Master d'Informatique - Spécialité Androide
Module MAOA

Recherche Opérationnelle et Optimisation Combinatoire

Partie A - PLNE compact et solveurs

Pierre Fouilhoux

Sorbonne Université

2019-2020

Programme linéaire en nombres entiers (PLNE)

Ce cas étant largement le plus étudié, on l'appelle parfois même simplement *Programme entier (ou mixte)* (Integer Program ou Mixed Integer Program) (MIP).

$$\begin{aligned} &\text{Maximiser } c_1^T x_1 + c_2^T x_2 \\ &\quad \text{sous les contraintes} \\ &A_1 x_1 + A_2 x_2 \leq b \\ &x_1 \in \boldsymbol{R}^{n_1} \\ &x_2 \in \mathbb{Z}^{n_2}. \end{aligned}$$

où c_1, c_2 sont des vecteurs et A_1 et A_2 des matrices
avec x_1 partie continue de la solution
et x_2 partie entière de la solution.

Les contraintes $x_2 \in \mathbb{Z}^{n_2}$ sont les *contraintes d'intégrité* (ou d'entiéreté en Belgique ou d'intégralité au Québec), appelées integrity or integrality constraint en anglais.

Complexité

Un problème linéaire continu peut être résolu en temps polynomial par la **méthode des ellipsoïdes** (Khachiyan 1979).

Il existe des algorithmes polynomiaux efficaces pour résoudre un programme linéaire : les algorithmes dits de **points intérieurs** initiés par Karmarkar (1984).

Néanmoins **l'algorithme du simplexe** (Dantzig 1947) est le plus célèbre (et le plus efficace dans le cas général) des algorithmes de résolution, bien qu'il ne soit pas polynomial !
L'algorithme du simplexe repose sur le fait qu'une solution optimale d'un programme linéaire peut être prise parmi les sommets du polyèdre de $\boldsymbol{R}^n$ déterminé par $Ax \leq b$: on ramène ainsi un problème d'optimisation continue à un problème combinatoire ! ! !

En revanche, la PLNE est un problème NP-difficile. Il est facile de montrer que la PLNE est un problème NP-difficile car de nombreux problèmes NP-difficiles peuvent être exprimés comme des PLNE.

Solveurs PL/PLNE

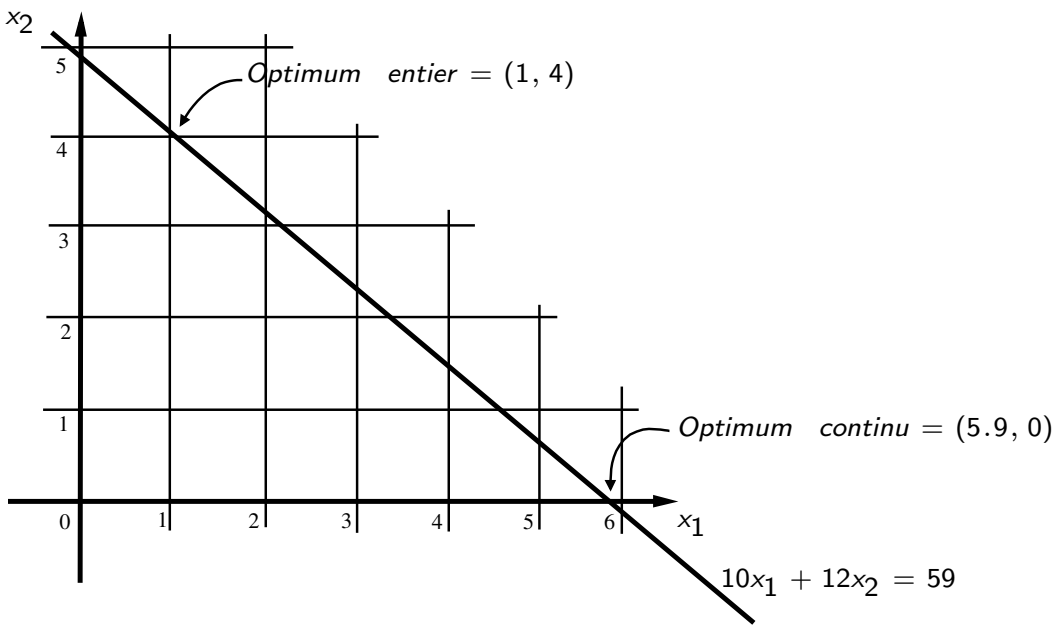
Il existe de nombreux solveurs de PL : des solveurs commerciaux Cplex (IBM), Xpress, Gurobi, et même Matlab ou Excel... ; des solveurs académiques Lp de COIN-OR, Soplex de la ZIB ; et des solveurs libres comme Glpk (gnu). Les meilleurs d'entre eux peuvent résoudre des PL jusqu'à 200000 variables et 200000 contraintes en quelques secondes.

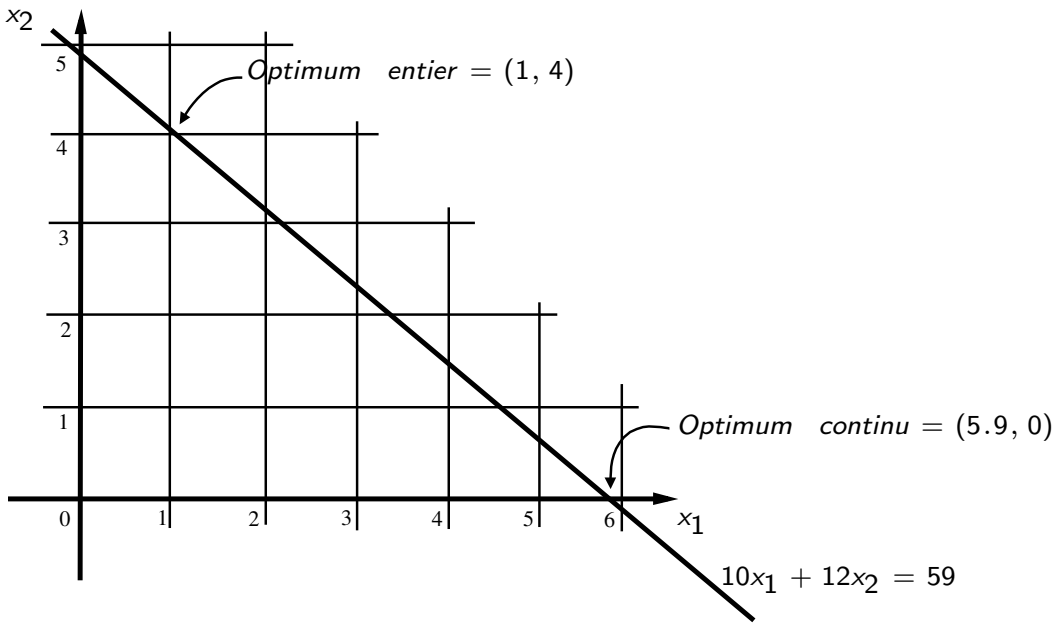
En revanche, les solveurs entiers performants sont beaucoup moins performants : ils sont en général liés aux solveurs PL : Glpk par exemple ne dépassent pas quelques 100 aine de variables et contraintes ; les solveurs commerciaux Cplex ou Gurobi sont les plus performants (Xpress est un peu en-dessous) pouvant réussir parfois quelques milliers de variables/contraintes ; un solveur "universitaire" les rattrape : SCIP de la ZIB. Un des objectifs de ce cours est de comprendre comment et dans quels cas ces solveurs atteignent de telles capacités.

La première remarque qui peut sauter aux yeux est d'imaginer que résoudre un PLNE revient à "arrondir" la solution de sa relaxation continue. L'exemple suivant témoigne de l'insuffisance de cette remarque :
Prenons un PLNE à deux variables et une seule contrainte, ce qui constitue le cas le plus simple que l'on puisse imaginer.

$$\begin{aligned} &\text{Maximiser } 10x_1 + 11x_2 \\ &10x_1 + 12x_2 \leq 59 \\ &x_1 \text{ et } x_2 \geq 0 \\ &x_1, x_2 \text{ entiers.} \end{aligned}$$

En dessinant le domaine de définition, on obtient la figure suivante :





On remarque alors que l'optimum de la relaxation continue a une valeur objective de 59 et celui de l'optimum entier est de 54 seulement. Mais surtout, on peut noter l'écart complet de structure et de position des deux points optimum (qui sont ici chacun solution optimale unique).

Problème d'optimisation combinatoire

Déterminer un plus grand (petit) élément dans un ensemble fini valué.

Etant donnés :
- un ensemble fini d'éléments $E = \{e_1, ..., e_n\}$
- un vecteur-poids $c = (c(e_1), ..., c(e_n))$ associé à E
- une famille $\mathcal{F}$ de sous-ensembles de E, les **solutions**.

Un problème d'optimisation ("linéaire") consiste à trouver
un ensemble $F \in \mathcal{F}$
de poids $c(F) = \sum_{e \in F} c(e)$ maximum (ou minimum), *i.e.*

$$\max \text{ ou } \min \{c(F) \mid F \in \mathcal{F}\}.$$

Formulation algébrique "naïve"

Associer une variable à chaque solution :
$t_F = 1$ si la solution $F \in \mathcal{F}$ est prise et 0 sinon

$$\text{Max} \sum_{F \in \mathcal{F}} c(F) t_F$$
$$\sum_{F \in \mathcal{F}} t_F \leq 1$$
$$t_F \in \{0,1\} \text{ pour tout } F \in \mathcal{F}.$$

Cette formulation prouve que tout problème d'Optimisation combinatoire peut s'écrire sous forme d'un PLNE !
Mais peut-on résoudre une telle formulation ayant nombre de variables égales au nombre de solutions ?

Formulation algébrique "naturelle"

Associer une variable 0-1 à chaque élément E :
$x_e = 1$ si l'élement e est pris dans la solution et 0 sinon

$$\begin{aligned} \text{Max} \sum_{e \in E} c(e) x_e \\ Ax \leq B \\ x_e \in \{0,1\} \text{ pour tout } e \in E. \end{aligned}$$

Cette formulation PLNE demande :
- de pourvoir définir par des inégalités $Ax \leq B$ le fait que les solutions décrites par la variable doivent être dans l'ensemble $\mathcal{F}$.
- de pouvoir résoudre la formulation ainsi obtenue.

Ce n'est pas toujours possible...

Exemple : le problème du sac-à-dos (knapsack) 0/1

Données : n objets $i = 1, ..., n$, de bénéfice c_i et de poids a_i,
Objectif : Ranger les objets dans un "sac" de poids maximum b avec un bénéfice maximal.

Le problème de *sac-à-dos* (*knapsack*) consiste à choisir les objets à prendre parmi les n objets de manière à avoir un bénéfice maximal et respecter la contrainte du poids à ne pas dépasser.

Ce problème se rencontre bien entendu dès que l'on part en randonnée en voulant emmener le plus possible d'objets utiles (nourriture, boissons,...). Mais ce problème est plus fréquemment utilisé pour remplir les camions de transport, les avions ou bateaux de fret et même pour gérer la mémoire d'un microprocesseur.

La formulation PLNE du problème de sac-à-dos est très simple. On utilise pour chaque objet $i \in \{1, ..., n\}$, une variable entière x_i correspondant au nombre de fois où l'objet i est choisi.

$$\begin{aligned} \text{Max} \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b, \\ x_i \in \{0, 1\}, \text{ pour } i = 1, ..., n. \end{aligned}$$

L'unique contrainte est dite *contrainte de sac-à-dos*.
Elle est l'unique contrainte de ce problème qui est pourtant NP-difficile !

Recouvrement, pavage et partition

Soit $E = \{1, ..., n\}$ un ensemble fini d'éléments.
Soit $E_1, ..., E_m$ des sous-ensembles de E.
A chaque ensemble E_j on associe un poids c_j, $j = 1, ..., m$.

Une famille $F \subseteq \{E_1, ..., E_m\}$ est dite

- un **recouvrement** de E si $\cup_{E_j \in F} E_j = E$, pour tout $j \in \{1, ..., m\}$,

- un **pavage** de E si $E_j \cap E_k = \emptyset$, pour tout $j \neq k \in \{1, ..., m\}$,

- une **partition** de E si F est à la fois un recouvrement et un pavage.

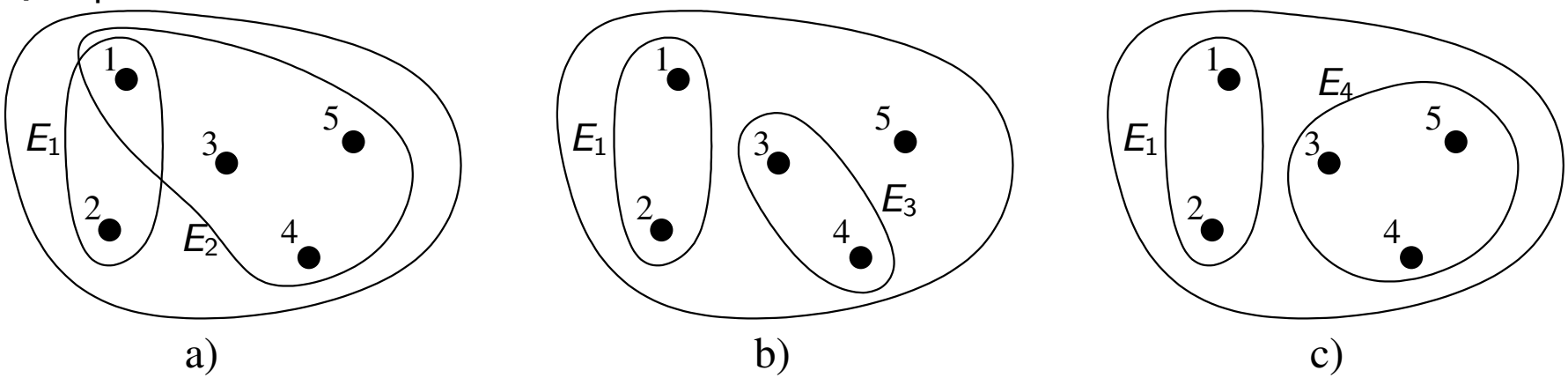
On peut interpréter les contraintes du problème de recouvrement (resp. pavage, partition) comme le fait qu'un élément de E doit être pris au moins une fois (resp. au plus une fois, exactement une fois).

Le **problème de recouvrement** (resp. *pavage*, *partition*) consiste à déterminer un recouvrement (resp. pavage, partition) dont la somme des poids des ensembles qui le forment est de poids minimum (resp. maximum, minimum/maximum).

Recouvrement, pavage et partition

Exemple
Données :
- 5 éléments $E = \{1, 2, 3, 4, 5\}$
- et 4 sous-ensembles $E_1 = \{1, 2\}$, $E_2 = \{1, 3, 4, 5\}$, $E_3 = \{3, 4\}$ et $E_4 = \{3, 4, 5\}$.

Quelques solutions :



a) $\{E_1, E_2\}$ est un recouvrement
b) $\{E_1, E_3\}$ est un pavage
c) $\{E_1, E_4\}$ est une partition.

Recouvrement, pavage et partition

Modélisation :
variables binaires $x_1, ..., x_m$ associées aux sous-ensembles $E_1, ..., E_m$.
Pour cela, on considère A la matrice en 0-1
dont les lignes correspondent aux éléments $1, ..., n$
et les colonnes aux sous-ensembles $E_1, ..., E_m$
et dont les coefficients sont $A_{ij} = 1$ si $i \in E_j$ et 0 sinon.

Recouvrement	Pavage	Partition
$\text{Min} \sum_{i=1}^{m} c_j x_j$	$\text{Max} \sum_{i=1}^{m} c_j x_j$	$\text{Max (ou Min)} \sum_{i=1}^{m} c_j x_j$
$Ax \geq \mathbb{1}$	$Ax \leq \mathbb{1}$	$Ax = \mathbb{1}$
$x \in \{0,1\}^m$	$x \in \{0,1\}^m$	$x \in \{0,1\}^m$

où $\mathbb{1}$ est un vecteur dont chaque composante est 1.

Recouvrement, pavage et partition

Dans l'exemple avec 4 sous-ensembles
$E_1 = \{1, 2\}$, $E_2 = \{1, 3, 4, 5\}$, $E_3 = \{3, 4\}$ et $E_4 = \{3, 4, 5\}$.

La matrice A est

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

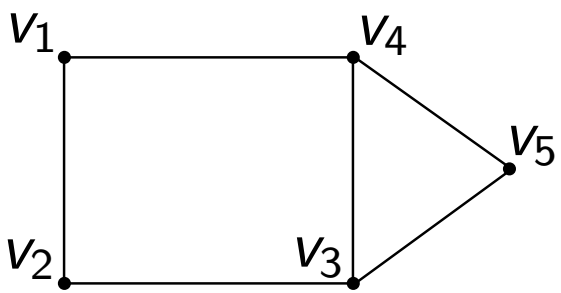
Pour le cas recouvrement, la première inégalité du PLNE est alors

$$x_1 + x_2 \geq 1$$

correspondant au fait qu'il faut choisir E_1 et/ou E_2 pour couvrir l'élément 1.

Exemple : le problème du stable

- $G = (V, E)$ un graphe non-orienté et $c(v)$ poids d'un sommet.
- Un stable de G est un sous-ensemble S de sommets de V tel qu'il n'existe aucune arête de E entre 2 sommets de S.
- Problème du stable de poids maximum : déterminer un stable S de G tel que $c(S) = \sum_{v \in S} c(v)$ soit maximum.



Problème NP-difficile !

Formulation compacte pour le problème du stable

$x(v) = 1$ si le sommet v est pris dans la solution et 0 sinon

$$\begin{aligned} \text{Max} \sum_{u \in V} c(u)x(u) & \\ x(u) + x(v) \leq 1, & \quad \text{pour tout } uv \in E, \\ x(u) \in \{0,1\}, & \quad \text{pour tout } u \in V. \end{aligned}$$

L'inégalité $x(u) + x(v) \leq 1$ est appelée inégalité aux arêtes.
Formulation compacte i.e. possède un nombre polynomial de contraintes et de variables.

Formulation PLNE difficile à résoudre par Branch-and-Bound.

Aperçu de la suite du cours

Comment "renforcer" une formulation PLNE

Une **clique** est un ensemble de sommets induisant un sous-graphe complet.
Or comme il y a au plus 1 sommet dans une clique K dans un stable, l'inégalité

$$\sum_{u \in K} x(u) \leq 1 \qquad \text{pour toute clique } K$$

est vérifiée pour tout stable.

Formulation non-compacte pour le problème du stable

On peut alors proposer une autre formulation non-compacte pour le problème du stable

$$\begin{aligned} \text{Max} \sum_{u \in V} & c(u)x(u) \\ \sum_{u \in K} & x(u) \leq 1 \qquad \text{pour toute clique } K, \\ & x(u) \in \{0,1\} \qquad \text{pour tout } u \in V. \end{aligned}$$

Il y a un nombre exponentiel de cliques dans un graphe...
donc cette formulation a un nombre exponentiel de contraintes...

Est-ce que cette formulation est "meilleure" ?

Voir cours "Branchement".

Parmi toutes ces inégalités, sont-elles toutes utiles ?

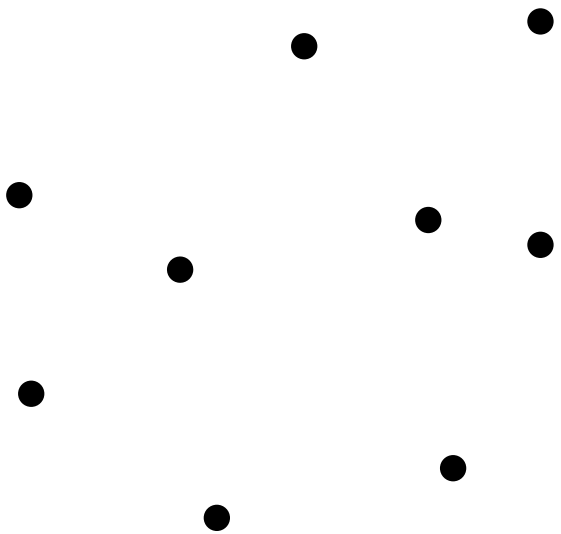
Voir cours "Approches polyédrales"

Comment résoudre une telle formulation ?

Voir cours "Algorithmes de coupes"

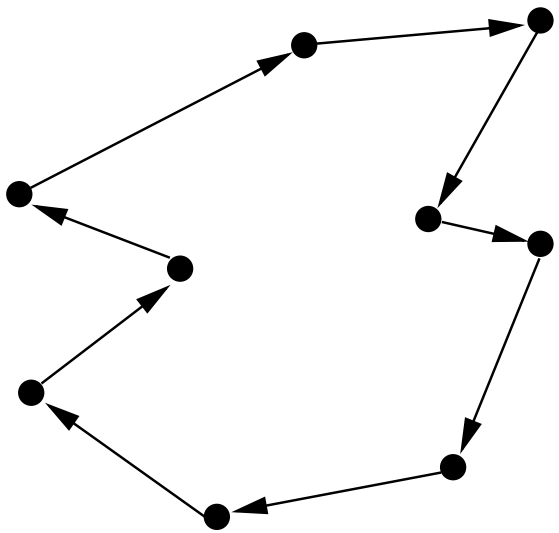
Modélisation du problème du voyageur de commerce (TSP)

Données : n villes avec c_{ij} coût de transport de i à j (asymétrique).
Objectif : Déterminer un circuit passant par toutes les villes de plus petit coût.



Modélisation du problème du voyageur de commerce (TSP)

Données : n villes avec c_{ij} coût de transport de i à j (asymétrique).
Objectif : Déterminer un circuit passant par toutes les villes de plus petit coût.



Formulation en variables naturelles

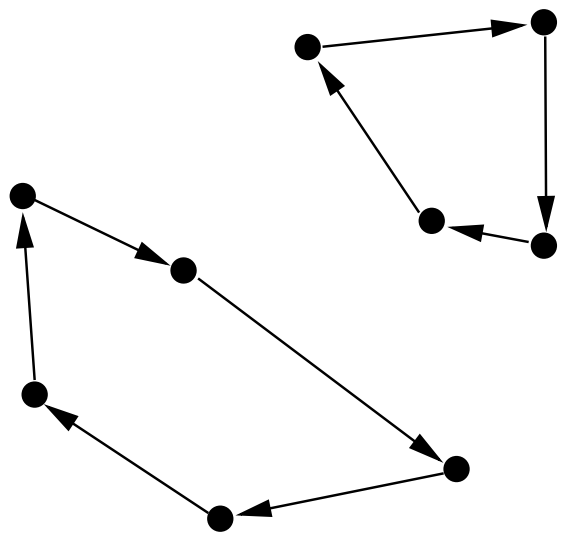
$x_{ij} = 1$ si l'arc (i, j) est dans le circuit et 0 sinon.

$$\text{Min} \sum_{i,j} c_{ij} x_{ij}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V,$$

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V,$$

$$x_{ij} \in \boldsymbol{N} \qquad \forall i \in V, j \in V \setminus \{i\}).$$



On peut ajouter également les inégalités $x_{ij} + x_{ji} \leq 1$.

Formulation en variables naturelles

- Relaxation linéaire de ce PLNE entière dans de nombreux cas de graphes (la matrice est totalement unimodulaire pour les graphes bipartis).
- Malheureusement, la solutions peut être faite de plusieurs cycles orientées (appelées sous-tours).

Il faut ajouter des inégalités pour "briser les sous-tours".

"Elimination des sous-tours" par la formulation MTZ

Formulation de Miller-Tucker-Zemlin (MTZ).
Ajout des variables réelles u_i, $i = 1, ..., n$, associées aux villes
Ajout des contraintes :

$$\begin{aligned} u_1 &= 1, \\ 2 \leq u_i \leq n & \qquad \forall i \in V \setminus \{1\}, \\ u_i - u_j + 1 \leq n(1 - x_{ij}) & \qquad \forall i \in V \setminus \{1\}, j \in V \setminus \{1, i\}. \end{aligned}$$

On apelle ces dernière inégalités *inégalités MTZ*.

"Elimination des sous-tours" par la formulation MTZ

Elles permettent d'éliminer les sous-tours :

- pour tout arc (i, j) où $x_{ij} = 1$, elles forcent $u_j \geq u_i + 1$, (dans le cas où $x_{ij} = 0$, elles restent valides car indiquent alors $u_i - u_j \leq n - 1$ qui est toujours vrai).
- si une solution entière de la formulation contient plus d'un sous-tour, alors l'un deux au moins ne contient pas le sommet 1 et, sur ce sous-tour, les variables u_i s'incrémentent à l'infini.

Remarque : dans le cas d'une solution réalisable pour la formulation, les variables u_i, $i = 1, ..., n$ indiquent la position de la ville i dans le tour.

"Elimination des sous-tours" par la formulation MTZ

Cette formulation possède n variables (continues) de plus, mais elle a l'énorme avantage d'être compacte !
On peut donc directement utiliser une procédure de branchement et séparation (Branch& Bound) et donc un solveur entier.

En revanche, c'est une **Très Très mauvaise formulation**
car elle possède une mauvaise relaxation linéaire (voir chapitre Branchement).

"Elimination des sous-tours" par des flots

Formulation utilisant des flots.
Ajout des variables de flots réelles z_{ij}, pour tout $i \in V$, j$\in V \setminus \{1, i\}$.
Ajout des contraintes :

$$\sum_{j \in V \setminus \{1\}} z_{1j} = |V| - 1,$$

$$\sum_{j \in V \setminus \{1,i\}} z_{ij} + 1 = \sum_{j \in V \setminus \{i\}} z_{ji} \quad \forall i \in V \setminus \{1\},$$

$$z_{ij} + z_{ji} \leq (|V| - 1)(x_{ij} + x_{ji}) \quad \forall i \in V, j \in V \setminus \{1, i\}$$

$$z_{ij} \geq 0 \quad \forall i \in V, j \in V \setminus \{1, i\}.$$

On peut remarquer que les variables z porte un flot de valeur $|V| - 1$ sortant du sommet 1, mais qu'il n'y pas d'inégalité de flot entrant au sommet 1.

"Elimination des sous-tours" par des flots

Ces inégalités permettent d'éliminer les sous-tours.

Preuve : En effet, pour x entier, le PLNE devient un PL de flot dans un graphe limité aux arcs sélectionnés par x. Supposons que ce graphe possède un sous-tour qui contient donc au moins 2 sommets. Le sommet 1 est dans un circuit ne passant pas par tous les sommets. Notons C le circuit passant par 1 qui contient donc au plus $|V| - 2$ sommets. En sortant de 1, le flot vaut $|V| - 1$ et à chaque sommet successif de C, le flot diminue d'une unité. Au dernier sommet i de C avant 1, le flot entrant est donc strictement supérieur à 3 et la somme des arcs sortant de i est strictement supérieure à 2 : ce qui est impossible car le seul arc sortant de 1 retenu dans le graphe est $(i, 1)$ qui n'est pas associé à une variable z : donc tous les z sortant de i sont nuls. Contradiction.

Cette formulation possèdes n^2 variables (continues) de plus et elle est encore compacte ! On peut l'utiliserr directement dans un solveur entier.

Sa valeur de relaxation linéaire est bien meilleure que celle utilisant MTZ. En revanche, les contraintes liant z et x (contraintes de "big M") ne permettent pas d'avoir une très bonne relaxation.

"Elimination des sous-tours" par des flots désagrégés

Formulation utilisant des flots "désagrégés" (dite de Maculan).
Ajout de $|V|$ ensemble de variables de flots réelles z_{ij}^k, pour tout $i \in V$, $j \in V \setminus \{1, i\}$ et pour tout $k \in V \setminus \{1\}$.
Ajout des contraintes :

$$\sum_{j \in V \setminus \{1\}} z_{1j}^k = 1 \qquad \forall k \in V \setminus \{1\}$$

$$\sum_{j \in V \setminus \{1,i\}} z_{ij}^k = \sum_{j \in V \setminus \{i\}} z_{ji}^k \qquad \forall k \in V \setminus \{1\}, \forall i \in V \setminus \{1, k\},$$

$$\sum_{j \in V \setminus \{1,k\}} z_{kj}^k + 1 = \sum_{j \in V \setminus \{k\}} z_{jk}^k \qquad \forall k \in V \setminus \{1\},$$

$$z_{ij}^k + z_{ji}^k \leq x_{ij} + x_{ji} \qquad \forall i \in V, j \in V \setminus \{i, 1\}, \forall k \in V \setminus \{1\}$$

$$z_{ij}^k \geq 0 \qquad \forall i \in V, j \in V \setminus \{1, i\}, \forall k \in V \setminus \{1\}.$$

On peut remarquer que chaque ensemble de variables z^k porte un flot de valeur 1 sortant du sommet 1, mais qu'il n'y pas d'inégalité de flot entrant au sommet 1.

"Elimination des sous-tours" par des flots désagrégés

Ces inégalités permettent d'éliminer les sous-tours.

Preuve : En effet, pour x entier, le PLNE devient un PL de k flots indépendants circulant sur des graphes limités aux arcs où x est non nul. S'il existe un sommet k qui n'est pas dans le circuit issu du sommet 1, le flot z^k partant de 1 reste à la valeur 1 jusqu'au dernier sommet i de C avant 1. Le flot sortant de 1 doit être porté par un z^k mais ils sont tous nuls car le seul arc de x non nul est $(i, 1)$. Contradiction.

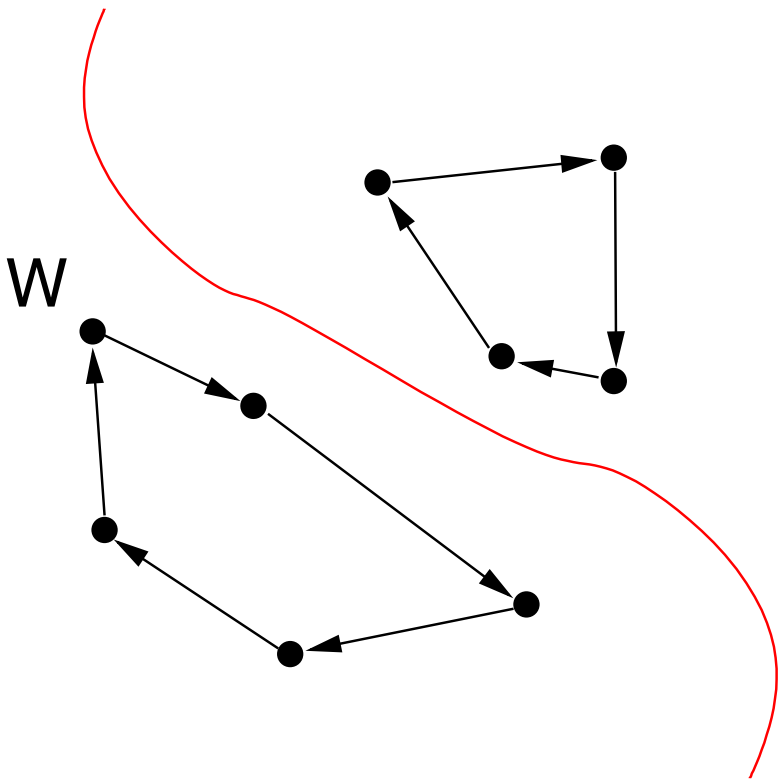
En fait la formulation par les flots précédentes peut être vues comme une agrégation des k flots z^k en un seul. On dit donc que cette formulation est "désagrégée" par rapport à la précécente. Son but est de faire disparaître les inégalités de "big M" reliant z et x.

Cette formulation a une très bonne valeur de relaxation. Toutefois elle possède n^3 variables (continues) supplémentaires ainsi qu'un nombre d'inégalité en n^2...

"Elimination des sous-tours" par la connexité

Théorème de Menger : un graphe est fortement connexe si et seulement si toute coupe du graphe contient au moins un arc.

$$\sum_{e \in \delta^+(W)} x(e) \geq 1, \text{ pour tout } W \subsetneq V \text{ et } W \neq \emptyset,$$

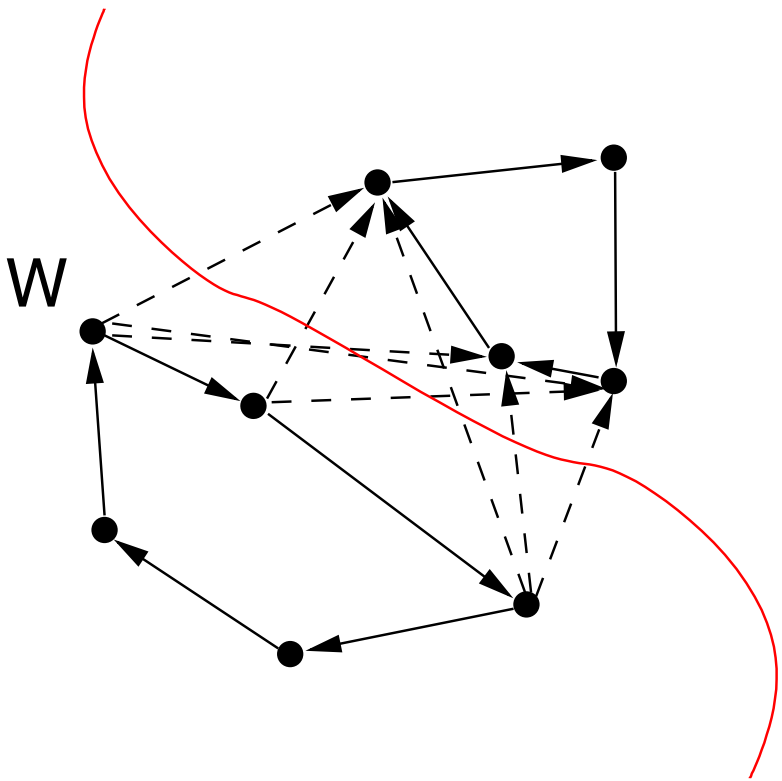


où $\delta + (W) = \{(i,j) \mid i \in W \text{ et } j \notin W\}$ est la coupe sortante de W.

"Elimination des sous-tours" par la connexité

Théorème de Menger : un graphe est fortement connexe si et seulement si toute coupe du graphe contient au moins un arc.

$$\sum_{e \in \delta^+(W)} x(e) \geq 1, \text{ pour tout } W \subsetneq V \text{ et } W \neq \emptyset,$$



où $\delta + (W) = \{(i,j) \mid i \in W \text{ et } j \notin W\}$ est la coupe sortante de W.

Ce n'est plus une formulation compacte : elle possède un **nombre exponentiel de contraintes !**

Formulation du TSP symétrique

$$Min \sum_{e \in E} c(e)x(e)$$

$$\sum_{e \in \delta(v)} x(e) = 2, \text{ pour tout } u \in V,$$

$$\sum_{e \in \delta(W)} x(e) \geq 2, \text{ pour tout } W \subsetneq V \text{ et } W \neq \emptyset,$$

$$x(e) \in \{0, 1\}, \text{ pour tout } e \in E.$$

Formulation "vedette" du TSP symétrique.
Renforcée par d'autres inégalités, elle est la technique record.

Coloration de graphe

Une k-**coloration** des sommets d'un graphe $G = (V, E)$ est une fonction $r : V \to \{1, ..., k\}$ telle que $r(u) \neq r(v)$ pour tout couple de sommets adjacents u, v.

Le **problème de coloration** consiste à déterminer le plus petit k tel que G soit k-coloriable.

Première formulation

- on associe à chaque sommet u de V un vecteur binaire à K dimensions $x_u = (x_u^1, ..., x_u^K)$, où K est une borne supérieure sur la coloration de G (au maximum $K = |V|$).
- on ajoute une variable binaire w_l par couleur $l = 1, ..., K$ indiquant si cette couleur a été utilisée ou non.

Le problème est donc équivalent au programme

$$\text{Min} \sum_{l=1}^{K} w_l$$

$$\sum_{l=1}^{K} x_u^l = 1, \qquad \text{pour tout } u \in V, \tag{1}$$

$$x_u^l + x_v^l \leq w_l, \qquad \text{pour tout } e = uv \in E \text{ et } 1 \leq l \leq K, \tag{2}$$

$$x_u^l \in \{0, 1\}, \qquad \text{pour tout } u \in V \text{ et } 1 \leq l \leq K.$$

Cette formulation contient énormément de **symétrie** c'est-à-dire des solutions très "proches" de même coût.

Deuxième formulation

Remarque : une coloration est une partition des sommets en stables.

Soit $\mathcal{S}$ l'ensemble des stables non vides de G.
On associe à chaque stable $S \in \mathcal{S}$ une variable binaire t_S.
Le problème de coloration est alors équivalent au programme en nombres entiers suivant (Mehrotra et Trick 1995).

$$\text{Min} \sum_{S \in \mathcal{S}} t_S$$

$$\sum_{S \in \mathcal{S} \mid u \in S} t_S = 1, \qquad \text{pour tout } u \in V, \tag{3}$$

$$t_S \in \{0,1\}, \qquad \text{pour tout } S \in \mathcal{S}. \tag{4}$$

Ici, il n'y a pas de symétries artificielles dues à une numérotation des colorations.

Elle possède un nombre polynomial d'inégalités... mais un nombre exponentiel de variable. (Voir cour "Génération de colonnes")

En conclusion

Un problème d'optimisation combinatoire possède de nombreuses formulations :
- compacte
- avec un nombre exponentiel de contraintes
- avec un nombre exponentiel de variables
- avec un nombre exponentiel de variables et de contraintes !

Comment choisir une bonne formulation ?
Cela dépend du cadre algorithmique de résolution : il n'existe pas d'outils génériques "magiques" pour toutes ces formulations et tous les problèmes...

Nous verrons les principaux outils algorithmiques et les plus célèbres de ces problèmes.

Cette section donne des pistes d'idées pour modéliser certains liens logiques entre des variables ou des contraintes du problème.

• *Cas simples*
Soient a ,b et c des évênements correspondant aux variables de décisions binaires x_a, x_b et x_c.

- Si a et b ne peuvent pas se produire tous les deux : $x_a + x_b \leq 1$.
- Si au moins des évènement parmi a et b doit se produire : $x_a + x_b \geq 1$.
- Si a se produit alors b doit se produire : $x_a \leq x_b$.
- On peut noter que l'inégalité précédente modélise heureusement la contraposée de la proposition logique correspondante : si b ne se produit pas, a ne doit pas se produire.
- Si a se produit, alors b et/ou c doivent se produire : $x_a \leq x_b + x_c$.
- Si a ne se produit pas, alors une quantité y positive doit être nulle, sinon y est libre dans $\boldsymbol{R}$. On doit fixer une quantité M telle que y ne peut jamais être supérieure à M lorsqu'on atteint l'optimum du problème. Une telle constante M existe, car sinon le problème serait non borné.
  On peut écrire $y \leq Mx_a$ (contrainte dite de "big M").

• *Maximiser la valeur minimale d'un ensemble de fonctions linéaires*
Si on veut maximiser la plus petite valeur prise par un ensemble de fonctions linéaires $a^i x$, $i = 1, ...m$, il suffit d'ajouter une variable z et les contraintes $z \leq a^i x$, $i = 1, ...m$: la fonction objective devient alors Max z.

• Le "ou" numérique
On veut représenter une variable x devant prendre des valeurs soit 0, soit être plus grande que L où L et x sont bornées par une valeur M.
On ajoute une variable $y \in \{0, 1\}$ et on utilise les contraintes

$$x \geq Ly \qquad \text{et} \qquad x \leq My.$$

• *Satisfaire le plus possible d'inégalités*
On dispose d'un lot de n contraintes $a^1x \leq b^1, a^2x \leq b^2, ..., a^nx \leq b^n$ qui potentiellement forment un ensemble de solutions vides. On souhaite une solution qui satisfasse le plus possible de contraintes. Pour chacune des contraintes $a^ix \leq b^i$, $i = 1, ..., n$, on détermine une valeur M_i suffisamment grande pour que $a^ix \leq b^i + M_i$ soit satisfaite quelque soit x.
On pose alors $y_1, ..., y_n$ des variables binaires et ce cas de figure se modélise alors de la façon suivante.

$$\begin{array}{lcl} a^1x \leq b^1 & & a^1x \leq b^1 + M_1y_1 \\ a^2x \leq b^2 & & a^2x \leq b^2 + M_2y_2 \\ ... & \Rightarrow & ... \\ a^nx \leq b^n & & a^nx \leq b^n + M_ny_n \\ & & Min \ \sum_{i=1}^{n} y_i \end{array}$$

Remarquons que si une seule contrainte doit être satisfaite parmi deux (c'est-à-dire si $k = 1$ et $n = 2$), on peut utiliser une seule variable binaire y en posant $y_1 = y$ et $y_2 = 1 - y$.

• *Implication entre contraintes*
Soit $a^1x \leq b^1$ et $a^2x \leq b^2$ deux contraintes telles que si $a^1x < b^1$, alors $a^2x \leq b^2$ doit être satisfaite, mais que, par contre, si $a^1x \geq b^1$, alors $a^2x \leq b^2$ peut ou non être satisfaite.
On prend M tel que $-a^1x \leq -b^1 + M$ et $a^2x \leq b^2 + M$ soient vérifiées pour toute valeurs de x. On peut utiliser une variable de décision binaire y et écrire alors :

$$-a^1x \leq -b^1 + M(1-y) \quad (5)$$
$$a^2x \leq b^2 + My \quad (6)$$

En effet, si $a^1 < b$, alors la contrainte (5) implique que $y = 0$, et ainsi la contrainte (6) est équivalente à $a^2x \leq b^2$ qui doit donc être satisfaite. Pour le cas contraire (i.e. si $a^1x \geq b^1$), alors y peut prendre la valeur 0 ou 1, c'est-à-dire que $a^2x \leq b^2$ peut être satisfaite ou non.

Programme mathématique

Un *Programme Mathématique* (mathematical program), noté PM, est un problème d'optimisation sous contrainte $(\mathcal{P})$ qui peut s'écrire de la façon suivante :

$$\begin{array}{ll} \text{Maximiser } f(x) & \\ \quad \text{sous les contraintes} & \\ \quad g_i(x) \leq 0 & i = 1, ..., m \\ \quad x \in S. & \end{array}$$

où
• S est une partie de $\boldsymbol{R}^n$ et x est un vecteur appelé *variable*, ces n composantes sont dites les *inconnues* du problème,
• la fonction $f : S \to \boldsymbol{R}$ est appelée *fonction objective* ou *objectif* (objective function),
• les fonctions $g_i : S \to \boldsymbol{R}$, $i = 1, ..., m$, forment des inégalités qui sont appelées les *contraintes* (constraint) du problème.

On peut remarquer qu'un PM peut être une maximisation ou une minimisation (il suffit de poser la fonction $f' = -f$.

On appelle *inégalités* une contrainte $g_i(x) \leq 0$ ou $g_i(x) \geq 0$: en cas de présences des deux contraints $g_i(x) \leq 0$ et $g_i(x) \geq 0$, on parle alors d'égalité $g_i(x) = 0$.

Un vecteur $\bar{x}$ vérifiant les contraintes d'un PM est dit *solution* ou *solution réalisable* du PM. L'ensemble des solutions d'un PM forme un *domaine de définition*. Le domaine de définition d'un PM peut être : vide (dans ce cas, le problème n'admet pas de solutions), dans le cas contraire, le PM admet des solutions. Sous certaines conditions, il peut exister des solutons x^* dites *optimales*, c'est-à-dire qui maximisent la fonction $f(x)$ sur toutes les solutions du PM.

Plusieurs cas de PM sont à mettre en évidence :
- si l'ensemble S est continu, on parle de *programme mathématique continu* (continuous) ,
- si l'ensemble S est discret (c'est-à-dire isomorphe à $\boldsymbol{N}^n$), on parle de *programme mathématique discret* (discrete) que l'on notera ici (PMD) ; on le dit également *entier* (integer program) si $S \subset \boldsymbol{N}^n$ ou même *binaire* si $S \subset \{0,1\}^n$. En fait tout PMD peut se ramener au cas d'un programme entier, voir même d'un programme binaire.
- si certaines composantes du vecteur x solution prennent leurs valeurs dans un ensemble discret et les autres dans un ensemble continu, on le dit *programme mathématique mixte*.

Dans le cas des programmes entiers (donc discrets également), on peut noter alors un (PMD) de la façon suivante :

$$\begin{array}{ll} \text{Maximiser } f(x) & \\ \quad \text{sous les contraintes} & \\ \quad g_i(x) \leq 0 & i = 1, ..., m \\ \quad x \in \mathbb{Z}^n. & \end{array}$$

On désigne alors $x \in \mathbb{Z}^n$ comme étant la *contrainte d'intégrité* (ou d'entiéreté en Belgique ou d'intégralité au Québec) (integrity or integrality constraint).
On appelle *relaxation* le fait de "relâcher", c'est-à-dire supprimer une contrainte du problème. Ainsi, un programme relaxé désignera un programme où l'on aura supprimé une ou plusieurs contraintes.

On appelle *relaxation continue* le fait de "relâcher" les contraintes d'intégrité du problème. Par abus de langage, on appelle aussi souvent *relaxation continue* le fait de résoudre le programme que où l'on a relâché les contraintes d'intégrité (l'expression désigne même parfois la solution optimale obtenue).

Programme convexe

Soit un ensemble $S \subset \boldsymbol{R}^n$ convexe (un ensemble de points tels que tout segment entre deux points est entièrment dans l'ensemble convexe). Une fonction $f : S \to \boldsymbol{R}$ est dite *fonction convexe* si elle vérifie

$$\forall x_1, x_2 \in S, \forall \lambda \in [0,1], \quad f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2).$$

Une fonction est dite *concave* si la fonction $-f$ est convexe.
Considérons alors le (PM) suivant

$$\begin{array}{l} \text{Minimiser } f(x) \\ x \in S. \end{array}$$

Si la fonction objective f d'un (PMD) est convexe (ou concave) et que l'ensemble S est un convexe fermé non-discret, on parle de *programme convexe*. Pour un programme convexe, tout optimum local est global. Les cas non-convexe (ou non-concave pour une maximisation), n'ont pas toujours d'optimum global.

Pour le cas des programmes convexes, suivant les propriétés de la fonction f : (continuité, différentiabilité,...), il existe des algorithmes plus ou moins efficace pour déterminer le minimum d'une fonction (méthode de Newton, méthode de sous-gradient,...).

Programme convexe sous contraintes

Considérons à présent le (PM) suivant

$$\begin{array}{ll} \text{Minimiser } f(x) & \\ \quad \text{sous les contraintes} & \\ \quad g_i(x) \leq 0 & i = 1, ..., m \\ \quad x \in S. & \end{array}$$

Si f est convexe et S convexe non discret, on parle de *programme convexe avec contrainte*. On sait déterminer par les conditions de Karush-Kuhn-Tucker dans quel cas ce programme possède ou non un optimum. Il existe également des dérivés des méthodes citées précédemment pour résoudre le problème.

La résolution de ces programmes est souvent appelée *Programmation non-linéaire* et constitue un domaine de recherche à part entière. Il existe des solveurs continues efficaces capables de résoudre un grand nombre de cas de ces programmes. Citons par exemple le freeware SolvOpt
http ://www.kfunigraz.ac.at/imawww/kuntsevich/solvopt.

On appelle aujourd'hui *programmation non-linéaire discrète* l'étude générale des programmes convexes sont contraintes le plus souvent linéaire. Il s'agit d'un domaine de recherche récent et très riche (mais que nous n'aborderons pas ici).

Programme quadratique

Si la fonction f est quadratique et les fonctions g sont linéaires : on parle de *programme quadratique discret* (PQD). Si f est quadratique et convexe et les fonctions g linéaires, on dit que le (PMD) est un *programme quadratique convexe discret*. On peut l'écrire sous la forme :

$$\begin{aligned} &\text{Maximiser } x^T Qx + Lx = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j + \sum_{i=1}^{n} l_i x_i \\ &\quad \text{sous les contraintes} \\ &\quad g(x) \leq 0 \\ &\quad x \in \mathbb{Z}^n. \end{aligned}$$

Si les fonctions f et g sont quadratiques, on parle parfois également de programme quadratique, on le nommera ici *programme quadratique à contraintes quadratique*.

Une matrice est dite *positive* (resp. *semi-définies positives*) si, pour tout $x \in R^n$, $x^T Ax = \sum_{i=1}^{n} \sum_{j=1} A_{ij} x_i x_j > 0$ (resp $x^T Ax \geq 0$). Dans le cas où la matrice Q d'un PQ est définie positive, la fonction f est convexe et il existe un optimum local. Résoudre la relaxation continue d'un PQD est un problème NP-difficile en général. Il existe de très performants algorithmes pour résoudre la relaxation continue des PQD avec contraintes linéaires. Ces algorithmes sont en fait inspirés des méthodes pour la PL en utilisant des principes proches de l'algorithme du simplexe ou des points intérieurs (appelé parfois dans ce cas Barrier algorithm). Les implémentations dans Cplex permettent de résoudre également de grands programmes. Des solveurs libres existent OpenOpt par exemple.

Programme quadratique

La version discrète de ces problèmes est difficile et les tailles solvables sont assez réduites (Cplex atteint quelques centaines de contraintes/variables).

Si Q est définie positive, c'est-à-dire de la *programmation quadratique convexe à contrainte quadratique*, la résolution est polynomiale : on sait la résoudre efficacement. On se ramène alors au cas d'un programme semi-défini.
Il existe un cas très particulier mais utile de contrainte non-linéaire que l'on sait bien résoudre (par exemple par CPLEX et GUROBI) : les formes coniques du 2nd ordre (second order cone program)

Programme semi-défini

Un autre type de programme continu, qui a priori sort un peu du cadre défini ici de la (PM) classique est celui du programme semi-défini.

On définit alors un *programme semi-défini* comme étant celui des deux problèmes ci-dessous

Maximiser $Trace(A_0^T X)$	Minimiser $c^t y$
sous les contraintes	sous les contraintes
$A_i X \leq c_i, i = 1, ..., m$	$\sum_{i=1,...,m} A_i y_i - A_0$ est semi-définie positive
X matrice semi-définie positive.	$y \in \boldsymbol{R}^m$

où A_i, $i = 0, ..., m$, sont des matrices semi-définies positives et $c \in \boldsymbol{R}^m$.

En fait, ces deux programmes sont dit duaux l'un de l'autre et ont la même solution optimale. Le programme dual écrit ci-dessus se rapproche de l'écriture d'un programme mathématiques.
Il est facile de montrer que tout programme quadratique convexe peut se ramener à un programme semi-défini et que la programmation semi-définie est une généralisation de la programmation linéaire.

Programme semi-défini

Des extensions des méthodes de points intérieurs permettent de résoudre efficacement la SDP : on trouvera des références à des techniques et des solveurs à http ://www-user.tu-chemnitz.de/~helmberg/semidef.html.

On commence peu à peu à considérer des SDP discrets. Par exemple l'outil BigCrunch http ://www-lipn.univ-paris13.fr/BiqCrunch qui s'appuie sur la programmation semi-définie pour résoudre tout problème quadratique entier.

Et surtout, les SDP jouent un grand rôle pour obtenir de bonnes relaxations de problèmes NP-difficiles : problème de coloration, problème de la coupe maximale,...

Convexification

Lorsqu'un programme mathématique n'est pas convexe, on peut parfois le ramener au cadre convexe pour le résoudre. On appelle *convexification* l'ensemble des astuces permettant de rendre convexe ou d'améliorer la convexité d'un PM ou d'un PMD.

Par exemple, si la matrice de la fonction objective d'un PMD n'est pas définie positive, il est possible de la rendre définie positive en la réécrivant différement (ou en lui ajoutant des variables fictives bien choisie).

C'est un domaine d'étude à part entière pour lequel nous ne regardons ici qu'un exemple.

Convexification

Considérons l'exemple suivant :
La fonction

$$q(x_1, x_2) = Cx_1x_2 \quad \forall (x_1, x_2) \in \{0,1\}^2$$

où C est une constante positive. Cette fonction n'est pas convexe car, pour tout x_1, x_2 pris dans $\boldsymbol{R}$, elle n'est pas toujours positive ou nulle. En revanche, la fonction

$$\tilde{q}(x_1, x_2) = \frac{1}{2}C(x_1 + x_2)^2 - \frac{1}{2}C(x_1 + x_2) \quad \forall (x_1, x_2) \in \{0,1\}^2$$

est convexe. En effet, la partie quadratique de $\tilde{q}$ est positive ou nulle pour toute valeur des variables dans $\boldsymbol{R}$.

Or on peut remarquer que pour tout $(x_1, x_2) \in \{0,1\}^2$, $x_1^2 = x_1$ et $x_2^2 = x_2$ pour des variables binaires ! Ainsi sur les nombres binaires, les deux fonctions coïncident !

$$\tilde{q}(x_1, x_2) = \frac{1}{2}C(x_1^2 + x_2^2 - 2x_1x_2) - \frac{1}{2}C(x_1 + x_2) = q(x_1, x_2)$$

On peut donc utiliser $\tilde{q}$ qui est convexe à la place de q.

Il existe ainsi plusieurs procédé de convexification automatique ou non.

Linéarisation

On appelle *linérisation* l'ensemble des astuces permettant de transformer en un PL ou un PLNE un PMD qui n'est pas linéaire à l'origine. Généralement, ces transformations demandent l'ajout de nombreuses variables supplémentaires.

Cette section comporte quelques astuces pour effectuer une telle linéarisation à partir d'une forme quadratique.

ATTENTION ces transformations ont un coût en nombres de variables et contraintes ajoutées : les techniques citées auparavant sont parfois bien plus efficaces !

Ecriture en variables binaires

Trois résultats permettent de formuler toute variable discrète par des variables binaires.

• *Une variable à valeurs dans un espace discret*
Soit x une variable prenant ses valeurs parmi les n possibilités $v_1, ..., v_n \in \boldsymbol{R}$. On peut alors poser n variables de décisions binaires $y_1, ..., y_n$ telle que $y_i = 1$ si $x = v_i$ et 0 sinon.

Ce cas peut alors se modéliser par les deux contraintes $x = \sum_{i=1}^{n} v_i y_i$ et $\sum_{i=1}^{n} y_i = 1$.

• *Ecriture en variables binaires*
Si l'on peut déterminer des bornes sur les variables, on peut utiliser l'idée du cas précédent pour écrire un PLNE en variables entières comme un PLNE à variables binaires. En effet, considérons une variable x à valeurs entières entre 0 et u, $u \in \boldsymbol{N}$. Soit n tel que $2^n \leq u < 2^{n+1}$. On remplace alors x par sa représentation binaire :
$x = \sum_{i=1}^{n} 2^i y^i$ où $y_i \in \{0, 1\}$ pour $i = 1, ..., n$.

• Carré d'une variable binaire
Soit $x \in \{0,1\}$. Alors la variable x^2 est équivalente à la variable x.

• Produit de deux variables binaires :
Soit $x \in \{0,1\}$ et $y \in \{0,1\}$, on veut obtenir une variable e ayant la valeur $e = xy$.
En fait, on a le résultat suivant :

$$e = xy \Leftrightarrow \begin{cases} e \leq x \\ e \leq y \\ e \geq x + y - 1 \\ e \geq 0 \\ e \in \boldsymbol{R} \end{cases}$$

On peut généraliser ce résultat au produit d'une variable binaire par une variable entière (bornée), au produit de plusieurs variables binaires, au carré d'une variables binaires,...

Conclusion : Au total de ces linéarisations, on peut remarquer que toute forme quadratique peut se ramener à un PLNE binaire ! Mais cela se fait au prix fort, en ajoutant de nombreuses variables et contraintes.

Conclusion

Tout au long de ce cours, nous verrons à quel point l'écriture sous forme d'un PLNE est un puissant outil de modélisation. En fait, on peut le voir souvent comme l'écriture naturelle algébrique d'un problème.

Inversement, comme nous l'avons cité, il n'existe pas de méthodes génériques efficaces pour résoudre un PLNE. Le fait de ramener aussi facilement un problème à un PLNE est donc parfois dangereux : on voit souvent des chercheurs ou des ingénieurs R&D conclure un travail de modélisation en affirmant que "comme on a ramené notre problème à un PLNE que les solveurs n'arrivent pas à résoudre, notre problème est difficile et nous allons utiliser des méthodes approchées". Cet état d'esprit, fort répandu malheureusement, est doublement faux. Tout d'abord, ramener un problème à un PLNE ne prouve en rien la difficulté d'un problème : ce n'est pas une preuve de complexité, il peut ainsi exister d'autres pistes théoriques ou algorithmiques pour résoudre le problème d'origine.

Il peut exister plusieurs PLNE modélisant le problème et différentes techniques pour le résoudre : c'est justement cette étude que nous allons mener dans ce cours.