Programmation Linéaire
introduction de lp_solve
Sécurité et Aide à la décision

Abdelkader Ouali
abdelkader.ouali@unicaen.fr

Université de Caen Normandie, 14032 Caen, France.

2020

lp_solve

- lp_solve[1] est un logiciel libre (sous licence LGLP) qui permet de modéliser et résoudre des programmes linéaires :
  - en nombres réels,
  - en nombres entiers,
  - en 0/1,
  - ou mixte.

  La licence LGLP permet :
  - d'exécuter le logiciel,
  - d'examiner le code et le fonctionnement du logiciel,
  - de redistribuer le logiciel (y compris une version modifiée).

- lp_solve est basé sur :
  - la méthode simplex révisée, et
  - la méthode Branch-and-bound pour les entiers.
- Autres Solveurs :
  - libre et open source : SCIP, MIPCL, etc.
  - Commercial : CPLEX, GERUBI, etc.

.

[1] http://lpsolve.sourceforge.net/5.5/

lp_solve : algorithmes dans le solveur, ...

- Aucune limite n'est donnée pour la taille du problème,
- Peut lire les formats standards MPS et LP,
- Fournit une API utilisable dans de nombreux langages,
- Implémente Devex, Steepest Edge pour les méthodes de primal et dual simplex,
- Possède des méthodes de réécriture du problème,
- ...

Une communauté via un groupe Yahoo :
http://groups.yahoo.com/group/lp_solve/.

`lp_solve` : Modes d'utilisation

Il existe trois façons d'utiliser `lp_solve`:

1. Exécuter le programme en donnant en argument un fichier contenant le programme linéaire
   - Nous verrons comment écrire un programme en utilisant `.lp`.
2. Appeler le programme au travers de l'API (C, Java, Delphi, Python, ...) ou au travers d'un programme de calcul formel (R, MatLab).
   - Nous verrons comment l'utiliser dans C/C++.
3. Utiliser l'IDE/Modeleur (disponible que sous Windows actuellement).

`lp_solve` : Modélisation

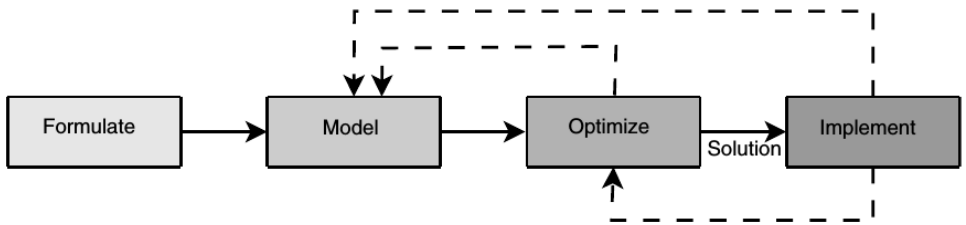

Figure: Schéma de résolution d'un problème réel.

Étant donné une formulation d'un problème à résoudre,
il faudrait déterminer d'abord :

- les **variables** et leurs domaines,
- les **contraintes** définissant les solutions.
- un **objectif** :
  - **minimiser** un coût, ou
  - **maximiser** un gain.

Modéliser un problème linéaire

Un fermier possède deux produits pour nourrir ses bêtes. Le produit A coûte 1,5E et donne 50 g. de protéine et 140 g. de glucide par kg. Le produit B coûte 0,85E et donne 70 g. de protéine et 20 g. de glucide par kg.

Chaque bête requiert 255 g. de protéines et 220 g. de glucides par jour.

Combien de produits doit-on acheter au minimum afin de satisfaire les nécessités en protéines et en glucides pour chaque bête ?

Modéliser un problème linéaire

Un fermier possède deux produits pour nourrir ses bêtes. Le produit A coûte 1,5E et donne 50 g. de protéine et 140 g. de glucide par kg. Le produit B coûte 0,85E et donne 70 g. de protéine et 20 g. de glucide par kg.

Chaque bête requiert 255 g. de protéines et 220 g. de glucides par jour.

Combien de produits doit-on acheter au minimum afin de satisfaire les nécessités en protéines et en glucides pour chaque bête ?

```
min :  1.5 alim_A + 0.85 alim_B;

50 alim_A + 70 alim_B >= 255;

140 alim_A + 20 alim_B >= 220;
```

Format d'un fichier *.lp

<fonction objective >
<contraintes> $*$
<déclaration des variables> $*$

Variables et domaines

Les variables doivent être décrites par des chaines alphanumériques :

- Peuvent contenir des lettres et des chiffres,
- Doivent commencer par une lettre,
- Peuvent contenir des minuscules et des majuscules,
- Peuvent contenir des caractères spéciaux parmi _[]{}/.&#$%@

Il est possible de borner le domaine en ajoutant des "pseudo" contraintes.

- Variable_@A $>$ 1;
- Variable_@A $<$ 10;

Contraintes

Les contraintes sont représentées par des combinaisons linéaires de variables et de constantes. Cette combinaison est suivie d'un comparateur parmi : "<" "<=" "=" ">" ">=" et d'un réel comme membre droit.

- 50 alim_A + 140 alim_B >= 255;
- 50 poids + 1 distance + 10.5 = 2500;

Fonction objectif

La fonction objectif commence par `min` ou `max` suivi de deux-points puis de l'expression arithmétique (du premier degré).

- min : 1.5 alim_A + 0.85 alim_B;
- max : 0.75 X_A + 0.85 X_B + 5;

La fonction peut aussi être laissée vide pour modéliser des problèmes de satisfaction plutôt que d'optimisation.

$$\begin{cases} 4 \times x + 2 \times y = 1; \\ x - 1 \times y < 0; \end{cases}$$

Fonction objectif

La fonction objectif commence par `min` ou `max` suivi de deux-points puis de l'expression arithmétique (du premier degré).

- min : 1.5 alim_A + 0.85 alim_B;
- max : 0.75 X_A + 0.85 X_B + 5;

La fonction peut aussi être laissée vide pour modéliser des problèmes de satisfaction plutôt que d'optimisation.

$$\begin{cases} 4 \times x + 2 \times y = 1; \\ x - 1 \times y < 0; \end{cases}$$

```
min :;
4 x + 2 y = 1;
x + -1 y < 0;
```