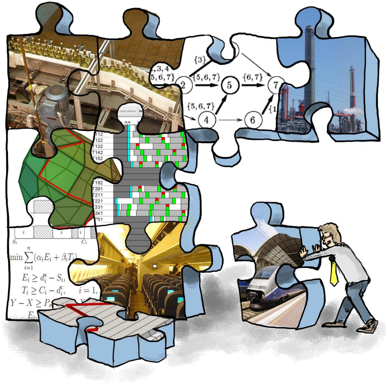
Optimisation et Recherche Opérationnelle

O. Grunder[1]

[1]Formation Informatique
Université de Technologie de Belfort-Montbéliard

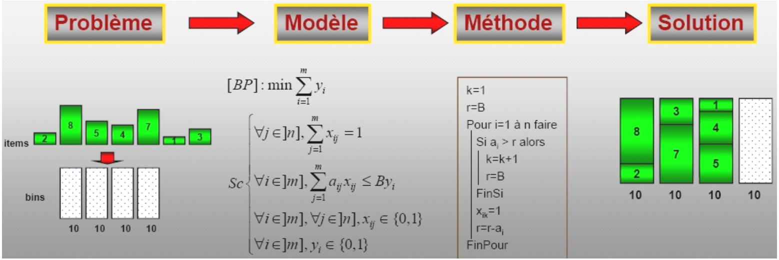
20 mars 2024

Le livre blanc de la Recherche Opérationnelle[Roadef 11]



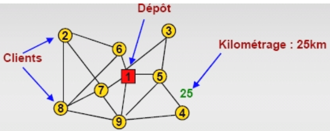
Crédits photo : Lionel Lagarde [Roadef 11]

Le « Bin Packing Problem »



Crédits photo : Lionel Lagarde [Roadef 11]

Le « Traveling Salesman Problem » ou « Problème du Voyageur de Commerce »



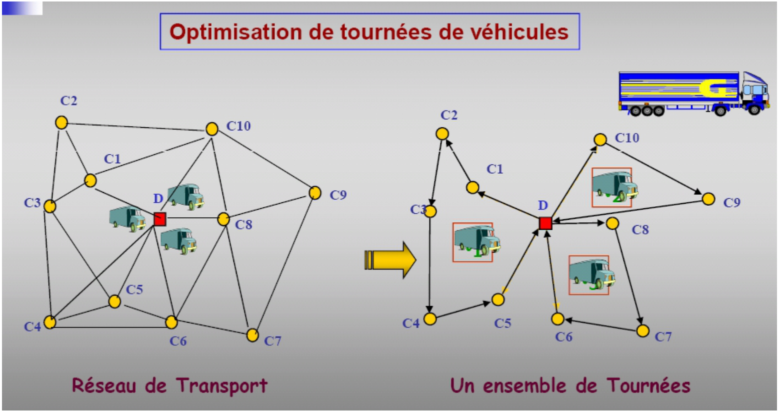
Objectif :
Partant du dépôt, visiter une une seule fois chacun des clients et revenir au dépôt en parcourant une distance totale minimale.

n	*Nombre de possibilités*	*Temps de calcul (ordre de grandeur)*
5	12	micro-seconde
10	181440	dixième de seconde
15	43 milliards	dizaine d'heures
20	60.10^{15}	milliers d'années
25	310.10^{21}	milliards d'années

Une possibilité par micro-seconde

Crédits photo : Lionel Lagarde [Roadef 11]

Le « Vehicle Routing Problem »



Crédits photo : Lionel Lagarde [Roadef 11]

Qu'est-ce que la Recherche Opérationnelle ?[Roadef 11]

- Approche quantitative permettant de produire de meilleures décisions dans les organisations (industrielles, humaines...)
- Outils et modèles pour optimiser le fonctionnement des systèmes industriels et économiques.
- Modèles pour permettre aux décideurs de faire des choix efficaces et robustes.
- Discipline carrefour entre les mathématiques, l'informatique et l'économie
- En prise directe avec l'industrie et rôle-clé dans le maintien de la compétitivité.
- Fait partie du domaine de l'aide à la décision et de l'intelligence artificielle au sens large

Apports de la Recherche Opérationnelle[Roadef 11]

- organisation des lignes de production d'automobiles → planification des missions spatiales,
- optimisation des portefeuilles bancaires → aide au séquençage de l'ADN
- mais aussi dans la vie de tous les jours pour
  - le recyclage des déchets,
  - l'organisation des ramassages scolaires,
  - les emplois du temps des infirmières ou
  - la couverture satellite des téléphones portables...

La RO en France[Roadef 11]

ORANGE LABS
GDF SUEZ
EURODECISION
BOUYGUES
AIR FRANCE
AIR LIQUIDE
ORDECSYS
ALMA
RENAULT
GOOGLE
ORACLE
SNCF
LA POSTE
AMADEUS
OPTILOGISTICS
COSYTEC
ARTELYS
EDF
ILOG
SFR

Exemple de problèmes de RO[Roadef 11]

- AIR LIQUIDE : organisation des tournées de camions
  - servir tous les clients à la bonne fréquence
  - remplir les camions
- ORANGE LABS : Dimensionnement des réseaux radiomobiles
  - optimisation de l''orientation (verticale et horizontale) des antennes,
  - optimisation des puissances de transmission.
- SFR : Dimensionnement du réseau ADSL
  - optimiser les investissements liés à l'acquisition de nouveaux clients sur les 3 000 répartiteurs où elle possède des infrastructures propres.
  - En cas de sous- capacité locale, l'offre proposée est dégradée aussi bien pour le client que pour l'opérateur.

Exemple de problèmes de RO[Roadef 11]

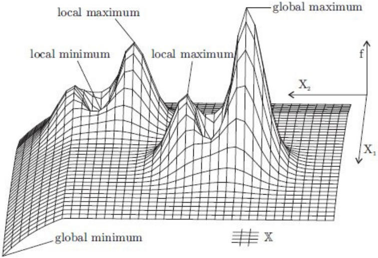
- ALMA - problèmes de découpe
  - chemise, navire, pelleteuse, toboggan
  - 50 % à 80 % du coût du tissu pour le prix d'un vêtement sortie d'usine,
  - dépenser le - de matière première = imbriquer au mieux les pièces qui composent ces produits.
- RENAULT : optimisation des transports pièces
  - transport des pièces des fournisseurs vers les usines de montage.
  - optimiser la constitution des piles de palettes de pièces,
  - optimiser le placement de ces piles dans un camion.
- AIR FRANCE : planification mensuelle du personnel navigant (10 000 hôtesses et stewards) en moins de 3h
- SNCF : SIOUCS
  - accident, travaux : réorganiser traffic des trains sur une voie unique
  - utilisation de modèles dérivés de la théorie de l'ordonnancement
  - temps de calcul d'une trentaine de secondes

Caractérisation d'un problème de RO

Un problème de RO est caractérisé par :

- des **paramètres** :
  - information lié à une partie du problème
  - nécessite d'être pris en compte pour la prise de décision
- des **variables**
  - représente un décision à prendre pour optimiser le problème considéré
- une **fonction objectif** (**coût**, **critère**)
  - formule mathématique qui sert de critère pour déterminer la meilleure solution au problème d'optimisation.
  - Elle associe à chaque solution une valeur réelle, voire entière
  - Ex : longueur d'un chemin, temps de fabrication, économie réalisée, nombre de routeurs traversés, ...
- des **contraintes**
  - condition que la solution du problème d'optimisation doit satisfaire pour être admissible.
  - contraintes d'égalité ou d'inégalité

Optimums



Définition

On appelle :
solution de (P) : tout x qui vérifie les contraintes du problème.
optimum global de (P) : toute solution x de (P) qui optimise (max ou min) le critère.
optimum local de (P) : toute solution $x^{\mathbf{0}}$ de (P) qui optimise le critère sur un voisinage $V(x^{\mathbf{0}})$.

Exemple du problème du sac à dos

KnapSack Problem (KSP)

- Soit un sac à dos d'une capacité donnée
- Soient des objets caractérisés par
  - un poids et
  - une valeur
- Le KSP consiste à identifier les objets à emporter dans le sac à dos
  - en maximisant la valeur emportée
  - tout en respectant la capacité du sac.

Modélisation du problème du sac à dos

Modélisation du KSP

- Paramètres :
  - capacité du sac B,
  - nombre d'objets n
  - poids a_i et valeur c_i pour chaque objet i
- Variables : x_i vaut 1 si on décide de prendre l'objet i, 0 sinon
- Fonction objectif : somme des valeurs des objets emportés $\sum_i c_i.x_i$
- Contrainte : la somme des poids des objets emportés $\sum_i a_i.x_i \leq B$

Modélisation du problème du sac à dos

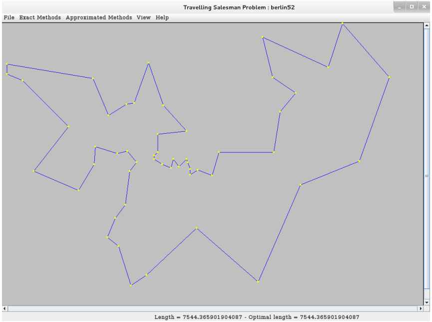
Modélisation du KSP

$$\left\{\begin{array}{rrrcc} max\, z = & \sum_i & c_i.x_i & & \\ & \sum_i & a_i.x_i & \leq & B \\ & & x_i & \in & \{0,1\} \end{array}\right.$$

Exemple du voyageur de commerce

Traveling Salesman Problem (TSP)

- Soit un ensemble de villes séparées par des distances données,
- Le TSP consiste à trouver le plus court chemin qui passe une et une seule fois par chacune des villes et revient à la ville de départ



Modélisation du voyageur de commerce

Modélisation du TSP

- Paramètres :
  - nombre de villes n
  - distance d_{ij} de ville i à ville j
- Variables : x_{ij} vaut 1 si on décide d'aller de i à j, 0 sinon
- Fonction objectif : somme des distances parcourues $\sum_{ij} d_{ij}.x_{ij}$
- Contraintes : former un circuit hamiltonien (visite de chaque ville une et une seule fois)

Modélisation du voyageur de commerce

Formulation mathématique du TSP

$$\left\{\begin{array}{rccccc} min\, z = & \sum_{ij} & d_{ij}.x_{ij} & & \\ \forall i, & \sum_j & x_{ij} & = & 1 \\ \forall j, & \sum_i & x_{ij} & = & 1 \\ \{(i,j) & /x_{ij} & =1\} & : & circuit \\ x_{ij} & binary & & & \end{array}\right.$$

Formulation mathématique générale

Problème (P) :

- Optimisation (max ou min) d'un critère $f(x_1, x_2, \ldots, x_n)$ (ft de coût, objectif),
- Tout en respectant un ensemble de contraintes pouvant être formulées sous forme d'équations algébriques : $g_i(x_1, x_2, \ldots, x_n) \leq 0$ pour $i = 1..m$

$$
(P) \left\{ \begin{array}{llllllll}
max & f(& x_1, & x_2, & \ldots, & x_n) & & \\
 & g_1(& x_1, & x_2, & \ldots, & x_n) & \leq & 0 \\
 & g_2(& x_1, & x_2, & \ldots, & x_n) & \leq & 0 \\
 & & & & & & & \\
 & g_m(& x_1, & x_2, & \ldots, & x_n) & \leq & 0
\end{array} \right.
$$

Formulation mathématique linéaire d'un problème de RO

Problème (P) :

- Formulation algébrique :

$$
(P)\left\{\begin{array}{llllllll}
max & z = & c_1x_1 & +c_2x_2 & +\dots & +c_nx_n & \\
 & & a_{11}x_1 & +a_{12}x_2 & +\dots & +a_{1n}x_n & \leq b_1 \\
 & & \dots & \dots & & \dots & \dots \\
 & & a_{i1}x_1 & +a_{i2}x_2 & +\dots & +a_{in}x_n & \leq b_i \\
 & & \dots & \dots & & \dots & \dots \\
 & & a_{m1}x_1 & +a_{m2}x_2 & +\dots & +a_{mn}x_n & \leq b_m \\
x_i & \geq 0 & & & & &
\end{array}\right.
$$

- Notation vectorielle :

$$
(P)\left\{\begin{array}{llll}
max & z = & c.x & \\
 & & A.x & \leq b \\
x & \geq 0 & &
\end{array}\right.
$$

Logiciels de calcul scientifique

- Matlab (1970) de MathWorks :
  - Optimization Toolbox
    - Problèmes de programmation quadratique et linéaire
    - Méthodes de résolution des problèmes de programmation d'entier binaire
    - Parallélisation des méthodes
- Octave (1988) : open source, licence GNU
- Scilab (1990) de l'INRIA : open source, GNU GPL
  - Module FOSSEE_Optimization_Toolbox, Quapro (problèmes de mises à jour) :programmation linéaire, quadratique

Solveurs commerciaux

- IBM ILOG CPLEX Optimizer
- Gurobi Optimizer 5.1
- LINDO Systems - Optimization Software : Integer Programming, Linear Programming, Nonlinear Programming, Stochastic Programming, Global Optimization
- FICO™ Xpress Optimization Suite

Solveurs non commerciaux/open source

- MINTO : solves MILP by a branch-and-bound algorithm with LP relaxations
- CBC : open-source MIP solver written in C++
- SYMPHONY : open-source solver for MILPs written in C
- GLPK : (GNU Linear Programming Kit) ANSI C callable library.
  - primal and dual simplex methods
  - primal-dual interior-point method
  - branch-and-cut method
  - application program interface (API)
  - stand-alone LP/MIP solver
- lp_solve : free linear (integer) solver
- `www.coin-or.org` : COmputational INfrastructure for Operations Research

Programme linéaire à 2 variables

Atelier d'assemblage

- 3 composants a, b et $c \rightarrow$ 2 produits P_1 et P_2

	P_1	P_2	Stock
a	2	3	180
b	2	1	120
c	1	3	150
Gain	3	4	

Revenu maximum ?

- Réponse intuitive : maximiser le produit P_2 le plus rentable
- $x_2 = min\{\frac{180}{3}, \frac{120}{1}, \frac{150}{3}\} = 50$ et $gain = 4x_2 = 200$

Formulation algébrique

		P_1	P_2	Stock
	a	2	3	180
	b	2	1	120
	c	1	3	150
objectif→	**Gain**	**3**	**4**	

- Appelons x_1 et x_2 les quantités respectives des produits P_1, P_2.
- Critère à maximiser :$max\ z = 3x_1 + 4x_2$

Formulation algébrique

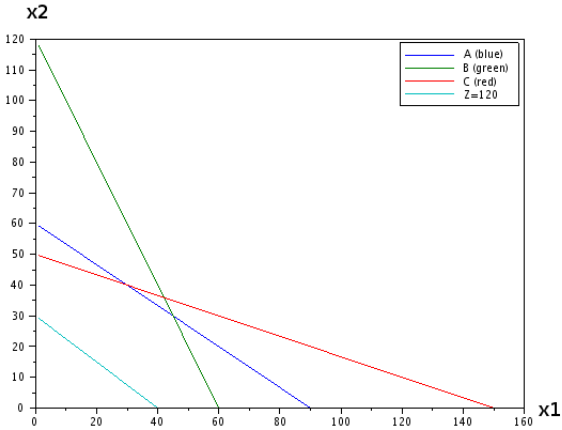
		P_1	P_2	Stock
contrainte (1)$\rightarrow$	a	**2**	**3**	**180**
contrainte (2)$\rightarrow$	b	**2**	**1**	**120**
contrainte (3)$\rightarrow$	c	**1**	**3**	**150**
	Gain	3	4	

- Contraintes sur les composants : $\begin{cases} 2x_1 & +3x_2 & \leq & 180\,(1) \\ 2x_1 & +x_2 & \leq & 120\,(2) \\ x_1 & +3x_2 & \leq & 150\,(3) \end{cases}$
- Contraintes de positivité des variables : $x_1 \geq 0; x_2 \geq 0$

Formulation algébrique finale

$$\left\{\begin{array}{rrrcr} max\, z = & 3x_1 & +4x_2 & & \\ & 2x_1 & +3x_2 & \leq & 180 \\ & 2x_1 & +x_2 & \leq & 120 \\ & x_1 & +3x_2 & \leq & 150 \\ x_i \geq 0 & & & & \end{array}\right.$$

Représentation graphique



Programme linéaire avec Matlab

- Matlab minimise : il faut changer l'objectif en $min\, z = -3x_1 - 4x_2$

```
>> X1 = optimvar('X1','LowerBound',0);
X2 = optimvar('X2','LowerBound',0);

% Creation du probleme et de l objectif (Matlab minimise l'objectif) :
linprob = optimproblem('Objective',-3*X1-4*X2);

% Definition des contraintes :
linprob.Constraints.cons1 = 2*X1 + 3*X2 <= 180 ;
linprob.Constraints.cons2 = 2*X1 + 1*X2 <= 120 ;
linprob.Constraints.cons3 = 1*X1 + 3*X2 <= 150 ;

% Resolution du programme linéaire :
linsol = solve(linprob);

% Affichage de la valeur de la solution :
evaluate(linprob.Objective,linsol)

% Affichage de la solution :
tbl = struct2table(linsol)
```

```
ans =

  -255


tbl =

  1×2 table

    X1    X2
    __    __

    45    30
```

Programme linéaire avec Octave

- Problème à résoudre :

$$\begin{cases} max\, z = 3x_1 & +4x_2 & & \\ 2x_1 & +3x_2 & \leq & 180 \\ 2x_1 & +x_2 & \leq & 120 \\ x_1 & +3x_2 & \leq & 150 \\ & x_i & \geq & 0 \end{cases}$$

```
A=[2 , 3; 2 , 1 ; 1 , 3];
b=[180;120;150];
c=[3;4];
lb = [0; 0];
ub = [];
ctype = "UUU";     # constraint Upperbound <=
vartype = "CC";    # variable Continuous
sense = -1;        # maximisation
param.msglev = 1;  # msg level: 1 = errors and warnings
param.itlim = 100; # Simplex iterations limit

# solve linear program with 'glpk'
[xmin, fmin, status, extra] = glpk (c, A, b, lb, ub, ctype, vartype, sense, param);
```

```
xmin =

   45
   30

fmin = 255
```

Programme linéaire avec Scilab

- Il faut changer l'objectif en $min\, z = -3x_1 - 4x_2$

```
--> A=[2 , 3; 2 , 1 ; 1 , 3];

--> b=[180;120;150];

--> c=[-3;-4];

--> // solve linear program of module 'FOSSEE_Optimization_Toolbox'

--> x = fot_linprog(c,A,b)

Optimal Solution.
 x  =

   45.   30.
```

Modélisation GLPK par GMPL

- Problème à résoudre : $\begin{cases} \max z = 3x_1 + 4x_2 & \\ 2x_1 + 3x_2 \leq 180 \\ 2x_1 + x_2 \leq 120 \\ x_1 + 3x_2 \leq 150 \\ x_i \geq 0 \end{cases}$
- Problem description with GMPL format : fichier « cm01.mod »

```
/* Decision variables */
var x1, integer, >= 0 ;
var x2, integer, >= 0 ;
/* Objective function */
maximize objz: 3*x1 + 4*x2 ;
/* Constraints */
s.t. R_Stock_A : 2*x1 + 3*x2 <= 180 ;
s.t. R_Stock_B : 2*x1 + x2 <= 120 ;
s.t. R_Stock_C : x1 + 3*x2 <= 150 ;
end ;
```

Résolution GLPK

$ glpsol --math cm01.mod --output res.log
GLPSOL : GLPK LP/MIP Solver, v4.45
...
GLPK Integer Optimizer, v4.45
...
Solving LP relaxation...
...
OPTIMAL SOLUTION FOUND
Integer optimization begins...
...
INTEGER OPTIMAL SOLUTION FOUND
...
Model has been successfully processed
Writing MIP solution to 'res.log'...

```
Status:     INTEGER OPTIMAL
Objective:  z = 255 (MAXimum)

   No.   Row name        Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
     1 z                         255
     2 R_Stock_A                 180                         180
     3 R_Stock_B                 120                         120
     4 R_Stock_C                 135                         150

   No. Column name       Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
     1 x1           *             45             0
     2 x2           *             30             0
```

GMPL (GNU Math Programming Language)

Scite Editor

File Edit Search View Tools Options Language Buffers Hel

1 ag41-cm01.mod

Modèle

```
/* Parameters */
param gain1 ;
param gain2 ;

/* Decision variables */
var x1, integer, >= 0 ;
var x2, integer, >= 0 ;

/* Objective function */
maximize z: gain1*x1 + gain2*x2 ;

/* Constraints */
s.t. R_Stock_A : 2*x1 + 3*x2 <= 180 ;
s.t. R_Stock_B : 2*x1 +   x2 <= 120 ;
s.t. R_Stock_C : x1   + 3*x2 <= 150 ;

solve;

printf "nbr de P1 =%d\n", x1 ;
printf "nbr de P2 =%d\n", x2 ;
printf "Gain total=%d\n", z ;
```

Données

```
data;

param gain1 := 3 ;
param gain2 := 4 ;

end ;
```

```
2 integer variables, none of which are binary
Preprocessing...
3 rows, 2 columns, 6 non-zeros
2 integer variables, none of which are binary
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  3.000e+00  ra
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part = 3
Solving LP relaxation...
GLPK Simplex Optimizer, v4.45
3 rows, 2 columns, 6 non-zeros
*     0: obj =   0.000000000e+00  infeas =  0.000e+(
*     4: obj =   2.550000000e+02  infeas =  0.000e+(
OPTIMAL SOLUTION FOUND
Integer optimization begins...
Gomory's cuts enabled
MIR cuts enabled
Cover cuts enabled
Clique cuts enabled
Creating the conflict graph...
The conflict graph is either empty or too big
+     4: mip =     not found yet <=              +inf
+     4: >>>>>   2.550000000e+02 <=   2.5500000
+     4: mip =   2.550000000e+02 <=     tree is emp
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (126146 bytes)
nbr de P1 =45
nbr de P2 =30
Gain total=255
Model has been successfully processed
>Exit code: 0
```

GMPL syntaxe du modèle

- Paramètres :

```
param f, integer, >=0 ;
param n := 10 ;
```

- Variables

```
var x, binary ;
var y, integer, >=0 ;
```

- Ensembles

```
set OBJECT := {'pomme', 'chocolat', "corde", "couteau"} ;
set TOWN := 1..n ;
```

- Expression indicée (équivalent à une boucle « **for** i **in** SET »)

```
forme simple : {i in SET} # equivalent « for i in SET »
    param weight{i in OBJETS} ;
    sum{i in OBJECTS} weight[i]*y[i] ;
forme générale : {entry1, entry2, ..., entrym [: predicate]}
    var x{i in TOWN, j in TOWN}, binary ;
    sum{i in TOWN, j in TOWN: i!=j} dist[i,j]*x[i,j] ;
```

GMPL syntaxe du modèle

- Objectif

```
maximize z1 : sum{i in OBJECTS} value[i]*y[i] ;
minimize z2 : sum{i in TOWN, j in TOWN, i!=j} dist[i,j]*x[i,j] ;
```

- Contraintes

```
s.t. Capacity : sum{i in OBJECTS} weight[i]*x[i] <= C ;
```

- Famille de contraintes paramétrées

```
set SAC := 1..3 ;
s.t. R{k in SAC} : sum{j in OBJECTS} weight[j]*x[j,k] <= C[k] ;
```

équivalent à : $\forall k = 1..3, \sum_{j=1}^{n} weight_j \times x_{j,k} \leq C_k$

- Commandes spéciales
  - solve : démarre la résolution, implicite à la fin de la section modèle
  - display, printf : affichage après « solve »
  - for{i in I [:condition]} : boucle

GMPL syntaxe des données

- Scalaire :

```
display mon_param := 13 ;
```

- tableau dim 1

```
set OBJECTS := {'pomme', 'chocolat', "corde", "couteau"} ;
param w{i in OBJECTS} ;
...
data;
param w := pomme      150
           chocolat   250
           corde     1000
           couteau    200 ;
```

- tableau dim 2

```
param d{i in 1..3, j in 1..3} ;
...
data;
param dist : 1    2    3 :=
       1     13   24   84
       2    120   26  102
       3     67  176   98 ;
```

GMPL Problème du sac à dos

```
set OBJ ;
param w {i in OBJ}; param v {i in OBJ};
param C ;
var x {i in OBJ}, binary ;
maximize z: sum{i in OBJ} v[i]*x[i] ;
s.t. RCap : sum{i in OBJ} w[i]*x[i] <= C ;
solve ;
/* Data section */
data;
param C := 7 ;
param :OBJ : v w :=
       obj1 15 3
       obj2  4 1
       obj3  7 3
       obj4  2 1
       obj5  1 1 ;
end ;
```

$$\left\{ \begin{array}{rrrrr} max\, z = & \sum_i & c_i.x_i & & \\ & \sum_i & a_i.x_i & \leq & B \\ & & x_i & \in & \{0,1\} \end{array} \right.$$

Modélisation : logique

Décisions A, B, C, ... associées aux variables binaires a, b, c, ... qui sont à 1 si on prend la décision correspondante.

- au moins N parmi A, B, C,... : $a+b+c+... \geq N$
- au plus N parmi A, B, C,... : $a+b+c+... \leq N$
- exactement N parmi A, B, C,... : $a+b+c+... = N$
- if A then B : $b \geq a$

Modélisation : logique

- d = d1 and d2 ou min(d1,d2)

$$\begin{cases} d & \leq & d_1 \\ d & \leq & d_2 \\ d & \geq & d_1 + d_2 - 1 \\ d & \geq & 0 \end{cases}$$

- d = d1 or d2 ou max(d1,d2)

$$\begin{cases} d & \geq & d_1 \\ d & \geq & d_2 \\ d & \leq & d_1 + d_2 \\ d & \leq & 1 \end{cases}$$

Modélisation : linéarisation d'un produit

- Produit $y = x \times d$ avec d binaire et $0 \leq x \leq U$

$$\left\{ \begin{array}{ccccc} 0 & \leq & y & \leq & U \times d \\ & & y & \leq & x \\ & & x - U \times (1 - d) & \leq & y \end{array} \right.$$

- Si $d = 0$:

$$\left\{ \begin{array}{ccccc} 0 & \leq & y & \leq & U \times 0 \\ & & y & \leq & x \\ & & x - U & \leq & y \end{array} \right.$$

- Si $d = 1$:

$$\left\{ \begin{array}{ccccc} 0 & \leq & y & \leq & U \\ & & y & \leq & x \\ & & x - U \times (1 - 1) & \leq & y \end{array} \right.$$

Modélisation : linéarisation d'un produit

- Produit $y = x \times d$ avec d binaire et $L \leq x \leq U$

$$\begin{cases} L \times d & \leq & y & \leq & U \times d \\ L \times (1-d) & \leq & x-y & \leq & U \times (1-d) \end{cases}$$

- Produit de 2 binaires : $d_3 = d_1 \times d_2$

$$\begin{cases} d_3 & \leq & d_1 \\ d_3 & \leq & d_2 \\ d_3 & \geq & d_1 + d_2 - 1 \end{cases}$$

Complexité des algorithmes

- Problème définit par des paramètres
- Instance d'un problème : fixer les valeurs des paramètres du problème
- Algorithme = suite d'opérations élémentaires (affectations, tests, boucles, ...) qui renvoie une solution pour une instance d'un problème à résoudre
- Un même problème peut être résolu par plusieurs algorithmes : important de comparer les performances des algorithmes

Complexité des algorithmes

- Les 2 critères d'évaluation sont :
  - temps de calcul (plus important)
  - espace mémoire

Définition

Complexité en temps = relation de proportionnalité entre le nombre d'opérations élémentaires à effectuer pour trouver une solution et la taille des données de départ (n).

- Par abus de langage, on appelera *complexité d'un algorithme*, sa complexité en temps.

Nombre d'opérations d'un algorithme

- Temps nécessaire pour la multiplication de matrices : $C = A \times B$.
  1. pour $i = 1$ à n
  2. pour $j = 1$ à n
  3. $C(i,j) = 0$
  4. pour $k = 1$ à n
  5. $C(i,j) = C(i,j) + A(i,k) \times B(k,j)$
- Complexité de l'algorithme en n^3

Nombre d'opérations d'un algorithme

- Algorithme de tri par sélection d'un tableau t
  1. pour i de 1 à n - 1
  2. min ← i
  3. pour j de i + 1 à n
  4. si t[j] < t[min], alors min ← j
  5. si min ≠ i, alors échanger t[i] et t[min]
- i = 1 => j prend n-1 valeurs
- i = 2 => j prend n-2 valeurs ...
- Nombre d'opérations = (n-1)+(n-2)+...+ 2 + 1 = n*(n-1)/2
- Complexité en n^2

Complexité

- Certains algorithmes ne donnent pas la même complexité suivant les données de départ. On parle alors de :
  - complexité dans le pire des cas : plus grande complexité pour toutes les données de taille n (plus importante),
  - complexité dans le meilleur des cas : plus petite complexité,
  - complexité en moyenne : difficile.
- Pourquoi utilise-t-on la complexité dans le pire des cas :
  - Certitude que l'algo ne pourra pas être pire !
  - plus facile à obtenir

Ordre de grandeur des complexités

- Soit f une fonction croissante sur l'ensemble des entiers positifs.

Définition

$O(f)$ est la classe des fonctions g définies sur l'ensemble des entiers positifs, telles que : $\exists\, c > 0$, $\exists\, n_0 > 0$ tels que $\forall\, n \geq n_0$, $g(n) \leq c.f(n)$.

- On note $O(1)$ l'ensemble des fts majorées par une constante.
  Exemple : $g(n) = 10/n \in O(1)$
- A une constante multiplicative près, de nombreuses classes sont équivalentes :
  $O(10000n^2) = O(0{,}001n^2) = O(n^2)$ et $O(e^n) = O(2^n)$.

Algorithmes polynomiaux

- Pour tout polynôme $P(n) = a_p n^p + ... + a_1 n^1 + a_0$, on a $P \in O(n^p)$.

Définition

Un algorithme polynomial de degré k (k constante indépendante de n) a une complexité en $O(n^k)$.

Ordre de grandeur des complexités

- Temps d'exécution d'un algorithme en $O(f)$ pour une donnée de taille n ($f(n)$ opérations) sur un ordinateur exécutant 10^9 ops/secondes.

	$n = 10$	$n = 100$	$n = 1000$
$f(n) = n$	$0,01\mu s$	$0,1\mu s$	$1\mu s$
$f(n) = n^3$	$1\mu s$	$1ms$	$1s$
$f(n) = n^5$	$100\mu s$	$10s$	$11,5jrs$
$f(n) = 2^n$	$1\mu s$	$40.10^6 ma$	$3,4.10^{278} ma$
$f(n) = n!$	$3,6ms$	$2,96.10^{135} ma$	

- ma = million d'années
- Age de la terre : $4,5.10^9$ années

Complexité des problèmes

Définition

Complexité d'un problème = complexités du meilleur de tous les algorithmes pouvant le résoudre.

- 3 catégories de problèmes (théorie de la complexité) :
  - Les problèmes de décision : répondre par oui/non
  - Les problèmes de calcul de la valeur optimale : recherche de la valeur optimale
  - Les problèmes d'optimisation : recherche de la solution optimale

Complexité des problèmes de décision : P et NP

Définition

Classe P : ensemble des problèmes polynomiaux, pouvant être résolu par un algorithme en $O(n^k)$, k constante indépendante de n

Exemple : parité ($O(1)$), plus court chemin dans un graphe ($O(n^2)$)

Définition

Un problème de décision Π est dans NP si la validité de toute solution de Π est vérifiable en temps polynomial.

Exemple :plus court (resp. long) chemin, TSP

- Note : $P \subset NP$

Problèmes de décision *NP*-complet

Classe NP-complet

Certains pbs de décision de NP sont plus difficiles que les autres (Cook, 70), de complexité au moins en $O(2^n)$.

- *SAT* (Pb de *satisfiabilité - Cook 70)*
  - soit n variables booléennes x_1, ..., x_n et m clauses parmi $\{x_1, ..., x_n, \overline{x_1}, ..., \overline{x_n}\}$
  - Question : existe-t-il une affectation des x_i de sorte que pour chacune des clauses, au moins l'un des littéraux soit *vrai* ?
  - Ex : $\{x_1, \overline{x_2}, \overline{x_3}\}$ et $\{\overline{x_1}, x_3\}$ $=> x_1 = x_3 = faux$ et $x_2 = vrai$.
- Plus long chemin élémentaire, Cycle hamiltonien, ...
- Difficile de résoudre optimalement des pbs NP-complets de grande taille.

Complexité des problèmes d'optimisation

On peut associer un pb de décision à tout pb d'optimisation

- Optimisation : trouver $s^* \in S / f(s^*) = min\{f(s), s \in S\}$
- Décision : existe-t-il $s^* \in S / f(s^*) < a$

- Quand le problème de décision est NP-complet, le problème d'optimisation est dit NP-difficile.

Problème d'optimisation NP-difficile

- La résolution exacte du problème d'optimisation ne peut se faire que par :
    - Recherche arborescente (type PSE)
    - Méthodes de coupe
    - Programmation dynamique (certains problèmes).
- Problèmes d'optimisation *NP*-difficiles :
    - Problème du Voyageur de Commerce (PVC, TSP)
    - Tournées de véhicules (VRP, PDP), affectations
    - Problèmes d'ordo. avec ressources, emploi du temps,
    - Programmation linéaire en nombres entiers.
- Résolution exacte non envisageable pour problèmes de grande taille :
    - algorithmes polynomiaux avec garantie relative de performance
    - métaheuristiques : recuit simulé, tabou, algo. génétiques, ...

NP-complétude au sens fort/faible

Définition

Un problème est NP-complet *au sens fort* si le problème est NP-complet à cause de sa structure et non pas à cause des valeurs des paramètres des instances.

Exemple : SAT, TSP

Définition

Un problème est NP-complet *au sens faible* si le problème est NP-complet à cause des valeurs des paramètres des instances qui apparaissent dans la complexité des algorithmes qui le résolvent

Exemple : problème du sac à dos

Historique de la recherche opérationnelle (RO)

Technique récente datant de la seconde guerre mondiale appliquée dans le cadre des "opérations" militaires de logistique.
Problèmes de la RO soulevés bien avant :

- XVIIème : décision dans l'incertain (Pascal, Fermat, Bernoulli, Waldegrave) dans les jeux de hasard : Gains des joueurs à un jeu de hasard après un nombre de coups donné ?
- XVIIIème : pb économique combinatoire des déblais/remblais (Monge) qui cherche à transporter du déblai sur un remblai au moindre coût.
- 1917 : théorie des files d'attente (Erlang) qui étudie les solutions optimales de gestion des files d'attente : cas de la gestion des réseaux téléphoniques (à l'époque).

Historique de la RO

- 1921-25 : théorie mathématique des jeux (Borel) étudie les problèmes pour lesquels les choix des joueurs ont des conséquences pour l'adversaire.
- 1936 : systématisation des graphes (König) par l'étude de leurs propriétés suivant leur structure.
- veille de la guerre 39-45 : programmation linéaire pour lesquels la fonction objectif et les contraintes sont toutes linéaires (Kantorovich).

Historique de la RO

- Equipe de Blackett 1940 (physicien anglais, prix Nobel 1948) :
  - Hétérogène : math, physique,
  - Informations et données fiables
  - Décision finale réservée à l'amirauté britannique
- Problèmes traités :
  - Implantation optimale des radars de surveillance britanniques,
  - Tir de DCA : nombre de tirs divisé par 5 (20000 :4000) pour détruire un avion
  - protection des convois de navires marchands entre la grande-bretagne et les USA.
    - pertes alliés indépendantes de taille du convoi
    - perte sous-marins ennemis proportionnel au carré des escorteurs
    - Taille des convois quadruplée (escorteurs seulement doublés) sans risque d'augmentation des pertes

Historique de la RO

- Après la guerre, les méthodes de RO ont ensuite été appliquées à l'économie industrielle.
- De nombreuses travaux scientifiques ont alors été menés pour développer l'ensemble des méthodes et techniques de la RO.

3 classes de problème de RO

- Problèmes combinatoires : investissements, niveaux d'activité, affectation, transport, ordonnancement ;
- Problèmes stochastiques : files d'attente, fiabilité, sûreté de fonctionnement, gestion de production ;
- Problèmes concurrentiels : politiques d'approvisionnement, de vente, etc.

Limites de la RO

- Approche mono-critère
  - Les techniques de RO visent à optimiser une seule et unique fonction.
    - minimiser ses investissements,
    - optimiser sa fabrication (productivité maximale et coût minimal),
    - déterminer le prix de vente le plus intéressant.
  - Tout doit être exprimé sous forme de contraintes sauf la fonction objectif
- Optimisation partielle
  - Modèle complet d'une entreprise trop complexe.
  - Pbs de RO souvent limités à une partie de l'entreprise (par ex. gestion des stocks)
  - Peut perturber les autres parties de l'entreprise

Objectifs et critères

- Limites de la RO : critères relatifs et temporels
  - Un critère d'optimisation à une période donnée peut ne plus être pertinent à une autre période, voire être complètement périmé. Les raisons sont nombreuses :
    - la technique progresse
    - les produits deviennent obsolètes
    - la législation évolue
    - les modes changent
- L'industriel fixe l'objectif et les contraintes, et prend la décision finale.
- La RO doit permettre de fournir une connaissance approfondie du problème pour aider à la prise de décision.

Une discipline carrefour

- La RO est plus une pratique qu'une science.
- Un modèle n'est intéressant que s'il permet d'obtenir de meilleurs résultats.
- La RO est une discipline carrefour entre :
  - l'économie (économie d'entreprise, analyse économique) : modèle initial,
  - les mathématiques (théorie des systèmes, méthodes d'optimisation ou statistiques) : choix d'une voie pour atteindre une solution,
  - l'informatique (algorithmique, structures de données, BD) : mise en oeuvre pratique.

Rentabilité de la RO

- La RO n'est pas gratuite. Par conséquent, la question de la rentabilité se pose tout naturellement.
- Deux exemples empruntés à A. Kaufmann :
  - Transport aérien : 400 vols, 600 liaisons entre 13 villes. Objectif : améliorer le plan des vols. Nombreuses contraintes : temps max de pilotage, temps supplémentaire passé au sol, repos obligatoire, retour périodique des appareils et équipages, maintenance, indemnités de déplacement, etc. Gain de 18%.
  - Usine sidérurgique : 3 chaînes de laminoirs avec une très moderne et une vétuste. Un simple programme linéaire à fait gagner 6% du coût total de production.
- Domaine militaire : coûts de la RO de l'ordre de 1% du budget de l'armée, avec des économies de 5 à 20 fois ce qu'elle coûte.

Bibliographie

- FAURE, Robert, GUILLAT-LE GARFF, Nicole-Sylvie, et BLOCH, Manuel. Précis de recherche opérationnelle. Dunod, 1979.
- ROSEAUX (GROUPE). *Exercices et problèmes résolus de recherche opérationnelle*. Masson, 1986.
- ALJ, Abderrahmane et FAURE, Robert. *Guide de la recherche opérationnelle*. Masson, 1990.
- Paschos, Vangelis Th. *Optimisation combinatoire. 1, Concepts fondamentaux.* Edition Paris : Hermes science, 2005.
- ROADEF, *Le livre blanc de la recherche opérationnelle*, www.roadef.org, 2011.
- TEGHEM, Jacques. Recherche Opérationnelle Tome 1,2. Ellipses, 2012.