整数計画法による定式化入門

藤江　哲也

整数計画法の魅力の一つとして，定式化の表現力の高さがある．特にバイナリ変数 (0-1 変数) が重要な役割を果たす．本稿では，初学者を対象に，いくつかの例を通して定式化を示すとともに，注意すべき点について説明する．

キーワード：定式化，整数計画，線形計画，バイナリ変数，線形化

1. はじめに

本稿では，整数線形計画 (Integer Linear Programming : ILP)[1]による定式化を扱う．

まずイントロダクションとして，ILP の典型的と思われる例題を 2 つ挙げる．例題 1 は，線形計画問題に「整数条件」が追加された形をしている．線形計画問題に触れたことのない方は，方程式の文章題を解くようなものと思って見てほしい．

例題 1. A 社では，テーブルとチェアを製造販売している．それぞれ製造工程が 2 つあり，1 個 (1 卓，1 脚) 当たりの所要時間および利益が次のように与えられている．

	テーブル	チェア
工程 1	2 時間	2 時間
工程 2	3 時間	5 時間
利益	4 千円	5 千円

この会社で働く職人の関係上，工程 1 は 1 日 7 時間，工程 2 は 1 日 14 時間とることができる．テーブルとチェアはすべて売れるものとした場合，利益を最大にするにはテーブルとチェアをそれぞれ 1 日当たり何個製造すればよいか．

テーブルとチェアの製造数をそれぞれ x_1, x_2 とする．このとき，この最大化問題は次のように定式化することができる．テーブルとチェアは整数個しか製造できないため，x_1, x_2 に整数条件がつけられている．

(ILP1)

最大化	$Z = 4x_1 + 5x_2$	(総利益 Z 千円)
条　件	$2x_1 + 2x_2 \leqq 7$	(工程 1 の所要時間),
	$3x_1 + 5x_2 \leqq 14$	(工程 2 の所要時間),
	$x_1, x_2 \geqq 0$	(製造数は 0 以上),
	x_1, x_2 : 整数	(製造数は整数).

(ILP1) から整数条件を除去すると線形計画問題になるが，これを (LP1) とする．(LP1) の最適解と最適値は $x_1 = 7/4$, $x_2 = 7/4$, $Z = 63/4 = 15.75$ であり，(ILP1) の最適解と最適値は $x_1 = 1$, $x_2 = 2$, $Z = 14$ である (図 1)．この例題では，(LP1) の最適解を，x_1 は切り捨て，x_2 は切り上げると (ILP1) の最適解が得られる．しかし，問題が複雑になると，このように線形計画問題の解を見て ILP の解を求めることは一般に困難である．

次の例題 2 は，バイナリ変数 (0-1 変数) を用いる問題である．ILP が高い表現力を発揮するのは，選択する／しないなどをバイナリ変数で表現できるためである．3 節以降を見てもわかるように，バイナリ変数は ILP による定式化において重要な役割を果たす．

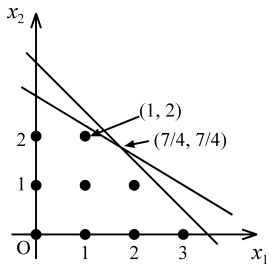

図 1　(ILP1) の図示

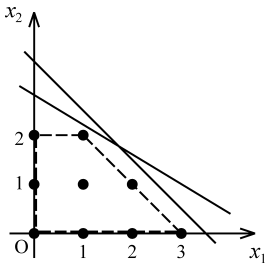

図 2　(ILP1′) の図示（破線）

[1] 混合整数計画 (Mixed Integer Programming : MIP) と同じ問題を指すが，「線形」を強調するために，本稿では ILP と表記する．

ふじえ　てつや
兵庫県立大学大学院経営研究科
〒 651–2197 兵庫県神戸市西区学園西町 8–2–1

 Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

例題 2 (ナップサック問題). B 社では，予算 160 (単位:百万円) を次年度の新規プロジェクトに計上している．ただし，候補となるプロジェクトは複数あり，予算の都合上すべてを採用することはできない．各プロジェクト (P1,. . .,P5) の NPV (正味現在価値) と予算 (いずれも推定値，単位：百万円) は次の通りである．

	P1	P2	P3	P4	P5
NPV	17	16	14	10	8
予算	60	50	40	30	20

このとき，予算 100 を超えることなく，NPV の合計を最大にするには，どのプロジェクトを採用すればよいか．

変数 x_j を，プロジェクト j を採用するとき 1，しないとき 0 を示すバイナリ変数とする．このとき，この問題は次のように定式化することができる．

(ILP2)

最大化　$Z = 17x_1 + 16x_2 + 14x_3 + 10x_4 + 8x_5$
(NPV の合計)

条　件　$60x_1 + 50x_2 + 40x_3 + 30x_4 + 20x_5 \leqq 100$
(予算の合計),

$x_1, \ldots, x_5 = 0$ または 1.

(ILP2) の最適解と最適値は $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$, $x_5 = 1$, $Z = 34$ である．すなわち，P2, P4, P5 を採用するとき NPV の合計が最大の 34 となる．

このように，ILP は

(a) 目的関数と制約式が線形 (線形不等式または線形等式)

(b) 変数の一部またはすべてが整数

である問題を対象とする．バイナリ変数は，0 以上 1 以下の整数変数であるため，(b) に含まれる．条件 (b) を除去する (バイナリ変数 x_j に対しては「$x_j = 0$ または 1」を「$0 \leqq x_j \leqq 1$」にする) と線形計画問題になるが，この操作は線形計画緩和 (linear programming relaxation) あるいは連続緩和 (continuous relaxation) とよばれ，ILP の解法において重要な役割を果たす．例題 1 の (LP1) は，(ILP1) の線形計画緩和問題である．

ILP の応用例は数多く存在する．和書では [7–10] などがあり，また，本学会誌でも応用例を数多く見つけることができる．しかし，本稿の目的は，それらを紹介することではなく，その背景にある ILP の表現力の高さを紹介することである．Dantzig による 1960 年の論文 [2]（[3] にも収録）は，ILP の解法 (Gomory の切除平面法) が開発されたことを受けて，さまざまな問題が ILP として定式化できることを示したものであり，“We shall show that a host of difficult, indeed seemingly impossible, problems of a nonlinear, nonconvex, and combinatorial character are now open for direct attack.” と述べている ([2], p. 30)．そして，目的関数が非線形 (nonlinear) の問題，目的関数や実行可能領域が非凸 (nonconvex) の問題，組合せ的な性質 (combinatorial character) をもつ問題 (組合せ最適化問題) の定式化が紹介されている．これに加え，ILP を使って問題を近似的に解くアプローチもある (3.3 節)．

ILP はこのようにさまざまな問題をカバーするとはいえ，「ILP として定式化できる」≠「整数計画ソルバーで解ける」であることに注意しなければならない．特に，変数や制約式の数が膨大になる場合はもちろん，定式化に非常に大きい数 (big-M) が含まれる場合も注意が必要である (2 節を参照)．このような場合，big-M で表現せず元の表現の特徴を使った解法を適用するのが妥当なこともある (制約プログラミング (constraint programming) との融合は，このアプローチである)．それでも，整数計画ソルバーおよび計算機パワーが急速に進歩し続けている現在，「整数計画ソルバーで解ける」問題の範囲は広がっており，ソルバーを利用する価値は十分高まっていると言える．

本稿では，[1, 5, 9, 15] を基に，ILP の定式化について解説する．また，最近の整数計画ソルバーで設定が可能な，SOS (Special Ordered Set)，半連続変数 (semi-continuous variable) を適宜取り上げて説明する．興味を持たれた方，あるいは本稿の内容では物足りない方は，直接これらの文献をご覧いただきたい．

2. 定式化について

ILP に限らず，数理最適化 (数理計画法) における定式化手順の一例は

1. 変数を定義する．
2. 問題の実行可能解を過不足なく表現するよう制約式を記述する．
3. 目的関数を記述する．

となるであろう [16]．例題 1 と例題 2 は，問題文からほぼ自動的に定式化を記述することができた．しかし，これらの問題に対しても定式化は一通りではない．例えば，例題 1 では $(x_1, x_2) = (0, 0), (0, 1), (0, 2), (1, 0),$

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

$(1,1)$, $(1,2)$, $(2,0)$, $(2,1)$, $(3,0)$ が実行可能解 (製造可能なテーブル数とチェア数の組合せ) であるから,

$$\begin{array}{lll}(\text{ILP1}') & \text{最大化} & Z = 4x_1 + 5x_2 \\ & \text{条　件} & x_1 + x_2 \leqq 3, \\ & & x_2 \leqq 2, \\ & & x_1, x_2 \geqq 0, \\ & & x_1, x_2 : \text{整数}\end{array}$$

も正しい定式化である (図 2). また, 例題 2 においても

$$\begin{array}{l}(\text{ILP2}') \\ \text{最大化}\quad Z = 17x_1 + 16x_2 + 14x_3 + 10x_4 + 8x_5 \\ \text{条　件}\quad x_1 + x_2 \leqq 1, \\ \qquad\quad x_1 + x_4 + x_5 \leqq 2, \\ \qquad\quad x_1 + x_2 + x_3 + x_4 \leqq 2, \\ \qquad\quad x_1 + x_2 + x_3 + x_5 \leqq 2, \\ \qquad\quad x_1, \ldots, x_5 = 0 \text{ または } 1\end{array}$$

が正しい定式化になっていることを示すことができる[2]. そして, このように線形制約 (線形不等式系) が異なると, 線形計画緩和問題も異なる. 例えば例題 1 では, (ILP1′) の線形計画緩和問題の実行可能解領域 (多面体) は, (ILP1) のそれに含まれている (図 1, 図 2). 前者は ILP の実行可能解集合 $(x_1, x_2) = (0,0), \ldots, (3,0)$ の凸包[3]であり, これ以上領域を小さくするように線形制約を記述することはできない. つまり, 凸包は最強の定式化 (理想の定式化) を与えるが, それを記述する線形制約をすべて求めることは一般に非常に困難である. ただし, 最近の整数計画ソルバーは凸包に近づける機能 (前処理, カット生成など) を備えている. 例えば例題 1 の制約 $2x_1 + 2x_2 \leqq 7$ は, 両辺を 2 で割ると $x_1 + x_2 \leqq 3.5$ となるが, 整数条件によって左辺は整数の値しかとらない. よって, $x_1 + x_2 \leqq 3$ としてよい. これが前処理の一つである.

また, 定式化手順の「1. 変数の定義」をどのようにするかによって, さらにバラエティに富んだ定式化が可能となる. これについては 4 節で紹介する.

ところで, 整数計画ソルバーによって解きやすい定式化を「良い」定式化とよぶことにする. ほぼすべての整数計画ソルバーのアルゴリズムは, 線形計画緩和をベースにした解法 (分枝限定法／分枝カット法) であるため, 一般に, 次の条件を満たすほど良い定式化であると言うことができる.

(a) 線形計画緩和がよい (凸包に近い)
(b) 変数の数が少ない
(c) 制約式の数が少ない

変数や制約式の数が少ない問題でも, 解くことが難しい場合があるのが ILP の特徴であると言える. 上記の (a)–(c) 以外にも, 極端に大きい・極端に小さい係数を含めないことなどが挙げられるが, まずは (a)–(c) に注目すべきであろう. 例えば big-M を含む定式化は係数が大きいうえに (a) の意味で弱いことが多く, よって多用しないことを心がけるべきとされている. また, 変数の上下限が予めわかっていれば, それを記述しておくのがよい.

3. 様々な表現方法

例題 2 のナップサック問題においてバイナリ変数を導入した. 例えば x_1 は, P1 が採用されるとき $x_1 = 1$, 採用されないとき $x_1 = 0$ であった. 本節では, このようなバイナリ変数を使ったさまざまな表現方法について概観する.

3.1 論理的関係 1

再び例題 2 を例として, いくつかの追加条件とその表現方法をリストアップする.

1. 採用されるプロジェクト数が高々3 (3 以下):
   $x_1 + x_2 + \cdots + x_5 \leqq 3$.
2. 採用されないプロジェクト数が高々3:
   $(1 - x_1) + (1 - x_2) + \cdots + (1 - x_5) \leqq 3$.
3. P1 または P2 を採用:
   $x_1 + x_2 \geqq 1$.
4. P1 を採用するならば P2 を採用:
   $x_1 \leqq x_2$.
5. P1, ..., P5 のうち採用できる数は 0 または 2:
   $x_1 + x_2 + \cdots + x_5 = 2y$, $y = 0$ または 1.
   あるいは, y を使わず

$$\begin{cases} +x_1 + x_2 + \cdots + x_5 \leqq 2, \\ -x_1 + x_2 + \cdots + x_5 \geqq 0, \\ +x_1 - x_2 + \cdots + x_5 \geqq 0, \\ \cdots, \\ +x_1 + x_2 + \cdots - x_5 \geqq 0 \end{cases}$$

と表現することもできる.

[2] これは極小被覆に基づく定式化であり, [7] の定理 8.9 を適用した.
[3] 集合 S を含む凸集合の中で (包含関係の意味で) 最小のものを凸包とよぶ. 例題 1, 例題 2 のように S が有限集合の場合, 凸包は多面体, つまり線形不等式系で記述できることが知られている (Weyl の定理).

 Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

表 1　4 の実行可能解

x_1	x_2
1	1
0	0
0	1

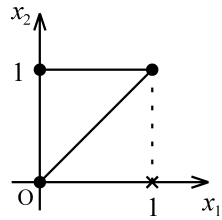

図 3　4 の図示

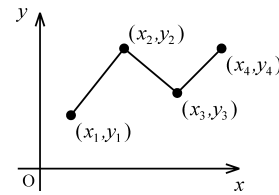

図 4　区分線形関数

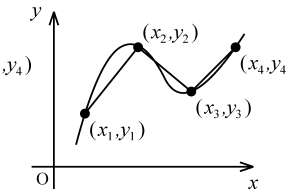

図 5　区分線形関数近似

2 については，x_1 の反転 (否定) は $1-x_1$ である ($x_1=1$ のとき $1-x_1=0$，$x_1=0$ のとき $1-x_1=1$) ことから導かれる．2 は $x_1+x_2+\cdots+x_5 \geqq 2$ と同じ意味 (採用されないプロジェクト数を 3 以下にするためには，2 つ以上のプロジェクトを採用しなければならない) であるが，反転を使うと上記のように直接的に記述することができる．

論理的関係はこれに限らず数多くあるが，このような式を導くための一助として，4 を例として説明する．4 では，変数は x_1 と x_2 であり，実行可能解は 3 通りある（表 1）．そしてこの凸包をとると，$x_1 \leqq x_2$ が導かれる (図 3)．

3.2　論理的関係 2

次に，制約式に関する論理的関係を取り上げる．ここでは，線形不等式を $\boldsymbol{a}^\top \boldsymbol{x} \leqq b$ と表している ($\top$ は転置記号)．

1. $\boldsymbol{a}_1^\top \boldsymbol{x} \leqq b_1$ または $\boldsymbol{a}_2^\top \boldsymbol{x} \leqq b_2$：

$$\begin{cases} \boldsymbol{a}_1^\top \boldsymbol{x} - b_1 \leqq M(1-y), \\ \boldsymbol{a}_2^\top \boldsymbol{x} - b_2 \leqq My, \\ y = 0 \text{ または } 1. \end{cases}$$

2. m 本の線形不等式 $\boldsymbol{a}_i^\top \boldsymbol{x} \leqq b_i$ $(i=1,\ldots,m)$ のうち p 本を満たす：

$$\begin{cases} \boldsymbol{a}_i^\top \boldsymbol{x} - b_i \leqq M(1-y_i) \quad (i=1,\ldots,m), \\ \displaystyle\sum_{i=1}^{m} y_i = p, \\ y_i = 0 \text{ または } 1 \quad (i=1,\ldots,m). \end{cases}$$

1, 2 とも線形不等式 $\boldsymbol{a}_i^\top \boldsymbol{x} \leqq b_i$ を線形不等式系 $A_i \boldsymbol{x} \leqq \boldsymbol{b}_i$ に拡張することができる．

1 は離接制約 (disjunctive constraint) と呼ばれている．M は十分大きい定数 (big-M) である．$y=1$ のとき，$\boldsymbol{a}_1^\top \boldsymbol{x} \leqq b_1$，つまり 1 番目の制約を有効にして 2 番目の制約を冗長にしている．よって，M の値としては，制約が冗長となるように選択する．2 は 1 の拡張である．

「$x=0$ または $\ell \leqq x \leqq u$」なる条件 (ただし $\ell > 0$，u は $+\infty$ でもよい) がある変数 x を半連続変数という．これは離接制約であるが，例えば u が有界の場合

$$\begin{cases} \ell y \leqq x \leqq uy, \\ y = 0 \text{ または } 1 \end{cases}$$

として表現することができる．u が $+\infty$ の場合，big-M を導入する．

3.3　非線形関数 1

図 4 に示す区分線形関数の表現を考える．応用例として，図 5 のように，非線形関数 $y=f(x)$ の区分線形関数近似がある．

区分線形関数上の点 (x,y) は，ある線分上にある．例えば (x_1,y_1) と (x_2,y_2) で結ばれる線分上にある場合，

$$(x,y) = t_1(x_1,y_1) + t_2(x_2,y_2), t_1+t_2=1, t_1,t_2 \geqq 0$$

として表現することができる[4]．これを考慮すると，一般に (x,y) は

$$\begin{cases} (x,y) = t_1(x_1,y_1) + \cdots + t_4(x_4,y_4), \\ t_1 + \cdots + t_4 = 1,\ t_1,\ldots,t_4 \geqq 0, \\ \text{高々2 つの隣り合う } t_i \text{ が正 (非零)} \end{cases}$$

として表現することができる．「高々2 つの隣り合う t_i が正」の形をした制約を満たす変数群 $(t_1,\ldots,t_4)$ を，SOS2 (SOS of Type 2) 変数群という．これは，バイナリ変数 z_1,z_2,z_3 を導入することで，次のように表現することができる．

$$\begin{cases} t_1 \leqq z_1,\ t_2 \leqq z_1+z_2,\ t_3 \leqq z_2+z_3,\ t_4 \leqq z_3, \\ z_1+z_2+z_3 = 1,\ z_1,z_2,z_3 = 0 \text{ または } 1. \end{cases} \tag{1}$$

絶対値関数 $y=|x|$ ($-\ell \leqq x \leqq u$，$\ell, u \geqq 0$ は定数)

[4] $(x,y) = t(x_1,y_1) + (1-t)(x_2,y_2)$，$0 \leqq t \leqq 1$ という表現と同じものである．実際，$t_1=t, t_2=1-t$ とおけばよい．

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

は区分線形関数であるため，$(-\ell,\ell),(0,0),(u,u)$ について上記の議論を適用することができる．ただし，この場合については

$$\begin{cases} -x \leqq y \leqq -x+2uz, \\ x \leqq y \leqq x+2\ell(1-z), \\ z=0 \text{ または } 1 \end{cases}$$

と表現することもできる．

また，$y=|x|$ を $y \geqq |x|$ に置き換えることができる場合には，さらに $y \geqq |x|$ を $y \geqq x$，$y \geqq -x$ に置き換えればよく，バイナリ変数を導入する必要はない．例えば，目的関数が $|f(x)|$ の最小化問題では，目的関数を y とし，制約式に $y \geqq f(x)$，$y \geqq -f(x)$ を追加すればよい．$y=\max\{a_1x+b_1,\ldots,a_mx+b_m\}$ に対しても同様である．これは線形計画法におけるテクニックであるが，重要と思われるためここでも取り上げた．

3.4 非線形関数 2

前節では非線形関数の区分線形関数近似を紹介したが，ここではバイナリ変数を含む非線形関数の線形化を取り上げる．

まず，バイナリ変数 x_1 と x_2 の積 $y=x_1x_2$ を考える．この場合，実行可能解は $(x_1,x_2,y)=(0,0,0),(1,0,0),(0,1,0),(1,1,1)$ であるから，

$$y=x_1x_2,\ x_1,x_2=0 \text{ または } 1$$

を

$$\begin{cases} 1-x_1-x_2+y \geqq 0,\ x_1-y \geqq 0,\ x_2-y \geqq 0, \\ x_1,x_2=0 \text{ または } 1 \end{cases}$$

に置き換えることができる．これらの線形不等式は，実行可能解の凸包から得られる．この拡張として，k 個の積 $y=x_1\cdots x_k$ ($x_1,\ldots,x_k$ はバイナリ変数) では，

$$\begin{cases} (k-1)-\sum_{i=1}^{k} x_i+y \geqq 0, \\ x_i-y \geqq 0 \quad (i=1,\ldots,k), \\ x_1,\ldots,x_k=0 \text{ または } 1 \end{cases}$$

とすることができる．よって，バイナリ変数の多項式は線形化できることがわかる．なぜなら，バイナリ変数 x_i は $x_i^n=x_i^{n-1}=\cdots=x_i^2=x_i$ を満たす (x_i に 0 と 1 を代入して確認できる) ため，例えば $x_1^2x_2^5x_4^3=x_1x_2x_4$ であるように，バイナリ変数の積は $x_{i_1}x_{i_2}\cdots x_{i_k}$ の形にすることができるためである．

次に，連続変数 x とバイナリ変数 z の積 $y=xz$ を考える．x には上下限制約 $\ell \leqq x \leqq u$ があると仮定する．このとき，

$$\begin{cases} \ell z \leqq y \leqq uz, \\ x-u(1-z) \leqq y \leqq x-\ell(1-z), \\ z=0 \text{ または } 1 \end{cases}$$

とすることができる．

3.5 整数変数からバイナリ変数への変換

バイナリ変数の表現力の高さを示すため，一般の整数変数も表現できることを紹介する．例えば

$$0 \leqq x_j \leqq 9,\ x_j : \text{整数}$$

に対して，次に挙げる複数の方法で変数 x_j を消去することができる．

$$\begin{cases} x_j=y_{1j}+2y_{2j}+2^2y_{3j}+2^3y_{4j}, \\ y_{1j},\ldots,y_{4j}=0 \text{ または } 1. \end{cases}$$

$$\begin{cases} x_j=y_{1j}+2y_{2j}+3y_{3j}+\cdots+9y_{9j}, \\ y_{1j}+\cdots+y_{9j} \leqq 1, \\ y_{1j},\ldots,y_{9j}=0 \text{ または } 1. \end{cases}$$

$$\begin{cases} x_j=y_{1j}+y_{2j}+y_{3j}+\cdots+y_{9j}, \\ y_{1j},\ldots,y_{9j}=0 \text{ または } 1. \end{cases}$$

最後の方法においては，$y_{1j} \geqq y_{2j} \geqq \cdots \geqq y_{9j}$ という制約を加えて解の対称性と冗長性を排除するのがよい．

また，x_j が限られた離散値を取る場合，例えば

$$x_j=3 \text{ または } 4 \text{ または } 9$$

の場合，

$$\begin{cases} x_j=3y_1+4y_2+9y_3, \\ y_1+y_2+y_3=1,\ y_1,y_2,y_3=0 \text{ または } 1 \end{cases}$$

と書き直すことができる．y_1,y_2,y_3 は，ちょうど一つが正 (非ゼロ) にならなければならず，このような変数群を SOS1 (SOS of Type 1) 変数群という．連続変数 x_1,x_2,x_3 (ただし，$x_1,x_2,x_3 \geqq 0$) が SOS1 の場合

$$\begin{cases} x_1 \leqq My_1,\ x_2 \leqq My_2,\ x_3 \leqq My_3, \\ x_1,x_2,x_3 \geqq 0, \\ y_1+y_2+y_3=1, \\ y_1,y_2,y_3=0 \text{ または } 1 \end{cases}$$

と表現することができる (M は十分大きい定数)．SOS2 の表現 ((1) 式) においても，SOS1 変数群 z_1,z_2,z_3 が現れている．

4. 変数の定義について

最後に，さまざまな変数の定義方法があることを紹

介するために，スケジューリング問題 (例題 3) と巡回セールスマン問題 (例題 4) を取り上げる．初学者の方は必ずしも詳細まで追う必要はなく，変数の定義によって定式化が大きく変わることを理解していただきたい．

例題 3. 4 つのジョブを 1 台の機械で処理する．ジョブに関するデータは次のように与えられている．

ジョブ j	1	2	3	4
w_j	2	1	3	5
p_j	3	2	5	7

ジョブ j の処理時間は p_j で与えられている．このとき，重み付き完了時刻和 $Z = w_1C_1 + \cdots + w_4C_4$ を最小にするジョブの処理順序を求めよ．C_j はジョブ j の完了時刻である．機械は同時に二つ以上のジョブを処理できず，また，一度処理を開始すると完了まで中断はできないものとする．

例えばジョブを $1,2,3,4$ の順に処理するとき，各ジョブの処理開始時刻 S_j と完了時刻 C_j は次のようになる．

ジョブ j	1	2	3	4
開始時刻 S_j	0	3	5	10
完了時刻 C_j	3	5	10	17

よって，$Z = w_1C_1 + \cdots + w_4C_4 = 2 \times 3 + \cdots + 5 \times 17 = 126$ である．最適な処理順序は $4,3,1,2$ である ($Z = 118$)．この問題は，スケジューリング理論では $1||\sum w_jC_j$ と記述される問題であり，容易に解くことができる．実際，w_j/p_j の降順に処理するスケジュールが最適になることが知られている [14]．しかし，この問題を拡張 (一機械から多機械への拡張，開始可能時刻の条件追加など) すると容易には解けなくなるため，$1||\sum w_jC_j$ に対する定式化を考えることは無意味ではない．ここでは [13] などを基として，さまざまな定式化を紹介する．[9] では $1||\sum w_jC_j$ の線形計画問題による定式化 (制約式の数が指数オーダー) が紹介されている．なお，以下ではジョブ数を n とし，簡単のため p_j はすべて正の整数と仮定する．

定式化 1 x_{jk} $(j,k = 1,\ldots,n, j \neq k)$ を，ジョブ j がジョブ k より先に処理される ($j \to k$ と記す) とき 1，されないとき 0 を示すバイナリ変数とする．このとき，

$$
\begin{aligned}
\text{最小化} \quad & Z = \sum_{j=1}^{n} w_j \left(\sum_{k \neq j} p_k x_{kj} + p_j \right) \\
\text{条　件} \quad & x_{kj} + x_{jk} = 1 \quad (j,k = 1,\ldots,n, j \neq k), \\
& x_{jk} + x_{k\ell} + x_{\ell j} \leqq 2 \\
& \qquad (j,k,\ell = 1,\ldots,n, j \neq k, j \neq \ell, k \neq \ell), \\
& x_{jk} = 0 \text{ または } 1 \quad (j,k = 1,\ldots,n, j \neq k)
\end{aligned}
$$

と定式化することができる．ジョブ j の完了時刻は $C_j = \sum_{k \neq j} p_k x_{kj} + p_j$ と表現することができるので，これが目的関数に使われている．2 番目の制約式は，推移律 ($j \to k$, $k \to \ell$ ならば $j \to \ell$) を表している．

定式化 2 時刻を単位時間ごとに区切り，時刻 $t-1$ に始まり時刻 t に終わる範囲を期間 t とよぶ．また，計画期間を $1,\ldots,T$ とする．このとき x_{jt} $(j = 1,\ldots,n, t = 1,\ldots,T - p_j + 1)$ を，ジョブ j が期間 t に処理を開始するとき 1，しないとき 0 を示すバイナリ変数とすると，

$$
\begin{aligned}
\text{最小化} \quad & Z = \sum_{j=1}^{n} w_j \left(\sum_{t=1}^{T-p_j+1} (t + p_j - 1) x_{jt} \right) \\
\text{条　件} \quad & \sum_{t=1}^{T-p_j+1} x_{jt} = 1 \quad (j = 1,\ldots,n), \\
& \sum_{j=1}^{n} \sum_{s=t-p_j+1}^{t} x_{js} \leqq 1 \quad (t = 1\ldots,T), \\
& x_{jt} = 0 \text{ または } 1 \\
& \qquad (j = 1,\ldots,n, t = 1,\ldots,T - p_j + 1)
\end{aligned}
$$

と定式化することができる．

定式化 3 ジョブ j の完了時刻 C_j $(j = 1,\ldots,n)$ を変数とする．

$$
\begin{aligned}
\text{最小化} \quad & Z = \sum_{j=1}^{n} w_j C_j \\
\text{条　件} \quad & C_j \geqq p_j \quad (j = 1,\ldots,n), \\
& C_k \geqq C_j + p_k \text{ または } C_j \geqq C_k + p_j \\
& \qquad (j,k = 1,\ldots,n, j \neq k).
\end{aligned}
$$

この定式化には離接制約「$C_k \geqq C_j + p_k$ または $C_j \geqq C_k + p_j$」が含まれる．これは，3.2 節で紹介した方法を適用すると

$$
\begin{cases}
C_j - C_k + p_k \leqq M(1 - y_{jk}), \\
C_k - C_j + p_j \leqq M y_{jk}, \\
y_{jk} = 0 \text{ または } 1
\end{cases}
$$

とすることができる．

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

例題 4 ((非対称) 巡回セールスマン問題). 頂点集合 (都市の集合) を $V=\{1,\dots,n\}$ とし，頂点 i から j への費用 (距離，移動時間等) を c_{ij} とする．このとき，すべての頂点 (都市) をちょうど一度ずつ通る巡回路のうち，総費用最小のものを求めよ．

この問題に対して，次の定式化が標準的である [4, 8–10]．x_{ij} を，巡回路として i から j に直接移動するとき 1，そうでないとき 0 を示すバイナリ変数とする．また，A を直接移動できる頂点ペア (i,j) の集合とする．このとき，

$$
\begin{aligned}
\text{最小化}\quad & Z=\sum_{(i,j)\in A} c_{ij}x_{ij}\\
\text{条　件}\quad & \sum_{i:(i,j)\in A} x_{ij}=1 \quad (j\in V),\\
& \sum_{j:(i,j)\in A} x_{ij}=1 \quad (i\in V),\\
& \sum_{\substack{(i,j)\in A:\\ i\in S, j\in S}} x_{ij}\leqq |S|-1\\
& \qquad\qquad (S\subseteq V\setminus\{1\},\ |S|\geqq 2), \qquad (2)\\
& x_{ij}=0 \text{ または } 1 \quad (i,j=1,\dots,n).
\end{aligned}
$$

(2) は部分巡回路除去制約とよばれている．(2) は制約式の数が $O(2^n)$ あり，n が大きくなると整数計画ソルバーで直接解かせることが難しくなる．そこで，制約式の数を抑える方法が研究されている．例えば (2) を

$$
\begin{aligned}
& u_i-u_j+(n-1)x_{ij}\leqq n-2\\
& \qquad (i,j\in V\setminus\{1\},\ i\neq j) \qquad (3)
\end{aligned}
$$

に置き換えることができる [11]．この置き換えによって制約式の数は $O(n^2)$ にまで減少するが，弱い定式化になる．これを見るため，例として $S=\{i,j,k\}$ を考える．この S に対する (2) 式は

$$x_{ij}+x_{ji}+x_{ik}+x_{ki}+x_{jk}+x_{kj}\leqq 2$$

である．一方，有向閉路 (部分巡回路)$(i,j),(j,k),(k,i)$ のそれぞれについて (3) 式の和をとると，変数 u が消去され

$$x_{ij}+x_{jk}+x_{ki}\leqq 3\times\frac{n-2}{n-1}$$

が得られる．右辺は 3 より小さいため，(3) は部分巡回路除去制約になっているものの，定式化としては弱いことが推測され，実際その通りであることが知られている．詳細については最近の解説 [6, 9, 12] をご覧いただきたい．

5. おわりに

本稿では，整数計画法の定式化に焦点を当てて解説を行った．専門的な内容も含まれているが，(教科書的な) 例題 1 よりもさらに広い世界や可能性が整数計画法にあることが伝われば，本稿の役目は果たされたと考えている．

本稿で紹介したように，同じ問題でもさまざまな定式化が可能な場合があるが，そうでなくとも定式化の実際には試行錯誤が欠かせないはずである．整数計画ソルバーは使いやすさの点でも向上しており，試行錯誤する際の道具としても便利なものである．この点も含め，本特集の他の解説を是非参考にしていただきたい．

参考文献

[1] D.-S. Chen, R. G. Baston and Y. Dang, *Applied Integer Programming: Modeling and Solution*, Wiley, 2010.

[2] G. B. Dantzig, "On the Significance of Solving Linear Programming Problems with Some Integer Variables," *Econometrica*, **28** (1960), 30–44.

[3] G. B. Dantzig, *Linear Programming and Extensions*," Princeton University Press, 1963. (小山昭雄訳,『線型計画法とその周辺』, ホルト・サウンダース・ジャパン, 1983.)

[4] G. B. Dantzig, D. R. Fulkerson and S. M. Johnson, "Solution of a Large Scale Traveling Salesman Problem," *Operations Research*, **2** (1954), 393–410.

[5] Fair Isaac Corporation, *FICO Xpress Optimization Suite, MIP formulations and linearizations, Quick reference*, 2010.

[6] 藤江哲也,「最近の混合整数計画ソルバーの進展について」『オペレーションズ・リサーチ』, **56** (2011), 263–268.

[7] 今野浩,『整数計画法』, 産業図書, 1981.

[8] 今野浩, 鈴木久敏 (編),『整数計画法と組合せ最適化』, 日科技連出版社, 1982.

[9] 久保幹雄,『サプライ・チェイン最適化ハンドブック』, 朝倉書店, 2007.

[10] 久保幹雄, 田村明久, 松井知己 (編),『応用数理計画ハンドブック』, 朝倉書店, 2002.

[11] C. Miller, A. Tucker and R. Zemlin, "Integer Programming Formulations and Traveling Salesman Problems," *Journal of the Association for Computing Machinery*, **7** (1960), 326–329.

[12] 沼田一道,「汎用 MIP ソルバによる巡回セールスマン問題の求解—多項式オーダ本数の部分巡回路除去制約—」,『オペレーションズ・リサーチ』, **56** (2011), 452–455.

[13] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, 1995.

[14] W. E. Smith, "Various Optimizer for Single-

 Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

Stage Production," *Naval Research Logistics Quarterly*, **3** (1956), 59–66.
[15] H. P. Williams, *Model Building in Mathematical Programming*," John Wiley and Sons, 1993. (前田英次郎監訳, 小林英三訳, 『数理計画モデルの作成法』, 産業図書, 1995.)
[16] L. Wolsey, *Integer Programming*, Wiley, 1998.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.