データ分析ライブラリーを用いた最適化モデルの作り方

斎藤　努

ORに関連するPythonの有用なライブラリーを紹介する．これらは，単独で使っても役に立つが，組み合わせることで相乗効果をもたらす．本稿では，データ分析と最適化のライブラリーを組み合わせて，最適化モデルを作成する方法を紹介する．単独で使った場合と比較することでその有用性を示す．

キーワード：データ分析，最適化，モデル，Python，Jupyter，pandas，NumPy，PuLP

1. はじめに

データ分析のツールとしてPythonが使われるようになってきた．しかし，最適化のツールとしてPythonを使っている人はまだ少ない．最適化は，図1のようにさまざまな分野で使われている．近年，計算速度の向上により，以前は解けなかった問題も解けるようになってきた．

「ORを探せ！」ポスター[1]では，身のまわりにあるいろいろなORをわかりやすく紹介している．このような幅広いORの研究者の中においてもPythonを使う人が増えてきていることが感じられる．

本稿では，Pythonを使うメリットと，最適化モデルにデータ分析ライブラリーが役に立つことを紹介する．

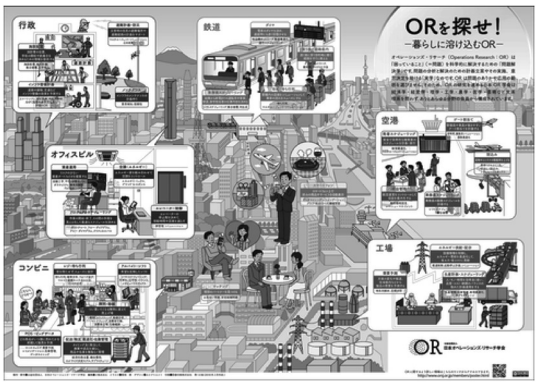

図1　「ORを探せ！」ポスター

2. PythonとOR

Pythonには，科学技術計算向けのライブラリーが数多くある．オペレーションズ・リサーチ関連のライブラリーも多数ある．ここでは，表1のライブラリーを簡単に紹介する．これらは，macOS，LinuxやWindowsなどの各種OSで実行できる．また，標準的なPCでも数十万変数の最適化も可能である．

表1　紹介するライブラリー

名称	説明
Jupyter	Jupyter Notebookによる可視化や試行錯誤に適した実行環境
NumPy	効率的な多次元配列や数学の関数
pandas	表などのデータ形式をサポートするデータ分析の関数
PuLP	数理最適化のモデルの作成
NetworkX	グラフ関連の機能
ortoolpy	数理最適化のいろいろな問題の解法など
dual	双対問題作成

これらは無料で利用できる．具体的な使い方については後述する．Pythonをインストール済みであれば，コマンドプロンプトで以下のようにしてライブラリーをインストールできる[2]．

```
pip install jupyter pandas pulp \
    networkx ortoolpy dual
```

さいとう　つとむ
株式会社ビープラウド
〒151–0051 東京都渋谷区千駄ヶ谷5–32–7 南新宿ビル4F
https://www.beproud.jp/

[1] 「ORを探せ！」ポスター
http://www.orsj.or.jp/members/poster.html

[2] macでは，`pip`ではなく`pip3`を使うこと．Linuxで管理者権限を付ける場合は`pip`の代わりに`sudo pip`を使うこと．ユーザー環境にインストールする場合は`--user`オプションをつけること．Windowsで`pip`が使えない場合は代わりに`py -3 -m pip`が使える可能性がある．また，pandasをインストールすればNumPyも入る．

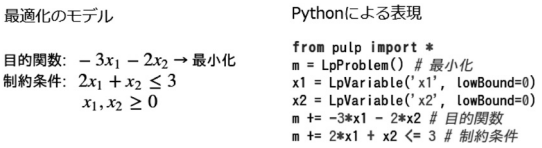

図 2　簡易な記述

Python のインストールは，Python.jp[3]の環境構築ガイドを参考にされたい．ここでは，Python3.7 を想定している．

最適化などの活用で，Python を使うメリットを挙げよう．

- **学習しやすい**：予約語も少なく，文法がシンプルなので，覚えやすい．
- **簡易な記述でわかりやすい**：定式化と同じような形で最適化モデルを作成できる（図 2）．
- **Python に対応した最適化ソフトウェア**：有料，無料含めて多数の最適化ソフトウェアがあるが，Python から利用できるものがいろいろある．
- **多数のライブラリー**：パッケージコミュニティサイト https://pypi.org/ だけでも約 15 万ものパッケージが公開されている．これらのライブラリーを組み合わせれば，下記のようなことも Python だけで簡単にできる．
  - ・インターネットからデータをスクレイピング
  - ・分析フレームワークの利用
  - ・データの読込，加工
  - ・データ分析
  - ・機械学習の分類や回帰などの処理
  - ・画像データの分析
  - ・グラフやネットワーク構造の分析
  - ・最適化のモデル作成や実行
  - ・結果の可視化

本稿では，データ分析と最適化のライブラリーを組み合わせて，最適化モデルを作成する方法を紹介する．

3.　実行環境の使い方

Jupyter Notebook [1] の使い方を紹介する．Jupyter Notebook はノートブックというファイルを扱う．分析での試行錯誤や可視化による確認に便利である．以下のようなことができる．なお，本稿のコードは，Jupyter Notebook での実行を想定している．

- **セルによる管理**：ノートブックは，セルと呼ばれる単位で構成される．セルは，コードやマークダウンなどを入力できる．マークダウンでは，画像，動画や TeX など多彩な要素を扱える．
- **コードの実行**：セルのコードは，いつでもどこでも実行でき，結果もノートブックに残せる．ファイルに保存されたノートブックを，別の環境に移して実行することもできる．

起動方法

コマンドプロンプトで，`jupyter notebook` と入力する．カーネルサーバーが起動し，ローカルであれば，ブラウザーも起動する．操作はブラウザー上で行う．

新規ノートブック作成

`New` ボタンの `Python3` を選ぶ．

3.1　使い方

ノートブックでは，セルを選択すると**編集モード**と**コマンドモード**のどちらかになる．入力は，セル内にカーソルが表示される編集モードで行う．編集モードとコマンドモードの切り替えは，Esc キーと Enter キーで行う．

セルには，コード，マークダウンや Raw NBConvert などの種類がある．セルを選択して，上部のプルダウンメニューやショートカットキーで切り替えられる．筆者は，ショートカットキー操作をおすすめする．下記にコマンドモードでよく使うものを挙げる．

- ・h キー：ショートカットの表示
- ・y，m，r キー：セルの種類をそれぞれ Code，マークダウン，Raw NBConvert に切替
- ・スペース，Shift+ スペースキー：それぞれ下，上へのスクロール
- ・a，b キー：それぞれ上，下へ新規セルの挿入
- ・x，c，v，z キー：それぞれセルのカット，コピー，ペースト，アンドゥ
- ・M キー：下のセルとの結合
- ・l（エル）キー：行番号の表示／非表示切替
- ・s キー：ノートブックをファイルに保存

コードを入力したら，Shift+Enter キーで実行できる．コード入力中は，タブキーで補完できる．関数にカーソルを置いてタブキーを続けて 2 回押すと，その関数のヘルプを確認できる．

Jupyter Notebook を終了したい場合は，保存してから，ブラウザーを閉じ，起動時のコマンドプロンプトで，Control+C キーを 2 回押してプロセスを終了させればよい．

マジックコマンド

「%」や「%%」で始まるものをそれぞれ，ラインマジックコマンド，セルマジックコマンドと言い，行やセルに対して有効となる．よく使うマジックコマンド

[3] https://www.python.jp/

を挙げる．

・%whos：変数一覧表示
・%time：時間計測
・%timeit：複数回の実行による精度の高い時間計測
・%matplotlib inline：グラフなどをブラウザー内に出力する設定

PDF へ保存

PDF ファイルとして保存する方法の一つとして，印刷時に PDF への保存を選ぶ方法がある．

4. データ分析ライブラリーの使い方

pandas の使い方を紹介する．主なデータ構造として，DataFrame と Series がある．それぞれ表と列に対応する．この DataFrame を使って最適化モデルで使う**変数表**と呼ぶものを作成する．

変数表とは

・変数の列をもつ．
・基本的に 1 行 1 変数である．
・変数の属性は，対応する行によって表される．それにより，$x_{i,j,k}$ のような一見しただけでは，どういう属性をもっているかわからない変数の代わりに，わかりやすい形で変数の属性が参照できる．
・pandas の豊富な機能を使って，簡潔に最適化のモデルを作成できる．
・pandas のベースの NumPy を使って，効率よくモデルを作成できる．
・ソルバーで解いた結果も表に入れることにより，結果の加工も簡単にできる．

pandas と最適化ライブラリー PuLP を組み合わせることで相乗効果が得られる．pandas には，膨大な機能があるが，本稿では一部を簡単に紹介する．

データの作成

メモリから表や列を作成できるが，多くの場合，ファイルやデータベースなどから作成することになる．以下のように CSV ファイル (`'sample.csv'`) から読み込める．

```
import numpy as np, pandas as pd
df = pd.read_csv('sample.csv')
df
```

	A	B	C
0	…	…	…
1	…	…	…

Jupyter Notebook では，DataFrame (`df`) を評価すると，整形された表として出力される．

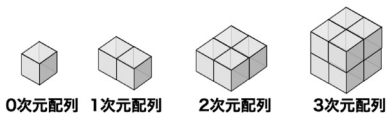

図 3　多次元配列

左の列の `0, 1` を**行ラベル**，上の行の `A, B, C` を**列ラベル**という．行ラベルは `index` で，列ラベルは `column` でアクセスできる．ラベルとは別に 0 から始まる通し番号を利用できる．それぞれ**行番号**，**列番号**という．

NumPy の多次元配列

NumPy は多次元配列を扱うライブラリーであり，Python で広く利用されている（pandas のベースになっている）．pandas の操作方法の多くは NumPy 由来なので，NumPy を理解すると，pandas の理解も深まる．多次元配列とは，図 3 のような，さまざまな次元の配列である．

0 次元配列を**スカラー**，1 次元配列を**ベクトル**，2 次元配列を行列，3 次元以上の配列を**テンソル**という．Python のリストとは異なり，要素をすべて同じ型にすることにより，無駄な型変換をなくし高速に計算できるよう工夫されている．

列の追加

```
df['Var'] = None
```

新たに `Var` という列を作成し，右辺で初期化する．すでに存在していれば上書きとなる．右辺をスカラーにすると，ブロードキャスト（後述）により各要素として設定する．DataFrame のメンバーは `T` を除き小文字で始まるので，追加する列名を大文字で始めるとよい．そうすれば，`df.Var` のようにアクセスできる．

インデックス参照

`df.iloc[i, j]` で `i` 行目，`j` 列目の要素となる．`i, j` は番号である．`df.loc[i, j]` で行ラベル `i`，列ラベル `j` の要素となる．どちらも各次元でスライス[4]が利用できる．

ファンシーインデックス参照

`df[['A', 'B']]` のように列ラベルのリストで当該列の表を取得できる．

ブールインデックス参照

`df[[False, True]]` のようにインデックスとして**行数分のブール値リスト**を指定すると，True の行だけ抜き出せる．条件で抽出する場合によく使う．

[4] `a:b` と指定すると，`a` から `b` の前までとなる．

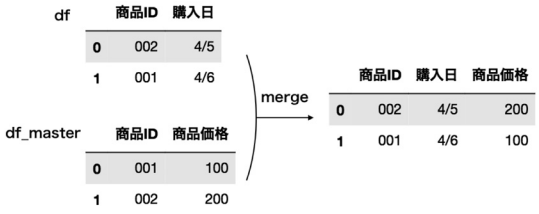

図 4　merge 関数

ブロードキャスト

形状が異なる多次元配列同士の演算を可能にするしくみを**ブロードキャスト**と言う．特定の次元の要素数が 1 の場合，同じ値を使いまわすことにより，簡易な記述と高速な演算を可能にする．スカラーとの演算では要素ごとに演算する．

条件抽出

ブールインデックスを組み合わせて複雑な条件を指定できる．たとえば，A 列が 2 または B 列が 5 の行の抽出は `df[(df.A==2) | (df.B==5)]` と記述できる．

ユニバーサル関数

DataFrame や Series を対象として，要素ごとに演算し元と同じ形状で返す関数を**ユニバーサル関数**と言う．たとえば，`df.abs()` とすると，全要素を絶対値に変換した DataFrame が作成される．

便利な関数 (groupby)

関数 `groupby` で指定した列（複数列可）の値が同じ行をグルーピングする．グルーピングの結果は，ループでも直接でも使える．直接使う場合は，取り出す値を関数[5]で決める必要がある．

便利な関数 (merge)

関数 `merge` で二つの表を結合できる．たとえば，`df` の商品 ID に対応する商品価格の列を追加したいとき，両方の列をもつ `df_master` を使って，`df.merge(df_master, on='商品 ID')` とする（図 4）．

グラフ描画

pandas では，たとえば，下記のように簡単にグラフ描画ができる．

```
%matplotlib inline
df.plot()
```

ベースは matplotlib という描画ライブラリーを用いている．pandas での描画は，matplotlib よりシンプルにできる．

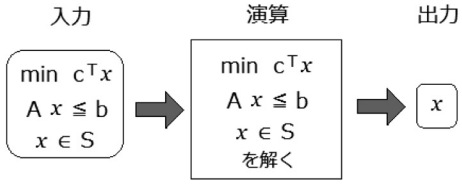

図 5　ソルバーの入出力

4.1　データ分析ライブラリーのまとめ

- pandas ではいろいろな作業を簡単に記述できる．書籍などで一通り把握するとよいだろう．
- pandas のベースは NumPy であり，NumPy の知識は pandas で役に立つ．
- pandas の描画は，matplotlib ベースなので，複雑な描画をしたい場合は，matplotlib の知識が役に立つ．
- pandas の特徴は，さまざまなデータ処理をシンプルに扱えることである．最適化モデルの作成も一種のデータ処理であるため，pandas を使うことで**簡単にかつわかりやすく最適化モデルを作成**できる．

5.　最適化ライブラリーの使い方

いろいろな最適化ライブラリーがあるが，ここでは PuLP の使い方を紹介する．PuLP では線形最適化と混合整数最適化を扱える．ほかにも非線形最適化を扱える OpenOpt やグラフ理論を扱える NetworkX などがある．

最適化問題を解くためには，以下のステップを行う．

- 最適化のモデルを作成する
- ソルバーを呼び出して解を得る

ソルバーは，最適化モデルを入力とし，モデルを解いて，変数の値（解）を出力とするソフトウェアである（図 5）．

PuLP は数理モデルを作成するためのソフトウェア（モデラー）であり，COIN プロジェクトで開発された．PuLP では，ソルバーとして CBC，Gurobi，GLPK などいろいろなものが使える．PuLP をインストールすると CBC も一緒にインストールされる[6]．

以下，PuLP の利用方法を簡単に紹介する．

ライブラリーのインポート

```
from pulp import (
    LpProblem, LpMaximize, LpStatus,
    LpVariable, lpDot, lpSum, value)
```

[5] `mean` や `first` など

[6] ソルバーを指定しないと CBC が用いられる．

数理モデルの作成

最小化問題のとき

```
m = LpPrblem()
```

最大化問題のとき

```
m = LpProblem(sense=LpMaximize)
```

変数の作成

連続変数（**非負変数**）：0 以上の連続変数

```
x = LpVariable(変数名, lowBound=0)
```

連続変数（**自由変数**）：任意の連続変数（負も OK）

```
x = LpVariable(変数名)
```

0-1 変数：0 または 1 の**バイナリー変数**

```
x = LpVariable(変数名, cat=LpBinary)
```

連続変数のリスト

```
x = [LpVariable(変数_i, lowBound=0)
              for i in range(個数)]
```

0-1 変数のリスト

```
x = [LpVariable(変数_i,cat=LpBinary)
              for i in range(個数)]
```

重要：変数名は必ず異なるようにしなければならない.

下記の ortoolpy ライブラリーの関数を使うと，変数名を指定することなく，連続変数のリストや 0-1 変数のリストを簡単に作成できる.

・`ortoolpy.addvars(個数) # 連続`

・`ortoolpy.addbinvars(個数) # 0-1`

目的関数の設定

```
m += 式
```

二度以上設定しても，最後のコードだけ有効となる．設定した目的関数は，`m.objective` で参照できる.

制約条件の追加

```
m += 式 == 式
m += 式 <= 式
m += 式 >= 式
```

式の例

```
2 * x + 3 * y - 5
```

和の書き方

```
lpSum(変数のリスト)
```

内積の書き方

```
lpDot(係数のリスト, 変数のリスト)
```

ソルバーの実行

```
m.solve()
```

ステータス

```
m.status  # 実行結果の整数値
LpStatus[m.status]  # 実行結果の文字列
```

PuLP で用意されているステータスの一覧を表2 に挙げる.

表 2　PuLP のステータス一覧

整数値	文字列	説明
1	Optimal	MIP gap（後述）内での厳密解が得られた
−1	Infeasible	実行可能領域が空
−2	Unbounded	非有界（いくらでも最適解をよくできる）
0	Not Solved	時間制限で止めた場合など（解が実行可能解の場合もある）
−3	Undefined	PuLP で判断できない場合

変数や式や目的関数の値

```
value(変数)
value(式)
value(m.objective)
```

線形最適化のサンプル問題

例題 1.

材料 A と B から合成できる化学製品 X と Y をたくさん生産したい.

X を 1 kg 作るのに，A が 1 kg，B が 3 kg 必要である.

Y を 1 kg 作るのに，A が 2 kg，B が 1 kg 必要である.

また，X も Y も 1 kg 当たりの価格は 100 円である.

材料 A は 16 kg，B は 18 kg しかないときに，X と Y の価格の合計が最大になるようにするには，X と Y をどれだけ生産すれば良いか求めよ（図 7）.

変数	$: x, y \geqq 0$
目的関数	$: 100x + 100y \rightarrow$ 最大化
制約条件	$: x + 2y \leqq 16$
	$3x + y \leqq 18$

図 6　定式化

定式化（図 6）を PuLP で書くと下記のようになる．

```
m = LpProblem(sense=LpMaximize)  # 数理モデル
x = LpVariable('x', lowBound=0)  # 変数
y = LpVariable('y', lowBound=0)  # 変数
m += 100 * x + 100 * y  # 目的関数
m += x + 2 * y <= 16  # 材料Aの上限の制約条件
m += 3 * x + y <= 18  # 材料Bの上限の制約条件
m.solve()  # ソルバーの実行
print(value(x), value(y))  # 4, 6
```

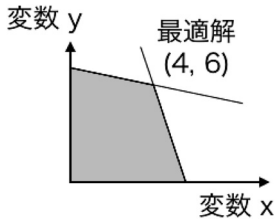

図 7　実行可能領域と最適解

最適化ライブラリーのまとめ

・PuLP では，`m +=`という独特の記述をするが，シンプルにモデルを作成できる．
・PuLP をインストールしただけで，ソルバーも使える．また，モデルを変えることなくいろいろなソルバーに切り替えられる．

6.　最適化モデルの作り方

PuLP と pandas を組み合わせて，pandas の表で変数を管理すると，定式化をわかりやすくモデル化できる．

輸送最適化問題を例にしてモデル作成を見てみよう．

例題 2.

倉庫群から組み立て工場群へ部品を搬送したい．輸送費が最小となる計画を求めたい．

・倉庫群から工場群への輸送量を決めたい → 変数
・輸送コストを最小化したい → 目的関数
・各倉庫からの搬出は，供給可能量以下 → 制約
・各工場への搬入は，需要量以上 → 制約

輸送費		組み立て工場				
		F1	F2	F3	F4	供給
倉庫	W1	10	10	11	17	35
	W2	16	19	12	14	41
	W3	15	12	14	12	42
	需要	28	29	31	25	

図 8　輸送費用と倉庫の供給と工場の需要

パラメータの設定

必要なパラメータを設定する（数字は図 8 と同じ）．

```
import numpy as np, pandas as pd
from numpy.random import randint
from itertools import product
from pulp import (
    LpProblem, LpMaximize, value,
    LpVariable, lpDot, lpSum)
np.random.seed(1)
nw, nf = 3, 4
rnw, rnf = range(nw), range(nf)
pr = list(product(rnw, rnf))
供給 = randint(30, 50, nw)
需要 = randint(20, 40, nf)
輸送費 = randint(10, 20, (nw,nf))
```

pandas を使わない数理モデル

変数は，添え字でアクセスする．結果は，(倉庫，工場) ごとの輸送量である．

```
m1 = LpProblem()
v1 = {(i, j): LpVariable('v%d_%d'%
    (i,j),lowBound=0) for i,j in pr}
m1 += lpSum(輸送費[i][j] * v1[i, j]
            for i, j in pr)
for i in rnw:
    e = lpSum(v1[i, j] for j in rnf)
    m1 += e <= 供給[i]
for j in rnf:
    e = lpSum(v1[i, j] for i in rnw)
    m1 += e >= 需要[j]
m1.solve()
{k:value(x) for k,x in v1.items()
    if value(x) > 0}
```

```
{(0, 0): 28.0,
 (0, 1): 7.0,
 (1, 2): 31.0,
 (1, 3): 5.0,
 (2, 1): 22.0,
 (2, 3): 20.0}
```

pandas を使った数理モデル

変数は，表の属性でアクセスできる．まず，表を作成しよう．`flatten` は，2 次元を 1 次元にする．

```
df = pd.DataFrame([(i, j) for i,j in
    pr], columns=['倉庫', '工場'])
df['輸送費'] = 輸送費.flatten()
df[:3]  # 最初の3行表示
```

..	倉庫	工場	輸送費
0	0	0	10
1	0	1	10
2	0	2	11

同様に数理モデルを作ってみよう．`apply(value)` とすることで各変数の値を取り出せる．

```
m2 = LpProblem()
df['Var'] = [LpVariable('v%d' % i,
    lowBound=0) for i in df.index]
m2 += lpDot(df.輸送費, df.Var)
for k, v in df.groupby('倉庫'):
    m2 += lpSum(v.Var) <= 供給[k]
for k, v in df.groupby('工場'):
    m2 += lpSum(v.Var) >= 需要[k]
m2.solve()
df['Val'] = df.Var.apply(value)
df[df.Val > 0]
```

..	倉庫	工場	輸送費	Var	Val
0	0	0	10	v0	28.0
1	0	1	10	v1	7.0
6	1	2	12	v6	31.0
7	1	3	14	v7	5.0
9	2	1	12	v9	22.0
11	2	3	12	v11	20.0

添え字を使った表現は，添え字が何を表しているか覚えていないといけなかった．しかし，PuLP と pandas を組み合わせることによって，下記のように，数理モデルが理解しやすくなる．

・単なる“i”などではなく，“倉庫”などの列名が使える
・pandas の条件式を使って，数式を組み立てられる
・pandas の便利な関数（groupby など）を使ってモデルを作成できる
・結果も表に追加できる
・pandas で結果を加工できる

pandas と PuLP を使った最適化モデルのまとめ

- 準備
  ・`from pulp import LpProblem,LpMaximize,LpVariable,value`
- 変数表 (`df`) を用意
- モデルを作成
  ・最小化：`m = LpProblem()`
  ・最大化：`m = LpProblem(sense=LpMaximize)`
- 変数表に自由変数の列を追加
  `df['Var'] = [LpVariable('v%d'%i) for i in df.index]`
- 変数表と pandas の機能を使いながら，目的関数と制約条件を追加
- ソルバーで求解：`m.solve()`
- 結果の値を取得
  ・目的関数の値：`objval = value(m.objective)`
  ・変数の値：`df['Val'] = df.Var.apply(value)`

7. おわりに

過去作成した定式化のプログラムを見返すと，変数の添字が `i, j, k, l, m` とあると，どれが何を意味しているかわかりにくいし，また，その値が `0, 1, 2` のときに，何を指しているかもわかりにくかった．pandas を使ってモデル化すると，以下のメリットが得られた．

・変数が何に対応しているのかすぐわかる．また，その属性の値は数字でなく文字列も使えるので直接的に把握しやすい．
・モデルの修正が行いやすい．デバッグ時に特定の条件を追加することが，簡単にできる．

本稿のテクニックは，シンプルなモデルだと恩恵を受けにくいが，ビジネスで必要とされる複雑なモデルでは開発効率を数倍にするだろう．

ここでは紹介できなかった内容は，参考文献 [2, 3] に詳しく紹介している[7]．

参考文献

[1] 池内孝啓，片柳薫子，岩尾エマはるか，@driller，『Python ユーザのための Jupyter [実践] 入門』，技術評論社，2017.
[2] 斉藤努，『データ分析ライブラリーを用いた最適化モデルの作り方』，近代科学社，2018（発行予定）．
[3] 穴井宏和，斉藤努，『今日から使える！ 組合せ最適化：離散問題ガイドブック』，講談社，2018.

[7] 下記からサンプルプログラムをダウンロードできる．
https://github.com/SaitoTsutomu/opt-model-book

