Sequential Competitive Facility Location

Kodai Adachi

2024 年 10 月 10 日

1　整数緩和問題へのアプローチ

- 本稿では、分枝限定法、切除平面法、分枝カット法のアイデアを説明するため、次の問題 (P) を例にとる。
- 問題 (P) は以下で定義する：

$$
\begin{aligned}
\text{(P) 最小化} \quad & -x_1 - x_2 \\
\text{条 件} \quad & -2x_1 + 3x_2 \leq 6, \\
& 10x_1 + 4x_2 \leq 37, \\
& 2x_1 - 4x_2 \leq 5, \\
& x_1 \geq 0, x_2 \geq 0, \\
& x_1, x_2 \in \boldsymbol{Z}.
\end{aligned}
\tag{1.1}
$$

- 図 1 に示されるように、黒丸（●）が問題 (P) の実行可能解を表す。最適解は $(x_1, x_2) = (2, 3)$ であり、最適値は -5 である。

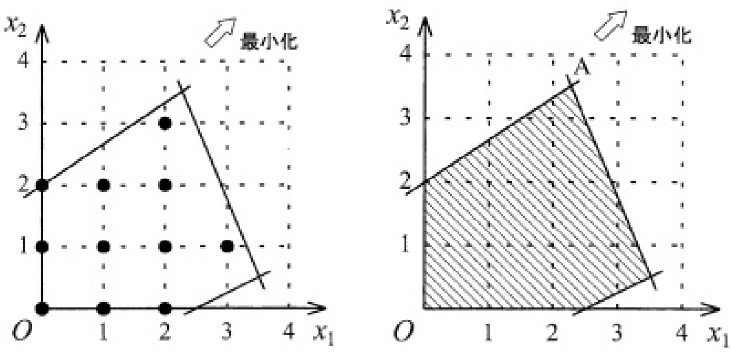


図 1　(P)　　図 2　(P)

- 分枝限定法、切除平面法、分枝カット法では、問題 (P) 自身を解く代わりに「緩和問題」(relaxation problem) を解く。
- 本稿では、問題 (P) の整数制約 $x_1, x_2 \in \mathbb{Z}$ を除去した「線形計画緩和問題」(linear programming relaxation problem, 以下 LP 緩和問題と呼ぶ) を扱う。これを $(\overline{\mathrm{P}})$ と表す。
- 線形計画緩和問題 $(\overline{\mathrm{P}})$ の目的関数および制約条件は以下の通りである。

$$
\begin{aligned}
(\overline{\mathrm{P}}) \text{ 最小化} \quad & -x_1 - x_2 \\
\text{条 件} \quad & -2x_1 + 3x_2 \leq 6, \\
& 10x_1 + 4x_2 \leq 37, \\
& 2x_1 - 4x_2 \leq 5, \\
& x_1 \geq 0, x_2 \geq 0.
\end{aligned}
\tag{1.2}
$$

- LP 緩和問題の解は図 2 のようになり、最適解は点 $A\left(\frac{87}{38}, \frac{67}{19}\right) = (2.28\cdots, 3.52\cdots)$ であり、最適値は $-\frac{221}{38} = -5.81\cdots$ となる。
- 線形計画緩和問題に関して、以下の重要な事実が成り立つ。

(R_1): 常に「 $(\overline{\mathrm{P}})$ の最適値 $\leq$ (P) の最適値」が成り立つ。

(R_2): $(\overline{\mathrm{P}})$ の最適解が (P) の実行可能解である場合、すなわち x_1, x_2 が整数値を取るとき、それは (P) の最適解である。

(R_3): $(\overline{\mathrm{P}})$ に実行可能解が存在しなければ、(P) にも実行可能解は存在しない。

- (R_2) より、$(\overline{\mathrm{P}})$ の最適解が整数値を取る場合、(P) は解けたことになる。しかし、この例では $(\overline{\mathrm{P}})$ の最適解は整数値を取っていない。
- このような場合に、問題 (P) の最適解を求めるために分枝限定法、切除平面法、分枝カット法が用いられる。

2 分枝限定法 (branch and bound)

- 分枝限定法では、問題 (P) を分割し、新たな子問題を複数生成する。
  - 一般的な方法として、整数値をとらない変数（例: $x_1 = 2.28\cdots$）を選び、以下の 2 つの子問題を生成する。
    - 子問題 (P_1): 問題 (P) に制約 $x_1 \leq 2 = \lfloor 2.28 \rfloor$ を加えたもの。
    - 子問題 (P_2): 問題 (P) に制約 $x_1 \geq 3 = \lceil 2.28 \rceil$ を加えたもの。
  - これを「**分枝操作 (branching)**」と呼び、図 3 に示されるように、(P) の実行可能解はこの分割により得られる。
  - (P_1) または (P_2) の最適解が、元の問題 (P) の最適解となる。

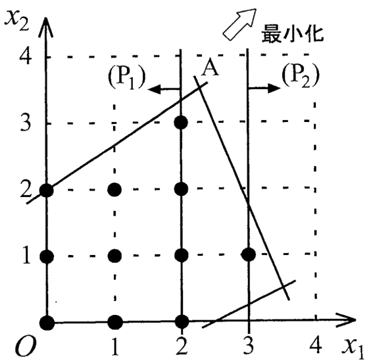
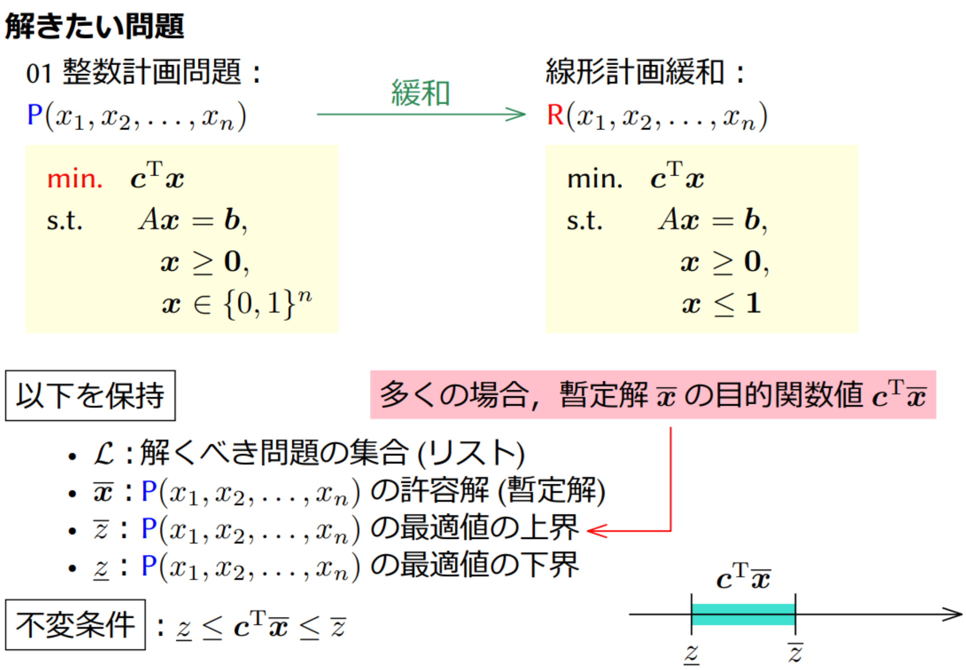


図 3　分枝操作

- 分枝限定法では、分枝操作を繰り返して最適解を求めるが、(R_1), (R_2), (R_3) により、子問題の分枝を終了することができる。この操作を「**限定操作 (bounding)**」と呼ぶ。
- 特定の子問題 (P_k) について、LP 緩和問題 $(\overline{\mathrm{P}_k})$ を解き、条件 (R_2) または (R_3) が成立すれば、(P_k) は解けたことになる。
- また、条件 (R_1) より、以下の事実が成り立つ。
  - (R_1'): 既知の (P) の実行可能解があり、その値を z とする。このとき「$z \leq (\overline{\mathrm{P}_k})$ の最適値」であるならば、(P_k) の中に z よりも良い解は存在しない。
- よって、条件 (R_1') が成立すれば、(P_k) を限定することができる。
- 以上が分枝限定法の概略である。
- 以下岡本吉央先生の授業資料から抜粋



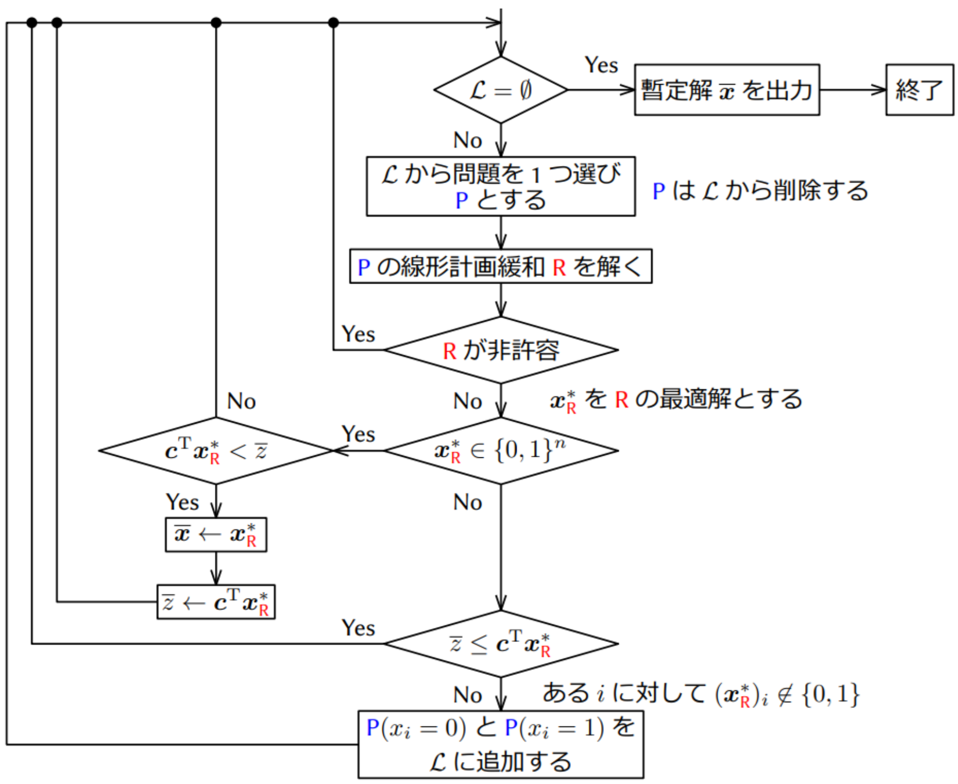


図 1　branch and bound のフロー図

3　切除平面法 (cutting plane method)

- $(\overline{\mathrm{P}})$ の最適解が整数値をとらない場合、分枝限定法では (P) を分割して子問題を生成する。
- 一方、切除平面法では、問題 (P) に新たな不等式を追加することで解を改善することを試みる。
- 用語の定義:
  - 妥当不等式（valid inequality）：（MIP）のすべての実行可能解 x に対して $p^{\top}x \leq q$ が成り立つ不等式。
  - 切除平面（cutting plane, cut）：妥当不等式 $p^{\top}x \leq q$ を加えることで、$\{x \in \boldsymbol{R}^n \mid p^{\top}x \leq q, Ax \leq b\} \subsetneq \{x \in \boldsymbol{R}^n \mid Ax \leq b\}$ が成り立つ不等式。

- 図 4 に示されるように、LP 緩和問題に対してカットを追加すると、最適解が変化する。例では、最適解が点 A から点 B へ移動した。

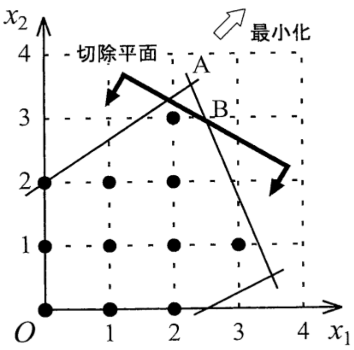


図 4　カットの例

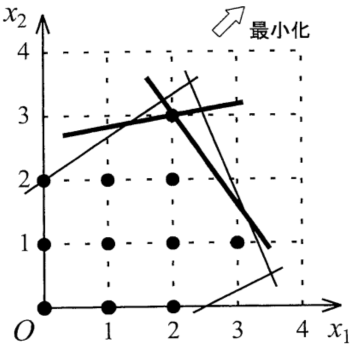


図 5　カットの例（2）

- このように、最適解を削除するカットを繰り返し追加することで、最終的に（P）の最適解に至ることが期待できる。
- 実際、図 5 に示すように、最適解 $(x_1, x_2) = (2, 3)$ が明確に現れるようにカットを追加することで、LP 緩和問題を解き、（P）の最適解を得ることができる。
- R. Gomory は 1958 年の論文で、全整数計画問題に対する小数カット（fractional cut）を提案し、有限回のカットで最適解が得られることを示した。
- さらに、1960 年の論文で、Gomory は混合整数計画問題に対する混合整数カット（mixed-integer cut）を提案し、ある条件下で有限収束性を示した。
- この強力な理論により、整数計画の研究は大きく発展したが、実際のコンピュータ実装では切除平面法の収束が遅いことや数値誤差の問題があり、解法の主流は分枝限定法へと移行した。

4　分枝カット法 (branch and cut)

- 分枝限定法が解法の主役となり、ほとんどの商用ソフトウェアが分枝限定法を採用するようになった。
- 緩和問題の最適解を削除するカットを加えると、緩和問題の性質 (R_1')、(R_2)、(R_3) が成立しやすくなるという利点がある。
- 図 4 は、カットを加えることで下界値が上昇し、(R_1') が成立しやすくなる様子を示している。
- このことから、分枝限定法に切除平面法を組み込むことは自然な考えであり、実際 1970 年代に発行された本や論文でもその可能性が示唆されている。
- しかし、切除平面法はしばらくの間忘れ去られることとなった。これは、当時の期待が高すぎたために失望が大きく、無視されるようになったと考えられる。
- 切除平面法が実用的でなかった理由の一つに、当時は現在のような強力な線形計画ソルバー（LP ソルバー）がなく、理論がコンピュータ実装に対して理論が先行しすぎていたことが挙げられる。
- 現在の LP ソルバーの出現やコンピュータ環境の発展により、当時の理論が実用的に意味を持つようになった。
- これにより、分枝限定法の精密化が進み、1960 年代以降に提案されたさまざまな工夫が、現在の LP ソルバーに無理なく組み込まれるようになった。
- 最後に、（MIP）に対する分枝カット法の手続きを図 6 に示す。
- 各子問題 (MIP_k) についてカットの生成を試みるが、生成されたカットが（MIP）に対して妥当である場合、これを global カットと呼ぶ。
- （MIP）に対して妥当な不等式は、すべての子問題に対しても妥当であるため、これらは分枝カット法の実行中に保持される。

Algorithm 1: 分枝カット法

1. **初期化**

   L を未処理の子問題リストとする。$L = \{(\mathrm{MIP})\}$ とし、暫定値を $z = -\infty$ として初期化する。

2. **終了判定**

   $L = \emptyset$ なら暫定解を出力し終了。

3. **子問題の選択**

   子問題 (MIP_k) を L から選択し、L から消去する。(主にスタックが用いられるらしい)

4. **子問題の評価**

   global カットを (MIP_k) に追加する。

   (a) LP 緩和問題を解く。(R_1') または (R_3) が成立した場合、Step 2 へ。(R_2) が成立した場合、暫定解を更新し、Step 2 へ。

   (b) 緩和問題の最適解をカットするカットを求める。もし見つかれば、(MIP_k) に追加し、global カットは保持しておく。Step 4-1 へ。見つからなければ Step 5 へ。

5. **分枝操作**

   子問題 (MIP_k) を複数の子問題に分割し、L に追加する。Step 2 に戻る。