UNIVERSITÉ D'ABOMEY-CALAVI (UAC)

- - - - - - -

ÉCOLE NATIONALE D'ÉCONOMIE APPLIQUÉE ET DE MANAGEMENT (ENEAM)

- - - - - - -

Licence 2 en Informatique de Gestion

- - - - - - -

UAC ENEAM

COURS DE RECHERCHE OPERATIONNELLE

Enseignant : S. Emeric Chris Gbodo, ISE

2024-2025

CHAPITRE 1

INTRODUCTION GÉNÉRALE

1.1 Présentation de la recherche opérationnelle

1.1.1 Définition

La recherche opérationnelle (RO) est une discipline des mathématiques appliquées qui se concentre sur l'utilisation optimale des ressources dans divers domaines, tels que l'industrie, le secteur public, l'économie, la finance, et plus récemment, dans les systèmes de santé, l'éducation et la gestion de problèmes environnementaux. La RO vise à offrir des outils scientifiques pour la résolution de problèmes organisationnels complexes et pour l'aide à la prise de décision, en permettant aux décideurs d'analyser les situations de manière rationnelle et structurée. Depuis une dizaine d'années, les applications de la recherche opérationnelle se sont élargies, intégrant des secteurs variés comme le marketing, la planification stratégique et les politiques publiques. Dans tous ces domaines, l'objectif est de fournir des méthodes quantitatives pour maximiser un profit, une performance, ou une efficacité, ou bien pour minimiser les coûts et les dépenses associés à la réalisation des objectifs organisationnels.

La recherche opérationnelle repose sur des méthodes d'analyse scientifique, des modèles mathématiques et des outils informatiques pour étudier les phénomènes organisationnels. Elle permet de modéliser des processus et des systèmes complexes en vue de trouver le meilleur résultat possible face aux contraintes existantes. En fournissant des modèles conceptuels de simulation et d'optimisation, la RO aide les décideurs à comprendre les implications de leurs choix, à anticiper les résultats et à arbitrer entre plusieurs options stratégiques.

1.1.2 Origine de la recherche opérationnelle

La recherche opérationnelle trouve ses racines au début du XXe siècle avec les premiers travaux sur la gestion des stocks, notamment la formule du lot économique, ou formule de Wilson, proposée par Harris en 1913 pour optimiser les approvisionnements. Cependant, ce n'est qu'à l'époque de la Seconde Guerre mondiale que la recherche opérationnelle se structure véritablement en tant que discipline, notamment grâce aux efforts d'éminents mathématiciens comme John von Neumann, George Dantzig et Patrick Blackett. En 1940, Patrick Blackett est appelé par l'état-major britannique pour diriger la première équipe de recherche opérationnelle, chargée de résoudre des problèmes d'optimisation cruciaux pour l'armée, comme l'implantation optimale de radars et la gestion des convois de ravitaillement. L'impact décisif de ces travaux dans l'efficacité militaire a conduit Blackett à recevoir le prix Nobel de physique, en particulier pour sa contribution à l'optimisation de l'emplacement des radars.

Le terme *opérationnelle* fait ici référence aux applications militaires, car la discipline visait à optimiser des opérations stratégiques. Cette dénomination est toutefois restée, même si les applications de la RO se sont par la suite étendues bien au-delà du domaine militaire, pour signifier plutôt une optimisation des *actions de manière effective dans divers domaines*.

Dans l'après-guerre, la recherche opérationnelle devient une méthodologie clé, fondée sur la modélisation et l'optimisation, permettant de formaliser des problèmes complexes pour trouver des solutions optimales. Elle entre progressivement dans les entreprises dans les années 1950, attirant un intérêt croissant dans les universités et les grandes écoles, où elle commence à être enseignée.

Cependant, dans les années 1970, la RO connaît un ralentissement, dû en partie à des attentes excessives et aux limites des moyens informatiques de l'époque, qui peinent à suivre les exigences des modèles mathématiques sophistiqués de la discipline. Il faudra attendre les années 1990 pour assister à un retour en force de la recherche opérationnelle, soutenue cette fois par les avancées technologiques dans le domaine de l'informatique. Ces nouvelles capacités permettent de traiter de grandes quantités de données et d'exécuter des calculs complexes, redonnant ainsi une impulsion significative à la discipline. Cette renaissance s'accompagne de la prolifération de logiciels d'optimisation et d'analyses commerciales, ainsi que de la création de nombreuses entreprises de conseil spécialisées.

Aujourd'hui, la recherche opérationnelle est une science centrale pour la prise de décision stratégique et opérationnelle dans divers domaines : industrie, transport, finance,

santé, environnement, et plus encore. Ses méthodologies, basées sur des outils de modélisation, de simulation et d'optimisation, continuent d'évoluer avec les avancées en intelligence artificielle et en analyse de données. Elle reste une discipline fondamentale pour l'optimisation des ressources et l'amélioration de l'efficacité dans des environnements complexes, consolidant son rôle d'outil essentiel pour la prise de décision moderne.

1.1.3 Les précurseurs de la recherche opérationnelle

Si l'on recherche des précurseurs de la recherche opérationnelle (RO), des figures comme Alcuin et Euler viennent à l'esprit, bien qu'ils aient abordé des problèmes similaires sans visée d'application concrète.

Alcuin et le problème du loup, de la chèvre et du chou

Alcuin, un moine irlandais chargé par Charlemagne de fonder l'école palatine, inventa le célèbre problème du loup, de la chèvre et du chou, une énigme classique de gestion de contraintes. Voici le contexte :

Un homme devait traverser une rivière avec un loup, une chèvre et un panier de choux. Cependant, il ne disposait que d'une barque capable de transporter au plus deux éléments à la fois. Il devait donc trouver une manière de faire passer tout le monde de l'autre côté sans laisser le loup avec la chèvre (qu'il pourrait dévorer) ni la chèvre avec les choux (qu'elle pourrait manger).

Ce problème met en lumière une approche d'optimisation sous contrainte, où l'on doit établir un plan d'action pour atteindre un objectif (traverser sans perte). Bien que conçu comme une énigme, ce type de raisonnement sur les contraintes et les ordres de priorité préfigure la réflexion en recherche opérationnelle.

Euler et les ponts de Königsberg

Le mathématicien allemand Leonhard Euler est également un pionnier indirect de la recherche opérationnelle. En 1736, les citoyens de Königsberg (aujourd'hui Kaliningrad) lui posèrent une question intrigante : est-il possible de parcourir tous les ponts de la ville en passant une fois et une seule fois par chacun des sept ponts ? Euler démontra que cela était impossible, développant ainsi les prémices de la théorie des graphes.

Ce type de question se retrouve aujourd'hui dans des problèmes de tournée, comme ceux du facteur ou de la collecte des déchets, où l'on cherche des parcours optimaux couvrant toutes les rues ou points d'un réseau. La solution d'Euler, fondée sur une modélisation graphique innovante, est l'une des premières à poser les bases de ce qui deviendra les chemins eulériens et les principes de la théorie des graphes, un outil essentiel en

recherche opérationnelle pour la modélisation et la résolution de problèmes de parcours.

Monge et le problème des déblais et des remblais

Le premier problème de recherche opérationnelle conçu dans un but pratique est sans doute celui du "problème des déblais et remblais", étudié en 1781 par le mathématicien français Gaspard Monge. Ce problème visait à optimiser le transport de sable pour combler des trous en minimisant la distance totale parcourue par le matériau. Monge proposa une solution basée sur la recherche d'un chemin optimal (ou géodésique) dans un espace continu. Cependant, sa solution n'était pas correcte, car elle ne prenait pas en compte les contraintes structurelles et les optimisations spécifiques de transport.

C'est dans les années 1940 que les méthodes de programmation linéaire et la théorie des flots ont permis de formaliser une solution optimale au problème de Monge. Ces outils ont montré que, pour minimiser les déplacements et les coûts de transport, une modélisation précise des flux de ressources est nécessaire. Ce type de problème est aujourd'hui fondamental en logistique et en optimisation des chaînes d'approvisionnement.

Ainsi, les travaux d'Alcuin, Euler et Monge, bien qu'initialement éloignés de la recherche opérationnelle moderne, ont introduit des approches de modélisation et de résolution de problèmes qui font partie intégrante de cette discipline. Chacun, à sa manière, a apporté des idées centrales autour de la gestion des contraintes, de la recherche d'optimisation et de la théorie des graphes, qui constituent aujourd'hui le socle de nombreuses applications pratiques en recherche opérationnelle.

1.1.4 Intérêts de la recherche opérationnelle en informatique de gestion

En informatique de gestion, la recherche opérationnelle est un atout précieux pour rationaliser les processus, réduire les coûts et optimiser l'efficacité des systèmes. Elle permet aux entreprises d'adopter une approche quantitative de la gestion, soutenue par des modèles et des algorithmes robustes, afin de répondre aux exigences d'un environnement concurrentiel complexe et en constante évolution. Grâce aux avancées en informatique et en intelligence artificielle, les outils et méthodes de la RO se renforcent, permettant aux entreprises de tirer encore davantage parti de ces techniques pour améliorer leur performance globale.

1.2 Applications de la recherche opérationnelle

1.2.1 Ordonnancement et gestion des projets

De nombreux travaux traitent de l'ordonnancement et de la gestion de projets, mais aussi de logistique (tournées de véhicules, conditionnement...), de planification, et de problèmes d'emploi du temps.

- ☞ La gestion de projet est une démarche visant à organiser de bout en bout le bon déroulement d'un projet. Lorsque la gestion de projet porte sur un ensemble de projets concourant à un même objectif, on parle de gestion de programme.
- ☞ La théorie de l'ordonnancement est une branche de la recherche opérationnelle qui s'intéresse au calcul de dates d'exécution optimales de tâches. Pour cela, il est très souvent nécessaire d'affecter en même temps les ressources nécessaires à l'exécution de ces tâches. Un problème d'ordonnancement peut être considéré comme un sous-problème de planification dans lequel il s'agit de décider de l'exécution opérationnelle des tâches planifiées. Les méthodes couramment utilisées pour ordonnancer un projet sont les méthodes MPM et PERT.

Exemple

La société OFMAS Bénin a reçu la maitrise d'œuvre de la construction de la route de pêche ralliant la ville de Cotonou à celle de Ouidah. Le tableau des antériorités des tâches est le suivant :

TABLE 1.1: Tableau des tâches et antériorités

Codes	Tâches	Durée (en jours)	Antériorités
A	Etude de faisabilité et d'impact environnemental	30	-
B	Acquisition des terrains	45	A
C	Aménagement des accès temporaires	20	B
D	Conception du plan de route	25	A
E	Déblaiement du terrain	15	B,C
F	Installation des canalisations de drainage	40	E
G	Construction des fondations de la route	60	E,D
H	Application de la première couche d'asphalte	30	F,G
I	Installation de la signalisation temporaire	10	C
J	Application de la couche de finition	20	H

K	Installation de la signalisation définitive	15	J
L	Inspection finale et validation	10	K

1.2.2 Plus court chemin

En théorie des graphes, l'algorithme de Dijkstra sert à résoudre le problème du plus court chemin. Il permet, par exemple, de déterminer le plus court chemin pour se rendre d'une ville à une autre, connaissant le réseau routier d'une région. Il s'applique à un graphe connexe dont le poids lié aux arêtes est un réel positif. L'algorithme porte le nom de son inventeur, l'informaticien néerlandais Edsger Dijkstra et a été publié en 1959.

Exemple

Vous avez un réseau routier reliant plusieurs grandes villes du Bénin. Le ministère des Transports souhaite trouver le trajet le plus court entre Cotonou et Parakou pour faciliter les déplacements et les échanges économiques. Chaque route entre les villes a une distance spécifique en kilomètres. Les données recueillies sont les suivantes :

— Cotonou - Abomey : 100 km

— Cotonou - Bohicon : 120 km

— Abomey - Dassa-Zoumè : 85 km

— Bohicon - Dassa-Zoumè : 90 km

— Bohicon - Savalou : 70 km

— Dassa-Zoumè - Parakou : 160 km

— Savalou - Parakou : 140 km

— Bohicon - Porto-Novo : 30 km

— Porto-Novo - Cotonou : 40 km

1.2.3 La programmation linéaire

La programmation linéaire est une méthode centrale en recherche opérationnelle, utilisée pour optimiser un objectif linéaire, tel que maximiser le profit ou minimiser les coûts, sous un ensemble de contraintes également linéaires.

Exemple

Une usine fabrique deux produits P1 et P2 à l'aide de trois matières premières M1, M2 et M3 dont on dispose en quantité limitée. On se pose le problème de l'utilisation

optimale de ce stock de matières premières, c'est-à-dire la détermination d'un schéma, d'un programme de fabrication tel que :

— les contraintes de ressources en matières premières soient respectées,

— le bénéfice réalisé par la vente de la production soit maximum.

La disponibilité en matières premières est de 18 unités de M_1, 8 unités de M_2 et 14 unités de M_3

Les caractéristiques de fabrication sont données dans le tableau suivant :

TABLE 1.2 – Caractéristiques de fabrication

	M_1	M_2	M_3
P_1	1	1	2
P_2	3	1	1

Le bénéfice de l'usine peut être exprimé sous la forme : $Z(x_1, x_2) = c_1x_1 + c_2x_2$

1.3 Formalisation générale d'un modèle en recherche opérationnelle

1.3.1 Concepts clés : Modélisation et optimisation

La modélisation mathématique consiste à représenter une situation réelle sous une forme mathématique pour lui appliquer des méthodes, techniques et théories mathématiques adaptées. Cette étape est fondamentale en recherche opérationnelle, car elle permet de transformer un problème concret en un modèle abstrait sur lequel on peut travailler. Ensuite, les résultats mathématiques obtenus sont traduits en prédictions ou en actions applicables dans le monde réel.

Bien que les problèmes d'organisation en entreprise ne soient pas mathématiques par nature, les outils mathématiques peuvent permettre d'y apporter des solutions. Il est donc nécessaire de reformuler ces problèmes dans un cadre mathématique, où les techniques de recherche opérationnelle pourront être appliquées. Cette formulation est le modèle du problème, et le processus de modélisation devient une étape cruciale pour toute résolution. En effet, la manière de modéliser un problème influence grandement sa complexité et les solutions possibles. Pour un même problème, il peut exister plusieurs modélisations, certaines rendant le problème insoluble, tandis que d'autres le simplifient.

Tous les éléments d'un problème n'ont pas forcément besoin d'être modélisés. Par exemple, dans le cadre de la planification d'une tournée de livraison, des aspects comme la couleur du véhicule ne sont pas pertinents. En revanche, des caractéristiques comme le type de produit transporté ou le statut du conducteur peuvent l'être, en fonction des objectifs spécifiques de l'optimisation. La phase de modélisation nécessite souvent une interaction étroite avec le commanditaire, pour clarifier les objectifs et sélectionner les éléments pertinents à modéliser.

L'une des principales difficultés réside dans la sélection des éléments à inclure dans le modèle. Il s'agit de trouver le juste équilibre entre un modèle simple, qui sera plus facile à résoudre, et un modèle plus réaliste, mais souvent plus complexe. Dans de nombreux cas, il est recommandé de commencer par un modèle simple qui capture l'essentiel du problème et permet de proposer des solutions rapides à mettre en œuvre. Cette approche favorise la construction d'une intuition pour le problème et permet de vérifier si des solutions simplifiées sont suffisantes.

En recherche opérationnelle, les questions essentielles à se poser pour structurer un modèle sont les suivantes :

- ☞ **Quelles sont les variables de décision ?** Ce sont les éléments que l'on peut ajuster pour explorer différentes solutions.
- ☞ **Quelles sont les contraintes ?** Ces limites imposées aux variables de décision définissent les valeurs acceptables pour chaque option envisagée.
- ☞ **Quelles sont les objectifs ?** Il s'agit de la quantité que l'on cherche à maximiser ou minimiser. Il est important de noter que, dans une optimisation stricte, on ne peut cibler qu'un seul critère à la fois. Par exemple, il est impossible d'optimiser simultanément la distance et le temps de trajet d'un itinéraire sans compromis, car les solutions optimales pour chaque critère peuvent différer. L'optimisation multiobjectif permet de hiérarchiser les critères ou de les pondérer pour obtenir une solution de compromis (comme une solution dite Pareto-optimale).

Une fois le modèle défini, le chercheur doit concevoir un algorithme qui respecte l'objectif fixé. Pour un même modèle, différents algorithmes peuvent être envisagés, chacun ayant ses avantages en termes de qualité de solution, rapidité d'exécution, et facilité d'implémentation. Selon les besoins, une solution rapide, mais approximative, peut être acceptable, tandis que dans d'autres cas, une solution optimale justifie un temps de calcul plus long.

1.3.2 Étapes d'une méthodologie de recherche opérationnelle

- ☞ Objectifs, contraintes, variables de décision
- ☞ Modélisation
- ☞ Proposition d'un algorithme et validité de l'algorithme
- ☞ Implémentation et évaluation de la solution
- ☞ Déploiement de la solution

CHAPITRE 2

THÉORIE DES GRAPHES ET GESTION DES PROJETS

La théorie des graphes a pris naissance au XVIIIe siècle avec les travaux pionniers de Leonhard Euler, qui a abordé des questions inspirées de problèmes réels, notamment le célèbre problème des ponts de Königsberg. Ce problème se posait ainsi : les habitants de la ville de Königsberg, traversée par plusieurs bras de rivière et reliée par des ponts, se demandaient s'il était possible de parcourir la ville en passant par chaque pont une seule fois et en revenant au point de départ. Euler a démontré qu'une telle traversée était impossible, posant ainsi les premières bases de ce qui deviendrait la théorie des graphes.

Depuis, cette discipline a évolué pour devenir une branche essentielle des mathématiques, grâce aux contributions de mathématiciens comme König, Menger, Cayley, ainsi que Berge et Erdös, qui ont élargi et formalisé ses concepts. La théorie des graphes permet de représenter des structures complexes sous forme de sommets (ou nœuds) et arcs (ou arêtes) reliant ces sommets. Elle fournit ainsi un langage universel pour modéliser des systèmes variés : réseaux de communication, systèmes de transport, relations biologiques entre espèces, réseaux de distribution électrique, et bien plus encore.

En pratique, les graphes permettent d'étudier les relations et interactions au sein de systèmes complexes en simplifiant les entités étudiées en points connectés entre eux. Ils sont ainsi devenus un puissant outil de réflexion, facilitant l'analyse et la résolution de problèmes en ramenant des questions complexes à l'étude de connexions entre éléments.

Aujourd'hui, la théorie des graphes est étroitement liée à l'informatique, où elle est appliquée pour résoudre des problèmes d'optimisation, de navigation de réseaux, ou encore d'analyse de données. Cette association s'explique par le caractère hautement algorithmique de la théorie des graphes, qui continue de susciter un intérêt actif pour la recherche

et les applications, que ce soit dans le développement d'algorithmes pour la recherche de chemins optimaux ou la détection de motifs dans des réseaux sociaux.

2.1 Graphes non orientés

2.1.1 Définitions

Un graphe fini $G = (V, E)$ est défini par un ensemble fini $V = \{v_1, v_2, \ldots, v_n\}$ dont les éléments sont appelés sommets, et par l'ensemble fini $E = \{e_1, e_2, \ldots, e_m\}$ dont les éléments sont appelés arêtes.

Une arête e de l'ensemble E est définie par une paire non ordonnée de sommets, appelées les extrémités de e. Si l'arête e relie les sommets a et b, on dira que ces sommets sont **adjacents**, ou **incidents** avec e, ou bien que l'arête e est **incidente** avec les sommets a et b.

On appelle **ordre** d'un graphe le nombre de sommets n de ce graphe.

2.1.2 Représentation graphique

Les graphes tirent leur nom du fait qu'on peut les représenter par des dessins. À chaque sommet de G, on fait correspondre un point distinct du plan et on relie les points correspondant aux extrémités de chaque arête. Il existe donc une infinité de représentations d'un graphe. Les arêtes ne sont pas forcément rectilignes.

Pour un graphe G Dans le plan, lorsque aucune arête ne coupe l'autre, on dit que G est **planaire**. Dans le cas contraire, on dit qu'il est non **non planaire**.

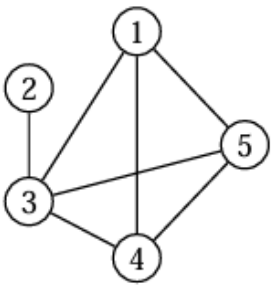


FIGURE 2.1 – Graphe non planaire

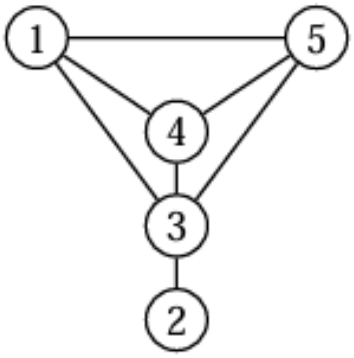


FIGURE 2.2 – Graphe planaire

2.1.3 Quelques types de graphe

Un graphe est **simple** si au plus une arête relie deux sommets et s'il n'y a pas de boucle sur un sommet. On peut avoir des graphes avec une arête qui relie un sommet à lui-même (une boucle), ou plusieurs arêtes reliant les deux mêmes sommets. Ces graphes sont appelés des **multigraphes**.

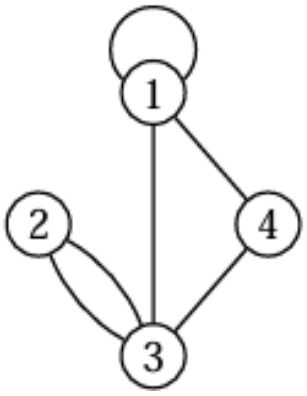


FIGURE 2.3 – Multigraphe

Un graphe est **connexe** s'il est possible, à partir de n'importe quel sommet, de rejoindre tous les autres en suivant les arêtes. Un graphe **non connexe** se décompose en **composantes connexes**. Sur le graphe ci-dessous, les composantes connexes sont $\{1, 2, 3, 4\}$ et $\{5, 6\}$.

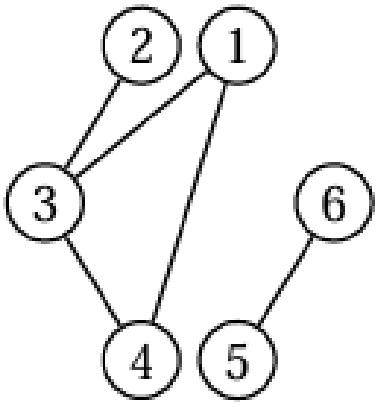


FIGURE 2.4 – Graphe non connexe

$V = \{1, 2, 3, 4, 5, 6\}$; $E = \{\{1, 3\}, \{1, 4\}, \{2, 3\}, \{3, 4\}, \{5, 6\}\}$

Un graphe est **complet** si chaque sommet du graphe est relié directement à tous les autres sommets.

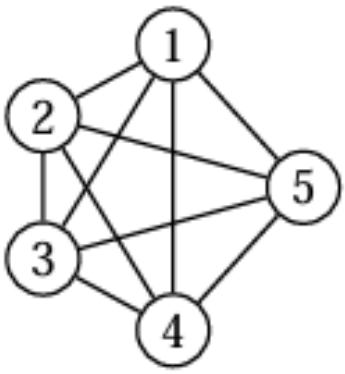


FIGURE 2.5 – Graphe complet

$V = \{1, 2, 3, 4, 5\}$; $E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$

Un graphe est **biparti** si ses sommets peuvent être divisés en deux ensembles X et Y, de sorte que toutes les arêtes du graphe relient un sommet dans X à un sommet dans Y.

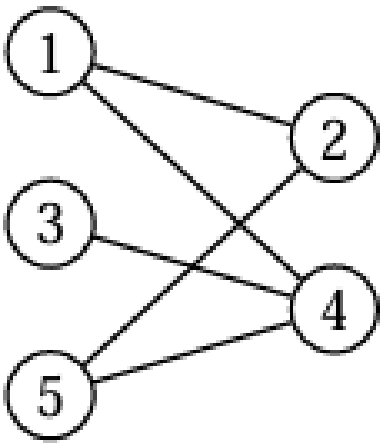


FIGURE 2.6 – Graphe biparti

$V = \{1, 2, 3, 4, 5\}$; $E = \{\{1, 2\}, \{1, 4\}, \{2, 5\}, \{3, 4\}, \{4, 5\}\}$

2.1.4 Graphe partiel et sous-graphe

Soit $G = (V, E)$ un graphe. Le graphe $G' = (V, E')$ est un **graphe partiel de G**, si E' est inclus dans E. Autrement dit, on obtient G' en enlevant une ou plusieurs arêtes au graphe G.

Pour un ensemble de sommets A inclus dans V, le **sous-graphe** de G induit par A est le graphe $G = (A, E(A))$ dont l'ensemble des sommets est A et l'ensemble des arêtes

$E(A)$ est formé de toutes les arêtes de G ayant leurs deux extrémités dans A. Autrement dit, on obtient G' en enlevant un ou plusieurs sommets au graphe G, ainsi que toutes les arêtes incidentes à ces sommets.

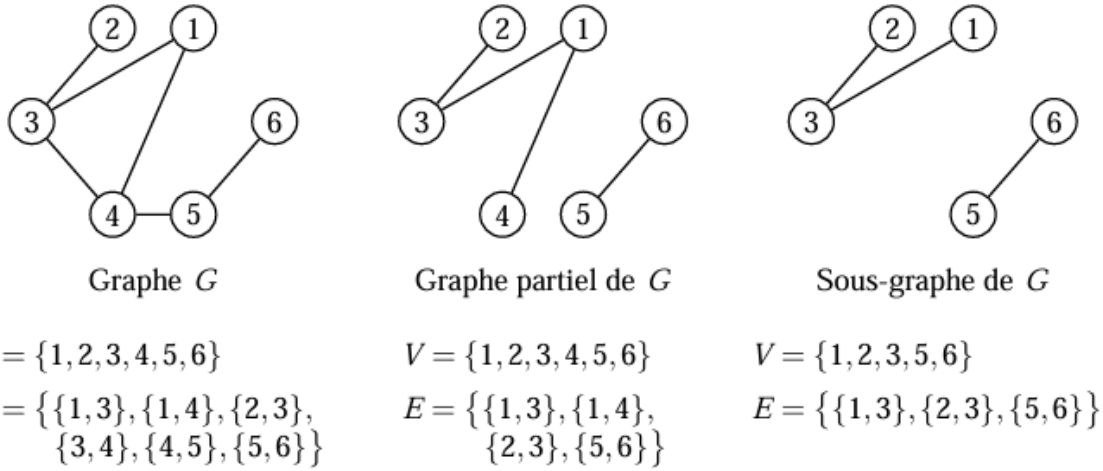


FIGURE 2.7 – Graphe partiel et sous-graphe

Un graphe partiel d'un sous-graphe est un **sous-graphe partiel**.

On appelle **clique**, un sous-graphe complet de G. Dans le graphe G ci-dessus, le sous-graphe $K = (V, E)$ avec $V = \{1, 3, 4\}$ et $E = \{\{1, 3\}, \{1, 4\}, \{3, 4\}\}$ est une clique.

2.1.5 Degrés

On appelle **degré du sommet** v, et on note $d(v)$, le nombre d'arêtes incidentes à ce sommet. Une boucle sur un sommet compte double dans la détermination du degré d'un sommet.

Lemme des poignées de mains : La somme des degrés des sommets d'un graphe est égale à deux fois le nombre d'arêtes.

On appelle **degré d'un graphe**, le degré maximum de tous ses sommets.

Un graphe dont tous les sommets ont le même degré est dit **régulier**. Considérant que k est ce degré commun, on dira que le graphe est *k-régulier*

2.1.6 Chaînes et cycles

Une **chaîne** dans 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

encadrée par ses extrémités.

On dira que la chaîne **relie** le premier sommet de la suite au dernier sommet. La chaîne a pour longueur le nombre d'arêtes de la chaîne. On ne change pas une chaîne en inversant l'ordre des éléments dans la suite correspondante.

On appelle **distance** entre deux sommets la longueur de la plus petite chaine les reliant.

On appelle **diamètre** d'un graphe la plus longue des distances entre deux sommets.

Une chaîne est **élémentaire** si chaque sommet y apparait au plus une fois.

Une chaîne est **simple** si chaque arête y apparait au plus une fois.

Une chaîne dont les sommets de départ et de fin sont les mêmes est appelée chaîne **fermée**.

Une chaîne fermée simple est appelée **cycle**.

2.1.7 Graphes eulériens

On appelle **cycle eulérien** d'un graphe G un cycle passant une et une seule fois par chacune des arêtes de G. Un graphe est dit **eulérien** s'il possède un cycle eulérien.

On appelle **chaîne eulérienne** d'un graphe G une chaîne passant une et une seule fois par chacune des arêtes de G. Un graphe ne possédant que des chaînes eulériennes est **semi-eulérien**.

Plus simplement, on peut dire qu'un graphe est eulérien (ou semi-eulérien) s'il est possible de dessiner le graphe sans lever le crayon et sans passer deux fois sur la même arête.

2.1.8 Graphes hamiltoniens

On appelle **cycle hamiltonien** d'un graphe G un cycle passant une et une seule fois par chacun des sommets de G. Un graphe est dit **hamiltonien** s'il possède un cycle hamiltonien.

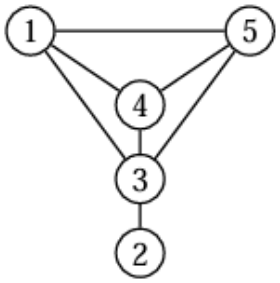
On appelle chaîne **hamiltonienne** d'un graphe G une chaîne passant une et une seule fois par chacun des sommets de G. Un graphe ne possédant que des chaînes hamiltoniennes est **semi-hamiltonien**.

Contrairement aux graphes eulériens, il n'existe pas de caractérisation simple des graphes (semi-)hamiltoniens. On peut énoncer quelques propriétés et conditions suffisantes :

- ☞ un graphe possédant un sommet de degré 1 ne peut pas être hamiltonien ;
- ☞ si un sommet dans un graphe est de degré 2, alors les deux arêtes incidentes à ce sommet doivent faire partie du cycle hamiltonien ;
- ☞ les graphes complets K_n sont hamiltoniens.

2.1.9 Représentations non graphiques d'un graphe

On peut représenter un graphe simple par une **matrice d'adjacences**. Une matrice $(n*m)$ est un tableau de n lignes et m colonnes. (i, j) désigne l'intersection de la ligne i et de la colonne j. Dans une matrice d'adjacences, les lignes et les colonnes représentent les sommets du graphe. La valeur **1** à la position (i, j) signifie que le sommet i est adjacent au sommet j.



$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Cette matrice a plusieurs caractéristiques :

- ☞ Elle est carrée : il y a autant de lignes que de colonnes.
- ☞ Il n'y a que des zéros sur la diagonale allant du coin supérieur gauche au coin inférieur droit. Un **1** sur la diagonale indiquerait une boucle.
- ☞ Elle est symétrique : $m_{ij} = m_{ji}$. On peut dire que la diagonale est un axe de symétrie.
- ☞ Une fois que l'on fixe l'ordre des sommets, il existe une matrice d'adjacences unique pour chaque graphe. Celle-ci n'est pas la matrice d'adjacences d'aucun autre graphe.

On peut aussi représenter un graphe simple en donnant pour chacun de ses sommets la liste des sommets auxquels il est adjacent. Ce sont les **listes d'adjacences**.

- ☞ 1 : 3,4,5
- ☞ 2 : 3
- ☞ 3 : 1,2,4,5
- ☞ 4 : 1,3,5
- ☞ 5 : 1,3,4

2.1.10 Arbres

On appelle **arbre** tout graphe connexe sans cycle. Un graphe sans cycle, mais non connexe est appelé une **forêt**.

Une **feuille** ou **sommet pendant** est un sommet de degré 1.

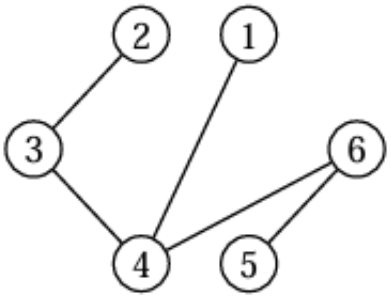


FIGURE 2.8 – Arbre

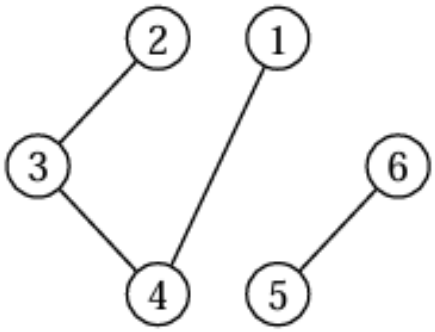


FIGURE 2.9 – Forêt

2.1.11 Une petite révision

1. Qu'est-ce que l'ordre d'un graphe ?

A. Le nombre d'arêtes dans le graphe

B. Le nombre de sommets dans le graphe

C. Le degré du sommet ayant le plus de voisins

D. La longueur du plus long chemin

2. Dans un graphe simple non orienté avec n sommets, quel est le nombre maximal d'arêtes ?

A. $\frac{n(n-1)}{2}$

B. $n(n-1)$

C. $\frac{n(n+1)}{2}$

D. n^2

3. Un graphe est considéré comme planaire si :

A. Il peut être dessiné sur le plan sans qu'aucune arête ne se croise

B. Il contient un cycle

C. Il possède au moins un pont

D. Tous les sommets son reliés par exactement deux arêtes

4. Lequel des graphes suivants est toujours hamiltonien

A. Un graphe biparti

B. Un graphe où tous les sommets ont un degré impair

C. Un graphe complet K_n

D. Un arbre

5. Dans un graphe représenté par une matrice d'adjacence, quelle propriété indiquerait une boucle sur un sommet

A. Une valeur non nulle dans une position diagonale

B. Une valeur non nulle sans toute position hors diagonale

C. Une valeur nulle dans toutes les positions diagonales

D. Des valeurs symétriques

6. Le degré d'un sommet dans un graphe est défini comme :

A. Le nombre de sommets qui lui sont connectés par des arêtes

B. Le nombre d'arêtes connectées à ce sommet

C. La distance maximale de ce sommet à tout autre sommet

D. La somme des distances de ce sommet à tous les autres sommets

7. Laquelle des affirmations suivantes à propos des graphes eulériens est correcte ?

A. Tout sommet d'un graphe eulérien doit avoir un degré impair.

B. Un cycle eulérien visite chaque sommet exactement une fois.

C. Un graphe connexe avec tous les sommets de degré pair est eulérien.

D. Un chemin eulérien doit être simple.

8. Quelle est la relation entre les degrés des sommets et le nombre d'arêtes dans tout graphe non orienté ?

A. La somme des degrés de tous les sommets est égale au nombre d'arêtes.

B. La somme des degrés de tous les sommets est deux fois le nombre d'arêtes.

C. Le produit des degrés de tous les sommets est égal au nombre d'arêtes.

D. La somme des degrés de tous les sommets est la moitié du nombre d'arêtes.

9. Un graphe est biparti si et seulement si ?

A. Tous les sommets ont le même degré

B. Il ne contient aucun cycle de longueur impaire

C. Il est planaire

D. Il est connexe et acyclique

10. Quelle est une condition nécessaire et suffisante pour qu'un graphe soit hamiltonien ?

A. Le graphe est 2-régulier

B. Le graphe a un cycle contenant tous les sommets

C. Le graphe est complet

D. Aucune condition simple n'existe

2.2 Graphes orientés

2.2.1 Définitions

En donnant un sens aux arêtes d'un graphe, on obtient un **digraphe** (ou **graphe orienté**). Un digraphe fini $G = (V, E)$ est défini par l'ensemble fini $V = \{v_1, v_2, \ldots, v_n\}$

dont les éléments sont appelés sommets, et par l'ensemble fini $E = \{e_1, e_2, \ldots, e_m\}$ dont les éléments sont appelés **arcs**.

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

2.2.2 Degré d'un sommet d'un digraphe

Soit v un sommet d'un graphe orienté.

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

2.2.3 Chemins et circuits

Un **chemin** conduisant du sommet a au sommet b est une suite ayant pour éléments alternativement des sommets et des arcs, commençant et se terminant par un sommet, et telle que chaque arc est encadré à gauche par son sommet origine et à droite par son sommet destination. On ne peut donc pas prendre les arcs à rebours. Par convention, tout chemin comporte au moins un arc.

On appelle **distance** la longueur du plus petit chemin les reliant. S'il n'existe pas de chemin entre les sommets x et y, on pose $d(x, y) = \infty$.

Un **circuit** est un chemin dont les sommets de départ et de fin sont les mêmes.

Les notions de chemins et de circuits sont analogues à celles des chaînes et des cycles pour les graphes non orientés.

2.2.4 Digraphe fortement connexe

Un digraphe est **fortement connexe**, si toute paire ordonnée (a, b) de sommets distincts du graphe est reliée par au moins un chemin. En d'autres termes, tout sommet est atteignable depuis tous les autres sommets par au moins un chemin.

On appelle **composante fortement connexe** tout sous-graphe induit *maximal* fortement connexe (*maximal* signifie qu'il n'y a pas de sous-graphe induit connexe plus grand contenant les sommets de la composante).

2.2.5 Représentations non graphiques des digraphes

Matrice d'adjacences

On peut représenter un digraphe par une matrice d'adjacences. Une matrice $n * m$ est un tableau de n lignes et m. (i, j) désigne l'intersection de la ligne i et de la colonne j.

Dans une matrice d'adjacences, les lignes et les colonnes représentent les sommets du graphe. Un **1** à la position (i, j) signifie qu'un arc part de i pour rejoindre j.

Cette matrice a plusieurs caractéristiques :

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

Listes d'adjacences

On peut aussi représenter un digraphe en donnant pour chacun de ses sommets la liste des sommets qu'on peut atteindre directement en suivant un arc (dans le sens de la flèche).

2.2.6 Digraphes sans circuits

Théorème : Le digraphe G est sans circuit si et seulement si on peut attribuer un nombre $r(v)$, appelé le **rang** de v, à chaque sommet$v$ de manière que pour tout arc (u, v) de G on ait $r(u) < r(v)$.

Algorithme de calcul de rang

Donnée : digraphe $G = (V, E)$ sans circuit
Résultat : rang $r(v)$ de chaque sommet $v \in V$ du digraphe G
Début
$r \leftarrow 0$
$X \leftarrow V$
R : l'ensemble des sommets de X sans prédécesseur dans X
Tant que X n'est pas vide **faire** :
$r(v) \leftarrow r$ pour tout sommet $v \in R$
$X \leftarrow X - R$
R : l'ensemble des sommets de X sans prédécesseur dans X

$r \leftarrow r + 1$

Fin tant que

Fin

2.2.7 Graphes de comparabilité

Un graphe est de comparabilité si on peut orienter ses arêtes de façon transitive, c'est-à-dire de telle sorte que s'il existe un arc de i vers j et un arc de j vers k, alors il existe également un arc de i vers k.

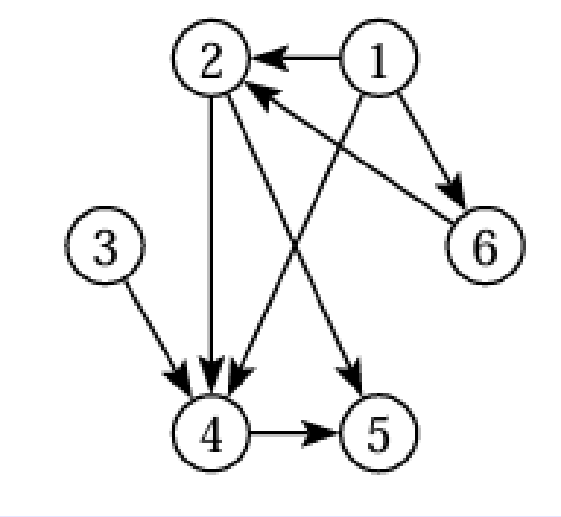
2.2.8 Applications

Application 1

Construire un graphe orienté dont les sommets sont les entiers compris entre 1 et 12 et dont les arcs représentent la relation *être diviseur de*.

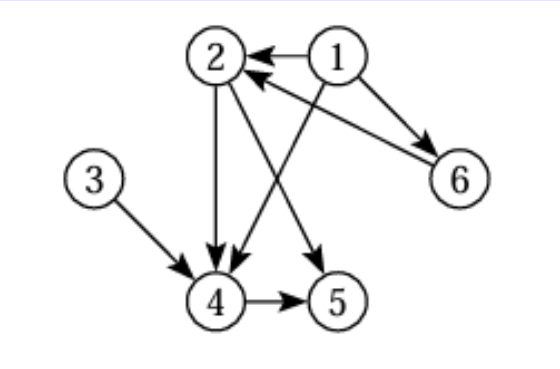
Application 2

Trouvez les degrés extérieurs et intérieurs de chacun des sommets du graphe ci-dessous :



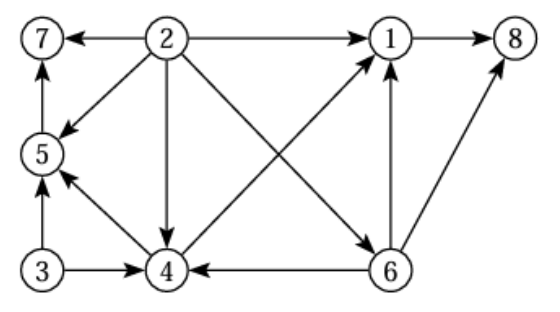
Application 3

Dresser la matrice d'adjacence et la liste d'adjacence du digraphe suivant :



Application 4

Attribuez un rang aux sommets du digraphe ci-dessous en utilisant l'algorithme de calcul de rang



2.3 Problème du plus court chemin : Algorithme de Dijkstra

2.3.1 Présentation

Edgser Wybe Dijkstra (1930-2002) a proposé en 1959 un algorithme qui permet de calculer le plus court chemin entre un sommet particulier et tous les autres. Le résultat est une **arborescence**, c'est-à-dire un arbre avec un sommet particulier appelé **racine**.

Numérotons les sommets du graphe $G = (V, E)$ de 1 à n. Supposons que l'on s'intéresse aux chemins partant du sommet 1. On construit un vecteur $\lambda = \lambda(1); \lambda(2); \ldots; \lambda(n)$ ayant n composantes tel que $\lambda(j)$ soit égal à la longueur du plus court chemin allant de 1 au sommet j. On initialise ce vecteur à c_{1j}, c'est-à-dire à la première ligne de la matrice des coûts du graphe, définie comme indiqué ci-dessous :

$$c_{ij} = \begin{cases} 0 & \text{si } i = j, \\ \infty & \text{si } i \neq j \text{ et } (i,j) \notin E, \\ \delta(i,j) & \text{si } i \neq j \text{ et } (i,j) \in E. \end{cases}$$

où $\delta(i,j) > 0$ est le poids de l'arc (i,j).

On construit un autre vecteur p pour mémoriser le chemin pour aller du sommet 1 au sommet voulu. La valeur $p(i)$ donne le sommet qui précède i dans le chemin.

On considère ensuite deux ensembles de sommets, S initialisé à 1 et T initialisé à 2,3,...,n. A chaque pas de l'algorithme, on ajoute à S un sommet jusqu'à ce que $S = V$ de telle sorte qye le vecteur λ donne à chaque étape la longueur minimale des chemins de 1 aux sommets de S.

2.3.2 Algorithme de Dijkstra

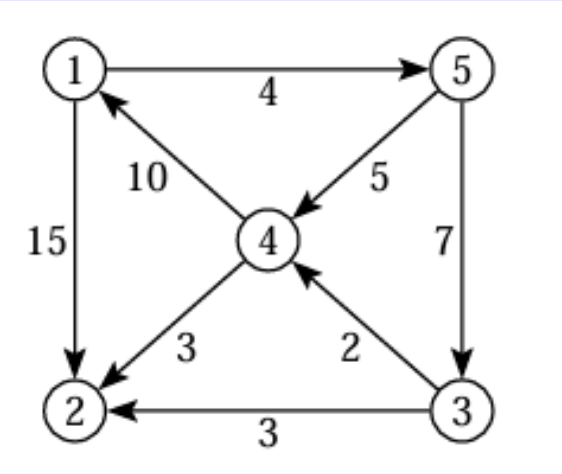
On suppose que le sommet de départ (qui sera la racine de l'arborescence) est le sommet numéroté 1. Notons qu'on peut toujours numéroter les sommets pour que ce soit le cas.

Initialisations
$\lambda(j) = c_{1j}$ et $p(j) =$ NIL , pour $1 \leq j \leq n$
Pour $2 \leq j \leq n$ **faire**
Si $c_{1j} < \infty$ **alors** $p(j) = 1$
$S = 1; T = 2, 3, \ldots, n$
Itérations
Tant que T n'est pas vide **faire**
Choisir i dans T tel que $\lambda(i)$ est minimum
Retirer i de T et l'ajouter à S.
Pour chaque successeur de j de i, avec j dans T, **faire** :
Si $\lambda(j) > \lambda(i) + \delta(i,j)$ **alors**
$\lambda(j) = \lambda(i) + \delta(i,j)$
$p(j) = i$

2.3.3 Applications

Application 1

En utilisant l'algorithme de Dijkstra, déterminez le plus court chemin permettant de quitter 1 à 5 en passant par tous les points.



Application 2

À partir de l'exemple de l'application 1.2.2., déterminez le plus court chemin de Cotonou à Parakou en utilisant l'algorithme de Dijkstra.

2.4 Réseau PERT : Project Evaluation and Review Technique

2.4.1 Présentation

Le problème du **plus long chemin** dans les digraphes sans circuit trouve une application dans l'ordonnancement et la planification des tâches composant un projet complexe, par exemple la construction d'une maison.

On fait correspondre à chaque tâche un arc d'un digraphe, sa durée d'exécution étant égale au poids de cet arc. Le digraphe reflète les précédences requises dans l'exécution du projet. Ainsi, la tâche correspondant à l'arc (i, j) ne peut commencer que si toutes les tâches correspondant à des arcs (k, i) ont été complétées. Le digraphe peut contenir des tâches fictives de durée nulle afin de forcer certaines précédences.

Les sommets du digraphe représentent des événements, début (fin) des activités correspondant aux arcs dont ils sont l'extrémité initiale (finale). Le fait que le digraphe

est sans circuit est garant de la faisabilité du projet. En effet, l'existence d'un circuit impliquerait une contradiction dans les précédences : une tâche devant en même temps précéder et suivre une autre !

On supposera dorénavant que les sommets ont déjà été numérotés de 1 à n de manière compatible avec leurs rangs, c'est-à-dire que $r(j) > r(i)$ implique $j > i$ (voir l'algorithme de calcul du rang). En plus, si le digraphe possède plusieurs sommets sans prédécesseur, on supposera avoir introduit un sommet 1 relié par un arc de durée nulle à chacun de ces sommets. Ce sommet indique le début du projet. De même, si le digraphe possède plusieurs sommets sans successeur, ceux-ci seront reliés par un arc de durée nulle à un dernier sommet n (fin du projet). Enfin, on supposera éliminés les arcs parallèles par l'introduction de tâches fictives.

2.4.2 Algorithme

Donnée : digraphe $G = (V, E)$ sans circuit des activités avec leur durée d_{ik}

Notations :

$P(i) = \{k \in V \mid (k, i) \in E\}$: c'est l'ensemble des sommets prédécesseurs de i.

$S(i) = \{k \in V \mid (i, k) \in E\}$: c'est l'ensemble des sommets successeurs de i.

Résultat :

- δ_i : début au plus tôt des activités correspondant aux arcs (i, k) partant de i,
- Φ_i : fin au plus tard des activités correspondant aux arcs (k, j) arrivant à i,
- durée du chemin critique.

Début

- Calcul des dates de début au plus tôt (récurrence en avançant dans le projet)

  $\delta_1 \leftarrow 0$

  Pour $k \leftarrow 2$ à n faire $\delta_k \leftarrow \max\{\delta_j + d_{jk} \mid j \in P(k)\}$

- Calcul des dates de fin au plus tard (récurrence en reculant dans le projet)

  $\Phi_n \leftarrow \delta_n$

  Pour $k \leftarrow n - 1$ à 1 faire $\Phi_k \leftarrow \min\{\Phi_j - d_{kj} \mid j \in S(k)\}$

Fin

2.4.3 Définitions

- ☞ Un **sommet** i est **critique** si $\delta_j = \Phi_j$.
- ☞ Un **arc** (i, j) est **critique** si ses extrémités sont des sommets critiques et $d_{ij} = \delta_j - \delta_i$.
- ☞ Un **chemin critique** est un chemin de 1 à n n'utilisant que des arcs critiques c'est-à-dire des activités telles que tout retard dans leur exécution provoquerait un retard de la fin du projet.
- ☞ La **durée du chemin critique** est donnée par δ_n (ou par Φ_n, les deux valeurs étant toujours égales). Elle correspond à la durée minimale du projet étant donné les durées des tâches le composant et les précédences respectives.

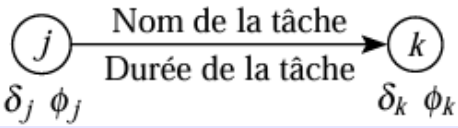
2.4.4 Applications

Application 1

Construisez le diagramme PERT des activités suivantes puis identifiez le chemin critique.

Tâches	Précédences	Durée (jours)
A	–	3
B	–	9
C	–	5
D	A	8
E	B	4
F	B	7
G	B	20
H	C, F	6
I	D, E	5

Par convention, l'écriture d'une tâche est la suivante :



Application 2

À partir de l'exemple de l'application 1.2.1. :

- ☞ Construisez le diagramme PERT correspondant.
- ☞ Quelle est la durée totale du projet ?
- ☞ Quel est le chemin critique de ce projet (si possible) ?
- ☞ De manière générale, comment résoudre le problème posé par un chemin critique ?
- ☞ Dans ce cas spécifique, si possible, quels sont les solutions que vous proposez ?

CHAPITRE 3

MODÈLES LINÉAIRES DE PROGRAMMATION

3.1 Présentation de la programmation linéaire

3.1.1 Définition et objectif de la programmation linéaire

La programmation linéaire est un domaine de l'optimisation qui consiste à maximiser ou minimiser une fonction appelée *fonction-objectif*, notée généralement $Z(X_j)$, où X_j représente un ensemble de n variables de décision. Cette optimisation est réalisée sous un ensemble de contraintes, exprimées par des relations linéaires entre les variables, notées $H_i(X_j) \leq k_i$ pour $i = 1, 2, \ldots, m$ où k_i sont des valeurs limites définies pour chaque contrainte. Les contraintes peuvent-être des inégalités (inférieures ou égales à une valeur donnée) ou des égalités, et elles décrivent les conditions ou les restrictions auxquelles les solutions doivent se conformer.

L'objectif de la programmation linéaire est donc de déterminer la meilleure combinaison possible des variables X_j dans le respect de toutes les contraintes, afin de maximiser ou de minimiser $Z(X_j)$. Ce type de problème est fondamental dans les entreprises, car il permet de modéliser de nombreux cas réels : gestion de production, allocation de ressources, logistique, gestion des stocks, planification financière, etc. En effet, chaque problème impliquant des choix dans un contexte de ressources limitées peut souvent être reformulé sous la forme d'un modèle de programmation linéaire. Les applications de la programmation linéaire se retrouvent ainsi dans pratiquement tous les domaines de la gestion, notamment en logistique pour optimiser les itinéraires, en finance pour maximiser les rendements d'un portefeuille sous certaines restrictions, ou encore en marketing pour allouer un budget publicitaire de façon optimale. Ce champ de l'optimisation est

également utilisé dans d'autres disciplines, telles que l'ingénierie, les télécommunications, l'agriculture, et bien d'autres, grâce à sa capacité à fournir des solutions efficaces aux problèmes complexes sous contraintes.

3.1.2 Définitions

De façon générale, un problème de programmation linéaire met en jeu plusieurs notions importantes :

- ☞ **Variable** : on appelle **variable** toute quantité utile à la résolution du problème dont le modèle doit déterminer la valeur.
- ☞ **Fonction objectif** : on appelle **fonction objectif** d'un problème d'optimisation le critère de choix entre les diverses solutions possibles.
- ☞ **Contraintes** : on appelle **contraintes du problème** toutes les relations limitant le choix des valeurs possibles des variables.
- ☞ **Solution réalisable** : on appelle **solution réalisable** tout *n-uplet* $(x_1, x_2, \ldots, x_n)$ vérifiant le système d'inéquations (d'équations) du problème de programmation linéaire.
- ☞ **Solution optimale** : on appelle **solution optimale** toute solution réalisable qui optimise la fonction objectif.
- ☞ L'ensemble des solutions réalisables du programme linéaire P est appelé domaine des solutions réalisables. Lorsque ce domaine est non vide, on dit que P est réalisable.

3.1.3 Formalisation générale d'un programme linéaire

La formalisation d'un programme est une tâche délicate, mais essentielle, car elle conditionne la découverte ultérieure de la bonne solution. Elle comporte les mêmes phases quelles que soient les techniques requises ultérieurement pour le traitement (programmation linéaire ou programmation non linéaire) :

- ☞ **La détection du problème et l'identification des variables** : ces variables doivent correspondre exactement aux préoccupations du responsable de la décision. En programmation linéaire, les variables sont des variables décisionnelles.
- ☞ **La formulation de la fonction économique ou fonction-objectif** traduisant les préférences du décideur exprimées sous la forme d'une fonction des variables identifiées.

☞ **La formulation des contraintes** : il est bien rare qu'un responsable dispose de toute liberté d'action. Le plus souvent, il existe des limites à ne pas dépasser qui revêtent la forme d'équations ou d'inéquations mathématiques.

Forme algébrique d'un programme linéaire

La forme générale d'un problème de programmation linéaire de n variables et p contraintes est :

$$\min \text{ ou } \max Z(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

sous les contraintes :

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq, \geq \text{ ou } = b_i$$

pour $i = 1, \ldots, p$; et $x_j \geq 0$, pour $j = 1, \ldots, n$.

Les coefficients c_j de la fonction objectif, les coefficients a_ij du premier membre des contraintes et les seconds membres b_i des contraintes sont des données du problème. Les x_j sont les variables du problème. Enfin, i représente l'indice de la $i^{ième}$ contrainte $(i = 1, \ldots, p)$ et j l'indice de la $j^{ième}$ variable $(j = 1, \ldots, n)$.

Forme matricielle d'un programme linéaire

La propriété de linéarité de la fonction-objectif et des contraintes dans un programme linéaire permet de le représenter sous forme matricielle.

Soit n le nombre de variables et p le nombre de contraintes. Posons $c^T = (c_1, c_2, \ldots, c_n)$ le vecteur des coefficients de la fonction objectif ; $b^T = (b_1, b_2, \ldots, b_p)$ le vecteur du second membre, c'est-à-dire des bornes supérieures des différentes contraintes. Soit A la matrice de p lignes et n colonnes, notée $A(p, n)$:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ldots & \vdots \\ a_{p1} & a_{p2} & \ldots & a_{pn} \end{pmatrix}$$

et $x^T = (x_1, x_2, \ldots, x_n)$ le vecteur des variables de décision.

La forme matricielle d'un programme linéaire s'écrit alors en trois lignes :

$$\max c^T x$$

$$\text{s.c. } Ax \leq b$$

$$x \geq 0$$

3.1.4 Les méthodes de résolution des problèmes de programmation linéaire

Résoudre un programme linéaire consiste à déterminer les valeurs optimales des variables de décision qui permettent de maximiser ou minimiser une fonction économique, appelée fonction-objectif. Plusieurs techniques existent pour aborder ce type de problème, parmi lesquelles **la méthode graphique** est souvent perçue comme la plus rapide et la plus simple. Cependant, cette méthode est limitée, car elle devient impraticable dès que le nombre de variables dépasse deux. En conséquence, des chercheurs ont développé des méthodes algorithmiques, permettant de trouver la solution optimale, même pour des problèmes de grande dimension comportant plusieurs variables et contraintes.

L'une des méthodes algorithmiques les plus puissantes est l'**algorithme du simplexe**. Cet algorithme est particulièrement efficace pour résoudre des problèmes complexes de programmation linéaire, mais il implique des calculs longs et parfois fastidieux, d'où l'importance de l'informatique pour automatiser ces calculs. Grâce aux logiciels de résolution de programmation linéaire (comme Excel Solver, R, etc.), le simplexe peut être appliqué de manière rapide et précise, même pour des problèmes de grande taille.

3.1.5 Application

Pour l'application 1.2.3 abordée à la section 2 du chapitre 1 veuillez :

- ☞ Définir les variables, la fonction-objectif et les contraintes
- ☞ Formaliser le problème sous forme algébrique
- ☞ Formaliser le problème sous forme matricielle

3.2 Résolution par la méthode graphique

Lorsqu'il n'y a que deux variables de décision, un problème linéaire peut être résolu de manière purement graphique. Pour cela, il convient de suivre un processus en trois (03) étapes :

- ☞ représentation graphique de la région réalisable

☞ représentation graphique de l'objectif

☞ détermination de la solution optimale

3.2.1 Représentation graphique de la région réalisable

On appelle **région réalisable**, l'ensemble des valeurs de variables de décision qui satisfont toutes les contraintes.

Dans un problème de programmation linéaire impliquant deux variables de décision x_1 et x_2, chaque paire (x_1, x_2) peut être représentée par un point dans le plan cartésien. Puisque les variables x_1 et x_2 sont souvent restreintes à des valeurs positives, les points correspondants se trouvent dans le quart Nord-Est du plan (où $x_1, x_2 \geq 0$.

Chaque contrainte permet de délimiter une partie du plan. Pour ce faire, il est d'abord convenable que les inégalités soient ramenées sous forme d'égalités afin de construire la droite d'équation correspondante. Cette droite délimite deux demi-plans. Selon la nature de la contrainte (sens de l'inégalité), on élimine les points ne répondant pas à notre contrainte. Pour savoir de quel côté de la droite se situe la région qui satisfait l'inégalité, on choisit un point de test (souvent $(0, 0)$ si possible). Si le point satisfait l'inégalité (de la contrainte), alors le demi-plan contenant ce point fait partie de la région réalisable. Sinon, on considère l'autre demi-plan. Le processus est répété pour chaque contrainte.

Les solutions réalisables du problème correspondent aux points du plan situés à l'intérieur (et sur les bords) de la région réalisable qui contient les solutions admissibles. La région réalisable, lorsqu'elle est représentée graphiquement, est souvent un polyèdre convexe dans le plan (dans le cas de deux variables, il s'agit d'un polygone). Les sommets du polyèdre sont les points d'intersection des droites formées par les contraintes. Ces sommets sont particulièrement importants, car dans la programmation linéaire, les solutions optimales se trouvent toujours à ces points extrêmes, conformément au **théorème des points extrêmes** (dans un problème de programmation linéaire, si une solution optimale existe, elle se trouve toujours à un des sommets ou points extrêmes de la région réalisable).

3.2.2 Représentation graphique de l'objectif

Lorsqu'on cherche à résoudre graphiquement un modèle de programmation linéaire, après avoir identifié la région réalisable, l'étape suivante consiste à déterminer les points qui maximisent ou minimisent la fonction objectif.

$$z = c_1 x_1 + c_2 x_2$$

Dans cette formulation, z, x_1, x_2 sont trois variables. Cela signifie que la fonction objectif définit un plan dans un espace tridimensionnel. Cependant, dans le cadre d'une résolution graphique, on se limite à deux dimensions (le plan $x_1 x_2$) en fixant z à une valeur constante.

$$z = k$$

où k est une valeur spécifique de l'objectif.

Pour une valeur donnée de k, l'équation de la fonction objectif devient une droite de la forme :

$$c_1 x_1 + c_2 x_2 = k$$

Ces droites, appelées lignes d'isovaleur, regroupent tous les points (x_1, x_2) qui donnent la même valeur de $z = k$. Autrement dit, tous ces points sont équivalents du point de vue de l'objectif. Comme c_1 et c_2 sont constants, toutes les lignes d'isovaleur associées à des valeurs différentes de k sont parallèles entre elles.

Le déplacement des lignes d'isovaleur permet d'identifier les points optimaux :

- ☞ **Augmentation de** k (maximisation) :

  Déplacer la droite parallèlement vers la droite ou vers le haut dans le plan correspond à augmenter la valeur de z. L'objectif est de pousser la droite aussi loin que possible tout en restant à l'intérieur de la région réalisable.

- ☞ **Diminution de** k (minimisation) :

  Déplacer la droite parallèlement vers la gauche ou vers le bas dans le plan correspond à diminuer la valeur de z. L'objectif est de pousser la droite aussi loin que possible tout en restant à l'intérieur de la région réalisable.

3.2.3 Détermination de la solution optimale

Pour maximiser (respectivement minimiser) l'objectif, il faut prendre la droite d'isovaleur de l'objectif qui touche la région réalisable et qui donne la plus grande (respectivement la plus petite) valeur à l'objectif. Le point de contact est un point optimum.

En considérant le théorème des points extrêmes, la solution optimale est située à un sommet de la région réalisable.

3.2.4 Application

Résoudre le programme de l'application 1.2.3.

3.3 Résolution par l'algorithme du Simplexe

3.3.1 Forme canonique et forme standard d'un programme linéaire

Un programme linéaire est sous **forme canonique** lorsque toutes ses contraintes sont des inégalités et toutes ses variables sont non négatives.

La représentation matricielle d'une forme canonique est :

$$\max c^T x$$

$$\text{s.c. } Ax \leq b$$

$$x \geq 0$$

$c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^p$ et $A \in \mathbb{R}^{m*n}$ avec n variables et p contraintes.

Un programme linéaire est sous **forme standard** lorsque toutes ses contraintes sont des égalités et toutes ses variables sont non-négatives.

La représentation matricielle d'une forme standard est :

$$\max c^T x$$

$$\text{s.c. } Ax = b$$

$$x \geq 0$$

$c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^p$ et $A \in \mathbb{R}^{m*n}$ avec n variables, p contraintes et $p < n$.

Théorème d'équivalence des formes standard et canonique : Tout programme linéaire peut s'écrire sous forme standard et sous forme canonique.

Le passage d'une forme canonique à une forme standard est possible avec l'ajout d'une **variable d'écart** à laquelle on impose d'être non-négative.

La **variable d'écart** est la quantité qui, ajoutée au membre de gauche d'une contrainte, permet de transformer la contrainte en égalité.

3.3.2 Principe de l'algorithme du Simplexe

L'algorithme du Simplexe consiste à déterminer une solution optimale en se déplaçant d'un sommet à un autre sommet adjacent, dans la région réalisable.

On appelle **sommet** du polygone un point intersection de deux contraintes à l'égalité vérifiant toutes les contraintes du problème.

On appelle **arête du polygone** les points de la région réalisable vérifiant une contrainte d'égalité. Géométriquement, il s'agit d'un segment de droite situé à la frontière de la région réalisable.

Deux **sommets adjacents** sont des sommets reliés par une arête.

3.3.3 Notion de solution de base

On appelle **variables hors base** (v.h.b.) les n variables de $\mathbb{R}^{n+p}$ fixées à zéro. Les p variables restantes sont appelées **variables de base** (v.d.b.).

On appelle **solution de base** une solution où en ayant choisi n variables hors base, on obtient une solution unique en résolvant les m contraintes d'égalités obtenues en ajoutant les variables d'écart.

On appelle **solution de base réalisable** une solution de base qui, en plus, vérifie les contraintes de positivité.

Propriété 3.1. : La notion géométrique de sommet du polygone correspond à la notion algébrique de solution de base réalisable.

Nous avons déjà expliqué que l'algorithme du Simplexe repose sur un déplacement de sommet en sommet adjacent. D'un point de vue algébrique, cela correspond à passer d'une solution de base réalisable à une autre. Par conséquent, la notion d'adjacence doit être généralisée pour inclure les solutions de base.

On appelle **solutions de base adjacentes** deux solutions de base dont les variables de base sont les mêmes sauf une qui est de base dans la première base et hors base dans la seconde.

Propriété 3.2. : La notion géométrique de sommets adjacents correspond à la notion algébrique de solutions de base réalisables adjacentes.

3.3.4 Étapes de la mise en œuvre de l'algorithme du Simplexe

On rappelle que l'algorithme du Simplexe consiste à passer d'une solution de base réalisable (correspondant à un sommet de la région réalisable) à une autre qui lui est

adjacente. À chaque itération, il s'agit de :

- ☞ choisir comme *variable entrante*, celle du coefficient le plus élevé dans la ligne objectif (ce qui revient à choisir la direction assurant plus grand taux d'accroissement à la fonction objectif)
- ☞ choisir comme *variable sortante*, la première variable de base à s'annuler (ce qui correspond à la rencontre de la première contrainte dans la direction choisie)
- ☞ faire le *pivotage* : la variable entrante prend la colonne de la variable sortante dans le système d'équation et dans l'expression de la fonction objectif.

Une manière de présenter simplement les calculs de cet algorithme est la présentation du *Simplexe en tableaux*.

3.3.5 L'algorithme du Simplexe en Tableaux

Notion de tableau Simplexe

Un **tableau Simplexe** est constitué des coefficients des équations algébriques sans le nom des variables. On aura donc :

- ☞ les coefficients de la fonction objectif ;
- ☞ les coefficients des variables dans le membre de gauche des contraintes ;
- ☞ les coefficients du membre de droite.

où l'on sépare les coefficients de l'objectif des contraintes d'une barre horizontale et les coefficients du membre de gauche des contraintes des coefficients du membre de droite par une barre verticale.

Plusieurs caractéristiques de ce tableau sont à remarquer :

- ☞ tout d'abord, les valeurs du membre de droite donnent les valeurs courantes des variables de base.
- ☞ ensuite la première ligne donne l'opposé des coefficients de la fonction objectif.
- ☞ enfin, le dernier coefficient de la première ligne donne la valeur courante de l'objectif.
- ☞ on identifie des variables de base à une colonne de coefficient de la matrice identité et à un coefficient objectif nul.

Mise en œuvre de l'algorithme

La mise en oeuvre de l'algorithme du Simplexe en tableaux est la suivante :

☞ **Préliminaires**

Dans un premier temps, il convient de mettre le programme linéaire sous la forme standard (par ajout des variables d'écart) dans le cas où le problème ne serait pas encore formalisé sous cette forme.

Par la suite, il s'agit de déterminer les variables hors base, les variables de base et la solution de base admissible. Ce serait la solution de l'opération à l'initialisation de l'algorithme.

☞ **Initialisation**

Construire le premier tableau Simplexe qui correspond aux données du tableau initial.

☞ **Itérations**

Chaque itération est répartie en quatre (04) étapes :

— **Choix de la variable entrante** : La variable entrante (pour signifier qu'*elle entre en base* ; ce qui indique qu'elle était hors base) est celle dont le coefficient est le plus négatif dans la ligne objectif. Une fois identifiée, la colonne de la variable concernée est encadrée et est appelée *colonne pivot*.

— **Choix de la variable sortante** : La variable sortante (pour signifier qu'*elle sort de la base* ; ce qui indique qu'elle était une variable de base) est la variable qui s'annule la première. Cela revient à calculer le minimum du rapport du coefficient du membre de droite de chaque contrainte sur le coefficient correspondant de la colonne pivot lorsque ce dernier est strictement positif. Dans le cas où le coefficient dans la colonne entrante est négatif ou nul, la ligne n'entre pas en compte dans le calcul du minimum. Une fois identifiée, la ligne de la variable concernée est encadrée et est appelée *ligne pivot*.

L'intersection de la ligne pivot et de la colonne pivot donne *l'élément pivot*. On appelle donc **élément pivot** le coefficient situé à l'intersection de la colonne pivot et de la ligne pivot.

— **Le pivotage** : en ligne, la variable sortante est remplacée par la variable entrante puis les modifications suivantes sont effectuées dans cet ordre : la ligne pivot est divisée par la valeur du pivot ; puis les variables de base forment une matrice identité (1 à l'intersection de chaque variable et 0 ailleurs) ; et enfin, les autres valeurs sont calculées à partir du tableau précédent avec la formule $a - \frac{e_{lp} * e_{cp}}{p}$ avec a la valeur de la case à calculer dans le tableau précédent ;

e_{lp} l'élément de la ligne pivot correspondant ; e_{cp} l'élément de la colonne pivot correspondant et p le pivot.

— Identifier la nouvelle solution de base et la nouvelle valeur de l'objectif

Le pivotage permet d'avoir un nouveau tableau et les itérations continuent jusqu'au critère d'arrêt. L'algorithme s'arrête lorsqu'il n'y a aucun candidat lors de la sélection de la variable entrante (plus de coefficient négatif). La solution courante obtenue à cette étape est **la solution optimale**.

3.3.6 Applications

Application 1

Résoudre le programme de l'application 1.2.3. par l'algorithme du Simplexe

Application 2

Résoudre le programme suivant par l'algorithme du Simplexe :

$$\begin{array}{lll} max\, z & = & 2x_1 + x_2 \\ \text{s.c.q.} & \left\{ \begin{array}{rcl} x_1 - 2x_2 & \leq & 2, \\ -2x_1 + x_2 & \leq & 2, \\ x_1, x_2 & \geq & 0. \end{array} \right. \end{array}$$

Que constatez-vous ? Quelle conclusion pouvons-nous en tirer ?

3.4 Analyse post-optimale

Dans cette section, nous examinerons comment la valeur optimale de la fonction objectif d'un programme linéaire est influencée par des modifications apportées à certains coefficients du problème. Une approche simple consisterait à appliquer l'algorithme du Simplexe au problème modifié et à analyser les changements dans la fonction objectif. Toutefois, nous verrons que, si la base optimale reste inchangée, il est possible de prédire directement l'effet des variations sur la fonction objectif sans effectuer de nouveaux calculs. Cette prédiction repose uniquement sur l'exploitation du tableau Simplexe optimal du problème initial. Pour cela, nous étudierons deux cas principaux :

☞ les variations des coefficients de la fonction objectif ;

☞ les variations des coefficients des membres de droite des contraintes.

Ces analyses nous permettront de mieux comprendre les interactions entre les données du problème et la solution optimale, tout en illustrant la puissance des outils analytiques en programmation linéaire.

3.4.1 Analyse post-optimale de l'objectif

La question fondamentale à laquelle nous cherchons à répondre est la suivante : *Quel est l'impact sur la valeur de la fonction objectif lorsqu'on augmente le prix de vente unitaire ou qu'on diminue le coût unitaire de production ?* Cette question revient à analyser comment les variations des coefficients de la fonction objectif influencent la solution optimale et, plus précisément, la valeur optimale de la fonction objectif.

Il est possible de prédire cette variation sous une condition clé : la solution optimale doit rester inchangée. En effet, pour de petites modifications des coefficients de la fonction objectif c_j la base optimale ne change pas, même si la valeur de la fonction objectif Z est affectée. Dans ce contexte, seul le profit optimal, ou toute autre mesure représentée par Z, évolue.

Le résultat suivant peut alors être établi : **La valeur optimale de la $j^{ième}$ variable, x_j^*, mesure l'augmentation de la fonction objectif si la marge unitaire c_j est augmentée d'une unité.**

Par exemple, si le coefficient c_j associé à la variable x_j augmente d'une unité, alors la fonction objectif augmentera de x_j^* unités. Ici, x_j^* représente la valeur de x_j, déterminée à partir de la solution au problème initial.

Cette prédiction reste valable tant que les variations de c_j sont faibles, de manière à ne pas modifier la base optimale. Si ces variations dépassent un certain seuil, une nouvelle base optimale pourrait émerger, rendant les prédictions issues de la base initiale invalides.

Pour garantir la validité des prédictions, il est essentiel d'identifier l'intervalle dans lequel c_j peut varier sans changer la base optimale. Cela implique :

- ☞ une analyse systématique des conditions de faisabilité et d'optimalité ;
- ☞ Une exploitation des coefficients du tableau Simplexe optimal pour délimiter l'intervalle admissible de c_j.

Par exemple, si c_j augmente ou diminue au-delà de cet intervalle, la solution optimale devra être recalculée en utilisant à nouveau l'algorithme du Simplexe.

Pour mieux comprendre cette dynamique, nous allons effectuer une analyse graphique basée sur les résultats de l'application 1.2.3. Cette démarche permettra d'illustrer visuel-

lement comment les variations de c_j influencent la solution optimale et d'expliquer la manière dont les intervalles de stabilité sont déterminés.

3.4.2 Analyse post-optimale du second membre des contraintes

La question à laquelle nous cherchons à répondre ici est la suivante : Quel est l'impact sur la valeur optimale de la fonction objectif lorsque la capacité disponible d'une ressource est augmentée ? Cette question revient à analyser l'effet des variations des coefficients du membre de droite (b_i) sur la solution optimale.

Lorsque la variation du coefficient b_i est suffisamment faible pour que la base optimale demeure inchangée, il est possible de prédire l'impact sur la fonction objectif en exploitant le tableau Simplexe optimal. Cette prévision repose sur la notion de prix caché (ou shadow price), noté y_i^*.

Le **prix caché** y_i^* mesure l'augmentation de la fonction objectif si l'on accroît d'une unité la capacité disponible (b_i). Dans le tableau Simplexe optimal, le prix caché y_i^* est le coefficient de la variable d'écart de la contrainte dans la ligne objectif.

Le prix caché y_i^* est une mesure locale valable uniquement dans un intervalle de variation spécifique de b_i, appelé **intervalle de stabilité**. Si b_i varie au-delà de cet intervalle, la base optimale peut changer, et il devient alors nécessaire de recalculer le problème pour déterminer un nouveau prix caché et une nouvelle solution optimale.

L'intervalle de stabilité est déterminé par l'analyse des contraintes et des coefficients du tableau Simplexe optimal. Tant que b_i reste dans cet intervalle, le prix caché y_i^* fournit une estimation fiable de l'impact sur la fonction objectif. Si b_i sort de l'intervalle de stabilité, un ajustement de la base optimale est requis, et l'impact sur la fonction objectif doit être recalculé en appliquant l'algorithme du Simplexe.

Pour mieux comprendre l'effet des variations de b_i nous procéderons à une analyse graphique, en nous appuyant sur les résultats de l'application 1.2.3. Cette démarche permettra d'illustrer visuellement comment les variations de capacité d'une ressource influencent la valeur optimale de la fonction objectif, tout en montrant les limites d'interprétation des prix cachés.

3.4.3 Application

Faites une analyse post-optimal de l'application 1.2.3. Vous considérerez les cas où les différents paramètres indiqués ci-dessus augmentent d'une unité.

3.5 Dual d'un programme linéaire

3.5.1 Construction du programme dual

À tout programme linéaire, on peut associer un autre programme linéaire appelé **programme dual**. Considérons un programme linéaire sous forme canonique, ce programme sera appelé **programme primal**.

Ainsi, sous forme matricielle, le programme primal est celui que nous connaissons déjà :

$$\max c^T x$$

$$\text{s.c. } Ax \leq b$$

$$x \geq 0$$

$c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^p$ et $A \in \mathbb{R}^{m*n}$ avec n variables et p contraintes.

Le programme linéaire dual est construit en associant à toute contrainte i du primal, une variable duale y_i et à toute variable j du primal, une contrainte. Les coûts associés aux variables du dual sont les seconds membres du primal et les seconds membres du dual sont les coûts du primal. Ainsi, nous avons :

$$\max b^T y$$

$$\text{s.c. } A^T x \geq c$$

$$y \geq 0$$

Pour passer du programme primal au programme dual, les correspondances suivantes sont à suivre :

Théorème 3.1 : Le problème dual du problème dual est le problème primal

Problème de Maximisation	Problème de Minimisation
Contraintes	
$\leq$	$\geq$
$=$	$=$
Variables	
$\geq$	$\geq$
non restreinte	non restreinte

TABLE 3.1 – Règles de construction

3.5.2 Théorème de la dualité

Ce théorème met en relation les solutions de ces deux (02) programmes linéaires (primal et dual) : de la solution de l'un, on déduit celle de l'autre.

Théorème 3.2 : Dualité faible

Considérons la paire primale-duale :

$$\max c^T x$$

$$\text{s.c. } Ax \leq b$$

$$x \geq 0$$

et

$$\max b^T y$$

$$\text{s.c. } A^T x \geq c$$

$$y \geq 0$$

Si x est une solution admissible du primal et y une solution admissible du dual, alors $c^T x \leq b^T y$. S'il y a égalité, alors x est une solution optimale du primal et y une solution optimale du dual.

Théorème 3.2 : Dualité forte

Considérons la paire primale-duale :

$$\max c^T x$$

$$\text{s.c. } Ax \leq b$$

$$x \geq 0$$

et

$$\max b^T y$$

$$\text{s.c. } A^T x \geq c$$

$$y \geq 0$$

Si le primal et le dual admettent tous les deux une solution admissible, ils ont tous deux une solution optimale finie et la même valeur objectif optimale. Si le primal (dual) est non borné, le dual (primal) n'admet pas de solution admissible.

3.5.3 Application

Construisez le programme dual de l'application 1.2.3. Quelle est la solution optimale ?