整数計画ソルバー入門

宮代　隆平

本稿では，これから整数計画ソルバーを使ってみようという人を主な対象にし，多くのソルバーで扱えるLP ファイル形式の文法と，非商用ソルバー SCIP を用いて整数計画問題を解く手順を解説する．また，整数計画ソルバーの現状や，ソルバーに関してよくある質問とそれに対する答えを述べる．

キーワード：整数計画法，ソルバー，LP ファイル，SCIP，NEOS

1. はじめに

整数計画ソルバー（以下，ソルバーと略す）の進歩は非常に速く，数年前には全く解けなかった問題が簡単に解けてしまうことも珍しくない．最近では，商用ソルバーのトライアルライセンスなども充実してきており，ソルバー周りの状況が以前とはかなり変わってきた．

本稿では，ソルバーを使い始めようとする人の「手引き書」として，ソルバーで扱えるファイル形式の文法を説明し，非商用ソルバーを用いて実際に問題を解くまでの手続きを紹介する．また，ソルバーを使った便利な機能や，ソルバーの現状，ソルバーに関してよくある質問について触れる．

なお，本稿の内容についてはできる限り正確を期すが，ソルバーの計算速度に関する記述や各種 URL など，時間が経つと内容が不正確になってしまう部分があると思われる．その点はご容赦いただきたい．

2. 実際に使ってみよう

まずは習うより慣れろということで，実際に整数計画問題をソルバーに入力できる形に表し，非商用ソルバーを使って解いてみることにする．

2.1 LP ファイルの作成

例 1 の混合整数計画問題を考える．この問題をソルバーに入力するために，以下では LP ファイルを作成する．LP ファイルとは，多くのソルバーが扱うことのできるファイル形式の一種で，数理計画問題を記述した，拡張子が.lp のテキストファイルである．ここでは天下り的だが，例 1 下枠内のテキストファイルを準備し，example1.lp と名前をつけて保存してほしい．

例 1　混合整数計画問題

$$
\begin{aligned}
&\text{minimize}\\
&-3x + 4.5y - 2z_1 + f\\
&\text{subject to}\\
&-g_{1,1} + g_{1,2} \leq 5,\\
&3g_{1,1} - 7g_{1,2} + z_2 \geq -10,\\
&2f - g_{1,1} = 6,\\
&x + 0.5y = -4.6,\\
&f \geq 0,\ y \geq 0,\ g_{1,2} \geq 0,\\
&g_{1,1} \in \mathbb{Z},\ g_{1,2} \in \mathbb{Z},\ z_1 \in \{0,1\},\ z_2 \in \{0,1\}.
\end{aligned}
$$

LP ファイル example1.lp

```
minimize
- 3 x + 4.5 y - 2 z(1) + f
subject to
c1:  - g(1,1) + g(1,2) <= 5
c2:  3 g(1,1) - 7 g(1,2) + z(2) >= - 10
c3:  2 f - g(1,1) = 6
c4:  x + 0.5 y = - 4.6
bounds
x free
g(1,1) free
general
g(1,1) g(1,2)
binary
z(1) z(2)
end
```

以上で LP ファイルの準備は終了である．制約式の部分などは，数式をほぼそのまま打ち込んでいるだけのことがわかると思う．example1.lp 中の **minimize**, **subject to**, **bounds**, **free**, **general**, **binary**, **end** は予約語で，例 1 の整数制約，0-1 制約，非負制約などはこれらの予約語を用いて表現されているのだが，ここでは深入りしない（LP ファイルの文法は 3 節で解説する）．

2.2 ソルバーの準備

次に，ソルバーを準備する．本稿では，非商用ソル

みやしろ　りゅうへい
東京農工大学大学院工学研究院
〒 184–8588 東京都小金井市中町 2–24–16

バー SCIP [11] を使うことにする（SCIP の詳細については 4.1 節で述べる）．以下では，Windows 環境におけるインストール手順について述べるが，他の環境でのインストールについてもほぼ同様である．

まずは，SCIP の web ページ [11] へ行き，左側のタブから “download” を選ぶ．2012 年 1 月現在，SCIP の最新バージョンは 2.1.1 である．“SCIP version 2.1.1” の “Binaries” から，“Windows/PC, 32bit, cl15”（あるいは 64bit）を選び，ライセンス条項を読み同意したらダウンロードを行う．なお SCIP の無償使用は，アカデミックユーザーにのみ許可されている．ライセンスの問題で無償使用が不可能な場合は，2.4 節を参考に NEOS サーバーを利用してほしい．

ダウンロードした zip ファイルを解凍すると，SCIP の実行ファイルが現れるので，ダブルクリックで SCIP の起動となる．

2.3 SCIP の操作

2.1 節で作成した LP ファイル `example1.lp` を，SCIP の実行ファイルと同じフォルダ／ディレクトリに置く．次に SCIP を起動させると，以下のようなコマンド入力待ち画面になる．

```
SCIP>
```

以下のコマンドでそれぞれ，`example1.lp` の読み込み，最適化開始，解の出力になる．

```
SCIP> read example1.lp
SCIP> optimize
SCIP> write solution example1.sol
```

最適化計算の途中は，分枝限定法の進行過程を示すログが示されるが，ここではその読み方は省略する[1]．`example1.sol` は解を示したテキストファイルで，次のようになっているはずである．

解を示したファイル `example1.sol`

```
solution status: optimal solution found
objective value:          13.3
z(1)                         1 (obj:-2)
z(2)                         1 (obj:0)
g(1,1)                      -3 (obj:0)
x                         -4.6 (obj:-3)
f                          1.5 (obj:1)
```

[1] コマンド `display display` で，ログに現れる用語の説明を見ることができる．

最初の行が解の状態を示しており，この場合はメッセージ通り最適解が見つかったことを表している．2 行目が目的関数値であり，最適値が 13.3 となっている．3 行目以降は，それぞれの行に変数の名前と対応する値が表示されている．ただし，**解の中で値が 0 となった変数は表示されない**[2]．右端の `(obj: )` は，それぞれの変数の目的関数における係数である．なお，ファイルへの出力ではなく，解を画面に表示させたいだけの場合は `display solution` とすればよい．

`example1.lp` は計算がすぐに終了するが，簡単には解けない整数計画問題も多い．最適化計算の途中で SCIP を停止させるには Ctrl-C を押す．その時点で許容解が求まっていれば，上記と同様に `write` コマンドを使うと最良暫定解を保存できる．再び `optimize` とすれば，中断したところから計算が再開可能である．

`quit` で SCIP を終了，`help` でコマンドのヘルプを表示できる．SCIP のチュートリアルは
http://scip.zib.de/doc/html/SHELL.html
に，SCIP に対するよくある質問は
http://scip.zib.de/doc/html/FAQ.html
に記載されている．

以上，LP ファイルで表された整数計画問題を解き，解を得るまでの手順を示した．見てのとおり，ソルバーは簡単な操作だけで使用できる．

2.4 NEOS サーバー

アカデミック環境ではない場合，SCIP を使用するにはライセンスを購入する必要がある．SCIP 以外の無償ソルバー（GLPK [3]，lp_solve [6] など）を用いるという手もあるが，ここではもう一つの代替手段である NEOS サーバー [9] を利用して SCIP を使ってみよう．

NEOS は，さまざまな最適化ソルバーがインストールされている公開サーバーであり，問題をアップロードすることで，誰でも商用／非商用ソルバーを無償で利用できる．ただし，ソルバーの機能がフルに使えるわけではなく，問題のファイルサイズや計算時間に制限があるが，通常のテストならば問題のない範囲である．

まずは，NEOS の web ページ [9] にアクセスし，“NEOS Solvers” という青いアイコンをクリックする．すると，最適化問題の種類と対応するソルバー，およびそれらが受けつけるファイル形式のリストが表示されるので，“Mixed Integer Linear Programming” から scip の［CPLEX Input］を選ぶ．ファイルをアップロー

[2] 最適解 `example1.sol`から変数 `z(2)`の値を0に変更した解も最適解であるため，環境によっては探索順序が変わり，`example1.sol` が `z(2)` の行を含まない場合もある．

ドするページに移ったら，“SCIP data (CPLEX-LP format file)” から先ほどの LP ファイル `example1.lp` をアップロードしよう．数秒待つと，結果が表示される．結果のページには，多くの統計情報が記載されているが，`example1.sol` と同じ記述がページの中ほどにあるはずである．

難しい問題をアップロードした場合は，しばらく待つとジョブ番号とパスワードが表示される．それらを用いると，管理用ページ[3]から自分の問題の進行状況が確認できる．なお，問題のアップロードの際にメールアドレスを入力しておくと，計算結果をメールで通知してくれるので便利である．

3. LP ファイル

本節では，LP ファイルの文法を解説する．LP ファイルの他にも数理計画問題を記述するファイル形式は複数あるが，文法が平易で可読性が高く，多くのソルバーが対応していることから本稿では LP ファイルを扱う．なお LP ファイルは，凸 2 次目的関数／制約式，Special Ordered Set (SOS), 半連続変数 (semi-continuous variable) などの記述も行うことができるが，ここでは（線形の）整数計画問題を記述するのに十分な文法を示すことにする．

LP ファイルは，目的関数セクション，制約式セクション，上下限セクション，変数型セクション，end 宣言に分かれている．以下ではこれらを順に記述する（適宜，`example1.lp` を参照してほしい）．

【目的関数セクション】

最大化問題の場合は `maximize`, 最小化問題の場合は `minimize` という予約語を最初に書く．続いて，線形の目的関数を書く．演算子と係数，係数と変数，変数と次の演算子の間はスペースで区切る（これは以降すべてのセクションで同じ）．目的関数セクションは複数行にわたっても構わないが，一つの変数名の途中での改行はできない．なお，目的関数に定数項が含まれている場合は，あらかじめ除いておく必要がある．

【制約式セクション】

予約語 `subject to` に続き，線形の制約式を記述する．一つの制約式が複数行にまたがっても構わないが，変数名の途中での改行はできない．また，新しい制約式は新しい行から始める必要がある．変数を含む項は等号／不等号の左辺に，定数項はあらかじめ計算して一つにまとめたものを等号／不等号と同じ行の右辺におく．等号は `=`, 不等号は `>=`, `>`, `<`, `<=` が使えるが，`>` は `>=` と，`<` は `<=` と同じ意味である．制約式の前には，**名前:** という形で制約式に名前をつけられるので，`c1:`, `c2:`, . . . のように通し番号をつけることを勧める．

【上下限セクション】

予約語 `bounds` に続き，それぞれの変数が非負変数なのか自由変数（負の値も取りうる変数）なのかを指定する[4]．ある変数が自由変数の場合には，一行に変数を一つずつ書き，その後ろにそれぞれ予約語 `free` をつける[5]．このセクションで `free` 宣言をされなかった変数は，**自動的に非負変数と定義される**．この点は間違いやすいので，注意が必要である．例えば，ある変数 `x` について，制約式セクションに `x <= - 7` とだけ書いて `free` 宣言をしないと，問題が不能となる．これは，`free` 宣言をしないことにより，`x` が非負変数であると自動的に定義されるため，`x <= - 7` と矛盾するからである．

【変数型セクション】

変数が整数変数の場合は，ここで指定する．予約語は `general` と `binary` があり，前者は変数を整数変数に指定し，後者は 0-1 変数に指定する．どちらの宣言も，一行に複数の変数を並べて書ける（スペースで区切る）．なお，上下限セクションで `free` 宣言していない変数は，`general` 宣言をしても非負の整数変数とみなされる．負の値も取りうる整数変数を定義するには，`free` と `general` の両方の宣言が必要である．

【end 宣言】

LP ファイルの最後の行には，予約語 `end` を置く．

以下は，LP ファイルに関する一般的な注意事項である．

LP ファイルには，いくつかの予約語（`maximize`, `minimize`, `subject to`, `bounds`, `free`, `general`, `binary`, `end` など）があるが，`free` 以外の予約語は

[3] http://www.neos-server.org/neos/admin.html

[4] 自由変数の宣言だけでなく各変数の上限・下限を指定することもできるが，変数の上限・下限は制約式セクションに制約式として書けるので，本稿では `free` の使い方だけを示す．

[5] 後述の変数型セクションと異なり，変数は一行に一つずつしか書けない．

新しい行から始め，その後ろに改行を入れる．また，¥ の後ろから改行までがコメントとして認識される．

LP ファイルでは演算子として +, -, =, <, > を用いるため，これらの文字は変数名には使えない（変数名の途中になら使用できるソルバーもある）．また *, /, [,], ^ も変数名に使用できない（他の意味がある）ことが多いので注意してほしい．

LP ファイルにおいては，例えば x(1) などと書いてもこれに**配列の意味は無く**，単にそういう名前の変数だとして認識される．x)(1))) などという変数も可能である．もちろん，人間にとって変数の意味がわかりやすいように，$x_{1,2}$ は x(1,2), x_1_2 などと書くことが推奨される．

LP ファイル中で係数の 1 や，単項演算子として働く + は省略してもよい．制約式 + 1 x - 1 y = + 1 と x - y = 1 は同じ意味である．

以上，LP ファイルの基本的な文法を解説した．ここまでの段階で，任意の（線形）整数計画問題が記述できる．例 1 の混合整数計画問題では f と y が非負変数，x が自由変数，$g_{1,2}$ が非負の整数変数，$g_{1,1}$ が整数変数で非負制約が無いもの（自由変数＋整数制約），z_1 と z_2 が 0-1 変数であったが，example1.lp においてきちんと対応していることを確認されたい．

4. ソルバーについて

本節では，非商用ソルバー SCIP と，SCIP を用いた便利な機能を紹介する．また，ソルバーの現状について概説する．

4.1 SCIP とは

SCIP [11] は，ドイツの Zuse Institute Berlin で開発されている非商用ソルバーである．数ある非商用ソルバーの中で，本稿で SCIP を取り上げた理由は以下のとおりである．

- **非商用ソルバーとしては最も高速なものの一つ**（いくつかの商用ソルバーよりも高速）．
- バイナリ版が準備されており，コンパイルせずにすぐ使える．ソースコードも全て公開されている．
- コマンドラインインターフェースがシンプルで使いやすい．
- LP ファイルに対応しているため，商用ソルバーへの切り替えが容易である．
- アカデミックユースは無償，また NEOS サーバーでも利用できる．

非商用ソルバーでは，古くは lp_solve [6], 少し前までは GLPK [3] がよく用いられていたが，2012 年現在，非商用ソルバーで計算速度を求めるなら，迷わず SCIP を使うべきである．

4.2 SCIP を用いた便利な機能

複数の問題を解く際には，バッチ処理ができると便利である．以下では，OS のリダイレクト機能を利用したバッチ処理の方法を述べる．

SCIP で，example2.lp と example3.lp を解く場合，以下のテキストファイルを準備する．

バッチ処理のためのファイル script.txt

```
read example2.lp
optimize
write solution example2.sol
read example3.lp
optimize
write solution example3.sol
quit
```

上記を記載したテキストファイルを script.txt, SCIP の実行ファイル名を scip.exe だとし，これらと LP ファイルを同じディレクトリに置いて，OS のコンソール（Windows の場合コマンドプロンプト）から

```
scip.exe < script.txt
```

と実行することで，script.txt の中身を SCIP 上で順に自動実行してくれる．計算終了後，example2.sol と example3.sol に結果が残る．script.txt を編集することで，例えば問題ごとに SCIP のオプションを変えることも可能である．

目的によっては，最適解を一つ求めるのではなく，「（整数変数の値が異なる）最適解の個数を数えたい」という場合がある．そのような場合は，まず最適解を求め，解き終わった後に元問題に「目的関数 = 最適値」という制約式を追加した新しい問題を作る．そして新しい LP ファイルを SCIP で read した後，count というコマンドを使えばよい．前述の example1.lp の場合は，最適値が 13.3 であるので制約式セクションに

```
c5:  - 3 x + 4.5 y - 2 z(1) + f = 13.3
```

と追加し，問題を読み込み直した後 count とすると，

```
Feasible Solutions:  2 (0 non-trivial ...
```


と表示されるので，最適解は 2 個存在することがわかる．また，最適解の個数を数えるだけでなく最適解そのものをすべて列挙する，あるいは許容解の個数を求める／列挙を行うことも可能である[6]．しかし，いずれの操作も計算に時間がかかり，また問題によっては解の個数が容易に非現実的な多さになるので，使う際には注意が必要である．

以下は，LP ファイルの変換に関する便利な機能である．LP ファイルの 1 行は，ソルバーによって異なるが，ある程度の長さ以内（255 文字など）で改行しなければならない．このコントロールが面倒な場合は，LP ファイル生成の際に変数一つごとに改行してしまい，ファイルを一度 SCIP に読み込ませて別名の LP ファイルで保存すると，1 行 100 文字程度できれいに整形してくれる．

LP ファイルの整形

```
SCIP> read example4.lp
SCIP> write problem example5.lp
```

また，インターネットにアップロードされているベンチマーク問題などは，LP ファイルではなく MPS ファイル[7] という形式になっていることが多い．MPS ファイルもテキストファイルだが，そのままでは可読性が低い．これも SCIP を用いると相互に変換が可能である．

LP ファイルと MPS ファイルの変換

```
SCIP> read example6.mps
SCIP> write problem example6.lp
SCIP> read example7.lp
SCIP> write problem example7.mps
```

本稿で紹介した LP ファイルの文法は，正確には CPLEX 形式の LP ファイルと呼ばれる文法[8]のサブセットである．LP ファイルの文法は各ソルバーによって独自に拡張されている場合があり，本稿で紹介した LP ファイルを扱えないソルバーもあり得る．そのような時には，SCIP で MPS ファイルに変換して試してみるとよい（非商用ソルバー GLPK [3]，lp_solve [6] などでも同様の変換が可能[9]）．

4.3 ソルバーの現状

2012 年初頭において，整数計画問題を扱うソルバーは商用／非商用を含め多数あり，サーベイ [2] に記載されているだけでも 40 本以上が存在する．ソルバーを使用する際には，以下のような観点から検討して，目的に合ったものを選ぶとよい．

- 価格およびライセンス形態（商用／非商用，アカデミック／トライアルライセンスの有無，…）
- インターフェース（CLI, GUI, モデリングツール，API, …）
- 扱える問題のフォーマット（LP ファイル，MPS ファイル，…）
- 扱える問題の種類（LP, IP, QP, QCP, QCQIP, 非線形, …）
- ソルバーの速度
- マルチコア上の並列分枝限定法に対応しているか

商用ソルバーの通常ライセンスは数十万〜数百万円程度のものが大多数だが，アカデミックライセンス（無償〜数十万円程度）が用意されているソルバーも多い．また，最近は商用ソルバーにおいてもトライアルライセンスの提供が珍しくなく，そのようなソルバーではライセンスを購入する前の試用が可能になっている．

一般的には，非商用ソルバーより商用ソルバーのほうが高速だが，商用ソルバーの中にもかなりの速度差があるのが事実である．各種の最適化ソルバーのベンチマーク実験結果を継続的に公開している Mittelmann の web ページ [7] によると，2012 年初頭時点での整数計画ソルバーは，CPLEX [5]，Gurobi [4]，XPRESS [1]（アルファベット順，いずれも商用）が計算速度の意味で三強のようである．

各種のソルバーは，バージョンが上がると性能が大幅に向上することが多い（例えば，商用ソルバー CPLEX 7.1/8.1/9.1/10.1/11.1 のベンチマーク結果 [8] を参照）．可能な限り，最新版を使用することを推奨する．

最近では CPU がマルチコア化しているため，高速な商用ソルバーはたいていが並列分枝限定法に対応している．難しい問題を解く際にはソルバーの対応状況も考慮するとよいだろう．

[6] http://scip.zib.de/doc/html/COUNTER.html を参照．
[7] MPS ファイルの文法は，下記の URL[6] などを参照．
http://lpsolve.sourceforge.net/5.5/mps-format.htm
[8] 文法は下記の URL[6] などを参照．
http://lpsolve.sourceforge.net/5.5/CPLEX-format.htm
[9] GLPK[3] では `glpk.exe --check --lp example8.lp --wfreemps example8.mps` とすると LP ファイルから MPS ファイルに変換できる．

5. よくある質問

本節では，ソルバーに関してよくある質問をあげ，それに対するコメントを記載する．

質問

大きな問題の LP ファイルを生成するにはどうしたらよいか？

LP ファイルには，プログラミング言語の配列のように，変数をまとめて扱う機能はない．つまり，$\sum_{i=1}^{100} x_i \leq 3$ を意味したい場合には，

```
x(1) + x(2) + x(3) + 中略 + x(100) <= 3
```

と左辺項を 100 個ベタに，適度に改行しながら書くしかない．よって，大きなデータの問題を扱う場合には手作業ではなく何らかの機械的な生成手段が必要になる．以下の三つが主な方法である．

1. プログラミング言語で LP ファイルを生成する．
2. 数理計画モデリングツールを使う．
3. ソルバーに準備されている API などで，ダイレクトに問題生成および求解を行う．

最初の方法は，使い慣れているプログラミング言語でテキストを出力させればよいので，一番お手軽である．また，問題を LP ファイルを介して扱うため，あるソルバーから他のソルバーに乗り換えることも容易である．ただしあくまでも LP ファイルベースであるため，他の方法と比較するとソルバーの複雑な制御が難しいというデメリットもあるが，問題を独立に解くだけならばこの方法でもほとんど不便はない．

2 番目の方法のメリットは，数式で表現されたモデルからプログラムが作りやすいということである．例えば，あるモデリング言語では $\sum_{i=1}^{100} x_i \leq 3$ を表現するのに `sum(i in 1..100)(x[i]) <= 3` と書けばよいなど，数式からプログラムへ直感的に書き換えが可能である．他にも，よく使うタイプの制約式に対応する予約語が準備されているなど，数学的定式化が完成してからの変換作業は非常に高速に行える．ただし，数理計画モデリングツールを購入し，そこで使用されているモデリング言語を習得する手間がかかる．汎用のプログラミング言語を覚えるよりは容易だが，ある程度の時間的・金銭的な初期投資が必要である．

3 番目の方法は，ソルバー内部での分枝順序の操作をはじめ，分枝限定法の高度な制御が可能になるのが一番のメリットである．また列生成法など，複数の問題間で情報をやりとりする必要がある場合には，この方法を使用するのが効率的である．ただし API の書式などはソルバーごとに異なるため，別ソルバーへの流用は難しい．

表 1　巨大な 0-1 整数計画問題で簡単なもの [10]

問題名	0-1 変数の個数	制約式の本数
neos-885524	91670	65
rail01	117527	46843
n3seq24	119856	6044
netdiversion	129180	119589

表 2　小さな 0-1 整数計画問題で未解決のもの [10]

問題名	0-1 変数の個数	制約式の本数
sts405	405	27270
sts729	729	88452
queens-30	900	960
neos-807456	1635	840

質問

どのくらいのサイズの問題が，どのくらいの時間で解けるのか？ この問題は何時間で解けるか？

これらの質問に対するもっとも正確な答えは，「問題の形および数値データに強く依存するので，解いてみるまでわからない」である．しばしば難易度の指標とされる変数の個数や制約式の本数は，あくまでも目安にすぎないので，注意されたい．表 1 は，整数計画問題の世界標準的なベンチマークサイト MIPLIB 2010 [10] から，「商用ソルバーで 1 時間以内に解ける問題」のうち，0-1 整数計画問題で変数が多い順に 4 問をリストアップした表である．また，表 2 は同じく [10] から「最適解がいまだに不明な問題」のうち，0-1 整数計画問題で変数が少ない順に 4 問をリストアップした表である．10 万変数，10 万制約式で簡単に解ける問題がある一方，1000 変数未満の小さな問題でも未解決のものがあり，変数の個数や制約式の本数だけでは安易に計算時間を決められないことがわかる．

しかし「問題依存」の一言ではひどいので，異論のある方も多いと思うが，筆者は最近では以下のように答えることにしている．これはあくまでも目安であり，繰り返しになるが，**数百変数でも全く解けない問題も存在する**ことに注意しよう．

◇ 1000 変数以下：悩む前に実行してみる．
◇ 数千変数程度：解けることも多い．

◇ 10000〜数万変数程度：解けないことが多い．
◇ 数万変数以上：解きやすい性質の問題のみ解ける．

なお，（最適性の証明は不要で）質の良い解を求めれば十分な場合には，問題の性質にもよるが，一回り大きなサイズのものが扱える．

質問

非商用のソルバーで（所望の時間内に）解けなかったが，商用ソルバーを買えば解けるか？ 商用ソルバーは非商用ソルバーの何倍高速か？

非商用ソルバーと高速な商用ソルバーとの速度差は，**問題が難しくなるごとに大きくなる**．問題サイズが小さな時に非商用ソルバーで 20 秒，高速な商用ソルバーで 5 秒で解けたとしても，サイズが大きくなったときに数時間と数分になってしまうことは珍しくない．本稿で紹介した SCIP も，2012 年現在で最高速の非商用ソルバーの一つであるとはいえ，高速な商用ソルバーとはまだかなりの速度差がある．一方で，高速な商用ソルバーでも歯が立たない問題もそこら中にあるため，商用ソルバーを買えば何でも解決というわけではないのが残念なところである．

結局，「商用ソルバーに手を出すべきなのか？」と迷った場合には，各ソルバーのトライアルライセンスなどを利用し，購入前に十分テストするのが得策だと思われる．また，2.4 節で紹介した NEOS サーバーでも，いくつかの高速な商用ソルバーが使えるため，有効に活用するとよい[10]．

6. おわりに

筆者は 2006 年に，とある整数計画問題[11]を 40 日間かけて何とか解いた経験がある．この記事を書くにあたり，同じ問題を最新の 環境で解き直してみたところ，たったの 5 時間で計算が終わり唖然としてしまった．「ソルバーの進歩は著しい」と自分でも言ってはいるものの，改めてその凄さを思い知った次第である．これからソルバーを使おうと思っている方だけでなく，以前に使ってみたけれど……という方も，最新のソルバーを試す価値は大いにあると思われる．

本稿で見たように，ソルバーを使い始めるのは非常に簡単である．研究開発だけにとどまらず，教育目的あるいは日常生活においても活用していただき，整数計画法のパワーを実感していただければ嬉しい．

参考文献

[1] FICO, FICO Xpress, http://www.msi-jp.com/xpress/
[2] R. Fourer, “Software Survey: Linear Programming,” *OR/MS Today*, **38** (2011). http://www.informs.org/ORMS-Today/Public-Articles/June-Volume-38-Number-3/Software-Survey-Linear-Programming http://www.orms-today.org/surveys/LP/LP-survey.html
[3] GNU Project, GLPK (GNU linear programming kit), http://www.gnu.org/software/glpk/glpk.html
[4] Gurobi Optimization, Gurobi Optimizer, http://www.gurobi.com/ http://www.octobersky.jp/products/gurobi/gurobi.html
[5] IBM, IBM ILOG CPLEX, http://www-06.ibm.com/software/jp/websphere/ilog/optimization/core-products-technologies/cplex/
[6] lp_solve, http://lpsolve.sourceforge.net/
[7] H. Mittelmann, Benchmarks for Optimization Software, http://plato.asu.edu/bench.html
[8] 宮代隆平，「ここまで解ける整数計画―近年の発展―」，第 20 回 RAMP シンポジウム論文集，日本オペレーションズ・リサーチ学会，2008, 1–21.
[9] NEOS Server for Optimization, http://www.neos-server.org/neos/
[10] Zuse Institute Berlin, MIPLIB 2010, http://miplib.zib.de/miplib2010.php
[11] Zuse Institute Berlin, SCIP (Solving Constraint Integer Programs), http://scip.zib.de/

[10] 例えば，MPS ファイルに変換すれば NEOS サーバーで商用ソルバー Gurobi [4] を使うことができる．
[11] MIPLIB 2010 [10] の `go19` に制約式を数本追加した問題．0-1 変数 441 個だけからなる小さな問題であるが，2006 年の時点では非常に難しかった．