

Glpk: GNU Linear Programming Kit

Glpk signifie GNU Linear Programming Kit. Il s'agit d'un ensemble d'algorithmes permettant de résoudre différents problèmes allant de la programmation linéaire (PL) à la programmation linéaire en nombres entiers (PLNE).

Glpk est codé en langage C Ansi et on peut accéder directement aux algorithmes par une API C (Application Program Interface), c'est-à-dire un ensemble de fonctions C. Il est également possible d'accéder aux fonctionnalités de Glpk à partir d'un logiciel, dit solveur, appelé *glpsol*. Glpk peut traiter des programmes linéaires fournis sous différents formats (lp, mps) ou décrits à partir d'un langage spécialisé, appelé modeleur, le *GNU MathProg*.

Ce document résume les fonctionnalités principales du logiciel glpsol ainsi que les formats principaux de données et de description. Pour obtenir des documents sur la bibliothèque C API, installer le logiciel sous linux ou windows, avoir accès à une Faq,... vous pouvez accéder à la page dédiée à glpk sur le site général GNU: <http://www.gnu.org/software/glpk/glpk.html>.

Glpk est une partie du GNU Project et est distribuée en logiciel libre à sources disponibles (free software, open source), sous la licence publique générale GNU qui permet de la redistribuer ou de la modifier selon les termes publiés par la Free Software Foundation.

Algorithmes utilisés

Glpk résout des programmes linéaires (PL) et des programmes linéaires en nombres entiers (PLNE). Ces deux types de problèmes sont très proche en apparence mais demandent à ce jour des traitements bien différents.

Résoudre un programme linéaire peut se faire en temps polynomial. Glpk propose les 2 plus célèbres algorithmes connus: la méthode du simplexe (qui n'est pas polynomiale mais qui est très efficace) et une méthode de points intérieurs (qui est polynomiale).

Résoudre un PLNE est un problème NP-difficile. Glpk utilise le seul algorithme connu capable de résoudre efficacement tous les PLNE: la méthode de branchement (Branch-and-Bound) qui est exponentielle dans le pire des cas.

L'interface API, le logiciel glpsol ou le modeleur GNU MathProg?

Il existe ainsi trois façons d'accéder aux algorithmes de Glpk. Tous lesolveurs linéaires similaires à Glpk possèdent ces 3 accès.

Si l'on désire une utilisation optimale, sécurisée, capable de fonctionner pour des grandes tailles de problèmes, il faut utiliser bien entendu l'interface API. Elle existe à l'origine en C Ansi mais on peut trouver des interfaces externes pour Java, Delphi ou Glpkmex Matlab.

Pour résoudre des programmes de taille plus réduite, le logiciel glpsol est facile d'utilisation. Il prend un fichier d'entrée sous des formats simples et produit un fichier en retour contenant le résultat. Il est ainsi possible de créer un logiciel complet utilisant glpsol, c'est-à-dire un programmant créant le fichier d'entrée pour glpsol et lisant son fichier de sortie.

Parmi les fichiers d'entrée possible, le format modeleur peut être très utile. Un modeleur décrit le PL ou le PLNE en utilisant une description mathématique assez simple. Ce format de fichier peut ainsi être lu par glpsol ou par l'interface API.

Le logiciel glpsol

Si votre système Unix/Linux ou Windows est bien configuré, la commande `glpsol` (sous un terminal Unix/Linux ou sous une fenêtre Dos) vous permet d'accéder au logiciel glpsol.

L'usage de glpsol est `glpsol [options...] nomdefichier`

Avec les options principales suivantes:

- `help`: permet d'accéder à la liste de toutes les options.
- `check`: ne lance pas d'algorithme, sert à tester les données d'entrées
- `cpxlp`: format Cplex lp pour le fichier d'entrée PL ou PLNE `nomdefichier`
- `mps`: idem format mps,
- `glp`: idem format GNU lp
- `math`: idem format modeleur GNU MathProg.
- `wcpxlp fichiersortie`: écrit dans `fichiersortie` le PL ou le PLNE au format Cplex LP (idem avec `-wmps`, `-wglp`)
- `o fichiersortie`: écrit dans `fichiersortie` l'ensemble des résultats dans un fichier "txt"

Il existe également des options spécifiques aux algorithmes utilisés comme par exemple le choix entre algorithme du simplexe et des points intérieurs, de la base initiale pour la méthode de simplexe ou la stratégie de branchement dans le Branch-and-Bound.

Exemples:

`glpsol -check -wmps entree.mps -cpxlp entree.lp`: transfert du format `.lp` au format `.mps`

`glpsol -o solution.txt -cpxlp entree.lp`: résoud `entree.lp` et donne la solution dans `sortie.txt`

Format Cplex lp

Glpk peut donc lire différents formats d'entrée. Le plus simple et le plus répandu a été créé pour le logiciel Cplex (Ilog). Il revient à écrire un PL ou un PLNE sous le format habituel. Voici un petit exemple de programme linéaire sous le format Cplex lp.

```
\Problem name: exemple.lp
Maximize
  obj: 5 x1 + 7 x2
Subject To
  c1: 3 x1 - 2 x2 <= -9
  c2: 10 x1 + 6 x2 >= 11
  c3: 8 x1 + x2 = -6
Bounds
  -Inf <= x1 <= 0
  1 <= x2 <= 4
End
```

Cet exemple se passe de commentaires, mais il faut noter que:

- la première ligne, optionnelle, indique un nom de problème
- les noms de la fonction obj ou des contraintes sont optionnels
- les variables peuvent prendre des noms quelconques
- Si une variable n'a pas borne inférieure dans la section Bounds, elle est considérée positive ou nulle.

Pour les PL mixtes ou entiers, on ajoute une section.

- Integers contenant la liste des variables entières
- Binary contenant la liste des variables binaires

Fichier de sortie

Glpsol peut fournir en sortie un fichier `.txt` donnant la solution du PL ou du PLNE.

Le tableau suivant est le fichier `.txt` correspondant au programme précédent. Il indique le statut de la solution (optimale ou irréalizable) et la valeur de la fonction objective optimale. Dans le cas d'un PLNE, une valeur supplémentaire finissant par (LP) est en fait la valeur du PLNE relaxée des contraintes d'intégrité. .

```
Problem:
Rows:    3
Columns: 2
Non-zeros: 6
Status:  OPTIMAL
```

Objective: obj = 21.75 (MAXimum)

No.	Row name	St	Activity	Lower bound	Upper bound	Marginal
1	c1	B	-11.75			-9
2	c2	B	11.5	11		
3	c3	NS	-6	-6	=	0.625

No.	Column name	St	Activity	Lower bound	Upper bound	Marginal
1	x1	B	-1.25			0
2	x2	NU	4	1	4	6.375

Les deux sections concernent les contraintes/lignes (Rows) et les variables/colonnes (Columns), pour lesquelles sont indiquées:

No le numéro des contraintes et des variables

Row (Column) name le nom des contraintes et de variables

St: le statut de la ligne/colonne: B=en base (pour la variable ou la variable d'écart associée à la contrainte), NF ML NU NS=Hors-base libre, à borne inf, à borne sup ou valeur fixée.

Activity: Valeur d'activité de la ligne/colonne dans la solution: pour les lignes, c'est la somme des variables fois leurs coefficients et pour les variables, c'est leur valeur.

Lower/Upper bound: sans commentaire.

Marginal: pour les variables hors-base, c'est le coût réduit.

Le modeleur GNU MathProg

Le langage-modeleur GNU MathProg est une partie du langage AMPL. Il s'agit d'un langage de description d'un PL sous un format assez naturel. En fait, ce format permet de diviser un PL entre sa structure et ses données. Pour l'utiliser, glpsol nécessite ainsi plusieurs options:

- math: pour indiquer que le fichier est dans ce format
- data fichierdata: indique le fichier des données
- display filename: redirige la sortie vers un fichier

Pour comprendre ce qu'est un modeleur, prenons le célèbre exemple de Dantzig (1963): trouver le coût minimal de transport en respectant les demandes des clients et de l'approvisionnement des usines. La description du problème dans un fichier de format "modeleur" se comprend elle-même:

Fichier Transport_description.math

```
set I;
/* I est l'ensemble des usines de production */

set J;
/* J est l'ensemble des clients */

param a{i in I};
/* a[i] est la capacité de production de l'usine i en nombre de caisses */

param b{j in J};
/* b[j] est la demande du client j en nombre de caisses */

param d{i in I, j in J};
/* d[i,j] est la distance en milliers de kilomètres entre une usine i et un client j */

param f;
/* f est le coût de transport en dollars par caisse et par milliers de kilomètre */
```

```

param c{i in I, j in J} := f * d[i,j] / 1000;
/* c[i,j] est le coût de transport en Kilo-dollars par caisse de l'usine i au client j */

var x{i in I, j in J} >= 0;
/* x[i,j] est une variable de décision égale au nombre de caisses transportées de l'usine i vers le client j */
/* Si on veut les variables x entières, on écrit: var x{i in I, j in J} >= 0, integer; */

minimize cost: sum{i in I, j in J} c[i,j] * x[i,j];
/* Objectif: coût total de transport minimum en kilo-dollars */

s.t. CteCapacite{i in I}: sum{j in J} x[i,j] <= a[i];
/* Contrainte due à la capacité max de production de chaque usine i */

s.t. CteDemande{j in J}: sum{i in I} x[i,j] >= b[j];
/* Contrainte due à la demande de chaque client j */

```

Dans un deuxième fichier, on place les données de l'instance du problème. Notez que les dimensions du problème ne sont pas données dans le fichier de description.

Fichier Transport_pbm_numero1.data

```

data;

set I := Seattle San-Diego;

set J := New-York Chicago Topeka;

param a := Seattle      350
          San-Diego     600;

param b := New-York     325
          Chicago       300
          Topeka        275;

param d :           New-York   Chicago   Topeka :=
          Seattle    2.5        1.7       1.8
          San-Diego  2.5        1.8       1.4 ;

param f := 90;

end;

```

En fait un même fichier de description produira autant de PL ou PLNE qu'il y a de fichiers de données associés. Un fichier modeleur plus un fichier de données permet ainsi à glpsol de générer un PL ou un PLNE. On peut visualiser le résultat par exemple en écrivant le PL ou PLNE au format Cplex lp avec `glpsol -check -wcpxlp Transport_pbm_numero1.lp -data Transport_pbm_numero1.data.lp -math Transport_description.math`

Ce langage est très riche, pour en maîtriser les finesses, le fichier lang.ps, qui est fourni avec les fichiers d'installation, contient une description complète du langage. Mais cette génération par fichier de description reste néanmoins une étape potentiellement coûteuse pour un problème de très grande taille et il est préférable d'utiliser directement l'interface API dans ce cas.