整数計画法入門

松井知己
東京大学大学院工学系研究科計数工学専攻
http://www.misojiro.t.u-tokyo.ac.jp/~tomomi/

1 組合せ最適化問題の例

1.1 ナップサック問題

荷物の集合 $\{1,2,\ldots,n\}$. 荷物 i の価値を w_i, 荷物 i の重量を a_i, ナップサックの重量制限を b とする.

$$\text{maximize} \sum_{i=1}^{n} w_i x_i, \quad \text{subject to } \sum_{i=1}^{n} a_i x_i \leq b, \ \ x_1,\ldots,x_n \in \{1,0\}.$$

類似問題: 多重ナップサック問題, 一般化割当問題, ビンパッキング問題, subset sum 問題.

1.2 巡回セールスマン問題

$G=(V,E)$: 頂点集合 V と枝集合 E からなる完全有向グラフ. ただし $V=\{1,2,\ldots,n\}$, $E=\{(i,j) \mid i,j \in V,\ i \neq j\}$ とする. 枝 (i,j) の枝重み w_{ij}.

$$\begin{array}{ll} \text{minimize} & \sum_{\{i,j\}\in E} w_{ij}x_{ij} \\ \text{subject to} & \sum_{i\in V-j} x_{ij} = 1 \quad (\forall i \in V), \ \sum_{j\in V-i} x_{ij} = 1 \quad (\forall j \in V), \\ & \sum_{i\in S}\sum_{i\in V-S} x_{ij} \geq 1 \quad (\emptyset \neq S \subset V), \ \ x_{ij} \in \{0,1\} \ \ ((i,j)\in E). \end{array}$$

類似問題: 対称巡回セールスマン問題, 非対称巡回セールスマン問題, 平面巡回セールスマン問題, m 人巡回セールスマン問題, 1 機械スケジューリング問題, ハミルトン閉路問題.

1.3 資源配分問題

供給施設の候補の集合 $\{1,2,\ldots,n\}$, 需要地の集合 $\{1,2,\ldots,m\}$. 施設 i の建設費用を w_i とし供給能力を a_i とする. また, 施設 i から需要地 j までの単位量当たりの輸送費用を c_{ij} とし. 需要地 j の需要量を b_j とする.

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} w_i y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \\ \text{subject to} & \sum_{i=1}^{n} x_{ij} = b_j \quad (j \in \{1,2,\ldots,m\}), \ \sum_{j=1}^{m} x_{ij} \leq a_i y_i \quad (i \in \{1,2,\ldots,n\}), \\ & y_i \in \{0,1\}, \ x_{ij} \geq 0 \ \ (i \in \{1,2,\ldots,n\}; j \in \{1,2,\ldots,m\}). \end{array}$$

2 計算量について

2.1 組合せ的爆発

n	$\log n$	$n^{0.5}$	n^2	2^n	$n!$
10	3.32	3.16	10^2	1.02×10^3	3.6×10^6
100	6.64	10.00	10^4	1.27×10^{30}	9.33×10^{157}
1000	9.97	31.62	10^6	1.07×10^{301}	4.02×10^{2567}

2.2 多項式時間算法と NP- 完全

算法のスピードの測り方： 最悪ケース評価, 平均計算時間評価.

問題の入力サイズ: 入力する数字の個数と数値. ただし数値の入力サイズは, 必要なビット数.

多項式時間算法： 最悪ケースでも, 入力サイズの多項式時間で終了する. 現実的に使える算法は, 多項式時間算法であると言われる事が多い.

NP- 完全 (NP- 困難)： 多項式時間算法の存在が絶望視されている問題の集まり. 現在千問以上の問題が知られており, そのリストは今も増えつづけている. (NP- 完全と NP- 困難の違いは, かなり専門的).

3 解法の種類

厳密解法: 最適解を 1 つ見つける解法. 動的計画法, 分枝限定法, 切除平面法, 分枝切除法.

近似解法: 出力される解の最適解に対する精度が保証されている解法. ランダム丸め法, 半正定値近似解法.

発見的解法： 出力される解の精度保証は無い解法. 貪欲解法, 局所探索法, タブーサーチ, シミュレーテッドアニーリング, 遺伝的アルゴリズム ....

4 定式化のテクニック

線形緩和問題の許容解領域が小さくなるように, 制約を付け加える.

十分大きな定数の導入について

変数 x が製品の生産量を表し, 生産費用が $f(x)$ で表されるとする. 可能な生産量の領域が Ω で表されるとして, 生産費用を最小化する問題 $\min\{f(x) \mid x \geq 0,\ \ x \in \Omega\}$ について議論しよう. 生産費用に固定費用が付いているアフィン関数

$$f(x) = \begin{cases} 0 & (x = 0), \\ a + bx & (x > 0), \end{cases}$$

で生産費用が表されるとする. 上記の問題の関数を以下のように明示的に書き変えることができる;

$$\min\{ay + bx \mid x \geq 0,\ \ x \in \Omega,\ \ My \geq x,\ \ y \in \{0, 1\}\}.$$

上記で M は十分大きな数値を入れれば良い. しかしながら, M として必要以上に大きな数値を入れることは、 整数計画ソフトウェア等での求解時間の増加を招く事が多い. 残念ながら、 M をどのくらいまで小さくできるかを知る効率的な方法は、 一般的には存在しない。

冗長な制約式の導入

少ない本数の不等式で表すのがいつでも良いとは限らない. 例えば,

$$\{\boldsymbol{x} \in \mathrm{R}^4 \mid x_1 + x_2 = 1,\ x_3 + x_4 = 1,\ \ x_1, x_2, x_3, x_4 \in \{0, 1\}\}$$

という集合は,

$$\{\boldsymbol{x} \in \mathrm{R}^4 \mid x_1 + x_2 + 3x_3 + 3x_4 = 4,\ x_1, x_2, x_3, x_4 \in \{0, 1\}\}$$

と等式制約 1 本で表すこともできるが, このような変形は, 通常は計算時間の増加を招くので, やってはいけない. 一般に, 整数制約を取り除いた際 (0-1 変数であれば 0 以上 1 以下という制約に取り替えた際), 許容領域が狭い方がソフトウェア等での計算時間が短くなり, 良い定式化となることが多い. 上記の例では, 最初の良い定式化の整数制約を取り替えると

$$\Omega_1 = \{\boldsymbol{x} \in \mathrm{R}^4 \mid x_1 + x_2 = 1,\ x_3 + x_4 = 1,\ 0 \leq x_1, x_2, x_3, x_4 \leq 1\}$$

となり, あとの悪い定式化の制約式を取り替えると

$$\Omega_2 = \{\boldsymbol{x} \in \mathrm{R}^4 \mid x_1 + x_2 + 3x_3 + 3x_4 = 4,\ 0 \leq x_1, x_2, x_3, x_4 \leq 1\}$$

となり $\Omega_1 \subset \Omega_2$ が成り立っている. さらに Ω_2 は, $(1, 1, 4/3, 0)$ といった, Ω_1 に含まれない解を含んでいる. 求解時間が変わる理由は, 多くのソフトウェアでは, 次節の緩和法をなんらかの形で内部に組み込んでいるため, 緩和した際の解領域が狭い方が緩和法の性能が向上するからである

非線形制約の取り扱い

ソフトウェアによっては, 非線形の制約を扱えるものもあるが, これは可能な限り使わない方が良い. 例えば 2 次式の制約を扱えるソフトウェアであっても, 「$x_1+x_2+x_3$ が 0 または 2 という値を取る」ことを, $(x_1+x_2+x_3-1)^2=1$ という制約で表す事は勧められない. その代わりに $\{(x_1,x_2,x_3) \mid x_1+x_2+x_3 \le 2,\ x_1-x_2-x_3 \le 0,\ -x_1+x_2-x_3 \le 0,\ -x_1-x_2+x_3 \le 0\}$ という制約を用いる事により, 計算時間を大幅に短縮できるだろう. これは, 非線形の制約は一般には扱いが困難であり, これに対処できる性能の良い解法はあまり知られていないためである. そのため, 非線形制約を扱えるソフトでも, 内部での処理は原始的な算法しか組み込んでいない事がしばしばある. また, $\neq$ を扱えるソフトウェアであっても, この使用は出来る限り避ける方が良い.

資源配分問題

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^{n} y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \\
\text{subject to} & \sum_{i=1}^{n} x_{ij} = b_j \quad (j \in \{1,2,\ldots,m\}),\ \sum_{j=1}^{m} x_{ij} \le a_i y_i \quad (i \in \{1,2,\ldots,n\}), \\
& x_{ij} \le b_j y_i \quad (i \in \{1,2,\ldots,n\}; j \in \{1,2,\ldots,m\}), \\
& y_i \in \{0,1\},\ \ x_{ij} \ge 0 \quad (i \in \{1,2,\ldots,n\}; j \in \{1,2,\ldots,m\}).
\end{array}
$$

5 分枝限定法

緩和問題を何度も解くことで, 元問題を解く. 分枝操作；問題を子問題に分ける (場合分けする). 緩和問題で, 整数になっていない変数を固定することで場合分けすることが多い. 限定操作；解かなくて良い子問題を見つけ, 分枝操作を中止する. 次のような理由で限定操作を行う事ができる. (1) 最適解が偶然見つかる (例; 緩和問題の最適解が元問題の許容解になっている), (2) 実行不能になる, (3) 最適値の上界が, 暫定解（現在までに分かった最も良い解） の目的関数値より悪い (小さい).

分枝限定法の基本的な枠組みは以下のように記述できる. ただし, ここでは最大化問題について説明している. 下界値とは最適値以下であることが分かっている値で, 発見的解法などで得られる値をである. 上界値は, 最適値以上であることが分かっている値で, 緩和問題を解いて得られる. 元問題を P_0 とする.

Step1: 暫定解$\boldsymbol{x}$ を適当な初期実行可能解とする, z を$\boldsymbol{x}$の目的関数値値, $\mathcal{N}=\{(\mathrm{P}_0)\}$(元問題), $k=1$ とする.

Step2: $\mathcal{N}=\emptyset$ ならば, $\boldsymbol{x}$を最適解として出力し終了.
そうでなければ, $\mathcal{N}$ から適当な子問題を選びそれを P_k とし, $\mathcal{N}$ から P_k を取り除く.

step3: P_k の緩和問題を解き，解を $\overline{\boldsymbol{x}}^k$, 得られた上界値を z^k とする.
緩和問題が許容解を持たないならば Step2 へ戻る.

step4: $\overline{\boldsymbol{x}}^k$ が元問題 P_0 の実行可能解の時.
$z > \overline{z}_k$ なら (元問題 P_0 のよりよい許容解が得られたので) $\boldsymbol{x} := \overline{\boldsymbol{x}}^k$, $z := \overline{z}^k$ と更新する. Step2 へ戻る.

step5: $\overline{z}^k \le z$ ならば （子問題 P_k の最適解は $\boldsymbol{x}$より目的関数値が大きくないので，） Step2 へ戻る.

step6: P_k の実行可能領域を分割した子問題を生成し, それらを $\mathcal{N}$ に加え, Step2 へ戻る.

分枝限定法は基本的に列挙法であり， 最悪の場合を考えれば全ての解を列挙してしまうこともあり得る. しかし現実には極めて効果的に働き, ほんの僅かな回数緩和問題を解くことにより, かなり大規模な問題でさえも最適解を得ることに成功している. 無駄な列挙を削除するには, より小さな上界値を得ることが必要不可欠である. その他にも，分枝限定法の効率化には, 子問題の選択法や子問題の分割法などを工夫する必要がある.

以下のナップサック問題を解く分枝限定法を記述する.

$$\mathrm{P0}:\ \max\{3x_1+4x_2+x_3+2x_4 \mid 2x_1+3x_2+x_3+3x_4 \le 4,\ x_1,x_2,x_3,x_4 \in \{0,1\}.\}$$

線形緩和問題とは, 以下の問題を指す.

$$\max\{3x_1+4x_2+x_3+2x_4 \mid 2x_1+3x_2+x_3+3x_4 \le 4,\ 1 \ge x_1,x_2,x_3,x_4 \ge 0.\}$$

線形緩和問題の最適解は $(z; x_1, x_2, x_3, x_4) = (17/3; 1, 2/3, 0, 0)$. この最適解は, 単位重さ当たりの価値が最も高いものから, ナップサックに詰めるだけで求めることができる.

変数 x_2 が 0 または 1 で場合分けを行う.

$x_2 = 0$ としたときの問題は, 以下のようになる.

$$\text{P1: } \max\{3x_1 + x_3 + 2x_4 \mid 2x_1 + x_3 + 3x_4 \leq 4, \ x_1, x_3, x_4 \in \{0, 1\}.\}$$

線形緩和問題の最適解は $(z; x_1, x_2, x_3, x_4) = (14/3; 1, 0, 1, 1/3)$. $x_2 = 0$ の上で, 変数 x_4 が 0 または 1 で場合分けを行う.

$x_2 = 0, x_4 = 0$ のときの問題は, 以下のようになる.

$$\text{P2: } \max\{3x_1 + x_3 \mid 2x_1 + x_3 \leq 4, \ x_1, x_3 \in \{0, 1\}.\}$$

線形緩和問題の最適解は $(z; x_1, x_2, x_3, x_4) = (4; 1, 0, 1, 0)$. 線形緩和問題の最適解が, 整数解になっている.

$x_2 = 0$ の上で, 変数 x_4 が 0 または 1 で場合分けを行う. $x_2 = 0, x_4 = 1$ のときの問題は, 以下のようになる.

$$\text{P3: } \max\{3x_1 + x_3 + 2 \mid 2x_1 + x_3 \leq 1, \ x_1, x_3 \in \{0, 1\}.\}$$

線形緩和問題の最適解は $(z; x_1, x_2, x_3, x_4) = (7/2; 1/2, 0, 0, 1)$. 現在までに見つかった解 $(z; x_1, x_2, x_3, x_4) = (4; 1, 0, 1, 0)$ より線形緩和問題の目的関数値が低い.

変数 x_2 が 0 または 1 で場合分けを行う. $x_2 = 1$ としたときの問題は, 以下のようになる.

$$\text{P4: } \max\{3x_1 + x_3 + 2x_4 + 4 \mid 2x_1 + x_3 + 3x_4 \leq 1, \ x_1, x_3, x_4 \in \{0, 1\}.\}$$

線形緩和問題の最適解 $(z; x_1, x_2, x_3, x_4) = (11/2; 1/2, 1, 0, 0)$.

$x_2 = 1$ の上で, 変数 x_1 が 0 または 1 で場合分けを行う. $x_2 = 1, x_1 = 0$ のときの問題は, 以下のようになる.

$$\text{P5: } x_3 + 2x_4 + 4 \mid x_3 + 3x_4 \leq 1, \ x_3, x_4 \in \{0, 1\}.\}$$

線形緩和問題の最適解 $(z; x_1, x_2, x_3, x_4) = (5; 0, 1, 1, 0)$. 線形緩和問題の最適解が整数解となった. 現在までに求まった最も良い解に比べ, より良い解 $(z; x_1, x_2, x_3, x_4) = (5; 0, 1, 1, 0)$ が求まった.

$x_2 = 1, x_1 = 1$ のときの問題は, 以下のようになる.

$$\text{P6: } \{x_3 + 2x_4 + 7 \mid x_3 + 3x_4 \leq -1, \ x_3, x_4 \in \{0, 1\}.\}$$

線形緩和問題の解が存在しない (実行不能).

商業コードのほとんどは, 上界として線形緩和を用いている.

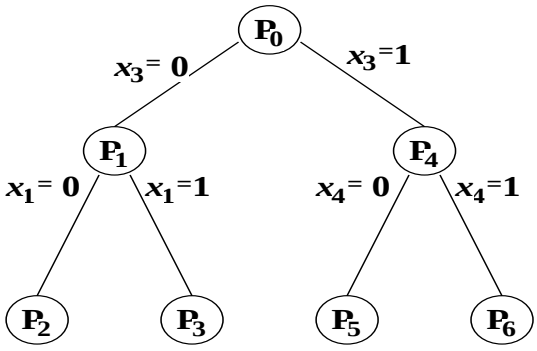

図 1: 分枝木.

5.1 釘付けテスト

変数の多い問題を解く際, 実際は意味の無い変数を前処理を行って見分け, これを取り除くことは非常に重要である. これには, 各変数を固定して（釘付けして）分枝限定法を 1 段行う, 釘付けテストを用いる事が多い. 以下では, ナップサック問題を用いてこれを説明する.

次の最大化ナップサック問題の問題例を用いる．

$$\begin{aligned} \text{maximize} \quad & 9x_1 + 30x_2 + 21x_3 + 15x_4 + 23x_5 + 28x_6 + 7x_7 \\ \text{subject to} \quad & 1x_1 + 8x_2 + 6x_3 + 9x_4 + 15x_5 + 24x_6 + 21x_7 \leq 34, \\ & x_1, x_2, x_3, x_4, x_5, x_6, x_7 \in \{1, 0\}. \end{aligned}$$

線形緩和問題の最適解は $(1, 1, 1, 1, 2/3, 0, 0)$, 最適値は 90.33 である．貪欲解法を用いると， 許容解として $(1, 1, 1, 1, 0, 0, 0)$ が得られ， この許容解の目的関数値が 75 である．以下，この許容解を暫定解，目的関数値 75 を暫定値と呼ぶ．

例えば，変数 x_2 を値 0 に固定すると， 以下の事が分かる．問題は，

$$\begin{aligned} \text{max.} \quad & 9x_1 + 30x_2 + 21x_3 + 15x_4 + 23x_5 + 28x_6 + 7x_7 \\ \text{s.t.} \quad & 1x_1 + 8x_2 + 6x_3 + 9x_4 + 15x_5 + 24x_6 + 21x_7 \leq 34, \\ & x_2 = 0, x_1, x_3, x_4, x_5, x_6, x_7 \in \{1, 0\}. \end{aligned}$$

となり，この問題の線形緩和問題の最適解は， $(1, 0, 1, 1, 1, 3/24, 0)$, 最適値は 71.5 である． 緩和問題の最適値が暫定値未満であることから， 変数 x_2 を 0 とした解は，原問題の解の最適解と成り得ない． ゆえに，変数 $x_2 = 1$ と固定しても良い．（釘付けした際の緩和問題の最適値と， 暫定値が同じであっても， $x_2 = 1$ と固定しても良い． これは， $x_2 = 0$ となる解で最も良いものでも， 暫定解と同じでしかないので， 暫定解が有る事から $x_2 = 0$ の解を探索する必要は無い．）

線形緩和問題の最適解で，値が 1 となった変数については， それを 0 に釘付けして（固定して）線形緩和問題を解き， 以下の手続きを行う．

(1) $x_1 = 0$ と釘付けする: 線形緩和問題の最適解は $(0, 1, 1, 1, 11/15, 0, 0)$, 最適値は 82.87 である．残念ながら，情報は得られない．

(2) $x_2 = 0$ と釘付けする: 線形緩和問題の最適解は $(1, 0, 1, 1, 1, 3/24, 0)$, 最適値は 71.5 である．暫定値以下であることから， $x_2 = 1$ を満たす最適解が有る事が分かる．

(3) $x_3 = 0$ と釘付けする: 線形緩和問題の最適解は $(1, 1, 0, 1, 1, 1/24, 0)$, 最適値は 78.16 である．残念ながら，情報は得られない．

(4) $x_4 = 0$ と釘付けする: 線形緩和問題の最適解は $(1, 1, 1, 0, 1, 4/24, 0)$, 最適値は 86.66 である．残念ながら，情報は得られない．

線形緩和問題の最適解で，値が 0 と 1 の間の値となった変数については， それを 0 と 1 に釘付けして（固定して）線形緩和問題を解き， 以下の操作を行う．

(5–0) $x_5 = 0$ と釘付けする: 線形緩和問題の最適解は $(1, 1, 1, 1, 0, 5/12, 0)$, 最適値は 86.66 である．残念ながら，情報は得られない．

(5–1) $x_5 = 1$ と釘付けする: 線形緩和問題の最適解は $(1, 1, 1, 4/9, 1, 0, 0)$, 最適値は 89.66 である．残念ながら，情報は得られない．

線形緩和問題の最適解で，値が 0 となった変数については， それを 1 に釘付けして（固定して）線形緩和問題を解き， 以下の操作を行う．

(6) $x_6 = 1$ と釘付けする: 線形緩和問題の最適解は $(1, 1, 1/6, 0, 0, 1, 0)$, 最適値は 70.5 である．暫定値以下であることから， $x_6 = 0$ を満たす最適解が有る事が分かる．

(7) $x_7 = 1$ と釘付けする: 線形緩和問題の最適解は $(1, 1, 4/6, 0, 0, 0, 1)$, 最適値は 74 である．暫定値以下であることから， $x_7 = 0$ を満たす最適解が有る事が分かる．

上記より，原問題は次の問題に縮小されることが分かる．

$$\begin{aligned} \text{max.} \quad & 9x_1 + 21x_3 + 15x_4 + 23x_5 + 30 \\ \text{s.t.} \quad & 1x_1 + 6x_3 + 9x_4 + 15x_5 \leq 26, \\ & x_2 = 1, x_6 = x_7 = 0, x_1, x_3, x_4, x_5 \in \{1, 0\}. \end{aligned}$$

釘付けテストは，分枝限定法の前処理として行う事ができる．釘付けテストに用いる暫定解は， 適当な発見的解法を用いて求められる．厳密解法の効率化のためにも， 多少時間をかけてより良い暫定解を求め， これを釘付けテストに用いる方が良い．

分枝限定法において，釘付けテストを各子問題毎に用いる事も可能であるが，計算時間が増大するため，前処理として 1 回のみ行う事が多い．

6 無料ソフトウェア

- `http://www.saitech-inc.com/math.htm#sopt`
  SOPT demonstration software
- `http://www.lindo.com/`
  free download versions of LINDO software
- `http://www.dash.co.uk/`
  free student edition of XPRESS-MP
- `ftp://ftp.es.ele.tue.nl/pub/lp_solve`
  public domain soft lp-solve (windows 版： lp20w95.zip)
  簡易日本語マニュアル `http://www.misojiro.t.u-tokyo.ac.jp/~tomomi/soft/manual_LPsolve.html`

参考文献

[1] 茨木俊秀, 「組合せ最適化 – 分枝限定法を中心として –」, 産業図書, 1983.

[2] 伊理正夫, 今野浩, 刀根薫 監訳, 「最適化ハンドブック」, 朝倉書店, 1995.

[3] 岩田茂樹, 「NP 完全問題入門」, 共立出版, 1995.

[4] 今野浩, 「整数計画法」, 産業図書, 1981.

[5] 今野浩, 鈴木久敏 編, 「整数計画法と組合せ最適化」, 日科技連, 1982.

[6] 山本芳嗣, 久保幹雄, 「巡回セールスマン問題への招待」, 朝倉書店, 1997.

[7] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. Shmoys, *The Traveling Salesman Problem*, Wiley, 1985.

[8] S. Martello and P. Toth, *Knapsack Problems : Algorithms and Computer Implementations*, Wiley, 1990.

[9] P. B. Mirchandani and R. L. Francis, *Discrete Location Theory*, Wiley, 1990.

[10] G. Reinelt, *The Traveling Salesman : Computational Solutions for TSP Applications*, Springer Verlag, 1994.