \geq 0. \end{cases}$$

Problème Dual - Vision économique

Exemple (Vision du Dual)

- Maintenant un autre fabricant **VISION** souhaite produire les mêmes produits P_1 et P_2 mais ne dispose pas des disponibilités nécessaires au niveau de ses ateliers.
- Pour cela, il souhaite acheter l'utilisation des disponibilités **600h**, **500h** et **300h** des machines M_1 , M_2 et M_3 respectivement pour produire P_1 et P_2 .
- Il cherche à déterminer le prix unitaire d'achat de chaque heure de ces machines afin de faire une proposition minimale convaincante au fabricant **META**.

But

Minimiser le coût d'acquisition des ressources M_1 , M_2 et M_3 tout en restons concurrentiel.

Problème Dual - Vision économique

Exemple (Vision du Dual)

Pour cela on considère les variables suivantes :

- y_1 le coût d'achat d'une heure d'utilisation de M_1 .
- y_2 le coût d'achat d'une heure d'utilisation de M_2 .
- y_3 le coût d'achat d'une heure d'utilisation de M_3 .

On peut modéliser donc le problème **DUAL** comme suit :

$$[Min] w = 600y_1 + 500y_2 + 300y_3$$

$$sc \quad \begin{cases} 50y_1 + 30y_2 + 20y_3 \geq 320 \\ 10y_1 + 20y_2 + 15y_3 \geq 550 \\ y_1, y_2, y_3 \geq 0. \end{cases}$$

Problèmes Primal - Dual - Vision économique

Pour le fabricant META : On cherche à déterminer le plan de fabrication permettant de **maximiser** le profit.

Pour le fabricant VISION : On cherche à déterminer l'offre d'achat permettant de **minimiser** le coût de sa production.

PRIMAL vs DUAL

$$[\text{Max}] z = 320x_1 + 550x_2$$

$$\text{sc} \quad \begin{cases} 50x_1 + 10x_2 \leq 600 \\ 30x_1 + 20x_2 \leq 500 \\ 20x_1 + 15x_2 \leq 300 \\ x_1 \geq 0, x_2 \geq 0, \end{cases}$$

$$[\text{Min}] w = 600y_1 + 500y_2 + 300y_3$$

$$\text{sc} \quad \begin{cases} 50y_1 + 30y_2 + 20y_3 \geq 320 \\ 10y_1 + 20y_2 + 15y_3 \geq 550 \\ y_1, y_2, y_3 \geq 0. \end{cases}$$

Dualité

- La dualité est un concept important en Recherche Opérationnelle.
- Tout programme linéaire admet un programme dual. Le premier est alors appelé **PRIMAL** et le second est son **DUAL**.
- Le **DUAL** a une interprétation économique différente du **PRIMAL** .
- Le **DUAL** et le **PRIMAL** sont liés et on peut trouver la solution de l'un dès que la solution de l'autre est bien connue.
- La recherche du DUAL peut souvent s'imposer si l'on voit que le PRIMAL paraît difficile à résoudre.

Problèmes Primal - Dual : Définition

Matrice A de taille $m \times n$

Vecteurs $\mathbf{c} \in \mathbb{R}^n$ et $\mathbf{b} \in \mathbb{R}^m$.

Définition (problème dual)

Au programme linéaire **primal**

$$(PL) \left\{ \begin{array}{l} \max_{\mathbf{x} \in \mathbb{R}^n} [F(\mathbf{x}) = \mathbf{c}^\top \mathbf{x}] \\ \left\{ \begin{array}{l} A\mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \geq 0 \end{array} \right. \end{array} \right.$$

on associe le programme linéaire **dual**

$$(PLD) \left\{ \begin{array}{l} \min_{\mathbf{y} \in \mathbb{R}^m} [G(\mathbf{y}) = \mathbf{b}^\top \mathbf{y}] \\ \left\{ \begin{array}{l} A^\top \mathbf{y} \geq \mathbf{c} \\ \mathbf{y} \geq 0 \end{array} \right. \end{array} \right.$$

Problèmes Primal - Dual

Programme linéaire primal

$$\max_{\mathbf{x} \in \mathbb{R}^n} [\mathbf{F}(\mathbf{x}) = \mathbf{c}^\top \mathbf{x}]$$

$$(PL) \quad \begin{cases} A\mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \geq 0 \end{cases}$$

$$\min_{\mathbf{y} \in \mathbb{R}^m} [\mathbf{G}(\mathbf{y}) = \mathbf{b}^\top \mathbf{y}]$$

$$(PLD) \quad \begin{cases} A^\top \mathbf{y} \geq \mathbf{c} \\ \mathbf{y} \geq 0 \end{cases}$$

Comparaison primal/dual.

Primal	Dual
--------	------

$$\max(F) \leftrightarrow \min(G)$$

coefficient **c** de $F \leftrightarrow$ second membre **c**

second membre **b** \leftrightarrow coefficient **b** de G

m contraintes inégalités ($A\mathbf{x} \leq \mathbf{b}$) \leftrightarrow m contraintes de positivité ($\mathbf{y} \geq 0$)

n contraintes de positivité ($\mathbf{x} \geq 0$) \leftrightarrow n contraintes inégalités ($A^\top \mathbf{y} \geq \mathbf{c}$)

Problèmes Primal - Dual

Règles de passage :

	Min	Max
Primal		Dual
Dual		Primal
Variable ≥ 0	Contrainte \leq	
Variable ≤ 0	Contrainte \geq	
Variable ≤ 0	Contrainte $=$	
Contrainte \leq	Variable ≤ 0	
Contrainte $=$	Variable ≤ 0	
Contrainte \geq	Variable ≥ 0	

Problèmes Primal - Dual

Exemple

Donner le programme dual du programme primal suivant :

$$(PL) : \left\{ \begin{array}{l} [Max] z = 4x_1 + 5x_2 + 2x_3 \\ sc \quad \left\{ \begin{array}{l} 2x_1 + 4x_2 = 3 \\ x_1 + x_3 \geq 2 \\ 3x_1 + x_2 + x_3 \leq 10 \\ x_2 + x_3 \leq 1 \\ x_1 \geq 0, x_2 \leq 0, x_3 \geq 0 \end{array} \right. \end{array} \right.$$

Le programme dual du primal précédent est :

$$(PLD) : \left\{ \begin{array}{l} [Min] w = 3y_1 + 2y_2 + 10y_3 + y_4 \\ sc \quad \left\{ \begin{array}{l} 2y_1 + y_2 + 3y_3 \geq 4 \\ 4y_1 + y_3 + y_4 \leq 5 \\ y_2 + y_3 + y_4 \geq 2 \\ y_1 \leq 0, y_2 \leq 0, y_3 \geq 0, y_4 \geq 0 \end{array} \right. \end{array} \right.$$

Problèmes Primal - Dual

Exemple

Donner le programme dual du programme primal suivant :

$$(PL) : \left\{ \begin{array}{l} \text{Max} \quad z = 4x_1 + 5x_2 + 2x_3 \\ \text{sc} \quad \left\{ \begin{array}{l} 2x_1 + 4x_2 = 3 \\ x_1 + x_3 \geq 2 \\ 3x_1 + x_2 + x_3 \leq 10 \\ x_2 + x_3 \leq 1 \\ x_1 \geq 0, x_2 \leq 0, x_3 \geq 0 \end{array} \right. \end{array} \right.$$

Le programme dual du primal précédent est :

$$(PLD) : \left\{ \begin{array}{l} \text{Min} \quad w = 3y_1 + 2y_2 + 10y_3 + y_4 \\ \text{sc} \quad \left\{ \begin{array}{l} 2y_1 + y_2 + 3y_3 \geq 4 \\ 4y_1 + y_3 + y_4 \leq 5 \\ y_2 + y_3 + y_4 \geq 2 \\ y_1 \leq 0, y_2 \leq 0, y_3 \geq 0, y_4 \geq 0 \end{array} \right. \end{array} \right.$$

Propriété

Le dual du dual est le primal.

Preuve. On considère un programme linéaire (*PL*) sous forme canonique.
Alors son dual (*PLD*) est :

$$(PLD) \left\{ \begin{array}{l} \min_{\mathbf{y}} [G(\mathbf{y}) = \mathbf{b}^\top \mathbf{y}] \\ \left\{ \begin{array}{l} \mathbf{A}^\top \mathbf{y} \geq \mathbf{c} \\ \mathbf{y} \geq 0 \end{array} \right. \end{array} \right. \iff (PLD) \left\{ \begin{array}{l} \max_{\mathbf{y}} [-G(\mathbf{y}) = (-\mathbf{b})^\top \mathbf{y}] \\ \left\{ \begin{array}{l} -\mathbf{A}^\top \mathbf{y} \leq -\mathbf{c} \\ \mathbf{y} \geq 0 \end{array} \right. \end{array} \right.$$

On prend le dual du dual :

$$(PLDD) \left\{ \begin{array}{l} \min_{\mathbf{x}} [(-\mathbf{c})^\top \mathbf{x}] \\ \left\{ \begin{array}{l} (-\mathbf{A}^\top)^\top \mathbf{x} \geq -\mathbf{b} \\ \mathbf{x} \geq 0 \end{array} \right. \end{array} \right. \iff (PL) \left\{ \begin{array}{l} \max_{\mathbf{x}} [\mathbf{c}^\top \mathbf{x}] \\ \left\{ \begin{array}{l} \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \geq 0 \end{array} \right. \end{array} \right.$$

Théorème (THÉORÈME FAIBLE DE DUALITÉ)

Soit \mathbf{x} une solution réalisable d'un (PL) sous forme canonique et \mathbf{y} une solution réalisable du problème dual (PLD). Alors :

- ① $F(\mathbf{x}) \leq G(\mathbf{y})$
- ② Si $F(\mathbf{x}) = G(\mathbf{y})$ alors \mathbf{x} et \mathbf{y} sont des solutions optimales de (PL) et (PLD) respectivement.

Preuve.

- ① On a $A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0$ et $A^\top \mathbf{y} \geq \mathbf{c}, \mathbf{y} \geq 0$.

$$F(\mathbf{x}) = \mathbf{c}^\top \mathbf{x} \leq \left(A^\top \mathbf{y} \right)^\top \mathbf{x} = \mathbf{y}^\top \underbrace{A\mathbf{x}}_{\leq \mathbf{b}} \leq \mathbf{y}^\top \mathbf{b} = G(\mathbf{y})$$

- ② Soient \mathbf{x}^* et \mathbf{y}^* des solutions réalisables de (PL) et (PLD) telles que $F(\mathbf{x}^*) = G(\mathbf{y}^*)$. D'après 1., pour \mathbf{x} solution réalisable de (PL), on a $F(\mathbf{x}) \leq G(\mathbf{y}^*) = F(\mathbf{x}^*)$ donc \mathbf{x}^* est une solution réalisable optimale. Idem pour \mathbf{y}^* .

Théorème (THÉORÈME FORT DE DUALITÉ)

Si le problème primal (PL) admet une solution réalisable optimale \mathbf{x}^ alors le problème dual (PLD) admet lui aussi une solution réalisable optimale \mathbf{y}^* avec*

$$F(\mathbf{x}^*) = G(\mathbf{y}^*).$$

Preuve. Preuve. On suppose (PL) mis sous forme standard. S'il existe une solution réalisable optimale, alors il existe une solution de base réalisable optimale $\mathbf{x}_{B^*} = A_{B^*}^{-1}\mathbf{b}$. On choisit alors

$$\mathbf{y}^* = (A_{B^*}^{-1})^\top \mathbf{c}_{B^*}.$$

On montre que \mathbf{y}^* est une solution réalisable optimale pour le dual (PLD).

Dualité - Théorèmes

Rappel : Il y a 3 cas possibles (et seulement 3) pour le problème primal (PL) :

- ① il existe (au moins) une solution optimale.
- ② l'ensemble \mathcal{D}_R des solutions réalisables n'est pas borné et l'optimum est infini.
- ③ pas de solution réalisable ($\mathcal{D}_R = \emptyset$).

Théorème (Relations entre solutions de (PL) et (PLD))

Etant donnés un problème primal (PL) et son dual (PLD), une et une seule des trois situations suivantes a lieu

- a) les deux problèmes possèdent chacun des solutions optimales (à l'optimum, les coûts sont égaux).
- b) un des problèmes possède une solution réalisable avec un optimum infini, l'autre n'a pas de solution.
- c) aucun des deux problèmes ne possède de solution réalisable.

Dualité - Conditions d'optimalité primal-dual (COPD)

On se place dans le cas (a) où les problèmes primal et dual possèdent chacun des solutions optimales (optimum fini).

Théorème (COPD)

Soient \mathbf{x} et \mathbf{y} des solutions réalisables respectivement du problème primal (PL) et du problème dual (PLD).

Alors \mathbf{x} et \mathbf{y} sont des solutions réalisables optimales si et seulement si les conditions d'optimalité primal-dual (COPD) suivantes sont vérifiées :

- Si une **contrainte** est satisfaite en tant qu'une **inégalité stricte** (PL) ou (PLD) alors **la variable correspondante** de l'autre programme **est nulle**.
- Si la valeur d'une **variable** dans (PL) ou (PLD) est **strictement positive** alors **la contrainte correspondante** de l'autre programme est **une égalité**.

Dualité - Conditions d'optimalité primal-dual (COPD)

\mathbf{x}^* et \mathbf{y}^* solutions réalisables optimales de (PL) et (PLD) respectivement.

$$\left\{ \begin{array}{l} \bullet \quad \sum_{j=1}^n a_{ij} \cdot x_j^* < b_i \Rightarrow y_i^* = 0 \\ \bullet \quad \sum_{i=1}^m a_{ij} \cdot y_i^* < c_j \Rightarrow x_j^* = 0 \end{array} \right.$$

$$\left\{ \begin{array}{l} \bullet \quad y_i^* > 0 \Rightarrow \sum_{j=1}^n a_{ij} x_j^* = b_i \\ \bullet \quad x_j^* > 0 \Rightarrow \sum_{i=1}^m a_{ij} y_i^* = c_j \end{array} \right.$$

Problèmes Primal - Dual

Exemple (Passage entre Primal et Dual)

Considérons le (**PL**) suivant et son dual (**PLD**) .

$$\begin{array}{ll} \text{(PL)} & \left\{ \begin{array}{ll} [\text{Max}] z = 4x_1 + 5x_2 \\ 2x_1 + x_2 \leq 8 \\ x_1 + 2x_2 \leq 7 \\ x_2 \leq 3 \\ x_1, x_2 \geq 0 \end{array} \right. \\ & \quad \quad \quad \text{(PLD)} \left\{ \begin{array}{ll} [\text{Min}] w = 8y_1 + 7y_2 + 3y_3 \\ 2y_1 + y_2 \geq 4 \\ y_1 + 2y_2 + y_3 \geq 5 \\ y_1, y_2, y_3 \geq 0 \end{array} \right. \end{array}$$

La solution optimale de (**PL**) est $x_1^* = 3$, $x_2^* = 2$ et $z^* = 22$.

- Vu que la 3^{ème} contrainte du (**PL**) n'est pas saturée alors $y_3 = 0$.
- Vu que $x_1^* > 0$ et $x_2^* > 0$ alors $2y_1 + y_2 = 4$ et $y_1 + 2y_2 + y_3 = 5$.

On obtient alors

$$y_1^* = 1, \quad y_2^* = 2, \quad y_3^* = 0, \quad \text{et} \quad w^* = 22$$

Contenu du module

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- Résolution des PL
- Dualité
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- Parcours eulériens et hamiltoniens des graphes
- Problème de plus court chemin
- Problèmes de transport et de flots

Post-Optimisation - Introduction

Après l'obtention de la solution optimale d'un programme linéaire, il est judicieux de se poser la question sur la **sensibilité** de cette solution aux paramètres du programme linéaire ?

Motivation

Les paramètres utilisés dans la formulation d'un PL ne sont en général que des estimations, donc elle peuvent subir des variations.

D'autres parts, dans un environnement dynamique évolutif, plusieurs facteurs peuvent changer régulièrement.

L'analyse de **sensibilité post-optimale** permet de déterminer la sensibilité d'un PL par rapport aux données :

- Une variation des données entraîne-t-elle un changement important de la solution optimale ?
- L'analyse poste-optimale permet de déterminer des intervalles de variations des données pour lesquels la base optimale n'est pas modifiée.

Post-Optimisation - Introduction

Objectifs

Prédire l'effet sur la fonction objectif suivant la variation des données :

- ① coefficients de la fonction objectif
- ② coefficients du membre de droite des contraintes

① Supposons que l'un des coefficients c_i de la fonction objectif a été changé. On cherche à déterminer un intervalle dans lequel peut varier le coefficient c_i sans que la solution de base optimale change (x_i^*) et à mesurer la variation de la fonction objectif z^* .

On parle d'**Intervalle d'optimalité**

② Supposons que l'un des coefficients b_i du membre de droite des contraintes a été changé. Quel est l'intervalle toléré du changement de ce coefficient afin de ne pas modifier la solution de base optimale (x_i^*) et quelle sera la variation de la fonction objectif z^*

On parle d'**Intervalle de réalisabilité**

Post-Optimisation - Coefficients de la fonction objectif

Exemple

Une société fabrique deux produits **A** et **B** en utilisant 3 machines.

Le tableau suivant donne le nombre d'heures nécessaires pour la fabrication de chaque produit au niveau de ces machines, la disponibilité de chaque machine et le prix de vente moyen de chaque produit.

Machine	A	B	Disponibilité (h)
Machine 1	1	2	20
Machine 2	2	1	22
Machine 3	1	1	12
Prix moyen (UM)	300	200	

Dans ce problème on cherche à identifier la fabrication optimale en **A** et **B** afin de maximiser le prix de vente total.

Post-Optimisation - Coefficients de la fonction objectif

Après

- modélisation de ce problème sous forme d'un programme linéaire.
- écriture du problème sous sa forme standard
- la résolution de ce problème par la méthode simplexe.

On obtient la solution suivante :

Solution Optimale

$$\left\{ \begin{array}{l} [\text{Max}] z = 300x_1 + 200x_2 \\ x_1 = 10 - e_2 + e_3 \\ x_2 = 2 + e_2 - 2e_3 \\ e_1 = 6 - e_2 + 3e_3 \end{array} \right.$$

La solution optimale $(x_1^*, x_2^*, e_1^*, e_2^*, e_3^*) = (10, 2, 6, 0, 0)$ et $z^* = 3400$.

Post-Optimisation - Coefficients de la fonction objectif

Sensibilité par rapport à c_1

Supposons que le prix de vente $c_1 = 300$ du produit A commence à fluctuer dans le marché et on souhaite savoir l'intervalle d'optimalité de c_1 qui permet de maintenir la même solution optimale $(x_1^*, x_2^*) = (10, 2)$.

Pour cela, on considère le nouveau coefficient $c'_1 = c_1 + \delta$.

On va avoir donc :

$$\begin{aligned} [\max] z &= (300 + \delta)x_1 + 200x_2 \\ &= (300 + \delta)(10 - e_2 + e_3) + 200(2 + e_2 - 2e_3) \\ &= (3400 + 10\delta) + (-100 - \delta)e_2 + (\delta - 100)e_3 \end{aligned}$$

Afin de garder la même solution optimale il faut que $-100 - \delta \leq 0$ et $\delta - 100 \leq 0$.

Ce qui conduit à $-100 \leq \delta \leq 100$.

Par la suite,

$$c'_1 \in [200; 400] \quad \text{et} \quad z^* \in [2400; 4400]$$

Post-Optimisation - Coefficients de la fonction objectif

Sensibilité par rapport à c_2

Supposons que le prix de vente $c_2 = 200$ du produit **B** commence à fluctuer dans le marché et on souhaite savoir l'intervalle d'optimalité de c_2 qui permet de maintenir la même solution optimale $(x_1^*, x_2^*) = (10, 2)$.

Pour cela, on considère le nouveau coefficient $c'_2 = c_2 + \delta$.

On va avoir donc :

$$\begin{aligned} [\max] \quad z &= 300x_1 + (200 + \delta)x_2 \\ &= 300(10 - e_2 + e_3) + (200 + \delta)(2 + e_2 - 2e_3) \\ &= (3400 + 2\delta) + (-100 + \delta)e_2 + (-100 - 2\delta)e_3 \end{aligned}$$

Afin de garder la même solution optimale il faut que $-100 + \delta \leq 0$ et $-100 - 2\delta \leq 0$. Ce qui conduit à $-50 \leq \delta \leq 100$.

Par la suite,

$$c'_2 \in [150; 300] \quad \text{et} \quad z^* \in [3300; 3600]$$

Post-Optimisation - Variation du second membre

Sensibilité par rapport à b_2 (Contrainte saturée)

Supposons que la disponibilité $b_2 = 22$ de la 2^{ème} machine peut évoluer et on souhaite savoir l'intervalle de réalisabilité de b_2 qui permet de maintenir la même solution optimale $(x_1^*, x_2^*, e_1^*, e_2^*, e_3^*) = (10, 2, 6, 0, 0)$.

Pour cela, on considère le nouveau coefficient du second membre $\mathbf{b}'_2 = \mathbf{b}_2 + \Delta$.

On obtient :

$$\left\{ \begin{array}{l} [\text{Max}] z = 3400 + 100\Delta - 100e_2 - 100e_3 \\ x_1 = 10 + \Delta - e_2 + e_3 \\ x_2 = 2 - \Delta + e_2 - 2e_3 \\ e_1 = 6 + \Delta - e_2 + 3e_3 \end{array} \right.$$

Pour $e_2^* = e_3^* = 0$, on doit avoir $10 + \Delta \geq 0$, $2 - \Delta \geq 0$ et $6 + \Delta \geq 0$.

Ceci donne $-6 \leq \Delta \leq 2$.

Par la suite,

$$\mathbf{b}'_2 \in [16; 24] \quad \text{et} \quad \mathbf{z}^* \in [2800; 3600]$$

Post-Optimisation - Variation du second membre

Sensibilité par rapport à b_3 (Contrainte saturée)

Supposons que la disponibilité $b_3 = 12$ de la 3^{ème} machine peut évoluer et on souhaite savoir l'intervalle de réalisabilité de b_3 qui permet de maintenir la même solution optimale $(x_1^*, x_2^*, e_1^*, e_2^*, e_3^*) = (10, 2, 6, 0, 0)$.

Pour cela, on considère le nouveau coefficient du second membre $\mathbf{b}'_3 = \mathbf{b}_3 + \Delta$.

On obtient :

$$\left\{ \begin{array}{l} [\text{Max}] z = 3400 + 100\Delta - 100e_2 - 100e_3 \\ x_1 = 10 - \Delta - e_2 + e_3 \\ x_2 = 2 + 2\Delta + e_2 - 2e_3 \\ e_1 = 6 - 3\Delta - e_2 + 3e_3 \end{array} \right.$$

Pour $e_2^* = e_3^* = 0$, on doit avoir $10 - \Delta \geq 0$, $2 + 2\Delta \geq 0$ et $6 - 3\Delta \geq 0$.

Ceci donne $-1 \leq \Delta \leq 2$. Par la suite,

$$\mathbf{b}'_3 \in [11; 14] \quad \text{et} \quad \mathbf{z}^* \in [3300; 3600]$$

Post-Optimisation - Variation du second membre

Sensibilité par rapport à b_1 (Contrainte marginale)

Supposons que la disponibilité $b_1 = 20$ de la 1^{ère} machine peut évoluer et on souhaite savoir l'intervalle de réalisabilité de b_1 qui permet de maintenir la même solution optimale $(x_1^*, x_2^*, e_1^*, e_2^*, e_3^*) = (10, 2, 6, 0, 0)$.

Pour cela, on considère le nouveau coefficient du second membre $\mathbf{b}'_1 = \mathbf{b}_1 + \Delta$.

On obtient :

$$\left\{ \begin{array}{l} [\text{Max}] z = 3400 - 100e_2 - 100e_3 \\ x_1 = 10 - e_2 + e_3 \\ x_2 = 2 + e_2 - 2e_3 \\ e_1 = 6 + \Delta - e_2 + 3e_3 \end{array} \right.$$

Pour $e_2^* = e_3^* = 0$, on doit avoir $6 + \Delta \geq 0$.

Ceci donne $\Delta \geq -6$.

Par la suite,

$$\mathbf{b}'_1 \in [14; +\infty[\quad \text{et} \quad \mathbf{z}^* = 3400$$

Programmation linéaire

- La programmation linéaire constitue un élément très répondu dans la modélisation en Recherche opérationnelle.
- Elle permet de résoudre plusieurs problèmes réels et de réaliser des gains considérables et d'assurer des optimisations importantes.
- Il reste très important d'accorder à l'interprétation économique du dual et à la phase post-optimisation une attention particulière afin de tirer profit au maximum de cet outil de RO incontournable
- L'étude de cette partie nous ouvre la voix vers l'étude de certains programme linéaire particuliers et d'aborder la problématique de détermination de la solution de base admissible initiale dans l'algorithme Simplexe.
- Explorer la programmation linéaire entière et la programmation non linéaire.



Recherche Opérationnelle R.O.

Partie 3: Optimisation Combinatoire

Pr. Abdessamad Kamouss

Cycle Ingénieur
ENSAM Casablanca

Contenu du module

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- Résolution des PL
- Dualité
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- Parcours eulériens et hamiltoniens des graphes
- Problème de plus court chemin
- Problèmes de transport et de flots

Contenu du module

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- Résolution des PL
- Dualité
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- Parcours eulériens et hamiltoniens des graphes
- Problème de plus court chemin
- Problèmes de transport et de flots

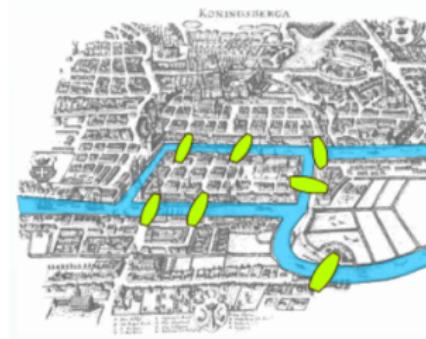
Définition des graphes

Définition des graphes

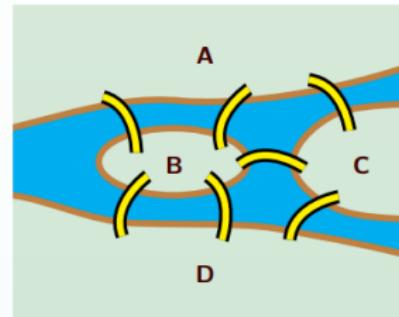
- Un graphe est tout d'abord :
 - un ensemble d'**éléments** ;
 - un ensemble de **relations** entre ces éléments.
 - Point de vue mathématique : une relation binaire
 - Point de vue pratique : représentation abstraite d'un réseau.
- Deux grandes familles :
 - les graphes **non-orientés** ;
 - les graphes **orientés**.
- Nombreuses conceptualisations et modélisations possibles.
- Utilisés dans des domaines très variés : économie, informatique, industrie, chimie, sociologie,...

Définition des graphes

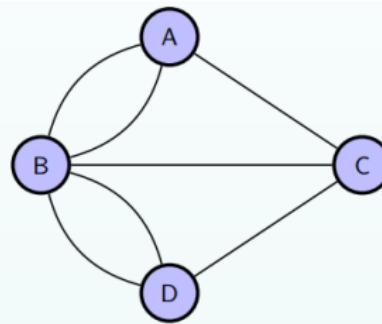
Problème des sept ponts de Königsberg (Euler, 1735)



↔



Représentation sous forme de graphe :



Définition des graphes

La théorie des graphes étudie des structures discrètes en forme de «réseaux»

Définition d'un graphe

Un graphe (S, \mathcal{A}) est la donnée de deux ensembles d'objets :

- S : l'ensemble des sommets ou noeuds.
- \mathcal{A} : un sous ensemble de $S \times S$ représentant l'ensemble des arêtes ou arcs.
 - **arêtes** symbolisant une relation symétrique,
 - **arcs** symbolisant une relation orientée.

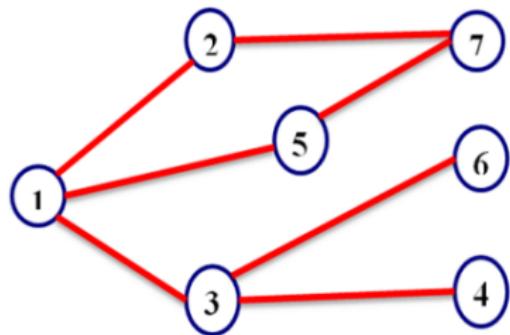
Types de graphes

Nous pouvons alors distinguer deux familles de graphes :

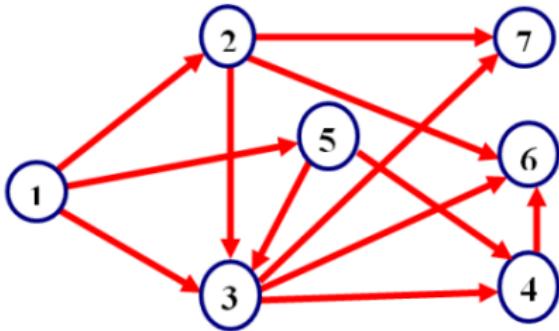
- les graphes dirigés (en anglais, directed graphs ou digraph) ;
- les graphes symétriques (en anglais, undirected graphs).

Types de graphes

Graphe Symétrique



Graphe Orienté



Modélisation

- réseaux de transports routier, d'eau, d'électricité
- réseaux informatiques
- réseaux sociaux : (Facebook : graphe non orienté, twiter : graphe orienté)
- graphe du web
- réseau de transports de données (téléphonie, wifi, réseaux informatique...)
- représentation d'un algorithme, du déroulement d'un jeu ;
- réseaux de régulation génétique
- organisation logistique
- ordonnancement de projet
- ...

Définition (Graphe symétrique)

Un **graphe non-orienté ou symétrique** est une relation binaire \mathcal{G} sur un ensemble \mathcal{A} , notée $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ où \mathcal{S} est l'ensemble des sommets et \mathcal{A} l'ensemble des arêtes qui est formé des $\{x, y\} \in \mathcal{A}$ qui sont en relation.

Vocabulaire des graphes symétriques

- L'**ordre** d'un graphe est son nombre de sommets.
- La **taille** d'un graphe est son nombre d'arêtes.
- Si $\{x, y\}$ est une arête, x et y sont voisins et sont appelés les extrémités de cet arête.

Définition (Graphe orienté)

Un **graphe dirigé ou orienté** est une relation binaire \mathcal{G} sur un ensemble \mathcal{A} , notée $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ où \mathcal{S} est l'ensemble des sommets et \mathcal{A} est l'ensemble des arcs qui sont formé des couples $(x, y) \in \mathcal{S} \times \mathcal{S}$ qui sont en relation.

Vocabulaire des graphes orientés

- L'**ordre** d'un graphe est son nombre de sommets.
- La **taille** d'un graphe est son nombre d'arcs.
- Pour l'arc (x, y) , on dit que :
 - x et y sont adjacents.
 - x est son **origine** et y son **extrémité**.
 - y est un **successeur** (ou suivant) de x
 - x est un **prédecesseur** (ou précédent) de y .

Degré d'un sommet

Définitions (Degré)

- Un **graphe simple** est un graphe qui ne contient pas des multi-arêtes ou multi-arcs, c'est-à-dire qu'entre deux sommets il n'existe qu'une seule arête ou un seul arc dans chaque sens.
- Soit x un sommet d'un graphe symétrique. Le **degré** de x , noté $d(x)$, est le nombre d'arêtes incidentes à x , c'est-à-dire contenant x .
- Soit x un sommet d'un graphe orienté. On note $d^+(x)$ le nombre d'arcs ayant x comme extrémité initiale, et $d^-(x)$ le nombre d'arcs ayant x comme extrémité finale. Ainsi, on a le **degré de** x , noté $d(x)$ vérifie : $d(x) = d^+(x) + d^-(x)$.

On a :

- $\sum_{x \in S} d(x) = 2|A|$.
- $\sum_{x \in S} d^+(x) = \sum_{x \in S} d^-(x) = |A|$.

représentation de Graphe - Matrice d'adjacence

Afin de représenter un graphe, on a plusieurs possibilités :

- Matrice d'adjacence
- Matrice d'incidence
- Listes d'adjacence
- ...

Matrice d'adjacence d'un graphe simple

Soit $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ un graphe simple de n sommets numérotés de 1 à n .

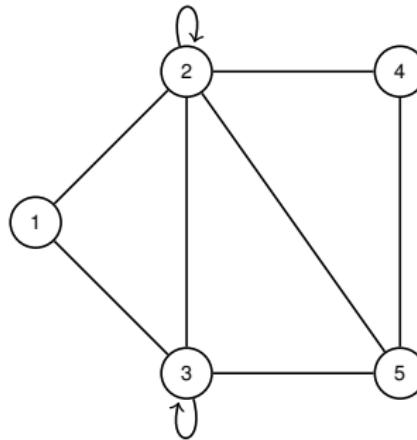
La matrice d'adjacence $A = (a_{ij})$ de \mathcal{G} est la matrice carrée d'ordre n où chaque ligne et chaque colonne représentent un sommet et qui est définie par :

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & & \vdots & & \vdots \\ a_{n1} & \cdots & a_{nj} & \cdots & a_{nn} \end{pmatrix}$$

Graphe - Matrice d'adjacence

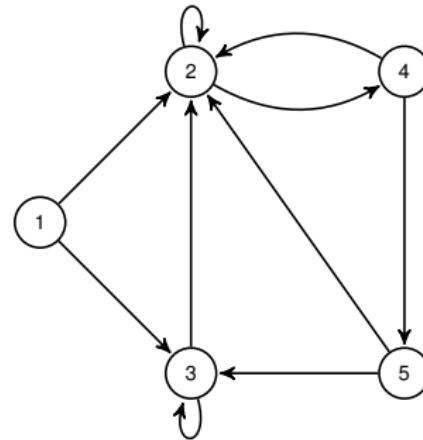
Graphe symétrique non valué

$$a_{ij} = \begin{cases} 1 & \text{si } \{i, j\} \in \mathcal{A} \\ 0 & \text{sinon.} \end{cases}$$



Graphe orienté non valué

$$a_{ij} = \begin{cases} 1 & \text{si } (i, j) \in \mathcal{A} \\ 0 & \text{sinon} \end{cases}$$



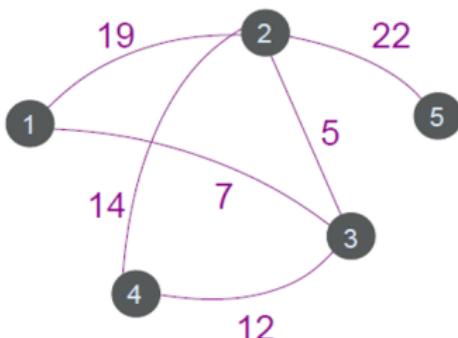
$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Graphe - Matrice d'adjacence

Graphe symétrique valué

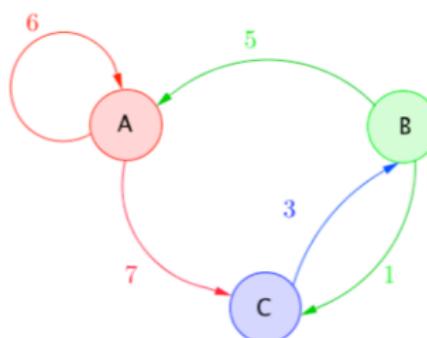
$$a_{ij} = \begin{cases} p & \text{si } \{i, j\} \in \mathcal{A} \\ 0 & \text{si } i = j \\ +\infty & \text{sinon.} \end{cases}$$



$$A = \begin{pmatrix} 0 & 19 & 7 & +\infty & +\infty \\ 19 & 0 & 5 & 14 & 22 \\ 7 & 5 & 0 & 12 & +\infty \\ +\infty & 14 & 12 & 0 & +\infty \\ +\infty & 22 & +\infty & +\infty & 0 \end{pmatrix}$$

Graphe orienté valué

$$a_{ij} = \begin{cases} p & \text{si } (i, j) \in \mathcal{A} \\ +\infty & \text{sinon} \end{cases}$$



$$A = \begin{pmatrix} 6 & +\infty & 7 \\ 5 & +\infty & 1 \\ +\infty & 3 & +\infty \end{pmatrix}$$

Contenu du module

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- Résolution des PL
- Dualité
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- **Parcours eulériens et hamiltoniens des graphes**
- Problème de plus court chemin
- Problèmes de transport et de flots

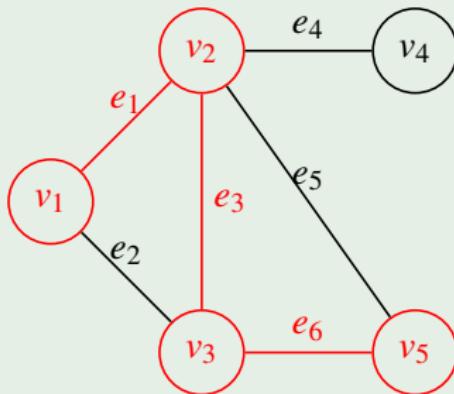
Chaîne dans un graphe symétrique

- Une **chaîne** dans un graphe G est une suite ayant pour éléments alternativement des sommets et des arêtes, commençant et se terminant par un sommet et telle que chaque arête est encadré par ses extrémités.
- On dit que la chaîne relie le premier sommet et le dernier sommet.
- $I(C)$ est la **longueur d'une chaîne** C qui représente son nombre d'arêtes.
- $d(x,y)$ est la **distance** entre deux sommets x et y qui représente la longueur de la plus petite chaîne les reliant.
- $\delta(G)$ est le **diamètre d'un graphe** G qui représente la plus grande distance entre deux sommets.

Graphe - Chaîne

Exemple

Dans le graphe G suivant :



- On considère la chaîne : $\mathcal{C} = (v_1, e_1, v_2, e_3, v_3, e_6, v_5)$. $l(\mathcal{C}) = 3$.
- La distance entre v_1 et v_5 est $d(v_1, v_5) = 2$.
- Le diamètre de G est $\delta(G) = 2$

Graphe - Chaîne

Les graphes de diamètre 0 sont les graphes stables (sans arête)

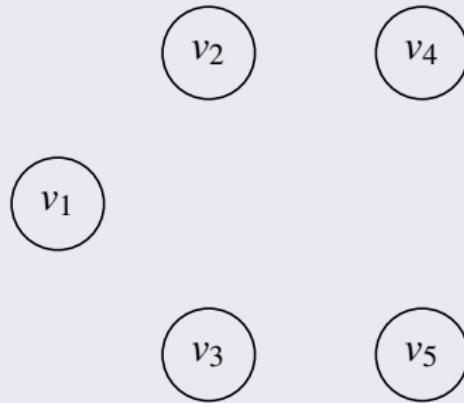
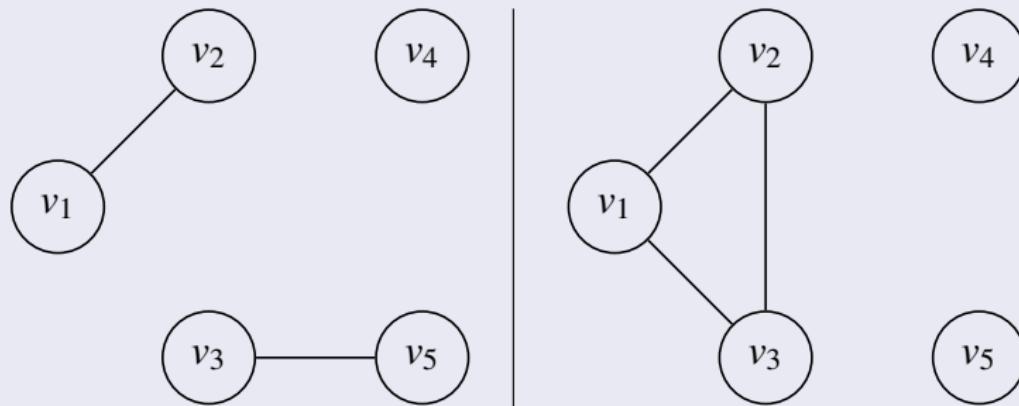


Figure – Graphe stable

Graphe - Chaîne

Graphes de diamètre 1



Chaîne élémentaire et chaîne simple

- **Chaîne élémentaire :**

si chaque **sommet** apparaît au plus une fois.

- **Chaîne simple :**

si chaque **arête** apparaît au plus une fois.

- **Chaîne fermée :**

Une chaîne dont les extrémités sont les mêmes.

- **Cycle**

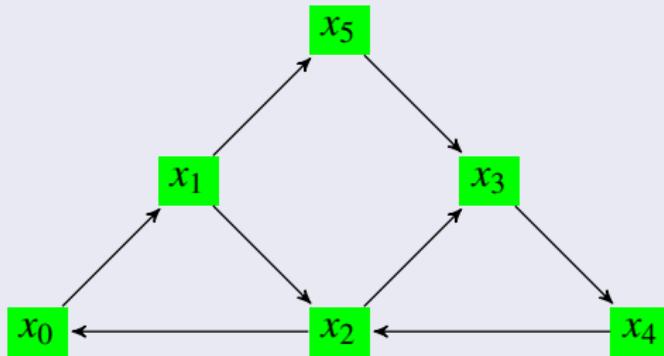
Une chaîne simple et fermée.

Chemin dans un digraphe

- Un **chemin de x à y** est une suite finie d'arcs consécutifs, reliant x à y .
La **longueur d'un chemin** est le nombre d'arcs du chemin.
- **Chemin élémentaire :**
un chemin ne passant pas deux fois par un même **sommet** du graphe
- **Chemin simple :**
un chemin ne passant pas deux fois par un même **arc** du graphe.
- **Chemin fermée :**
un chemin dont l'origine et l'extrémité coïncident.
- **Circuit :**
un chemin simple dont les deux extrémités sont identiques.

Graphe - Chemin - Circuit

Exemple



- $(x_0, x_1, x_2, x_3, x_4)$ est un chemin.
- $(x_0, x_1, x_2, x_3, x_4, x_2, x_0)$ est un circuit.

Graphe eulérien et graphe semi-eulérien

- **Graphe eulérien**

Un graphe symétrique est dit **eulérien** s'il abrite un cycle (chaîne simple fermée) passant une et une seule fois par toutes les arêtes.

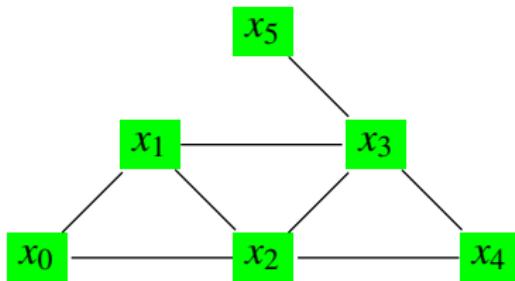
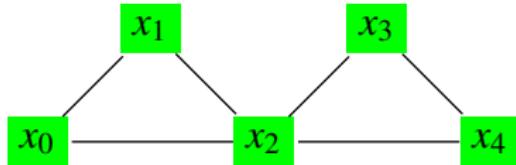
- **Graphe semi-eulérien**

Un graphe symétrique est dit **semi-eulérien** s'il abrite une chaîne simple passant une et une seule fois par toutes les arêtes.

Intuitivement :

Un graphe est eulérien (ou semi-eulérien) s'il est possible de le dessiner sans lever le crayon et sans passer deux fois par le même trait.

Graphe eulérien



Graphe eulérien

Graphe semi-eulérien

Graphe hamiltonien et graphe semi-hamiltonien

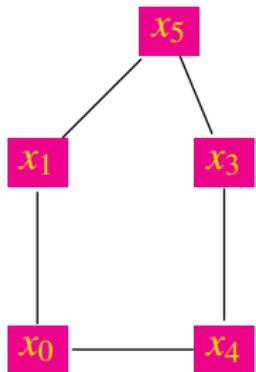
- **Graphe hamiltonien**

Un graphe symétrique est dit **hamiltonien** s'il abrite un cycle (chaîne élémentaire fermée) passant une et une seule fois par tous les sommets.

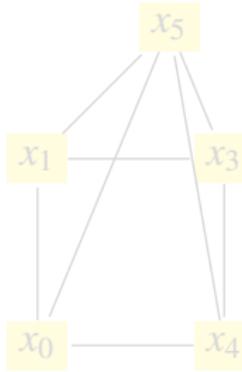
- **Graphe semi-hamiltonien**

Un graphe symétrique est dit **semi-hamiltonien** s'il abrite une chaîne (chaîne élémentaire) passant une et une seule fois par tous les sommets.

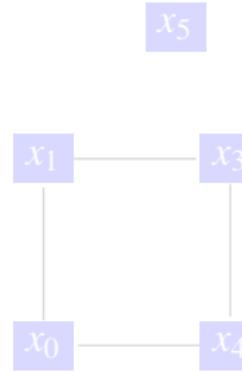
Graphe eulérien/hamiltonien - Exemples



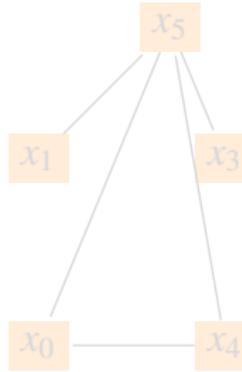
H et E



H et \bar{E}

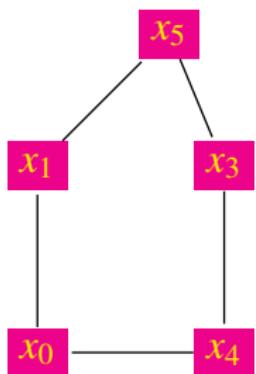


\bar{H} et E

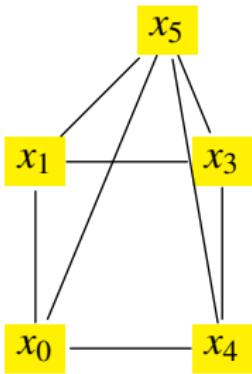


\bar{H} et \bar{E}

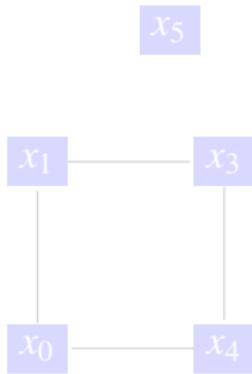
Graphe eulérien/hamiltonien - Exemples



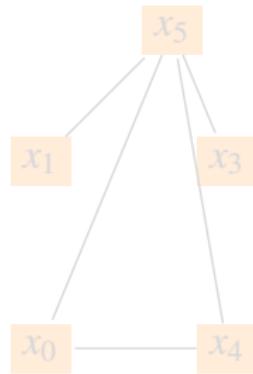
H et E



H et \bar{E}

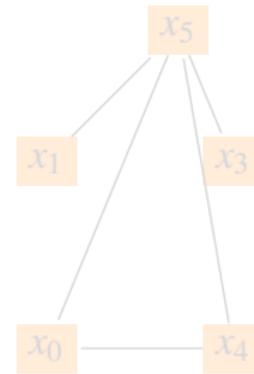
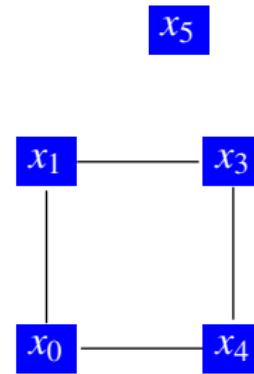
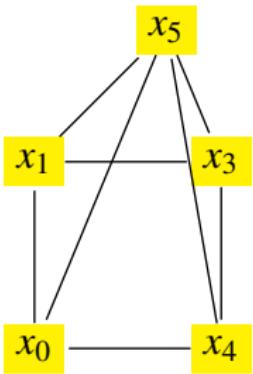
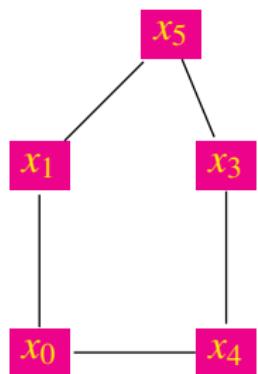


\bar{H} et E

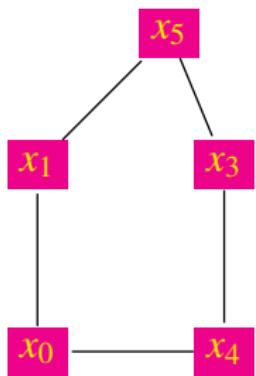


\bar{H} et \bar{E}

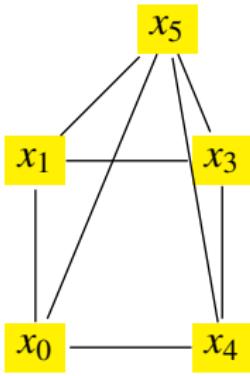
Graphe eulérien/hamiltonien - Exemples



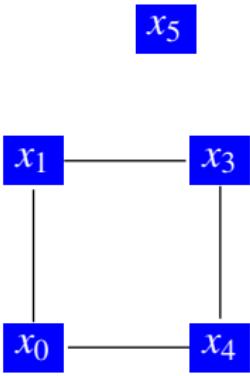
Graphe eulérien/hamiltonien - Exemples



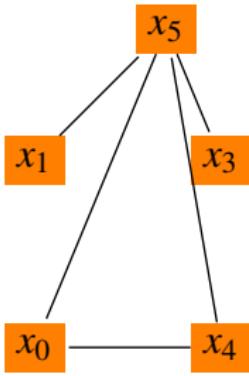
H et E



H et \bar{E}



\bar{H} et E



\bar{H} et \bar{E}

Caractérisation des graphes eulériens et semi-eulériens

Théorème d'Euler

Soit $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ un graphe symétrique. Notons par $d(x)$ le degré de x .

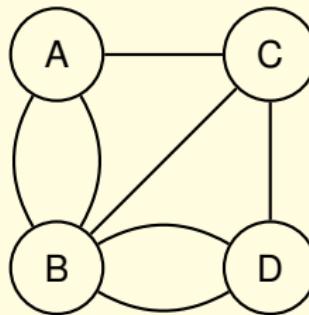
- (i) Si pour tout $x \in \mathcal{S}$ on a $d(x)$ est pair , alors \mathcal{G} admet **un cycle eulérien**.
- (ii) Si il existe uniquement deux sommets distincts x_1 et x_2 qui ont **des degrés impaires** pour tout $x \in \mathcal{S} \setminus \{x_1, x_2\}$ $d(x)$ est pair , alors \mathcal{G} abrite **une chaîne eulérienne** d'extrémités x_1 et x_2 .
- (iii) Dans tous les autres cas \mathcal{G} **n'abrite pas de chaîne eulérienne**.

Graphe eulérien

Réponse négative au problème d'Euler

Euler souhaitait se promener dans sa ville natale en passant une et une seule fois par chaque pont.

Donc le graphe doit être eulérien.



Mais, d'après le résultat précédent, ce n'est pas possible puisque les degrés des sommets ne sont pas tous pairs.

Graphe hamiltonien

Caractérisation des graphes hamiltoniens

Théorème de Dirac

On considère un graphe simple à n sommets ($n \geq 3$).

Si chaque sommet a un degré au moins égale à $\frac{n}{2}$ alors **le graphe est hamiltonien.**

Théorème de Ore

On considère un graphe simple à n sommets ($n \geq 3$).

Si la somme des degrés de toute paire de sommets non adjacents vaut au moins n alors **le graphe est hamiltonien.**

Attention. Les deux résultats précédents représentent des conditions suffisantes mais qui ne sont pas nécessaires.

Contenu du module

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- Résolution des PL
- Dualité
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- Parcours eulériens et hamiltoniens des graphes
- **Problème de plus court chemin**
- Problèmes de transport et de flots

Problème de plus court chemin

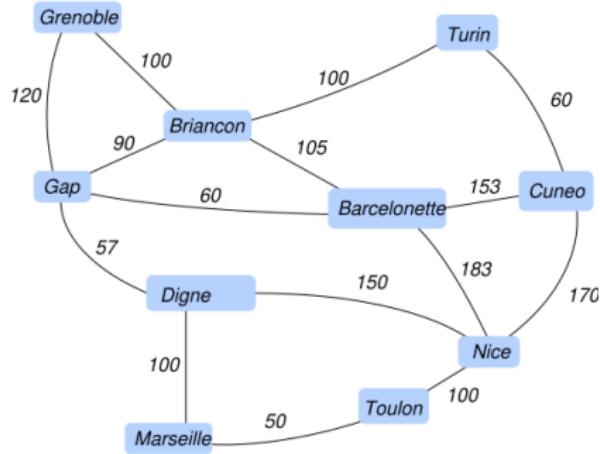
Les problèmes de cheminement dans les graphes comptent parmi les problèmes les plus anciens de la théorie des graphes.

En particulier la recherche d'un plus court chemin **PCC** constitue un problème amplement investi et fait partie des plus importants problèmes étudiés par leurs applications.

Plusieurs applications :

- Recherche d'un trajet le plus court.
- Recherche de la route la plus rapide.
- Recherche de la stratégie optimale.
- ...

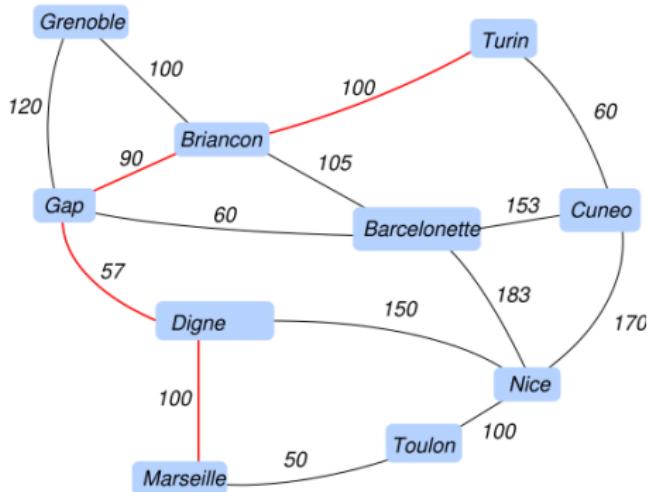
Trajet le plus court entre deux villes : On considère la carte suivante permettant de relier Marseille et Turin :



On cherche à déterminer sur cette carte le plus court chemin entre ces deux villes.

Trajet le plus court entre deux villes :

Le plus court chemin entre Marseille et Turin :



Définitions

- **Une racine** dans un graphe orienté (resp. non-orienté) est un sommet r tel qu'il existe un chemin (resp. une chaîne) de r à u pour tout sommet $u \neq r$.
- Un circuit (resp. cycle) de longueur négative est appelé un **circuit (resp. cycle) absorbant**.

Théorème

Une condition nécessaire et suffisante pour trouver un plus court chemin d'un sommet **s** à tous les sommets du graphe est que :

- **s** soit racine du graphe,
- le graphe ne contient pas de circuit/cycle absorbant.

Théorème de Sous-optimalité

Soit C un plus court chemin de u à v et u', v' deux sommets de C .

Alors le sous-chemin de C de u' à v' est aussi un plus court chemin de u' à v' .

Preuve :

On considère le sous-chemin chemin C_1 de u' à v' contenu dans le chemin C .

Supposons par absurdité qu'il existe un autre chemin C_2 de u' à v' qui est plus court que C_1 .

Donc le chemin C' de u à v constitué des sous-chemins suivants : sous-chemin de u à u' , C_2 et sous-chemin de v' à v sera plus court que le chemin C .

Ce qui représente une contradiction.

Algorithmes de recherche du PCC

Nous allons étudier deux algorithmes qui permettent de résoudre des problèmes de recherche de plus courts chemins à racine unique :

- un algorithme (**Dijkstra**) qui peut être utilisé pour des graphes dont les pondérations de tous les arêtes ou les arcs sont positives ou nulles ;
- un algorithme (**Bellman-Ford**) qui peut être utilisé pour n'importe quel graphe avec éventuellement des pondérations négatives mais qui ne comportant pas de circuit absorbant.

Algorithme de Dijkstra



Algorithme de Dijkstra

Dijkstra est un algorithme de recherche de distance et de plus court chemin entre **un sommet racine fixé s et tous les autres sommets** d'un graphe.

Il est composée de deux étapes :

Initialisation :

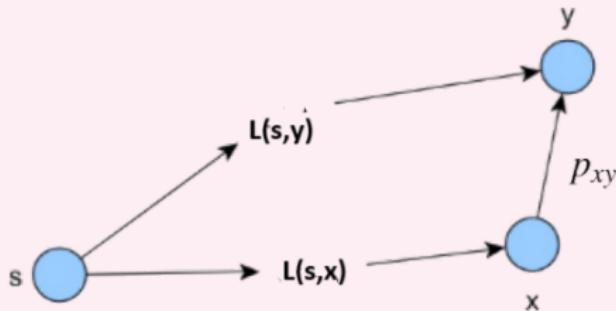
- Attribuer à chaque sommet successeur de **s** une distance égale au poid entre **s** et ce successeur.
- Les autres sommets non reliés directement à **s**, la valeur $+\infty$.

On commence les itérations :

- Au début de chaque itération, on choisit un sommet **x** parmi ceux qui n'ont pas encore été traités, dont la distance à **s** est minimale.
- Pour chaque successeur **y** de **x**, on compare la valeur actuelle de la longueur du chemin déjà trouvé **L(s,y)** avec la longueur du nouveau chemin **L(s,x) + p_{xy}**.

Algorithme de Dijkstra

Algorithme de Dijkstra



- On met alors éventuellement à jour la distance $L(s,y)$ si la nouvelle valeur trouvée est meilleure.
- Quand tous les successeurs du sommet x ont été examinés, on rajoute x à la liste des sommets traités, et l'on recommence par une nouvelle itération, avec le choix d'un nouveau sommet.
- Les itérations se terminent lorsque la totalité des sommets sont traités.

Algorithme de Dijkstra

Notations

Soit $G = (S, \mathcal{A})$ un graphe orienté ou symétrique à valuation **positives ou nulles**.

Soient s le sommet racine à partir duquel on va chercher les plus courts chemins.

Afin d'exécuter l'algorithme Dijkstra on va définir les tableaux **L**, **P** et **M** suivants :

- **L** un tableau de n cases destiné à contenir les distances de s aux autres sommets.
- **P** un tableau de n cases destiné à contenir les prédécesseurs de chacun des sommets dans un plus court chemin d'origine s .
- **M** la liste de tous les sommets déjà traités (et donc $S \setminus M$ sera la liste de tous les sommets à traiter).

Algorithme de Dijkstra

Initialisation

```
1:  $L[s] \leftarrow 0$ 
2:  $P[s] \leftarrow s$  ( $s$  est la racine)
3: for ( $x \neq s$ ) do
4:   if ( $x$  est un successeur de  $s$ ) then
5:      $L[x] \leftarrow p_{sx}$ 
6:      $P[x] \leftarrow s$ 
7:   else
8:      $L[x] \leftarrow +\infty$ 
9:      $P[x] \leftarrow \emptyset$ 
10:  end if
11: end for
12:  $M \leftarrow \{s\}$ 
```

Algorithme de Dijkstra

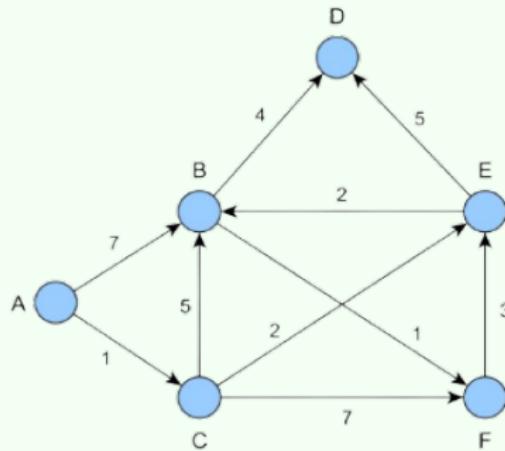
Traitement

```
1: while ( $M \neq S$ ) do
2:   choisir  $x \in S \setminus M$  tq : pour tout  $y \in S \setminus M$ ,  $L[x] \leq L[y]$ 
3:   for ( $y \in S \setminus M$  tel que  $y$  soit un successeur de  $x$ ) do
4:     if ( $L[x] + p_{xy} < L[y]$ ) then
5:        $L[y] \leftarrow L[x] + p_{xy}$ 
6:        $P[y] \leftarrow x$ 
7:     end if
8:   end for
9:    $M \leftarrow M \cup \{x\}$ 
10: end while
```

Algorithme de Dijkstra

Exemple

Considérons le graphe G orienté valué dont la représentation sagittale est la suivante :

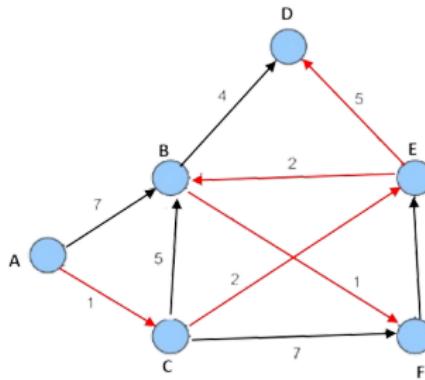


Question : Appliquer l'algorithme de Dijkstra à ce graphe en partant du sommet A

Algorithme de Dijkstra

Initialisation et Traitement :

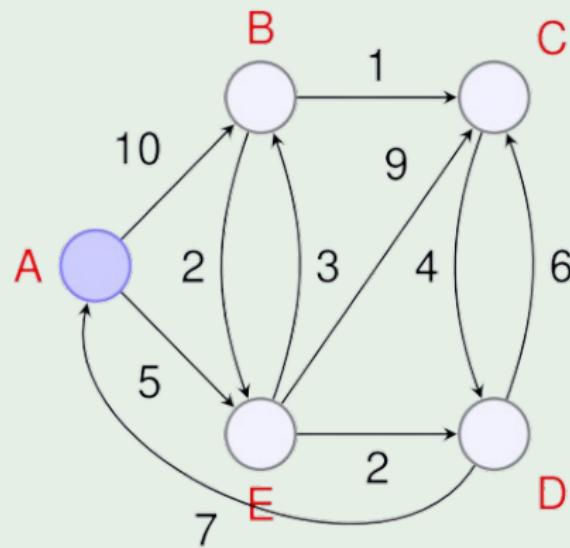
	L						P					
	A	B	C	D	E	F	A	B	C	D	E	F
A	0	7	1	$+\infty$	$+\infty$	$+\infty$	A	A	A	-	-	-
A, C	0	6	1	$+\infty$	3	8	A	C	A	-	C	C
A, C, E	0	5	1	8	3	8	A	E	A	E	C	C
A, C, E, B	0	5	1	8	3	6	A	E	A	E	C	B
A, C, E, B, F	0	5	1	8	3	6	A	E	A	E	C	B
A, C, E, B, F, D	0	5	1	8	3	6	A	E	A	E	C	B



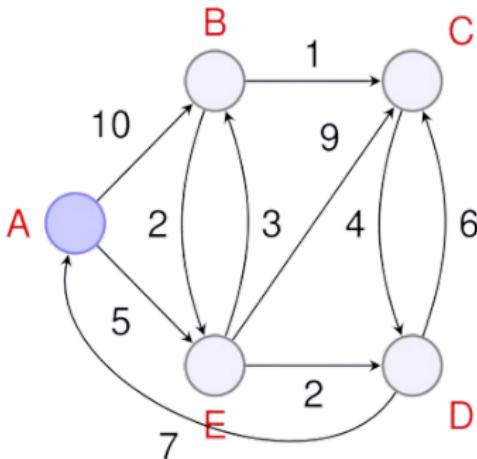
Algorithme de Dijkstra

Exemple (Application de l'algorithme de Dijkstra)

Donner les plus courts chemins d'origine le sommet A dans le graphe suivant :



Algorithme de Dijkstra

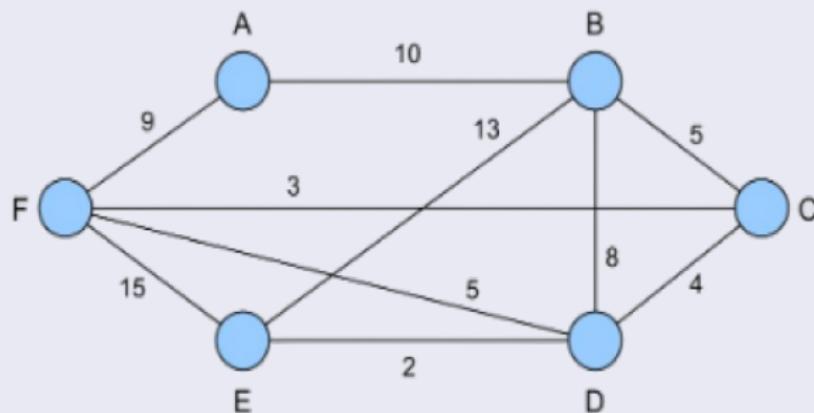


	L					P				
	A	B	C	D	E	A	B	C	D	E
A	0	10	$+\infty$	$+\infty$	5	A	A	-	-	A
AE	0	8	14	7	5	A	E	E	E	A
AED	0	8	13	7	5	A	E	D	E	A
AEDB	0	8	9	7	5	A	E	B	E	A
AEDBC	0	8	9	7	5	A	E	B	E	A

Algorithme de Dijkstra

Exercice

Considérons le graphe non orienté valué dont la représentation sagittale est la suivante :



Question : Appliquer l'algorithme de Dijkstra à ce graphe en partant du sommet A

Algorithme de Bellman-Ford



Algorithme de Bellman-Ford

Bellman-Ford est un algorithme de recherche de distance et de plus court chemin entre **un sommet racine fixé s** et **tous les autres sommets** d'un graphe.

Notations

- **s** le sommet du graphe à partir duquel on va rechercher tous les plus courts chemins vers les autres sommets du graphe.
- **L** le tableau de dimension n qui contient les distances de **s** aux autres sommets.
- **P** le tableau de dimension n qui contient le prédécesseur de chacun des sommets dans un plus court chemin d'origine **s** .
- **Continue** un booléen indiquant la fin ou non de l'algorithme.

Algorithme de Bellman-Ford

Initialisation

```
1:  $L[s] = 0$ 
2:  $P[s] = s$ 
3: for ( $x \neq s$ ) do
4:    $L[x] \leftarrow +\infty$ 
5:    $P[x] \leftarrow \emptyset$ 
6: end for
7: Continue  $\leftarrow$  Vrai
```

Algorithme de Bellman-Ford

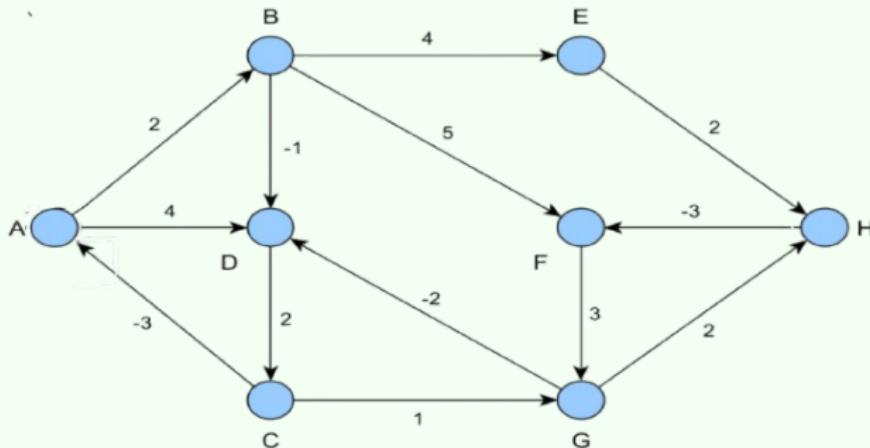
Traitement

```
1: while (Continue) do
2:   Continue  $\leftarrow$  Faux
3:   for ( $x \in S$ ) do
4:     for ( $y \in S$  tq  $y$  soit un successeur de  $x$ ) do
5:       if  $L[x] + p_{xy} < L[y]$  then
6:          $L[y] = L[x] + p_{xy}$ 
7:          $P[y] = x$ 
8:         Continue  $\leftarrow$  Vrai
9:       end if
10:      end for
11:    end for
12: end while
```

Algorithme de Bellman-Ford

Exemple

Considérons le graphe G orienté valué dont la représentation sagittale est la suivante :

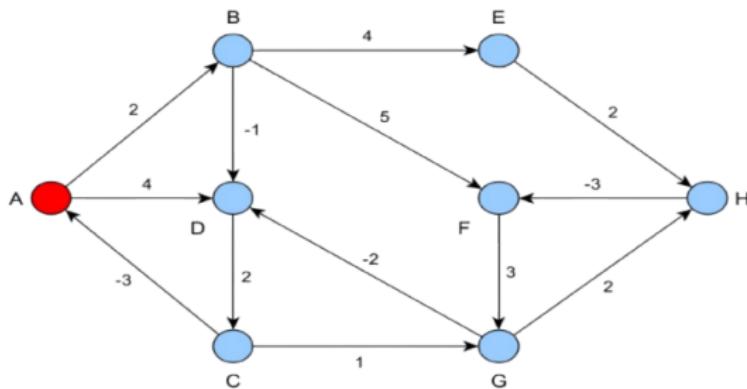


Question : Appliquer l'algorithme de Bellman-Ford à ce graphe en partant du sommet A

Algorithme de Bellman-Ford

Initialisation

Durant cette phase d'initialisation on commence par le sommet racine A et on va attribuer la valeur $+\infty$ à la table L et la valeur \emptyset à la table P pour tous les sommets à l'exception de l'origine A



	L								P							
	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
A	0	∞	A	-	-	-	-	-	-	-						

Algorithme de Bellman-Ford

Traitement : 1^{ère} itération

Dans cette itération on procède au traitement de la totalité des sommets dans l'ordre alphabétique.

	L								P							
	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
Init	0	∞	A	-	-	-	-	-	-	-						
A	0	2	∞	4	∞	∞	∞	∞	A	A	-	A	-	-	-	-
B	0	2	∞	1	6	7	∞	∞	A	A	-	B	B	B	-	-
C	0	2	∞	1	6	7	∞	∞	A	A	-	B	B	B	-	-
D	0	2	3	1	6	7	∞	∞	A	A	D	B	B	B	-	-
E	0	2	3	1	6	7	∞	8	A	A	D	B	B	B	-	E
F	0	2	3	1	6	7	10	8	A	A	D	B	B	B	F	E
G	0	2	3	1	6	7	10	8	A	A	D	B	B	B	F	E
H	0	2	3	1	6	5	10	8	A	A	D	B	B	H	F	E

Algorithme de Bellman-Ford

Traitement : 2^{ème} itération

Dans cette itération on procède au traitement de la totalité des sommets dans l'ordre alphabétique.

	L								P							
	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
pré	0	2	3	1	6	5	10	8	A	A	D	B	B	H	F	E
A	0	2	3	1	6	5	10	8	A	A	D	B	B	H	F	E
B	0	2	3	1	6	5	10	8	A	A	D	B	B	H	F	E
C	0	2	3	1	6	5	4	8	A	A	D	B	B	H	C	E
D	0	2	3	1	6	5	4	8	A	A	D	B	B	H	C	E
E	0	2	3	1	6	5	4	8	A	A	D	B	B	H	C	E
F	0	2	3	1	6	5	4	8	A	A	D	B	B	H	C	E
G	0	2	3	1	6	5	4	6	A	A	D	B	B	H	C	G
H	0	2	3	1	6	3	4	6	A	A	D	B	B	H	C	G

Algorithme de Bellman-Ford

Traitement : 3^{ème} itération

Dans cette itération on procède au traitement de la totalité des sommets dans l'ordre alphabétique.

	L								P							
	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
pré	0	2	3	1	6	3	4	6	A	A	D	B	B	H	C	G
A	0	2	3	1	6	3	4	6	A	A	D	B	B	H	C	G
B	0	2	3	1	6	3	4	6	A	A	D	B	B	H	C	G
C	0	2	3	1	6	3	4	6	A	A	D	B	B	H	C	G
D	0	2	3	1	6	3	4	6	A	A	D	B	B	H	C	G
E	0	2	3	1	6	3	4	6	A	A	D	B	B	H	C	G
F	0	2	3	1	6	3	4	6	A	A	D	B	B	H	C	G
G	0	2	3	1	6	3	4	6	A	A	D	B	B	H	C	G
H	0	2	3	1	6	3	4	6	A	A	D	B	B	H	C	G

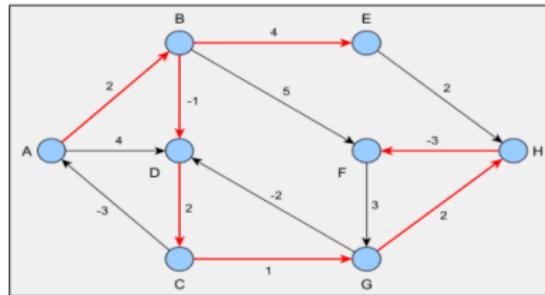
Algorithme de Bellman-Ford

Traitement : Convergence de l'algorithme

Après le passage d'une itération sans aucun changement la variable **Continue** reste sur la valeur **Faux** donc l'algorithme converge.

On obtient donc le résultat suivant :

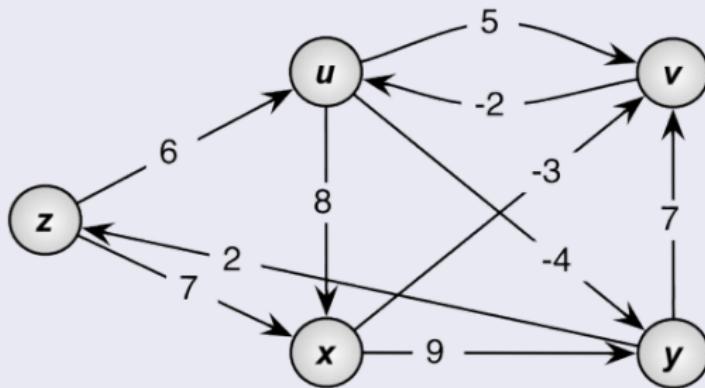
	L								P							
	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
Fin	0	2	3	1	6	3	4	6	A	A	D	B	B	H	C	G



Algorithme de Bellman-Ford

Exercice

Considérons le graphe non orienté valué dont la représentation sagittale est la suivante :



Question : Appliquer l'algorithme de Bellman-Ford à ce graphe en partant du sommet z

Contenu du module

1 Introduction à la recherche opérationnelle

2 Programmation linéaire

- Modélisation en Programmation linéaire
- Résolution des PL
- Dualité
- Post-Optimisation

3 Optimisation combinatoire

- Généralités sur les graphes
- Parcours eulériens et hamiltoniens des graphes
- Problème de plus court chemin
- **Problèmes de transport et de flots**

Introduction : Problème de transport

Problème de transport - Flot maximal

- Le problème du **flot maximal** sur un **réseau de transport** est une problématique centrale dans la recherche opérationnelle et l'optimisation combinatoire.
- Il consiste à maximiser la quantité de flot qui peut être acheminée d'une ou plusieurs **sources** vers une ou plusieurs **destinations** à travers un **réseau de transport** (ou un graphe orienté), tout en respectant certaines contraintes.
- Le réseau est représenté par un graphe où les noeuds sont des points (représentant des villes, des serveurs, ou tout autre point de connexion), et les arcs sont des liens entre ces noeuds, ayant chacun une capacité maximale de transport.
- L'objectif est de déterminer le **flot maximal** que l'on peut envoyer des sources (noeuds de départ) aux puits (noeuds d'arrivée), en tenant compte des capacités des arcs.

Introduction : Problème de transport - Flot maximal

Applications

Le problème du flot maximal a de nombreuses applications pratiques, notamment dans :

- Gestion des réseaux de transport
 - Trafic routier et transport public
 - Aéroports et ports
- Réseaux de communication et transmission de données
 - Internet et réseaux informatiques
 - Télécommunications
- Distribution des ressources
 - Réseaux d'eau, d'électricité ou de gaz
 - Logistique et approvisionnement
- Planification et gestion des projets
 - Gestion des flux de production
 - Planification de projets

Introduction : Problème de transport - Flot maximal

Applications

- Assignation de tâches ou de ressources
 - Affectation des travailleurs
 - Gestion des matchs sportifs
- Biologie et médecine
 - Réseaux sanguins ou lymphatiques :
 - Recherche génomique et biologie computationnelle
- Problèmes financiers
 - Flux de capitaux en investissement
 - Flux des transferts financiers
- Matching bipartite et problèmes d'affectation
- ...

Réseau de transport

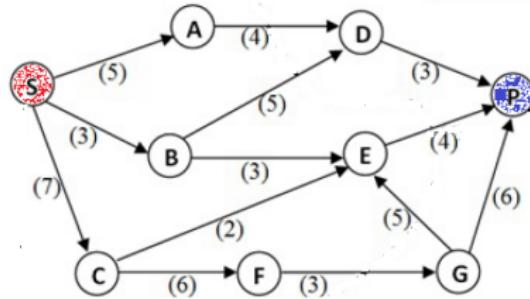
Définition (Réseau de transport)

On appelle **réseau de transport** un graphe orienté simple antisymétrique $G = (S, \mathcal{A})$ valué positivement sans boucle, ayant :

- une racine **s**,
- un puits **p**,

Les valuations des arcs sont appelées **capacités** c_{ij} .

On dénote $C = \{c_{ij}, (i, j) \in \mathcal{A}\}$ et on dit que $G = [S, \mathcal{A}, C]$ est un **réseau de transport avec capacités**.



Définition (Flot)

On appelle **flot réalisable** sur un **réseau de transport avec capacités** $G = [S, \mathcal{A}, C]$ une application $\phi : \mathcal{A} \rightarrow \mathbb{R}$ qui vérifie :

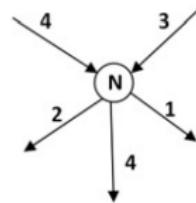
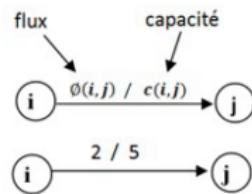
1 Les contraintes de capacité :

Pour tout arc (i, j) de \mathcal{A} on a $0 \leq \phi_{ij} \leq c_{ij}$;

2 Les contraintes de conservation (Kirchoff) :

Pour tout sommet i de $S \setminus \{s, p\}$, on a : $\sum_{k \in \mathcal{A}^-(i)} \phi_{ki} = \sum_{j \in \mathcal{A}^+(i)} \phi_{ij}$

$$\sum_{k \in \mathcal{A}^+(s)} \phi_{sk} = \sum_{j \in \mathcal{A}^-(p)} \phi_{jp}$$



Définitions (Flot)

Dans le contexte de la définition précédente :

- ① $V(\phi) = \sum_{k \in \mathcal{A}^+(s)} \phi_{sk} = \sum_{j \in \mathcal{A}^-(p)} \phi_{jp}$ est appelée la **valeur du flot**.

Elle représente le volume total que met en circulation le flot réalisable ϕ dans le réseau de transport.

- ② La quantité $C_{r_{ij}} = c_{ij} - \phi_{ij}$ est **la capacité résiduelle** de l'arc (i, j) .
- ③ Si $\phi_{ij} = c_{ij}$, on dit dans ce cas que l'arc (i, j) est **saturé**.

Lemme (généralisation des contraintes de Kirchoff)

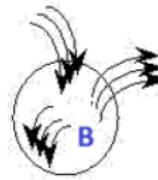
Si B est un sous-ensemble de S alors :

*le **flot sortant** de B est égal au **flot entrant** dans B .*

Preuve :

On somme l'égalité de Kirchoff sur l'ensemble des sommets de B .

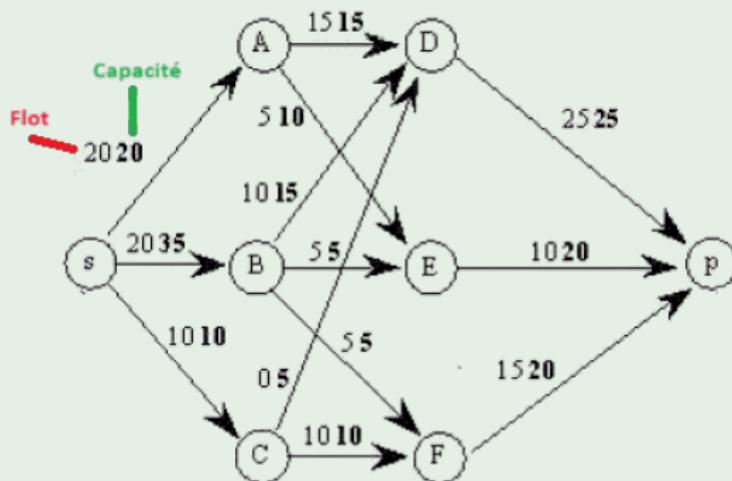
On a les flots des arcs qui ont les 2 extrémités dans B vont apparaître de chaque côté de l'égalité donc vont se simplifier.



Il reste alors d'un côté de l'égalité la somme des flux des arcs entrants dans B et de l'autre côté de l'égalité, la somme des flux des arcs sortants de B .

Exemple

Le schéma suivant représente un exemple de flot sur un réseau de transport :



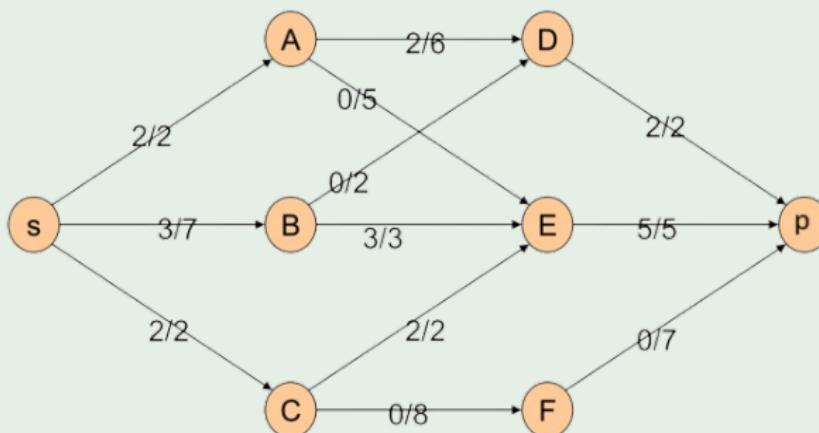
- Pour le sommet B : flot entrant = 20 et flot sortant = $10+5+5 = 20$.
- Si on considère le sous ensemble $X = \{B, C, D, E\}$:
Flot entrant = $15+5+20+10 = 50$ et flot sortant = $25+10+5+10 = 50$

Flot Complet

Définition (Flot Complet)

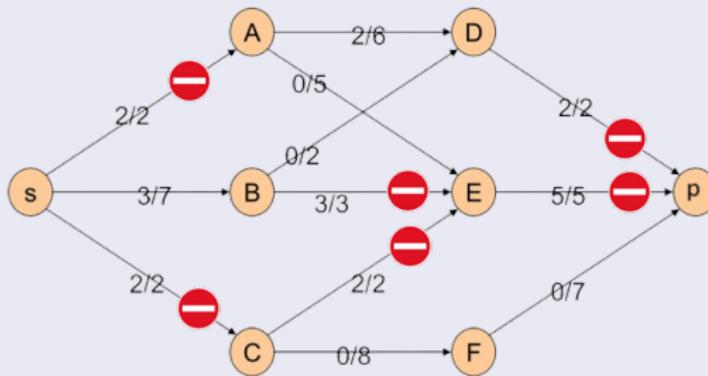
On dit qu'un **flot est complet** si tout chemin du réseau de transport allant de s à p contient au moins un **arc saturé**.

Exemple



Amélioration du flot

On considère le flot suivant :



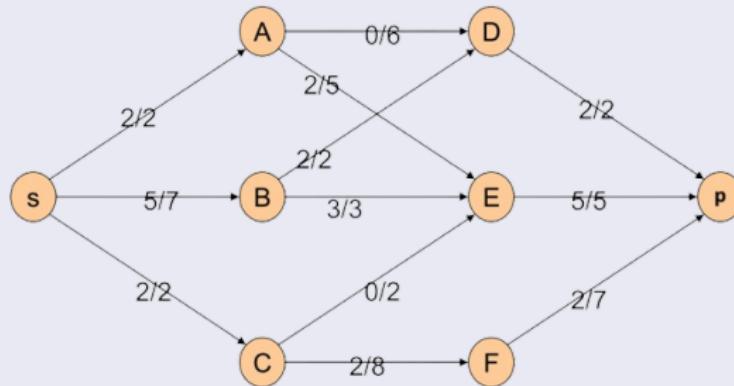
On constate que la valeur de ce flot est $V(\phi) = 7$.

Question

Peut-on améliorer la valeur de ce flot ?

Flot maximal

Sur le réseau de transport précédent, on peut obtenir un flot meilleur :



On constate que la valeur de ce flot est $V(\phi) = 9$.

Question

Est-ce que ce flot est maximal ?

Définition (Flot maximal)

Soit $G = [S, \mathcal{A}, C]$ un réseau de transport avec capacités.

Le flot de valeur maximale ϕ^* qui est, parmi l'ensemble des flots réalisables, celui qui maximise la quantité $V(\phi)$ est appelé **flot maximal** :

$$(\forall \phi \text{ un flot réalisable}) : V(\phi^*) \geq V(\phi)$$

Remarques

- Un flot maximal n'est pas nécessairement unique.
- Le problème de flot maximal peut-être modélisé par programmation linéaire.
- Mais les caractéristiques de celui-ci permettent d'élaborer des algorithmes de résolutions exactes plus efficaces dans la recherche du flot maximal : **algorithme de Ford-Fulkerson**.

Flot maximal - Algorithme de Ford-Fulkerson

Algorithme de Ford-Fulkerson

Soit $G = [S, \mathcal{A}, C]$ un réseau de transport avec capacités.

On commence par un flot réalisable initial et on suit les étapes suivantes :

- ① Marquer l'entrée **s** par +.
- ② Soit i un sommet marqué non examiné.

Étudier tous les successeurs j de i :

Marquer j par **+i** : s'il est non marqué et si $\phi_{ij} < c_{ij}$.

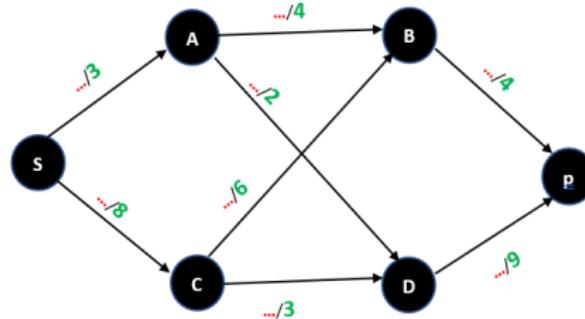
Étudier tous les prédecesseurs k de i :

Marquer k par **-i** : s'il est non marqué et si $\phi_{ki} > 0$.

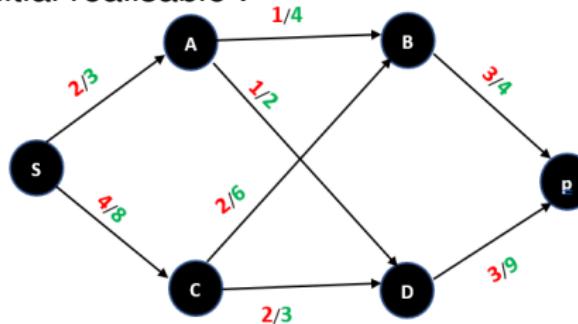
- ③ **Si** le sommet **p** est marqué, aller en (4)
- Sinon** : **S**'il reste des sommets marqués non examinés, aller en (2).
- Sinon** le flot est maximal, **FIN**.
- ④ Améliorer le flot à l'aide de la chaîne améliorante ayant permis de marquer **p** et effacer les marques, sauf celle de **s**, et aller en (1).

Exemple : Algorithme de Ford-Fulkerson

On considère le réseau de transport suivant :



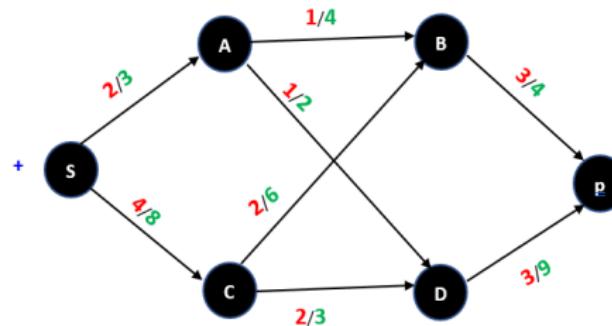
On cherche un flot initial réalisable :



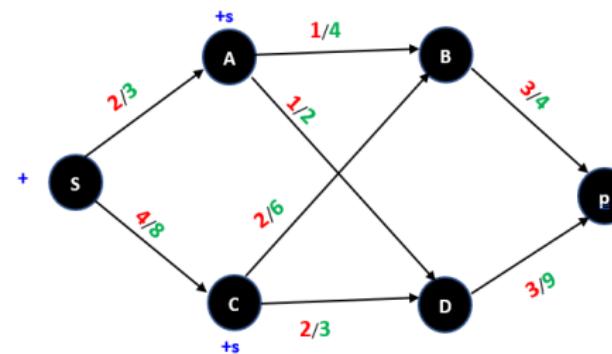
On constate que la valeur de ce flot est $V(\phi) = 6$.

Exemple : Algorithme de Ford-Fulkerson

On commence le marquage par le sommet **S** :

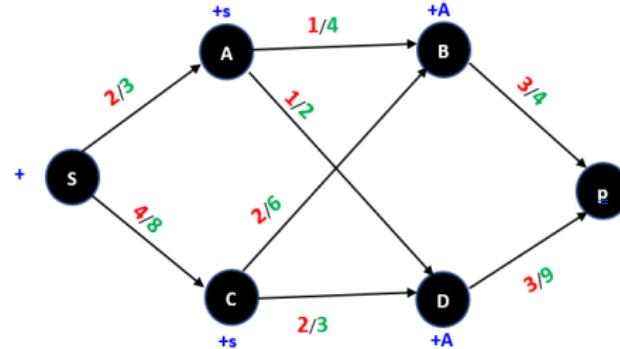


On examine le sommet **S** :

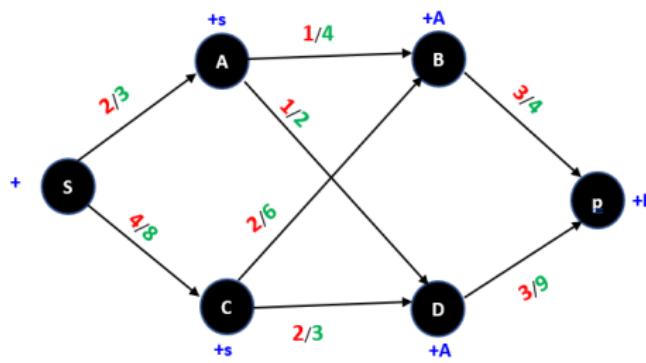


Exemple : Algorithme de Ford-Fulkerson

On examine le sommet A puis le sommet C :



On examine le sommet B puis le sommet D :

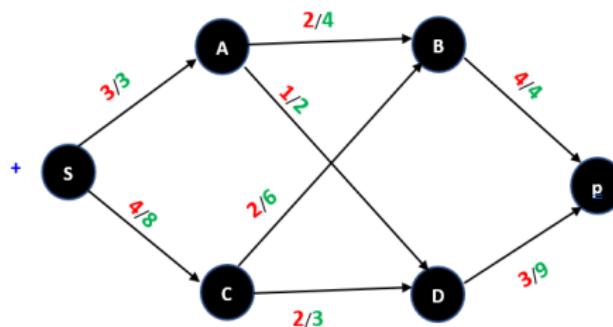


Exemple : Algorithme de Ford-Fulkerson

- Puisque le sommet **p** a été marqué, on va améliorer le flot à l'aide de la chaîne améliorante ayant permis de marquer **p**.



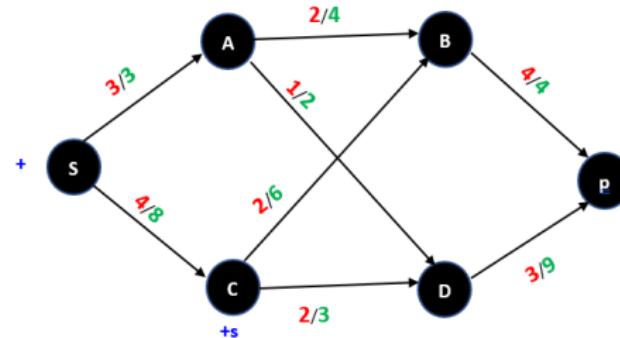
- La chaîne améliorante est
- La valeur possible d'amélioration est $\alpha = \min\{3 - 2; 4 - 1; 4 - 3\} = 1$.
- On obtient donc le flux amélioré suivant :



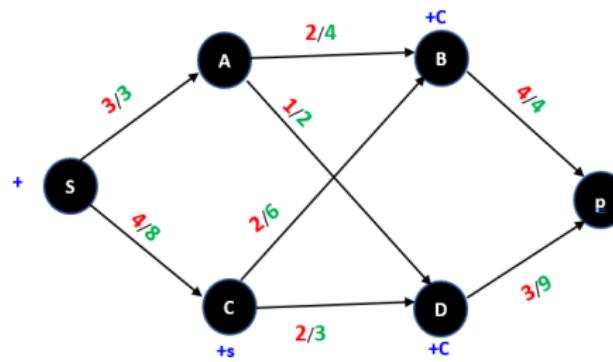
- La valeur de ce nouveau flot est $V(\phi) = 7$.

Exemple : Algorithme de Ford-Fulkerson

On examine le sommet **S** :

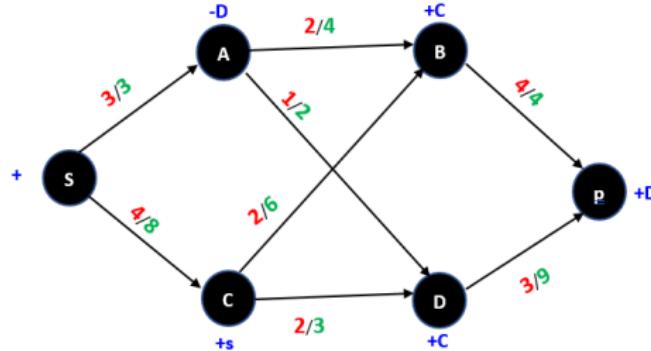


On examine le sommet **C** :



Exemple : Algorithme de Ford-Fulkerson

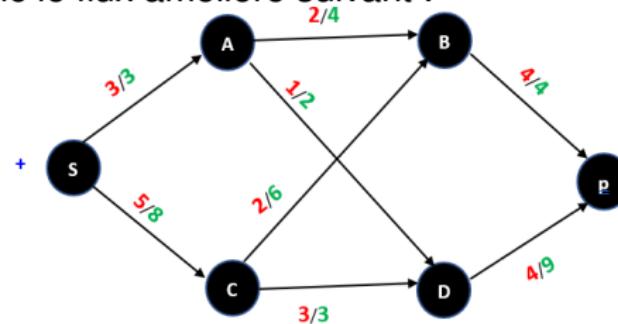
On examine le sommet **D** puis le sommet **B** :



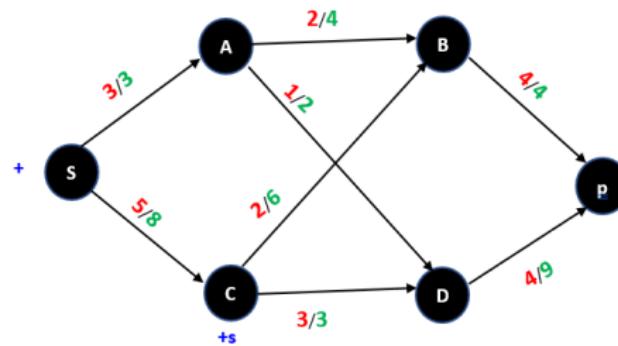
- Puisque le sommet **p** a été marqué, on va améliorer le flot à l'aide de la chaîne améliorante ayant permis de marquer **p**.
- La chaîne améliorante est
- La valeur possible d'amélioration est $\alpha = \min\{8 - 4; 3 - 2; 9 - 3\} = 1$.

Exemple : Algorithme de Ford-Fulkerson

- On obtient donc le flux amélioré suivant :

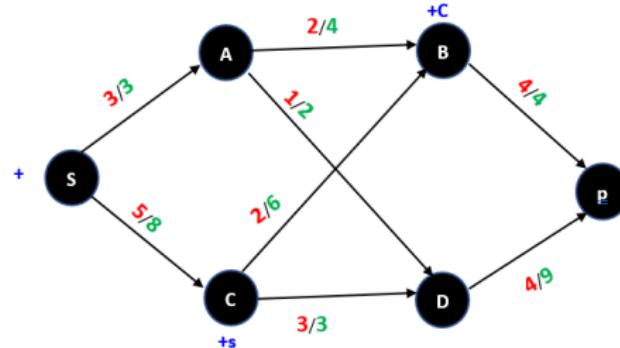


- La valeur de ce nouveau flot est $V(\phi) = 8$.
- On examine le sommet **S** :

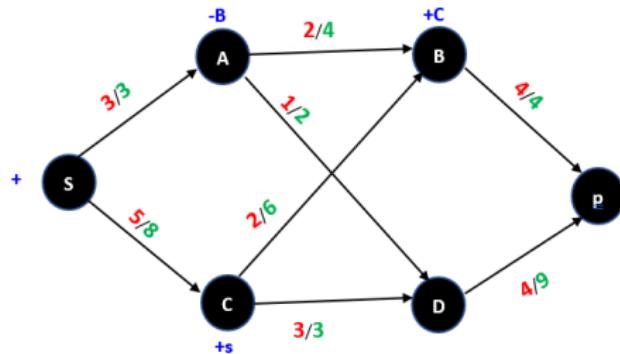


Exemple : Algorithme de Ford-Fulkerson

On examine le sommet **C** :

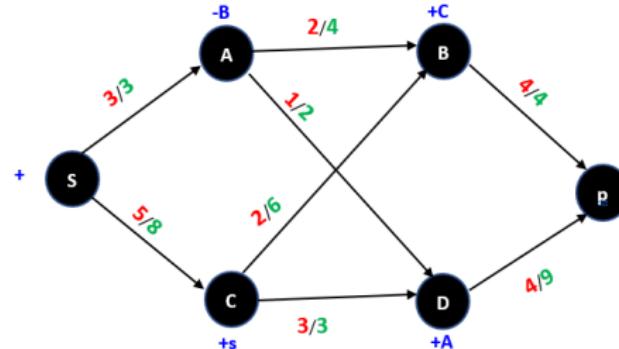


On examine le sommet **B** :

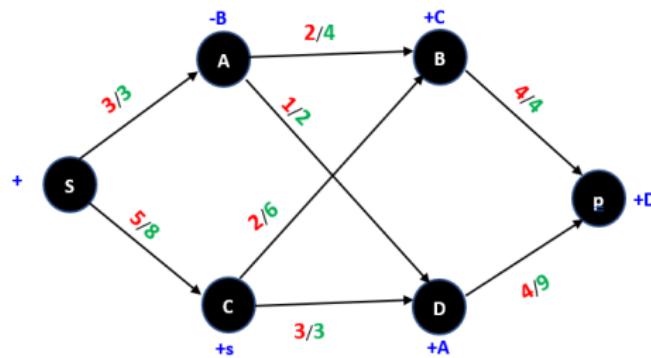


Exemple : Algorithme de Ford-Fulkerson

On examine le sommet A :



On examine le sommet D :



Exemple : Algorithme de Ford-Fulkerson

- Puisque le sommet **p** a été marqué, on va améliorer le flot à l'aide de la chaîne améliorante ayant permis de marquer **p**.

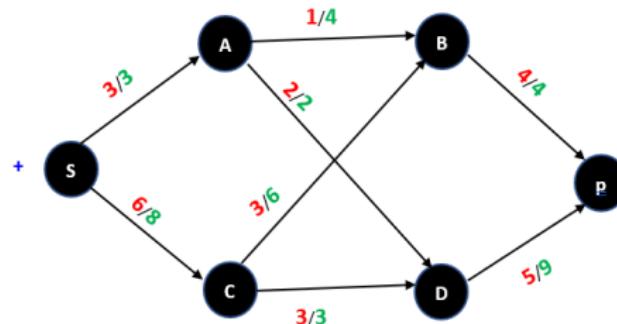
- La chaîne améliorante est



- La valeur possible d'amélioration est

$$\alpha = \min\{8 - 5; 6 - 2; 2 - 1; 9 - 4\} = 1.$$

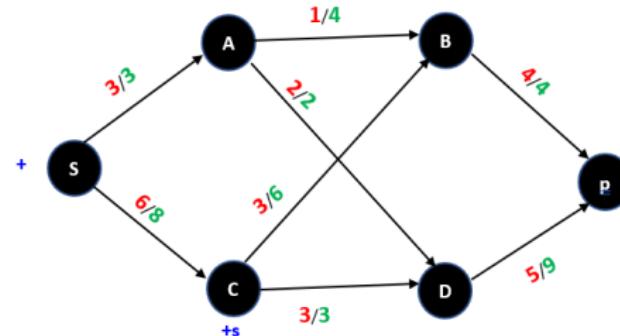
- On obtient donc le flux amélioré suivant :



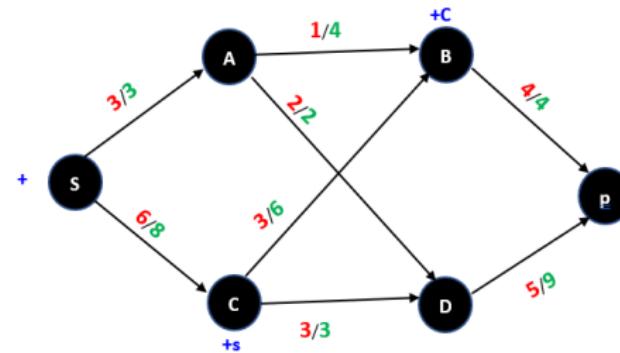
- La valeur de ce nouveau flot est $V(\phi) = 9$.

Exemple : Algorithme de Ford-Fulkerson

On examine le sommet **S** :

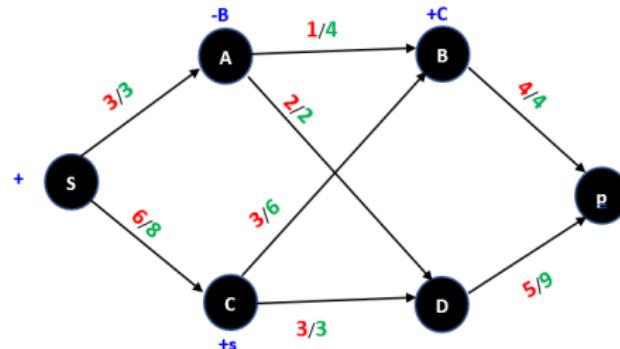


On examine le sommet **C** :



Exemple : Algorithme de Ford-Fulkerson

On examine le sommet **B** :



On examine le sommet **A**, mais on ne peut pas marquer un autre sommet. Puisqu'il ne reste aucun sommet marqué non examiné et on n'arrive pas à marquer le sommet puis **p** alors :

- Le dernier flot obtenu est **maximal**
- La valeur maximale du flot est $V(\phi^*) = 9$.

- Pourquoi l'algorithme de Ford-Fulkerson permet d'obtenir une solution exacte au problème de flot maximal ?
- Quelques définitions et propriétés permettent de justifier ceci

Définition (Coupe)

Une **coupe** K dans le réseau de transport $G = [S, \mathcal{A}, C]$, est une partition de S en deux sous-ensembles U et \bar{U} tels que $s \in U$ et $p \in \bar{U}$.

La **capacité d'une coupe** est définie par :

$$c(K) = \sum_{\substack{(i,j) \in \mathcal{A} \\ i \in U \text{ et } j \in \bar{U}}} c_{ij}$$

Donc $c(K)$ désigne la somme des capacités des arcs sortants de U et allant vers \bar{U} .

Proposition (Coupe)

Pour toute coupe $K = \{U, \bar{U}\}$ la valeur de tout flot ϕ est la valeur du flot circulant entre U et \bar{U} est $\phi_{U \rightarrow \bar{U}} - \phi_{\bar{U} \rightarrow U}$, i.e :

$$V(\phi) = \sum_{i \in U, j \in \bar{U}} \phi_{ij} - \sum_{j \in \bar{U}, i \in U} \phi_{ji}$$

Preuve : On a $V(\phi) = \sum_{i \in U} \left(\sum_j \phi_{ij} - \sum_j \phi_{ji} \right)$

$$V(\phi) = \sum_{i \in U} \left(\sum_{j \in U} \phi_{ij} + \sum_{j \in \bar{U}} \phi_{ij} - \sum_{j \in U} \phi_{ji} - \sum_{j \in \bar{U}} \phi_{ji} \right)$$

En simplifiant : $V(\phi) = \sum_{i \in U, j \in \bar{U}} \phi_{ij} - \sum_{j \in \bar{U}, i \in U} \phi_{ji}$

Proposition (Coupe)

Pour toute coupe $K = \{U, \bar{U}\}$ et un flot réalisable ϕ , on a :

$$c(K) \geq V(\phi)$$

Preuve :

Pour tout arc $(i, j) \in \mathcal{A}$, nous avons : $0 \leq \phi_{ij} \leq c_{ij}$ d'où :

$$\begin{aligned} c(K) &= \sum_{i \in U, j \in \bar{U}} c_{ij} \\ &\geq \sum_{i \in U, j \in \bar{U}} \phi_{ij} \\ &\geq \sum_{i \in U, j \in \bar{U}} \phi_{ij} - \sum_{j \in \bar{U}, i \in U} \phi_{ji} \\ &\geq V(\phi) \end{aligned}$$

Bien fondé de l'algorithme de Ford-Fulkerson

Le résultat théorique sur lequel repose la recherche d'un flot de valeur maximale dans le théorème de Ford-Fulkerson :

Théorème : Coupe minimale - Flot maximal

Soit $G = [S, \mathcal{A}, C]$ un réseau de transport avec capacités.

On note par \mathbb{F} la famille des flots réalisables de G et par \mathbb{K} la famille des coupes dans G .

On considère les deux quantités suivantes :

$$V_M = V(\phi^*) = \max_{\phi \in \mathbb{F}} \{V(\phi)\}$$

$$c_m = \min_{K \in \mathbb{K}} \{c(K)\}$$

Alors

$$V_M = c_m.$$

Bien fondé de l'algorithme de Ford-Fulkerson

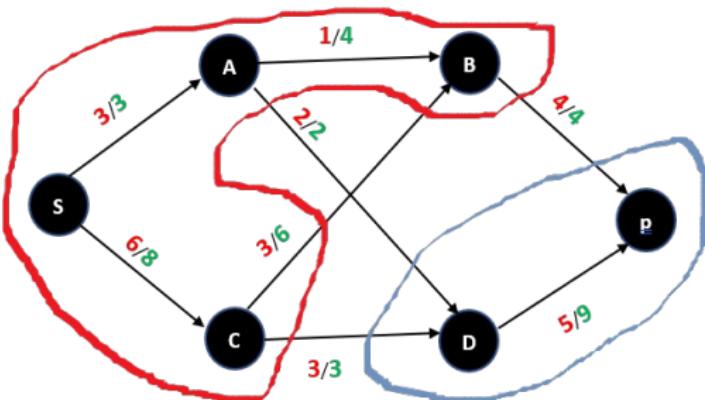
On reprend l'exemple précédent dans la dernière itération :
On considère la coupe

On considère la coupe

$K = \{U, \overline{U}\}$ avec

$$U = \{s;A;B;C\}$$

et $\overline{U} = \{D; p\}$.



- Tout $(i, j) \in \mathcal{A}$ tel que $i \in U$ et $j \in \overline{U}$ est saturé.
 - Tout $(j, i) \in \mathcal{A}$ tel que $j \in \overline{U}$ et $i \in U$ vérifie $\phi_{ji} = 0$.
 - Alors

$$c(K) = \sum_{i \in U, j \in \bar{U}} c_{ij} = \sum_{i \in U, j \in \bar{U}} \phi_{ij} - \sum_{j \in \bar{U}, i \in U} \phi_{ji} = V(\phi^*)$$

$$\min \quad \quad \quad \max \quad \quad \quad 0 \quad \quad \quad \max$$

Optimisation combinatoire

- L'optimisation combinatoire constitue une branche très importante de la recherche opérationnelle.
- Elle permet de résoudre plusieurs problèmes réels et de réaliser des gains considérables et d'assurer des optimisations dans les problème de la recherche de chemins et de cycles optimaux ainsi que l'optimisation de flots dans les réseaux.
- Il reste très important d'aborder :
 - **Les problèmes de couverture et d'affectation** : Problème de couverture minimale, Problème d'affectation, Problème de couplage maximal.
 - **Les problèmes de partitionnement et de découpe** : Partitionnement de graphes et Problème du sac à dos.
 - **Les problème de tournées et de chemins** : Problème du voyageur de commerce, Problème de tournées de véhicules.