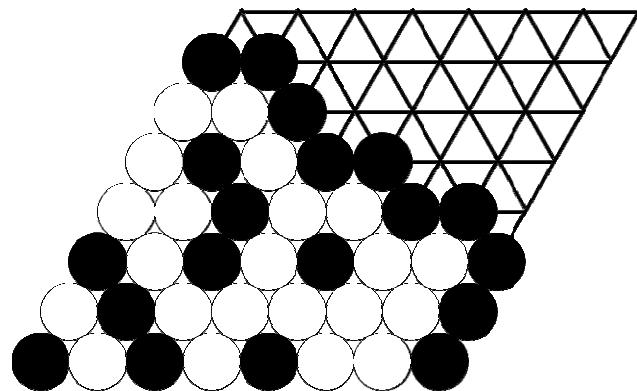


**INTRODUCTION
à
LA RECHERCHE OPERATIONNELLE
et
L'OPTIMISATION COMBINATOIRE**



AVANT PROPOS

L'enseignement à l'ENSIMAG repose sur un ensemble de cours spécifiques. Mais pour un futur ingénieur-informaticien, recevoir uniquement une formation spécialisée comporte un très grand risque de devenir rapidement un "spécialiste-kleenex" (on a vraiment besoin de lui, on l'utilise et on le jette). La vocation de l'ENSIMAG est donc non seulement de former des ingénieurs généralistes mais aussi de les préparer à la vie économique et industrielle, de plus en plus complexe. Etre un ingénieur de haut niveau ce n'est pas seulement connaître l'informatique mais c'est surtout maîtriser toutes les autres composantes et acquérir les facultés d'abstraction et de formalisation, être capable de s'adapter à l'inévitable croissance de la complexité des tâches auxquelles il sera confronté.

Cette nécessité est réalisée, en grande partie, par l'enseignement structuré de la Recherche Opérationnelle à l'ENSIMAG. Toutes les Grandes Ecoles, qui prétendent former ce qu'on appelle traditionnellement des ingénieurs-mathématiciens, attachent à l'enseignement de la Recherche Opérationnelle une très grande importance.

En ce qui concerne l'intitulé, il existe plusieurs définitions formelles, en voici deux. La première, proposée par la Société de Recherche Opérationnelle de Grande-Bretagne :

"La Recherche Opérationnelle"

consiste en l'application des méthodes scientifiques pour résoudre les problèmes complexes rencontrés dans la direction et la gestion de grands systèmes d'hommes, de machines, de matériaux et d'argent dans l'Industrie, le Commerce, l'Administration et la Défense.

la caractéristique de l'approche est le développement d'un modèle scientifique du système (incluant la mesure de facteurs tels que le hasard et le risque) avec lequel on tente de prévoir et de comparer les résultats de diverses décisions ou stratégies.

le but est d'aider la direction à déterminer sa politique de manière scientifique."

Une deuxième proposée par la Société de Recherche Opérationnelle américaine :

"La Recherche Opérationnelle"

est l'ensemble des techniques scientifiques permettant de déterminer une meilleure organisation des systèmes "hommes-machines" sous contraintes de ressources limitées."

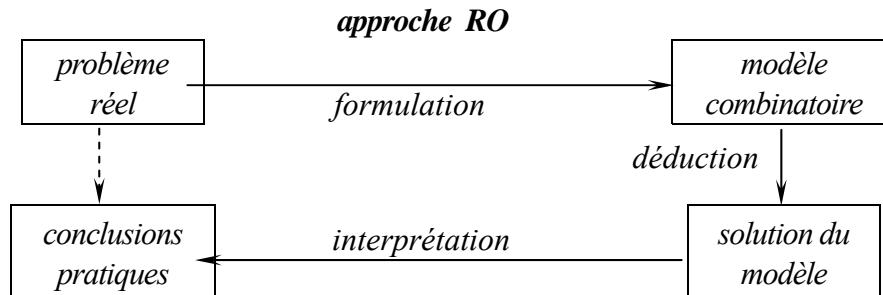
On constate malheureusement qu'aujourd'hui encore, pour la plupart des gens, le terme Recherche Opérationnelle est peu évocateur. Pour les industriels, qui s'impatientent d'avoir vite des résultats pratiques, le mot *recherche* est trop souvent le synonyme d'une abstraction inutile, et le terme *opérationnelle*, mis à part éventuellement son emploi dans le langage militaire, ne signifie rien.

Pour expliquer ce qu'est la Recherche Opérationnelle, il faut dire d'abord, en simplifiant, que c'est la recherche, donc une approche scientifique, dont les résultats doivent être opérationnels dans la pratique, malgré une complexité élevée et une très grande taille du problème.

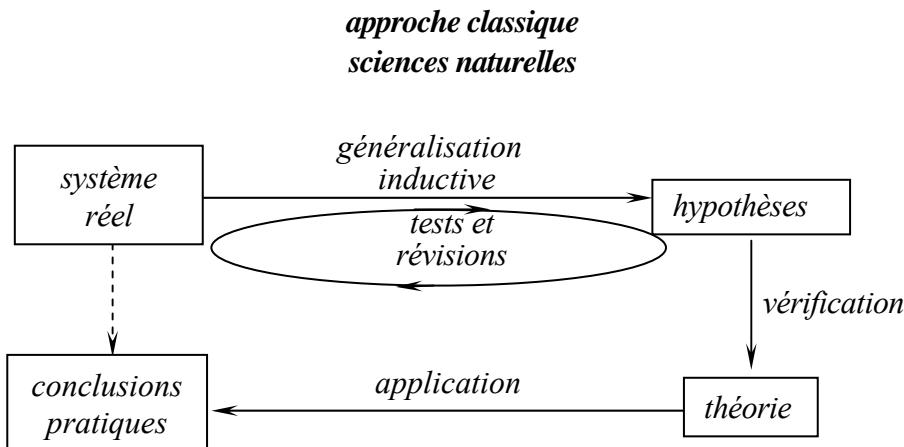
Puisque l'on étudie très fréquemment des problèmes de décision, on rencontre dans la littérature le synonyme de la Recherche Opérationnelle qui est :

Techniques Scientifiques d'Aide à la Décision.

Une caractéristique de la Recherche Opérationnelle, qui la distingue des autres domaines scientifiques, est de prendre en charge *la modélisation* du problème ainsi que *la résolution* de ce modèle, pour ensuite confronter la solution du modèle avec *la réalité*. Cette triple approche des problèmes est sans doute l'aspect le plus attrayant de la Recherche Opérationnelle car trop souvent, dans d'autres domaines, ces trois parties ont été déconnectées pour être étudiées séparément.



Ainsi, la physique bâtit les théories (par exemple les lois universelles) permettant la modélisation du problème, mais la résolution de ces modèles est complètement séparée et confiée à une autre discipline comme les *Mathématiques Appliquées* (*méthodes numériques* ou *calculs scientifiques* par exemple), et ce sont finalement les expériences industrielles qui permettront d'ajouter une retouche pratique pour réussir une réalisation finale.



En Recherche Opérationnelle, on commence par la modélisation d'un problème concret en fonction des outils disponibles, ce qui signifie que le modèle retenu est inspiré par les méthodes de résolution envisageables qui elles-mêmes vont servir à interpréter la réalité.

La phase de la modélisation est illustrée sur plusieurs exemples classiques, très fréquemment rencontrés dans la pratique. La règle principale dans la phase de modélisation est de ne pas charger le modèle avec des complications inutiles. En Recherche Opérationnelle, *plus compliqué* n'est pas nécessairement synonyme de *meilleur*. La tendance est plutôt d'enlever le superflu et de profiter pleinement des particularités du problème concret pour faciliter la tâche de solution. Malgré une jeunesse relative – la Recherche Opérationnelle n'a que 60 ans – on trouve le principe de modélisation : “pourquoi faire compliqué si l'on peut faire simple” (connu sous le nom du *rasoir d'Occam*) dans la philosophie de Guillaume d'Occam (1300-1350), qui date du XIV^e siècle.

Ce cours est tout d'abord consacré aux techniques mathématiques utilisées en Recherche Opérationnelle, c'est-à-dire à la description des outils mis à la disposition du modélisateur. Il faut noter que les problèmes que cherche à résoudre la Recherche Opérationnelle ont permis de développer de nouvelles techniques mathématiques. En particulier de nombreux modèles déterministes (c'est-à-dire quand tous les paramètres sont connus avec exactitude) se formulent comme des problèmes de mathématiques sur des structures finies. La Recherche Opérationnelle, dans son aspect mathématique rejoint donc la Théorie des graphes et, plus généralement la Combinatoire. Elle rejoint également certaines branches de l'Informatique Fondamentale par les questions qu'elle se pose concernant l'efficacité des algorithmes mis en œuvre. La Recherche Opérationnelle par les outils qu'elle développe apparaît très fortement liée aux Mathématiques Appliquées, à l'Informatique et également aux Sciences Economiques et Sociales.

Cet enseignement décrit donc des méthodes générales importantes qui doivent faire partie de la culture scientifique de tout ingénieur, car signalons que ces dernières années les outils scientifiques développés à partir de problèmes classiques de Recherche Opérationnelle ont trouvé de nouvelles applications dans d'autres domaines très variés. En particulier, la Recherche Opérationnelle l'intervient inévitablement dans l'organisation des systèmes complexes de toutes les natures.

Pour pouvoir présenter en 1^{ère} année la Recherche Opérationnelle en 24 séances de cours-TD il fallait bien faire un choix pédagogique.

L'objectif principal de ce cours est donc de présenter la *programmation linéaire* - la technique la plus célèbre de la recherche opérationnelle, qui se trouve être à la fois un outil efficace de *formulation* et de *Résolution* de modèles que l'on rencontre fréquemment, mais aussi un outil mathématique très riche puisqu'il donne un éclairage sur les méthodes d'optimisation continue (en particulier la théorie de la dualité) et les méthodes d'optimisation discrète. Il s'agit de problèmes d'optimisation dont l'objectif est de maximiser (ou minimiser) une fonction linéaire tandis que les variables doivent satisfaire un ensemble d'équations et/ou d'inéquations linéaires. Ce cours décrit en détail l'algorithme du simplexe ainsi que la structure des programmes linéaires et la théorie de la dualité, dont l'interprétation économique apporta à Kantorovitch le prix Nobel en sciences économiques en 1975.

La Recherche Opérationnelle, dans son aspect mathématique rejoint la *Théorie des Graphes* et, plus généralement la *Combinatoire*. Les problèmes que l'on cherche à résoudre, ont permis de développer de nouvelles techniques mathématiques.

Cet enseignement permettra de se familiariser avec les graphes, qui sont un outil essentiel de l'Optimisation Combinatoire. L'accent sera mis sur la modélisation et sur l'existence de résultats généraux et de diverses méthodes de résolution. Nous allons souligner de nombreuses applications à divers problèmes dans les réseaux, qui apparaîtront dans d'autres cours.

Nous n'entrerons pas dans les profondeurs théoriques des sujets proposés. Le but est de donner des connaissances de base aux ingénieurs qui rencontreront dans leur travail des problèmes d'optimisation discrète et non de former des mathématiciens capables de résoudre de nouveaux problèmes d'optimisation sur les réseaux.

Pour être capable de détecter qu'un problème, que l'on rencontre, relève de l'optimisation combinatoire et de faire le lien entre ce problème et les méthodes qui peuvent s'y appliquer, il faut connaître les techniques et problématiques de base.

Dans la plupart des cas on doit abandonner les méthodes que les élèves sont habitués à voir en cours classique d'analyse ou d'algèbre. La Combinatoire a développé ses propres méthodes, parfois nouvelles et surprenantes, auxquelles il va falloir s'habituer. Ainsi on essaye d'éviter au maximum les calculs avec de longues transformations et on priviliege un certain style de raisonnement. A titre d'exemple, soit à démontrer l'identité remarquable suivante :

$$\binom{n}{p} = \binom{n-1}{p} + \binom{n-1}{p-1} \quad \text{où } \binom{n}{p} \text{ désigne le nombre}$$

de partie à p élément (p -parties) de l'ensemble à n élément (ou n -ensemble).

Chaque lycéen connaît la formule :

$$\binom{n}{p} = \frac{n!}{p!(n-p)!}$$

et avec quelques notions élémentaires de calcul algébrique (factorisation, dénominateur commun...etc.) on arrive à faire les transformations nécessaires.

Une méthode combinatoire consiste à choisir un élément x_0 de notre n -ensemble; ce choix détermine naturellement une partition de l'ensemble de p -parties en deux sous-ensembles : un sous-ensemble de p -parties qui contiennent x_0 et un sous-ensemble de p -parties qui ne contiennent pas x_0 .

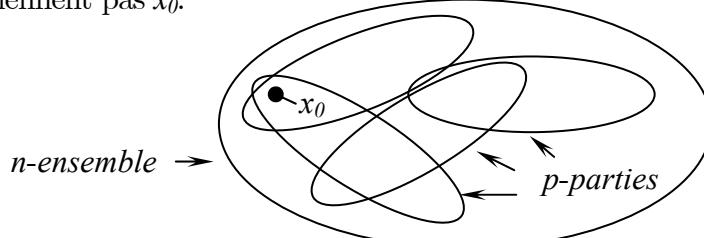


Figure 1.

Enlevant x_0 de notre n -ensemble on obtient un $(n-1)$ -ensemble. Ses p -parties ne contiennent pas x_0 et ses $(p-1)$ -parties sont en bijection avec les p -parties de notre n -ensemble qui contiennent x_0 – d'où l'identité recherchée. Pour la démontrer, il n'était pas nécessaire de connaître la formule explicite pour calculer le nombre p -parties du n -ensemble. Dans cet exemple on avait encore la possibilité de choisir entre le raisonnement ou le calcul, mais il y a des problèmes qui ne se présentent pas avec des formules.

Voici un autre exemple où le raisonnement combinatoire est le seul moyen rapide et convaincant de la démonstration. Considérons l'échiquier classique (8×8) avec deux cases tronquées comme l'indique la figure 2.

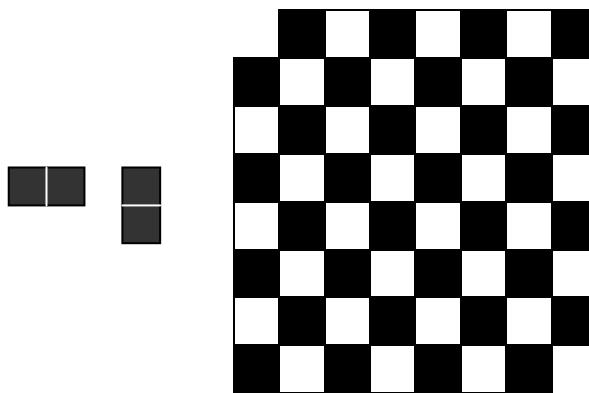


Figure 2.

On dispose de 31 pièces de dominos avec les dimensions correspondantes à deux cases voisines sur l'échiquier.

Peut-on recouvrir les 62 cases de l'échiquier tronqué avec ces 31 dominos ?

La réponse est NON. Pour la justifier inutile d'énumérer tous les cas (il y en a un nombre exponentiel). Il suffit de constater que les deux cases tronquées sont blanches et un domino, quelle que soit sa position sur l'échiquier, recouvre une case blanche et une case noire. Il est impossible de recouvrir ainsi cet échiquier, car le nombre de cases blanches est différent du nombre de cases noires. Cependant si l'on retire de l'échiquier le même nombre de cases blanches et de cases noires, le nombre maximum de dominos qui le recouvre est encore très difficile à déterminer sans la théorie du couplage dans les graphes.

Le dernier exemple illustre que l'aspect combinatoire d'un problème est parfois caché par l'aspect "continu" superficiel.

Imaginons une montre avec un cadran et deux aiguilles, indiquant les heures et les minutes. Nous voulons savoir à quelle moment la surface contenant XII, symétrique par rapport à la droite verticale (XII,VI) non occupée par les deux aiguilles est maximale?

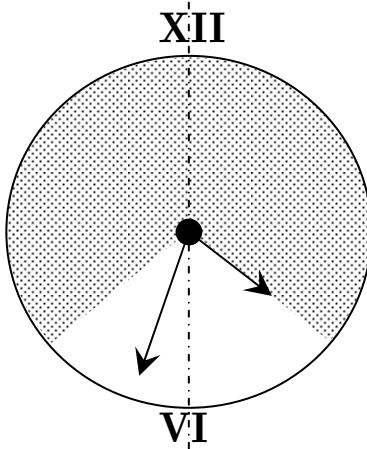


Figure 3.

La même question pourrait concerner une montre avec trois aiguilles, la troisième indiquant les secondes. La réponse serait triviale, si chaque aiguille faisait un nombre impair de tours pendant une journée entière (au milieu de la journée toutes les aiguilles sont en bas).

Cette formulation est seulement une vulgarisation des problèmes très sérieux auxquels s'intéresse la théorie des nombres¹. Certains résultats ont eu des conséquences intéressantes dans la théorie des flots dans les réseaux et dans la transmission des signaux périodiques.

Malgré les apparences "physiques" (mouvement continu des aiguilles), ce problème possède une structure tout à fait discrète (finie). Bien sûr, il y a une infinité de positions relatives des deux aiguilles mais seules les positions symétriques par rapport à la droite verticale (XII,VI) peuvent être optimales. Pourquoi ? Parce que, grâce à la continuité du mouvement, on peut, dans une situation où une aiguille est placée plus haut que l'autre, baisser d'un *epsilon* la position de celle-ci en augmentant ainsi la surface à maximiser. Pendant que la petite aiguille fait un tour, la grande fait 12 tours (12 heures). Les positions sont symétriques à chaque instant où la somme des distances parcourues par les deux aiguilles est un nombre quelconque de tours complets. Les distances étant proportionnelles aux vitesses (1 et 12 respectivement), la première position symétrique arrive à l'instant $1/(1+12)$ (à peu près 23 secondes après une heure moins cinq) et nous avons $1+12=13$ positions symétriques pendant une journée de 12 heures.

Il y a deux positions symétriques aux alentours de la mi-journée (6h) représentées par deux fractions : $6/13$ ($\approx 5h32'18''$) et $7/13$. Le même raisonnement permet de résoudre le cas

¹ Bienia W. et al: "Flows, View Obstructions and the Lonely Runner" Journal of Combinatorial Theory. Series B, 1998.

de trois aiguilles. Les arguments combinatoires ont permis de résoudre facilement beaucoup de problèmes de ce type, qui se soumettent difficilement au traitement continu comme par exemple l'analyse harmonique.

D'après les trois exemples cités la Combinatoire risque d'apparaître comme une collection de problèmes peut-être intéressants en eux-mêmes, mais sans relations entre eux et ne formant pas une théorie. Elle est devenue aujourd'hui l'une des branches les plus florissantes de l'Algèbre moderne et ses méthodes interviennent même dans les problèmes d'Algèbre les plus classiques. La théorie de graphes, unifiée et abstraite, développe des problématiques et des concepts qui lui sont propres, constituant ainsi pour l'informatique théorique, pour l'optimisation et la recherche opérationnelle une source intarissable de modèles et de résultats théoriques. La théorie de graphes est aussi un exemple fécond d'interaction entre domaines fondamentaux et appliqués.

Sans entrer trop loin dans les détails théoriques, les exercices illustreront les problèmes de la Recherche Opérationnelle. Ce polycopié, à but essentiellement pédagogique, doit être considéré comme "un glossaire de base" en Recherche Opérationnelle dans sa formule destinée au futur ingénieur de l'ENSIMAG et aussi comme un manuel de référence qui contient quelques explications théoriques et les exercices réalisés en travaux dirigés pendant les séances hebdomadaires. Quelques problèmes un peu plus complexes, demandant une réflexion personnelle et une utilisation des logiciels sont proposés en travaux pratiques. Leur étude permet de mieux comprendre et d'approfondir le contenu de ce cours ainsi que d'avoir un avant-goût de la suite du cursus des élèves de l'ENSIMAG :

en 1^{ère} année:

- cours d'algorithme;

en 2^{ème} année:

- cours d'optimisation combinatoire;
- travaux d'études et de recherche (TER);
- projets d'optimisation combinatoire;

en 3^{ème} année:

- cours de l'option Logistique, Optimisation et Systèmes d'Information ;
- Master Recherche **ROCO** (Recherche Opérationnelle, Combinatoire et Optimisation);
- projet industriel de fin d'études.

Wojciech BIENIA

Grenoble, le 30 janvier 2009.

1ère partie :

Graphes et Applications

1. Le concept du graphe

Un *graphe* (non orienté) G est un couple (X, E) où $X=\{x_1, x_2, \dots, x_n\}$ est un ensemble fini non vide (*sommets* de G , parfois *nœuds* ou *points*) et $E=\{e_1, e_2, \dots, e_m\}$ est une famille des paires (non ordonnées) d'éléments de X (*arêtes* de G). Pour une arête $e=\{x, y\}$ x et y sont ses extrémités. Lorsque $x=y$, e est une boucle. Deux *arêtes* sont *parallèles* lorsqu'elles ont les mêmes extrémités; un ensemble d'arêtes parallèles est une *arête multiple*. Un *graphe simple* n'admet pas de boucle ni d'arête multiple.

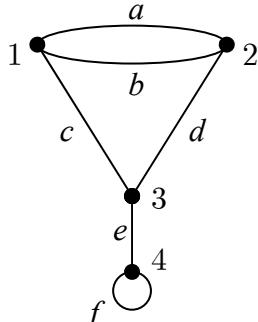


Figure 4.

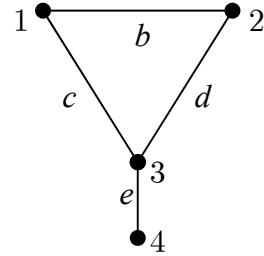


Figure 5.

La figure 4 représente le graphe $G = (X, E)$ avec $X=\{1, 2, 3, 4\}$ $E=\{a, b, c, d, e, f\}$ où $a=\{1, 2\}$, $b=\{1, 2\}$, $c=\{1, 3\}$, $d=\{3, 2\}$, $e=\{3, 4\}$, $f=\{4, 4\}$; f est une boucle et a, b sont deux arêtes parallèles. On considère donc les arêtes comme des paires "étiquetées". La figure 5 représente un graphe simple. Le nom graphe est lié au fait, que l'objet ainsi défini peut être représenté graphiquement. Cette illustration facilite la compréhension de certaines propriétés. Mais l'objet "graphe" est défini d'une façon abstraite et il existe plusieurs représentations formelles possibles que l'on peut choisir en fonction des besoins. Nous allons voir plus loin qu'il est très commode de se servir d'une représentation matricielle pour effectuer une démonstration d'un théorème, et que pour réaliser un algorithme, la représentation par la liste de voisins est beaucoup plus "économique".

Un sommet $x \in X$ et une arête $e \in E$ sont *incidents* si x est une extrémité de e . Deux sommets (arêtes) sont *adjacents* lorsqu'ils sont incidents à une même arête (sommet). Ces deux relations ouvrent deux possibilités pour créer les représentations matricielles d'un graphe. Ainsi à tout graphe avec n sommets et m arêtes on peut associer soit une matrice $n \times n$, appelée *matrice d'adjacence* $A(G)=(a_{xy})$ dont les lignes et les colonnes sont indexées par les sommets du graphe et l'élément a_{xy} indique le nombre d'arêtes ayant x et y comme extrémités, soit une matrice $n \times m$, appelée *matrice d'incidence* $B(G)=(b_{xe})$ dont les lignes sont indexées par les sommets et les colonnes par les arêtes du graphe et l'élément b_{xe} indique le nombre de fois où le sommet x est incident à l'arête e . On note que $b_{xe} \in \{0; 1; 2\}$ et que la somme des éléments d'une colonne est 2.

Avec le graphe de la figure 1 les deux matrices se présentent ainsi:

$$A(G) : \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 2 & 1 & 0 \\ 2 & 2 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 0 & 0 & 1 & 1 \end{array}$$

$$B(G) : \begin{array}{c|cccccc} & a & b & c & d & e & f \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 & 1 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1 & 2 \end{array}$$

L'ensemble des sommets adjacents à x est appelé ensemble des *voisins* de x et noté $N(x)$. Le *degré* $d(x)$ d'un sommet x est le nombre d'arêtes incidentes à x (une boucle comptée deux fois). Si $d(x)=0$ le sommet x est *isolé*, si $d(x)=1$ le sommet x est *pendant*. Une *arête pendante* est une arête incidente à un sommet pendant. Un graphe simple avec n sommets deux à deux adjacents est un *graphe complet* noté K_n . Ainsi, dans un graphe complet, tous les sommets ont le même degré $d(x)=n-1$.

EXEMPLE 1: Trouver un graphe d'ordre 3 avec ses trois degrés différents. Démontrer, que chaque graphe simple $G=(X,E)$ d'ordre au moins 2, contient au moins deux sommets de même degré.

SOLUTION : Voici le graphe avec les degrés 1, 3, 2 :



Dans la deuxième partie considérons d'abord un graphe simple avec n sommets et sans sommet isolé. Les degrés de ses sommets appartiennent à l'ensemble $D=\{1; 2; \dots; n-1\}$ de cardinal $n-1$. Puisqu'il y a n sommets alors il n'existe pas d'injection de X dans D . Il existe donc au moins deux sommets, i et j , tels que $d(x_i)=d(x_j)=d \in D$. Si le graphe admet un sommet isolé unique, alors on ajuste le raisonnement précédent pour un graphe d'ordre $n-1$ et quand il y a au moins deux sommets isolés, alors ils sont tous de degré identique ($=0$). ■

Dans cette démonstration nous avons appliqué un principe fréquemment utilisé dans les raisonnements combinatoires. Connu sous le nom de *principe des cages à pigeons* (ou encore *principe des tiroirs*) il possède l'interprétation suivante:

Soient n pigeons (objets) mis dans m cages (tiroirs) ; si $n > m$ alors il existe au moins deux pigeons (objets) dans une même cage (tiroir).

Ce principe est une conséquence du principe d'induction sur \mathbb{N} .

L'identité remarquable suivante permettra plus tard de résoudre efficacement certains problèmes de dénombrement dans des graphes. Sa démonstration est un exemple d'application d'un autre principe, le principe d'égalité (bien connu sous l'expression : "en calculant une quantité de deux manières différentes").

THEOREME 1 : Pour tout graphe $G=(X,E)$ on a : $\sum_{x \in X} d(x) = 2|E|$.

Preuve. Considérons la matrice d'incidence $\mathbf{B}(G)=(b_{xe})$. La somme des éléments de la ligne indexée par x est le nombre d'arêtes incidentes à x , donc égale à $d(x)$. La somme des éléments de chaque colonne e est égale à 2, car une arête est incidente à deux sommets. Calculons la somme S de tous les éléments de la matrice \mathbf{B} d'abord horizontalement, ensuite verticalement. On obtient $S = \sum_{x \in X} d(x) = 2|E|$. ■

EXEMPLE 2: Nous présentons en annexe le théorème du point fixe de Brouwer comme une simple conséquence du lemme combinatoire dû à Sperner (1928), dont la démonstration s'appuie sur ce théorème.

EXEMPLE 3: Pour garantir la sécurité, une banque a installé un coffre-fort à plusieurs serrures. Ce coffre possède un nombre minimum de serrures avec la caractéristique suivante: chacun des 8 membres du personnel possède un certain nombre de clés, mais l'ouverture n'est possible qu'en présence d'au moins 3 membres quelconques.

- Quel est le nombre de serrures installées?
- Quel est le nombre de clés que chaque personne possède?

SOLUTION : Si un sommet représente une personne alors une arête $\{i,j\}$ peut être interprétée comme une serrure inviolable quand seulement deux personnes, i et j , essayent d'ouvrir le coffre. Cette serrure doit exister, car d'après l'énoncé l'ouverture du coffre nécessite la présence d'au moins trois personnes. On se propose de chercher un nombre minimum de serrures, alors on considère un graphe simple (inutile de multiplier des arêtes ou mettre des

boucles!). Ce graphe est complet puisque pour deux personnes quelconques doit exister une serrure qu'ils ne peuvent pas ouvrir. La situation est modélisée par K_8 , qui possède 28 arêtes. Avant de conclure que notre coffre a 28 serrures, il faut encore justifier que les arêtes distinctes de K_8 représentent les serrures différentes ! C'est le cas, car si deux arêtes $\{i,j\}$ et $\{i,k\}$ (pour i, j, k distinctes) représentaient la même serrure, alors cette serrure serait inviolable par trois personnes i, j, k . Si deux arêtes $\{i,j\}$ et $\{k,l\}$ (pour i, j, k et l distinctes) représentaient la même serrure, alors cette serrure serait inviolable par quatre personnes i, j, k et l . La réponse à la deuxième question est que chaque personne doit être en possession de 21 clés de serrures correspondant aux arêtes non incidentes au sommet qui représente cette personne ($21=28-7$). ■

Le *graphe complémentaire* du graphe simple $G=(X,E)$ est un graphe $\bar{G}=(X,\bar{E})$ où le complément de E est pris par rapport à l'ensemble des arêtes du graphe complet K_n .

On dit que $H=(Y,F)$ est un *sous-graphe* du graphe $G=(X,E)$ engendré par Y lorsque $Y \subseteq X$ et F est formé de la totalité des arêtes de E dont les extrémités sont dans Y . On appelle un *graphe partiel* de $G=(X,E)$ tout graphe $H=(X,F)$ où $F \subseteq E$.

Une clique est un ensemble de sommets qui engendre un graphe complet. On note $\omega(G) := \max \{p \mid K_p \text{ est une clique de } G\}$. Un stable est un ensemble de sommets deux à deux non adjacents. Le *nombre de stabilité* $\alpha(G) := \max \{|S| ; S \text{ est un stable de } G\}$. Un graphe sans boucle est k -parti si son ensemble de sommets admet une partition en k stables. Cette notion correspond également à la partition en cliques dans le graphe complémentaire.

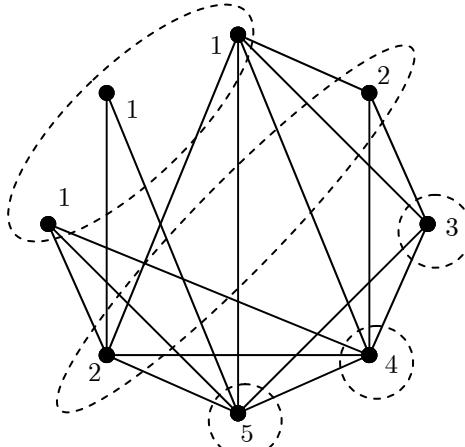


Figure 6. Une partition en 5 stable

On rencontre très fréquemment dans de nombreuses applications les graphes *bipartis* ($k=2$). On note $G=(X,Y;E)$ un graphe biparti. Si tout sommet de X est adjacent à tout sommet de Y et G est simple, alors on dit G est *biparti complet* (il sera noté $K_{p,q}$ où $p=|X|$ et $q=|Y|$).

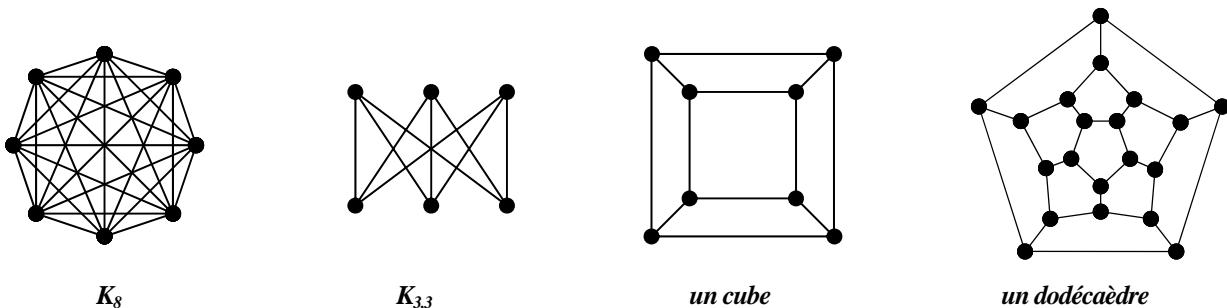
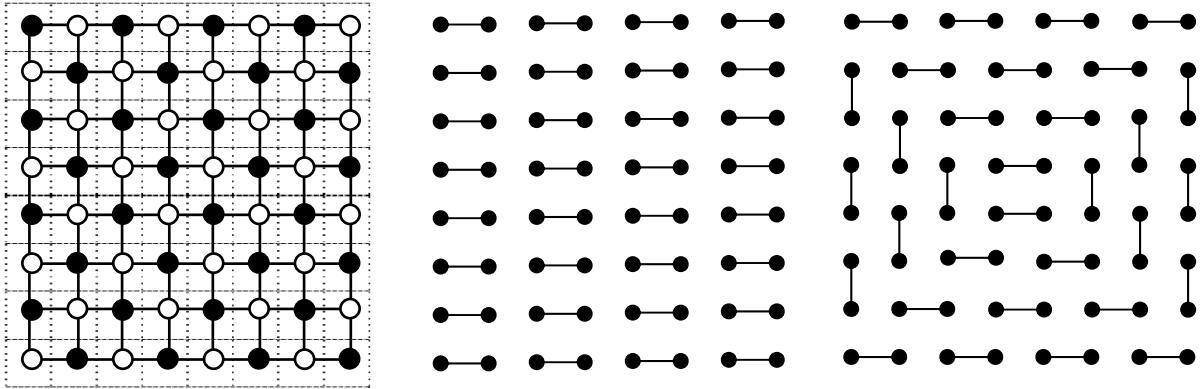


Figure 7.

La notion de stabilité concerne les sommets du graphe. On peut introduire une notion duale concernant des arêtes. Un *coupillage* est un ensemble d'arêtes deux à deux non adjacentes. Le coupillage est *parfait* si $\forall x \in X, \exists$ une arête du coupillage incidente à x .

EXEMPLE 4: Ci-dessous un graphe qui modélise l'échiquier classique 8×8 : les sommets représentent les cases et les arêtes indiquent des positions possibles des dominos. Ce graphe est biparti (les cases blanches et noires) et admet plusieurs coupages parfaits:

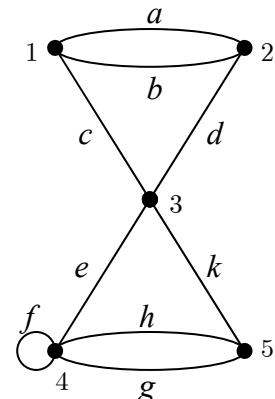


Une *chaîne de longueur* k est une séquence alternée de sommets et d'arêtes $\Gamma = (x_0, e_1, x_1, e_2, \dots, x_{k-1}, e_k, x_k)$ telle que, pour tout i , les extrémités de e_i sont x_{i-1} et x_i . Lorsque $k > 0$ et $x_0 = x_k$ Γ est une *chaîne fermée*. Si e_i sont distincts Γ est une *chaîne simple*. Si x_i sont distincts Γ est une *chaîne élémentaire*. Un *cycle* est une chaîne fermée simple.

Voici quelques exemples dans le graphe ci-contre :

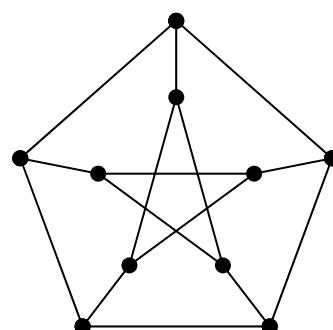
chaîne simple : $4e3c1a2d3k5$;
 chaîne élémentaire : $1a2d3e4h5$;
 cycle : $1c3e4g5k3d2a1$;
 cycle élémentaire : $1c3d2b1$.

Chaque “objet” *élémentaire* est *simple* mais la réciproque n'est pas vraie comme le montrent nos exemples.



Si l'on s'intéresse aux chaînes et cycles extrémaux on obtient des objets remarquables, fréquemment rencontrés dans de nombreuses applications et largement étudiés en théorie. Dans un graphe de n sommets, une *chaîne élémentaire* de longueur $n-1$ est dit *hamiltonien* (il comporte donc $n-1$ arcs et n sommets). Un *cycle hamiltonien* est un cycle élémentaire de longueur n . Un *graphe* est dit *hamiltonien* s'il possède un cycle hamiltonien. On doit le tout premier résultat dans cette matière à Sir William R. Hamilton (1805-1865) – mathématicien irlandais.

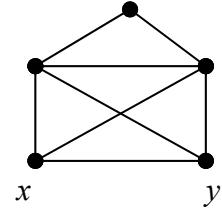
Le problème de l'existence d'élément hamiltonien dans un graphe quelconque est très difficile. Ci-contre le graphe de *Petersen*, qui contient une chaîne hamiltonienne mais ne contient pas de cycle hamiltonien et $\forall x, G \setminus \{x\}$ contient un cycle hamiltonien.



Le concept précédent est basé sur l'obligation de visiter chaque sommet une et une seule fois. Une autre idée est de traverser toutes les arêtes une seule fois.

Dans un graphe avec m arêtes, une chaîne simple de longueur m est dite eulérien. Un cycle eulérien est un circuit de longueur m . Un graphe est dit eulérien s'il possède un cycle eulérien. Ces objets ont été étudiés par Leonhard Euler (1707-1783) – mathématicien suisse.

La figure ci-contre présente un graphe qui n'est pas eulérien mais qui admet une chaîne eulérienne entre les sommets x et y . On démontre plus loin (p.33) que le problème de l'existence d'un élément eulérien dans un graphe est facile.



Soit $T \subseteq X$ un sous-ensemble de sommets du graphe $G=(X;E)$. On appelle **T -joint** un sous-ensemble d'arêtes $J \subseteq E$ tel que T est exactement l'ensemble des sommets de degré impair du graphe partiel $G=(X;J)$. Ci-dessous nous illustrons deux T -joints pour $T=\{1, 4\}$.



On dit qu'un graphe est *planaire* s'il est possible de le représenter sur un plan de sorte que les sommets soient des points distincts, les arêtes des courbes simples, et que deux arêtes ne se rencontrent pas en dehors de leurs extrémités.

On appelle *subdivision* du graphe un graphe obtenu en remplaçant des arêtes par des chaînes élémentaires.



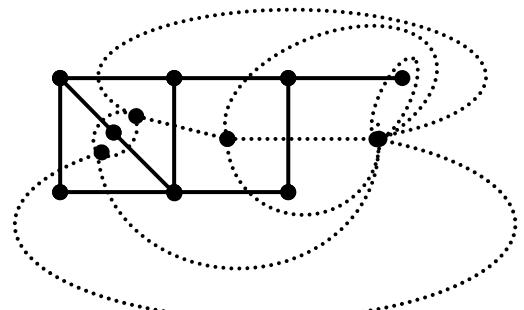
Figure 8. Des subdivisions de K_5 et de $K_{3,3}$.

THEOREME 2 (Kuratowski 1930) : *Un graphe est planaire si et seulement si il ne contient pas (comme sous-graphe engendré) de subdivision de K_5 ni de $K_{3,3}$.* ■

Considérons un graphe planaire \mathbf{G} , connexe et sans sommets isolés. On lui fera correspondre un graphe planaire \mathbf{G}^* (appelé le *graphe dual* de \mathbf{G}) de la façon suivante :

à l'intérieur de toute face de \mathbf{G} , on place un sommet x^* de \mathbf{G}^* ; à toute arête e de \mathbf{G} on fera correspondre une arête e^* de \mathbf{G}^* qui reliera les sommets x^* et y^* correspondant aux faces qui se trouvent de part et d'autre de e .

Evidemment $(\mathbf{G}^*)^* = \mathbf{G}$.



2. Coloration

Soit $G=(X,E)$ un graphe non-orienté. Etant donné un entier k , une k -coloration des sommets de G est une application $c: X \rightarrow \{1, 2, \dots, k\}$ telle que, pour chaque arête $\{x,y\}$ de G , on ait $c(x) \neq c(y)$.

Lorsque $c(x)=i$, nous dirons que le sommet x reçoit la couleur i , ou que la couleur i est affectée à x . On remarque que l'ensemble $\{x \mid c(x)=i\}$ des sommets de couleur i ($i=1, 2, \dots, k$) est un stable de G (sous-ensemble ne contenant aucune arête); une k -coloration peut donc être vue aussi comme une partition de l'ensemble des sommets de G en (au plus) k stables.

Si G admet une k -coloration on dit qu'il est k -colorable. On cherche généralement à trouver une coloration aussi économique que possible, c'est-à-dire qui minimise le nombre de couleurs utilisées. On définit ainsi le *nombre chromatique* $\chi(G)$ d'un graphe G par:

$$\chi(G) = \min \{k, G \text{ est } k\text{-colorable}\}.$$

La détermination de $\chi(G)$ est en général un problème très complexe (\mathcal{NP} -difficile), qui suscite de nombreuses recherches de pointe en théorie des graphes. Nous nous contenterons ici de donner quelques bornes supérieures et inférieures sur ce nombre, et nous présenterons quelques heuristiques permettant de construire une coloration d'un graphe donné.

On observe facilement que tout graphe G d'ordre n satisfait $\chi(G) \leq n$, puisque l'affectation d'une couleur différente à chaque sommet est une coloration. D'autre part, si K_p est une clique de G (ensemble de sommets deux à deux adjacents), toute coloration de G doit utiliser au moins p couleurs sur K_p (mais éventuellement ce nombre de couleurs ne suffit même pas pour colorier tout le graphe). Donc, si l'on pose $\omega(G) = \max \{p \mid K_p \text{ est une clique de } G\}$, on voit que $\chi(G) \geq \omega(G)$. Ces deux observations sont récapitulées dans le lemme suivant :

LEMME 1: Pour tout graphe G sur n sommets on a : $\omega(G) \leq \chi(G) \leq n$.

Un encadrement plus fin du nombre chromatique est obtenu dans le théorème suivant. Soit $\Delta(G)$ le degré maximum de G .

THEOREME 3 (Brooks, 1941)

- (1) *Tout graphe G satisfait $\chi(G) \leq \Delta(G)+1$.*
- (2) *En fait, on a $\chi(G) \leq \Delta(G)$ sauf dans deux cas spéciaux :*
 - a) $\Delta(G)=2$ et une composante connexe de G est un cycle impair (et dans ce cas $\chi(G)=3$);
 - b) G a une composante connexe qui est une clique de taille $\Delta(G)+1$ (et dans ce cas $\chi(G)=\Delta(G)+1$).

PREUVE. Ecrivons pour simplifier $\Delta := \Delta(G)$.

Preuve de (1). Soit x un sommet quelconque de G . Par hypothèse de récurrence, nous pouvons trouver une $(\Delta+1)$ -coloration c de $G \setminus x$. Puisque x a au plus Δ voisins, l'une des $\Delta+1$ couleurs $1, \dots, \Delta+1$ utilisée par c n'est pas présente dans le voisinage de x . Supposons par exemple qu'aucun voisin de x n'a reçu par c la couleur i (i dans $\{1, \dots, \Delta\}$). En affectant la couleur i à x , nous obtenons une $(\Delta+1)$ -coloration de G .

Cette preuve de (1) peut être affinée pour donner une preuve par récurrence de la partie (2) de ce théorème, comme suit. Les deux cas spéciaux a) et b) sont faciles à vérifier.

Supposons maintenant que nous ne soyons pas dans l'un des deux cas spéciaux, et soit x un sommet quelconque de G . Par hypothèse de récurrence, nous pouvons trouver une Δ -coloration c de $G \setminus x$.

1^{er} cas: l'une des Δ couleurs $1, \dots, \Delta$ n'est pas présente dans le voisinage de x . Dans ce cas, nous pouvons affecter cette couleur à x et obtenir une Δ -coloration de G .

2^{eme} cas: toutes les couleurs $1, \dots, \Delta$ sont présentes dans le voisinage de x . Pour chaque paire i, j de couleurs (i, j dans $\{1, \dots, \Delta\}$), appelons $H_{i,j}$ le sous-graphe de G induit par les sommets

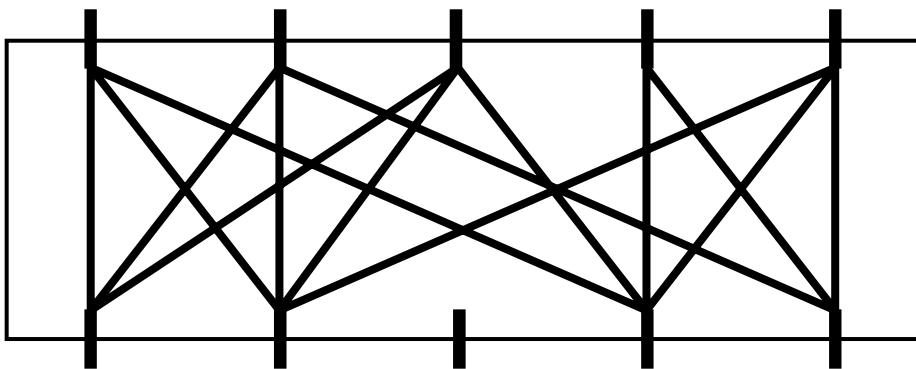
de couleur i ou j . Alors on peut prouver qu'il existe deux couleurs i, j ayant la propriété suivante :

*Pour chaque composante connexe C du graphe $H_{i,j}$,
tous les sommets de l'ensemble $C \cap N(x)$ ont la même couleur.*

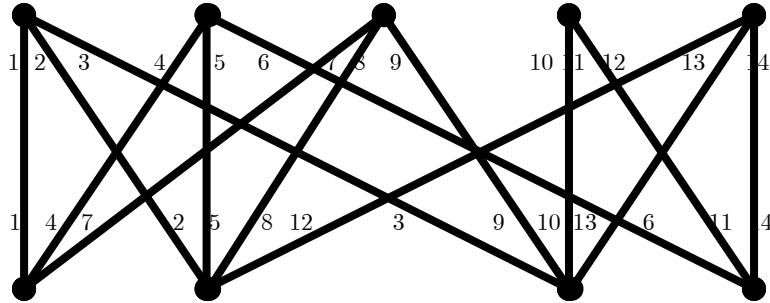
Ceci nous permet d'effectuer l'opération suivante: pour chaque composante connexe C du graphe $H_{i,j}$ qui est telle que, tous les sommets de $C \cap N(x)$ ont la couleur j , échanger les couleurs i et j sur tous les sommets de C ("échange bichromatique"). Après cette opération, on obtient une nouvelle coloration c' de $G - x$ pour laquelle il n'y a plus de voisins de x de couleur j (ceci parce que $C \cap N(x)$ ne contenait pas de sommet de couleur i). On peut donc affecter la couleur j à x . ■

EXEMPLE d'une application de la coloration des graphes.

Sur une plaquette rectangulaire on veut réaliser le schéma électrique suivant:



Les positions des entrées et des sorties sont prédéfinies. Déterminer le nombre minimum de couches de circuits imprimés que l'on doit utiliser.



En numérotant sur les entrées les liens avec les sorties de 1 à 14 on obtient, en lisant sur les sorties les liens avec les entrées, la permutation suivante:

$$\pi = (1, 4, 7, 2, 5, 8, 12, 3, 9, 10, 13, 6, 11, 14).$$

On peut donc construire le *graphe de permutation* dont les sommets sont numérotés de 1 à 14 et où on met une arête $\{i,j\}$ si et seulement si $i < j$ et $\pi_i > \pi_j$.

Une couche sans intersection correspond à une sous-suite croissante qui forme un stable dans le graphe de permutation. Le nombre minimum de couches de circuits imprimés correspond au nombre chromatique de ce graphe. Voici un algorithme de coloration du graphe de permutation. Il partitionne la permutation $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ en un nombre minimum de sous-suites extraites croissantes :

```

i:= 1; q:= 1 ; S1:= {π1} ; d1:= 1 ;
tant que i < n faire
    i:=i+1 ; nouveau := vrai ; k:=1 ;
    tant que (nouveau et k ≤ q) faire
        si πi > πdk alors Sk:=Sk ∪ {πi} ; dk:=i ; nouveau:=faux;
        sinon k:=k+1 ;
    fin si
fin tant que
si nouveau alors q:= q+1 ; Sq:= {πi} ; dq:= i ; fin si
fin tant que

```

COMMENTAIRES : i – l'indice courant de π , q – le nombre courant de sous-suites créées, S_j – $j^{\text{ème}}$ sous-suite croissante, d_j – l'indice du dernier élément de S_j . A l'étape $i > 1$, on suppose que la séquence $(\pi_1, \pi_2, \dots, \pi_{i-1})$ a été partitionnée en q sous-suites extraites croissantes. On range alors π en queue du premier S_j disponible, sinon on crée S_{q+1} .

Dans notre cas cet algorithme fournit: $S_1=\{1, 4, 7, 8, 12, 13, 14\}$, $S_2=\{2, 5, 9, 10, 11\}$, $S_3=\{3, 6\}$. Trois est le nombre minimum de sous-suites extraites croissantes, car l'ensemble $\{6, 10, 12\}$ forme un K_3 .

Il existe toujours une clique K_q qui traverse les S_j ($j=1, 2, \dots, q$), car $\forall j \geq 2$, $\forall x \in S_j$, $\exists y \in S_{j-1}$, tel que $x < y$ et x est examiné après y par l'algorithme (PREUVE: lorsqu'on range x dans S_j , on lui associe $y = \pi_{d_{j-1}}$).

L'algorithme présenté suit la stratégie dite *gloutonne* : on ordonne les sommets, on donne les couleurs et on colorie les sommets dans l'ordre avec la plus petite couleur disponible. Le résultat est exact pour l'ordre naturel pour les graphes de permutation, mais l'ordre "idéal" est très difficile à déterminer pour un graphe quelconque.

3. Connexité des graphes (non orientés)

Un graphe $G=(X, E)$ est dit *connexe* s'il possède la propriété suivante :

$$\forall x, y \in X, x=y \text{ ou } \exists \text{une chaîne entre } x \text{ et } y.$$

Pour un graphe quelconque la relation binaire $\mathcal{R} \subset X \times X$, définie par :

$$x \mathcal{R} y \Leftrightarrow x=y \text{ ou } \exists \text{une chaîne entre } x \text{ et } y$$

est une relation d'équivalence. Le sous-graphe de G , engendré par une classe d'équivalence de cette relation s'appelle *composante connexe* du graphe G . En d'autres mots, un graphe est *connexe* s'il ne possède qu'une seule composante connexe.

La notion du cocycle permet d'étudier la connexité.

Soit S un sous-ensemble de sommets du graphe $G=(X, E)$. On appelle *cocycle* associé à S , l'ensemble $\omega_G(S)$ des arêtes de E ayant exactement une extrémité dans S . Un sous-ensemble U d'arêtes est appelé *cocycle* du graphe, s'il existe une partie S de sommets telle que $U=\omega_G(S)$. Sur la Figure 9 le cocycle associé à $\{1, 2, 3\}$ est $\{14, 34, 35, 25\}$. C'est aussi le cocycle associé à $\{4, 5, 6\}$.

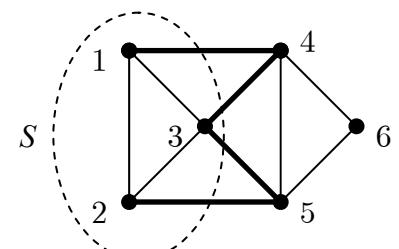


Figure 9.

THEOREME 4 : Un graphe $G=(X, E)$ est connexe si et seulement si, pour tout sous-ensemble de sommets S tel que $\emptyset \neq S \neq X$ on a : $\omega_G(S) \neq \emptyset$.

PREUVE.

\Rightarrow Soit S un sous-ensemble de sommets S tel que $\emptyset \neq S \neq X$; soient $x \in S$ et $y \in X \setminus S$.

Puisque G est connexe il existe une chaîne $\Gamma=(x, e_1, x_1, e_2, \dots, x_{k-1}, e_k, y)$. Soit p le plus grand indice tel que $x_p \in S$. On a: $p \leq k-1$ et $x_{p+1} \in X \setminus S$ donc $e_p = \{x_p, x_{p+1}\} \in \omega_G(S)$.

\Leftarrow Supposons que $G=(X,E)$ n'est pas connexe. Considérons $G_I=(X_I, E)$ une composante connexe de G . L'ensemble $X \setminus X_I$ n'est pas vide et il n'y a aucune arête entre un sommet de X_I et un sommet de $X \setminus X_I$ ce qui veut dire que $\omega_G(X_I) = \emptyset$. ■

On appelle *arbre* un graphe *connexe et sans cycle*. Un graphe *sans cycle* est une forêt.



Figure 10. Un arbre et une forêt de 2 arbres

Tout arbre, d'ordre au moins 2, possède au moins deux sommets pendants (des sommets de degré 1). Il suffit de considérer les extrémités a et b d'une chaîne élémentaire maximale (c'est-à-dire telle que sa prolongation n'est plus possible – voir la figure 11; si $d(b) \neq 1$ alors il existe un cycle).

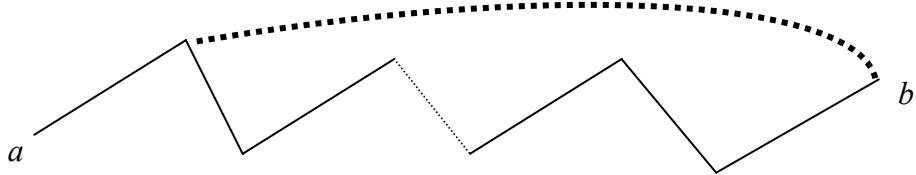


Figure 11.

4. Explorations des graphes

Après avoir modélisé un problème par un graphe on a très souvent besoin d'examiner ses sommets. On appelle *exploration* tout procédé déterministe qui permet de fixer un ordre dans l'examen des sommets de telle façon que l'ensemble des sommets examinés engendre un graphe connexe. L'exploration d'un graphe à partir d'un sommet de départ (appelé *racine* de l'exploration) permet de trouver la composante connexe à laquelle appartient ce sommet et aussi de structurer les sommets d'un graphe connexe. On peut imaginer plusieurs explorations possibles mais nous allons présenter seulement deux, le plus fréquemment utilisées :

*exploration en largeur*² et *exploration en profondeur*³ du graphe.

4.1. Exploration en largeur :

Sa conception est basée sur une partition en couches des sommets d'un graphe connexe. On définit d'abord la distance $d(x,y)$ entre deux sommets x et y , comme la longueur d'une plus courte chaîne entre ces deux sommets. On fixe un sommet initial r , appelé *racine* de l'exploration. La séquence d'ensembles: $C_0 = \{r\}$; $C_1 = \{x \in X \mid d(r,x)=1\}$; ...; $C_k = \{x \in X \mid d(r,x)=k\}$ est appelée la *partition en couches* associée à r .

On visite les sommets en construisant, une par une, les couches. On marque la racine et ensuite les sommets de la première couche, la deuxième etc. En voici une implémentation. Un sommet marqué est dit exploré quand on aura fini de marquer ses voisins. On finit de marquer une couche avant de passer à la couche suivante. On applique la règle: premier marqué – premier exploré (règle bien connue en informatique sous l'abréviation **FIFO** : First In – First Out). Pour gérer ainsi les sommets on peut utiliser un tableau FILE et deux cur-

Dans la bibliographie anglo-saxonne ces méthodes portent respectivement le nom :

² **BFS** Breadth–First Search

³ **DFS** Depth–First Search

seurs : tête et queue. Voici une procédure d'exploration en largeur (avec le graphe représenté par les listes d'adjacence) :

```

Procédure LARGEUR (G: graphe; racine: entier; FILE: tableau [1...n] de entier);

    pour i de 1 à n faire:
        MARQUE[i]:= faux;
    fin pour
    tête:=1; queue:= 1; FILE[tête]:=racine; MARQUE[r]:= vrai;

    tantque tête ≤ queue faire:
        x:= FILE[tête];
        Liste:= liste de voisins de x dans G;

        tantque Liste ≠ nil faire:
            y:= Liste↑.nom;

            si non MARQUE[y] alors
                MARQUE[y]:= vrai;
                queue:= queue +1;
                FILE[queue]:= y;
            fin si

            Liste:= Liste↑.suivant;
        fin tantque

        tête:= tête +1;
    fin tantque
fin LARGEUR

```

↑
initialisation

← boucle principale

```

Procédure PROFONDEUR
(G: graphe; racine: entier; PILE: tableau [1...n] de entier);


```

```

    pour i de 1 à n faire:
        MARQUE[i]:= faux;
    fin pour
    h:=1; PILE[h]:=racine; MARQUE[r]:= vrai;

    tantque h ≥ 1 faire:
        x:= PILE[h];
        si P[y]≠ nil faire:
            y:=P[N[x]];

            si non MARQUE[y] alors
                MARQUE[y]:= vrai;
                h:= h+1;
                PILE[h]:= y;
            fin si

            Mettre à jour P[x]
            (P[x] augmente d'1 ou devient 0);
            sinon
                h:= h-1;
            fin si
        fin tantque
    fin PROFONDEUR

```

← initialisation

← boucle principale

4.2. Exploration en profondeur

Sa conception est basée sur le prolongement d'une chaîne élémentaire d'un graphe connexe. On visite les sommets en partant de la racine r de l'exploration. Lorsqu'on arrive à un sommet x on ne visite pas tous ses voisins, mais un de ses voisins non encore atteints: on cherche à prolonger la chaîne de r à x . Si tous les voisins de x ont été visités, alors on l'abandonne pour examiner le sommet y qui le précède dans la chaîne de r à x et on cherche à prolonger la chaîne de r à y .

Dans l'implémentation proposée ci-dessus on applique la règle: dernier marqué – premier exploré (règle bien connu en informatique sous l'abréviation LIFO: **L**ast **I**n – **F**irst **O**ut). Pour gérer ainsi les sommets on peut utiliser un tableau PILE et un indice hauteur. Une procédure d'exploration en profondeur (avec le graphe représenté par la table d'adjacence) est présentée à la page précédente.

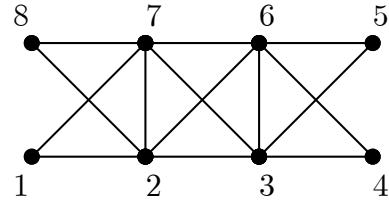
Il est facile de vérifier que dans chaque de ces deux exploration les données occupent $O(n+m)$ cases mémoires et leur exécution nécessite $O(n+m)$ opérations.

Soit le graphe G à explorer avec une représentation par listes d'adjacence :

```

1 : 2 → 7 → nil
2 : 6 → 8 → 1 → 3 → 7 → nil
3 : 4 → 5 → 2 → 6 → 7 → nil
4 : 3 → 6 → nil
5 : 3 → 6 → nil
6 : 7 → 3 → 4 → 5 → 2 → nil
7 : 3 → 6 → 8 → 1 → 2 → nil
8 : 2 → 7 → nil

```



ou bien par la table d'adjacence:

	1	2	3	4	5	6	7	8
P	1	3	8	13	15	17	22	26

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
N	2	7	6	8	1	3	7	4	5	2	3	7	3	6	3	6	7	3	4	5	2	3	6	8	1	2	2	7

Alors, pendant l'exploration en largeur à partir du sommet 8 on visite les sommets dans l'ordre présenté sur la Figure 12.

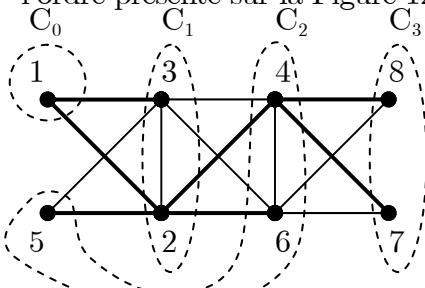


Figure 13.

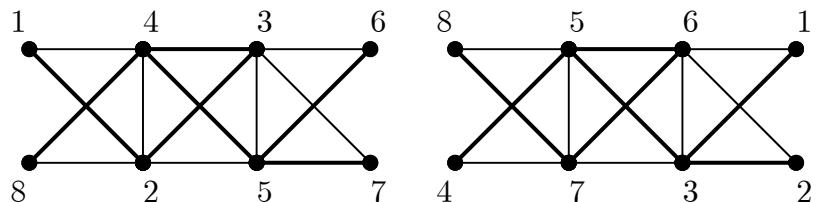


Figure 14.

Pendant l'exploration en profondeur à partir du sommet 8 on marque les sommets dans l'ordre présenté sur la Figure 13 (*numérotation préfixe*). On peut aussi créer une *numérotation postfixe* (ou *suffixe*) le numéro étant attribué à un sommet au moment où la prolongation d'une chaîne se termine (Figure 14).

L'ordre de visites des sommets à l'issue de ces algorithmes dépend de l'ordre de leur apparition dans les listes d'adjacence.

Chacun des deux algorithmes proposés détermine une chaîne élémentaire unique entre la racine et chaque sommet visité. Les graphes partiels qui en résultent possèdent donc une propriété intéressante : ils ne comportent pas de cycle.

5. Arbres

On appelle *arbre* un graphe connexe sans cycle. Une *forêt* est un graphe sans cycle. Evidemment chaque composante connexe d'une forêt est un arbre.

On peut aussi ajouter à un graphe un sommet pendant (de degré 1). Cette opération consiste à choisir un sommet $y \in X$ et construire, à partir du graphe $G = (X, E)$ et $x \notin X$, un nouveau graphe, noté $G+x$, définie par : $G+x = (X \cup \{x\}, E \cup \{x, y\})$. Cette opération peut servir pour construire inductivement les arbres.

THEOREME 5 : La classe \mathbb{A} des arbres est définie inductivement par le schéma suivant:

base: le graphe à un sommet;

règle: Si $T \in \mathbb{A}$ alors $T+x \in \mathbb{A}$

PREUVE. Il est clair que ce schéma engendre un graphe connexe et sans cycle (à la base c'est vrai et par ajout d'un sommet pendant on conserve ces deux propriétés). On démontre par récurrence sur le nombre de sommets que tout arbre peut être construit avec ce schéma: l'arbre avec un sommet est dans la base et dans l'arbre avec n sommets, $n \geq 2$, il suffit d'enlever un sommet pendant et ensuite utiliser l'hypothèse de récurrence. ■

Ce schéma implique que les arbres peuvent aussi être caractérisés en mélangeant les propriétés "topologiques" et "numériques".

THEOREME 6 : Soit $G = (X, E)$ un graphe d'ordre $|X| = n \geq 2$. Les propriétés suivantes sont équivalentes:

- (1) G est connexe et sans cycles;
- (2) G est sans cycles et admet $n-1$ arêtes;
- (3) G est connexe et admet $n-1$ arêtes;

PREUVE. Trivialement (1) \Rightarrow (2) et (1) \Rightarrow (3), car l'égalité $|E|=|X|-1$ est conservée par le schéma de construction des arbres. Pour montrer les implications inverses, on peut utiliser le fait que le graphe G ayant la propriété (2) ou (3) admet un sommet pendant, donc il est généré par le schéma de construction des arbres. En effet, $|E|=|X|-1$ implique qu'il existe un sommet x avec $d(x) \leq 1$; car si $d(x) \geq 2$ pour tous les sommets x , avec $\sum_{x \in X} d(x) = 2|E|$ on aurait $|E| \geq |X|$. Si G est connexe, alors il est sans sommet isolé et avec $n \geq 2$, donc $d(x)=1$. Si G est sans cycle (une forêt) alors il admet au moins une arête (car $|E|=|X|-1 \geq 1$ avec $n \geq 2$); une composante connexe avec au moins une arête est un arbre alors il existe au moins deux sommets pendants. ■

Le résultat suivant explique le lien entre les arbres et la connexité.

THEOREME 7 : Les propriétés suivantes sont équivalentes:

- (1) $G = (X, E)$ est connexe et sans cycles;
- (2) G est connexe-minimal (si on supprime une arête quelconque, il n'est plus connexe);
- (3) G est sans boucle et tout couple de sommets est relié par une chaîne unique.

PREUVE. (1) \Rightarrow (2): soit $e = \{x, y\}$ une arête de E ($x \neq y$ car G est sans cycle). Si $G \setminus e$ est connexe, alors il existe une chaîne entre x et y (qui ne passe donc pas par e) et G contiendrait un cycle passant par e : contradiction.

(2) \Rightarrow (3): supposons qu'il existe deux sommets x et y entre lesquels il y a deux chaînes différentes Γ_1 et Γ_2 ; soit $e = \{a, b\} \in \Gamma_1 \setminus \Gamma_2$; le sous-graphe engendré par l'ensemble d'arêtes $(\Gamma_1 \cup \Gamma_2) \setminus e$ est donc connexe; par conséquent, il existe dans G une chaîne entre a et b ne contenant pas e , ce qui implique que $G \setminus e$ est connexe: contradiction.

(3) \Rightarrow (1): l'existence d'une chaîne garantit la connexité et l'unicité de la chaîne – l'absence de cycle. ■

Les graphes partiels d'un graphe donné sont partiellement ordonnés par l'inclusion des ensembles des arêtes. Le Théorème 7 exprime le fait que les arbres sont des minimaux dans la classe des graphes connexes. Plus précisément, un graphe $G=(X,E)$ est connexe si et seulement si il contient un graphe partiel $A=(X,T)$ qui est un arbre. Un tel arbre est dit *couvertant* (car il couvre tous les sommets du graphe) ou *arbre du graphe G*. Son complémentaire par rapport à G , c'est-à-dire le graphe $K=(X;E \setminus T)$, est appelé *co-arbre associé* à A . Plus généralement, on appelle co-arbre de G tout graphe partiel de G dont le complémentaire est un arbre couvrant de G .

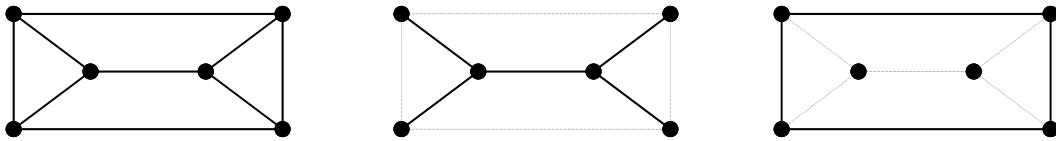


Figure 15. Un graphe G , un arbre couvrant et son co-arbre associé

La notion de co-arbre permet de mieux comprendre le lien entre les cycles et cocycles – deux concepts duals déjà introduits.

THEOREME 7 : Soit $G=(X,E)$ un graphe connexe, alors:

- (1) $A=(X;T)$ est un arbre de $G \Leftrightarrow A$ est sans cycles et maximal pour cette propriété; en ajoutant une arête e de $E \setminus T$, on crée un cycle unique dans $A+e$;
- (2) $K=(X;F)$ est un co-arbre de $G \Leftrightarrow K$ ne contient pas de cocycles de G et maximal pour cette propriété; en ajoutant une arête e de $E \setminus F$, on crée un cocycle unique dans $K+e$;

PREUVE. On va démontrer seulement (2), car le (1) résulte du Théorème 7 .

\Rightarrow Il faut montrer d'abord que F ne contient aucun cocycle de G . Soit un sous-ensemble d'arêtes $U \subseteq E$ et un sous-ensemble de sommets $S \subset X$, tel que $\emptyset \neq S \neq X$ et $U = \omega_G(S)$. On a : $\omega_A(S) \neq \emptyset$, car A est connexe ; $\omega_A(S) \subseteq \omega_G(S)$, car $T \subseteq E$; alors $F \cap \omega_G(S) \neq \emptyset$, donc $U \not\subseteq F$.

\Leftarrow Soit $K=(X;F)$ un graphe partiel de G qui vérifie les deux propriétés annoncées. Il faut montrer que $A=(X;E \setminus F)$ est un arbre de G . Si A n'est pas connexe, il existe S , tel que $\emptyset \neq S \neq X$ et $\omega_A(S) = \emptyset$. Puisque $\omega_G(S) \neq \emptyset$, on aurait $\omega_G(S) \subseteq F$; contradiction. On va montrer que A est connexe-minimal. En effet, s'il existe une arête $f \in F$, telle que $A \setminus f$ est connexe, alors son complément $K+f$ ne contient aucun cocycle – contradiction avec la maximalité de K . En conclusion A est un arbre. ■

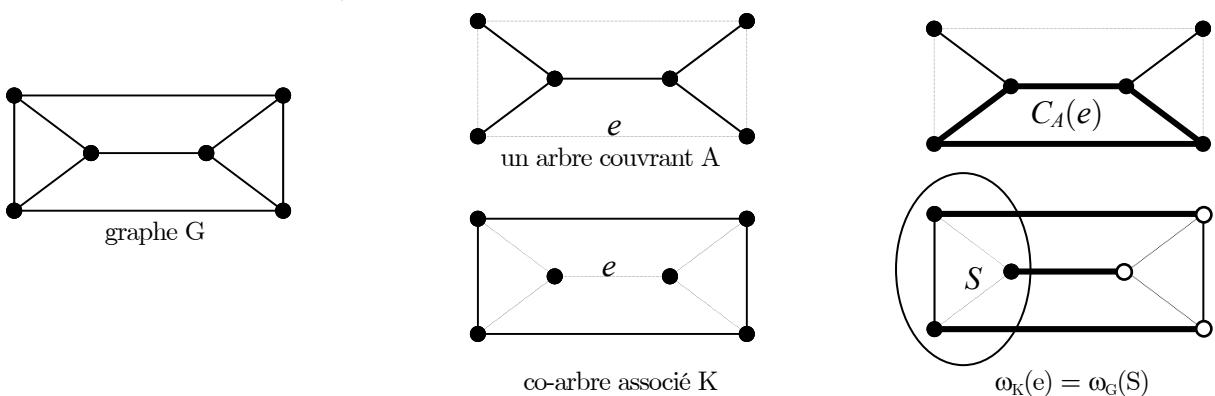


Figure 16.

On va noter $C_A(e)$ – l'unique cycle élémentaire créé dans $A+e$ en ajoutant l'arête e de $E \setminus T$ à l'arbre A (l'existence de ce cycle est mis en évidence par le théorème 7).

Dans un graphe $G=(X,E)$ connexe, un cocycle $\omega_G(S)$ est dit *élémentaire* si les deux sous-graphes, celui engendré par S et celui engendré $X \setminus S$, sont connexes. Le théorème 7 met donc en évidence l'existence de l'unique cocycle élémentaire créé dans $K+e$ en ajoutant l'arête e de $E \setminus F$ au co-arbre K ; on va le noter $\omega_K(e)$.

5.1. L'arbre de poids minimum d'un graphe connexe

Soit $G=(X,E)$ un graphe connexe d'ordre n et $p: E \rightarrow R$ une fonction qui associe à toute arête de G un poids réel. Pour chaque arbre $A=(X,T)$ du G on définit son poids

$$p(A) = \sum_{e \in T} p(e).$$

On veut déterminer un arbre couvrant de G de poids minimum. (Le problème de maximisation peut être résolu par le remplacement de la fonction p par $-p$.) Pour illustrer ce problème, considérons un graphe avec ses arêtes valuées, représenté sur la Figure 17. A côté nous donnons quelques arbres couvrants avec leur poids. En général, il est impossible de réaliser la recherche de l'arbre de poids minimum par énumération explicite, car le graphe complet d'ordre n admet n^{n-2} arbres couvrants (Cayley, 1897). Le besoin d'une méthode efficace semble évident. Nous allons présenter ici deux algorithmes. Leur justification, liée à certaines propriétés remarquables de l'arbre de poids minimum, est présentée plus loin.

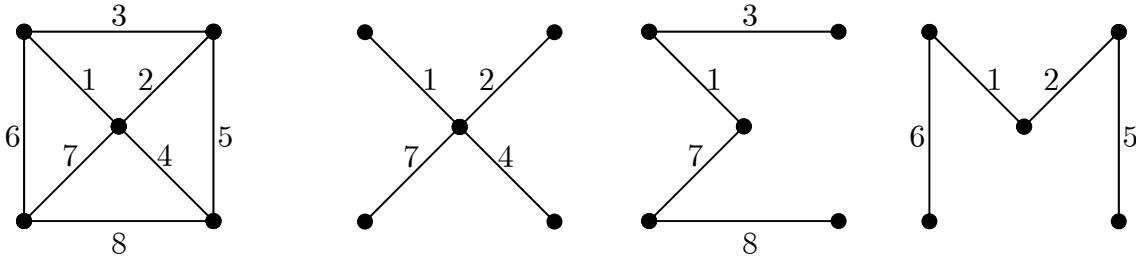


Figure 17. $p(X)=14$ $p(\Sigma)=19$ $p(M)=14$

Les propriétés données par le théorème suivant sont à la base de la solution au problème posé.

THEOREME 9: Soit $A=(X,T)$ un arbre du graphe connexe $G=(X,E)$ et $K=(X;E \setminus T)$ le co-arbre associé à A et $p: E \rightarrow R$ une fonction qui associe à toute arête de G un poids réel. Les propriétés suivantes sont équivalentes:

- (1) $A=(X;T)$ est un arbre de poids minimum;
- (2) $\forall e \in E \setminus T, \forall f \in C_A(e)$ on a: $p(f) \leq p(e)$;
- (3) $\forall e \in T, \forall f \in \omega_K(e)$ on a: $p(f) \geq p(e)$.

Preuve. On va seulement démontrer $(1) \Leftrightarrow (2)$, car la preuve de $(1) \Leftrightarrow (3)$ est similaire.

$(1) \Rightarrow (2)$: Soit $A=(X;T)$ un arbre de poids minimum et $e \in E \setminus T \neq \emptyset$. Pour $f \in C_A(e)$ le graphe $A'=A+f \setminus e$ est un arbre. On a $p(A')=p(A)+p(f) \setminus p(e)$. A est de poids minimum donc $p(A') \geq p(A)$ et cela implique $p(f) \leq p(e)$.

$(2) \Rightarrow (1)$: Soit $B=(X;T)$ un arbre qui vérifie (2). Nous allons montrer que l'arbre de poids minimum $A=(X;T_1)$, qui possède un nombre d'arêtes communes avec l'arbre B (c'est-à-dire $|T \cap T_1|$ est maximum), est identique à B . Supposons le contraire. Soit $e_1=\{x_1, y_1\} \in T_1 \setminus T$. Puisque $e_1 \notin T$, alors T contient un chaîne élémentaire Γ entre x_1 et y_1 . Notons $K=(X, E \setminus T_1)$ le co-arbre associé à $A=(X;T_1)$ dans $G=(X,E)$ et considérons le cocycle $\omega_K(e_1)$. Puisque $\Gamma \cap \omega_K(e_1) \neq \emptyset$, il existe $e_2 \in \Gamma \cap \omega_K(e_1)$. Nous constatons donc, qu'il existe $e_2 \in T \setminus T_1$, tel que $e_2 \in C_B(e_1)$ et, puisque B vérifie (2) par hypothèse, on a: $p(e_2) \leq p(e_1)$. Nous avons aussi: $e_1 \in C_A(e_2)$ et, puisque A , étant de poids minimum, vérifie (2), on a $p(e_1) \leq p(e_2)$. Considérons l'arbre $A'=A+e_2 \setminus e_1=(X;T'_1)$. C'est un arbre de poids minimum, car d'après ce qui précède $p(e_1)=p(e_2)$, ayant une arête commune de plus (car $e_2 \in T \cap T'_1$) avec B – contradiction. ■

ALGORITHME DE KRUSKAL:

- 1°** Trier E pour obtenir la liste (e_1, e_2, \dots, e_m) telle que :
 $p(e_i) \leq p(e_{i+1})$ pour $i=1, 2, \dots, m-1$.
- 2°** Construire la séquence: $T_1=\emptyset$; $T_{i+1}=T_i \cup \{e_k\}$
où $k = \min\{j; T_i+e_j$ est sans cycle}.
- $A_n=(X; T_n)$ est un arbre de poids minimum.

Cet algorithme appliqué au graphe de la Figure 17 fournit l'arbre de poids minimum $p(A_5)=13$ suivant :

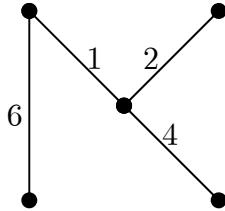


Figure 18.

L'algorithme de Kruskal est remarquable de simplicité dans sa conception. Dans la liste ordonnée, chaque arête est examinée une seule fois. La conception : ordonner et ensuite examiner dans l'ordre, s'appelle *algorithme glouton*.

Bien que l'exécution de l'algorithme de Kruskal se fasse en temps linéaire par rapport à la liste triée des arêtes, néanmoins il faut d'abord trier des arêtes, ce qui coûte $O(mlgm)$ si l'on utilise le meilleur algorithme (tri par fusion). Les graphes construits à chaque étape ne sont pas obligatoirement connexes mais il faut s'assurer qu'aucun cycle n'est créé. En utilisant des structures de données bien adaptées on arrive à atteindre la complexité globale de $O(mlgm)$. Dans le cas de l'algorithme de Kruskal, la préparation des données "dure" donc plus longtemps que l'exécution de la méthode. En règle générale, on améliore sensiblement la vitesse d'exécution d'une procédure par une structure des données adaptée au problème particulier.

Pour éviter de trier une liste de longueur m , ce qui coûte $O(mlgm)$, une autre idée consiste à déterminer seulement le minimum qui coûte seulement $O(m)$.

Algorithme de Prim:

Construire la double séquence:

$T_1=\emptyset$; $S_1=\{x_1\}$ $T_{i+1}=T_i \cup \{e_k\}$; $S_{i+1}=S_i \cup \{x_{i+1}\}$;
où $e=\{x, x_{i+1}\}$ est une arête de poids minimum du cocycle $\omega(S_i)$.
 $A_n=(S_n; T_n)=(X; T_n)$ est un arbre de poids minimum.

Cet algorithme détermine (en partant d'une même liste triée des arêtes d'un graphe connexe arbitraire) le même arbre couvrant que l'algorithme de Kruskal. Voici une implémentation dont la complexité est en $O(n^2)$.

Nous pouvons toujours supposer que le graphe connexe $G=(X, E)$ est simple (on garde seulement une arête de poids minimum s'il existe plusieurs arêtes parallèles et on enlève des boucles) et complet (s'il ne l'est pas on peut simuler l'absence d'une arête en lui affectant un poids très grand – une telle arête ne figure jamais dans un arbre de poids minimum). Les données du problème (le graphe et les poids des arêtes) seront donc parfaitement représentées par une matrice $n \times n$, appelée matrice des poids $G=(g_{ij})$, définie par:

$$g_{ij} = \begin{cases} p(\{ij\}); & \text{si l'arête } \{ij\} \text{ appartient à } E; \\ \infty & \text{pour les boucles et quand l'arête } \{ij\} \text{ n'appartient pas à } E. \end{cases}$$

Le Théorème 9 permet de justifier ces deux algorithmes. A titre d'exemple nous présentons une *justification de l'algorithme de Kruskal* :

$A=(X,T_n)$ est un arbre, car il est sans cycles et contient $n-1$ arêtes. A est de poids maximum car il vérifie la condition (2) du Théorème 9. En effet, soit $e_r \in E \setminus T_n$ et supposons qu'il existe une arête $e_s \in C_A(e_r)$, pour laquelle la condition n'est pas satisfaite, c'est-à-dire $p(e_r) < p(e_s)$. Les e_i étant ordonnées on a: $r < s$. Puisque $e_s \in T_n$, notons i l'étape où elle a été choisie: $e_s \in T_i$ et $e_s \notin T_{i-1}$. Considérons l'arbre $A' = A + e_r \setminus e_s$. On a $T_{i-1} \subseteq A$. Puisque A est sans cycle, e_r ne forme pas de cycle dans T_{i-1} , mais, d'après ce qui précède, l'arête avec le plus petit indice, qui ne forme pas de cycle dans T_{i-1} , est e_s . Donc $s \leq r$ – contradiction. ■

Nous proposons une implémentation de l'algorithme de Prim. Puisque on ajoute un sommet à la liste S et une arête de poids minimum dans $\omega(S)$, nous allons donc utiliser deux tableaux N et P définis par:

$N[i] =$ le numéro du voisin de i dans S relié par une arête de poids minimum du cocycle $\omega(S)$;
 $P[i] =$ le poids de l'arête $\{i, V[i]\}$ (les sommets étant identifiés avec les n premiers entiers positifs)

et on marquera le passage du sommet k dans S en posant $P[k] = \infty$.

```

Procédure PRIM (T: liste d'arêtes);
pour i de 2 à n faire:
    V[i]:= 1;
    P[i]:= P[i,1]           ← initialisation
fin pour
T:= nil;
pour i de 2 à n faire:
    min:= P[i];
    k:= i
    pour j de i+1 à n faire:
        si P[j] < min alors
            min:= P[j];
            k:= j
        fin si
        ajouter l'arête {k,V[k]} à T;
        P[k]:= ∞
    pour j de 2 à n faire:
        si (G[j,k] < P[j] et P[j] < ∞) alors
            P[j]:= P[j,k];
            V[j]:= k;
        fin si
    fin pour
fin pour
fin PRIM

```

6. Connectivité

6.1. Graphes h -connexes

Nous allons considérer dans ce chapitre un graphe simple $G = (X, E)$ connexe, d'ordre n . La connectivité $\kappa(G)$ de ce graphe est le nombre minimum de sommets dont l'élimination disconnecte G ou le réduit à un sommet unique. Si G n'est pas une clique, il existe deux sommets a et b non adjacents, donc $X \setminus \{a, b\}$ est un ensemble dont l'élimination disconnecte G , et par conséquent:

$$\kappa(G) \leq |X - \{a, b\}| = n - 2.$$

Au contraire, si G est la n -clique K_n , on a $\kappa(K_n) = n - 1$.

Un graphe G est dit *h -connexe* si sa connectivité $\kappa(G)$ est $\geq h$. Si G n'est pas une clique, et si \mathcal{A} désigne la famille des *ensembles d'articulation* (c'est-à-dire des ensembles $A \subseteq X$ dont la suppression disconnecte le graphe), G est *h -connexe* si et seulement si :

$$\kappa(G) = \min_{A \in \mathcal{A}} |A| \geq h$$

G est *h -connexe*, signifie donc que l'on a simultanément les deux conditions suivantes:

- 1° $h < n - 1$,
- 2° il n'existe pas d'ensembles d'articulation A avec $|A| < h$.

THEOREME 10 : *Dans un graphe simple connexe G , on a : $\kappa(G) \leq \min_{x \in X} d_G(x)$.*

PREUVE. Si le graphe G a n sommets, on a $\kappa(G) \leq n - 1$; soit x_0 un sommet de degré minimum h , et supposons $h < \kappa(G)$; l'ensemble des voisins de x_0 , $N(x_0)$, est de cardinalité $h < n - 1$, donc sa suppression disconnecte G , donc G n'est pas une clique, et l'on peut écrire :

$$\kappa(G) = \min_{A \in \mathcal{A}} |A| \leq |N(x_0)| = h < \kappa(G). \quad \text{D'où la contradiction.} \blacksquare$$

6.2. Points d'articulation et blocs

Un *point d'articulation* d'un graphe est un sommet dont la suppression augmente le nombre de composantes connexes; un *isthme* est une arête dont la suppression a le même effet. On peut donc redéfinir : *un graphe est 2-connexe si et seulement si il est connexe, d'ordre $n \geq 3$, et n'admet pas de points d'articulation.*

Dans un graphe G , on appelle *bloc* un ensemble A de sommets qui engendre un sous-graphe G_A connexe sans points d'articulation, et maximal avec cette propriété; le sous-graphe G_A est alors soit 2-connexe (si $|A| > 2$), soit un isthme de G (si $|A| = 2$), soit un point isolé de G (si $|A|=1$). On appelle *corde* d'un cycle élémentaire une arête qui relie deux sommets non consécutifs de ce cycle. Un cycle de longueur 3 n'admet donc pas de cordes, alors qu'un cycle de longueur 4 peut avoir 0, 1 ou 2 cordes.

Un *cactus* est un graphe connexe dont chaque bloc est constitué soit par un isthme, soit par un cycle élémentaire sans cordes. Le lecteur vérifiera que le cactus sur la Figure 22 a 4 points d'articulation, 2 isthmes, 5 blocs.

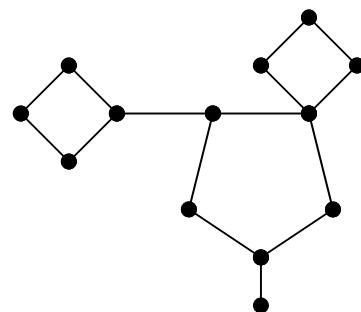


Figure 22. Cactus

Les principales propriétés caractéristiques des graphes 2-connexes sont formulées par l'énoncé suivant :

THEOREME 10 : *Etant donné un graphe G d'ordre ≥ 3 , les propriétés suivantes sont équivalentes :*

- (1) *G est 2-connexe ;*
- (2) *par deux sommets de G , il passe toujours un cycle élémentaire ;*
- (3) *par un sommet et une arête de G , il passe toujours un cycle élémentaire ;*
- (4) *par deux arêtes de G , il passe toujours un cycle élémentaire ;*
- (5) *par deux arêtes de G , il passe toujours un cocycle élémentaire ;*
- (6) *par deux arêtes adjacentes il passe toujours un cycle élémentaire.*

7. Couplages

Rappelons qu'un *couplage* \mathcal{C} est un ensemble d'arêtes deux à deux non adjacentes. Le *couplage* est *parfait* si $\forall x \in X, \exists$ une arête du couplage incidente à x . On dit qu'un sommet x est *saturé* par un couplage \mathcal{C} s'il existe une arête de \mathcal{C} attachée à x . Dans le cas contraire on dit que le sommet x est *insaturé* par \mathcal{C} . Par *couplage maximum* on comprend un couplage de cardinalité maximum.

EXEMPLE 6: *Echiquier tronqué* (voir **Avant propos** et aussi l'**EXEMPLE 4 p.10**).

EXEMPLE 7: La bataille d'Angleterre.

En 1941, les escadrilles anglaises se componaient d'avions biplaces, mais certains pilotes ne pouvaient pas faire équipe dans le même avion par suite de différences de langues ou d'habitudes. Avec ces contraintes, quel est le nombre maximum d'avions que l'on peut utiliser simultanément ?

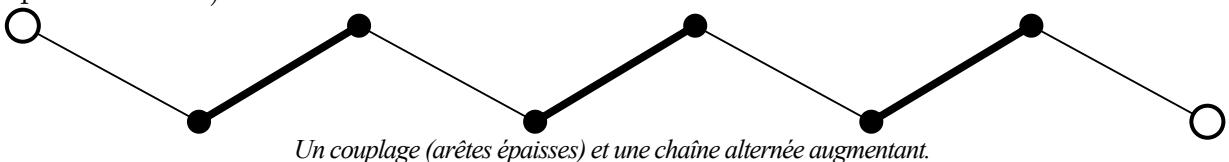
Il s'agit de rechercher le couplage maximum d'un graphe dont les sommets représentent les pilotes, deux sommets étant joints si les pilotes correspondants peuvent faire équipe.

EXEMPLE 8: Problème des collégiens et des collégiennes.

Dans un collège mixte américain, toute jeune fille a k "boy-friends" et tout garçon a k "girl-friends"; est-il possible de faire danser simultanément chaque jeune fille avec un de ses boy-friends et chaque garçon avec une de ses girl-friends.

EXEMPLE 9: Les couples stables. (en Travaux Pratiques)

Considérons un graphe $G=(X;E)$ et un couplage \mathcal{C} de G ; on appelle *chaîne alternée* (relativement au couplage \mathcal{C}) une chaîne simple (c'est-à-dire n'utilisant pas deux fois la même arête) dont les arêtes sont alternativement dans \mathcal{C} et dans $E \setminus \mathcal{C}$ (c'est-à-dire alternativement épaisses et fines).



Une chaîne alternée (relativement au couplage \mathcal{C}) reliant deux points insaturés distincts est dite *augmentante*, car en interchangeant les arêtes fines et les arêtes épaisses le long de cette chaîne, on obtient un nouveau couplage \mathcal{C}' , avec $|\mathcal{C}'|=|\mathcal{C}|+1$.

THEOREME 12 (Berge, 1957): *Un couplage \mathcal{C}_0 est maximum si et seulement si il n'existe pas de chaîne alternée reliant un point insaturé à un autre point insaturé.*

PREUVE:

\Rightarrow En effet, si \mathcal{C}_0 est un couplage qui n'a pas la propriété de l'énoncé, il existe une chaîne alternée reliant deux points insaturés distincts; en interchangeant les arêtes fines et les arêtes épaisses le long de cette chaîne, on obtient un nouveau couplage \mathcal{C}_1 , avec $|\mathcal{C}_1|=|\mathcal{C}_0|+1$; donc le couplage \mathcal{C}_0 n'est pas maximum.

\Leftarrow Soit \mathcal{C}_0 un couplage pour lequel il n'existe pas de chaîne alternée reliant deux points insaturés distincts. Considérons un couplage maximum \mathcal{C}_1 . Il vérifie la condition nécessaire (il n'existe pas de chaîne alternée relative à \mathcal{C}_1 reliant deux points insaturés distincts). Les composantes connexes du graphe partiel qui a pour arêtes $(\mathcal{C}_0 \setminus \mathcal{C}_1) \cup (\mathcal{C}_1 \setminus \mathcal{C}_0)$ sont de trois types :

1. point isolé;

2. cycle élémentaire pair, dont les arêtes appartiennent alternativement aux deux couplages;

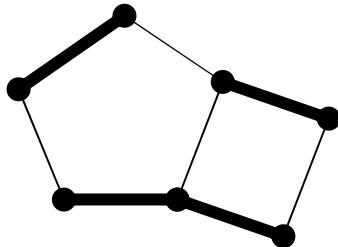
3. chaîne élémentaire, dont les extrémités sont des points distincts insaturés pour un des deux couplages.

Les chaînes élémentaires sont nécessairement paires (elles relient un point saturé avec un point insaturé) donc $|\mathcal{C}_0 \setminus \mathcal{C}_1| = |\mathcal{C}_1 \setminus \mathcal{C}_0|$ ce que implique $|\mathcal{C}_1| = |\mathcal{C}_0|$; donc le couplage \mathcal{C}_0 est maximum. ■

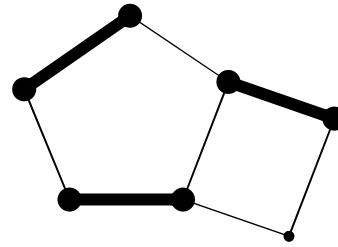
7.1. Le problème du recouvrement minimum

Étant donné un graphe on appelle *recouvrement* un sous-ensemble d'arêtes $F \subseteq E$ tel que tout sommet $x \in X$ soit l'extrémité d'au moins une arête de F .

Par *recouvrement minimum* on comprend un recouvrement de cardinalité minimum (on note ce nombre $\rho(G) = \min \{ |F| ; F \text{ est un recouvrement de } G \}$).



recouvrement minimum



couplage maximum

On dénote par $T \subseteq X$ un ensemble *transversal* c'est-à-dire un ensemble de sommets tel que toute arête de G ait au moins une de ses extrémités dans T . Le nombre minimum de sommets d'un transversal désigné par $\tau(G)$ s'appelle le *nombre de transversalité*.

EXERCICES :

Démontrer les relations suivantes valables pour tout graphe $G=(X;E)$ ($n=|X|$) :

- 1) $\alpha(G) + \tau(G) = n$
- 2) $v(G) + \rho(G) = n$ (si G n'a pas de points isolés)
- 3) $v(G) \leq \tau(G)$
- 4) a) Un recouvrement minimal est minimum si et seulement si il contient un couplage maximum.
b) Un couplage maximal est maximum si et seulement si il est contenu dans un recouvrement minimum.

THEOREME 13: Dans un graphe $G=(X, E)$ un couplage maximum C et un recouvrement minimum F vérifient : $|C|+|F|=|X|$. Étant donné un couplage maximum C on obtient un recouvrement minimum F_1 , en ajoutant pour tout sommet y insaturé dans le couplage C , une arête e de G incidente à y . Étant donné un recouvrement minimum F on obtient un couplage maximum C_1 , en retirant de proche en proche une arête e de F encore adjacente à une arête de F non encore retirée.

PREUVE:

En effet si C est un couplage maximum, d'après la construction proposée, F_1 est un recouvrement et l'on a : $|F_1|=|C|+(|X|-2|C|)=|X|-|C|$.

D'autre part, si F est un recouvrement minimum, d'après la construction proposée, C_1 est un couplage et chaque arête retirée crée exactement un point insaturé pour C_1 , d'où : $|F|-|C_1|=|X|-2|C_1|$ ou $|F|=|X|-|C_1|$.

Comme $|C_1|\leq|C|$, on a $|F_1|=|X|-|C|\leq|X|-|C_1|=|F|$, donc F_1 est un recouvrement minimum et $|F_1|=|F|$. Ainsi $|C_1|=|C|$ et, par conséquent, le couplage C_1 est un couplage maximum. Entre autre, on a bien $|C|+|F|=|X|$. ■

7.2. Couplages dans un graphe biparti.

THEOREME (König 1931) : Dans un graphe biparti $G=(X, Y; E)$ on a : $\nu(G)=\tau(G)$.

Avant de présenter une démonstration, voici un algorithme pour la recherche du couplage maximum dans un graphe biparti $G=(X, Y; E)$, appelé couramment *méthode hongroise*.

Soit \mathcal{C}_0 un couplage initial (éventuellement \emptyset).

- 1 Marquer * tous les sommets de X insaturés dans \mathcal{C}_0 . Pour chaque sommet ainsi marqué appliquer alternativement **2, 3, 2, 3, 2, 3, 2, 3...**
- 2 Choisir un sommet de X qui vient d'être marqué (mettons x_i) et marquer " x_i " tous les sommets de Y encore non marqués et relié avec x_i par les arêtes qui n'appartiennent pas au couplage \mathcal{C}_0 .
- 3 Choisir un sommet de Y qui vient d'être marqué dans **2** (mettons y_j) et marquer " y_j " tous les sommets de X encore non marqués et relié avec y_j par les arêtes qui appartiennent au couplage \mathcal{C}_0 .

Il s'agit évidemment d'une implémentation particulière de l'exploration en largeur. Effectuons ce marquage jusqu'à ce que l'on ne puisse plus marquer un nouveau sommet. Alors on se retrouve dans une des deux situations:

Fin1. Un sommet de Y insaturé dans \mathcal{C}_0 est marqué.

Dans ce cas il existe une chaîne alternée reliant deux points insaturés distincts et nous pouvons, en interchangeant les arêtes fines et les arêtes épaisses le long de cette chaîne, obtenir un nouveau couplage \mathcal{C}_1 , avec $|\mathcal{C}_1|=|\mathcal{C}_0|+1$. Ensuite on recommence le marquage.

Fin2. Il n'est plus possible de continuer le marquage et **Fin1** n'a pas lieu.

Dans ce cas le couplage \mathcal{C}_0 est maximum. Pour le justifier nous donnons la

PREUVE algorithmique du **THEOREME de König** :

Supposons que l'algorithme précédent fini en **Fin2** en partant d'un couplage \mathcal{C} . Notons par N_X les sommets non marqués dans X et par M_Y les sommets marqués dans Y . Nous allons prouver que :

- a) $T=N_X \cup M_Y$ est un transversal (c'est-à-dire un ensemble de sommets tel que toute arête de G ait au moins une de ses extrémités dans T); b) $|\mathcal{C}|=|T|$; c) couplage \mathcal{C} est maximum.

a) Si $T = N_X \cup M_Y$ n'est pas un transversal, alors il existe une arête $e = \{x, y\}$ telle que $x \notin N_X$ et $y \notin M_Y$. Si $e \in \mathcal{C}$ alors puisque x est marqué on peut marquer y en appliquant **2** – contradiction. Si $e \notin \mathcal{C}$ alors puisque y est marqué on peut marquer x à partir de y en appliquant **3** – contradiction.

b) Soit $y \in M_Y$. Puisque **Fin1** n'a pas lieu, alors y est incident à exactement une arête de \mathcal{C} (\mathcal{C} est un couplage!). Notons cette arête $e = \{x, y\}$. D'après **3**, x est marqué ($x \notin N_X$). Considérons un sommet $x' \in N_X$. Puisque x' n'est pas marqué, alors il est incident à exactement une arête de \mathcal{C} (sinon il devrait porter la marque *), disons $e' = \{x', y'\}$, et dans ce cas y' n'est pas marqué (car sinon, on peut marquer x'), alors aucune arête de \mathcal{C} ne peut coupler un sommet de N_X avec un sommet de M_Y . Chaque arête de \mathcal{C} a une de ses extrémités dans N_X ou dans M_Y alors $|\mathcal{C}| = |T|$ donc le couplage est maximum. ■

EXEMPLE 10: Appliquons la méthode hongroise pour trouver un couplage maximum dans le graphe ci-contre :

On commence, par exemple, avec un couplage maximal

$$\mathcal{C}_0 = \{(x_2, y_2), (x_3, y_3), (x_4, y_4)\}.$$

1^{ère} itération:

Pas 1.: marquer x_1 et x_5 avec *.

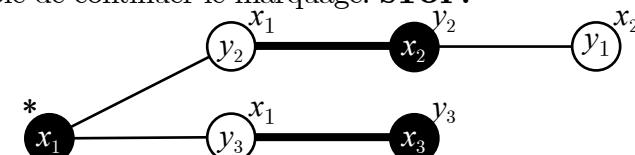
Pas 2.: choisir x_1 et marquer y_2 et y_3 avec x_1 .

Pas 3.: choisir y_2 et marquer x_2 avec y_2 .

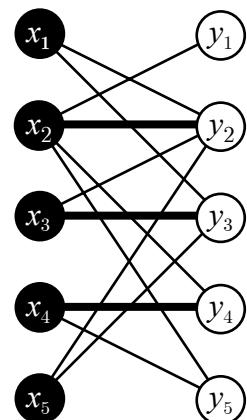
choisir y_3 et marquer x_3 avec y_3 .

Pas 4.: choisir x_2 et marquer y_1 avec x_2 .

Il n'est plus possible de continuer le marquage. **STOP!**



L'arbre hongrois enraciné en x_2 à l'issue de la 1^{ère} itération



2^{ème} itération:

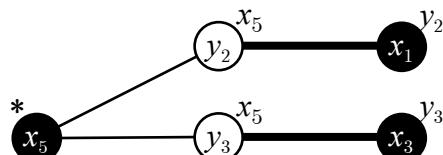
Pas 1.: marquer x_5 avec *.

Pas 2.: choisir x_5 et marquer y_2 et y_3 avec x_5 .

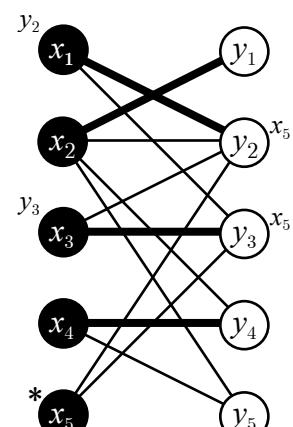
Pas 3.: choisir y_2 et marquer x_1 avec y_2 ;

choisir y_3 et marquer x_3 avec y_3 .

Il n'est plus possible de continuer le marquage. **STOP!**



L'arbre hongrois enraciné en x_5 à l'issue de la 2^{ème} itération.



Le couplage \mathcal{C}_1 est donc maximum (non parfait). Les sommets noirs non marqués et les sommets blancs marqués à la dernière itération forment un ensemble transversal minimum $T = \{x_2, x_4, y_2, y_3\}$ de cardinalité 4.

Il existe aussi une version “valuée” de cet algorithme pour rechercher un couplage parfait de poids minimum (avec les arêtes valuées).

Si le graphe biparti $G=(X,Y;E)$ admet un couplage saturant tous les sommets de X , on dit que l'on peut *coupler X dans Y*. Pour $A \subseteq X \cup Y$ on dénote par $N(A)$ l'ensemble des sommets du graphe adjacents à l'ensemble A .

THEOREME de König-Hall: *Dans un graphe biparti $G=(X,Y;E)$ on peut coupler X dans Y si et seulement si on a : $|N(A)| \geq |A|$ pour tout $A \subseteq X$.*

PREUVE : La condition est trivialement nécessaire. Pour démontrer qu'elle est suffisante on va utiliser le théorème de König. Il suffit de prouver que pour chaque transversal T on a $|T| \geq |X|$. En effet, $|T|=|T \cap X|+|T \cap Y| \geq |T \cap X|+|N(X \setminus T)| \geq |T \cap X|+|X \setminus T|=|X|$. ■

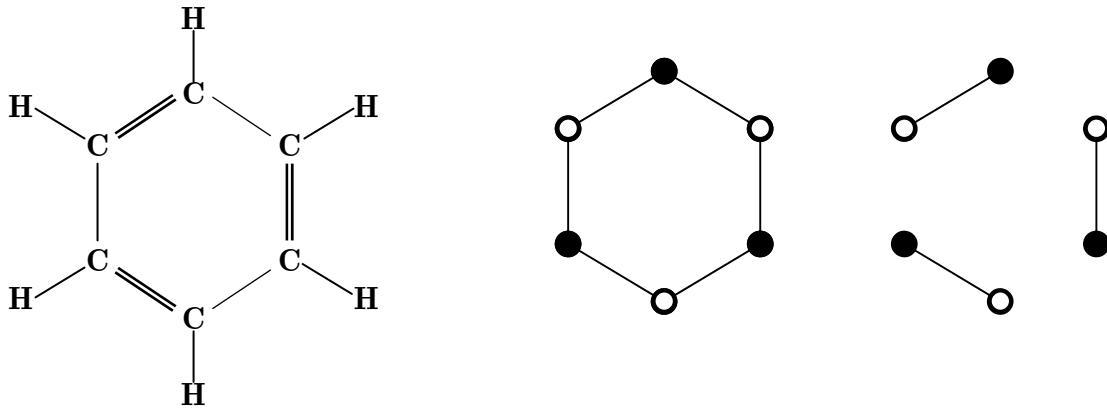
On montre facilement l'équivalence du théorème de König-Hall avec celui de König.

Le théorème de König implique aussi : $\alpha(G)=\rho(G)$ (remarque: si T est un ensemble transversal alors son complémentaire $S=(X \cup Y) \setminus T$ est un stable et vice versa).

On peut appliquer le théorème de König-Hall pour résoudre le problème des collégiens et des collégiennes. Soit $G=(X,Y;E)$ un graphe biparti qui représente la relation d'amitié entre les garçons et les filles : on a $|X|=|Y|$, car $k|X|=k|Y|=|E|$. Puisque $A \subseteq N(N(A))$, alors $k|N(A)| \geq k|A|$ d'où $|N(A)| \geq |A|$. Le théorème de König-Hall implique donc une réponse positive à la question posée.

7.3. Application à la chimie.

Les molécules chimiques sont traditionnellement représentées par les graphes. Voici un exemple bien connu de benzène C_6H_6 :



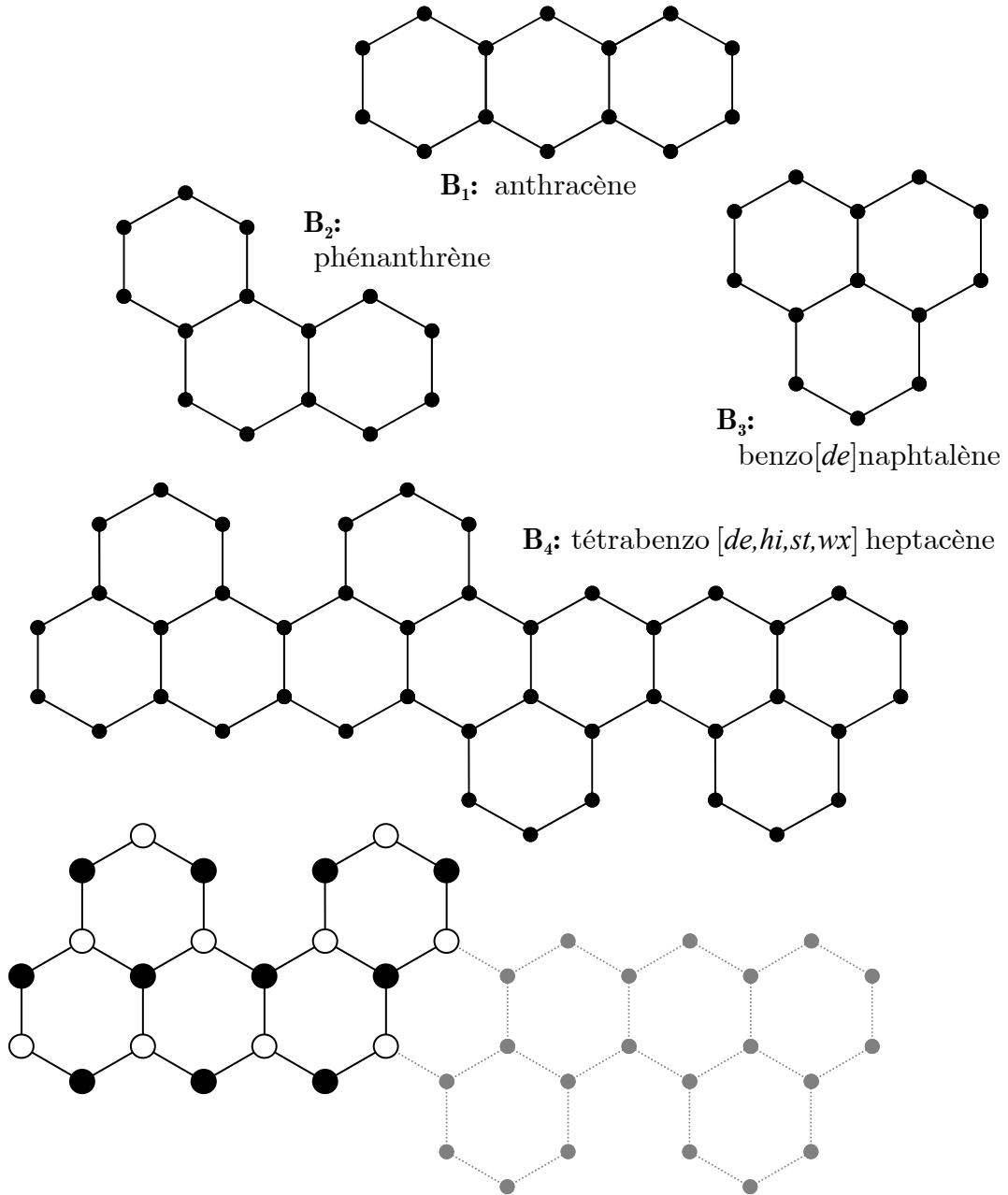
En chimie organique, cette représentation topologique des molécules est parfois encore plus simplifiée. Les atomes d'hydrogène sont ignorés et on s'intéresse seulement aux fragments des molécules représentables par des motifs où les liens simples et doubles alternent. Tous ces graphes sont évidemment bipartis car on peut, en suivant les arêtes pour les déplacements, affecter alternativement à leurs sommets les couleurs: noire et blanche, sans contradiction puisque, par hypothèse, les liens simples et doubles alternent.

Les graphes simplifiés représentant des molécules chimiques s'appellent *graphes de Hückel*. Sur le dessin nous avons illustré le graphe de Hückel associé au benzène et un couplage parfait de ce graphe.

Est-ce qu'un graphe quelconque pourrait représenter un composant chimique existant ou pouvant être synthétisé? La *théorie de résonance* évoque deux postulats :

- (1) certaines classes de composants chimiques peuvent être réellement synthétisées seulement si leur graphes de Hückel admettent un couplage parfait.
- (2) la stabilité d'un composant chimique dépend du nombre de couplages parfaits.

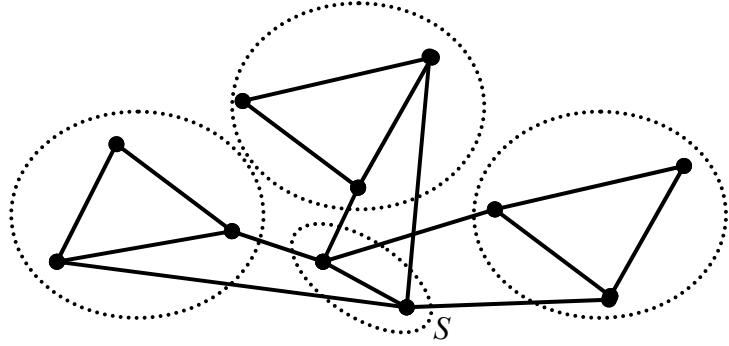
La figure ci-dessous représente quatre exemples de graphes de la famille des benzènoïdes (deux hexagones sont disjoints ou possèdent une arête commune). Par énumération explicite on constate que B_1 admet 4 couplages parfaits et B_2 admet 5 couplages parfaits et B_3 et B_4 aucun. Autrement dit, B_2 sera plus stable que B_1 mais on peut synthétiser les deux. Par contre il est impossible d'obtenir B_3 et B_4 . Pourquoi B_3 et B_4 n'admettent-ils aucun couplage parfait ? Voici les arguments combinatoires. Le cas de B_3 est simple: il possède un nombre impair de sommets (13). Dans le cas de B_4 , la coloration donne toujours 21 sommets blancs et 21 sommets noirs mais le couplage parfait n'existe pas car l'ensemble des 11 sommets noirs devrait être couplé avec l'ensemble des 10 sommets blancs ce qui est impossible (théorème de König–Hall).



Si $G=(X;E)$ est un graphe arbitraire, pas nécessairement biparti, il existe le résultat suivant:

THEOREME 14 (Tutte, 1947): *G admet un couplage parfait si et seulement si pour tout sous-ensemble de sommets $S \subseteq X$ le nombre de composantes connexes impaires de $G \setminus S$ est $\leq |S|$.* ■

L'exemple ci-contre montre que la condition est nécessaire. La démonstration qu'elle est aussi suffisante n'est pas prévue dans ce cours.



8. Orientation d'un graphe

On obtient un graphe orienté $G=(X;A)$ lorsque, pour toute arête $e \in E$ d'un graphe non orienté $G=(X;E)$ ses extrémités sont ordonnées. Chaque arête se transforme en un *arc* qui est un couple de sommets (initial et terminal). Si $a=(x,y)$ est un arc, on dit que “*y est un successeur de x*” et “*x est un prédécesseur de y*”.

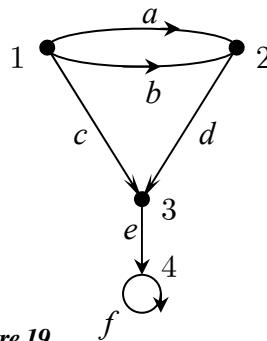


Figure 19.

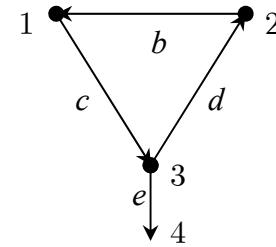


Figure 20.

On note $N^+(x)$ ($N^-(x)$) l'ensemble des successeurs (prédécesseurs) de x .
On a : $N^+(x) \cup N^-(x) = N(x)$.

L'élément a_{xy} de la matrice d'adjacence $\mathbf{A}(G)=(a_{xy})$ d'un graphe orienté représente le nombre d'arcs d'extrémité initiale x et d'extrémité terminale y . L'élément b_{xa} de la matrice d'incidence $\mathbf{B}(G)=(b_{xa})$ d'un graphe orienté est définie par :

$$b_{xa} = \begin{cases} +1 & \text{si } x \text{ est l'extrémité initiale de l'arc } a ; \\ -1 & \text{si } x \text{ est l'extrémité terminale de l'arc } a ; \\ 0 & \text{si l'arc } a \text{ est une boucle et les autres cas.} \end{cases}$$

Cette représentation est significative seulement si le graphe est sans boucle, car elle ne permet pas d'identifier la “position” d'une boucle.

Voici ces deux matrices pour le graphe de la figure 19:

	1	2	3	4
1	0	2	1	0
2	0	0	1	0
3	0	0	0	1
4	0	0	0	1

$\mathbf{A}(G):$

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
1	1	1	1	0	0	0
2	-1	-1	0	1	0	0
3	0	0	-1	-1	1	0
4	0	0	0	0	-1	0

$\mathbf{B}(G):$

La matrice d'adjacence n'est plus symétrique, comme c'était le cas pour des graphes non orientés. La somme des éléments d'une colonne de la matrice d'incidence est 0. Les éléments $b_{xe} \in \{-1; 0; +1\}$ (ceci est vrai aussi pour le déterminant de chaque sous-matrice carrée extraite de $\mathbf{B}(G)$; une telle matrice est dite *totalelement unimodulaire*).

On note $d^-(x)$ (resp. $d^+(x)$) le *degré rentrant* (resp. *sortant*), c'est-à-dire le nombre d'arcs ayant x comme extrémité terminale (resp. initiale).

Sur la figure 19 $d^+(2)=1$ et $d^-(2)=2$. On a toujours: $d^+(x)+d^-(x)=d(x)$. On appelle une *source* un sommet s avec $d^-(s)=0$. Un *puits* est un sommet p avec $d^+(p)=0$.

THEOREME 2: Pour tout graphe orienté $G=(X;E)$ on a: $\sum_{x \in X} d^+(x) = \sum_{x \in X} d^-(x) = |E|$.

PREUVE. Considérons la matrice d'incidence $B(G)=(b_{xe})$. Si le graphe est sans boucle, la somme de tous les éléments de la matrice B est nulle car chaque colonne e contient un +1 et un -1. La somme d'éléments +1 (resp. -1) de chaque ligne x est le nombre d'arcs ayant x comme extrémité initiale (resp. terminal), donc égale à $d^+(x)$ (resp. $-d^-(x)$). On obtient donc : $\sum_{x \in X} d^+(x) - \sum_{x \in X} d^-(x) = 0$ et $\sum_{x \in X} d^+(x) + \sum_{x \in X} d^-(x) = \sum_{x \in X} d(x) = 2|E|$ d'où $\sum_{x \in X} d^+(x) = \sum_{x \in X} d^-(x) = |E|$. Si G possède k boucles en x , alors chacune des valeurs $d^+(x)$, $d^-(x)$ et $|E|$ augment de k , donc double égalité reste toujours valide. ■

9. Représentations des graphes en machine

Avant de programmer un algorithme, pour confier à l'ordinateur la réalisation des calculs, il faut choisir une bonne représentation du graphe dans la machine, pour éviter un encombrement inutile de la mémoire et pour accélérer la vitesse d'exécution de l'algorithme. Dans la pratique la mémoire et le temps de calculs sont limités, il est donc souhaitable de fournir à chaque méthode de solution deux certificats :

- une estimation d'encombrement de la mémoire par les données en fonction de certains paramètres ;
- une estimation du nombre d'itérations nécessaires pour que l'algorithme résolve le plus mauvais cas (complexité de l'algorithme, en fonction de la taille des données).

Comme il est toujours très difficile de donner ces deux nombres avec exactitude (de très nombreux détails entrent en considération), on se contente de les exprimer en termes de l'*ordre de grandeur asymptotique*. Rappelons que la notation $g(n)=O(f(n))$ (prononcer : $g(n)$ égale grand O de $f(n)$) signifie, qu'il existe une constante $c > 0$ telle que pour n assez grand on a : $|g(n)| \leq c|f(n)|$. On dit aussi, que g est *asymptotiquement dominée par* f . Cette estimation nous fournit suffisamment d'informations dans les problèmes d'évaluation.

Soit $G=(X;E)$ un graphe avec n sommets (numérotés 1, 2, ..., n) et m arcs. Nous considérons ici les graphes orientés – le cas non orienté s'en déduit facilement. On peut distinguer les représentations *tabulaires* et les représentations par *listes chaînées*.

Concernant le premier type de représentation, il faut d'abord citer les deux matrices : d'adjacence et d'incidence du graphe. Malheureusement elles sont encombrantes ; elles demandent $O(n^2)$ et $O(nm)$ cases mémoires respectivement.

On peut aussi énumérer l'ensemble des arcs en utilisant deux tableaux-vecteurs, où la case i de chaque tableau mémorise respectivement le sommet initial et le sommet terminal de l'arc a_i , comme l'illustre la Figure 21.

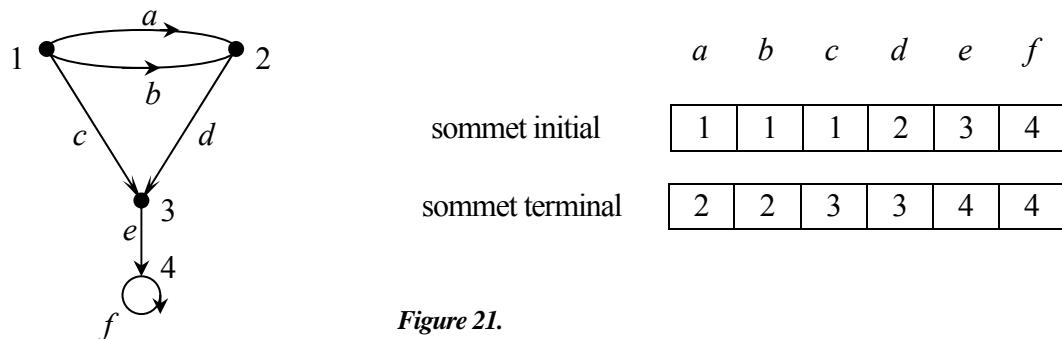


Figure 21.

Cette représentation nécessite $O(m)$ cases mémoires.

Une troisième méthode place dans la *table d'adjacence* (N sur la Figure 22) les listes des extrémités terminales des différents arcs sortants des sommets consécutifs. Pour repérer le premier élément de chaque liste on utilise un deuxième tableau (P sur la Figure 22), qui à la position i donne l'adresse du début de la liste des successeurs du sommet i .

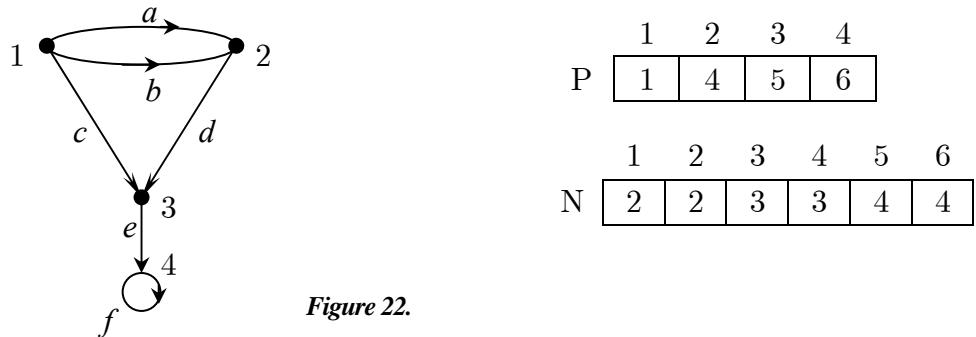


Figure 22.

Sur la figure 22, la liste des successeurs du sommet 2 commence à l'adresse 4 du deuxième tableau. L'encombrement en espace est $O(n+m)$.

Les langages de programmation plus modernes qui comportent des pointeurs, permettent de présenter le graphe par *listes d'adjacence chaînées*. On procède comme pour la table d'adjacence avec la différence que les listes ne sont pas placées, l'une après l'autre, dans un tableau fixe mais le graphe est représenté par un tableau de pointeurs.

Pour le graphe de la figure précédente on obtient :

```

1 : 2 → 2 → 3 → nil
2 : 3 → nil
3 : 4 → nil
4 : 4 → nil
5 : nil ...

```

Cette représentation nécessite aussi $O(n+m)$ cases mémoires, mais contrairement à la table d'adjacence, elle permet de réaliser très facilement les transformations graphiques élémentaires telles que la suppression ou l'adjonction de sommets ou d'arcs.

A titre d'exemple, si le graphe est représenté par listes des successeurs $G[1,2,\dots,n]$ alors on peut obtenir les listes des prédécesseurs $P[1,2,\dots,n]$, en utilisant l'algorithme suivant :

```

Algorithme Prédécesseurs (G[1,2,...,n], P[1,2,...,n]);
pour i de 1 à n faire
    P[i] := nil
fin pour

pour i de 1 à n faire
    pour s rencontré dans G[i] faire
        ajouter i à P[s]
    fin pour
fin pour
fin Prédécessseurs

```

La complexité de cet algorithme est $O(n+m)$, car la première boucle se réalise en $O(n)$ et les deux dernières boucles coûtent $O(m)$, puisque l'adjonction d'un élément à une liste est indépendante de la longueur de celle-ci, donc $O(1)$.

10. Connexité dans des graphes orientés

On peut, sans aucune difficulté, adapter aux graphes orientés les deux procédures d'exploration, étudiées précédemment dans le cas non orienté. Mais l'orientation permet de définir des notions de cheminement plus fines. Le Tableau 1 présente certains objets “non orientés” avec leur correspondant “orienté”:

Tableau 1. Quelques objets orientés et non orientés

graphe non orienté		graphe orienté
	arête	
	chaîne	
	cycle	
	arbre	

Un *chemin* de longueur k est une séquence alternée de sommets et d'arcs $\Gamma=(x_0, e_1, x_1, e_2, \dots, x_{k-1}, e_k, x_k)$ telle que, pour tout i , e_i admet x_{i-1} comme sommet initial et x_i comme extrémité terminale. Lorsque $k > 0$ et $x_0 = x_k$ Γ est un *chemin fermé*. Si e_i sont distincts Γ est un *chemin simple*. Si les x_i sont distincts Γ est un *chemin élémentaire*. On appelle *circuit* tout chemin fermé simple. Les notions de chaîne et de cycle s'entendent au cas orienté, si l'on ne précise plus l'orientation – par exemple dans la définition d'une chaîne on considère que l'arc e_i est incident à x_{i-1} et x_i .

Dans le paragraphe 1 (p.12) nous avons évoqué le problème d'existence d'un cycle eulérien dans un graphe non orienté. Nous allons maintenant illustrer sur cet exemple comment un problème dans un graphe non orienté introduit naturellement une orientation.

THEOREME d'Euler (1766). *Un graphe (non orienté) connexe admet un cycle eulérien si et seulement si tous ses sommets sont de degré pair.*

COROLLAIRE. *Un graphe (non orienté) connexe admet une chaîne eulérienne si et seulement si le nombre des sommets de degré impair est 0 ou 2.*

Preuve. Sans perdre en généralité, on peut supposer que le graphe G est sans boucle. Les conditions sont évidemment nécessaires car dans un cycle eulérien toutes les arêtes sont différentes ; chaque fois quand on traverse un sommet x arbitraire, on entre par une arête et on en sort par une autre, alors son degré est pair.

Pour montrer que la condition est suffisante remarquons d'abord que dans un graphe connexe (avec au moins deux sommets) il existe au moins une arête (car il n'y a pas de sommet isolé) et la condition $d(x) \geq 2$ implique qu'il existe un cycle. Pour le constater, considérons une chaîne élémentaire maximale (c'est-à-dire telle que sa prolongation n'est plus possible) et son extrémité terminale b .

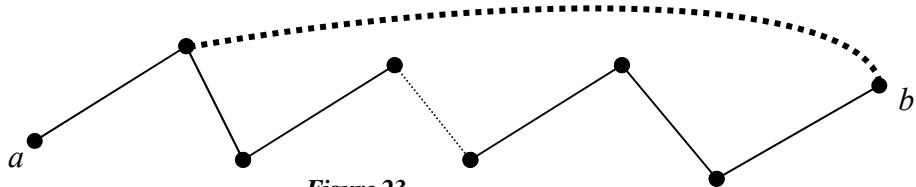


Figure 23.

Il existe au moins deux arêtes incidentes à b et, puisque la prolongation n'est plus possible, alors l'autre extrémité appartient à la chaîne – on vient de détecter un cycle. Pour prouver l'existence d'un cycle de longueur $m=|E|$, raisonnons par induction. Pour $m=1$ cette propriété est trivialement vraie. Soit $m>1$ et supposons la propriété vraie pour tout $k < m$. Dans le graphe $G=(X;E)$ avec $m=|E|$, considérons un cycle C qui comporte un nombre maximum

d'arcs. Montrons que ce cycle comporte toutes les arêtes du graphe. En effet, sinon, il existe une arête e , qui n'appartient pas au cycle C et dont l'une des extrémités est dans C (car G est connexe). Considérons maintenant un graphe partiel F de G , composé de toutes les arêtes qui n'appartiennent pas au cycle C . Les sommets de ce graphe ont les degrés pairs. D'après l'hypothèse de récurrence, la composante connexe de ce graphe partiel à laquelle appartient l'arête e , admet le cycle eulérien C' . Puisque les deux cycles possèdent au moins un sommet commun, alors on peut les réunir pour obtenir un nouveau cycle qui comporte un nombre plus grand d'arêtes que C – contradiction avec le choix du C . ■

Le théorème annonce seulement une condition de l'existence d'un cycle eulérien mais sa démonstration présentée ici offre une méthode constructive (la récurrence) pour le déterminer. La figure 29 représente un graphe dont tous les sommets sont de degré pair. (Puisque ce graphe est simple on pourra noter des cycles uniquement par la suite des sommets). Supposons que, dans un premier temps, on arrive seulement à déterminer un cycle quelconque en oubliant quelques arêtes, par exemple 1254361. Le graphe partiel composé des arêtes oubliées est connexe et on retrouve son cycle eulérien 23562. En réunissant ces deux cycles en leur premier sommet commun (ici 2) on obtient le cycle eulérien du graphe : 12356254361.

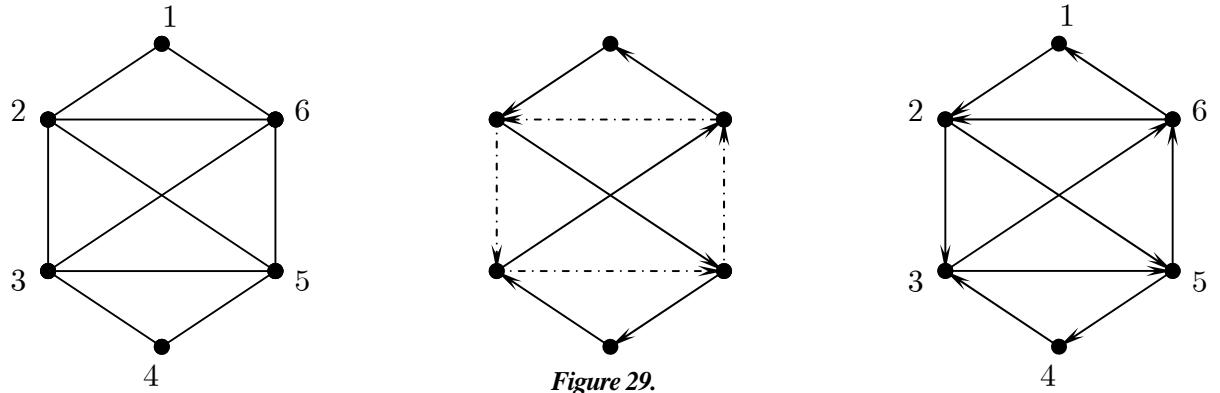


Figure 29.

Du théorème d'Euler on peut aussi déduire l'existence d'une orientation particulière. Un graphe, non orienté, dont les degrés sont pairs admet une orientation telle que, pour tout sommet x , on a $d^+(x) = d^-(x)$.

Quand une orientation est donnée, on a :

THEOREME 16 : *Un graphe orienté $G=(X;E)$ connexe admet un circuit eulérien si et seulement si : pour tout sommet $x \in X$, on a $d^+(x) = d^-(x)$.*

On dira que x peut atteindre y (ou que y peut être atteint à partir de x) s'il existe dans G un chemin de x à y . Ce chemin peut être de longueur zéro, c'est-à-dire que l'on considère que chaque sommet peut s'atteindre lui-même; donc la relation "x peut atteindre y" est considérée comme réflexive. Cette relation est aussi transitive ; voir le lemme ci-dessous. En général il existe des graphes dans lesquels cette relation n'est pas symétrique.

LEMME 2 : *Soit $G=(X;E)$ un graphe orienté. Si x, y, z sont des sommets tels que x peut atteindre y et y peut atteindre z , alors x peut atteindre z .*

LEMME 3 : *Dans un graphe orienté $G=(X;E)$ les deux assertions suivantes sont équivalentes :*

- (1) *Il existe deux sommets distincts x, y tels que x peut atteindre y et y peut atteindre x .*
- (2) *Le graphe G contient un circuit.*

EXERCICE. Démontrer ces deux lemmes.

10.1. Graphes sans circuit

Les graphes sans circuit possèdent une structure spéciale qu'il est facile de déterminer. On rappelle que, si x et y sont deux sommets distincts dans un graphe sans circuit, alors au plus une (éventuellement aucune) des assertions “ x peut atteindre y ” et “ y peut atteindre x ” est vraie (voir le lemme ci-dessus).

Dans un graphe orienté G , un *puits* est un sommet sans successeur, et une *source* est un sommet sans prédécesseur. Un graphe ne contient pas forcément un puits ou une source.

THEOREME fondamental des graphes sans circuit :

Tout graphe orienté sans circuit, avec un nombre de sommets fini, contient un sommet puits ($d^+(x)=0$) et un sommet source ($d^-(x)=0$).

PREUVE. Si x , un sommet quelconque, n'est pas un puits, alors il existe un arc (x,y) avec $y \neq x$, car le graphe est sans circuit, donc sans boucle. Considérons un chemin élémentaire Γ maximal, c'est-à-dire que l'on ne peut plus augmenter sa longueur. Si z est le sommet terminal de ce chemin alors z est un puits. En effet, dans le cas contraire on peut augmenter la longueur du chemin (si le successeur de $z \notin \Gamma$) ou on ferme un circuit (si le successeur de $z \in \Gamma$) – contradiction dans les deux cas.

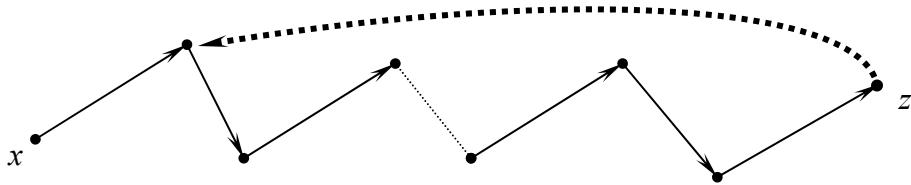


Figure 23.

L'existence d'un puits dans le graphe avec l'orientation opposée implique l'existence d'une source dans le graphe original. ■

On notera que:

- Un sommet puits ne peut pas atteindre un autre sommet que lui-même.
- Un sommet source ne peut être atteint que par lui-même.
- Cependant, un sommet source ne peut pas forcément atteindre tous les sommets (s'il existe plusieurs sources, chacune ne peut pas atteindre les autres).
- De même un sommet puits ne peut pas forcément être atteint par tous les sommets (s'il existe plusieurs puits).

EXERCICE: Montrer qu'un sommet source peut atteindre tous les autres sommets si et seulement si il n'existe qu'un seul sommet source. Même question avec les puits.

COROLLAIRE : *Un graphe $G=(X;E)$ est sans circuit si et seulement si l'ensemble de ses sommets X admet une partition $X=N_0 \cup N_1 \cup \dots \cup N_p$ (appelée partition en niveaux stables), telle que, pour tout sommet x , pour tout $i \leq p$,*

$$(*) \quad x \in N_i \Leftrightarrow i \text{ est la longueur maximum d'un chemin se terminant en } x.$$

PREUVE. Si le graphe G admet un circuit, alors il existe des chemins non élémentaires de longueur infinie donc la partition avec la propriété (*) n'est pas possible.

Si le G est sans circuit, alors on peut construire la séquence d'ensembles :

N_0 – ensemble des sources de G ; N_1 – ensemble des sources de $G \setminus N_0$; ...

N_i – ensemble des sources de $G \setminus (N_0 \cup N_1 \cup \dots \cup N_{i-1})$.

Pour tout i tel que $X \setminus (N_0 \cup N_1 \cup \dots \cup N_{i-1}) \neq \emptyset$ les ensembles N_i sont non vides – d'après le théorème fondamental des graphes sans circuit et le fait qu'un sous-graphe du graphe sans circuit est sans circuit. Evidemment l'ensemble des sources d'un graphe forme un ensemble stable. On démontre facilement par récurrence sur i que les x vérifient la propriété (*). ■

Ce corollaire caractérise en fait la structure des graphes sans circuit. On peut donc dire que ces graphes sont créés à partir d'une séquence ordonnée des stables (niveaux) avec les arcs partant d'un sommet de niveau inférieur vers un sommet de niveau supérieur.

Dans les graphes sans circuit il est possible de réaliser un *tri topologique*, c'est-à-dire d'affecter des numéros différents aux sommets du graphe de telle façon que, quels que soient deux sommets x et y , si (x,y) est un arc du graphe, alors $n(x) < n(y)$. Pour réaliser un tri topologique, on peut numérotter les n sommets d'un graphe $G=(X;E)$ sans circuit, en affectant les numéros de 1 à n de la façon suivante : on numérote d'abord les sommets appartenant à N_0 puis N_1 et ainsi de suite. On vérifie sans difficulté que cette numérotation, qui dépend de l'ordre dans lequel les sommets d'un niveau ont été numérotés, constitue toujours un tri topologique.

10.2. Composantes fortement connexes

Pour distinguer la connexité “ordinaire” définie dans le cas non orienté de la connexité “orientée” on utilise la notion de forte connexité.

Soit $G=(X;E)$ un graphe orienté. Soit \mathcal{R} la relation binaire définie sur X par :

$$x\mathcal{R}y \text{ si et seulement si } x \text{ peut atteindre } y \text{ et } y \text{ peut atteindre } x.$$

On vérifie aisément que \mathcal{R} est une relation d'équivalence. Cette relation définit donc une partition de X formée par les classes d'équivalence de \mathcal{R} . Le sous-graphe de G , engendré par une classe d'équivalence de cette relation, s'appelle *composante fortement connexe* du graphe G . Une composante fortement connexe peut avoir n'importe quelle taille, de 1 jusqu'à $|X|$. Un graphe est *fortement connexe* s'il ne possède qu'une seule composante connexe. Si au contraire il a plusieurs composantes fortement connexes nous dirons qu'il n'est pas fortement connexe.

Le graphe fortement connexe est donc caractérisé par la propriété, que par chaque paire des sommets passe un circuit (le chemin de x à y avec le chemin de y à x).

Le fait qu'un graphe soit fortement connexe signifie, dans le langage courant, que l'on *peut aller de n'importe quel point à n'importe quel autre*. C'est souvent une propriété désirable dans un réseau de communications. D'autres réseaux, cependant, n'ont pas cette propriété (pour diverses raisons, il est alors considéré comme souhaitable que des sommets privilégiés puissent atteindre plus de sommets que d'autres).

LEMME 4 : Soit G est un graphe ayant au moins deux composantes fortement connexes, et soit C , C' deux composantes fortement connexes quelconques de G . Alors, soit il n'y a aucun arc entre C et C' , soit tous les arcs entre ces deux composantes vont de C vers C' , soit tous les arcs entre ces deux composantes vont de C' vers C .

PREUVE. Supposons au contraire que deux composantes fortement connexes C , C' de G soient telles qu'il existe au moins un arc de C à C' et au moins un arc de C' à C . On vérifie alors facilement que pour toute paire de sommets (x,y) avec x dans C et y dans C' , il existe un chemin de x à y et un chemin de y à x ; il doit donc exister une composante fortement connexe C'' contenant x et y (et en fait C'' contient $C \cup C'$), ce qui contredit la définition des composantes fortement connexes. ■

Une *composante fortement connexe* est dite *initiale* s'il n'existe aucun arc entrant dans cette composante à partir d'une autre composante. Une composante fortement connexe est dite *terminale* s'il n'existe aucun arc sortant de cette composante vers une autre composante. Nous allons voir immédiatement que tout graphe orienté possède au moins une composante initiale et une terminale (éventuellement égales).

Soit G un graphe orienté et C_1, \dots, C_p ($p \geq 1$) ses composantes fortement connexes. On peut lui associer un graphe G_r , appelé *graphe réduit*, dont l'ensemble des sommets est $\{C_1, \dots, C_p\}$, et où $(C_i C_j)$ est un arc de G_r si et seulement si, il existe au moins un arc dans G entre les deux composantes fortement connexes C_i et C_j et tous les arcs entre elles vont de C_i vers C_j . Cette définition est cohérente, grâce au Lemme 4.

Quelques remarques:

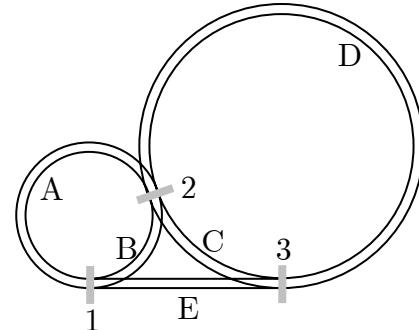
- Pour tout graphe orienté G , le graphe G_r est sans circuit.
- Si G est sans circuit, alors $G_r = G$.
- Si G est fortement connexe, alors G_r a un seul sommet.
- Un sommet x peut atteindre un sommet y dans G si et seulement si, soit: x et y sont dans la même composante connexe de G et dans ce cas y peut aussi atteindre x , soit x est dans C_i , y dans C_j , où C_i, C_j sont deux composantes fortement connexes différentes de G , et C_i peut atteindre C_j dans G_r (et dans ce cas, y ne peut pas atteindre x dans G).

THEOREME 17: *Tout graphe orienté G possède au moins une composante fortement connexe initiale et au moins une composante fortement connexe terminale.*

PREUVE. Soit G un graphe orienté. Puisque G_r est un graphe sans circuit, par le théorème fondamental ci-dessus (p.35), il possède au moins un sommet source C_i (et au moins un sommet puits C_j). Par la définition du graphe réduit, ceci signifie, que C_i (resp. C_j) est une composante fortement connexe initiale (resp. terminale) de G . ■

On peut déterminer les composantes fortement connexes de n'importe quel graphe orienté par une version de l'algorithme très efficace d'exploration Depth-First Search (recherche en profondeur) dû à Tarjan.

EXEMPLE 10: Les rails du train électrique de Yann sont composés de cinq parties A, B, C, D et E, assemblées à l'aide de trois aiguillages 1, 2 et 3, d'après le schéma ci-contre. Si le train avance (sans faire de marche arrière), alors il peut traverser le segment E au plus une fois, quelle que soit sa position de départ. Expliquer ce phénomène en se servant de la notion de forte connexité. Discuter la situation où l'on ajoute un segment F (symétrique à E) reliant A et D.



SOLUTION : La notion de forte connexité peut complètement expliquer ce phénomène. On doit donc étudier les passages possibles entre différents morceaux de rails. Les morceaux de rails peuvent être interprétés comme des sommets, et les passages possibles grâce aux aiguillages adéquats comme des arcs. Puisque ni la position du train au départ ni le sens du parcours ne sont connus, il va falloir considérer deux fois plus de sommets, par exemple le sommet A12 (resp. A21) signifie que le train se trouve, à un moment donné, sur le morceau A et avance de 1 à 2 (resp. de 2 à 1). La figure 24 représente le graphe orienté d'ordre 10 qui modélise le comportement dynamique de notre circuit électrique.

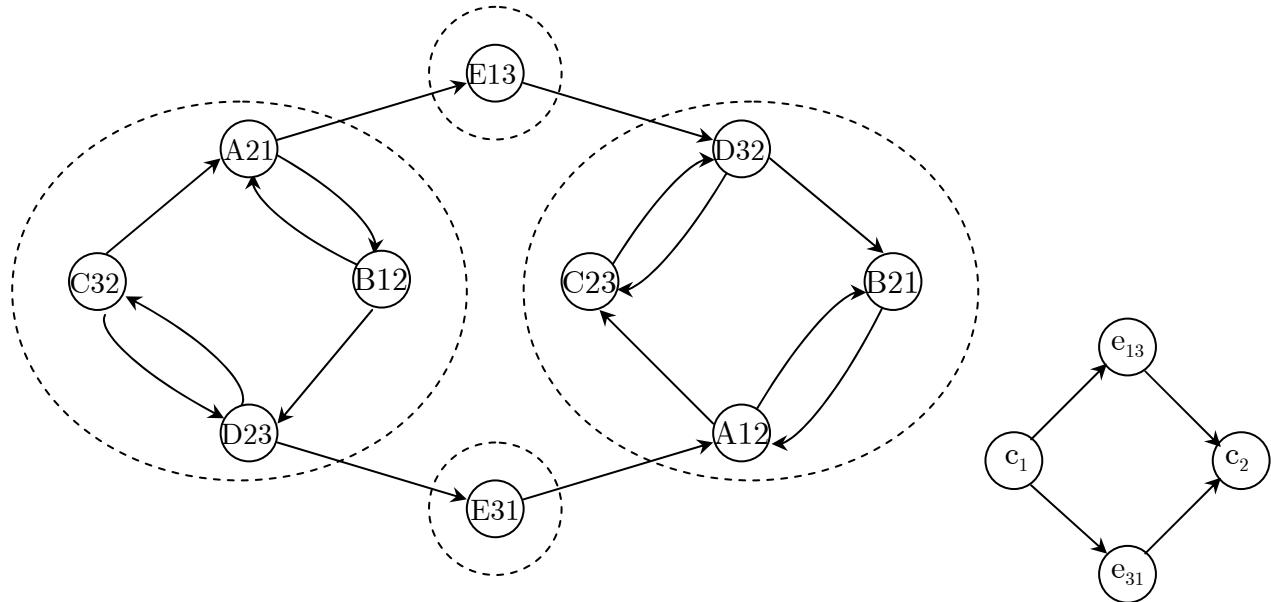
Pour revenir à un endroit le train doit parcourir un circuit dans ce graphe. Notre graphe n'est pas fortement connexe car il n'y a pas de chemin de B21 à C32.

On peut utiliser l'exploration en profondeur pour détecter les composantes fortement connexes. Voici le schéma de l'algorithme :

PHASE 1. Effectuer l'exploration en profondeur à partir d'un sommet quelconque et numérotter les sommets en ordre postfixe. S'il reste des sommets relancer l'exploration en profondeur en continuant la numérotation; on obtient une forêt d'arborescence : la première arborescence est numérotée de 1 à p , la suivante de $p+1$ à q et ainsi de suite.

PHASE 2. Inverser l'orientation de tous les arcs. Effectuer l'exploration en profondeur à partir du sommet avec le plus grand numéro. Lorsque la recherche s'arrête, les sommets marqués forment une composante fortement connexe. S'il reste des sommets, relancer l'exploration en profondeur à partir du sommet restant avec le plus grand numéro. Les sommets marqués lors ce deuxième parcours forment une nouvelle composante fortement connexe et ainsi de suite.

On voit facilement que la complexité de cette algorithme est celle de l'encombrement en mémoire du graphe (par exemple $O(n+m)$ pour les listes de successeurs). Le lecteur trouvera les justifications nécessaires dans [3] ou [12].



En appliquant cet algorithme, on détecte sans difficulté les quatre composantes fortement connexes dont deux se réduisent à un sommet (**Figure 24**). On s'intéresse maintenant aux relations entre les composantes fortement connexes de ce graphe. On construit le graphe réduit. Dans notre cas on obtient le graphe sur la **Figure 25**.

Ainsi, si, au départ, le train se trouvait sur un sommet (ceci détermine l'endroit et le sens du parcours) de la composante c_1 alors on peut (en manipulant convenablement les aiguillages) le laisser circuler dans cette composante ou bien le faire sortir. Une fois sorti, le train n'y reviendra plus. Il passera obligatoirement ou bien par la composante e_{13} ou bien par e_{31} . Mais ces deux composantes se réduisent à un point (il n'y a aucun circuit dans ces composantes!) alors, dans les deux cas, le train repartira vers la composante c_2 sans retour possible. On laisse au lecteur le soin de vérifier, que le graphe devient fortement connexe si l'on installe en plus un morceau de rail F, symétriquement à E. ■

La situation présentée dans l'exercice précédent a une sérieuse signification pratique. Dans l'organisation de la circulation routière d'une ville importante, on préfère des voies à sens unique pour des raisons de sécurité (moins d'accidents face à face, très graves). Il faut que le graphe de cette organisation soit fortement connexe.

La méthode de solution de l'exercice précédent met aussi en évidence le fait, que pour connaître la structure générale des graphes non fortement connexes, il suffit en fait d'étudier les graphes sans circuit.

11. Le plus court chemin

Soit $G=(X;E)$ un graphe orienté et $d: E \rightarrow \mathbb{R}$ une fonction qui associe à toute arête de G un poids réel. Souvent on parle de *distance*, mais, puisque il n'y a pas de restriction $d \geq 0$, elle peut s'interpréter aussi comme le coût ou le profit d'une décision. Pour chaque chemin Γ dans G on définit son poids

$$l(\Gamma) = \sum_{e \in \Gamma} d(e),$$

que l'on suppose nul si Γ ne comporte aucun arc.

Traditionnellement on parle de longueur, mais évidemment cette valeur peut s'interpréter par exemple comme le coût ou le profit d'une politique, considérée comme une suite de décisions. La longueur ainsi définie est une généralisation de la longueur (topologique) d'une chaîne et d'un chemin introduit au début de cours, qui correspondaient au cas $d(e) \equiv 1$.

Pour présenter le problème de recherche d'un chemin de poids optimum nous allons nous concentrer sur le problème du plus court chemin, car le problème de maximisation peut être résolu par le remplacement de la fonction d par $-d$.

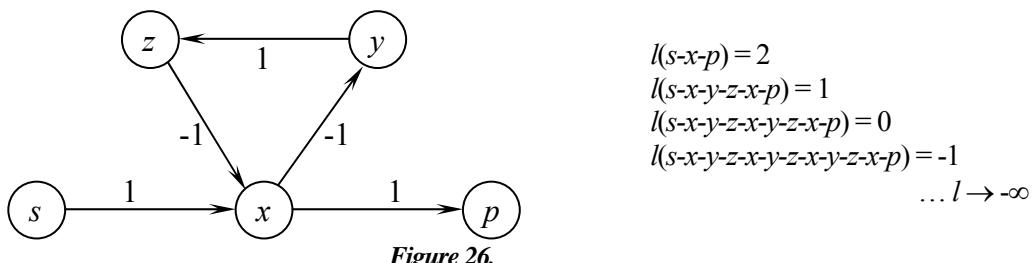
Cette recherche devrait permettre donc de trouver deux choses : la valeur minimale $l(\Gamma)$ et le chemin Γ qui réalise ce minimum. Si un plus court chemin d'un sommet x à un sommet y existe alors sa longueur sera appelée *plus courte distance de x à y* .

En formulant le problème du plus court chemin il faut d'abord bien préciser s'il s'agit de chercher le plus court chemin entre :

- (A) deux sommets fixés, s (une source) et p (un puits) ;
- (B) un sommet fixé s et tous les sommets $x \in X$;
- (C) chaque couple de sommets x et y du graphe ;

car la complexité des algorithmes peut en dépendre.

Ensuite, dans le problème (A) par exemple, il faut tout d'abord s'assurer qu'il existe au moins un chemin de s à p . Ce problème est facile à résoudre par l'exploration du graphe à partir de s . Et enfin, l'existence d'un chemin de s à p ne garantit pas encore qu'il existe un plus court chemin. L'exemple de la figure 28 montre bien, que l'on peut trouver des chemins de plus en plus courts, en incluant dans le chemin le circuit (x, y, z, x) .



Ce phénomène est dû à l'existence d'un circuit, dit *circuit absorbant*, de longueur strictement négative.

Pour éviter ce problème, on peut aussi considérer les problèmes (A'), (B') et (C') où l'on recherche à chaque fois le plus court chemin élémentaire. Ces problèmes ont toujours une solution, car il existe un nombre fini de chemins élémentaires. Bien que fini, ce nombre peut être exponentiel et nous ne connaissons pas de méthode efficace pour résoudre le problème du plus court chemin élémentaire en général. Par contre, il est évident qu'en l'absence de circuit absorbant, les problèmes de recherche d'un plus court chemin et celui du plus court chemin élémentaire sont équivalents, car le plus court chemin est obligatoirement élémentaire.

11.1. Le plus court chemin dans des graphes sans circuit.

Soit $G=(X;E)$ un graphe orienté sans circuit, admettant une arborescence couvrante enracinée en x_1 ; $d: E \rightarrow \mathbb{R}$ une fonction-distance. Supposons que l'ensemble des sommets $X=\{x_1, x_2, \dots, x_n\}$ a déjà été trié (c'est-à-dire : si $(x_i, x_j) \in E$ alors $i < j$).

L'algorithme suivant fournit l'arborescence des plus courts chemins enracinée en x_1 , ainsi que les distances π_i de x_1 à x_i :

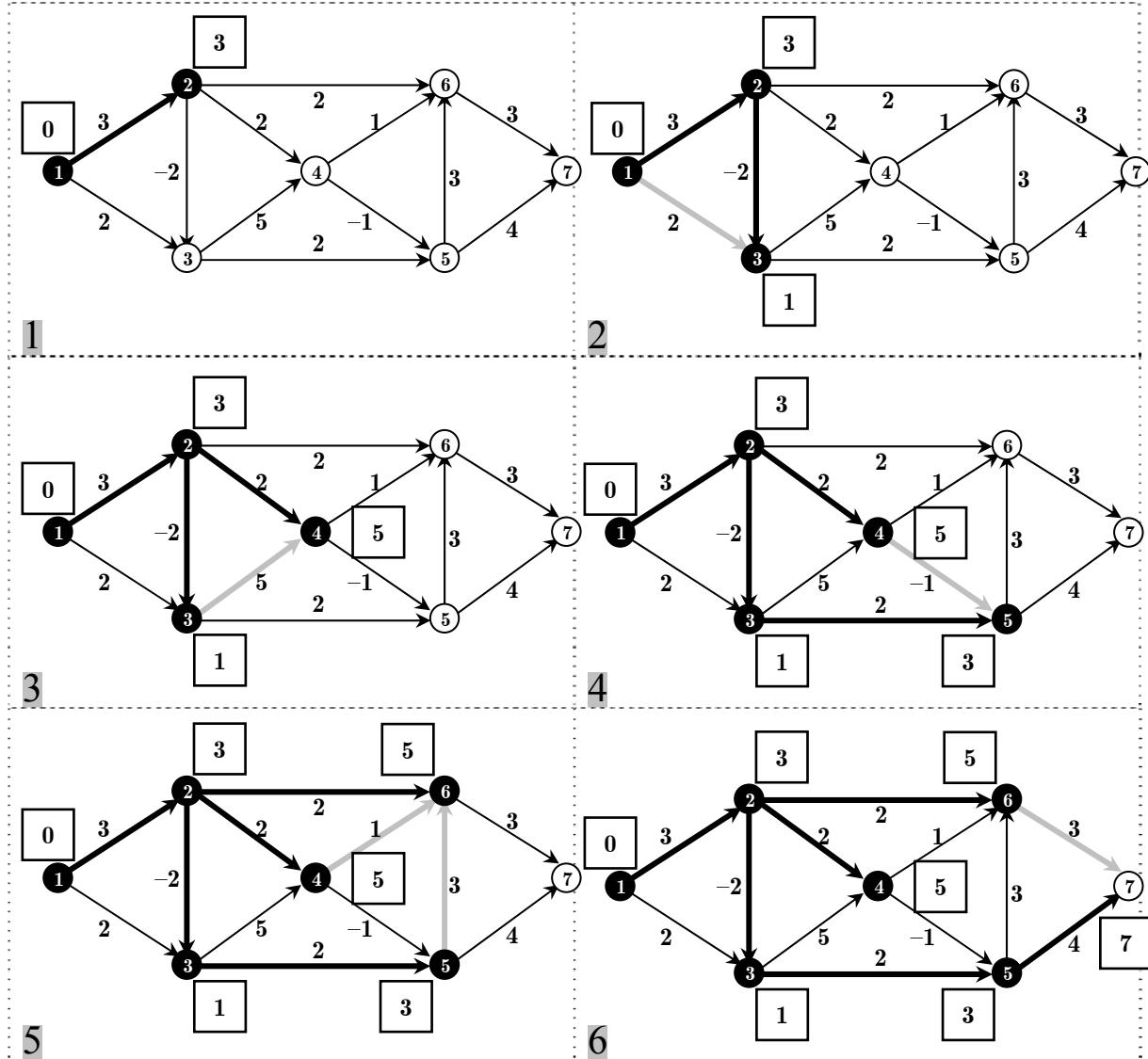
ALGORITHME DE BELLMAN ET FORD.

```

 $\pi_1 := 0; A := \emptyset;$  pour i de 2 à n faire:
     $\pi_i := \min\{\pi_j + d(j, i); j \text{ tel que } (x_j, x_i) \in E\}$ 
     $A := A \cup \{(x_j, x_i) \in E, \text{ tel que } \pi_i = \pi_j + d(j, i)\}$ 
        (j le plus petit indice avec cette propriété)
fin pour
  
```

Il faut signaler que malgré l'intention de chercher seulement le plus court chemin de la racine à un deuxième sommet du graphe (problème A), cet algorithme fournit au passage tous les plus courts chemins de la racine et tous les sommets du graphe (problème B).

Nous allons appliquer cet algorithme sur le graphe suivant pour trouver le plus court chemin de **1** à **7**. À chaque itération les arcs gris sont analysés et l'arc gras noir est choisi.



11.2. Le plus court chemin dans des graphes comportant des circuits.

Très nombreuses applications montrent que l'on ne peut pas se limiter à étudier seulement les graphes sans circuit. Il est évident que dans les cas de graphes non orientés, où une arête peut être parcourue dans les deux sens, ou dans de nombreux modèles liés au trafic routier ou aérien les circuits sont bien présents. Pour éviter des circuits absorbants Dijkstra ajoute l'hypothèse que tous les coûts doivent être non négatifs ($d(e) \geq 0$).

ALGORITHME DE DIJKSTRA.

Soit $G=(X;E)$ un graphe orienté admettant une arborescence couvrante enracinée en r ; $d:E \rightarrow \mathbb{R}_+$ une fonction-distance non-négative. L'algorithme suivant fournit l'arborescence des plus courts chemins enracinée en r et les distances $\pi(x)$ de r à x :

```

initialisation:  $\pi(r) := 0$ ;  $\pi(x) := \infty$  pour  $x \neq r$  ;
 $S := \{r\}$ ;  $A := \emptyset$ ;  $p = r$ ;
itération 1: corrections des marques temporaires:
si  $S = X$  alors FIN, sinon:
pour tous  $x \in X \setminus S$  tels que  $e = (p; x) \in E$ ; si  $\pi(x) > \pi(p) + d(e)$ , alors
 $\pi(x) := \pi(p) + d(e)$ ;
itération 2: acceptation définitive d'une marque minimale:
soit  $x \in X \setminus S$ ; tel que  $\pi(x) = \min\{\pi(y); y \in X \setminus S\}$ ;
soit  $e \in E$ ,  $e = (z; x)$ , tel que  $z \in S$  et  $\pi(x) = \pi(z) + d(e)$ ;
 $S := S \cup \{x\}$ ;  $A := A \cup \{e\}$ ;  $p := x$ ;
revenir à l'itération 1;
FIN

```

Nous allons appliquer cet algorithme sur le graphe ci-contre, pour trouver l'arborescence des plus courts chemins enracinée en r . À la page suivante nous présentons le déroulement détaillé de cet algorithme. Les itérations *impaires* présentent des *corrections* des marques temporaires et les itérations *paires* présentent les *acceptations définitives* d'une marque minimale.

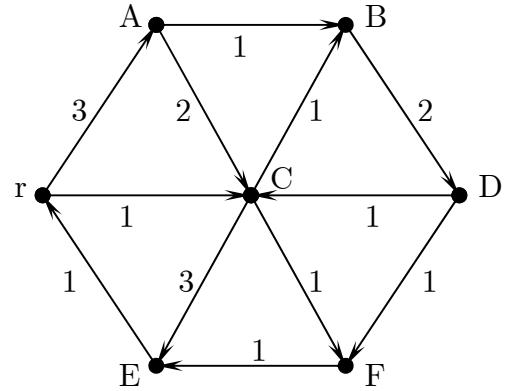
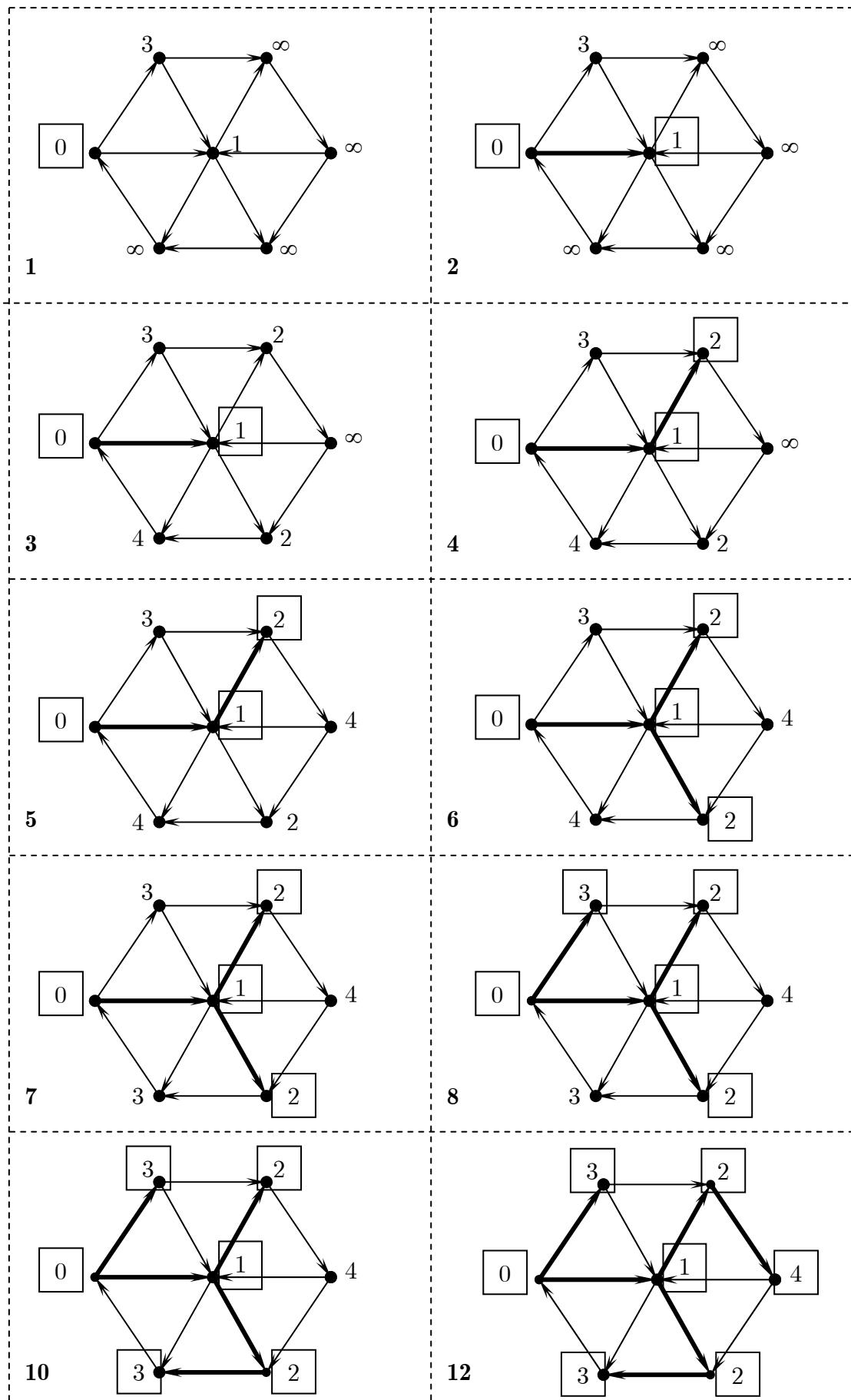


Figure 27.

11.3. Conclusions et remarques.

Pour ne pas alourdir la présentation des deux algorithmes, celui de Bellman et Ford et de Dijkstra, nous avons fait des hypothèses supplémentaires sur les graphes soumis au traitement: l'absence de circuits ou les coûts non négatifs. Si l'on essaie de lancer ces algorithmes sur un graphe quelconque le résultat peut surprendre. Comme il est impossible de faire un tri topologique dans un graphe qui comporte des circuits, l'algorithme de Bellman s'arrêtera inévitablement. Par contre celui de Dijkstra va "tourner" malgré l'existence de coûts négatifs et l'on risque d'obtenir un résultat final erroné. On peut donc concevoir un algorithme général, qui corrige, pas à pas, les "erreurs" de l'algorithme de Dijkstra et s'arrête avec l'arborescence des plus courts chemins enracinée en r ou bien en détectant un circuit absorbant.



Nous avons défini la longueur d'un chemin comme une fonction additive des coûts des ses arcs. Les algorithmes proposés s'adaptent facilement au cas où la longueur est considérée comme le *produit* des valeurs associées aux arcs. En réalité pour résoudre un problème donné il est possible de faire les modifications nécessaires pour se ramener aux cas classiques.

EXEMPLE 5: Dans un réseau de télécommunication le signal doit être transporté du point A vers le point B par l'intermédiaire de quelques autres points relais. Nous connaissons les probabilités p_{ij} ($i,j=1, 2, \dots, 12$) que le signal qui passe entre le point i et le point j soit transmis sans erreur : $p_{1,12} = p_{7,8} = 0,1$; $p_{11,12} = 0,2$; $p_{1,2} = 0,3$; $p_{1,8} = p_{9,10} = 0,4$; $p_{1,6} = p_{8,11} = 0,5$; $p_{6,7} = p_{10,11} = 0,6$; $p_{2,3} = p_{2,5} = 0,7$; $p_{4,5} = p_{5,6} = 0,8$; $p_{2,7} = p_{3,4} = p_{8,9} = 0,9$. La transmission peut se faire dans les deux sens avec $p_{ij} = p_{ji}$. On pose $p_{ii} = 1$ et les autres probabilités nulles. Les liaisons étant indépendantes, la probabilité de passer sans erreur par une chaîne est égale au produit de toutes les probabilités qui composent cette chaîne. On cherche le parcours entre A=4 et B=10 qui maximise cette probabilité.

SOLUTION : Le graphe non orienté qui modélise les relais et les transmissions possibles est représenté sur la Figure 28. Le problème est de trouver une chaîne optimale entre les sommets 4 et 10. Etant donné que les problèmes de cheminement ont été étudiés pour des graphes orientés, on remplace chaque arête par deux arcs opposés pour obtenir un graphe orienté symétrique. Deuxième problème, la modélisation pose la "longueur" qui n'est pas définie ici à la façon classique comme la somme des poids de tous ses arcs mais comme produit de ces poids. Or, une propriété remarquable :

$$\log_2(ab) = \log_2(a) + \log_2(b)$$

permet de transformer notre problème en problème additif classique en remplaçant les poids par leurs logarithmes. On ne peut pas utiliser l'algorithme de Bellman et Ford car le réseau comporte des circuits. On pose $c_{ij} := -\log_2 p_{ij}$. Puisque $0 < p_{ij} \leq 1$ on a $c_{ij} \geq 0$. Il ne faut pas oublier que la fonction $y = -\log_2(x)$ est strictement décroissante alors en utilisant cette transformation on doit chercher le plus court chemin. Cette transformation complique le calcul numérique. On va donc considérer le raisonnement précédent comme la preuve que l'on peut appliquer directement sur les données originelles p_{ij} une version modifiée de l'algorithme de DIJKSTRA où on remplace l'opération d'addition par la multiplication et l'opération Min par Max. On trouve ainsi la chaîne 4–5–6–1–8–9–10 avec la valeur 0,04608.

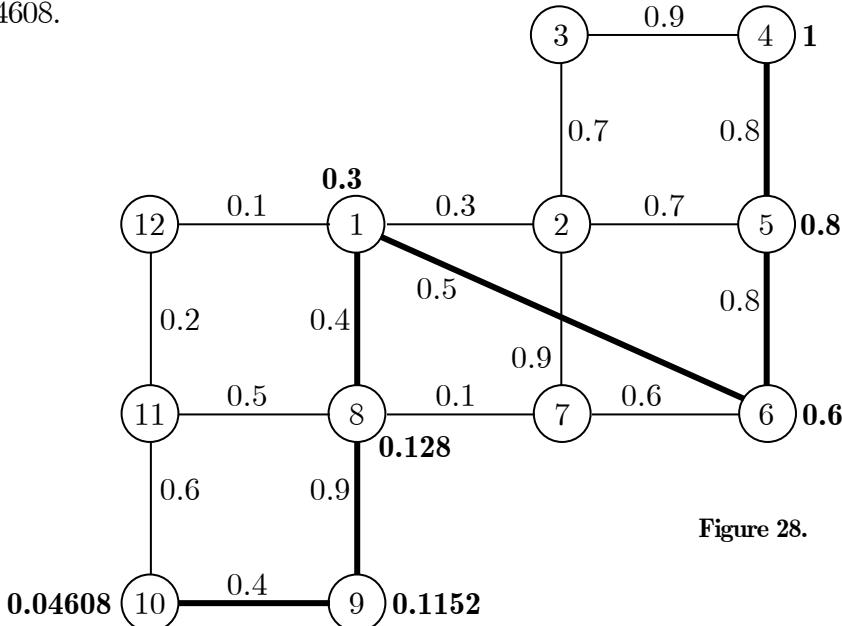


Figure 28.

12. Ordonnancements.

Quel que soit le poste occupé, le niveau de responsabilité ou la position dans hiérarchie, un ingénieur est fréquemment confronté aux problèmes liés à la bonne organisation des projets plus ou moins complexes. Avec de nombreuses contraintes technologiques, économiques ou sociales la réussite passe finalement par une meilleure organisation de l'ensemble des tâches qui composent un projet. Comment réaliser ces tâches, avec des machines disponibles, sous de nombreuses contraintes, de la façon optimale – tel est le sujet d'une branche de la recherche opérationnelle intitulée ordonnancements. C'est le domaine qui se développe le plus rapidement aujourd'hui (vu le nombre de publications scientifiques) et qui donne le plus d'applications directes dans la pratique quotidienne. Très souvent les nombreux axes de recherche théorique (presque pure!) sont inspirés par des problèmes et des situations réelles. Nous donnons seulement dans ce cours un aperçu du domaine. Les ordonnancements simples sont présentés en détail et seulement quelques extensions complexes sont traités dans les exercices ou travaux pratiques proposés. A titre d'exemple, les deux sujets de projets de fin d'études proposés à l'ENSIMAG en recherche opérationnelle : l'organisation optimale de la campagne de fauchage des routes et l'organisation optimale de la viabilité hivernale, font partie du domaine des ordonnancements complexes.

Introduction.

Dans chaque problème d'ordonnancement, quel que soit son niveau de complexité, on doit d'abord préciser son environnement, c'est-à-dire un ensemble de **tâches** à exécuter sur un ensemble de **processeurs** disponibles. On peut donc imaginer par exemple que :

- on doit fabriquer une pièce sur une machine ;
- une opération doit être réalisée par un chirurgien ;
- les sous-programmes d'un programme informatique doivent être effectués par des microprocesseurs parallèles ;
- la tonte de l'herbe sur l'accotement d'une route doit être réalisée par des tondeuses ou des épaveuses.

A chaque tâche on peut associer plusieurs paramètres (durée, date de lancement, date limite, etc.). Les différentes tâches peuvent être liées par des contraintes (précédences, préemption possible ou non, etc.). Les processeurs ont leurs *vitesses d'exécution des tâches* et leurs *type* : parallèles - peuvent remplir les même fonctions; dédiés - sont spécialisés à l'exécution de certaines tâches. Des nombreux problèmes se posent: quelle tâche sur quel processeur (affectation, allocation, etc.)? Dans quel ordre réaliser les tâches sur les processeurs ? Bien évidemment il faut préciser l'*objectif*, comme par exemple, minimiser une date de fin d'une tâche, minimiser la somme des retards (c'est-à-dire des dépassements des dates limites) etc. La théorie de l'ordonnancement propose des modèles, principes et techniques de solution des nombreux problèmes de ce genre.

EXEMPLE 6 : Une entreprise fabrique des jouets en bois. Chaque pièce doit être découpée, ensuite poncée et finalement peinte. Son atelier comporte des machines qui réalisent automatiquement la découpe (P_1), le ponçage (P_2) et la peinture (P_3). Les durées en

secondes intervenant dans le processus de production de trois pièces T_1 , T_2 et T_3 sont indiquées dans le tableau:

	T_1	T_2	T_3
P_1	40	80	70
P_2	50	60	80
P_3	70	60	50

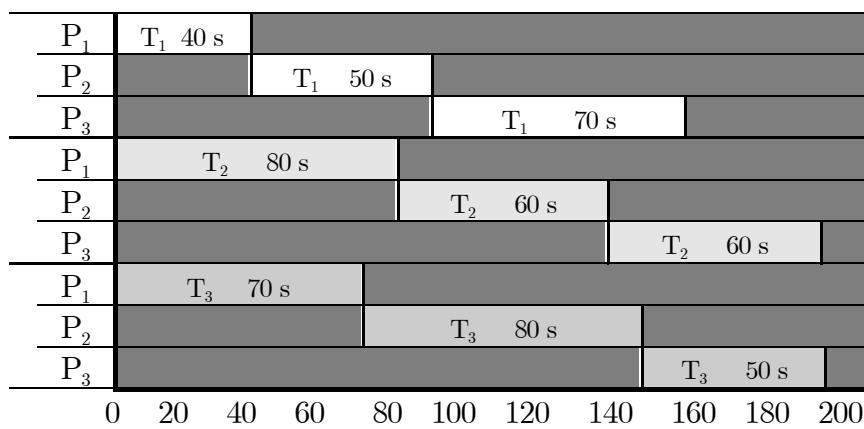
Calculer la durée totale minimale de fabrication d'une série de ces trois pièces.

En langage d'ordonnancement nous cherchons ici une organisation optimale de traitement des trois tâches, chacune composée de trois tâches élémentaires, exécutées successivement, dans l'ordre donné, par trois processeurs spécialisés. Au total, il y a donc 9 tâches élémentaires.

Pour présenter les difficultés qui se posent devant ce type de problèmes nous allons utiliser les **diagrammes de GANTT**. L'axe horizontal de ce diagramme représente le temps et les différents niveaux sont réservés aux processeurs. Une tâche élémentaire est représentée par un rectangle de longueur correspondante à sa durée. Il faut signaler qu'il s'agit d'une méthode de présentation élégante et claire, et non d'une technique efficace de solution des problèmes d'ordonnancement. On travaille avec quelques hypothèses de base : on suppose qu'à chaque instant un processeur ne peut pas exécuter plus d'une tâche et, à chaque instant, une tâche ne peut pas être exécutée par plus qu'un seul processeur. Un processeur ne peut donc commencer l'exécution d'une tâche que si elle est disponible.

En revenant à notre problème, supposons dans le premier temps qu'il y a un nombre suffisant de machines (trois de chaque type), pour réaliser certaines tâches simultanément. Dans

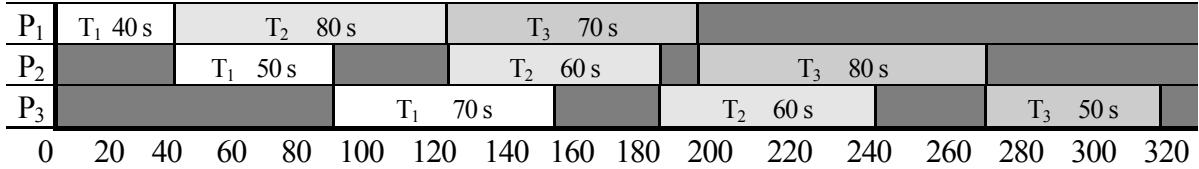
ce cas nous devons respecter seulement quelques contraintes d'antériorité, concernant l'ordre : découpage – ponçage – peinture, dans la production de chaque pièce. Le diagramme de Gantt se présente ainsi:



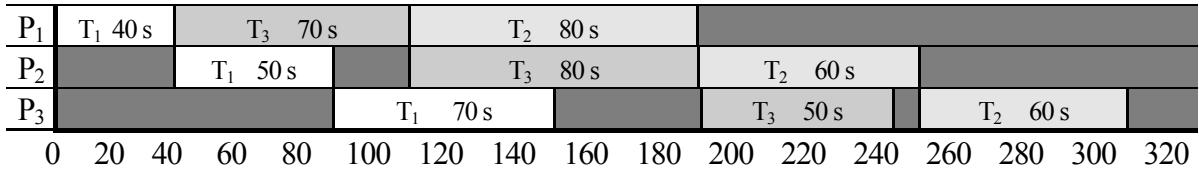
et on peut en déduire la durée totale de production de 200 secondes (pour notre série des pièces). On constate beaucoup de temps mort (les tâches en gris foncé) et très rarement une entreprise peut se permettre un tel luxe.

Le problème se complique s'il n'y a qu'une seule machine de chaque type. C'est l'ordre de traitement des pièces sur chaque machine qui va jouer un rôle fondamental. Sup-

posons, par exemple, que chaque machine est programmée pour traiter les pièces dans l'ordre P1 – P2 – P3. Dans ce cas le diagramme de Gantt se présente ainsi:



La durée totale de production est de 320 secondes. Il suffit maintenant de changer l'ordre de traitement des deux dernières pièces et reprogrammer chaque machine pour traiter les pièces dans l'ordre P1→P3→P2 pour diminuer encore la durée totale de production :



Si rien ne nous oblige d'avoir le même ordre des tâches sur tous les processeurs, alors en général, avec n tâches et m processeurs, on a, à priori, $(n!)^m$ possibilités ! Devant cette “explosion combinatoire” de cas, les problèmes de recherche de la durée minimum sont intractables par énumération explicite. Ces problèmes, où chaque tâche est traitée par tous les processeurs dans le même ordre, porte le nom du **Flow-Shop**.

Pour l'information, nous donnons quelques précisions à ce sujet: pour $m=2$ le problème est facile, car il peut être résolu par algorithme de Johnson en temps $O(n^2)$ (voir **Exercice 3**); pour $m \geq 3$ ce sont les problèmes les plus durs en combinatoire, avec seule distinction de cas $m=3$ pour lequel on sait qu'il existe toujours une solution optimale avec un ordre identique de traitement des tâches sur les trois processeurs (voir **Exercice 2**).

Ordonnancements simples.

On parle d'un problème *d'ordonnancement simple* quand :

1. le nombre de processeurs parallèles n'est pas limité ;
2. les tâches élémentaires, qui ont leur durée d'exécution connues, sont soumises seulement aux contraintes de précédence (T_i avant T_j signifie, que la tâche T_i doit être terminée avant de commencer T_j).

Pour un projet, composé de plusieurs tâches élémentaires, on cherche un planning (ou calendrier) d'exécution des tâches (c'est-à-dire un ensemble de dates affectées à chaque tâche, qui respecte les durées des tâches et les contraintes de précédence), qui minimise la durée totale du projet (c'est-à-dire le temps entre le début et la fin du projet). Les méthodes de solution de tous les problèmes d'ordonnancements simples sont basées sur le fait, que toutes les tâches forment un ensemble partiellement ordonné par la relation de précédente des tâches. Si on modélise cet ordre par un graphe orienté où les arcs seront valués par les durées, alors la durée totale d'un projet sera la valeur du plus long chemin du début jusqu'à la fin du projet.

Nous avons le choix dans la modélisation des tâches par un graphe – soit par les sommets soit par les arcs, d'où les deux méthodes : **potentiels-tâches** et **potentiels-événements**, présentées en détails sur le même exemple.

EXEMPLE 7 : La construction traditionnelle d'une maison individuelle commence par l'établissement des fondations. Supposons que ce projet se décompose en cinq tâches élémentaires:

- T - terrassement - durée 3 jours;**
- G - mise en place de la grue - durée 1 jour;**
- R - branchements aux réseaux d'eau et EDF - durée 2 jours;**
- B - coulage d'une dalle de béton - durée 3 jours;**
- S - installation de la fosse septique - durée 4 jours.**

Les tâches **T, G et R** peuvent démarrer tout de suite, par contre on ne peut installer la fosse septique ni couler la dalle de béton qu'après avoir effectué les travaux de terrassement. Il est aussi évident que la pose d'une dalle de béton nécessite de l'eau et aussi la présence de la grue qui, pour fonctionner, a besoin de l'électricité.

Le problème est de proposer un calendrier d'exécution des tâches respectant les contraintes d'antériorité (contraintes de précédence) et permettant de réaliser l'ensemble des tâches en un temps minimum.

Plus précisément on va donc s'intéresser à la **durée totale** minimum et, pour chaque tâche x :

la **date au plus tôt**, notée $\pi(x)$, qui est la date minimum à laquelle on peut démarrer l'exécution de la tâche x compte tenu de toutes les données et contraintes du problème (par convention, le projet démarre toujours au moment 0);

la **date au plus tard**, notée $\eta(x)$, qui est la date limite à laquelle on doit commencer l'exécution de la tâche x sous peine de retarder l'exécution totale du projet ;

la **marge** représente le délai dont on dispose pour démarrer la tâche en respectant la durée totale minimum (on a évidemment $m(x) = \eta(x) - \pi(x)$).

Les tâches qui ont une marge nulle seront appelées **tâches critiques**. Tout retard pris dans l'exécution d'une tâche critique entraîne un retard dans la réalisation du projet. Il est donc intéressant d'établir une liste des tâches critiques.

Pour la commodité il est toujours souhaitable d'extraire dans un tableau les données essentielles :

tâche	durée (jours)	contraintes : après... :
T	3	-
G	1	-
R	2	-
B	3	T, G, R
S	4	T

Solution avec la méthode potentiels-tâches.

On construit le graphe d'ordonnancement où les sommets représentent les tâches et les arcs les contraintes d'antériorité.

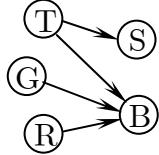


Fig.29.

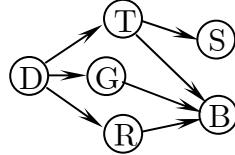


Fig.30.

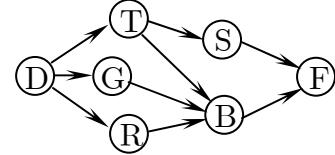


Fig.31.

Ce graphe est sans circuit (Fig.29.). Il contient trois sources, donc l'arborescence couvrante n'existe pas. Pour cette raison il faut ajouter un sommet supplémentaire, une super-source qui précède les sources existantes. Ce sommet (D sur la Fig.30.) s'interprète comme le début du projet. Ce n'est pas une "vraie" tâche - c'est un moment de durée zéro. D'autre part, notre graphe admet deux puits, alors il faut aussi ajouter un super-puits comme successeur des puits existants. Ce sommet (F sur la Fig.31) s'interprète comme la fin de projet. Cette opération garantit qu'il existe une anti-arborescence couvrante avec l'anti-racine en F.

Ensuite on affecte à chaque arc la durée de la tâche correspondant à son sommet initial (Fig.32.).

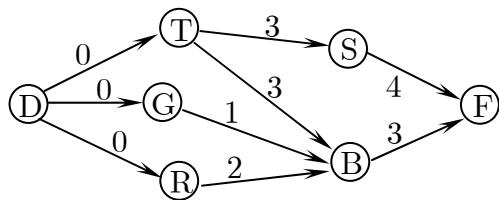


Fig. 32.

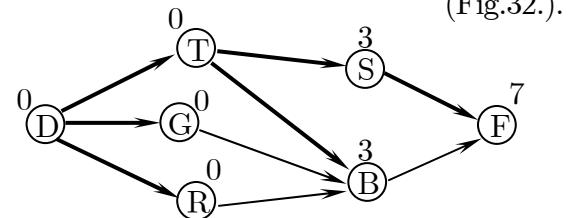


Fig. 33.

Les valeurs π , déterminées par l'algorithme de Bellman et Ford pour la recherche du plus long chemin à partir du sommet D (Fig.33.), s'interprètent comme les dates au plus tôt. Entre autres $\pi(F)$ représente la durée totale minimum du projet. Pour pouvoir calculer l'ensemble des dates au plus tard, on trouve d'abord l'anti-arborescence des plus long chemins (avec leur valeur π') d'un sommet quelconque au sommet F (Fig.34.).

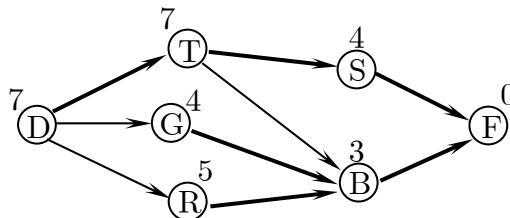


Fig.34.

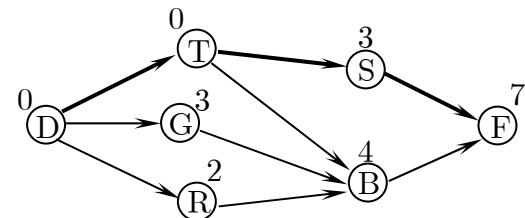


Fig. 35.

Les dates au plus tard, représentées sur la Fig.35., sont calculées grâce à la relation :

$$\eta(x) = \pi(F) - \pi'(x) .$$

Un plus long chemin de D à F (en caractère gras sur la Fig.35.) s'appelle **chemin critique**. Les tâches critiques sont celles situées sur un chemin critique. Dans notre exemple, en plus du début et de la fin du projet, il y a deux tâches critiques **T** et **S**. La durée totale minimum est de 7 jours. On peut regrouper tous ces résultats dans un tableau :

tâche	durée (jours)	contraintes : après... :	π	η	m
D (début)	0	-	0	0	0
T	3	D	0	0	0
G	1	D	0	3	3
R	2	D	0	2	2
B	3	T, G, R	3	4	1
S	4	T	3	3	0
F (fin)	0	B, S	7	7	0

Solution avec la méthode potentiels-événements* (PERT).

Dans cette modélisation une tâche est représentée par un arc. Son sommet initial correspond au début de cette tâche et son sommet terminal à la fin. Les sommets représentent donc des événements, instants privilégiés de l'exécution du projet. Comme dans la méthode précédente, les contraintes d'antériorité sont aussi représentées par des arcs (pointillés – pour les distinguer des tâches réelles). On considère que ces arcs correspondent aux tâches fictives de durée nulle.

Notre problème d'ordonnancement se présente initialement ainsi :

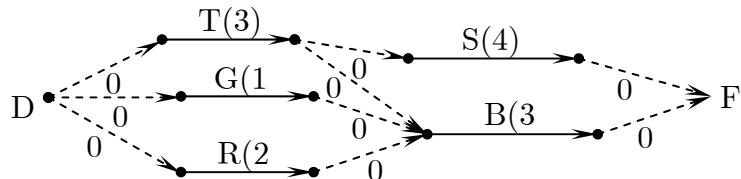


Fig.36.

Ce graphe, beaucoup trop compliqué, est sans intérêt pratique et il faut obligatoirement appliquer une procédure de simplification :

On peut identifier (contracter) deux extrémités d'une tâche fictive et enlever la tâche fictive.

Il faut prendre garde que chaque opération de réduction préserve toutes les contraintes d'antériorité et n'en ajoute pas de nouvelles.

Ainsi, par une suite de réduction, on arrive facilement au graphe suivant :

* Dans la bibliographie anglo-saxonne cette méthode porte le nom
PERT (Program Evaluation and Review Technique)

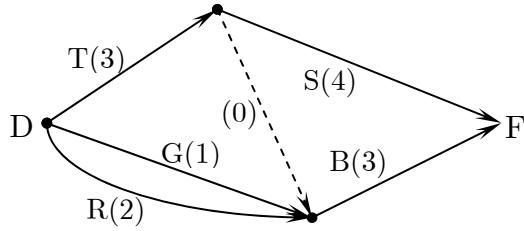


Fig.37.

C'est le plus petit graphe PERT que l'on puisse obtenir sur notre exemple, car une contraction supplémentaire, pour enlever le dernier arc pointillé, ajouterait, entre autres, une nouvelle contrainte "S après G", qui n'existe pas auparavant.

A ce modèle simplifié on applique le même traitement que dans la méthode précédente, à savoir la recherche des plus longs chemins entre D et F.

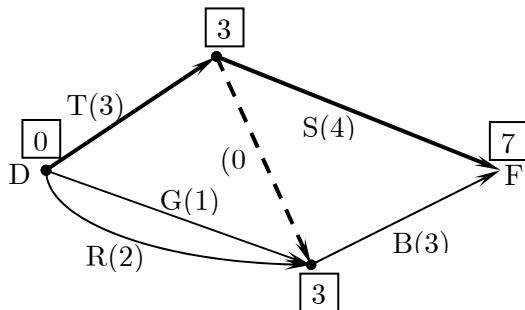


Fig.38.

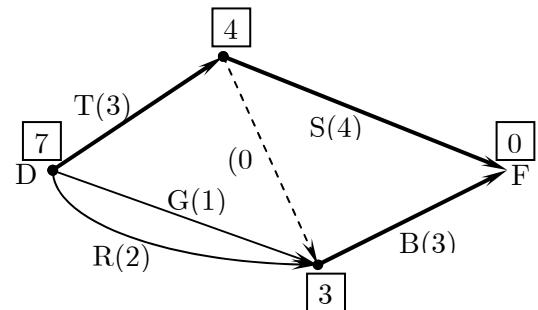


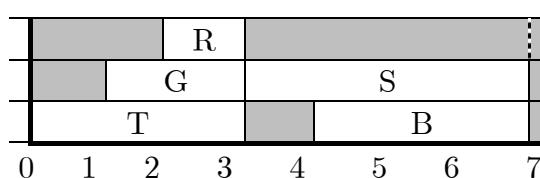
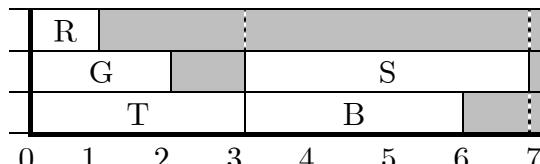
Fig.39.

Puisque une tâche est représentée par un arc $e=(x; y)$ et les algorithmes de plus long chemin affectent des potentiels (les nombres encadrés, π sur la Fig.38. et π' sur la Fig.39.) aux sommets, il faut encore préciser les formules pour déterminer les dates au plus tôt et au plus tard :

la date au plus tôt de $e = \pi(x)$ (x est le sommet initial de l'arc e) ;

la date au plus tard de $e = \pi(F) - \pi'(y) - d(e)$ (y est le sommet terminal de e et $d(e)$ la durée de la tâche).

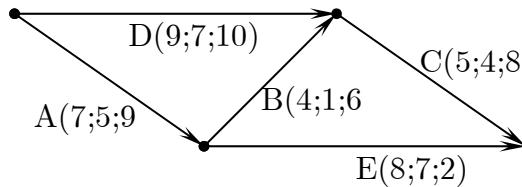
Les deux méthodes donnent, bien évidemment, les mêmes résultats que l'on peut aussi interpréter en utilisant les diagrammes de GANTT :



Sur le premier diagramme, qui permet de retrouver les dates au plus tôt, les rectangles sont mis sur leurs positions limites à gauche (en respectant les contraintes d'antériorité). Le deuxième diagramme permet de retrouver les dates au plus tard car les rectangles ont été repoussés sur leurs positions limites à droite (en respectant les contraintes d'antériorité et sans dépasser la date limite 7). Les marges se mesurent par les déplacements des rectangles - ceux qui n'ont pas bougé représentent les tâches critiques.

Nous proposons dans l'exemple suivant une extension intéressante du problème d'ordonnancement simple.

EXEMPLE 8 : *Le réseau PERT suivant représente les contraintes d'antériorité entre différentes tâches d'un projet complexe:*



Pour certaines tâches, il est possible de réduire la durée normalement prévue en lui affectant des moyens supplémentaires. Les trois nombres à côté de chaque tâche représentent respectivement: sa durée normale d'exécution, la durée "accélérée" la plus petite possible et le coût de réduction par jour.

- Quelle est la durée minimale d'exécution du projet avec les temps normalement prévus pour chaque tâche?*
- Quelle est la durée minimale d'exécution du projet avec les durées "accélérées" pour chaque tâche? Quel est le coût total consacré à cette diminution?*
- Quel est le budget minimum nécessaire pour réduire la durée du projet d'un jour? de deux jours?*
- Quel est le budget minimum nécessaire pour réduire la durée du projet au minimum?*

Solution: Les réponses aux questions a) (**16** jours) et b) (**12** jours) s'obtiennent par la solution de deux problèmes d'ordonnancements simples.

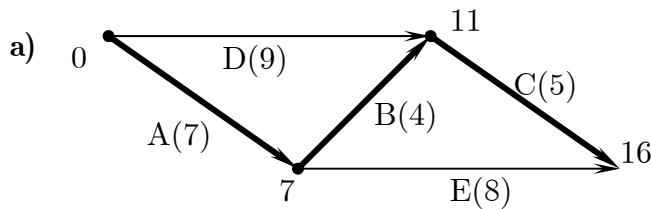


Fig.40.

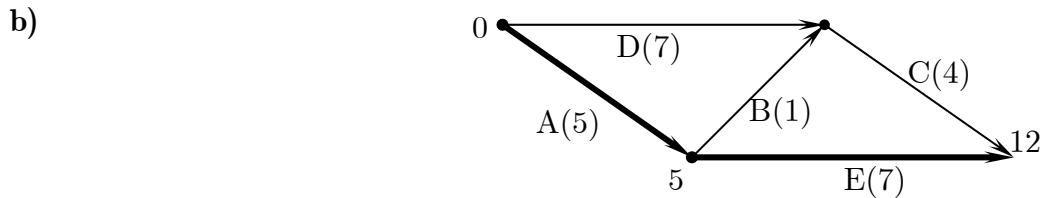


Fig.41.

Le coût total $(7-5) \times 9 + (4-1) \times 6 + (5-4) \times 8 + (9-7) \times 10 + (8-7) \times 2 = 66$ nécessaire pour l'organisation du projet envisagée dans b) est trop élevé car certaines diminutions sont payées inutilement. Par exemple le gain d'un jour sur la tâche B et la tâche D suffit pour respecter la durée de 12 jours.

c) On s'intéresse seulement au graphe partiel composé des tâches critiques pour lesquelles la diminution est possible (le nombre entre parenthèses est le coût de diminution de la durée d'une tâche d'un jour).

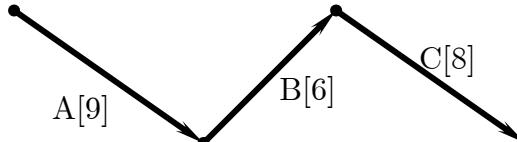


Fig. 42.

Nous avons ici le chemin critique unique. Gagner 1 jour peut se faire sur une des trois tâches le composant. Pour consacrer un budget minimal et gagner 1 jour il faut choisir la diminution d'un jour de la tâche B (la diminution qui coûte 6 est la moins chère).

Pour gagner un deuxième jour on ne peut pas continuer ainsi car il faut d'abord mettre à jour les tâches critiques. Avec la nouvelle durée de trois jours pour la tâche B le graphe partiel des tâches critiques est présenté sur la Fig.43.

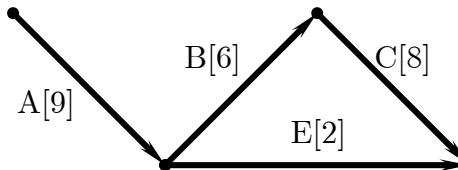


Fig. 43.

Il est évident que par la seule diminution d'un deuxième jour sur B on ne change pas la durée totale de projet car il existe un autre chemin critique, composé des deux tâches A et E, de durée 15 jours. Pour diminuer la durée totale il faut simultanément "couper" tous les chemins critiques du début du projet jusqu'à la fin. Nous rappelons qu'en étudiant la connectivité des graphes non-orientés nous avons introduit la notion de cocycle (p.16.).

Dans le cas orienté le cocycle associé à S , $\omega(S)$, est partitionnable en deux parties:

$$\omega^+(S) = \{e = (x,y) \in E; \text{ tel que } x \in S\} \quad \text{et} \quad \omega^-(S) = \{e = (x,y) \in E; \text{ tel que } y \in S\}.$$

Pour couper tous les chemins de le début jusqu'à la fin du projet on doit s'intéresser aux $\omega^+(S)$ tel que le début appartient à S .

En voici quelques exemples de $\omega^+(S)$ avec leur coût qui est la somme des coûts de diminution d'un jour des tâches le composant.

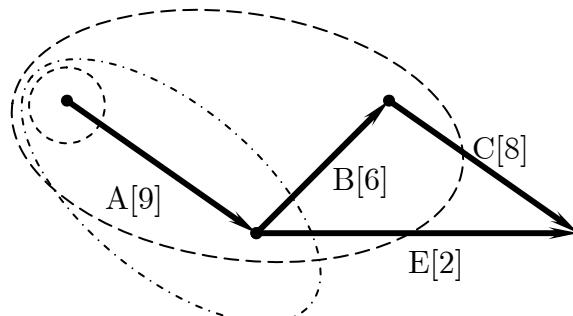


Fig. 44.

$\omega^+(S)$	coût
A	9
B; E	$6+2=8$
C; E	$8+2=10$

Pourtant il faut aussi tenir compte des parties $\omega^-(S)$. Par exemple la partition du cocycle présenté sur la Fig.21.

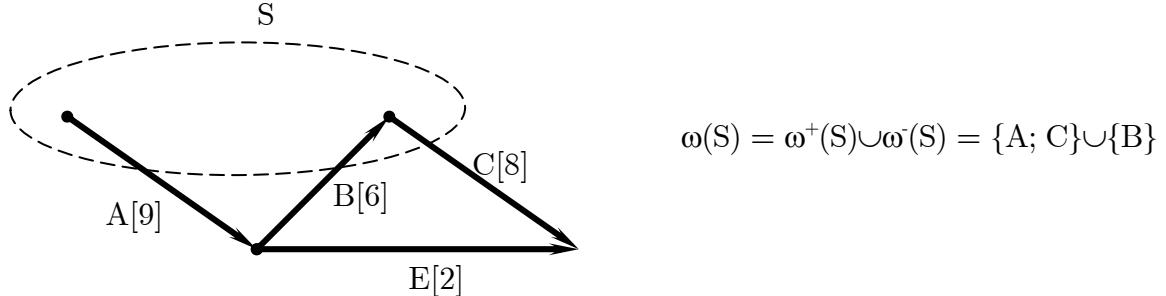


Fig. 45.

s'interprète ainsi: pour gagner finalement un jour on peut diminuer d'un jour les deux tâches A et C et augmenter d'un jour la tâche B. Effectivement, dans notre cas, puisque la tâche B été précédemment diminuée, on peut se demander si cette diminution est toujours intéressante dans la suite et envisager le retour à la durée précédente. Le coût de cette opération: $9+8-6=11$ est plus grand que dans d'autres cas, alors, pour gagner un jour supplémentaire, il faut choisir la coupe (le cocycle) $\{B; E\}$ de coût 8.

Avec les nouvelles durées toutes les tâches deviennent critiques et la durée totale atteint 14. La figure suivante représente la liste exhaustive de tous les cocycles orientés ainsi que leurs coûts.

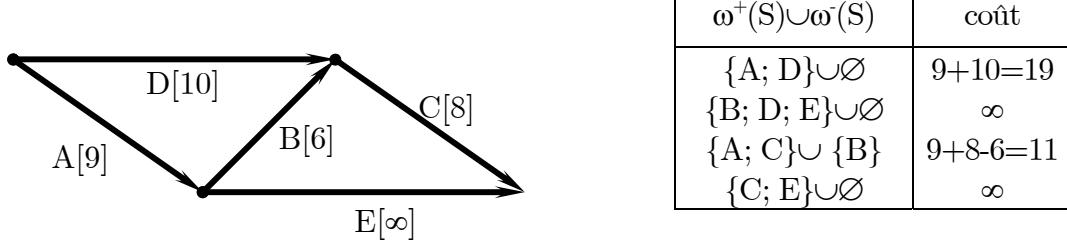


Fig. 46.

Nous avons simulé par un coût infini $[\infty]$ l'impossibilité de diminution de la durée de la tâche E. Le budget minimal pour gagner un jour supplémentaire est obtenu par la diminution d'un jour des tâches A et C et le retour à la durée de 3 jours de la tâche B.

Avec les nouvelles durées toutes les tâches restent critiques et la durée totale atteint 13. Il reste un seul cocycle, à savoir $\{A; D\} \cup \emptyset$, avec le coût fini de 19.

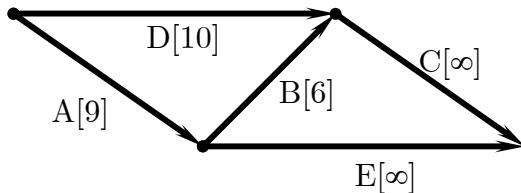


Fig. 47.

En diminuant d'un jour les tâches A et D on atteint la durée totale limite de 12 jours.

La courbe, linéaire par morceaux et toujours convexe (Fig.48.), représente le budget minimal supplémentaire en fonction de la durée envisageable:

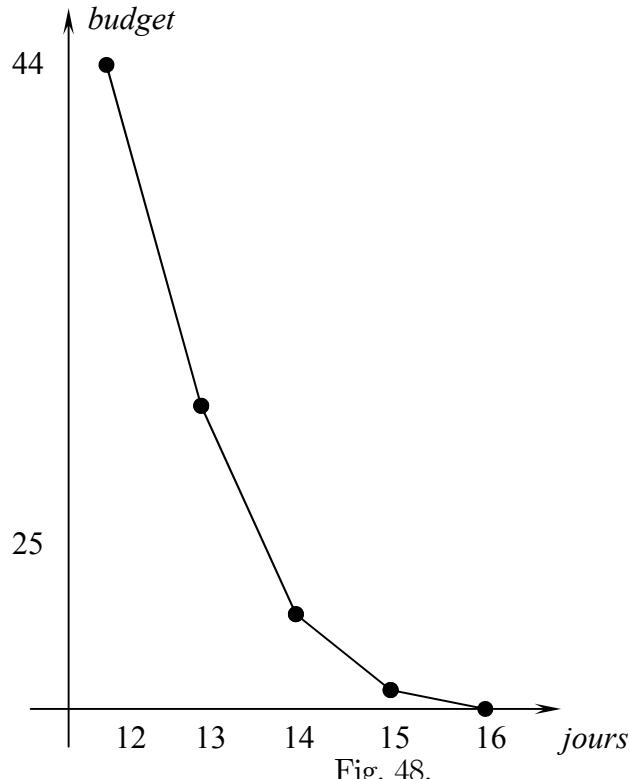


Fig. 48.

La justification simple de la méthode présentée passe par la théorie du flot maximum.

Exercice 1:

1) Dessiner un réseau potentiels-tâches puis PERT aussi simple que possible qui modélise l'ordonnancement des tâches du problème présenté dans l'EXEMPLE 6 p.47, en supposant que nous ne disposons que d'une seule machine de chaque type et que les machines ont été programmées pour traiter les tâches dans l'ordre T_3, T_2, T_1 .

Calculer la durée totale de fabrication et indiquer les tâches critiques.

2) Nous avons la possibilité de remplacer chaque processeur P_1, P_2 et P_3 par sa version plus moderne P'_1, P'_2 et P'_3 qui diminue la durée de chaque opération de 10 s. Quelle sera la nouvelle durée minimum de fabrication?

3) Le remplacement des trois machines est très coûteux. Combien de machines suffira-t-il de remplacer si nous sommes satisfaits d'une durée de fabrication égale à 280 s?

Exercice 2:

Démontrer, que dans le Flow-Shop, avec n tâches et m processeurs où on cherche à minimiser la durée totale, il existe toujours une solution optimale vérifiant les deux propriétés suivantes :

- 1) l'ordre de traitement de n tâches par le premier processeur (P_1) est le même que celui sur le deuxième processeur (P_2) ;
- 2) l'ordre de traitement de n tâches par le dernier processeur (P_m) est le même que celui sur l'avant-dernier processeur (P_{m-1}).

En déduire, que dans le problème avec $m=3$ processeurs on peut se limiter à chercher une permutation commune des n tâches sur les trois processeurs.

Vérifier, par énumération explicite, que dans le cas de la Fig.49., où deux tâches sont traitées par quatre processeurs, les ordonnancements de permutation (même ordre des tâches sur chaque processeur) ne sont pas optimaux.

	T_1	T_2
P_1	4	1
P_2	1	4
P_3	1	4
P_4	4	1

Fig. 49.

Exercice 3:

Dans un atelier, quatre tâches T_1 , T_2 , T_3 et T_4 doivent passer sur le processeur P_1 avant de passer sur P_2 . Les durées d_{kn} de traitement de la tâche T_n par le processeur P_k sont données par le tableau suivant:

	T_1	T_2	T_3	T_4
P_1	4	4	8	3
P_2	5	2	6	5

On cherche à déterminer l'ordre de passage des pièces qui minimise la durée totale de traitement, ce qui relève des ordonnancements complexes.

Algorithme de Johnson (1954):

- 1° Déterminer l'ensemble des tâches $S = \{T_i \mid telle\ que\ d_{1i} \leq d_{2i}\}$.
- 2° Ordonner d'abord les tâches de S dans l'ordre croissant des d_{1i} et ensuite les autres tâches dans l'ordre décroissant des d_{2j} .

Traiter l'exemple numérique.

2^{ème} partie :

PROGRAMMATION LINÉAIRE

1. Modéliser par un programme linéaire

Les cours traditionnels d'optimisation supposent la pré-existence d'un modèle mathématique et d'un critère à optimiser, alors ils présentent surtout les méthodes permettant de trouver des solutions optimales pour ces modèles. Pour permettre d'améliorer le comportement des systèmes réels et aussi de mieux les connaître, le travail de modélisation doit être réalisé avec le plus grand soin. On ne peut pas découpler l'aspect modélisation de l'aspect optimisation. Nous allons donc commencer par l'étape de modélisation. Nous illustrons à l'aide d'exemples simples les trois modèles classiques dits : *de production* (ou plutôt d'utilisation des ressources), *de transport* et *d'alimentation* ainsi que leurs modèles duals (sans que la notion de dualité soit abordée).

EXEMPLE 1:

Dans une usine chimique on produit deux sortes d'engrais minéraux complexes en mélangeant plusieurs éléments fertilisants. Le *produit I*, composé d'un kilogramme de nitrates et de deux kilogrammes de sel de potassium, rapporte 7€ à la vente, et le *produit II*, composé d'un kilogramme de nitrates, un kilogramme de phosphates et trois kilogrammes de sel de potassium rapporte 9€. Actuellement il reste encore en stock 8kg de nitrates, 4kg de phosphates et 19kg de sel de potassium. Quelles quantités de chaque produit faut-il produire pour maximiser le bénéfice?

Les données peuvent être regroupées dans le tableau suivant:

<i>produit</i>	<i>I</i>	<i>II</i>	
<i>profit (€)</i>	7	9	<i>stock</i>
	<i>N</i>	1	8
<i>compositions</i>	<i>P</i>	0	4
	<i>K</i>	2	19

Il faut décider quelles quantités x_1 du produit I et x_2 du produit II doit-on préparer pour rendre le profit maximum, tout en respectant les contraintes de rareté des ressources. Le problème peut se formuler de la façon suivante:

$$\begin{aligned}
 [1] \quad & \text{maximiser :} && z = 7x_1 + 9x_2 \\
 & \text{sous les contraintes :} && x_1 + x_2 \leq 8 \\
 & && x_2 \leq 4 \\
 & && 2x_1 + 3x_2 \leq 19 \\
 & && x_1 \geq 0, x_2 \geq 0
 \end{aligned}$$

On constate clairement l'apparition d'une matrice A (ici de dimensions 3×2), de deux vecteurs: \mathbf{b} (le second membre) appartenant à \mathbb{R}^3 , et \mathbf{c} (les coefficients de la fonction-objectif) appartenant à \mathbb{R}^2 , alors notre problème peut s'écrire comme suit:

$$\begin{aligned}
 [2] \quad & \text{maximiser:} && z = \mathbf{c} \cdot \mathbf{x} \\
 & \text{sous les contraintes :} && A \mathbf{x} \leq \mathbf{b} \\
 & && \mathbf{x} \geq 0.
 \end{aligned}$$

Ceci est, par définition, un *programme linéaire sous forme canonique*. Le formalisme algébrique nous offre une notation compacte qui facilite en pratique le calcul numérique – la méthode des tableaux s'appuie la dessus et la saisie des données pour un traitement par des moyens

informatiques est ainsi standardisée. On va supposer, pour ne pas trop charger la notation, que *toutes les conventions algébriques sont respectées* : ainsi par exemple dans [2] \mathbf{c} est un vecteur-ligne et \mathbf{x} un vecteur-colonne, même si parfois dans le texte ses coordonnées sont présentées en ligne.

EXEMPLE 1: (suite)

Une coopérative agricole négocie (c'est-à-dire veut minimiser) le prix d'un kilogramme de chaque composant pour acheter "en vrac" la totalité du stock. Comment déterminer ces prix pour que la vente "en vrac" soit au moins aussi intéressante que la vente des mélanges?

L'ensemble de données n'a pas changé mais maintenant on nous demande de déterminer les prix unitaires y_1, y_2, y_3 de trois "matières premières". Avec ces prix hypothétiques, le produit I aura la valeur $y_1 + 2y_3$ et le produit II $y_1 + y_2 + 3y_3$. Pour l'usine, il faut que ces valeurs ne soient pas inférieures aux profits respectifs des deux produits (soit 7 et 9 €). D'autre part, la coopérative veut minimiser le coût total d'achat qui s'élève à $8y_1 + 4y_2 + 19y_3$ euros. On obtient finalement le programme suivant :

$$\begin{aligned} \text{minimiser: } w &= 8y_1 + 4y_2 + 19y_3 \\ \text{sous: } & y_1 + 2y_3 \geq 7 \\ [3] \quad & y_1 + y_2 + 3y_3 \geq 9 \\ & y_1 \geq 0, y_2 \geq 0, y_3 \geq 0. \end{aligned}$$

Avec les notations algébriques précédemment utilisées, il peut être réécrit sous la forme suivante:

$$\begin{aligned} \text{minimiser: } w &= \mathbf{y} \mathbf{b} \\ [4] \quad \text{sous: } & \mathbf{y} \mathbf{A} \geq \mathbf{c} \\ & \mathbf{y} \geq \mathbf{0}. \end{aligned}$$

Ceci est aussi un programme linéaire sous *forme canonique*. L'important dans la forme canonique, c'est que toutes les variables sont astreintes à être positives ou nulles, et les contraintes sont des inégalités, étant donné que le sens d'une inégalité peut être changé par multiplication par -1 , et, puisque $\min(w) = -\max(-w)$, on peut changer le sens de l'optimisation. Il suffit donc de maximiser la fonction-objectif avec les coefficients opposés.

EXEMPLE 2:

Un modèle de voiture est assemblé dans un des trois ateliers situés dans les villes V_1, V_2 et V_3 . Le moteur qui équipe ce modèle est fourni par une des deux usines situées dans les villes U_1 et U_2 . Les besoins hebdomadaires des trois ateliers d'assemblage sont au moins 5, 4 et 3 moteurs. Chaque usine peut fournir au plus 6 moteurs. Le seul souci pour la direction est de minimiser le coût total de transport des moteurs entre les deux lieux de fabrication et les trois ateliers d'assemblage. Le tableau suivant donne les coûts unitaires (par moteur transporté) pour tous les trajets envisageables.

	\mathbf{a}	V_1	V_2	V_3
de				
U_1	38	27	48	
U_2	37	58	45	

Comment minimiser le coût total de transport en respectant l'offre et la demande ?

Pour présenter les données, l'on peut se servir du graphe biparti, orienté et valué.

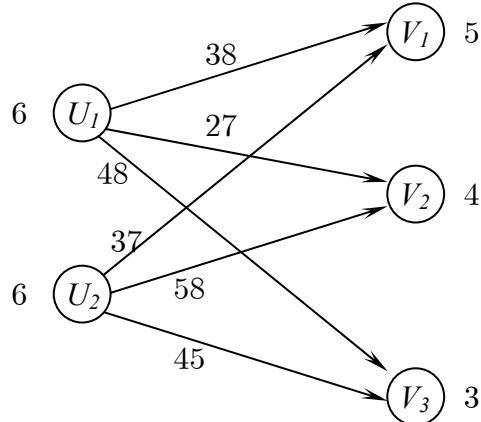


Figure 1. Les données de l' EXEMPLE 2

Il faut d'abord déterminer des quantités x_{ij} de moteurs transportés de la ville U_i ($i=1, 2$) à l'atelier V_j ($j=1, 2, 3$). Le coût total de transport représente le critère. Il y a deux types de contraintes:

- on ne peut pas dépasser les quantités disponibles dans chaque ville;
- on doit fournir au moins les quantités nécessaires pour chaque atelier.

On obtient donc la formulation suivante:

$$\begin{aligned}
 & \text{minimiser: } z = 38x_{1,1} + 27x_{1,2} + 48x_{1,3} + 37x_{2,1} + 58x_{2,2} + 45x_{2,3} \\
 & \text{sous:} \quad \begin{aligned} x_{1,1} + x_{1,2} + x_{1,3} &\leq 6 \\ x_{2,1} + x_{2,2} + x_{2,3} &\leq 6 \\ x_{1,1} + x_{2,1} &\geq 5 \\ x_{1,2} + x_{2,2} &\geq 4 \\ x_{1,3} + x_{2,3} &\geq 3 \end{aligned} \\
 [5] \quad & x_{1,1} \geq 0, \quad x_{1,2} \geq 0, \quad x_{1,3} \geq 0, \quad x_{2,1} \geq 0, \quad x_{2,2} \geq 0, \quad x_{2,3} \geq 0.
 \end{aligned}$$

Si nous considérons ce problème écrit sous forme canonique

$$\begin{aligned}
 & \text{minimiser: } z = \mathbf{c} \mathbf{x} \\
 [6] \quad & \text{sous:} \quad -\mathbf{A} \mathbf{x} \geq \mathbf{b} \\
 & \quad \mathbf{x} \geq \mathbf{0}.
 \end{aligned}$$

alors la matrice \mathbf{A} est la matrice d'incidence aux arcs (voir "Graphes et Applications") du graphe biparti sur la Figure 1.

Dans notre cas on constate que l'on peut remplacer les inégalités par des égalités (car l'offre est égale à la demande) et écrire:

$$\begin{aligned}
 & \text{minimiser: } z = \mathbf{c} \mathbf{x} \\
 [7] \quad & \text{sous:} \quad -\mathbf{A} \mathbf{x} = \mathbf{b} \\
 & \quad \mathbf{x} \geq \mathbf{0}.
 \end{aligned}$$

ce qui est, par définition, un *programme linéaire sous forme standard* (toutes les variables sont astreintes à être positives ou nulles et les contraintes sont des équations).

SUJET DE REFLEXION: Donner la formulation générale du problème de transport décri dans l'Exemple 2 (page 4) et justifier formellement que l'on peut toujours le considérer sous la forme [7], quelles que soient les valeurs de quantités offertes et demandées, et avec $\mathbf{c} \geq \mathbf{0}$.

EXEMPLE 2: (suite)

Etant difficile à résoudre par les moyens propres de l'entreprise, ce problème a été confié à un transporteur indépendant. Quant à la gestion de ce transport, on imagine de fixer des "prix d'achat" conventionnels (une sorte de caution) pour un moteur, payés par le transporteur indépendamment à chaque fabricant, et des "prix de vente" (une sorte de remboursement) payés par chaque usine d'assemblage au transporteur. Comment fixer ces deux prix de façon optimale dans ces nouvelles conditions économiques ?

Cette fois-ci on doit déterminer les prix p_i ($i=1, 2$) de l'achat pour les moteurs fournis par l'usine U_i , ainsi que le prix q_j ($j=1, 2, 3$) de vente d'un moteur à chaque atelier V_j . Pour que ces prix p_i et q_j soient compétitifs avec les coûts actuels de transport c_{ij} , il faut que $q_j - p_i \leq c_{ij}$. Le transporteur va ainsi gagner $5q_1 + 4q_2 + 3q_3$ grâce à la vente, mais il a dû investir $6p_1 + 6p_2$ pour acheter. Le programme linéaire se présente ainsi:

$$\begin{aligned}
 & \text{maximiser: } w = -6p_1 - 6p_2 + 5q_1 + 4q_2 + 3q_3 \\
 & \text{sous:} \quad q_1 - p_1 \leq 38 \\
 & \quad q_2 - p_1 \leq 27 \\
 [8] \quad & \quad q_3 - p_1 \leq 48 \\
 & \quad q_1 - p_2 \leq 37 \\
 & \quad q_2 - p_2 \leq 58 \\
 & \quad q_3 - p_2 \leq 45 \\
 & \quad p_1 \geq 0, p_2 \geq 0, q_1 \geq 0, q_2 \geq 0, q_3 \geq 0.
 \end{aligned}$$

Avec la convention $\mathbf{y} = (p_1, p_2, q_1, q_2, q_3)$, on peut aussi le noter, d'après [6], comme suit :

$$\begin{aligned}
 & \text{maximiser: } w = \mathbf{y} \mathbf{b} \\
 [9] \quad & \text{sous:} \quad -\mathbf{yA} \geq \mathbf{c} \\
 & \quad \mathbf{y} \geq \mathbf{0}.
 \end{aligned}$$

EXEMPLE 3:

Le régime nutritionnel d'un sportif devrait garantir au moins 9 unités de vitamine A et 19 unités de vitamine C par jour. On trouve sur le marché six produits (numérotés de 1 à 6) riches en ces vitamines. Un kilogramme de chacun de ces produits contient respectivement 1, 0, 2, 2, 1, 2 unités de vitamine A et 0, 1, 3, 1, 3, 2 unités de vitamine C et coûte respectivement 3€50, 3€00, 5€80, 5€00, 2€70, 2€20. Quels produits faut-il acheter, et en quelles quantités, pour se nourrir en minimisant les dépenses?

En choisissant x_i ($i=1, 2, \dots, 6$) la quantité (en kg) du produit i à acheter, on obtient le modèle suivant :

$$\begin{aligned}
 & \text{minimiser: } z = 35x_1 + 30x_2 + 58x_3 + 50x_4 + 27x_5 + 22x_6 \\
 & \text{sous:} \quad x_1 + 2x_3 + 2x_4 + 2x_5 + 2x_6 \geq 9 \text{ besoins en vit. A} \\
 [10] \quad & \quad x_2 + 3x_3 + x_4 + 2x_5 + 2x_6 \geq 19 \text{ besoins en vit. C} \\
 & \quad x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0.
 \end{aligned}$$

EXEMPLE 3: (suite)

Une firme pharmaceutique envisage la production de vitamines "pures" en pastilles. Elle veut fixer les prix de chaque unité de vitamine, pour maximiser le bénéfice d'une portion nécessaire vendue, en écartant en même temps la concurrence de tous les six produits cités.

Si l'on note par a (respectivement c) le prix d'une unité de vitamine A (respectivement C), on obtient le modèle suivant :

$$\begin{aligned}
 & \text{maximiser: } w = 9a + 19c \\
 & \text{sous:} \quad \begin{aligned} a &\leq 35 \\ c &\leq 30 \\ 2a + 3c &\leq 58 \\ 2a + c &\leq 50 \\ a + 3c &\leq 27 \\ 2a + 2c &\leq 22 \\ a \geq 0, \quad c \geq 0. \end{aligned} \\
 [11]
 \end{aligned}$$

2. Recherche d'une solution optimale

Dans les cas de deux variables seulement, on peut trouver la solution optimale d'un programme linéaire par une méthode géométrique. Nous allons l'expliquer sur le modèle créé dans l'Exemple 1 page 3. Le système de contraintes du programme [1] s'interprète comme l'intersection de cinq demi-plans fermés.

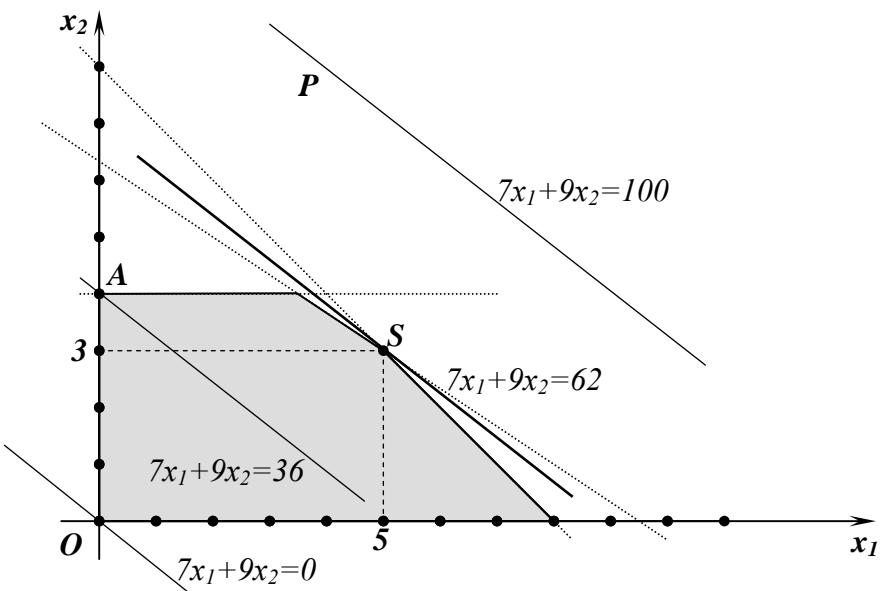


Figure 2. L'ensemble des solutions réalisables

Un polygone convexe (un pentagone dans ce cas) représente l'ensemble des *solutions réalisables* (admissibles) du modèle. Puisque l'interprétation de la fonction-objectif $z = 7x_1 + 9x_2$ exige l'introduction d'une troisième dimension, on peut se servir des coupes, illustrées sur la Figure 2 pour $z = 0, 36, 62$ et 100 .

Les droites qui en résultent passent respectivement par les points $O(0;0)$, $A(0;4)$, $S(5;3)$ et $P(4;8)$. Les translations (déplacements parallèles) de la droite $7x_1 + 9x_2 = 0$ dans la direction de son vecteur normal $[7;9]$, augmentent la valeur de la fonction-objectif. Il faut pouvoir trouver des solutions réalisables sur la droite déplacée (ce qui n'est pas le cas de la dernière droite). La plus grande valeur que l'on peut ainsi obtenir est 62 , quand la droite parallèle passe par le point $S(5;3)$. On découvre ainsi la solution optimale $x_1 = 5$, $x_2 = 3$ et sa valeur $z_{\max} = 62$.

Cette interprétation montre clairement la possibilité d'envisager une méthode combinatoire (le nombre des sommets du polygone étant toujours fini) de solution des programmes linéaires, qui a priori ont un caractère continu.

SUJET DE REFLEXION: Démontrer, en utilisant l'interprétation géométrique, que le problème non-linéaire suivant:

$$\text{minimiser } \{|x-1| + |3x-9|\}$$

où x est une variable réelle, peut être formulé comme un programme linéaire.

1.2.1. Standardisation

La résolution géométrique ne s'applique que dans de "petits" cas. En général, on doit effectuer les calculs numériques. Pour ce faire, la formulation avec "égalités" s'avère la plus adaptée. On va donc chercher la solution des programmes linéaires présentés sous la forme standard seulement. Dans certaines applications, on rencontre des formulations différentes, mais, dans tous les cas, on peut les ramener à une forme standard. Si, par exemple, les contraintes sont de type :

$$\sum_{i=1}^n a_{i,j} \leq b_j \quad (\text{ou} \quad \sum_{i=1}^n a_{i,j} \geq b_j), \quad \text{on introduit des variables d'écart } e_j \geq 0 \text{ telles que :}$$

$$\sum_{i=1}^n a_{i,j} + e_j = b_j \quad (\text{respectivement} \quad \sum_{i=1}^n a_{i,j} - e_j = b_j).$$

S'il manque la contrainte de positivité en x_i , il suffit de la remplacer par $x_i^+ - x_i^-$ avec $x_i^+ \geq 0$ et $x_i^- \geq 0$. Inversement, si les contraintes sont des équations de type :

$$\sum_{i=1}^n a_{i,j} = b_j$$

alors il suffit de les remplacer par les couples: $\sum_{i=1}^n a_{i,j} \leq b_j$ et $-\sum_{i=1}^n a_{i,j} \leq -b_j$

pour obtenir la forme canonique.

Enfin, on va supposer que la matrice A du programme linéaire sous forme standard, de dimensions $m \times n$, est de *rang m*. Si en réalité $rg(A) < m$, alors il y a soit redondance (dans un tel cas il suffit d'éliminer les lignes inutiles), soit incompatibilité entre les contraintes. Dans ce deuxième cas il y a soit une erreur de données numériques, soit une erreur de modélisation, voire même une incompatibilité avec la programmation linéaire.

SUJET DE REFLEXION: Démontrer que la matrice A du problème de transport de l'Exemple 2 page 4 dans sa formulation [7] est de rang 4 et que l'on peut éliminer n'importe quelle ligne. Généraliser.

1.2.2. Exemple de résolution "à la main"

Reprendons le programme linéaire [1] de l'Exemple 1 page 4. Après l'introduction de trois variables d'écart positives, que l'on va continuer à appeler x_3, x_4 et x_5 , il peut s'écrire sous la forme standard :

$$\begin{aligned} \text{maximiser: } z &= 7x_1 + 9x_2 \\ \text{sous: } & \begin{array}{rcl} x_1 + x_2 + x_3 & = & 8 \\ & x_2 + x_4 & = & 4 \\ & 2x_1 + 3x_2 + x_5 & = & 19 \\ & x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0. \end{array} \\ [12] \end{aligned}$$

Toutes les solutions du système d'équations dans ce cas forment une variété linéaire de

dimension $2=5(\text{variables})-3(\text{contraintes})$. Pour calculer une solution particulière, on peut choisir les valeurs pour deux variables pour en déduire les valeurs des trois autres et aussi de z . Dans notre exemple, on peut exprimer sans aucune difficulté x_3, x_4, x_5 et z en fonction de x_1 et x_2 :

$$\begin{aligned} [13] \quad & x_3 = 8 - x_1 - x_2 \\ & x_4 = 4 - x_2 \\ & \frac{x_5 = 19 - 2x_1 - 3x_2}{z = 0 + 7x_1 + 9x_2} \end{aligned}$$

Ce système peut être considéré comme un “dictionnaire” où les “mots” x_3, x_4, x_5 sont “expliqués” par des “mots” x_1 et x_2 . Formellement, on dit que x_3, x_4, x_5 sont les *variables de base* et x_1 et x_2 - les *variables hors-base*. En pratique il est inutile de déplacer les variables hors-base à droite. Les considérations précédentes sont valables aussi dans la formulation [13]. Si l'on note \mathbf{x}_B le vecteur des variables de base (dans l'ordre de leur apparition dans les lignes successives) et \mathbf{x}_N des variables hors-base, on peut remplacer

$$\begin{aligned} [14] \quad & \text{maximiser: } z = \mathbf{c} \cdot \mathbf{x} \quad & \text{maximiser: } z = \mathbf{c}_B \mathbf{x}_B + \mathbf{c}_N \mathbf{x}_N \\ & \text{sous: } \mathbf{A} \mathbf{x} = \mathbf{b} \quad \text{par [15]} \quad \text{sous: } \mathbf{B} \mathbf{x}_B + \mathbf{N} \mathbf{x}_N = \mathbf{b} \\ & \mathbf{x} \geq 0. \quad & \mathbf{x}_B \geq 0, \mathbf{x}_N \geq 0. \end{aligned}$$

Revenons à notre programme linéaire [13]. Une transformation suffit pour exprimer x_3, x_2, x_5 et z en fonction de x_1 et x_4 :

$$\begin{aligned} [16] \quad & x_3 = 4 - x_1 + x_4 \\ & x_2 = 4 - x_4 \\ & \frac{x_5 = 7 - 2x_1 + 3x_4}{z = 36 + 7x_1 - 9x_4} \end{aligned}$$

mais on ne pourra jamais exprimer x_1, x_3, x_5 et z en fonction de x_2 et x_4 . Ceci est dû au fait que la sous-matrice carrée \mathbf{B} de \mathbf{A} , qui contient ses colonnes 1, 3 et 5, n'est pas inversible :

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 1 \end{pmatrix} \quad \text{et} \quad \det(\mathbf{B}) = \begin{vmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 1 \end{vmatrix} = 0.$$

Une *base* est donc une m -uple ordonnée ($m = \text{nombre de ligne de } \mathbf{A} = \text{rg}(\mathbf{A})$), telle que la sous-matrice carrée \mathbf{B} , formée par des colonnes de \mathbf{A} avec les mêmes indices, est inversible. Lorsqu'aucune ambiguïté n'en résultera (le contexte indiquant lequel de ces objets est considéré), nous désignerons par la même lettre \mathbf{B} une base en tant que:

- l'ensemble (ordonné) des variables de base (p.ex. $\mathbf{B}=\{x_3, x_2, x_5\}$ dans [16]);
- l'ensemble (ordonné) des indices de ces variables (p.ex. $\mathbf{B}=\{3, 2, 5\}$ dans [16]);
- la sous-matrice carrée de \mathbf{A} , inversible, qui contient des colonnes indexées par \mathbf{B}

(par exemple $\mathbf{B} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 1 \end{pmatrix}$ composée des colonnes 3, 2, 5 dans [12]).

1.2.3. Définitions et notations

Pour obtenir rapidement une solution réalisable à partir d'un dictionnaire, on peut annuler les variables hors-base. Ceci permet de déterminer très facilement les valeurs des variables

de base. La solution ainsi obtenue s'appelle la *solution de base* associée à la base \mathbf{B} . Par exemple dans [13] c'est $\mathbf{x}=(0, 0, 8, 4, 19)$ et dans [16] c'est $\mathbf{x}=(0, 4, 4, 0, 7)$. En général, pour une forme standard ([15] ou [16]), $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$ et $\mathbf{x}_N = \mathbf{0}$.

Quand $\mathbf{x}_B \geq \mathbf{0}$, la solution de base \mathbf{B} (ou, tout simplement, la base) est dite *admissible* (ou *réalisable*). Si, en plus, une au moins des composantes de \mathbf{x}_B est nulle, alors on dit que la solution de base (ou base) est *dégénérée*. Les deux solutions citées sont les solutions de base réalisables non dégénérées. Par contre, la solution associée à la base $\mathbf{B}=\{2,4,5\}$, $\mathbf{x}=(0,8,0,-4,-5)$, est non admissible (et non dégénérée). Il est inutile de déplacer constamment les variables des deux côtés. Les transformations simples, correspondant à la multiplication par \mathbf{B}^{-1} , permettent de réécrire les programmes [13] et [16] sous la forme:

$$\begin{aligned} \text{maximiser: } & z - z_0 = c'_{\mathbf{N}} \mathbf{x}_{\mathbf{N}} \\ [17] \quad \text{sous: } & \mathbf{x}_B + \mathbf{N}' \mathbf{x}_{\mathbf{N}} = \mathbf{b} \\ & \mathbf{x}_B \geq \mathbf{0}, \quad \mathbf{x}_{\mathbf{N}} \geq \mathbf{0}. \end{aligned}$$

En pratique il est préférable de garder toujours l'ordre original des variables au lieu de les permuter à chaque fois. Cette forme standard où

- la matrice associée aux variables de base est la matrice identité (à une permutation près), et
- les coefficients de la fonction-objectif associés aux variables de base sont nuls,

est appelée *forme canonique par rapport à la base \mathbf{B}* .

Il est aussi commode d'écrire la fonction-objectif dans le même style que le système des contraintes (c'est-à-dire variables à gauche – constantes à droite). Ainsi les formulations [15] et [18] remplacent celles présentées par [13] et [16]. Elles présentent notre système sous la forme canonique par rapport à la base $\mathbf{B}=\{3,4,5\}$ et $\mathbf{B}=\{3,2,5\}$.

$$\begin{array}{rcl} x_1 + x_2 + x_3 & = & 8 \\ [18] \quad x_2 + x_4 & = & 4 \\ 2x_1 + 3x_2 + x_5 & = & 19 \\ \hline z - 7x_1 + 9x_2 & = & 0 \end{array} \quad \begin{array}{rcl} x_1 + x_3 - x_4 & = & 4 \\ [19] \quad x_2 + x_4 & = & 4 \\ 2x_1 + -3x_4 + x_5 & = & 7 \\ \hline z - 7x_1 + 9x_4 & = & 36 \end{array}$$

La valeur actuelle de la fonction-objectif donnée par la deuxième solution de base est 36, car $x_1=x_4=0$ (ce sont les variables hors-base). Seule une augmentation de x_1 peut améliorer ce résultat, car une augmentation de x_4 implique une baisse de la valeur de la fonction-objectif. Il faut donc introduire x_1 dans la nouvelle base en faisant sortir une variable de la base actuelle (une base contient toujours trois éléments dans notre cas). Un choix judicieux est nécessaire pour que la nouvelle solution de base soit réalisable. Ce problème est résolu dans la méthode du simplexe présentée par la suite. Ici nous admettons que la nouvelle base est $\mathbf{B}=\{3,2,5\} \cup \{1\} \setminus \{5\} = \{3,2,1\}$ et la forme canonique par rapport à cette base est :

$$\begin{aligned} [20] \quad & + x_3 + 0.5x_4 - 0.5x_5 = 0.5 \\ & x_2 + x_4 = 4 \\ & x_1 + -1.5x_4 + 0.5x_5 = 3.5 \\ \hline z & - 1.5x_4 + 3.5x_5 = 60.5 \end{aligned}$$

On voit qu'une augmentation de x_4 peut encore améliorer la valeur de la fonction-objectif. Mettons cette variable à la place de x_3 dans la base précédente et la forme cano-

nique par rapport à la nouvelle base $\mathbf{B} = \{3, 2, 5\} \cup \{4\} \setminus \{3\} = \{4, 2, 1\}$ est:

$$\begin{array}{rcl} & 2x_3 - x_4 - x_5 = 1 \\ [21] & x_2 - 2x_3 + x_5 = 3 \\ & x_1 + 3x_3 - x_5 = 5 \\ \hline & z + 3x_3 + 2x_5 = 62 \end{array}$$

Tous les coefficients de la fonction-objectif sont négatifs ou nuls alors plus aucune augmentation n'est possible. Il est donc clair que la solution $x_1=5, x_2=3, x_3=0, x_4=1, x_5=0$ de valeur 62 est optimale.

Comme nous avons pu le constater, un programme linéaire écrit sous la forme canonique par rapport à une base réalisable est intéressant pour trois raisons:

1° la solution de base associée se lit immédiatement: $x_B = b; x_N = 0;$

2° l'évaluation de la fonction-objectif pour cette solution se lit immédiatement:

$$z = z_0 \text{ (c'est la constante en bas à droite);}$$

3° on peut facilement vérifier si la solution courante est optimale ou pas.

1.2.4. Itération du simplexe

Nous allons décrire l'itération générale de la méthode du simplexe, qui peut être considéré comme une adaptation de la *méthode de Gauss* de résolution de systèmes linéaires. Appelée traditionnellement "opération de pivotage", elle consiste à transformer une forme [22] canonique par rapport à une base réalisable \mathbf{B} en une forme [23] canonique par rapport à une autre base \mathbf{B}' , qui diffèrent d'un seul élément ($|\mathbf{B} \cap \mathbf{B}'| = m-1$) (d'où le nom de la méthode – simplexe) :

$$\begin{array}{lll} \text{maximiser: } & z - z_0 = c_N x_N & \text{maximiser: } z - z'_0 = c'_N x_{N'} \\ [22] \quad \text{sous: } & x_B + N x_N = b & [23] \quad \text{sous: } x_{B'} + N' x_{N'} = b \\ & x_B \geq 0, \quad x_N \geq 0. & x_{B'} \geq 0, \quad x_{N'} \geq 0. \end{array}$$

Le problème essentiel est de choisir judicieusement le pivot pour que la nouvelle base soit de nouveau réalisable et pour que la valeur de la solution croisse. Techniquement il y a deux étapes: le choix du pivot et ensuite le pivotage. En voici la description détaillée:

Soit s l'indice du plus grand coefficient de $z - z_0$ dans [22], c.-à-d. $c_s = \max\{c_1, c_2, \dots, c_n\}$. On étudie seulement le cas $c_s > 0$, car autrement la solution courante est optimale. Etant donné que $\mathbf{b} \geq \mathbf{0}$ (la solution courante est réalisable!) seulement les $a_{i,s} > 0$ sont des candidats pour devenir le pivot (la solution recherchée doit être réalisable). Soit r l'indice tel que :

$$b_r/a_{rs} = \min \{ b_i/a_{is}; i=1, 2, \dots, m; a_{rs} > 0 \}.$$

L'élément a_{rs} est le pivot. L'opération de pivotage met le [22] sous la forme [23], canonique par rapport à la nouvelle base $\mathbf{B}' = \mathbf{B} \cup \{s\} \setminus \{r\}$.

Dans la pratique, pour effectuer rapidement une série de pivotages, il est commode d'utiliser les tableaux. Voici l'ensemble des transformations précédentes sous la forme de tableaux (le pivot est encadré et les flèches désignent les variables entrant et sortant de la base):

1							
	z	x_1	x_2	x_3	x_4	x_5	
x_3	0	1	1	1	0	0	8
$\leftarrow x_4$	0	0	1	0	1	0	4
x_5	0	2	3	0	0	1	19
$\hline z$	1	-7	-9	0	0	0	0

2							
	z	x_1	x_2	x_3	x_4	x_5	
x_3	0	1	0	1	-1	0	4
x_2	0	0	1	0	1	0	4
$\leftarrow x_5$	0	2	0	0	-3	1	7
$\hline z$	1	-7	0	0	9	0	36

3							4						
		\downarrow											
\leftarrow	x_3	0	0	0	1	0.5		-0.5	0.5	x_4	0	0	0
	x_2	0	0	1	0	1		0	4	x_2	0	0	1
	x_1	0	1	0	0	-1.5	0.5		3.5	x_1	0	1	0
	z	1	0	0	0	-1.5	3.5	60.5		z	1	0	0

1.2.5. Algorithme du simplexe

L'algorithme du simplexe se déroule généralement en deux phases. La *Phase I* consiste à chercher si le système d'équations définissant les contraintes du programme linéaire :

$$\begin{array}{ll} \text{maximiser:} & z - z_0 = \mathbf{c} \mathbf{x} \\ [24] \quad \text{sous:} & \mathbf{A} \mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{array}$$

a une solution réalisable ou pas. Dans ce deuxième cas l'algorithme est terminé, dans le premier la *phase I* fournit une solution de base réalisable. La *phase II* consiste à trouver une solution optimale de base à partir de la solution réalisable fourni par la phase I. Etant donné que les deux phases sont similaires on préfère écrire d'abord la phase II:

Phase II de l'algorithme du simplexe:

- le problème est un problème de maximisation [24];
- la forme est canonique par rapport à une base réalisable, mise dans le tableau correspondant au système:

$$\begin{array}{l} \mathbf{x}_B + N \mathbf{x}_N = \mathbf{b} \\ z - \mathbf{c}_N \mathbf{x}_N = z_0 \end{array}$$

(Attention! Les tableaux utilisent les opposés des coefficients de la fonction-objectif.)

début

tant qu'il existe un coefficient strictement positif dans la fonction-objectif faire:

début

choisir une variable x_s telle que $c_s > 0$
(x_s entre dans la base)

(Attention! Dans le tableau $-c_s < 0$)

si tous les $a_{is} \leq 0$ STOP:

le programme linéaire n'a pas de solution
optimale finie c.-à-d. $z_{\max} = +\infty$

sinon

début

soit $d = \min \{ b_i / a_{is} : a_{is} > 0, i=1, \dots, m\};$
soit r une des lignes qui a donné la valeur de d :
Pivoter (r,s)

fin

fin

fin

Pivoter (r, s): (cette fonction remet un programme linéaire sous forme canonique par rapport à la nouvelle base: la variable x_s entre dans la base, et celle qui est actuellement dans la base avec un coefficient 1 dans la ligne r en sort)

début

diviser la ligne r par a_{rs}

pour $k=1$ à m et $k \neq r$

- retrancher à la ligne k , a_{ks} fois la nouvelle ligne r ;
 - retrancher c_j fois à la fonction-objectif la nouvelle ligne r ;
- fin pour*

fin

On peut choisir dans la fonction-objectif un coefficient strictement positif quelconque. Le choix du plus grand était proposé intuitivement en 1947 par G.B.Dantzig, le “père” de la méthode, pour accélérer la convergence (ce n'est pas toujours le cas comme l'on le constate dans l'Exemple 1).

Les transformations citées sont données par les formules suivantes:

$$a'_{rj} = a_{rj} / a_{rs} \quad j = 1, 2, \dots, n;$$

$$a'_{ij} = a_{ij} - a_{is} a'_{rj} \quad i = 1, 2, \dots, m; \quad i \neq r; \quad j = 1, 2, \dots, n;$$

$$b'_r = b_r / a_{rs}$$

$$b'_i = b_i - b'_r a'_{rj} \quad i = 1, 2, \dots, m; \quad i \neq r;$$

$$c'_j = c_j - c_s a'_{rj}$$

$$z'_0 = z_0 + b'_r c_s;$$

soit k l'indice de la r -ième variable dans \mathbf{B} :

$$\mathbf{B}' = \mathbf{B} \cup \{s\} \setminus \{k\}.$$

1.2.6. Justification de l'algorithme du simplexe

Supposons qu'on obtient la forme :

$$\begin{aligned} [25] \quad & \text{maximiser:} && z - z_0 = \mathbf{c}_N \mathbf{x}_N \\ & \text{sous:} && \mathbf{x}_B + N \mathbf{x}_N = \mathbf{b} \\ & && \mathbf{x}_B \geq 0, \quad \mathbf{x}_N \geq 0. \end{aligned}$$

avec $\mathbf{c} \leq \mathbf{0}$. Une solution réalisable quelconque (qu'elle soit de base ou pas) vérifie $\mathbf{x}_N \geq 0$, alors on a : $z - z_0 = \mathbf{c}_N \mathbf{x}_N \leq 0$ c.-à-d. $z \leq z_0$. La solution de base \mathbf{B} vérifie $z = z_0$, ce qui prouve l'optimalité.

Supposons que dans la forme [25] il existe un indice s tel que $c_s > 0$ et tous les $a_{is} \leq 0$. En augmentant x_s autant que l'on veut, on obtient toujours une solution réalisable:

$$x'_B = b - (a_{1s}, a_{2s}, \dots, a_{ms})x_s \geq 0$$

de valeur $z = z_0 + c_s x_s$ aussi grande que l'on veut. Le programme linéaire n'a pas de solution optimale finie c'est-à-dire $z_{\max} = +\infty$.

1.2.7. Finitude de l'algorithme du simplexe

Le nombre de bases d'un programme linéaire est fini, car borné trivialement par le choix de m parmi n colonnes. Il est donc évident que si l'algorithme ne finit pas, alors il doit cycler, c'est-à-dire revenir périodiquement à la même base. Ceci n'est pas possible quand les valeurs associées aux solutions de base successives croissent strictement, ce qui est le cas des solutions de base non-dégénérées ($z' = z_0 + b_r c_s$, alors $b_r > 0$ implique $z' > z_0$; chaque solution est strictement meilleure que la précédente).

On rencontre dans la pratique des cas de dégénérescence qui proviennent du modèle (le problème d'affectation) ou étant des résultats des arrondis; mais le cyclage est un phénomène très rare et il est très difficile de construire des exemples. Aujourd'hui il existe des règles qui garantissent la finitude dans le cas général. La plus récente (R.G. Bland – 1977) dite "du plus petit indice" consiste à choisir:

- le plus petit s pour lequel $c_s > 0$ comme l'indice de variable qui entre dans la base ;
- le plus petit r (parmi les indices des lignes qui ont donné la valeur d) comme l'indice de la ligne sur laquelle on effectue le pivotage.

A noter aussi l'existence de la méthode des perturbations (addition des petits ε au second membre) qui donne une règle lexicographique. (voir V. Chvatal).

1.2.8. Phase I (initialisation du simplexe)

Nous pouvons toujours, grâce à quelques transformations, présenter notre modèle linéaire sous la forme standard

$$\begin{aligned} [26] \quad \text{minimiser: } \quad & z - z_0 = \mathbf{c} \mathbf{x} \\ \text{sous: } \quad & \mathbf{A} \mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

Le but de la phase I est de vérifier si le système de contraintes admet une solution réalisable et, si c'est le cas, de mettre le programme [26] sous forme canonique par rapport à une base réalisable, pour pouvoir ensuite effectuer la phase II, c'est-à-dire chercher une solution optimale.

Pour atteindre ce but on utilise des *variables artificielles* et on applique la phase II au *programme linéaire auxiliaire*. Voici la phase I en détail :

Supposons que $\mathbf{b} \geq \mathbf{0}$ dans [26] (sinon il suffit de multiplier les deux membres d'une même contrainte par $-I$). Nous allons associer à chaque équation i ($i=1, 2, \dots, m$) du système une variable v_i dite *variable artificielle*. Soit $\mathbf{v} = (v_1, v_2, \dots, v_m)$. Considérons alors le programme suivant:

$$\begin{aligned} [27] \quad \text{minimiser: } \quad & w = v_1 + \dots + v_m \\ \text{sous: } \quad & \mathbf{A} \mathbf{x} + \mathbf{v} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \quad \mathbf{v} \geq \mathbf{0}. \end{aligned}$$

appelé *programme auxiliaire*. Notons $\mathbb{I} = (1, 1, \dots, 1)$ le m -vecteur-ligne composé des 1.

On peut utiliser $\mathbf{v} = \mathbf{b} - \mathbf{A} \mathbf{x}$ pour présenter ce programme sous forme canonique par rapport à la base auxiliaire réalisable

$$\begin{aligned} [28] \quad \text{minimiser: } \quad & w - \mathbb{I} \mathbf{b} = -\mathbb{I} \mathbf{A} \mathbf{x} \\ \text{sous: } \quad & \mathbf{v} + \mathbf{A} \mathbf{x} = \mathbf{b} \\ & \mathbf{v} \geq \mathbf{0}, \quad \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

avec la solution de base évidente : $\mathbf{x}=\mathbf{0}; \mathbf{v}=\mathbf{b} \geq 0$ et une évaluation $w_0=\mathbf{b}^T\mathbf{b}=b_1+b_2+\dots+b_m$.

On peut le résoudre en appliquant la phase II du simplexe. Le programme [28] a toujours une solution optimale finie, car $w=v_1+v_2+\dots+v_m \geq 0$. Par conséquent, la phase II du simplexe fournira en un nombre fini d'itérations une solution optimale de base de [28].

Théorème 1:

Le programme linéaire [26] admet une solution réalisable si et seulement si la solution optimale du programme linéaire [27] est de valeur nulle.

DEMONSTRATION:

\Rightarrow : Si $\underline{\mathbf{x}}$ est une solution réalisable de [26], alors $(\underline{\mathbf{x}}; \mathbf{v}) = (\underline{\mathbf{x}}; \mathbf{0})$ est une solution réalisable de [27] de valeur $w = v_1+v_2+\dots+v_m = 0$ et, puisque $w \geq 0$, on a $w_{min} = 0$.

\Leftarrow : Si $w_{min} = 0$ pour une solution réalisable $(\underline{\mathbf{x}}; \mathbf{v})$ de [27], alors $v_i \geq 0$ implique $\underline{v}_i = 0$ pour tous les $i=1, 2, \dots, m$; c'est-à-dire $\mathbf{v}=\mathbf{0}$; cela signifie que $\mathbf{x}=\underline{\mathbf{x}}$ est une solution réalisable de [26]. ■

Etant donné que la démonstration du théorème précédent faisait abstraction de la méthode de résolution, il va falloir encore justifier que:

Théorème 2:

Si le programme linéaire [27] admet une solution réalisable, alors il admet une solution de base réalisable.

DEMONSTRATION:

En appliquant la phase II du simplexe il est évident que le programme auxiliaire [27] admet une solution de base optimale, mais il faut encore que cette base ne comporte aucune variable artificielle. Supposons donc que la variable v_r soit une variable de base optimale de [27]. Puisque $w = 0$ d'après le Théorème 1., alors on se trouve dans un cas de dégénérescence $v_r=b_r=0$. Nous avons considéré la matrice \mathbf{A} de [26] de plein rang, donc il existe dans la ligne r un élément $a_{rs} \neq 0$. Une opération de pivotage autour de cet élément fait entrer la variable x_s dans la base et en fait sortir v_r (cette opération change la base mais non la solution de base associée!). On peut ainsi faire sortir de la base optimale de [27] toutes les variables artificielles. ■

De ces résultats et de la finitude de l'algorithme du simplexe on déduit:

Théorème fondamental de la programmation linéaire:

- *Si un programme linéaire admet une solution réalisable, alors il admet une solution de base réalisable.*
- *Si un programme linéaire admet une solution optimale, alors il admet une solution de base optimale.*
- *Si un programme linéaire, dont la fonction-objectif à maximiser (à minimiser) est bornée supérieurement (inférieurement), alors il admet une solution de base optimale.*

1.2.8. Remarques sur la Phase I

1) Dans la pratique, il n'est pas toujours nécessaire d'ajouter une variable artificielle à chaque ligne de contraintes. Il suffit d'en choisir un nombre minimal pour qu'une matrice unitaire (à une permutation près) apparaisse.

2) Si le système possède une solution mais la matrice \mathbf{A} n'est pas de plein rang, alors la phase I met en évidence les lignes que l'on peut supprimer. Ceci découle directement de la démonstration du Théorème 2.

3) La méthode du simplexe présentée – dite “des deux phases” – n'est pas unique. On peut signaler aussi la méthode du “grand M” ou encore “la méthode duale”.

1.2.9. L'algorithme révisé du simplexe

Un examen attentif de la méthode présentée montre qu'il s'agit de transformer une forme [24] canonique par rapport à une base réalisable \mathbf{B} en une forme [25] canonique par rapport à une autre base \mathbf{B}' :

$$\begin{array}{ll} \text{maximiser: } z - z_0 = c_N x_N & \text{maximiser: } z = c_B x_B + c_N x_N \\ [29] \text{sous: } \begin{aligned} x_B + N x_N &= b \\ x_B \geq 0, \quad x_N \geq 0. \end{aligned} & \text{sous: } \begin{aligned} B x_B + N x_N &= b \\ x_B \geq 0, \quad x_N \geq 0. \end{aligned} \\ \\ \text{maximiser: } z - z'_0 = c'_N x_{N'} & \\ [30] \text{sous: } \begin{aligned} x_{B'} + N' x_{N'} &= b \\ x_{B'} \geq 0, \quad x_{N'} \geq 0. \end{aligned} & \end{array}$$

Puisque le problème essentiel est de déterminer si la base courante est optimale ou pas, l'opération de pivotage n'est pas nécessaire. Considérons donc les deux programmes linéaires :

$$\begin{array}{ll} \text{maximiser: } z = c x & \text{maximiser: } z = c_B x_B + c_N x_N \\ [31] \text{sous: } \begin{aligned} A x &= b \\ x \geq 0. \end{aligned} & \text{sous: } \begin{aligned} B x_B + N x_N &= b \\ x_B \geq 0, \quad x_N \geq 0. \end{aligned} \\ \\ \text{maximiser: } z - z_0 = c' x & \text{maximiser: } z - z_0 = c'_N x_N \\ [32] \text{sous: } \begin{aligned} A' x &= b' \\ x \geq 0. \end{aligned} & \text{sous: } \begin{aligned} x_B + N' x_{N'} &= b' \\ x_B \geq 0, \quad x_{N'} \geq 0. \end{aligned} \end{array}$$

ce dernier sous la forme canonique par rapport à la base \mathbf{B} .

Les relations entre eux sont les suivantes:

$$\begin{array}{ll} [33] \quad A' = B^{-1} A & \Leftrightarrow A = B A' \\ [34] \quad b' = B^{-1} b & \Leftrightarrow b = B b' \end{array}$$

et, comme $c' \mathbf{B} = \mathbf{0}$ alors il existe un vecteur $\pi \in \mathbb{R}^m$, dit vecteur-multiplicateur, tel que:

$$\begin{array}{ll} [35] \quad \pi B = c_B & (\Leftrightarrow \pi = c_B B^{-1}); \\ [36] \quad c' = c - \pi A & \\ [37] \quad z_0 = \pi b & \end{array}$$

En conclusion, pour vérifier si la base courante \mathbf{B} est optimale, il suffit de:

- 1° calculer, d'après [35], le vecteur-multiplicateur π ;
- 2° calculer c' d'après [36]; si $c' \leq \mathbf{0}$ alors la base courante \mathbf{B} est optimale; sinon choisir une variable x_s telle que $c_s > 0$ (x_s entre dans la base);
- 3° calculer la nouvelle colonne s , d'après [33] : $A'_s = B^{-1} A_s$ ($\Leftrightarrow A_s = B A'_s$)
si tous les $a'_{is} \leq 0$ alors **STOP**: le programme linéaire n'a pas de solution optimale finie c'est-à-dire $z_{\max} = +\infty$, sinon :
- 4° calculer le nouveau second membre b' , d'après [34];
soit $d = \min \{ b' / a'_{is}; a'_{is} > 0, i=1, \dots, m \}$; soit r une des lignes qui a donné la valeur de d ;
soit k l'indice de la r -ième variable dans \mathbf{B} : la nouvelle base est $\mathbf{B}' = \mathbf{B} \cup \{s\} \setminus \{k\}$.

Notons que l'algorithme révisé du simplexe n'apporte aucune modification fondamentale. Seule l'opération de pivotage a été oubliée – on revient à chaque itération au programme initial.

EXEMPLE 4:

En utilisant l'algorithme de simplexe et ensuite le simplexe révisé, résoudre par la méthode des deux phases le programme linéaire suivant :

$$\begin{aligned} \text{minimiser: } & z = 6x_1 + 7x_2 + 4x_3 \\ \text{sous: } & 4x_1 + 5x_2 + x_3 \leq 23 \\ & 5x_1 + 6x_2 + 2x_3 \geq 28 \\ & x_1 \geq 0, x_2 \geq 0, x_3 \geq 0. \end{aligned}$$

SOLUTION: Après avoir mis sous la forme standard on remarque que la phase d'amorçage s'impose et qu'une variable artificielle suffit. La méthode des tableaux avec le critère de Dantzig donne :

PHASE I						
1						
\leftarrow	x_1	x_2	x_3	e_1	e_2	v
e_1	4	5	1	1	0	23
V	5	6	2	0	-1	28
I	-5	-6	-2	0	1	0
II	6	7	4	0	0	x
						0

PHASE II						
2						
\leftarrow	x_1	x_2	x_3	e_1	e_2	v
x_2	0.8	1	0.2	0.2	0	0
v	0.2	0	0.8	-1.2	-1	1
I	-0.2	0	-0.8	1.2	1	0
II	0.4	0	2.6	-1.4	0	x
						-32.2

3						
\leftarrow	x_1	x_2	x_3	e_1	e_2	v
x_2	0.75	1	0	0.5	0.25	-0.25
x_3	0.25	0	1	-1.5	1.25	1.25
I	0	0	0	0	0	1
II	-0.25	0	0	2.5	3.25	x
						0

4						
\leftarrow	x_1	x_2	x_3	e_1	e_2	
x_2	0	1	-4	5	4	3
x_1	1	0	4	-6	-5	2
II	0	0	1	1	2	-33

Le *simplexe révisé* (avec la première rangée rencontrée): en partant du premier tableau, $B=\{4;6\}$ n'est pas optimale, x_1 entre dans la base et $B'=\{4;6\}\cup\{I\}\setminus\{6\}=\{4;I\}$; le vecteur multiplicateur $\pi=(0;1)$ (la phase I) et cette base est optimale car $c'=(0;0;0;0;0;-1)$; on passe à la phase II : le vecteur multiplicateur $\pi=(0;-6)$; on recalcule le vecteur $c'=(0;0.2;-1.6;0;0.8)$ et cette base n'est pas optimale; x_2 entre dans la base. On cherche le nouveau second membre et la nouvelle deuxième colonne:

$$\begin{cases} b_1 + 4b_2 = 23 \\ 5b_2 = 28 \end{cases} \Leftrightarrow \begin{cases} b_1 = 0,6 \\ b_2 = 5,6 \end{cases}$$

$$\begin{cases} a_1 + 4a_2 = 5 \\ 5a_2 = 6 \end{cases} \Leftrightarrow \begin{cases} a_1 = 0,2 \\ a_2 = 1,2 \end{cases}$$

et puisque $b_1/a_1 < b_2/a_2$, le pivot se trouve dans la première ligne et c'est donc la première variable (dans l'ordre) qui sort de la base: $B'=\{4;I\}\cup\{2\}\setminus\{4\}=\{2;I\}$; on cherche le vecteur multiplicateur π .

$$(\pi_I; \pi_2) \begin{pmatrix} 5 & 4 \\ 6 & 5 \end{pmatrix} = (-7; -6)$$

donc $\pi=(1;-2)$ et cette base est optimale car $c'=(0;0;-1;-1;-2)$. On retrouve la solution optimale :

$$\begin{pmatrix} 5 & 4 \\ 6 & 5 \end{pmatrix} \begin{pmatrix} x_2 \\ x_1 \end{pmatrix} = \begin{pmatrix} 23 \\ 28 \end{pmatrix} \text{ donc } \begin{pmatrix} x_2 \\ x_1 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

de valeur -33.

3. La dualité

1.3.1. Définition et premiers résultats

Soit le programme linéaire (qui sera appelé *primal*) écrit sous forme canonique:

$$\begin{array}{ll} \text{maximiser:} & z = \mathbf{c} \cdot \mathbf{x} \\ [38] \quad \text{sous:} & \mathbf{A} \mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{array}$$

On appelle *dual* de ce programme linéaire (d'après J. Von Neumann), le programme linéaire suivant:

$$\begin{array}{ll} \text{minimiser:} & w = \mathbf{y} \cdot \mathbf{b} \\ [39] \quad \text{sous:} & \mathbf{y} \cdot \mathbf{A} \geq \mathbf{c} \\ & \mathbf{y} \geq \mathbf{0}. \end{array}$$

Les trois premiers exemples de modèles classiques (voir les pages 3 à 7) illustrent clairement que deux programmes linéaires duals ne constituent pas deux problèmes distincts mais deux aspects du même problème.

On note que :

- les variables du dual sont en bijection avec les contraintes du primal tandis que les contraintes du dual sont en bijection avec les variables du primal;
- le second membre de l'ensemble des contraintes du primal donne fournit les coefficients de la fonction-objectif duale et les coefficients de la fonction-objectif primale forment le second membre de l'ensemble des contraintes du dual.

Voici cette correspondance dans le cas de l'Exemple 1 :

$$\begin{array}{ll} \text{maximiser :} & z = 7x_1 + 9x_2 \\ \text{sous:} & x_1 + x_2 \leq 8 \\ & x_2 \leq 4 \\ & 2x_1 + 3x_2 \leq 19 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{array} \quad \begin{array}{ll} \text{minimiser:} & w = 8y_1 + 4y_2 + 19y_3 \\ \text{sous:} & y_1 \geq 0 \\ & y_2 \geq 0 \\ & y_3 \geq 0 \\ & y_1 + 2y_3 \geq 7 \\ & y_1 + y_2 + 2y_3 \geq 9 \end{array}$$

Le théorème suivant justifie l'utilisation du terme *dual*.

Théorème 4:

Le dual du dual est le primal.

DEMONSTRATION:

En partant du programme [34], on peut le réécrire:

$$\begin{array}{ll} \text{maximiser:} & w = -\mathbf{y} \cdot \mathbf{b} \\ [40] \quad \text{sous:} & -\mathbf{y} \cdot \mathbf{A} \leq -\mathbf{c} \\ & \mathbf{y} \geq \mathbf{0}. \end{array}$$

En appliquant la définition du programme dual on obtient:

$$\begin{array}{ll} \text{minimiser:} & v = -\mathbf{c} \cdot \mathbf{x} \\ [41] \quad \text{sous:} & -\mathbf{A} \mathbf{x} \geq -\mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{array}$$

ce qui est équivalent à [38]. ■

Cette démonstration met en évidence le fait qu'il est commode de savoir trouver le dual d'un programme linéaire sans avoir à écrire d'abord ce programme sous forme canonique. Pour cela, étudions encore le passage au dual du programme linéaire sous forme standard:

$$[42] \quad \begin{array}{ll} \text{maximiser:} & z = \mathbf{c} \mathbf{x} \\ \text{sous:} & \mathbf{A} \mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{array}$$

Il peut s'écrire:

$$[43] \quad \begin{array}{ll} \text{maximiser:} & z = \mathbf{c} \mathbf{x} \\ \text{sous:} & \mathbf{A} \mathbf{x} \leq \mathbf{b} \\ & -\mathbf{A} \mathbf{x} \leq -\mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{array}$$

et son dual est:

$$[44] \quad \begin{array}{ll} \text{minimiser:} & \mathbf{w} = \mathbf{y}' \mathbf{b} - \mathbf{y}'' \mathbf{b} \\ \text{sous:} & \mathbf{y}' \mathbf{A} - \mathbf{y}'' \mathbf{A} \geq \mathbf{c} \\ & \mathbf{y}' \geq \mathbf{0}, \mathbf{y}'' \geq \mathbf{0}. \end{array}$$

En posant $\mathbf{y} = \mathbf{y}' - \mathbf{y}''$ on retrouve la forme:

$$[45] \quad \begin{array}{ll} \text{minimiser:} & \mathbf{w} = \mathbf{y} \mathbf{b} \\ \text{sous:} & \mathbf{y} \mathbf{A} \geq \mathbf{c} \\ \text{où } \mathbf{y} \text{ est non astreinte!} & \end{array}$$

L'ensemble des transformations est synthétisé dans le schéma suivant:

<i>programme: fonction:</i>	<i>primal (dual)</i>	\Leftrightarrow	<i>dual (primal)</i>
	$\hat{\text{à}} \text{ maximiser}$	\Leftrightarrow	$\hat{\text{à}} \text{ minimiser}$
<i>i^{ème} contrainte:</i>	$\leq b_i$	\Leftrightarrow	≥ 0
	$= b_i$	\Leftrightarrow	<i>i^{ème} variable:</i> non astreinte
	$\geq b_i$	\Leftrightarrow	≤ 0
<i>j^{ème} variable:</i>	≥ 0	\Leftrightarrow	$\geq c_j$
	<i>non astreinte</i>	\Leftrightarrow	<i>j^{ème} contrainte:</i> $= c_j$
	≤ 0	\Leftrightarrow	$\leq c_j$

SUJET DE REFLEXION: Vérifier que ce schéma peut être également lu de la droite vers la gauche !

2.3.2. Relation entre les valeurs des fonctions-objectifs des programmes linéaires duals

Théorème 5:

Soit un couple de programmes linéaires duals:

$$[P] \quad \begin{array}{ll} \text{maximiser:} & z = \mathbf{c} \mathbf{x} \\ \text{sous:} & \mathbf{A} \mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{array} \quad [D] \quad \begin{array}{ll} \text{minimiser:} & w = \mathbf{y} \mathbf{b} \\ \text{sous:} & \mathbf{y} \mathbf{A} \geq \mathbf{c} \\ & \mathbf{y} \geq \mathbf{0}. \end{array}$$

Pour tout $\underline{\mathbf{x}}$ solution réalisable de [P] et tout $\underline{\mathbf{y}}$ solution réalisable de [D], on a:

$$z(\underline{\mathbf{x}}) = \mathbf{c} \underline{\mathbf{x}} \leq \mathbf{y} \mathbf{b} = w(\underline{\mathbf{y}})$$

DÉMONSTRATION:

Puisque $\underline{\mathbf{x}}$ et $\underline{\mathbf{y}}$ sont réalisables respectivement pour [P] et [D], alors de $\mathbf{A} \underline{\mathbf{x}} \leq \mathbf{b}$ et $\mathbf{y} \geq \mathbf{0}$ on déduit: $\mathbf{y}(\mathbf{A} \underline{\mathbf{x}}) \leq \mathbf{y} \mathbf{b} = w(\underline{\mathbf{y}})$. De même, nous avons $\mathbf{y} \mathbf{A} \geq \mathbf{c}$ et $\underline{\mathbf{x}} \geq \mathbf{0}$, alors:

$$z(\underline{\mathbf{x}}) = \mathbf{c} \underline{\mathbf{x}} \leq (\mathbf{y} \mathbf{A}) \underline{\mathbf{x}} = \mathbf{y}(\mathbf{A} \underline{\mathbf{x}}) \leq \mathbf{y} \mathbf{b} = w(\underline{\mathbf{y}}).$$

COROLLAIRE 1:

Soit $\underline{\mathbf{x}}$ une solution réalisable de [P] et soit $\underline{\mathbf{y}}$ une solution réalisable de [D], telles que:

$$z(\underline{\mathbf{x}}) = \mathbf{c} \underline{\mathbf{x}} = \mathbf{y} \mathbf{b} = w(\underline{\mathbf{y}}).$$

Alors $\underline{\mathbf{x}}$ et $\underline{\mathbf{y}}$ sont des solutions optimales de [P] et [D] respectivement.

DEMONSTRATION: (évidente à partir du théorème 5). ■

COROLLAIRE 2:

Si [P] et [D] ont des solutions réalisables alors ils ont des solutions optimales \mathbf{x}_{opt} et \mathbf{y}_{opt} et

$$z(\mathbf{x}_{opt}) = \mathbf{c}^T \mathbf{x}_{opt} \leq \mathbf{y}_{opt}^T \mathbf{b} = w(\mathbf{y}_{opt}).$$

DEMONSTRATION:

Si [P] a une solution réalisable mais n'a pas de solution optimale finie alors d'après l'algorithme du simplexe, on peut trouver une solution réalisable de valeur aussi grande que l'on veut. Mais ceci est impossible quand les deux programmes ont des solutions réalisables d'après le corollaire 1. Les deux programmes ont donc des solutions optimales, et, puisque celles-ci sont évidemment réalisables, on a le résultat d'après le corollaire 1. ■

COROLLAIRE 3:

Si [P] (resp. [D]) admet une solution réalisable mais n'admet pas de solution optimale alors [D] (resp. [P]) n'admet pas de solution réalisable.

DEMONSTRATION: (reformulation du corollaire précédent). ■

2.3.3. Théorème fondamental de la dualité

Théorème 6: Soit un couple de programmes linéaires duals:

$$\begin{array}{llll} \text{[P]} & \begin{array}{l} \text{maximiser: } z = \mathbf{c}^T \mathbf{x} \\ \text{sous: } \mathbf{A} \mathbf{x} \leq \mathbf{b} \\ \mathbf{x} \geq \mathbf{0}. \end{array} & \text{[D]} & \begin{array}{l} \text{minimiser: } w = \mathbf{y}^T \mathbf{b} \\ \text{sous: } \mathbf{y}^T \mathbf{A} \geq \mathbf{c} \\ \mathbf{y} \geq \mathbf{0}. \end{array} \end{array}$$

Si [P] admet une solution optimale \mathbf{x}_{opt} alors [D] admet une solution optimale \mathbf{y}_{opt} et

$$z(\mathbf{x}_{opt}) = \mathbf{c}^T \mathbf{x}_{opt} = \mathbf{y}_{opt}^T \mathbf{b} = w(\mathbf{y}_{opt}).$$

DEMONSTRATION:

En ajoutant les variables d'écart on peut réécrire [P] sous la forme standard:

$$\begin{array}{ll} \text{[46]} & \begin{array}{l} \text{maximiser: } z = \mathbf{c}^T \mathbf{x} \\ \text{sous: } \mathbf{A} \mathbf{x} + \mathbf{e} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{0}, \mathbf{e} \geq \mathbf{0}. \end{array} \end{array}$$

Soit $\boldsymbol{\pi} \in \mathbb{R}^m$, le vecteur-multiplicateur relatif à la base optimale. Les formules du simplexe révisé (page 7) donnent:

$$\text{[47]} \quad \mathbf{c}' = (\mathbf{c} - \boldsymbol{\pi} \mathbf{A}; \mathbf{0} - \boldsymbol{\pi}) \leq \mathbf{0}$$

$$\text{[48]} \quad z_{\max} = \boldsymbol{\pi}^T \mathbf{b}$$

En conclusion, [47] signifie que le vecteur-multiplicateur $\boldsymbol{\pi}$ relatif à la base optimale est une solution réalisable du [P]: $\boldsymbol{\pi} \mathbf{A} \geq \mathbf{c}$ et $\boldsymbol{\pi} \geq \mathbf{0}$, et puisque $w(\boldsymbol{\pi}) = \boldsymbol{\pi}^T \mathbf{b} = z_{\max}$, alors la solution $\boldsymbol{\pi}$ est optimale d'après le corollaire 1. ■

Tous les principaux résultats de la dualité sont résumés dans le tableau suivant:

		[P] a une solution réalisable		[P] n'a pas de solution réalisable
		[P] a une solution optimale	[P] n'a pas de solution optimale	
		<i>Théorème de dualité</i> $z_{\max} = w_{\min}$	<i>Impossible !</i>	<i>Impossible !</i>
[D]	[D] a une solution réalisable	<i>Impossible !</i>	<i>Impossible !</i>	$w \rightarrow -\infty$
[D]	[D] n'a pas de solution optimale	<i>Impossible !</i>	<i>possible</i>	

2.3.4. Relation entre les solutions des programmes linéaires duals: - théorème des écarts complémentaires

Théorème 7:

Soit un couple de programmes linéaires duals:

$$\begin{array}{llll} \text{maximiser: } & z = \mathbf{c} \mathbf{x} & \text{minimiser: } & w = \mathbf{y} \mathbf{b} \\ [\text{P}] & \text{sous: } & [\text{D}] & \text{sous: } \\ & \mathbf{A} \mathbf{x} \leq \mathbf{b} & & \mathbf{y} \mathbf{A} \geq \mathbf{c} \\ & \mathbf{x} \geq 0. & & \mathbf{y} \geq 0. \end{array}$$

Soit $\underline{\mathbf{x}}$ une solution réalisable de [P] et soit $\underline{\mathbf{y}}$ une solution réalisable de [D].

Alors $\underline{\mathbf{x}}$ et $\underline{\mathbf{y}}$ sont des solutions optimales de [P] et [D] si et seulement si:

- (I) $a_{i1}\underline{x}_1 + a_{i2}\underline{x}_2 + \dots + a_{in}\underline{x}_n < b_i \Rightarrow \underline{y}_i = 0 ;$ pour tout $i = 1, 2, \dots, m$
- (II) $\underline{x}_j > 0 \Rightarrow a_{1j}\underline{y}_1 + a_{2j}\underline{y}_2 + \dots + a_{nj}\underline{y}_n = c_j ;$ pour tout $j = 1, 2, \dots, n$

DÉMONSTRATION:

On sait déjà que $\underline{\mathbf{x}}$ et $\underline{\mathbf{y}}$, des solutions réalisables de [P] et [D], sont des solutions optimales si et seulement si : $\mathbf{c} \underline{\mathbf{x}} = \mathbf{y} \mathbf{A} \underline{\mathbf{x}} = \mathbf{y} \mathbf{b}$, c'est-à-dire si et seulement si

$$\begin{aligned} \underline{\mathbf{y}}(\mathbf{b} - \mathbf{A} \underline{\mathbf{x}}) &= 0 \\ (\underline{\mathbf{y}} \mathbf{A} - \mathbf{c}) \underline{\mathbf{x}} &= 0. \end{aligned}$$

Soit $\mathbf{p} \cdot \mathbf{q} = p_1 q_1 + p_2 q_2 + \dots + p_k q_k$ le produit scalaire de deux vecteurs $\mathbf{p} \geq 0$ et $\mathbf{q} \geq 0$. Il est évident que $\mathbf{p} \cdot \mathbf{q} = 0$ si et seulement si pour tout $i = 1, 2, \dots, k$ on a $q_i > 0 \Rightarrow p_i = 0$, d'où le théorème. ■

2.3.5. Interprétation géométrique de la dualité

Revenons à l'Exemple 1 page 3 et son interprétation géométrique (Figure 2 page 8):

$$\begin{aligned}
 & \text{maximiser:} \quad z = 7x_1 + 9x_2 \\
 & \text{sous:} \quad x_1 + x_2 \leq 8 \\
 [49] \quad & \quad x_2 \leq 4 \\
 & \quad 2x_1 + 3x_2 \leq 19 \\
 & \quad x_1 \geq 0, x_2 \geq 0.
 \end{aligned}$$

Ce problème a la solution optimale $x_1=5, x_2=3$ de valeur $z_{\max} = 62$ déterminée par l'intersection des deux droites:

$$\left\{
 \begin{array}{l}
 x_1 + x_2 = 8 \\
 2x_1 + 3x_2 = 19
 \end{array}
 \right.$$

Pour confirmer algébriquement l'optimalité, il suffit d'étudier la combinaison linéaire de la première et la troisième ligne de contraintes de [49] avec les coefficients 3 et 2 :

$$7x_1 + 9x_2 = 3(x_1 + x_2) + 2(2x_1 + 3x_2) \leq 3 \cdot 8 + 2 \cdot 19 = 62$$

Ceci signifie que l'ensemble:

$$\left\{
 \begin{array}{l}
 x_1 + x_2 \leq 8 \\
 2x_1 + 3x_2 \leq 19
 \end{array}
 \right.$$

est inclus dans le demi-plan $7x_1 + 9x_2 \leq 62$ (voir la Figure 3):

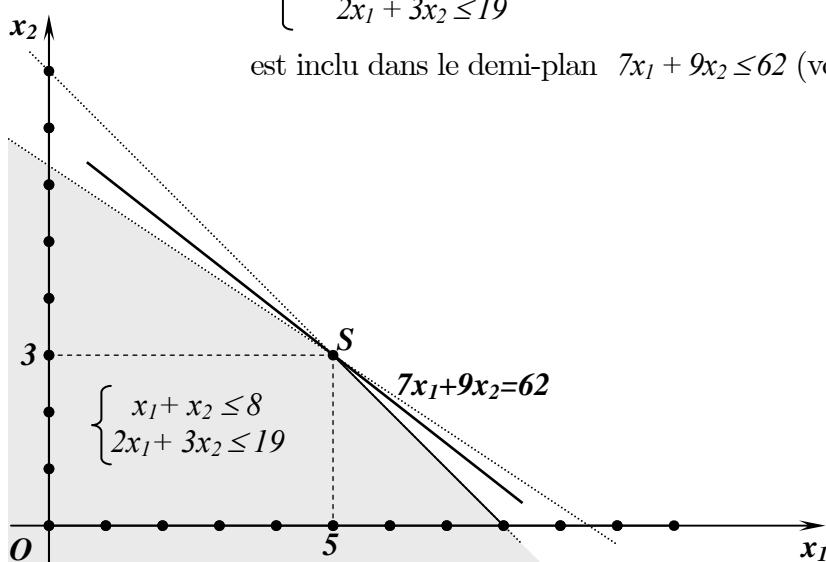


Figure 3.

Etudions maintenant la combinaison linéaire de la première et la troisième ligne de contraintes de [49] avec les coefficients non-négatifs quelconques y_1 et y_3 :

$$y_1(x_1 + x_2) + y_3(2x_1 + 3x_2) \leq 8y_1 + 19y_3.$$

Les droites de la famille

$$(y_1 + 2y_3)x_1 + (y_1 + 3y_3)x_2 = 8y_1 + 19y_3$$

pivotent autour du point $C(5; 3)$ entre deux droites extrêmes : $x_1 + x_2 = 8$ et $2x_1 + 3x_2 = 19$ (qui correspondent respectivement aux valeurs $y_1=1, y_3=0$ et $y_1=0, y_3=1$).

Cette interprétation montre clairement que nous pouvons choisir des valeurs y_1 et y_3 , pour que la droite de faisceau soit identique à $7x_1 + 9x_2 = 62$, il faut que $y_1 + 2y_3 = 8$ et $y_1 + 3y_3 = 19$. L'existence de ces deux valeurs est une conséquence du théorème fondamental de la dualité.

2.3.6. Interprétations économiques de la dualité

En général on peut distinguer facilement deux voies différentes pour donner des interprétations de la dualité. La première consiste à affecter une signification particulière aux variables primales et duales liée à la modélisation d'un problème particulier. Cette voie a été largement explorée au paragraphe 1.1, traitant des nombreux exemples de la modélisation simultanée d'une situation par deux modèles duaux. Une deuxième consiste à utiliser les résultats (des théorèmes) de la dualité, pour interpréter les valeurs numériques des variables duales à l'optimum des deux programmes duaux. Revenons à l'Exemple 1 page 3 et supposons que le second membre du programme [49] a été légèrement perturbé :

$$\begin{aligned}
 & \text{maximiser: } z_\varepsilon = 7x_1 + 9x_2 \\
 & \text{sous: } \begin{aligned} x_1 + x_2 &\leq 8 + \varepsilon_1 \\ x_2 &\leq 4 + \varepsilon_2 \\ 2x_1 + 3x_2 &\leq 19 + \varepsilon_3 \\ x_1, x_2 &\geq 0. \end{aligned} \\
 [50]
 \end{aligned}$$

La solution optimale a changé (voir l'interprétation géométrique – les points extrêmes sont en général légèrement déplacés), mais on peut donner une formule simple pour la perturbation de la valeur optimale, basée sur le théorème fondamental de la dualité. En effet, dans une situation non dégénérée la solution optimale du programme dual perturbé:

$$\begin{aligned}
 & \text{minimiser: } w_\varepsilon = (8 + \varepsilon_1)y_1 + (4 + \varepsilon_2)y_2 + (19 + \varepsilon_3)y_3 \\
 & \text{sous: } \begin{aligned} y_1 + 2y_3 &\geq 7 \\ y_1 + y_2 + 2y_3 &\geq 9 \\ y_1, y_2, y_3 &\geq 0. \end{aligned}
 \end{aligned}$$

reste inchangée pour une perturbation marginale. La nouvelle valeur perturbée s'exprime par :

$$z_{\max} = w_{\min} = (8 + \varepsilon_1)\underline{y}_1 + (4 + \varepsilon_2)\underline{y}_2 + (19 + \varepsilon_3)\underline{y}_3 = z_{\max} + \varepsilon_1\underline{y}_1 + \varepsilon_2\underline{y}_2 + \varepsilon_3\underline{y}_3$$

où $\underline{\mathbf{y}} = (\underline{y}_1, \underline{y}_2, \underline{y}_3)$ est une solution optimale du programme dual non perturbé [4].

EXEMPLE 5: A l'usine de l'Exemple 1 constate que la solution optimale ($x_1=5$ et $x_2=3$) prévoit l'utilisation totale des matières premières I et III ($x_1+x_2=8$ et $2x_1+3x_2=19$) tandis que la matière II est surabondante ($x_2<4$). Le problème est de savoir à quel prix unitaire a-t-on intérêt à acheter la matière première I. Il est évident qu'un prix trop élevé risque d'anéantir le profit supplémentaire.

On a donc le programme linéaire:

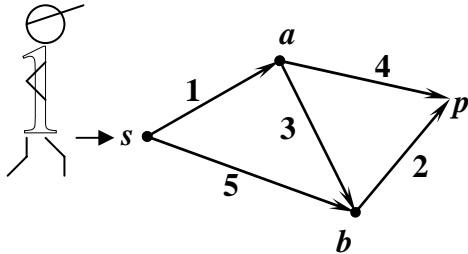
$$\begin{aligned}
 & \text{maximiser: } z_\varepsilon = 7x_1 + 9x_2 \\
 & \text{sous: } \begin{aligned} x_1 + x_2 &\leq 8 + \varepsilon \\ x_2 &\leq 4 \\ 2x_1 + 3x_2 &\leq 19 \\ x_1, x_2 &\geq 0. \end{aligned} \\
 [51]
 \end{aligned}$$

dont la solution primaire a changé mais les deux solutions duales (du programme dual initial et du programme perturbé) restent inchangées: $\underline{y}_1=3$, $\underline{y}_2=0$, $\underline{y}_3=2$, ce qui permet de calculer l'augmentation du profit supplémentaire: $z_{\max} - z_{\max} = \varepsilon\underline{y}_1$.

On conclut que l'on a intérêt à acheter une quantité supplémentaire de matière première I si son prix unitaire d'achat y_1 est inférieur ou égal à la valeur numérique de la première variable duale: $y_1 \leq \underline{y}_1 = 3$. En langage économique on dit que la valeur de la variable duale est le *coût marginal* de la matière première correspondante. La solution $\underline{y}_2=0$ signifie que l'achat de matière première II n'est pas une source du profit supplémentaire.

2.3.6. Programmation linéaire et graphes

Nous allons d'abord montrer que le problème de recherche d'un plus court chemin se modélise comme un simple programme linéaire. Ensuite, grâce à une propriété remarquable de la matrice du système de contraintes et avec les données bien structurées on peut le résoudre sans faire directement appel à la méthode du simplexe.



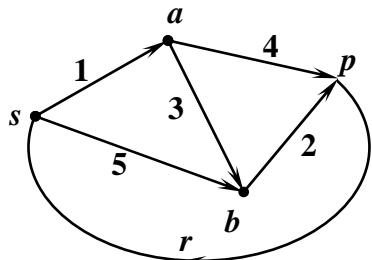
$$B(G) = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} s \\ a \\ b \\ p \end{matrix} & \left[\begin{array}{ccccc} 1 & 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 & -1 \\ 0 & -1 & 0 & -1 & 0 \end{array} \right] \end{matrix}$$

Fig. 50.

Imaginons un touriste qui, en partant de l'endroit s , veut emprunter le plus court chemin pour arriver au point p et qu'il dispose d'une carte détaillée avec les distances schématisées sur le graphe de la Fig.50. Il est clair qu'au départ il doit choisir l'arc 1 ou 5 (ici on identifie l'indice et la distance). De même il arrivera en p par 2 ou par 4. S'il décide de partir par 1, il sera bientôt obligé (au point a) de faire le choix entre 3 ou 4 pour continuer. En utilisant cinq variables booléennes $x_i \in \{0;1\}$ on peut exprimer ces choix par le système linéaire:

$$\left\{ \begin{array}{l} I = x_1 + x_5 \\ x_1 = x_3 + x_4 \\ x_3 + x_5 = x_2 \\ x_2 + x_4 = I \end{array} \right. \quad \left\{ \begin{array}{l} -x_1 \quad -x_5 = -1 \\ x_1 \quad -x_3 - x_4 = 0 \\ -x_2 + x_3 + x_5 = 0 \\ x_2 \quad + x_4 = 1 \end{array} \right.$$

L'objectif est de minimiser la valeur $x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5$. Après avoir mis en forme standard $-Ax = b$, on remarque que A est la matrice d'incidence aux arcs du graphe de la Fig.50. et $[-A, b]$ est la matrice d'incidence aux arcs du même graphe avec un arc, dit "de retour", supplémentaire entre p et s .



$$(-A, b) = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & r \end{matrix} \\ \begin{matrix} s \\ a \\ b \\ p \end{matrix} & \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \end{array} \right] \end{matrix}$$

Fig. 51.

On démontre que toute matrice d'incidence est *totalement unimodulaire*, ce qui veut dire que non seulement ses coefficients sont ± 1 et 0, mais aussi que le déterminant de toute matrice carrée extraite prend une de ces trois valeurs. Ceci implique que pour toute solution de base réalisable du système $-Ax = b$ avec $x \geq 0$, on a $x_i \in \{0;1\}$. Puisque les contraintes traduisent la continuité de trajet pour notre touriste (s'il arrive à un sommet par un arc il doit repartir par un autre arc) chaque solution de base réalisable (optimale ou pas) s'interprète comme un chemin de s à p .

En conclusion, le problème (I) se traduit par la recherche d'une solution de base optimale du programme linéaire suivant:

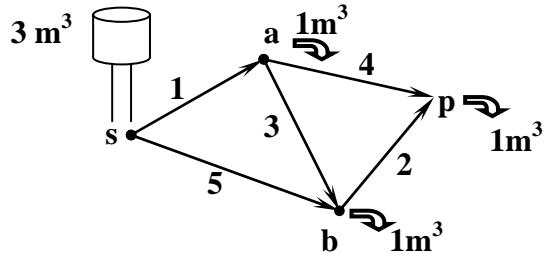
$$\text{minimiser : } z = \mathbf{d}\mathbf{x}$$

$$\text{sous : } -\mathbf{A}\mathbf{x} = \mathbf{b}$$

$$\mathbf{x} \geq 0,$$

où \mathbf{A} est la matrice d'incidence aux arcs du graphe, \mathbf{d} est le vecteur-ligne composé des distances associées aux arcs et \mathbf{b} est le vecteur-colonne représentant l'arc de retour entre le début et la fin du chemin que l'on recherche.

Pour modéliser le problème (II) étudions d'abord l'exemple suivant. Imaginons le château d'eau situé dans le sommet s qui peut fournir 1m^3 d'eau à chaque consommateur situé dans un autre sommet du graphe (figure ci-contre)



Dans cet exemple les nombres associés aux arcs représentent le coût de transport d'un mètre cube d'eau par ces arcs. Comment distribuer à chaque client un hectolitre d'eau au coût minimum? Il est clair qu'il faut approvisionner chaque client par le chemin le moins coûteux (le plus court au sens du coût). Il s'agit donc de "superposer" plusieurs problèmes du plus court chemin. Dans le cas de la Fig. ce problème, modélisé par le programme linéaire, se présente ainsi :

$$\text{minimiser: } z = x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5$$

$$\text{sous } \left\{ \begin{array}{l} 3 = x_1 + x_5 \\ x_1 = x_3 + x_4 + 1 \\ x_3 + x_5 = x_2 + 1 \\ x_2 + x_4 = 1 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{l} -x_1 & & -x_5 = -3 \\ x_1 & -x_3 - x_4 & = 1 \\ -x_2 + x_3 & + x_5 = 1 \\ x_2 & + x_4 & = 1 \end{array} \right.$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0.$$

Maintenant on cherche à déterminer le tarif pour chaque consommateur. Considérons le vecteur $\boldsymbol{\pi} = (\pi_s; \pi_a; \pi_b; \pi_p)$ des prix d'un m^3 d'eau pour chaque client. Evidemment $\pi_s=0$ et aucun consommateur n'acceptera de payer plus que son voisin, plus le coût d'approvisionnement depuis un de ses voisins. Par exemple les exigences légitimes de p sont: $\pi_p \leq \pi_a + 4$ et $\pi_p \leq \pi_b + 2$. En respectant toutes ces contraintes la Compagnie des Eaux veut maximiser son profit $\pi_a + \pi_b + \pi_p$.

Si l'on considère le programme linéaire précédent sous la forme:

$$\text{minimiser : } z = \mathbf{d}\mathbf{x}$$

$$\text{sous : } -\mathbf{A}\mathbf{x} = \mathbf{b}$$

$$\mathbf{x} \geq 0,$$

le raisonnement ci-dessus est directement exprimé par son dual:

$$\text{maximiser : } w = \boldsymbol{\pi}\mathbf{b}$$

$$\text{sous : } -\boldsymbol{\pi}'\mathbf{A} \leq \mathbf{d}$$

car on peut, sans perdre en généralité, poser $\pi_s = 0$ (ceci est dû au fait que les lignes de la matrice \mathbf{A} sont linéairement dépendantes – leur somme est nulle – et les valeurs $\boldsymbol{\pi}$ sont déterminées à une constante additive près).

En conclusion, pour un graphe quelconque d'ordre n , le problème (II) se traduit par la recherche d'une solution de base optimale du programme linéaire suivant:

$$\begin{aligned} & \text{minimiser : } z = \mathbf{d}\mathbf{x} \\ & \text{sous : } -\mathbf{A}\mathbf{x} = \mathbf{b} \\ & \quad \mathbf{x} \geq 0, \end{aligned}$$

où \mathbf{A} est la matrice d'incidence aux arcs du graphe, \mathbf{d} est le vecteur-ligne composé des distances associées aux arcs et \mathbf{b} est le vecteur-colonne avec la valeur $1-n$ à la position correspondante au sommet s et le 1 partout ailleurs.

Son dual:

$$\begin{aligned} & \text{maximiser : } w = \boldsymbol{\pi}\mathbf{b} \\ & \text{sous : } -\boldsymbol{\pi}^t \mathbf{A} \leq \mathbf{d} \\ & \quad \boldsymbol{\pi}_s = 0 \end{aligned}$$

est connu sous le nom du *problème de la tension maximum* par rapport au sommet s (les composantes du vecteur $\boldsymbol{\pi}$ s'interprètent comme potentiel, la tension d'un arc est définie comme la différence de potentiels affectés à ses sommets: terminal et initial). Le théorème suivant formalise l'interprétation des variables duales.

Théorème: Soit $G=(X;E)$ un graphe orienté et $d: E \rightarrow \mathbb{R}$ une fonction qui associe à toute arête de G un poids réel. Les nombres π_x associés aux sommets sont les plus courtes distances entre un sommet s et les sommets $x \in X$ si et seulement si

- (i) $\pi_s = 0$;
- (ii) $-\boldsymbol{\pi}^t \mathbf{A} \leq \mathbf{d}$
- (iii) le graphe partiel $G'=(X;E')$ où $E'=\{e=(x,y) \in E \text{ telles que } \pi_y - \pi_x = d(e)\}$ admet s comme racine.

DEMONSTRATION:

\Rightarrow Si $\pi_s < 0$ il existerait un circuit absorbant (un chemin de s à s), alors $\pi_s = 0$ (la longueur du chemin qui n'emprunte aucun arc).

Soit $e=(x,y)$ un arc quelconque, Γ le plus court chemin de s à x et Γ' le chemin de s à y obtenu en adjoignant l'arc e à Γ . La longueur du plus court chemin de s à y ne peut pas dépasser la longueur de Γ' alors $\pi_y \leq \pi_x + d(e)$. Ceci implique (ii).

Si $e=(x,y)$ appartient à un plus court chemin Γ_{sy} de s à y alors Γ_{sx} et Γ_{sy} sont des plus courts chemins des à x et de s à y respectivement. On a:

$$l(\Gamma_{sy}) = \pi_y = l(\Gamma_{sx}) + d(e) = \pi_x + d(e)$$

ce qui prouve (iii).

\Leftarrow Si les conditions (i) et (ii) sont vérifiées, pour tous les chemins Γ on a:

$$l(\Gamma) = \sum_{e \in \Gamma} d(e) = \sum_{e \in \Gamma} d(x,y) \geq \sum_{e \in \Gamma} d(x';x'') = \sum_{e \in \Gamma} (\pi_{x''} - \pi_x) = \pi_x - \pi_s = \pi_x$$

alors π_x est une borne inférieure de la plus courte distance de s à x . Cette borne est atteinte pour un chemin de s à x dans G' . ■

Si le programme linéaire

$$\begin{aligned} & \text{maximiser : } w = \mathbf{b}\boldsymbol{\pi} \\ & \text{sous : } -\boldsymbol{\pi}^t \mathbf{A} \leq \mathbf{d} \\ & \quad \boldsymbol{\pi}_s = 0 \end{aligned}$$

admet une solution optimale, son dual:

$$\begin{aligned} & \text{minimiser : } z = \mathbf{d}\mathbf{x} \\ & \text{sous : } -\mathbf{A}\mathbf{x} = \mathbf{b} \\ & \quad \mathbf{x} \geq 0, \end{aligned}$$

aussi. Le théorème précédent permet de retrouver l'interprétation de la base optimale de ce système en termes de graphes. \mathbf{A} – la matrice d'incidence aux arcs du graphe d'ordre n , n'est pas de plein rang car la somme de ses lignes donne le vecteur nul, alors $\text{rg}(\mathbf{A}) \leq n-1$. Puisque le graphe admet s comme racine il est donc connexe (au sens non orienté), alors il admet un arbre comme graphe partiel. Le nombre d'arêtes de cet arbre est $n-1$ et on démontre facilement que les colonnes de \mathbf{A} qui correspondent à ces arêtes sont linéairement indépendantes, alors $\text{rg}(\mathbf{A}) = n-1$. La base optimale du programme linéaire

$$\begin{aligned} & \text{minimiser : } z = \mathbf{d}\mathbf{x} \\ & \text{sous : } -\mathbf{A}\mathbf{x} = \mathbf{b} \\ & \quad \mathbf{x} \geq 0, \end{aligned}$$

s'interprète comme une arborescence du graphe, enracinée en s , appelée *l'arborescence des plus courts chemins issus de s* .

Notons que ce n'est pas l'arborescence enracinée en s de poids minimum, ce que l'on peut constater sur la Fig.4: l'arborescence des plus courts chemins issus de s est obtenue en supprimant les arcs 2 et 5; par contre l'arborescence enracinée en s de poids minimum est obtenue en supprimant les arcs 4 et 5.

Avant de passer à la présentation des méthodes numériques de recherche du plus court chemin nous allons présenter une méthode analogique, dans le cas du graphe symétrique (si $(x,y) \in E$ alors $(y,x) \in E$). Ceci peut être considéré aussi comme une interprétation physique de la dualité.

Imaginons un modèle du graphe réalisé avec des ficelles: les sommets sont des noeuds et longueurs des tronçons de ficelles sont proportionnelles aux distances associées aux arcs correspondants. On tient le noeud s dans la main et on accroche des poids unitaires aux autres noeuds. Si on laisse pendre librement ce réseau, on résout simultanément le problème des plus courts chemins issus de s et celui de la tension maximum. Les ficelles tendues correspondent aux arcs des plus courts chemins.

4. Programmation linéaire paramétrique.

Des exemples de l'interprétation économique au chapitre précédent mettent en évidence le problème de calcul de bornes pour lesquelles le raisonnement suivi reste valable. En général l'étude des variations de la solution optimale d'un programme linéaire quand certaines données sont modifiées porte le nom de programmation linéaire paramétrique. La dualité met en évidence deux cas où cette étude est facile à réaliser : le cas de la paramétrisation linéaire de la fonction objectif:

$$\begin{aligned} & \text{maximiser: } z = (c + \varepsilon d)\mathbf{x} \\ [42] \quad & \text{sous: } \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \quad \mathbf{x} \geq 0. \end{aligned}$$

et le cas de la paramétrisation linéaire du second membre:

$$\begin{aligned} & \text{maximiser: } z = cx \\ [43] \quad & \text{sous: } Ax \leq b + \mu f \\ & x \geq 0. \end{aligned}$$

où ε et μ sont des scalaires et d et f des vecteurs fixés.

Nous présentons ici les démarches nécessaires sur un exemple, et nous donnons aussi quelques cas en exercices. Nous signalons que la plupart des logiciels existant permettent de réaliser facilement cette analyse. Comme le met en évidence l'exemple 2, dans la plupart des cas il s'agit de la lecture intelligente de "la sortie de l'ordinateur". Pour une étude plus approfondie de ce sujet nous renvoyons les lecteurs intéressés aux chapitres d'analyse de "sensibilité", ou d'analyse "post-optimale" ou bien de la programmation linéaire paramétrique dans la littérature spécialisée (p.ex. V. Chvatal, M. Sakarovitch...)

Un artisan confiturier doit prendre une décisions importante, lui permettant de maximiser son futur profit net dans la situation suivante: il dispose d'une réserve de 8 tonnes de sucre et d'un capital de 40 000 €. Il peut acheter des fraises, cela lui coûtera immédiatement 2 € par kg de fraises traité et lui rapportera plus tard 4€50 par kg de confiture vendu. Une autre solution consiste à acheter des roses. Il devra alors débourser 15 € par kg de roses traité, mais aura un revenu ultérieur de 12€60 par kg de confiture vendu. Ici on admet que la production de confitures de fraises consiste à mélanger 50% de fruits et de 50 % de sucre et pour obtenir la confiture de roses on mélange 40% de fruits et de 60% de sucre.

Comment doit-il faire pour maximiser son bénéfice net?

Soient x_1 et x_2 les quantités (en kg) de sucre destinées respectivement aux confitures de fraises et de roses. La proportion des ingrédients dans la confiture de fraises est 1:1 (50%:50%). En mélangeant 1kg de sucre avec 1kg de fraises on investit 2€ mais on obtient 2kg de confiture vendue à $2 \times 4\text{€}50 = 9\text{€}$; le profit net escompté est donc de 7€. La proportion des ingrédients dans la confiture de roses est 3:2 (60%:40%). Pour 1kg de sucre il faut ajouter $\frac{2}{3}$ kg de roses. On investit $\frac{2}{3} \times 15\text{€} = 10\text{€}$ mais on obtient $(1 + \frac{2}{3})\text{kg} = \frac{5}{3}\text{kg}$ de confiture vendue à $\frac{5}{3} \times 12\text{€}60 = 21\text{€}$ le profit net escompté s'élève à 11€.

Modélisé sous la forme d'un programme linéaire, ce problème devient:

$$\begin{aligned} & \text{maximiser: } z = 7x_1 + 11x_2 \\ [44] \quad & \text{sous: } \begin{array}{l} x_1 + x_2 \leq 8000 \\ 2x_1 + 10x_2 \leq 40000 \\ x_1 \geq 0, x_2 \geq 0. \end{array} \\ & \boxed{1} \quad \downarrow \\ & \left(\begin{array}{c|ccccc|c} & z & x_1 & x_2 & x_3 & x_4 & \\ \hline x_3 & 0 & 1 & 1 & 1 & 0 & 8000 \\ x_4 & 0 & 2 & 10 & 0 & 1 & 40000 \\ \hline z & 1 & -7 & -11 & 0 & 0 & 0 \end{array} \right) \\ & \leftarrow \quad \boxed{2} \quad \downarrow \\ & \left(\begin{array}{c|ccccc|c} & z & x_1 & x_2 & x_3 & x_4 & \\ \hline x_3 & 0 & 0.8 & 0 & 1 & -0.1 & 4000 \\ x_2 & 0 & 0.2 & 1 & 0 & 0.1 & 4000 \\ \hline z & 1 & -4.8 & 0 & 0 & 1.1 & 44000 \end{array} \right) \\ & \boxed{3} \\ & \left(\begin{array}{c|ccccc|c} & z & x_1 & x_2 & x_3 & x_4 & \\ \hline x_1 & 0 & 1 & 0 & 1.25 & -0.125 & 5000 \\ x_2 & 0 & 0 & 1 & -0.25 & 0.125 & 3000 \\ \hline z & 1 & 0 & 0 & 6 & 0.5 & 68000 \end{array} \right) \end{aligned}$$

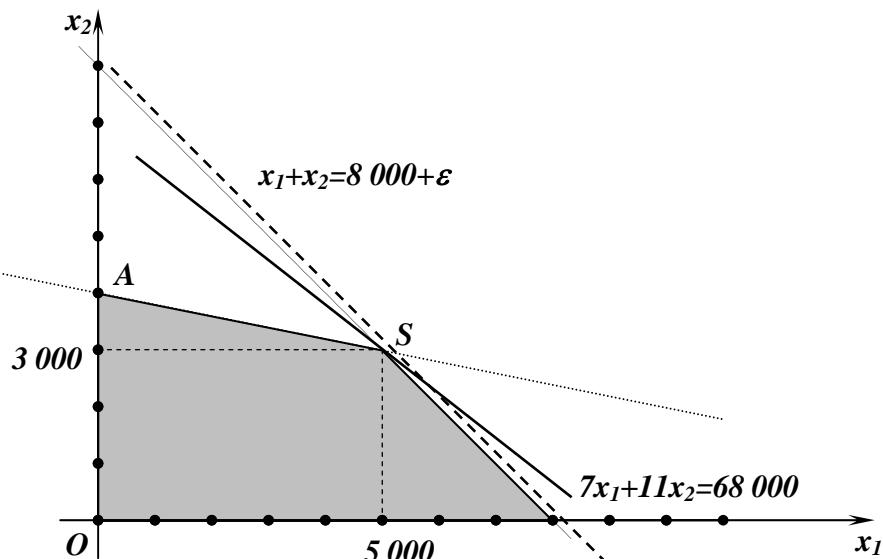
Le résultat du premier pivotage met en évidence le fait que le propriétaire ne pourra pas mettre tout son sucre dans la confiture de roses. La solution optimale est d'utiliser $x_1=5\ 000$ kg de sucre pour faire des confitures de fraises et $x_2=3\ 000$ kg de sucre pour faire la confiture de roses. Il faudra acheter 5 000 kg de fraises et 2 000 kg de roses. Le profit net s'élèvera dans ce cas à 68 000 €.

$$\begin{aligned} \text{minimiser: } & z = y_1 + 5y_2 \\ \text{sous: } & y_1 + 2y_2 \geq 7 \\ [45] & y_1 + 10y_2 \geq 11 \\ & y_1 \geq 0, y_2 \geq 0. \end{aligned}$$

La valeur de sa solution optimale $y_1=6$, $y_2=0,5$ figure déjà dans le dernier tableau du simplexe primal. Pour augmenter encore son bénéfice, le propriétaire peut soit acheter du sucre sur le marché soit emprunter de l'argent à la banque.

Etudions d'abord la première possibilité. L'examen du programme :

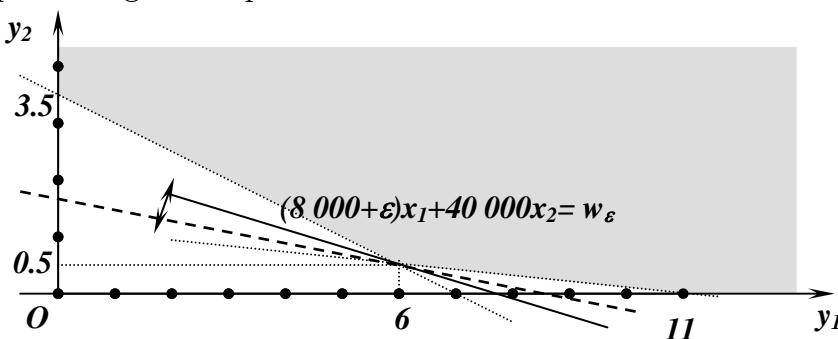
$$\begin{aligned} \text{maximiser: } & z = 7x_1 + 11x_2 \\ \text{sous: } & x_1 + x_2 \leq 8\ 000 + \varepsilon \\ [46] & 2x_1 + 10x_2 \leq 40\ 000 \\ & x_1 \geq 0, x_2 \geq 0. \end{aligned}$$



pour différentes valeurs de ε peut donner une réponse détaillée. La solution optimale étant perturbée, il est plus commode d'étudier son dual:

$$\begin{aligned} \text{minimiser: } & w_\varepsilon = (8\ 000 + \varepsilon)y_1 + 40\ 000y_2 \\ \text{sous: } & y_1 + 2y_2 \geq 7 \\ [47] & y_1 + 10y_2 \geq 11 \\ & y_1 \geq 0, y_2 \geq 0. \end{aligned}$$

avec son interprétation géométrique:



Sa solution optimale $y_1=6$, $y_2=0,5$ indique qu'il est intéressant d'acheter le sucre si son prix ne dépasse pas 6 € le kg. Pour trouver les limites de la quantité marginale ainsi achetée on remarque que la solution duale reste optimale quand la droite w_ε pivote entre la droite $y_1+2y_2=7$ et la droite $y_1+10y_2=11$. En présentant toutes ces trois droites sous les formes :

$$\begin{aligned} 20\,000y_1 + 40\,000y_2 &= 140\,000 \\ (8\,000+\varepsilon)y_1 + 40\,000y_2 &= w_\varepsilon \\ 4\,000y_1 + 40\,000y_2 &= 44\,000 \end{aligned}$$

on conclut : $4\,000 \leq 8\,000 + \varepsilon \leq 20\,000$ c.-à-d. $-4\,000 \leq \varepsilon \leq 12\,000$. On peut arriver à la même conclusion sans faire appel à l'interprétation géométrique, mais en se servant du dernier tableau du simplexe. En effet, avec la notation du [43] on peut considérer que le second membre a été perturbé par εf avec $f=(1;0)$. Il suffit donc d'utiliser la transformation de la colonne x_3 afin de découvrir la nouvelle solution perturbée : $x_1=5\,000+1,25\varepsilon$ et $x_2=3\,000-0,25\varepsilon$. La réalisabilité exige: $x_1 \geq 0$ et $x_2 \geq 0$, d'où les limites du ε .

Avec l'argent disponible on peut traiter au maximum 20 000 kg (= 40 000:2) de fraises. La borne 12 000 kg exprime le fait qu'on peut mettre tout le sucre ($8\,000+12\,000=20\,000$ kg) dans la confiture de fraises. Au-delà de cette quantité la solution optimale duale passe à $y_1=0$, ce qui signifie que le sucre supplémentaire, ne pouvant pas être mélangé avec la fraise, n'apportera plus de profit.

La borne négative signifie qu'il est parfois plus intéressant de vendre le sucre directement sur le marché et de ne pas le mettre dans la confiture. Ceci est vrai p.ex., si le prix de sucre est légèrement supérieur à 6€ le kg. Il ne faut tout de même pas vendre plus de 4 000 kg, car l'autre moitié qui reste peut être vendus dans la confiture de roses qui est un investissement plus rentable, que la vente directe, si le prix sur le marché extérieur ne dépasse pas 11 €. Au-delà (la nouvelle solution optimale du dual passe à $y_1=11$) on peut vendre la totalité du sucre. Une autre solution pour augmenter le bénéfice consiste à emprunter de l'argent à la banque. La valeur de la deuxième variable duale s'interprète cette fois-ci en termes financiers comme le taux d'intérêt. (Ici la matière première est un euro et la valeur de y_2 représente son "prix"). Supposons que le propriétaire souhaite emprunter à la banque une somme μ .

Ceci peut être formulé par le programme linéaire suivant:

$$\begin{array}{ll} \text{maximiser: } & z = 7x_1 + 11x_2 \\ \text{sous: } & \begin{aligned} x_1 + x_2 &\leq 8\,000 \\ 2x_1 + 10x_2 &\leq 40\,000 + \mu \\ x_1 \geq 0, x_2 \geq 0. \end{aligned} \end{array} \quad [48]$$

Le taux de référence est donné par la valeur de la solution optimale $y_2=0,5$. Ceci signifie qu'il est intéressant d'emprunter si le taux d'intérêt ne dépasse pas 50%.

Pour préciser quelles sont les limites il suffit d'utiliser la transformation de x_4 dans le dernier tableau du simplexe afin de découvrir la nouvelle solution perturbée:

$$\begin{aligned} x_1 &= 5\,000 - 0,125\mu \\ x_2 &= 3\,000 + 0,125\mu \end{aligned}$$

La réalisabilité nécessite que $x_1 \geq 0$ et $x_2 \geq 0$, d'où les limites du μ : $-24\,000 \leq \mu \leq 40\,000$.

S'il emprunte, il doit se limiter à 40 000 €. Avec l'argent disponible on peut acheter la quantité des roses nécessaire pour faire uniquement la confiture de roses avec le sucre disponible. Au-delà la solution optimale du dual du [48]:

$$\begin{aligned}
 & \text{minimiser:} && w_\mu = 8000y_1 + (40000+\mu)y_2 \\
 & \text{sous:} && y_1 + 2y_2 \geq 7 \\
 [49] & && y_1 + 10y_2 \geq 11 \\
 & && y_1 \geq 0, y_2 \geq 0.
 \end{aligned}$$

bascule vers le point $y_2 = 0$, ce qui signifie que l'argent supplémentaire ne pourrait pas être investi avec un profit (on peut emprunter à taux zéro !).

Si le taux d'intérêt est très élevé et dépasse 50%, le propriétaire a plutôt intérêt à se lancer lui-même dans une entreprise lucrative en investissant une partie de son capital à la bourse. S'il investit sur les marchés financiers, il doit se limiter à 24 000 €. Avec les 16 000 € restant on peut acheter des fraises pour faire la confiture avec le sucre disponible. Ceci est sûrement moins intéressant que le traitement des roses dans l'absolu, mais pas en termes de taux d'intérêt: 2 € investis dans 1kg de fraises apportent un profit net de 7 € c.-à-d. 350%, par contre 10 € investis dans 1kg de roses apporte un profit net de 11 € c.-à-d. 110% seulement! Evidemment, lors d'une grande instabilité des marchés financiers, quand le taux d'intérêt atteint 350%, il peut tout investir (la solution duale passe à $y_2=3.5$).

5. Programmation linéaire et ANALYSE des DONNEES.

Le théorème 1 donne une condition nécessaire et suffisante pour qu'un système linéaire $\mathbf{Ax}=\mathbf{b}$ admette une solution. Mais dans la pratique, assez souvent nous devons chercher une solution d'un système qui n'a pas de solution ! Ce problème, mathématiquement mal posé pour l'instant, doit parfois trouver une réponse opérationnelle. On peut citer deux situations dans lesquelles nous sommes amenés à calculer la meilleure solution approximative d'un système qui n'a pas de solution. La première, c'est quand le système original admet une solution (p.ex. il modélise un phénomène qui naturellement possède une solution) mais les coefficients de ce système sont perturbés par des incertitudes des instruments de mesure (exemple classique de ce cas : mesures effectuées par des géomètres). Une deuxième, c'est quand le système est seulement une "approche linéaire" d'un phénomène qui, en réalité, n'est pas tout à fait "linéaire" (exemple classique de ce deuxième cas: approximation linéaire des mouvements des étoiles sur la sphère céleste). Très souvent nous sommes confrontés à ces deux situations simultanément (p.ex. les mesures des concentrations en spectrophotométrie quantitative).

Il faut aussi donner un sens précis à la notion de la meilleure solution approchée. Les variables auxiliaires v_i introduites dans la phase I, sont des indicateurs d'éloignement entre le premier et le second membre du système. On va utiliser les notations suivantes: $\mathbf{Ax}=\mathbf{b}$ le système qui n'a pas de solution, \mathbf{x}^* sa solution approximative, et $\mathbf{Ax}+\mathbf{v}=\mathbf{b}$ le système auxiliaire. L'exemple suivant met en évidence une grande difficulté dans la conception de cette notion.

EXEMPLE 1 : Considérons le système:

$$\left\{
 \begin{array}{l}
 3x_1 + 4x_2 = 238 \\
 3x_1 + 2x_2 = 188 \\
 -2x_1 + 3x_2 = 165 \\
 x_1 - 3x_2 = -27 \\
 x_1 + x_2 = 73
 \end{array}
 \right. \quad \text{et ses trois solutions approximatives:}$$

a)	$x_1^*=46, x_2^*=25$
b)	$x_1^*=47, x_2^*=25$
c)	$x_1^*=47, x_2^*=24$

Les erreurs des ces trois solutions, représentées par les variables auxiliaires, sont regroupées dans le tableau suivant:

	v_1	v_2	v_3	v_4	v_5
a)	2	2	-2	2	2
b)	-1	-1	-4	1	1
c)	3	1	-1	-2	2

On va juger ces trois solutions d'après les trois critères suivants:

- 1) la plus grande erreur absolue: $\max \{ |v_1|, |v_2|, |v_3|, |v_4|, |v_5| \}$ (norme L_∞);
- 2) l'erreur moyenne: $\frac{1}{5}(|v_1| + |v_2| + |v_3| + |v_4| + |v_5|)$ (norme L_1);
- 3) l'erreur moyenne quadratique: $\frac{1}{5}(|v_1|^2 + |v_2|^2 + |v_3|^2 + |v_4|^2 + |v_5|^2)$ (norme L_2).

On constate que chaque solution est la meilleure dans une catégorie:

a) dans 1), b) dans 2) et c) dans 3).

Sans discuter l'importance du choix d'un critère particulier, on va montrer que la recherche d'une meilleure solution dans les cas des normes L_∞ et L_1 peut être formulée par des programmes linéaires.

Rappelons, que dans le cas où on considère la norme L_2 , la meilleure solution approchée du système $\mathbf{Ax}=\mathbf{b}$ qui n'a pas de solution est la solution \mathbf{x}^* du système ${}^t\mathbf{A}\mathbf{Ax} = {}^t\mathbf{Ab}$ (qui admet toujours une solution). On laisse le soin au lecteur de détailler les transformations nécessaires pour constater que:

- 1) le problème de L_∞ -approximation :

minimiser: $z = \max\{|v_1|, |v_2|, \dots, |v_m|\}$

sous: $\mathbf{Ax}+\mathbf{v}=\mathbf{b}$

et 2) le problème de L_1 -approximation :

minimiser: $z = |v_1| + |v_2| + \dots + |v_m|$

sous: $\mathbf{Ax}+\mathbf{v}=\mathbf{b}$

non-linéaires dans leur formulations, peuvent être reformulés par les programmes linéaires suivants :

minimiser: z

sous: $z\mathbb{I} + \mathbf{Ax} \geq \mathbf{b}$

$z\mathbb{I} - \mathbf{Ax} \geq -\mathbf{b}$

$z \geq 0$

minimiser: $z = v_1 + v_2 + \dots + v_m$

sous: $v + \mathbf{Ax} \geq \mathbf{b}$

$v - \mathbf{Ax} \geq -\mathbf{b}$

$v \geq 0$

Application numérique: Trouver, en utilisant le logiciel CPLEX:

- a) la meilleure L_∞ -approximation;
- b) la meilleure L_1 -approximation;
- c) la meilleure L_2 -approximation;

du système étudié dans l'Exemple 1 de ce chapitre.

6. Programmation linéaire et JEUX de STRATEGIES.

Introduction.

La théorie des jeux doit son importance au fait qu'elle permet de traiter certaines situations de conflits militaires ou économiques. On trouve les premières tentatives de formalisation dans les travaux de E. Borel (1921, 1924, 1927). En 1928, J. von Neumann donne une base solide à cette théorie en démontrant son célèbre théorème du "minimax". Les travaux de G.B. Dantzig, D. Gale, H.W. Kuhn et A.W. Tucker montrent les relations avec la programmation linéaire - le théorème du "minimax" est une simple conséquence du théorème fondamental de la dualité.

Le caractère fondamental de la théorie des jeux est que *les gains ou les pertes de chaque joueur dépendent non seulement de ses propres initiatives, mais aussi de celles de son adversaire*. Ce caractère se retrouve clairement dans certaines questions économiques (enchères, opérations boursières) et aussi dans l'art de la guerre.

Exemples introductifs.

EXEMPLE 1 : *Deux géants de la presse quotidienne: LE FIGARO et LIBERATION se partagent le marché de quelques millions de lecteurs. Supposons que les directeurs de chaque journal doivent choisir un titre à la "une" parmi les trois qui font l'actualité:*

T₁ - "Barack Obama à la Maison Blanche";

T₂ - "Cessez-le-feu dans la bande de Gaza";

T₃ - "La crise financière";

Grâce aux sondages fréquents de l'opinion publique on peut estimer le comportement du marché. En fonction des titres à la "une" le partage du marché est donné par le tableau suivant:

		LE FIGARO		
		T ₁	T ₂	T ₃
LIBERATION		T ₁	40%:60%	70%:30%
		T ₂	60%:40%	60%:40%
		T ₃	50%:50%	50%:50%
			40%:60%	40%:60%

Pour arriver à la solution optimale les raisonnements des deux directeurs sont les suivants: LE FIGARO ne va pas du tout tenir compte de T₂ car quelle que soit la décision de LIBERATION il peut toujours gagner une plus grande part en choisissant T₁. Il faut donc raisonner sur un tableau simplifié:

		LE FIGARO	
		T ₁	T ₃
LIBERATION		T ₁	40%:60%
		T ₂	60%:40%
		T ₃	50%:50%
			40%:60%

Ici on voit clairement que LIBERATION va choisir T_2 , car les résultats de cette stratégie dominent les autres quelle que soit la décision du FIGARO. Finalement LE FIGARO doit donc raisonner sur un tableau encore plus simplifié:

		LE FIGARO	
		T_1	T_3
LIBERATION	T_2	60%:40%	70%:30%

Le choix de LIBERATION en résulte (T_2) et le choix de FIGARO est clair: il faut choisir T_1 pour que les pertes soient minimisées. Ainsi les choix optimaux sont: T_2 pour LIBERATION et T_1 pour FIGARO. Ces choix garantissent aux deux adversaires un équilibre sûr: LIBERATION va gagner au moins 60% des parts du marché et FIGARO au moins 40%. Un choix différent de la part d'un adversaire risque de provoquer une diminution de son résultat.

EXEMPLE 2 : Xavier et Yann décident de jouer au jeu suivant: ils indiquent simultanément à l'aide des doigts d'une main un nombre. Si les deux nombres sont tous les deux pairs ou impairs, Xavier donne 6€ à Yann. Si le nombre choisi par Xavier est pair et celui de Yann impair, ce dernier donne 9€ à Xavier. Enfin si le nombre de Xavier est impair et celui de Yann pair, ce dernier donne 4€ à Xavier.

Ce jeu peut être représenté par le tableau suivant, qui indique, suivant le cas, quel est le gain de Xavier.

		Yann	
		pair	impair
Xavier	pair	-6	+9
	impair	+4	-6

Plus généralement on va étudier les jeux où deux concurrents X et Y s'affrontent. Supposons que le nombre de stratégies possibles pour ces deux adversaires soit fini. Les stratégies de X seront numérotées de 1 à m , celles de Y de 1 à n . On supposera que le résultat de cet affrontement est le transfert d'une valeur a_{ij} de Y à X . Cette valeur dépend uniquement de la stratégie i de X et de la stratégie j de Y . Le jeu est donc entièrement déterminé par sa matrice de gains $A=(a_{ij})$ des dimensions $m \times n$. On utilise parfois le terme - *duopole à somme nulle*.

On va maintenant étudier deux exemples qui montrent les difficultés quand on veut généraliser les conceptions précédentes aux jeux avec somme quelconque et avec ou sans coopération.

Conflit de couple :

Roméo et Juliette sont très amoureux. Chacun peut choisir une des deux stratégies : rester fidèle ou trahir l'autre. Si les deux restent fidèles chacun est vainqueur (+2) car leur union sera durable. Si l'un trahit mais pas l'autre, alors celui qui avait une aventure est heureux (au moins pour le moment) et gagne +1 mais la personne trahie est déçue et perd un point (-1). Si les deux trahissent alors ils auront des regrets, leur union est affaiblie et chacun perd deux points (-2).

Ce type de jeu peut être représenté par une bi-matrice :

On s'aperçoit que la stratégie de rester fidèle (coopérer) est dominante pour les deux joueurs dans ce jeu. Le problème est qu'il n'est pas toujours intéressant de jouer les stratégies dominantes comme le montre l'exemple suivant.

Dilemme de prisonnier :

Deux malfaiteurs, pris en flagrant délit, sont mis en garde-à-vue séparément sans pouvoir communiquer. On offre à chacun une remise de peine quand il avoue son crime et témoigne pour impliquer l'autre (stratégie A). Celui qui avoue passera uniquement un an en prison et l'autre, qui n'a pas avoué (stratégie N), 10 ans. Si les deux avouent ils passeront en prison 5 ans chacun. Si les deux nient le fait reproché, chacun risque deux ans de prison, étant donné qu'ils sont pris en flagrant délit.

	A	N
A	$(-5; -5)$	$(-1; -10)$
N	$(-10; -1)$	$(-2; -2)$

La stratégie A est dominante pour tous les deux (ils auront 5 ans chacun) et si tous les deux choisissent la "mauvaise" stratégie c'est-à-dire ils coopèrent et nient le fait reproché, alors ils prendront uniquement deux ans.

Jeux parfaits (c'est-à-dire avec le point d'équilibre).

EXEMPLE 3 : Considérons le jeu défini par la matrice de gains suivante:

		Y		
		1	2	3
X	1	-1	+2	+1
	2	+1	+1	+2
	3	0	0	-1

Pour le joueur Y la stratégie 1 est plus avantageuse que la stratégie 2 car $a_{i,1} \leq a_{i,2}$ quelle que soit la stratégie i adoptée par le joueur X. Le joueur Y ne choisirra donc certainement pas la stratégie 2, et on peut étudier le jeu simplifié suivant:

		Y	
		1	3
X	1	-1	+1
	2	+1	+2
	3	0	-1

Il est évident que X doit choisir sa stratégie 2 et, par conséquent Y doit choisir sa stratégie 1.

EXEMPLE 4 : Considérons maintenant le jeu suivant:

		Y		
		1	2	3
X	1	+3	0	-1
	2	-2	-1	+2
	3	+2	+1	+1

Il est impossible d'effectuer des simplifications analogues au cas précédent. Cependant on constate que si X choisit sa stratégie 3 il s'assure un gain au moins égal à 1, alors que son gain peut descendre à -1 s'il choisit la ligne 1, et à -2 s'il choisit la ligne 2. La stratégie 3 donne au joueur X la meilleure garantie - quelle que soit la stratégie de son adversaire il est sûr de gagner au moins +1.

De la même manière, si Y choisit sa stratégie 1 il risque de perdre 3, alors que ses pertes dans les pires cas peuvent atteindre 1 s'il choisit la colonne 2, et 2 s'il choisit la colonne 3. Pour minimiser ses pertes le joueur Y doit utiliser sa stratégie 2. Le couple constitué par la ligne 3 et la colonne 2 constitue une situation d'équilibre qu'aucun des deux joueurs n'a intérêt à quitter: X risque de gagner moins et Y de perdre encore plus. Ce jeu apporte +1 au joueur X.

Le formalisme conduisant à ces décisions est le suivant: le joueur X cherche une stratégie i^* dite **maximine**, c'est-à-dire telle que : $\min_j a_{i^*j} = \max_i \min_j a_{ij}$.

Le joueur Y cherche une stratégie j^* dite **minimax**, c.-à-d. telle que : $\max_i a_{ij^*} = \min_j \max_i a_{ij}$.

En général on a : $\max_i \min_j a_{ij} \leq \min_j \max_i a_{ij}$.

Dans l'**EXEMPLE 2** on constate : $\max_i \min_j a_{ij} = -6 < 4 = \min_j \max_i a_{ij}$, et dans l'**EXEMPLE 4** nous avons observé l'égalité :

Cette égalité entre les valeurs des deux stratégies a pour conséquence l'existence du couple (i^*, j^*) vérifiant :

$$a_{ij^*} \leq a_{i^*j^*} \leq a_{i^*j} \quad \text{quels que soient } i \text{ et } j;$$

que l'on appelle un **point-selle** ou **point d'équilibre**; $a_{i^*j^*}$ est la **valeur** du jeu.

Inversement, si (i^*, j^*) est un point-selle, alors on a : $\max_i \min_j a_{ij} = a_{i^*j^*} = \min_j \max_i a_{ij}$. (*Preuve:* puisque (i^*, j^*) est un point-selle on a : $\max_i a_{ij} \geq a_{i^*j^*}$ quel que soit j , et $\max_i a_{ij^*} = a_{i^*j^*}$, d'où $\min_j \max_i a_{ij} = a_{i^*j^*}$. L'autre égalité se démontre de la même façon.)

Jeux de ruse.

Dans l'**EXEMPLE 2** il n'y a pas de point d'équilibre. Les résultats diffèrent à chaque fois en fonction des stratégies appliquées par les deux joueurs. Pour étudier cette situation supposons un jeu quelconque déterminé par sa matrice de gains $A = (a_{ij})$ où les deux concurrents décident de rejouer plusieurs fois. Chacun des deux joueurs emploie la ruse, c'est-à-dire tente de deviner autant que possible ce que va jouer l'adversaire et, en même temps, dissimuler sa propre intention. Sans pouvoir déterminer les causes psychologiques de chaque décision, nous pouvons examiner le nombre de cas où différentes décisions sont prises. Supposons que le jeu a été répété N fois, $N = s_1 + s_2 + \dots + s_m = r_1 + r_2 + \dots + r_n$ où s_j désigne le nombre de fois où le joueur X a appliqué sa stratégie j et r_i désigne le nombre de fois où le joueur Y a appliqué sa stratégie i . Notons par x_j la "fréquence" d'utilisation de la stratégie j par le joueur X ($x_j = s_j/N$) et par y_i la "fréquence" d'utilisation de la stratégie i par le joueur Y ($y_i = r_i/N$). Le vecteur-ligne $\mathbf{x} = (x_1, x_2, \dots, x_m)$ vérifiant : $x_1 + x_2 + \dots + x_m = 1$ et $x_1 \geq 0, x_2 \geq 0, \dots, x_m \geq 0$ est appelé **stratégie mixte** du joueur X et le vecteur-colonne $\mathbf{y} = (y_1, y_2, \dots, y_n)$ vérifiant : $y_1 + y_2 + \dots + y_n = 1$ et $y_1 \geq 0, y_2 \geq 0, \dots, y_n \geq 0$ est appelé **stratégie mixte** du joueur Y.

Dans le cas général, le gain moyen par jeu qui résulte de l'application d'une stratégie mixte \mathbf{x} par le joueur X et une stratégie mixte \mathbf{y} par le joueur Y peut être exprimé par : $\mathbf{x}A\mathbf{y}$.

En adoptant une stratégie mixte \mathbf{x} le joueur X se garantit au moins le gain : $\min_y \mathbf{x}A\mathbf{y}$, où le minimum est pris sur tous les $\mathbf{y} \geq 0$ vérifiant $y_1 + y_2 + \dots + y_n = 1$. Il est très important de noter, que ce minimum, recherché pour toutes les stratégies mixtes \mathbf{y} , est atteint pour une stratégie pure du joueur Y, $\mathbf{y}^* = (0, \dots, 1, \dots, 0)$, c'est-à-dire:

$$\min_y \mathbf{x}A\mathbf{y} = \min_j \{a_{1j}x_1 + a_{2j}x_2 + \dots + a_{mj}x_m\}$$

Preuve : Posons $s = \min_j \{a_{1j}x_1 + a_{2j}x_2 + \dots + a_{mj}x_m\}$.

On a : $\mathbf{x}A\mathbf{y} = (\mathbf{x}\mathbf{A})\mathbf{y} \geq s y_1 + s y_2 + \dots + s y_n = s(y_1 + y_2 + \dots + y_n) = s$, d'où le résultat. \square

Par conséquent, le joueur X cherche une stratégie mixte qui lui permet de maximiser ce gain (la stratégie maximine). Formellement, il faut:

$$\begin{aligned} & \text{maximiser: } z = \min_j \{a_{1j}x_1 + a_{2j}x_2 + \dots + a_{mj}x_m\} \\ [50] \quad & \text{sous: } x_1 + x_2 + \dots + x_m = 1 \\ & x_1 \geq 0, x_2 \geq 0, \dots, x_m \geq 0. \end{aligned}$$

Ce problème, à l'origine non-linéaire (la fonction-objectif n'est pas linéaire!), peut être reformulé astucieusement par le programme linéaire suivant:

$$\begin{aligned} & \text{maximiser: } z \\ [51] \quad & \text{sous: } x_1 + x_2 + \dots + x_m = 1 \\ & z - (a_{1j}x_1 + a_{2j}x_2 + \dots + a_{mj}x_m) \leq 0 \quad (j=1, 2, \dots, n) \\ & x_1 \geq 0, x_2 \geq 0, \dots, x_m \geq 0; \end{aligned}$$

où la variable z est non-astreinte. Nous pouvons maintenant annoncer le célèbre

Théorème du minimax :

Pour toutes matrice $A m \times n$ il existe deux vecteurs stochastiques x^* (un m -vecteur-ligne) et y^* (un n -vecteur-colonne) tels que:

$$\min_y \mathbf{x}^* \mathbf{A} \mathbf{y} = \max_x \mathbf{x} \mathbf{A} \mathbf{y}^*$$

où le minimum est pris sur tous les $\mathbf{y} \geq 0$ vérifiant $y_1 + y_2 + \dots + y_n = 1$, et le maximum sur tous les $\mathbf{x} \geq 0$ vérifiant $x_1 + x_2 + \dots + x_m = 1$.

Démonstration: Considérons le problème [51] et son dual

$$\begin{aligned} & \text{minimiser: } w \\ [52] \quad & \text{sous: } y_1 + y_2 + \dots + y_n = 1 \\ & z - (a_{i1}y_1 + a_{i2}y_2 + \dots + a_{in}y_n) \geq 0 \quad (i=1, 2, \dots, m) \\ & y_1 \geq 0, y_2 \geq 0, \dots, y_n \geq 0. \end{aligned}$$

Les deux programmes admettent des solutions réalisables (les deux joueurs peuvent appliquer leurs stratégies pures quelconques) alors, d'après le théorème fondamental de la dualité les deux programmes admettent des solutions optimales de valeurs z^* et w^* respectivement, telles que $z^* = w^*$. Ces deux valeurs représentent les deux valeurs, gauche et droite, dans l'expression du théorème. \square

Cette valeur commune $v (=z^*=w^*)$ est appellée **la valeur du jeu**. Le jeu est considéré comme **juste** quand $v = 0$ et **inégal** dans le cas contraire.

Ce résultat montre que finalement la meilleure "ruse" consiste à prendre ses décisions aléatoirement (suivant une certaine distribution) et dans le cas de jeux avec un point d'équilibre le "hasard" devient une "certitude".

La stratégie optimale mixte $x^* = (x_1, x_2, \dots, x_m)$ du joueur X est particulièrement facile à retrouver directement de la formulation [50] dans le cas $m=2$, par une méthode graphique, ce que montre l'exemple suivant:

Exemple 5: Etudions le jeu donné par la matrice des gains suivante:

$$\begin{bmatrix} 4 & 1 & 2 & -1 \\ -2 & 2 & -1 & 5 \end{bmatrix}$$

Le gain moyen (par jeu) dépend bien sûr des stratégies choisies par le joueur Y, mais on peut donner une expression pour son minimum en fonction de la stratégie mixte $\mathbf{x} = (x_1, x_2)$ utilisée par le joueur X:

$$z(x_1, x_2) = \min \{ 4x_1 - 2x_2; x_1 + 2x_2; 2x_1 - x_2; -x_1 + 5x_2 \}.$$

En utilisant $x_1 + x_2 = 1$, on obtient facilement la valeur du jeu en fonction de x_2 . La fig.5 donne la représentation graphique du cas présenté. Le graphique de la fonction $z(x_2)$ est en gras.

On constate facilement qu'on atteint le maximum du gain moyen minimal garanti pour la stratégie mixte $\mathbf{x} = (x_1, x_2) = (\frac{2}{3}, \frac{1}{3})$. Ce jeu est inégal car le joueur X peut s'assurer un gain d'une valeur strictement positive ($v=1>0$).

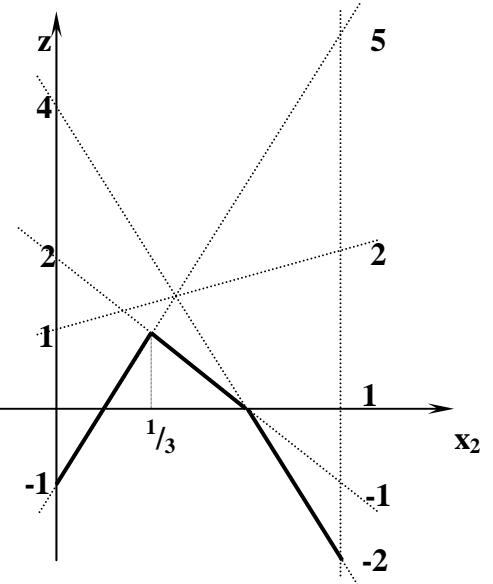


Fig.5.

Cette méthode appliquée à l'Exemple 2 donne une valeur du jeu nulle. Ce jeu est juste, malgré un certain déséquilibre des paies. La stratégie mixte optimale pour Xavier est $\mathbf{x} = (\frac{3}{5}, \frac{2}{5})$ ce qui signifie que pour atteindre l'objectif (ne pas perdre), Xavier doit dans un long jeu utiliser ses deux stratégies en proportion 3:2. Yann, par contre, pour se bien défendre doit jouer ses deux stratégies en proportion 2:3.

Deux applications.

1° Jeu “Morra”:

Chacun des deux joueurs doit miser un nombre: 1 ou 3, et parallèlement deviner la mise de son adversaire. On doit donc proposer une paire $(m; p)$ où m est le nombre misé et p est le pari sur la mise de l'adversaire. Si les joueurs se sont trompés ou ont bien deviné tous les deux les mises de leurs adversaires, alors le résultat du jeu est nul. Dans le cas où un seul joueur a bien deviné la mise de son adversaire il reçoit de ce dernier la paie égale à la somme des deux nombres misés. Le problème de nature psychologique dans ce jeu est qu'en misant 3 vous aller augmenter la valeur de la paie sans pourtant être sûr de gagner cette valeur élevée – vous risquez bien de la perdre.

Les quatre stratégies pures de chaque joueur sont $(1; 1)$, $(1; 3)$, $(3; 1)$, $(3; 3)$ et la matrice des gains de ce jeu s'écrit:

	$(1; 1)$	$(1; 3)$	$(3; 1)$	$(3; 3)$
$(1; 1)$	0	2	-4	0
$(1; 3)$	-2	0	0	4
$(3; 1)$	4	0	0	-6
$(3; 3)$	0	-4	6	0

Elle est antisymétrique donc la valeur du jeu est nulle (le jeu est juste). Quelle stratégie faut-il appliquer pour ne pas perdre? Le programme linéaire correspondant:

$$\begin{aligned}
 & \text{maximiser: } z \\
 & \text{sous:} \\
 & \quad x_1 + x_2 + x_3 + x_4 = 1 \\
 & \quad z + 2x_2 - 4x_3 \leq 0 \\
 & \quad z - 2x_1 + 4x_4 \leq 0 \\
 & \quad z + 4x_1 - 6x_4 \leq 0 \\
 & \quad z - 4x_2 + 6x_3 \leq 0 \\
 & \quad x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0.
 \end{aligned}$$

admet $\mathbf{x} = (0, \frac{2}{3}, \frac{1}{3}, 0)$ comme une de ses solutions optimales. Elle se traduit par le fait que le joueur X doit utiliser seulement ses stratégies 2 et 3 en proportion 2:1. L'interdiction de jouer la stratégie 4 (voir aussi Ex. 30) est surprenante car intuitivement la stratégie 4 est meilleure que la stratégie 3 (si on gagne, on gagne plus et si on perd, on perd moins!).

Une des solutions optimales du problème dual:

$$\begin{aligned}
 & \text{minimiser: } w \\
 & \text{sous:} \\
 & \quad y_1 + y_2 + y_3 + y_4 = 1 \\
 & \quad w - 2y_2 + 4y_3 \geq 0 \\
 & \quad w + 2y_1 - 4y_4 \geq 0 \\
 & \quad w - 4y_1 + 6y_4 \geq 0 \\
 & \quad w + 4y_2 - 6y_3 \geq 0 \\
 & \quad y_1 \geq 0, y_2 \geq 0, y_3 \geq 0, y_4 \geq 0.
 \end{aligned}$$

est $\mathbf{y} = (0, \frac{3}{5}, \frac{2}{5}, 0)$. Le joueur Y doit utiliser seulement ses stratégies 2 et 3 en proportion 3:2.

Supposons maintenant que les deux joueurs choisissent leur mise simultanément, mais le joueur X attend pour parier sur la mise du joueur Y , que celui-ci ait fait son pari. Est-ce que la connaissance du pari de l'adversaire permet au joueur X de gagner ce jeu grâce au pari astucieux? Dans la nouvelle situation, le joueur X peut bien sûr ignorer cette information supplémentaire ou, au contraire, l'utiliser, en réagissant par la même proposition ou choisir le résultat complémentaire. Le joueur X a donc huit stratégies et le jeu se résume par la matrice ci-contre.

	$(I; I)(I; 3)(3; I)(3; 3)$			
$(I; I)$	0	2	-4	0
$(I; 3)$	-2	0	0	4
$(3; I)$	4	0	0	-6
$(3; 3)$	0	-4	6	0
$(I; y)$	0	0	-4	4
$(I; \neg y)$	-2	2	0	0
$(3; y)$	4	-4	0	0
$(3; \neg y)$	0	0	6	-6

En adoptant la stratégie mixte $\mathbf{x} = (0, \frac{30}{52}, \frac{18}{52}, 0, 0, \frac{3}{52}, 0, \frac{1}{52})$ le joueur X se garantit un gain moyen par jeu strictement positif de $\frac{6}{52}$. Ceci signifie que le joueur X doit légèrement changer sa stratégie par rapport à la situation précédente, et utiliser l'information donnée par son adversaire mais seulement dans 4 cas sur 52, en faisant le pari complémentaire au pari de son adversaire, en misant 1 et 3 en proportion 3:1. Pour ne pas perdre plus dans cette nouvelle situation, son adversaire doit jouer toutes ses quatre stratégies en proportion 15:18:10:9, car sa stratégie mixte optimale est : $\mathbf{y} = (\frac{15}{52}, \frac{18}{52}, \frac{10}{52}, \frac{9}{52})$.

2° Poker simplifié de H.W. Kuhn (1950):

Dans des jeux de cartes comme le poker on observe parfois des stratégies de bluff. Avec une main faible il s'agit de prendre des risques et continuer le jeu pour forcer psychologiquement l'adversaire à abandonner, ou bien avec une main très forte, commencer le jeu timidement pour pousser l'adversaire à jouer fort et augmenter ainsi sa perte. Sur l'exemple du poker (très) simplifié nous allons montrer que ces types de stratégies trouvent une justification mathématique.

Voici la description du jeu : deux personnes X et Y jouent seulement avec trois cartes de valeurs 1, 2 et 3. Au départ, pour recevoir une carte, chacun des deux joueurs met en jeu une somme d'argent fixée à l'avance (une mise). Etant en possession d'une carte, les joueurs, à tour de rôle, annoncent "passe" ou "plus". En annonçant plus le joueur doit doubler sa mise. Par convention ce jeu peut se dérouler d'après l'un des cinq scénarios suivants:

- $X \text{ passe}, Y \text{ passe} \dots$ les joueurs découvrent leur carte et le possesseur de la carte plus grande gagne une mise.
- $X \text{ passe}, Y \text{ plus}, X \text{ passe} \dots$ Y gagne une mise.
- $X \text{ passe}, Y \text{ plus}, X \text{ plus} \dots$ les joueurs découvrent leur carte et le possesseur de la carte plus grande gagne deux mises.
- $X \text{ plus}, Y \text{ passe} \dots$ X gagne une mise.
- $X \text{ plus}, Y \text{ plus} \dots$ les joueurs découvrent leur carte et le possesseur de la carte plus grande gagne deux mises.

En faisant, pour l'instant, abstraction de la valeur de la carte dont il est en possession, le joueur X peut avoir trois types de réaction:

- 1° soit il commence par **passe** et si Y annonce **plus** il finit par **passe**;
- 2° soit il commence par **passe** et si Y annonce **plus** il finit par **plus**;
- 3° soit il commence par **plus**.

Ainsi nous allons pouvoir "coder" toutes les stratégies pures de X par des triplets (x_1, x_2, x_3) $x_i=1, 2$ ou 3 , en fonction de la réaction choisie quand il est en possession de la carte de valeur i . Par exemple 312 signifie que:

étant en possession de 1, le joueur X commence par **plus**;
quand il a 2 il commence par **passe** et si Y annonce **plus** il finit par **passe**, et
quand il a 3 dans sa main, il commence par **passe** et si Y annonce **plus** il finit par **plus**.

Les comportements du joueur Y diffèrent de ceux du joueur X car il réagit en fonction de l'annonce de X . Nous pouvons distinguer formellement ses quatre réactions:

- 1° soit il **passe** quelle que soit l'annonce du X ;
- 2° si X : **passe**, il annonce **passe**, si X : **plus**, il annonce **plus**;
- 3° si X : **passe**, il annonce **plus**, si X : **plus**, il annonce **passe**;
- 4° soit il annonce **plus** quelle que soit l'annonce du X .

De la même façon nous pouvons maintenant "coder" toutes les stratégies pur de Y par des triplets (y_1, y_2, y_3) $y_i=1, 2, 3$ ou 4 en fonction de la réaction choisie quand il est en possession de la carte de valeur i . Par exemple **124** signifie que

le joueur Y annonce **passe**, s'il est en possession de 1,
il répète l'annonce du X quand il a 2, et
il annonce toujours **plus** s'il a 3 dans sa main.

Pour chaque paire de stratégies pures des deux joueurs, la valeur du gain varie en fonction des cartes distribuées. Le tableau suivant présente les gains de X quand il applique la stratégie **312** et son adversaire sa stratégie **124**:

la main		annonces	le gain
X	Y	X - Y - X	de X
1	2	plus - plus	-2
1	3	plus - plus	-2
2	1	passe-passe	+1
2	3	passe-plus-passe	-1
3	1	passe-passe	+1
3	2	passe-passe	+1

A ce stade il faut déterminer une valeur de gain unique pour la mettre dans la matrice des gains. On va décider de choisir la moyenne des ces résultats. Dans l'exemple cité il s'agit de prendre:

$$\frac{1}{6}(-2-2+1-1+1+1) = \frac{-1}{3}.$$

Ces démarches conduisent à une matrice énorme quand on pense aux $3 \times 3 \times 3 = 27$ stratégies formelles de X et $4 \times 4 \times 4 = 64$ stratégies formelles de Y . On peut simplifier cette analyse en remarquant que le joueur X a au moins une stratégie optimale où il renonce à sa réaction 2° quand il a 1 en main, et il abandonne sa réaction 1 quand il a 3 en main. De même, le joueur Y a au moins une stratégie optimale où il renonce à ses réactions 2° et 4° quand il a 1 en main, et il utilise seulement sa réaction 4° quand il a 3 en main. On peut donc exclure de l'analyse toutes les stratégies de la forme: $(2, x_2, x_3)$, $(x_1, x_2, 1)$, $(2, y_2, y_3)$, $(4, y_2, y_3)$, $(y_1, y_2, 1)$, $(y_1, y_2, 2)$, $(y_1, y_2, 3)$.

Ceci réduit le nombre de stratégies de X à 12 et le nombre de stratégies de Y à 8, et ouvre de nouvelles possibilités de réduction. Ci-contre la matrice finale qui en résulte:

Le programme linéaire correspondant admet $\mathbf{x} = (\frac{1}{3}, 0, 0, \frac{1}{2}, \frac{1}{6}, 0, 0, 0, 0)$ comme solution optimale. Elle se traduit par le fait que le joueur X doit suivre les instructions très simples et quand il a en main :

–1– utiliser les stratégies 1 et 3 en proportion 5:1;

$$(\text{car } x_1 + x_2 + x_3 + x_4 = \frac{5}{6} \text{ et } x_5 + x_6 + x_7 + x_8 = \frac{1}{6});$$

–2– utiliser les stratégies 1 et 2 en proportion 1:1; ($\text{car } x_1 + x_2 + x_5 + x_6 = \frac{3}{6}$ et $x_3 + x_4 + x_7 + x_8 = \frac{1}{2}$);

–3– utiliser les stratégies 2 et 3 en proportion 1:1; ($\text{car } x_1 + x_3 + x_5 + x_7 = \frac{3}{6}$ et $x_2 + x_4 + x_6 + x_8 = \frac{3}{6}$).

	114	124	314	324
112	0	0	$-\frac{1}{6}$	$-\frac{1}{6}$
113	0	$\frac{1}{6}$	$-\frac{1}{3}$	$-\frac{1}{6}$
122	$-\frac{1}{6}$	$-\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
123	$-\frac{1}{6}$	0	0	$\frac{1}{6}$
312	$\frac{1}{6}$	$-\frac{1}{3}$	0	$-\frac{1}{2}$
313	$\frac{1}{6}$	$-\frac{1}{6}$	$-\frac{1}{6}$	$-\frac{1}{2}$
322	0	$-\frac{1}{2}$	$\frac{1}{3}$	$-\frac{1}{6}$
323	0	$-\frac{1}{3}$	$\frac{1}{6}$	$-\frac{1}{6}$

La première instruction préconise le bluff une fois sur six, quand on a la carte de valeur 1, car le joueur X doit annoncer plus étant en possession de la carte la plus faible. La troisième, elle aussi propose de bluffer une fois sur deux quand on est sûr de gagner.

Ce jeu assure au joueur Y le gain $\frac{1}{18}$ ($v = -\frac{1}{18}$) s'il applique sa stratégie mixte optimale $\mathbf{y} = (\frac{2}{3}, 0, 0, \frac{1}{3})$; c'est-à-dire s'il est en possession de:

–1– utiliser les stratégies 1 et 3 en proportion 2:1; ($\text{car } y_1 + y_2 = \frac{2}{3}$ et $y_3 + y_4 = \frac{1}{3}$);

–2– utiliser seulement ses stratégies 1 et 2 en proportion 2:1; ($\text{car } y_1 + y_3 = \frac{2}{3}$ et $y_2 + y_4 = \frac{1}{3}$);

–3– utiliser toujours sa stratégie 4; ($\text{car } y_1 + y_2 + y_3 + y_4 = 1$).

La première instruction préconise le bluff une fois sur trois, quand on a la carte de valeur 1, car le joueur Y doit répondre “plus” à “passe” étant en possession de la carte la plus faible. Il n'a pas intérêt par contre à sous-estimer ses cartes.

Liste des symboles :

$\alpha(G) = \max \{ |S|; S \text{ est un stable de } G \}$ – [alpha] le nombre de stabilité;

$\tau(G) = \min \{ |T| ; T \text{ est un transversal des arêtes de } G \} - [\tau]$ le nombre de transversalité

$\nu(G) = \max\{|\mathcal{C}|; \mathcal{C} \text{ est un couplage de } G\} - [nu] \text{ la cardinalité maximum d'un couplage;}$

$\rho(G) = \min \{ |F| ; F \text{ est un recouvrement de } G \} - [\rho]$ la cardinalité minimum d'un

recouvrement ;

$\chi(G) = \min \{k; G \text{ est } k\text{-colorable}\} - [khi] \text{ le nombre chromatique;}$

$\omega(G) = \max\{p; K_p \text{ est une clique de } G\}$ – [oméga] le nombre maximum de sommets qui

forment une clique ;

$\kappa(G)$ – [kappa] connectivité = le nombre minimum de sommets dont l'élimination

disconnecte G ou le réduit à un sommet unique.

Bibliographie.

- [1] C. BERGE. *Graphes et hypergraphes*. DUNOD, 2^{ème} édition, 1973.
 - [2] J.A. BONDY, U.S.R. MURTY. *Graph Theory with Applications*, AMERICAN ELSEVIER, New York, 1976.
 - [3] I. CHARON, A. GERMA, O. HUDRY. *Méthodes d'optimisation combinatoire..* MASSON, 1996.
 - [4] G. FINKE at al. *Recherche Opérationnelle et réseaux* traité IGAT, HERMES, 2002
 - [5] M. R. GAREY, D. S. JOHNSON. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. FREEMAN, 1979.
 - [6] M.C. GOLUMBIC. *Algorithmic Graph Theory and Perfect Graphs*. ACADEMIC PRESS, 1980.
 - [7] R. GOULD. *Graph Theory*. The Benjamin/Cummings Publishing Company, 1988.
 - [8] L. LOVASZ, M.D. PLUMMER. *Matching Theory*. Annals of Discrete Mathematics (29) ELSEVIER, 1986.
 - [9] J.R. EVANS, E. MINIEKA. *Optimization Algorithms for Networks and Graphs*. DEKKER, 1992.
 - [10] M. SAKAROVITCH. *Optimisation combinatoire*. HERMANN, 1984.
 - [11] A. SCHRIJVER. *Combinatorial Optimization*. Vol. A, B, C, SPRINGER, 2003.
 - [12] N.H. XUONG *Mathématiques discrètes et informatique*. MASSON, 1992.

Compléments

1° Point Fixe

THEOREME DE BROUWER (le cas $n=3$ en 1909): Toute application continue f d'un compact convexe de \mathbb{R}^n dans lui même admet un point fixe (c'est-à-dire un p tel que $f(p)=p$).

Pour $n=1$ c'est une simple conséquence de la propriété Darboux de la fonction $f(x)-x$. En effet, si f est une application continue d'un segment $[a;b]$ dans lui même alors on a : $f(a) \geq a$ et $f(b) \leq b$. Cela veut dire que $f(x)-x \geq 0$ pour $x=a$ et $f(x)-x \leq 0$ pour $x=b$, alors $\exists p$ tel que $f(p)-p=0$.

Généralisé à l'espace de fonctions, ce théorème permet d'affirmer l'existence de la solution de l'équation $dy/dx=f(x,y)$ avec les conditions initiales (x_0, y_0) , c'est-à-dire permet d'affirmer l'existence d'une fonction $g(x)$ telle que : $dg(x)/dx = f(x, g(x))$ et $g(x_0) = y_0$. Après la transformation en équation intégrale : $g(x) = y_0 + \int_{x_0}^x f(t, g(t)) dt$ on peut introduire l'application h qui, à

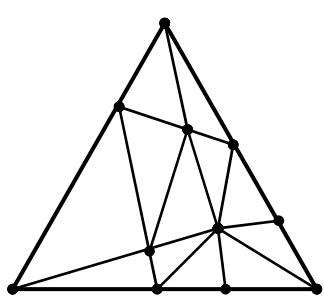
chaque fonction $\varphi(x)$ associe une fonction $h_\varphi(x) = y_0 + \int_{x_0}^x f(t, \varphi(t)) dt$.

Clairement, la fonction recherchée $g(x)$ est le point fixe ($h_g=g$) de l'application h .

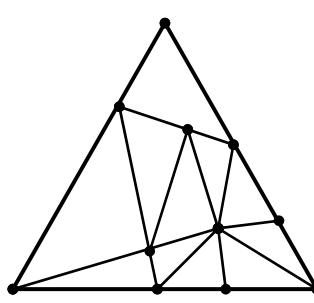
Soit $\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_n$ un ensemble de $n+1$ vecteurs affinement indépendants de \mathbb{R}^n . On appelle n -simplexe l'enveloppe convexe de ces points, c'est-à-dire : $\text{conv}\{\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_n\}$.

Ainsi un segment de la droite, un triangle dans le plan ou un tétraèdre dans l'espace sont des n -simplexes pour $n=1, 2, 3$.

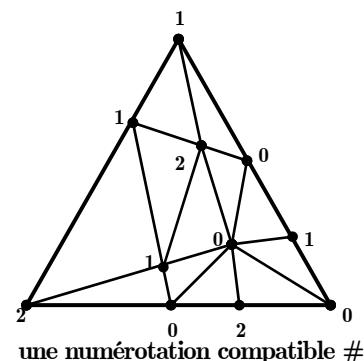
On appelle *triangulation (conforme)* une subdivision en nombre fini de n -simplexes à condition que l'intersection de deux d'entre eux soit une facette commune (ou l'ensemble vide). Dans la suite nous nous limitons au cas $n=2$ (par le souci de simplicité; le cas $n=1$ est trop trivial) :



une triangulation conforme



triangulation non conforme



une numérotation compatible #

Soit Δ un triangle (2-simplexe) de sommets $\{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2\}$ dans \mathbb{R}^2 triangulé par un ensemble fini \mathbf{T} de triangles dont on note S l'ensemble des sommets. Soit $\# : S \rightarrow \{0, 1, 2\}$ une application vérifiant la condition de compatibilité :

$$\text{si } \mathbf{v} \in \text{conv}\{\mathbf{v}_{i_0}, \dots, \mathbf{v}_{i_k}\} \text{ alors } \#(\mathbf{v}) \in \{i_0, \dots, i_k\}$$

(autrement dit, si un sommet est sur une face de Δ , le numéro que $\#$ lui attribue est celui de l'un des sommets de la face). On dit qu'un triangle de \mathbf{T} est bien numéroté si ses trois sommets sont numérotés 0, 1 et 2 (dans un ordre quelconque).

LEMME DE SPERNER :

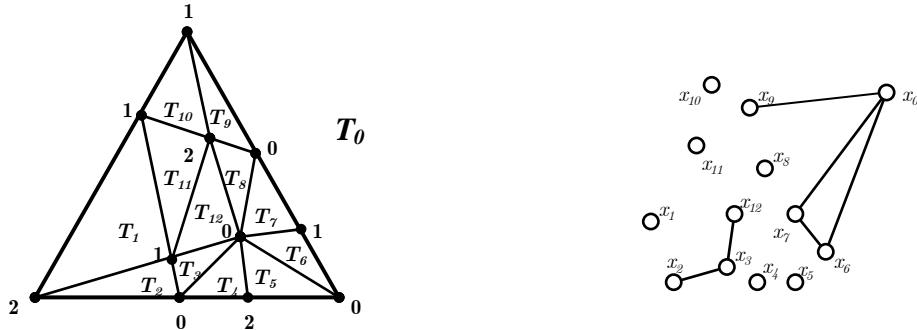
Le nombre de triangles bien numérotés de \mathbf{T} est impair (en particulier non nul).

PREUVE :

Notons T_0 – l’extérieur du triangle Δ et T_1, T_2, \dots, T_r les triangles de la triangulation \mathbf{T} . Construisons un graphe G avec l’ensemble de sommets $X=\{x_0, x_1, \dots, x_r\}$, en mettant une arête entre les sommets x_i et x_j si et seulement si les sommets de l’arête commune de T_i et T_j portent les numéro 0 et 1. On voit facilement que :

- le degré du sommet x_0 est impair ;
- $d(x_i) \leq 2$ pour $i=1, 2, \dots, r$.

Il existe donc parmi ces derniers, un nombre impair de sommets de degré 1, qui représentent évidemment les triangles bien numérotés. ■



construction du graphe G dans la démonstration du LEMME DE SPERNER

Les grandes lignes de la

DEMONSTRATION DU THEOREME DE BROUWER :

Chaque point \mathbf{v} d’un triangle de sommets $\{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2\}$ possède une représentation unique sous forme $\mathbf{v} = \lambda_0 \mathbf{v}_0 + \lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2$ avec $\lambda_0 + \lambda_1 + \lambda_2 = 1$ et $\lambda_i \geq 0$. Les trois nombres $(\lambda_0; \lambda_1; \lambda_2)$ sont les coordonnées barycentriques du point \mathbf{v} . Soit $f: \Delta \rightarrow \Delta$ une application continue et supposons que $f(\lambda_0; \lambda_1; \lambda_2) = (\lambda'_0; \lambda'_1; \lambda'_2)$. Notons $\mathbf{F}_i = \{(\lambda_0; \lambda_1; \lambda_2) \in \Delta \mid \lambda_i \geq \lambda'_i\}$. L’ensemble $\mathbf{F}_0 \cap \mathbf{F}_1 \cap \mathbf{F}_2$ (si non vide) est composé des points fixes de f . Pour chaque triangulation de Δ , il existe une numérotation compatible $\#$, telle que chaque sommet numéroté i appartient à \mathbf{F}_i . Le lemme de Sperner, la continuité de f et la compacité permettent de conclure que $\mathbf{F}_0 \cap \mathbf{F}_1 \cap \mathbf{F}_2 \neq \emptyset$. ■

TRAVAUX PRATIQUES :

Compléter les démonstrations ci-dessus. Préciser pourquoi :

- 1) dans la preuve du LEMME DE SPERNER le degré du sommet x_0 est impair et pourquoi les autres sommets de degré 1 représentent les triangles bien numérotés ?
- 2) dans la démonstration du THEOREME DE BROUWER :
 - il existe une numérotation compatible $\#$, telle que chaque sommet numéroté i appartient à \mathbf{F}_i ;
 - $\mathbf{F}_0 \cap \mathbf{F}_1 \cap \mathbf{F}_2$ (si non vide) est composé des points fixes de f ;
 - conclure que $\mathbf{F}_0 \cap \mathbf{F}_1 \cap \mathbf{F}_2 \neq \emptyset$.
- 3) Formuler et démontrer le LEMME DE SPERNER dans \mathbb{R}^n .
- 4) Interpréter les cas triviaux $n=0$ et 1 .

2° Capacité de Shannon

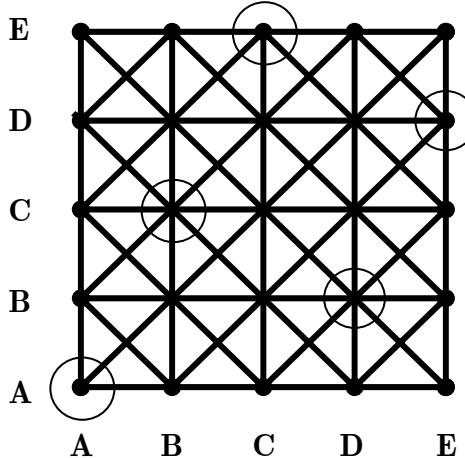
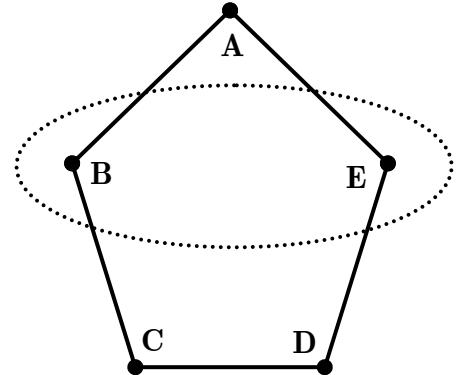
(1956):

On peut émettre cinq signaux A, B, C, D, E différents, mais à la réception certains d'entre eux peuvent être confondus (ce qui est noté par une arête). Quel est le nombre maximum de signaux qu'on peut adopter pour qu'à la réception il n'y ait pas de confusion possible entre les signaux adoptés?

Le problème revient à chercher un stable maximum.

Au lieu de signaux d'une lettre, on pourrait utiliser des mots de deux lettres, à condition que ces mots ne peuvent être confondus à la réception.

Par exemple, avec {B, E} on peut former : BB, BE, EB, EE – 4 mots. Mais on peut utiliser un code encore plus riche avec : AA, BC, CE, DB, ED.



Ces cinq mots forment un stable maximum dans un graphe que l'on appelle *produit normal* des graphes $G=(X,E)$ et $H=(Y,F)$: c'est le graphe $G \bullet H$ ayant $X \times Y$ pour ensemble de sommets, et où deux sommets (x,y) et (x',y') sont reliés si et seulement si on a: soit $x=x'$ et $\{y,y'\} \in F$, soit $\{x,x'\} \in E$ et $y=y'$, soit $\{x,x'\} \in E$ et $\{y,y'\} \in F$.

Dans l'exemple initial G est le cycle de longueur 5 et on constate que $a(G^2)=5$.

On appelle *capacité du graphe G* ("zero-error capacity") le nombre : $\chi(G) = \sup \sqrt[k]{\alpha(G^k)}$.

On a démontré que la capacité du cycle de longueur 5 est $\sqrt{5}$. On peut donc améliorer le code en utilisant des mots à deux lettres. Par contre, si le graphe G vérifie : $\alpha(G)=$ le nombre minimum de cliques qui partitionnent l'ensemble de sommets, il est facile de démontrer que $\chi(G)=\alpha(G)$ (le nombre de stabilité), que signifie que le code ne peut pas être amélioré.

Quand on repère le signal par la fréquence de sa modulation, deux signaux pourront être confondus si et seulement si, à la réception, les intervalles de fréquences correspondant à ces deux signaux s'intersectent. On sait alors que le graphe G représente une famille d'intervalles et il possède la propriété citée ci-dessus donc on ne peut pas améliorer le code en utilisant des mots plus longs.

3° Introduction à la complexité

Qu'est-ce qu'un algorithme efficace ?

La distinction entre l'existence d'objets mathématiques et leur construction est relativement récente. Le souci de prendre en considération le temps d'exécution des algorithmes n'apparaît que dans les années 1950–1960, dans plusieurs travaux, dont ceux de John von Neumann bien après les travaux d'Alan Turing, le concepteur du premier modèle mathématique de l'ordinateur. L'idée de la caractérisation des algorithmes efficaces remonte à Jack Edmonds (1965).

Si l'on perçoit assez bien ce que *fait* un algorithme, on a plus de difficulté à voir ce qu'il *est*. Dans les années 30 du siècle dernier, le mathématicien Alan Turing parvient à formaliser l'idée intuitive qu'un algorithme est une procédure que l'on peut exécuter mécaniquement, sans réfléchir. Il modélisa un processus de calcul par une machine, simple mais précise, capable d'effectuer quatre actions élémentaires. Ainsi on peut dédier une machine à toute procédure décomposable en une séquence desdites actions. La réalisation effective d'une telle machine n'est pas essentielle : *une machine de Turing* est une formulation mathématique d'*un algorithme*. Une idée intuitive et informelle d'un algorithme suffit pour comprendre ce chapitre.

Ainsi, pour résoudre un problème on opte pour la méthode algorithmique. Alors une première question se pose : existe-t-il des problèmes qui résistent à cette méthode? Pour répondre à une question si générale donnons d'abord la définition suivante :

Un *problème de reconnaissance* (*de décision*) est une question à laquelle on doit répondre par OUI ou par NON. Donner une réponse à un problème de reconnaissance consiste à fournir un algorithme pour le résoudre. Un problème de reconnaissance est dit *décidable* s'il existe un tel algorithme.

Le résultat suivant donne un aperçu des limites des machines de Turing quant à leur capacité à résoudre des problèmes : *il existe des problèmes de reconnaissance indécidables* !

Par ailleurs, il est facile de démontrer l'existence des problèmes qui résistent à la méthode algorithmique, mais il est beaucoup plus difficile d'en trouver un exemplaire précis. Grâce aux travaux de Gödel, ce fut fait. Mais ce sujet déborde de l'objectif de ce document et, dans la suite, on s'intéresse uniquement aux problèmes décidables.

La première idée qui vient à l'esprit quand on désire évaluer l'efficacité d'un algorithme est de l'essayer sur une série de problèmes que cet algorithme est supposé résoudre et mesurer aussi ses performances quand la taille des exemples augmente. Mais l'expérience montre que sur des problèmes constitués avec des nombres tirés au hasard, les algorithmes se comportaient mieux que dans les cas réels. On a abandonné l'idée de comportement moyen et on s'est donc tourné vers l'étude du plus mauvais cas. On associe à chaque algorithme un majorant du nombre d'opérations nécessaires ou plutôt une fonction reliant la taille du problème à cette borne. C'est-à-dire qu'on s'intéresse au comportement asymptotique de l'algorithme.

On dit qu'un algorithme est *polynomial* si son comportement pour des données arbitrairement grandes se décrit de la façon suivante :

pour tout n , pour des données qui n'occupent pas plus de n octets de mémoire d'un ordinateur, l'algorithme s'exécute en moins de cn^k opérations élémentaires, où c et k sont des constantes.

Cette définition impose seulement la manière dont le temps d'exécution augmente en fonction des données (le terme polynomial vient du fait que cn^k est un polynôme).

Un algorithme polynomial est dit *efficace*. Le problème justifiable d'un tel algorithme est dit *facile*.

Dans l'ensemble des problèmes décidables où la solution consiste à répondre soit OUI soit NON, on peut définir les trois classes : \mathcal{P} , \mathcal{NP} et $\text{co-}\mathcal{NP}$. L'ensemble des problèmes faciles forme une classe notée \mathcal{P} .

\mathcal{P} est donc l'ensemble des problèmes de décision que l'on peut *résoudre* avec un algorithme polynomial.

Les problèmes décidables qui n'appartiennent pas à la classe \mathcal{P} sont dits *difficiles* ou *intractables*. Mais ne pas connaître aujourd'hui d'algorithme polynomial résolvant un problème donné ne signifie pas qu'on n'en trouvera jamais. On conçoit aisément que la démonstration de l'inexistence d'un algorithme polynomial n'est pas une chose facile.

On contourne cette difficulté en définissant une classe plus large appelée \mathcal{NP} (*Non-deterministic Polynomial* en anglais).

Un problème de reconnaissance est de la classe \mathcal{NP} s'il existe un algorithme polynomial pour *vérifier* qu'une solution proposée ou devinée permet d'affirmer la réponse OUI.

On met donc dans cette classe tous les problèmes pour lesquels on peut convaincre un tiers de la réponse OUI si elle est vraie et si quelqu'un nous souffle le certificat, mais on ne peut pas nécessairement la trouver. Notons qu'on cherche seulement à *prouver* (et non à *trouver*) une réponse OUI. Et on ne s'intéresse pas à la justification d'une réponse NON. Il y a une absence de symétrie importante entre les OUI et NON. Pour un problème de la classe \mathcal{NP} on peut être incapable de proposer un algorithme polynomial pour vérifier l'instance NON.

Un problème de reconnaissance est de la classe $\text{co-}\mathcal{NP}$ s'il existe un algorithme polynomial pour *vérifier* qu'une solution proposée ou devinée permet d'affirmer la réponse NON.

Il est évident que \mathcal{P} est entièrement inclus dans \mathcal{NP} et dans $\text{co-}\mathcal{NP}$, car s'il existe un moyen de *trouver* rapidement, on n'a qu'à exécuter l'algorithme pour se convaincre. L'algorithme prend alors lui-même valeur de certificat.

Pour uniformiser la notation on pourrait de la même manière distinguer la classe \mathcal{P} et $\text{co-}\mathcal{P}$. Ceci est une complication inutile car évidemment $\mathcal{P}=\text{co-}\mathcal{P}$.

Est-ce que : $\mathcal{NP} \neq \text{co-}\mathcal{NP}$? Aujourd'hui cette question reste ouverte.

Parmi les plus grands défis actuels on trouve le problème “ $\mathcal{P}=\mathcal{NP}$?” qui consiste à prouver que les deux ensembles, \mathcal{P} et \mathcal{NP} , sont égaux ou non.

Une réponse à cette question coûterait un million de dollars au Clay Mathematics Institute, qui a lancé le concours Millennium Prize Problems. L'importance de ce problème est liée au fait que l'efficacité des cryptosystèmes de l'informatique et des banques est fondée sur l'hypothèse $\mathcal{P} \neq \mathcal{NP}$. Il s'agit aussi de comprendre les limites théoriques des ordinateurs, indépendamment du progrès technique. Même pour des problèmes de taille raisonnable, le nombre d'opérations pour la résolution peut être déraisonnablement grand, rendant le problème intractable.

En Combinatoire, qui étudie les ensembles finis, trouver un algorithme quelconque pour résoudre un problème ne pose pas de problème car, très souvent, toute instance du problème ne contient qu'un nombre fini de configurations à examiner. Mais, savoir qu'un algorithme s'arrête après un nombre fini d'étapes peut être d'un intérêt nul sur le plan opérationnel.

Aujourd'hui, beaucoup de problèmes sont dans \mathcal{P} . Par exemple, décider si OUI ou NON un nombre est divisible par 2, 3, ou par n'importe quel nombre. Un problème classique

(forte connexité d'un graphe orienté): étant donné le plan d'une ville, vérifier si l'administration urbaine a bien fait son travail et qu'on peut aller de n'importe quel endroit à n'importe quel autre, en respectant les rues en sens unique.

Problèmes NP -complets.

Puisque l'on ne sait pas aujourd'hui si les deux classes \mathcal{P} et NP coïncident ou si l'inclusion est stricte, on peut s'intéresser aux problèmes de NP "les plus difficiles" au sens où l'existence d'un algorithme polynomial pour résoudre un tel problème entraînerait l'existence d'algorithmes polynomiaux pour résoudre n'importe quel problème de NP . On les appelle problèmes NP -complets ou universels.

La notion essentielle qui permet de préciser la conception de NP -complétude est celle de *réduction polynomiale*: on dit qu'on peut réduire le problème de décision \mathcal{D} en un autre problème de décision \mathcal{D}' (on note cette relation $\mathcal{D} \leq \mathcal{D}'$), s'il existe un algorithme polynomial qui transforme chaque instance I de \mathcal{D} en une instance I' de \mathcal{D}' et le problème \mathcal{D} admet la réponse OUI pour l'instance I si et seulement si le problème \mathcal{D}' admet la réponse OUI pour l'instance I' .

Il faut souligner que la transformation polynomiale doit être décrite sans qu'on sache si la réponse à l'instance I est OUI ou NON. Aussi, il n'est pas nécessaire d'atteindre toutes les instances de \mathcal{D}' ; en revanche, il est indispensable de transformer toutes les instances de \mathcal{D} en instances de \mathcal{D}' . Ainsi $\mathcal{D} \leq \mathcal{D}'$ signifie que \mathcal{D}' est au moins aussi difficile que \mathcal{D} .

S'il existe un algorithme polynomial pour résoudre \mathcal{D}' et \mathcal{D} peut être réduit à \mathcal{D}' , alors il existe un algorithme polynomial pour résoudre \mathcal{D} . En effet, pour obtenir la réponse relative à l'instance I de \mathcal{D} , il suffit de transformer I polynomialement en I' de \mathcal{D}' et de résoudre celle-ci (toujours polynomialement); comme la transformation conserve les réponses, on résout I en résolvant I' .

On peut maintenant donner une définition plus rigoureuse :

un problème \mathcal{D} est NP -complet s'il est NP et si tout problème NP peut être réduit polynomialement à \mathcal{D} .

Le problème principal posé par cette définition était le suivant : existe-t-il des problèmes NP -complets ?

En 1971, Stephen Cook et indépendamment Leonid Levin ont présenté des problèmes concrets auxquels pouvaient être réduits tous les problèmes de NP en un temps polynomial. Ils ont ainsi démontré que la famille de problèmes universels est non vide. Il est remarquable que des problèmes aussi différents que ceux qui relèvent des mathématiques discrètes ou de la théorie des nombres puissent être réduits les unes aux autres. Depuis, la liste des problèmes NP -complets grandit vite (plus de mille sont connus à ce jour et les démonstrations de NP -complétude sont devenues pratique courante).

Pour certains problèmes, personne n'arrive à donner une solution réalisable en un temps accessible à l'homme et aux ordinateurs ou à démontrer un théorème qui saisit l'essentiel du problème. Pour démontrer qu'un problème est intraitable (= trop difficile), la principale manière de faire est aujourd'hui de montrer qu'il est *universel*. Pour montrer qu'un problème \mathcal{D} est NP -complet, il suffit de bien choisir un problème \mathcal{D}' dont on sait déjà qu'il est NP -complet et de démontrer que $\mathcal{D} \leq \mathcal{D}'$.

Une grande partie des problèmes appliqués appartient à la famille des problèmes "universels". C'est le cas du cycle hamiltonien.

QUESTION: dans les graphes ci-dessous (Fig.1 et 2), existe-t-il un cycle hamiltonien ?

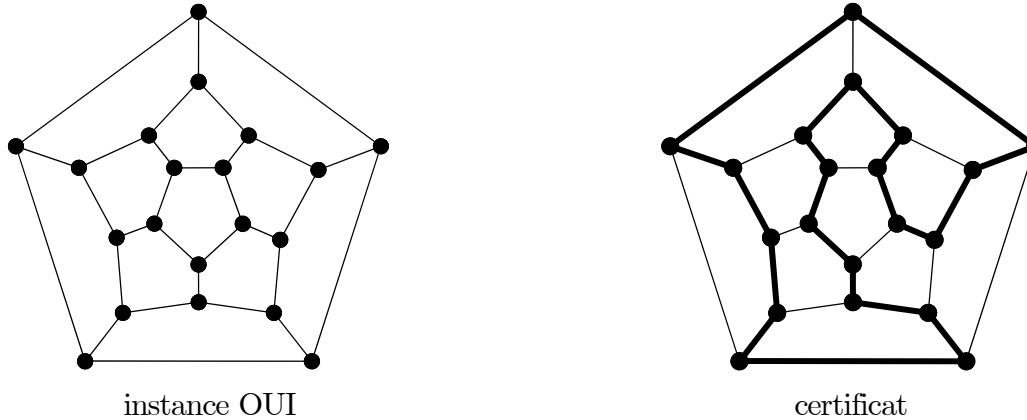


Fig.1.

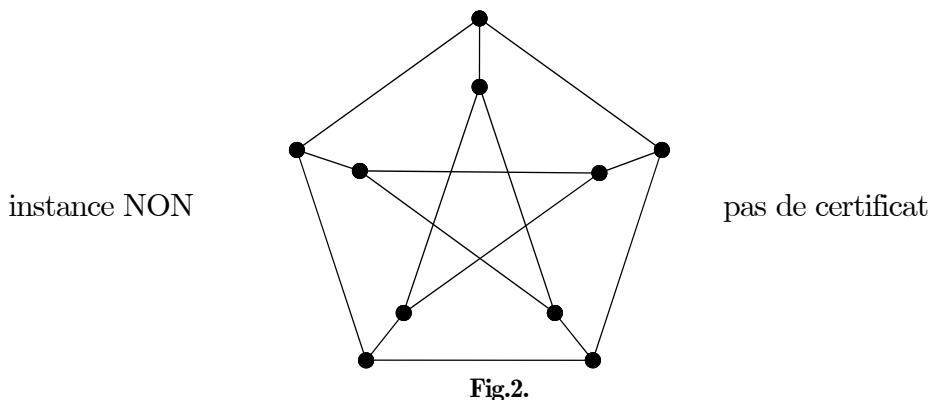


Fig.2.

La délivrance d'un certificat pour le OUI est évidente : il suffit de montrer une tournée. En revanche, on ne sait pas délivrer de certificat pour le NON. Certes, lorsque le graphe est petit, il est possible de trouver une preuve courte, mais personne ne sait démontrer qu'en augmentant la taille du problème, celle des preuves de la non-existence d'un tour ne va pas augmenter exponentiellement.

Il est probable que votre jeu préféré soit un problème **NP-complet** , comme c'est le cas du SOKOBAN (caisses), SOLITAIRE ou DEMINEUR.

Il faut remarquer que, a priori, les problèmes d'optimisation combinatoire ne font pas partie de problèmes de décision OUI-NON et pour cette raison ils n'appartiennent pas à la classe **NP** . Étant donné un problème d'optimisation combinatoire “trouver x_0 tel que $f(x_0) = \min_{x \in X} f(x)$ ” et un nombre k , on définit “*le problème de reconnaissance associé*” :

“existe-t-il x_0 tel que $f(x_0) \leq k$?”.

Un problème d'optimisation combinatoire est **NP-dur** si
le problème de reconnaissance associé est **NP-complet** .

Les problèmes **NP-dur** sont donc des problèmes au moins aussi difficiles que les problèmes **NP-complets** (l'appartenance à **NP** n'est pas exigée). Ainsi le problème de voyageur de commerce, consistant à déterminer un cycle hamiltonien le plus court est un exemple d'un problème **NP-dur** .

Théorèmes de bonne caractérisation.

La considération simultanée des classes **NP** et **co-NP** permet de mettre en évidence une classe de théorèmes dite de bonne caractérisation. On dit d'un théorème qu'il est une *bonne caractérisation* lorsqu'il affirme l'équivalence d'une propriété avec une propriété **co-NP** , c'est-à-dire que le problème considéré appartient à **$\text{NP} \cap \text{co-NP}$** .

Prenons par exemple une caractérisation donnée par le *théorème de Wilson* :

Un entier strictement positif p est un nombre premier si et seulement s'il divise $(p-1)!+1$, c'est-à-dire si et seulement si $(p-1)! \equiv -1 \pmod{p}$

Cette caractérisation n'est d'aucune utilité en pratique pour écrire un test informatique de primalité. Voici un exemple d'une bonne caractérisation :

THEOREME : *Un graphe est biparti si et seulement si il ne contient pas de cycle de longueur impaire.*

Pour un graphe avec n sommet il existe 2^{n-1} bipartitions et autant de sous ensemble de cardinalité impaire – à partir d'une certaine taille une énumération explicite est intractable. Mais vérifier qu'un cycle donné est impair ou que chaque partie d'une bipartition est stable nécessite moins de $O(n^2)$ opérations. Ce théorème affirme donc l'équivalence d'une propriété \mathcal{NP} avec une propriété $\text{co-}\mathcal{NP}$. Par ailleurs ce théorème indique un algorithme pour obtenir cette bipartition : il suffit de marquer alternativement 1, 2, 1, 2, etc. les sommets du graphe sans circuit de longueur impaire pendant une exploration en largeur. La 2-coloration d'un graphe est donc dans \mathcal{P} .

Le théorème de König-Hall est une bonne caractérisation car l'existence d'un couplage est \mathcal{NP} et la négation de la propriété annoncée par théorème, c'est-à-dire

“ il existe $A \subseteq X$, tel que $|N(A)| < |A|$ ”

est clairement \mathcal{NP} .

Explication didactique: il existe beaucoup de couplages (un nombre exponentiel par rapport au nombre de sommets), et il y a 2^{n-1} sous-ensemble de X – à partir d'une certaine taille une énumération explicite est intractable (pour les deux cas). Mais vérifier qu'un ensemble d'arêtes est un couplage ou qu'un ensemble de sommets est un certificat König-Hall est très simple.

Nous avons vu que la méthode hongroise permet de trouver en temps $O(nm)$ le couplage de taille maximum dans un graphe biparti – le problème du couplage maximum est dans \mathcal{P} .

On soupçonne que tous les problèmes bien caractérisés sont faciles mais la question
“ $\mathcal{NP} \cap \text{co-}\mathcal{NP} = \mathcal{P}$?” reste ouverte.

L'existence de chaîne hamiltonienne est \mathcal{NP} mais, aujourd'hui nous ne connaissons aucun certificat de l'absence de chaîne hamiltonienne dans un graphe arbitraire. Cela veut dire qu'il n'y a pas de bonne caractérisation des graphes hamiltoniens.

Une matrice à éléments dans $\{-1, 0, 1\}$ est dite *totalelement unimodulaire* si le déterminant de toute sous-matrice carrée est dans $\{-1, 0, 1\}$. La caractérisation suivante:

THEOREME : *M est totalement unimodulaire si et seulement si le déterminant de toute sous-matrice carrée n'est pas dans $\{-2, 2\}$.*

n'est pas une bonne caractérisation car ce théorème affirme l'équivalence de deux propriétés $\text{co-}\mathcal{NP}$ (il existe des algorithmes efficaces pour calculer le déterminant d'une matrice carrée).

Cependant il existe une bonne caractérisation d'une classe particulière de matrices totalement unimodulaires :

THEOREME : *La matrice d'incidence d'un graphe (non orienté) est totalement unimodulaire si et seulement si le graphe est biparti.* ■

PSPACE.

Les concepts précédents prennent uniquement en considération le temps de calcul c'est-à-dire le nombre d'opérations élémentaires effectuées par la machine de Turing. Mais ces opérations consistent en lecture-écriture dans les cases de la bande de cette machine alors le calcul nécessite aussi un certain nombre de ces casses. On dit qu'un problème appartient à la classe PSPACE si la machine de Turing déterministe peut le résoudre en utilisant moins de cn^k cases de sa bande, où c et k sont des constantes. Il est évident que chaque problème qui peut être résolu en temps polynomial nécessite aussi un espace polynomial mais on conjecture qu'il existe des problèmes pouvant être résolu en espace polynomial mais pas en temps polynomial. En utilisant la réduction polynomiale on peut définir les problèmes PSPACE-complets. Un problème \mathcal{D} est PSPACE-complet s'il est PSPACE et si tout problème PSPACE peut être réduit polynomialement à \mathcal{D} . Par exemple, beaucoup de problèmes de décision de type : “Pour un jeu donné, la position initiale d'un joueur assure-t-elle le gain ?” sont PSPACE-complets.

Complexité des problèmes d'énumération.

Les classes \mathcal{P} , \mathcal{NP} et $co\text{-}\mathcal{NP}$ sont définies pour les problèmes de décision. En général, nous pouvons distinguer les problèmes de décision, les problèmes de recherche et les problèmes d'énumération. Par exemple, nous pouvons citer trois problèmes différents concernant le couplage parfait : étant donné un graphe

- i. “Admet-il un couplage parfait ?”;
- ii. “Trouver un couplage parfait”;
- iii. “Combien de couplages parfaits différents admet-il?”

Il faut souligner que résoudre un problème d'énumération signifie trouver un nombre et non de faire une liste de toutes les solutions. On note $\#\mathcal{P}$ (number \mathcal{P}) la classe des problèmes d'énumération pour lesquels on peut donner la réponse en temps polynomial. Remarquons que même si la valeur retrouvée est exponentielle, elle peut être notée en utilisant un nombre polynomiale de chiffres dans le système binaire, alors ce n'est pas la “taille” de la réponse qui détermine la position du problème. On nous demande seulement de trouver ce nombre en temps polynomial.

Un problème d'énumération \mathcal{D} est $\#\mathcal{P}$ -complet s'il est $\#\mathcal{P}$ et si tout problème $\#\mathcal{P}$ peut être réduit polynomialement à \mathcal{D} .

On démontre par exemple, que le problème “Combien de couplages parfaits différents admet un graphe biparti donné ?” est $\#\mathcal{P}$ -complet. On pense que certains problèmes d'énumération resteront difficiles (intractables) même si $\mathcal{P}=\mathcal{NP}$.

Index

A

- algorithme*
 - de Bellman et Ford* 42
 - de Dijkstra* 43
 - de Kruskal* 23
 - de Prim* 24
 - glouton* 16, 23
 - arbre* 17
 - de poids minimum* 22
 - du graphe* 21
- arêtes* 9
- parallèles* 9

B

- bloc* 25
- bonne caractérisation* 109

C

- cactus* 25
- capacité de Shannon* 105
- chaîne* 12
 - alternée* 26
 - élémentaire* 12
 - fermée* 12
 - hamiltonien* 12
 - simple* 12
- chemin* 35
 - élémentaire* 35
 - fermé* 35
 - le plus court* 41
 - simple* 35
- co-arbre* 21
- cocycle* 16
 - du graphe* 16
- coloration* 14
- complexité* 106
- composante connexe* 16
- composante fortement connexe* 38
 - initiale* 38
 - terminale* 38
- condensation* 39
- connectivité* 25, 101
- corde* 25
- couplage* 12
 - maximum* 26
 - parfait* 12
- cycle* 12

- eulérien* 13
- hamiltonien* 12

D

- degré*
 - entrant* 32
 - sortant* 32
- distance* 41
 - la plus courte* 41

E

- ensemble d'articulation* 25
- exploration* 17
 - en largeur* 17
 - en profondeur* 17

G

- graphe* 9
- biparti* 11
 - biparti complet* 11
 - complémentaire* 11
 - complet* 9
 - connexe* 16
 - de permutation* 15
 - de Petersen* 12
 - dual* 13
 - eulérien* 13
 - hamiltonien* 12
 - h-connexe* 25
 - non orienté* 9
 - partiel* 11
 - planaire* 13
 - simple* 9
 - sous-graphe engendré* 11
 - subdivision du* 13

I

- isthme* 25

L

- listes d'adjacence chaînées* 34

M

- matrice d'adjacence* 9
- matrice d'incidence* 9
- matrice totalement unimodulaire* 32
- méthode hongroise* 28

N

- nombre*
 - chromatique* 14
 - de stabilité* 11
 - de transversalité* 27

P

- partition en niveaux* 37
- point d'articulation* 25
- prédecesseur* 32
- principe des cages à pigeons* 10

R

- Recherche Opérationnelle* 1
- recouvrement minimum* 27
- représentations*
 - par listes chaînées* 33
 - tabulaires* 33

S

- sommet*
 - degré d'un* 9
 - isolé* 9
 - pendant* 9
- sommets* 9
- subdivision du graphe* 13
- successeur* 32

T

- table d'adjacence* 34
- Théorème*
 - d'Euler* 35
 - de bonne caractérisation* 110
 - de König* 28
 - de König-Hall* 30
 - du point fixe* 103
- T-joint* 13
- transversal* 27
- tri topologique* 38
- triangulation conforme* 103

V

- voisins* 9

Table des matières

AVANT PROPOS	1
1^{ERE} PARTIE : GRAPHES ET APPLICATIONS.....	7
1. Le concept du graphe.....	9
2. Coloration	14
3. Connexité des graphes (non orientés)	16
4. Explorations des graphes	17
4.1. Exploration en largeur :	17
4.2. Exploration en profondeur.....	19
5. Arbres.....	20
5.1. L'arbre de poids minimum d'un graphe connexe	22
6. Connectivité.....	25
6.1. Graphes h -connexes.....	25
6.2. Points d'articulation et blocs.....	25
7. Couplages.....	26
7.1. Le problème du recouvrement minimum	27
7.2. Couplages dans un graphe biparti.....	28
7.3. Application à la chimie.	30
8. Orientation d'un graphe.....	32
9. Représentations des graphes en machine.....	33
10. Connexité dans des graphes orientés	34
10.1. Graphes sans circuit.....	37
10.2. Composantes fortement connexes.....	38
11. Le plus court chemin.....	41
11.1. Le plus court chemin dans des graphes sans circuit.....	42
11.2. Le plus court chemin dans des graphes comportant des circuits.....	43
11.3. Conclusions et remarques.....	43
12. Ordonnancements.....	46
2^{EME} PARTIE : PROGRAMMATION LINEAIRE.....	59
1. Modéliser par un programme linéaire	61
2. Recherche d'une solution optimale.....	65
3. La dualité	76
4. Programmation linéaire paramétrique.....	85
5. Programmation linéaire et ANALYSE des DONNEES.	89
6. Programmation linéaire et JEUX de STRATEGIES.	91
Liste des symboles :	100
Bibliographie.....	100
Compléments.....	101
1° Point Fixe.....	101
2° Capacité de Shannon.....	103
3° Introduction à la complexité.....	104
Index.....	110

