CHAPITRE 1

Modélisation linéaire et outils de résolution

Objectifs : Ce chapitre est une introduction à la programmation linéaire avec les solveurs GLPK et CPLEX. GLPK peut être utilisé dans plusieurs environnements, dont Gusek avec un langage de modélisation associé spécifique, ou Visual Studio en C++ ou encore avec NetBeans en Java. Les différents exemples traités permettent une découverte de ces environnements et de la librairie GLPK. Un exemple d'utilisation de CPLEX avec Visual Studio pour résoudre un programme linéaire est également présenté. Ce chapitre n'est pas un cours de programmation linéaire, mais rappelle les notions fondamentales au travers d'exemples simples. Certains exemples de ce chapitre sont issus de (Guéret et al., 2000) et ont été repris avec l'accord des auteurs.

Outils utilisés :

■ Gusek □ Choco ■ CPLEX PL □ CPLEX CP □ OPL Studio

1.1 Installation de Gusek

1.1.1 Un écosystème "simple"

L'environnement Gusek est un "outil" avec une interface graphique simple, qui implémente une partie du langage AMPL (A Modeling Language for Mathematical Programming) et cette partie définie GMPL (GNU Mathematical Programming Language). Gusek, à l'aide d'un langage de modélisation spécialisé, permet de modéliser des programmes linéaires qui peuvent ensuite être résolus grâce à un solveur linéaire. Gusek utilise le langage de modélisation GMPL qui implémente une partie du langage AMPL.

Dans ce chapitre, le solveur linéaire employé est GLPK : celui-ci est facilement accessible et, de plus, il est proposé sous la forme d'une librairie utilisable à partir d'un langage tiers comme le C++ ou Java.

1.1.2 Gusek (sous Windows et Macintosh)

Gusek est un outil de modélisation linéaire basé sur le solveur GLPK qui est fourni sous la forme d'un projet SourceForge. La page principale de Gusek est à l'adresse suivante :

http://gusek.sourceforge.net/gusek.html

Il suffit de télécharger la dernière version en cliquant sur GUSEK Download .

Gusek se présente sous la forme d'une archive au format .rar qu'il faut décompresser pour obtenir un répertoire nommé Gusek (par défaut) et qui contient (Figure 1-1) entre autres :

- GLPK le solveur linéaire avec lequel Gusek s'interface par défaut (glspsol.exe) ;
- Gusek lui-même, qui est basé sur l'éditeur SciTE (gusek.exe) ;
- des exemples (graphs.pdf) ;
- des documentations (gmpl.pdf, glpk-java.pdf) ;
- des librairies java (glpk-java.jar).

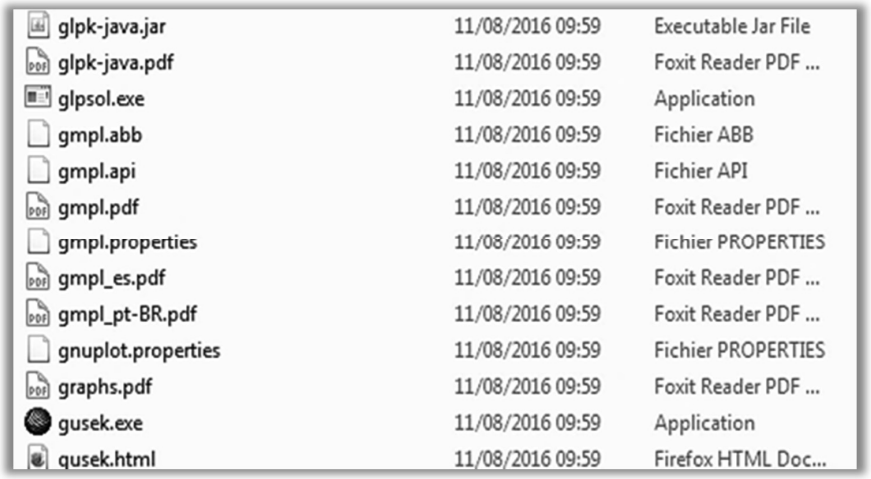

Figure 1-1. Contenu de l'archive gusek.rar

L'environnement démarre avec l'exécutable **gusek.exe** et constitue une "extension" de SciTE qui est un éditeur de texte gratuit et dont le site web est : **http://www.scintilla.org/SciTE.html**.

Cet environnement permet :

- de définir des modèles de programmation linéaire avec un langage simple et convivial ;
- de lancer leur résolution directement, par une simple pression sur un bouton de l'interface qui démarre automatiquement le solveur ;
- de récupérer et d'analyser les résultats.

Pour les utilisateurs de Mac ou Linux, il est possible d'installer Wine un émulateur de Windows afin de profiter de Gusek. Cet émulateur est disponible à l'adresse suivante : **https://www.winehq.org/**.

Wine se servant de la librairie X11 pour l'affichage, il est également recommandé d'installer un client X11 via, par exemple, l'environnement XQuartz (**https://www.xquartz.org/)**.

En fonction des bibliothèques déjà installées sous MacOS, il est possible que des installations supplémentaires soient nécessaires. Le résultat sous MacOS est une exécution légèrement plus lente que sous Windows, mais Gusek reste utilisable au quotidien. La Figure 1-2 présente les deux environnements Gusek, respectivement sous Windows et sous MacOS.

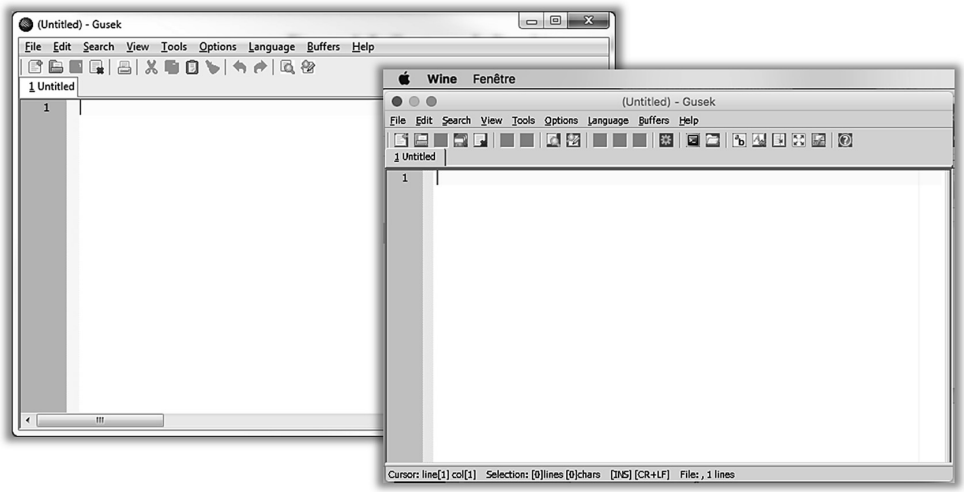

Figure 1-2. L'environnement Gusek sous Windows et sous MacOS

1.1.3 Un premier programme linéaire

Soit le problème d'optimisation suivant qui comprend trois variables et trois contraintes :

Minimiser $f(x_1, x_2, x_3) = 4x_1 + 2x_2 + x_3$

sous les contraintes : $x_1 + x_2 \geq 1.1$ (1)
$x_2 + x_3 \geq 3.2$ (2)
$x_1 + x_3 \geq 4.3$ (3)
$x_i \in \mathbb{R}, i \in \{1,2,3\}$

Cet exemple a été initialement présenté par (Fleury et Lacomme, 2009). L'implémentation avec Gusek comprend cinq parties : la déclaration des variables (mot clé **var**) ; la définition de la fonction objectif (mot clé **minimize**) ; l'écriture des contraintes (mot clé **s.t.**) ; l'appel à la méthode **solve** pour résoudre le modèle ; et la fin du programme (mot clé **end**).

```
var x{1..3}>=0;
minimize f: 4*x[1]+2*x[2]+x[3];
s.t. C1: x[1]+x[2]>=1.1;
s.t. C2: x[2]+x[3]>=3.2;
s.t. C3: x[1]+x[3]>=4.3;
solve;
end;
```

Afin de connaître les valeurs des variables x_1, x_2 et x_3, après la résolution du problème, celles-ci doivent être affichées dans la console. Le code à ajouter dans l'éditeur se situe après l'appel à la méthode **solve** (et avant **end**). La version finale du programme est alors celle de la Figure 1-3. La compilation qui permet de générer un fichier au format PL se fait avec l'icône et l'exécution grâce à l'icône , ou sinon à partir du menu "Tools", puis "Compile" et ensuite "Go".

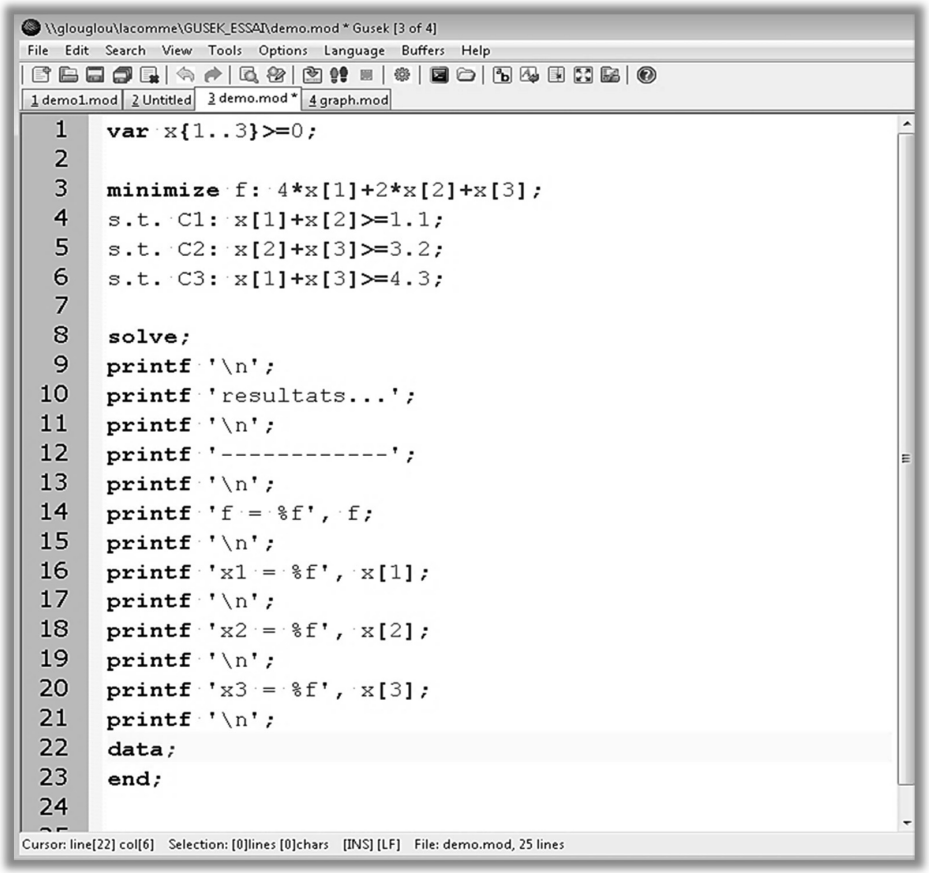

Figure 1-3. Implémentation dans Gusek

La Figure 1-4 présente le résultat de la compilation du modèle avec Gusek qui s'affiche dans la partie de droite de l'éditeur. Il est composé de trois types d'informations :

- des informations liées à la génération du PL (résultat de la compilation) qui sont affichées dans le premier cadre de la Figure 1-4. Cette partie indique si les données ont été lues, puis dans quel ordre les contraintes sont générées, ici la fonction objectif f a été générée en premier, ensuite les contraintes $C1$, puis $C2$ et enfin $C3$;
- des informations liées à la résolution du PL (résultat de la compilation) dans le second cadre de la Figure 1-4, avec notamment la fonction objectif aux différentes itérations, l'état de résolution (ici solution optimale trouvée), et juste en dessous de ce cadre, le temps de résolution (*time used*) et l'espace mémoire utilisé (*memory used*) ;
- des informations liées aux affichages requis par le programme (troisième cadre de la Figure 1-4) et qui correspondent à ceux demandés par le code de la Figure 1-3. Ce cadre donne également l'état du modèle, ici exécuté avec succès (*successfully processed*).

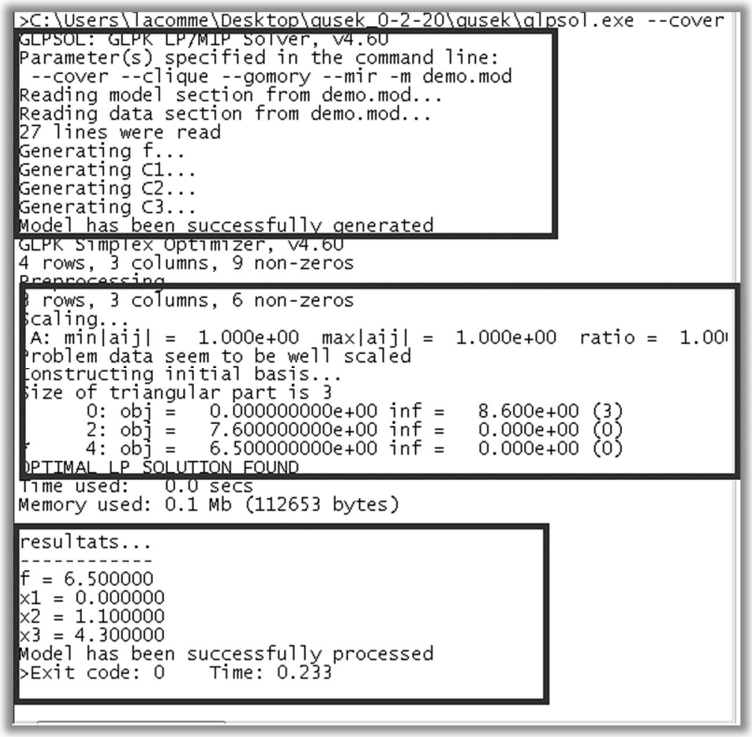

Figure 1-4. Résultats de la compilation et de l'exécution

Le programme linéaire est enrichi en accédant aux variables duales liées aux contraintes, les variables duales permettent de calculer le coût réduit : y_1 est la variable duale de la contrainte $C1$, y_2 de la contrainte $C2$ et y_3 de la contrainte $C3$.

Le coût réduit est la différence entre la valeur de la variable dans la fonction objectif et la somme des variables duales multipliées par la colonne correspondante dans la matrice A. Celle-ci représente les coefficients devant les variables dans chaque contrainte. La première ligne de la matrice A correspond à la contrainte $C1$, la seconde ligne à la contrainte $C2$, etc. La contrainte $C1$ peut s'écrire : $1x_1 + 1x_2 + 0x_3 \geq \cdots$ et donne donc la ligne $(1; 1; 0)$.

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

Le coût réduit pour la contrainte $C1$ donne donc : $\overline{c_1} = c_1 - \sum_{j=1}^{3} y_j . A_{j1}$

Ou encore : $\overline{c_1} = 4 - (2; 0; 1)\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = 4 - 2 - 1 = 1$

Et pour la contrainte $C2$: $\overline{c_2} = c_2 - \sum_{j=1}^{3} y_j . A_{j2}$

Ou encore : $\overline{c_2} = 2 - (2; 0; 1)\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = 2 - 2 = 0$

Et enfin, pour la contrainte 3 : $\overline{c_3} = c_3 - \sum_{j=1}^{3} y_j . A_{j3}$

Ou encore : $\bar{c_3} = 1 - (2; 0; 1)\begin{pmatrix}0\\0\\1\end{pmatrix} = 1 - 1 = 0$

Le code à ajouter pour afficher les valeurs duales est le suivant :

```
printf '\n';
printf 'Dual de C1 : %f', C1.dual;
printf '\n';
printf 'Dual de C2 : %f', C2.dual;
printf '\n';
printf 'Dual de C3 : %f', C3.dual;
printf '\n';
```

```
Time used:   0.0 secs
Memory used: 0.1 Mb (112653 bytes)
Display statement at line 8
x[3].val = 4.3
x[2].val = 1.1
x[1].val = 0

resultats...
------------------
f = 6.500000
x1 = 0.000000
x2 = 1.100000
x3 = 4.300000
------------------
Dual de C1 : 2.000000
Dual de C2 : 0.000000
Dual de C3 : 1.000000
Model has been successfully processed
>Exit code: 0    Time: 0.215
```

Figure 1-5. Affichage des variables duales

Tableau 1-1. Solution du primal

VARIABLE	VALEUR
x_1	0
x_2	1.1
x_3	4.3
y_1	2
y_2	0
y_3	1

L'affichage permet de récupérer alors les valeurs liées aux contraintes ici : $y_1 = 2$, $y_2 = 0$ et $y_3 = 1$ respectivement pour les contraintes $C1$, $C2$ et $C3$. Les valeurs des variables de la solution du problème sont présentées dans le Tableau 1-1.

L'interprétation géométrique des coûts réduits est simple et peut se résumer ainsi : "le coût réduit de la contrainte C_i, donne une information sur le gain potentiel de la fonction objectif en diminuant (car il s'agit ici d'un problème de minimisation) d'une unité la partie droite de la contrainte".

Ceci peut se vérifier facilement sur l'exemple : si la contrainte $C1$ ($x_1 + x_2 \geq 1.1$) est relâchée d'une unité de valeur (contrainte $C1bis$: $x_1 + x_2 \geq 0.1$), alors la fonction objectif décroît de y_1 unités (soit $y_1 = 2$ dans ce cas). La nouvelle contrainte est renommée $C1bis$ et le nouveau programme linéaire est modélisé comme suit :

```
var x{1..3}>=0;
minimize f: 4*x[1]+2*x[2]+x[3];
s.t. C1bis: x[1]+x[2]>=0.1;
s.t. C2: x[2]+x[3]>=3.2;
s.t. C3: x[1]+x[3]>=4.3;
solve;
end;
```

La valeur de la fonction objectif f_2 de ce nouveau programme linéaire est $f_2 = 4.5 = 6.5 - 2 = f_1 - y_1$, avec f_1 la valeur de la fonction objectif du programme linéaire précédent. La nouvelle solution est présentée sur la Figure 1-6.

```
OPTIMAL LP SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (112653 bytes)
Display statement at line 8
x[3].val = 4.3
x[2].val = 0.1
x[1].val = 0

resultats...
------------------
f = 4.500000
x1 = 0.000000
x2 = 0.100000
x3 = 4.300000
------------------
Dual de C1 : 2.000000
Dual de C2 : 0.000000
Dual de C3 : 1.000000
Model has been successfully processed
>Exit code: 0    Time: 0.216
```

Figure 1-6. Solution du PL avec une contrainte C1 relaxée

Si la contrainte (2) ($x_2 + x_3 \geq 3.2$) est relâchée dans le premier programme linéaire (à la place de la contrainte $C1$), alors la nouvelle contrainte est $C2bis$: $x_2 + x_3 \geq 2.2$; et le programme linéaire s'écrit comme suit :

```
var x{1..3}>=0;
minimize f: 4*x[1]+2*x[2]+x[3];
s.t. C1: x[1]+x[2]>=1.1;
s.t. C2bis: x[2]+x[3]>=2.2;
s.t. C3: x[1]+x[3]>=4.3;
solve;
end;
```

La solution de ce programme linéaire est la même que celle du programme linéaire où la contrainte $C2$ n'est pas relâchée, car le coût réduit de $C2$ est nul : $\overline{c_2} = 0$.

Le code ci-dessus est disponible dans la partie consacrée au chapitre 1 avec le lien **Téléchargement de Exemple_1.mod** sur la page web accompagnant ce livre. Le lien de téléchargement correspondant est :

http://fc.isima.fr/~lacomme/PL_PPC/exemples/chp_1/exemple1.mod

1.1.4 Un deuxième programme linéaire et rappels sur le dual

Un second programme linéaire est proposé, le précédent exemple est un problème de minimisation tandis que ce nouveau programme linéaire est un problème de maximisation. Ce nouvel exemple est le dual du programme linéaire de la section précédente (le programme linéaire précédent est donc le primal de celui-ci).

Maximiser $f(y_1, y_2, y_3) = 1.1y_1 + 3.2y_2 + 4.3y_3$

sous les contraintes : $y_1 + y_3 \leq 4.0$ (1)
$y_1 + y_2 \leq 2.0$ (2)
$y_2 + y_3 \leq 1.0$ (3)
$x_i \in \mathbb{R}, i \in \{1,2,3\}$

La version AMPL est la suivante :

```
var y{1..3}>=0;
maximize f: 1.1*y[1]+3.2*y[2]+4.3*y[3];
s.t. C1: y[1]+y[3]<=4.0;
s.t. C2: y[1]+y[2]<=2.0;
s.t. C3: y[2]+y[3]<=1.0;
solve;
end;
```

Puisque ce nouveau programme linéaire est le dual du programme linéaire de la section précédente, la fonction objectif est identique pour les deux problèmes (Figure 1-7). De plus la valeur de chaque variable est égale à la valeur de la duale du primal, et les variables duales des contraintes $\boldsymbol{C1}$, $\boldsymbol{C2}$ et $\boldsymbol{C3}$ donnent les solutions du primal (Tableau 1-2).

```
resultats...
------------------
f = 6.500000
y1 = 2.000000
y2 = 0.000000
y3 = 1.000000
------------------
Dual de C1 : 0.000000
Dual de C2 : 1.100000
Dual de C3 : 4.300000
Model has been successfully processed
>Exit code: 0    Time: 0.218
```

Figure 1-7. Résultats du dual

Tableau 1-2. Solution du dual

VARIABLE	VALEUR
y_1	2
y_2	0
y_3	1
x_1	0
x_2	1.1
x_3	4.3

Le calcul des coûts réduits est de nouveau effectué à titre d'exercice. Par exemple, pour la contrainte $C1$, le coût réduit associé est $\overline{c_1}$ avec $\overline{c_1} = c_1 - \sum_{j=1}^{3} y_j . A_{j1}$, soit : $\overline{c_1} = 1.1 - (0; 1.1; 4.3).\begin{pmatrix}1\\1\\0\end{pmatrix} = 1.1 - 1.1 = 0$

Et pour la contrainte $C2$, le coût réduit associé est : $\overline{c_2} = c_2 - \sum_{j=1}^{3} y_j . A_{j2}$

Ou encore : $\overline{c_2} = 3.2 - (0; 1.1; 4.3).\begin{pmatrix}0\\1\\1\end{pmatrix} = 2 - 1.1 - 4.3 = -2.2$

Et enfin, pour la contrainte $C3$, le coût réduit associé est : $\overline{c_3} = c_3 - \sum_{j=1}^{3} y_j . A_{j3}$

Ou encore : $\overline{c_3} = 4.3 - (0; 1.1; 4.3).\begin{pmatrix}1\\0\\1\end{pmatrix} = 4.3 - 4.3 = 0.$

1.1.5 Génération d'un programme linéaire au format .lp

Deux méthodes permettent de générer automatiquement un modèle Gusek en fichier ".lp".

La première méthode passe par une ligne de commande, tandis que la seconde directement par l'éditeur SciTE. Le fichier généré peut être lu puis résolu par un autre solveur plus puissant que GLPK et la génération d'un fichier ".lp" est également utile pour le débogage.