Programmation Linéaire
Introduction à l'API lp_solve
Sécurité et Aide à la décision

Abdelkader Ouali
abdelkader.ouali@unicaen.fr

Université de Caen Normandie, 14032 Caen, France.

2020

Utilisation de l'API lp_solve

Liens et page de références

L'API de `lp_solve` est accessible ici :
http://lpsolve.sourceforge.net/5.5/lp_solveAPIreference.htm

Inclusion et compilation

Pour utiliser l'API de `lp_solve` il est nécessaire d'inclure le header de la bibliothèque.

```
#include <lpsolve/lp_lib.h>
```

Et de réaliser le linkage suivant lors de la compilation.

```
g++ src.cpp -llpsolve55 -lcolamd -ldl -o MyExe
```

Créer la structure

Le programme linéaire est stocké dans une structure fournie par la bibliothèque.

```
lprec* lp;
```

Et cette structure doit être initialisée en utilisant la fonction `make_lp`.

```
lprec* lp;
int nbConstraint = 0;
int nbVariable = 2;
lp = make_lp(nbConstraint, nbVariable);
```

http://lpsolve.sourceforge.net/5.5/make_lp.htm

Ajouter une contrainte

Les coefficients des contraintes sont stockées dans des tableaux de REAL. La taille du tableau doit être égale au nombre de variables + 1. La cellule 1 (attention pas 0) correspond à la première variable, la 2 la deuxième variable et ainsi de suite.

`add_constraint` permet d'ajouter une contrainte dans le PL.

Ajout de $1 \times x_1 + 2 \times x_2 \geq 3$.

```
REAL row[nbVariable + 1];
row[1] = 1.0;
row[2] = 2.0;
add_constraint(lp, row, GE, 3.0);
```

Les comparateurs utilisables sont LE ($\leq$), EQ ($=$), et GE ($\geq$).

http://lpsolve.sourceforge.net/5.5/add_constraint.htm

Traduire une contrainte

1. $2x_2 \geq 3$
2. $0x_1 + 2x_2 + 0x_3 \geq 3$
3. $[0, 2, 0] \geq 3$

```
row[1] = 0.0;
row[2] = 2.0;
row[3] = 0.0;
add_constraint(lp, row, GE, 3.0);
```

Ajouter la fonction objectif

De la même façon, la fonction objectif est ajoutée dans un tableau de REAL.

`set_obj_fn` permet d'ajouter la fonction objectif dans le PL.

Ajout de Minimiser $1 \times x_1 + 1 \times x_2$.

```
row[1] = 1.0;
row[2] = 1.0;
set_obj_fn(lp, row);
```

http://lpsolve.sourceforge.net/5.5/set_obj_fn.htm

Par défaut la fonction objectif est une minimisation. Pour passer à une maximisation la fonction `set_maxim` peut être utilisée.

http://lpsolve.sourceforge.net/5.5/set_maxim.htm

Imposer aux variables un type

Afin d'imposer un type à une variable du problème, il est nécessaire d'utiliser une des fonctions suivantes :
http://lpsolve.sourceforge.net/5.5/set_int.htm
http://lpsolve.sourceforge.net/5.5/set_binary.htm

La colonne associée à la première variable est 1, celle de la seconde variable est 2, ...

```
set_int(lp,1,true);
set_binary(lp,2,true);
```

Résoudre le PL

Le PL est résolu en utilisant la fonction `solve`.

http://lpsolve.sourceforge.net/5.5/solve.htm

En cas de succès la valeur de retour sera `OPTIMAL(0)` et en cas d'échec `INFEASIBLE (2)` . La liste de toutes les valeurs de retour est sur la page de la fonction.

```
int ret = solve(lp);
```

Récupérer la solution

La valeur de la fonction objectif peut être récupérée grâce à la fonction `get_objective`.

http://lpsolve.sourceforge.net/5.5/get_objective.htm

et l'instanciation courante avec `get_variables`.

http://lpsolve.sourceforge.net/5.5/get_variables.htm

```
cout << "Objective value = " << get_objective(lp) << endl;
get_variables(lp, row);
cout << "x1=" << row[0] << endl;
cout << "x2=" << row[1] << endl;
```

Exemple

Modèle :

```
max :  15A + 10B;
 2A + 1B <= 24;
 1A + 2B <= 45;
 3A + 1B <= 30;
    int A, B;
```

Exemple

```
REAL row[2 + 1];
```

Exemple

```
REAL row[2 + 1];

row[1] = 2.0;
row[2] = 1.0;
add_constraint(lp, row, LE, 24.0); // 2A + 1B <= 24

row[1] = 1.0;
row[2] = 2.0;
add_constraint(lp, row, LE, 45.0); // 1A + 2B <= 45;

row[1] = 3.0;
row[2] = 1.0;
add_constraint(lp, row, LE, 30.0); // 3A + 1B <= 30;
```

Exemple

```
row[1] = 15.0;
row[2] = 10.0;
set_maxim(lp);
set_obj_fn(lp, row); // max :  15A + 10B;
```

Exemple

```
row[1] = 15.0;
row[2] = 10.0;
set_maxim(lp);
set_obj_fn(lp, row); // max :  15A + 10B;

if (solve(lp) == 0)
{
   cout << "Value = " << get_objective(lp) << endl;
   get_variables(lp, row);
   cout << "A=" << row[0] << " B=" << row[1] << endl;
}
```