4.2 Stochastic programmes

この章では、不確実性をモデル化する確率計画法について紹介する

確率計画法 (Stocastic programmes)

確率計画法は、不確実性をモデル化する手法の一つ

- あくまで線形計画法の特別な一種であり、別の手法というわけではない
- 現実的な状況では、以下の様に定式化に用いるデータが不確定な状況がある. 確率計画法では、ある特定の方法でででその不確実性をモデル化する. ロバスト最適化の一種とも言える.
    i. 完全に正確なデータを取得できない
    ii. そもそも、（モデル化に必要な）イベントが発生していない
        - 例: Section 4.1 の多期間問題

ロバスト最適化と確率計画法

普通、ロバスト最適化は不確実性が定量化できるか否かに関係なく適用可能. 本章で説明する確率計画法では、不確実性が定量化できる場合に限定される

- データの不確実性に対する解の安定性を評価するため
  - 参考: Section 6.3 で感度分析を取り扱う
- 完全にリスクを回避するようなアプローチもあるが、保守的になりすぎる傾向がある

不確実性を含むモデル

ここでは、以下のような状況を考える

1. stage1 の意思決定では、一部のデータしか使えない
2. stage1 の意思決定の結果と、追加のデータによって stage2 の意思決定がなされる

例:

1. 需要が決まる前に生産計画を建てなければいけない (stage1)
2. 生産と需要のバランスによって、過剰生産分を低価格で売ったり、追加生産したりする (stage2)

機会制約によるモデル化

Section 3.3 で説明した機会制約を使うことで不確実性をモデル化できる.
不確実な情報の取る値が、ある離散分布に従う場合、以下のようにする.

1. 可能性のある値それぞれに対応する変数を用意する
2. 目的関数を、コストそのものではなく、**期待コスト**の最小化とする

モデル化の具体例

stage1の変数を $x_1, x_2, \ldots, x_n$、stage2に対応する変数を $y_1, y_2, \ldots, y_n$ とする.可能な需要レベルが$d_j^{(1)}, d_j^{(2)}, \ldots, d_j^{(m)}$だった場合、それぞれの確率を表す重み p_r で足し合わせたものを目的関数にする(y_jが過剰生産量、z_jが追加生産量を表す)

$$\text{Minimize} \quad \sum_j c_j x_j + \sum_r p_r \left(\sum_j e_j y_j^{(r)} + \sum_j f_j z_j^{(r)} \right)$$

$$\text{subject to} \quad \sum_j a_{ij} x_j \leq b_i \text{ for all production constraints } i$$

$$\begin{aligned} x_j - y_j^{(r)} + z_j^{(r)} &= d_j^{(r)} \\ x_j, y_j^{(r)}, \ z_j^{(r)} &\geq 0 \end{aligned} \qquad \text{for all } j \text{ and } r$$

確率モデルと従来のモデルの違い

- 従来のモデルでは、未来のデータを点推定した値がモデルに組み込まれる
- 確率計画法によるモデル（確率モデル）では、未来のデータを確率値で重み付けしたリストで表現する
    - シナリオ数に比例して変数が増えるので、非常に大きなモデルになりがち
- 確率モデルでは、従来のモデルと同等に**良い**解が得られるわけではない
    - リスクを減らした、より現実的な解が求まる

4.3 Decomposing a large model

この章では、大きな問題の分解方法、特に Dantzig-Wolfe 分解というアルゴリズムについて説明する

モデルの分解

構造化モデル（複数のサブモデルの組合わせで構成されているようなモデル）の分解を議論する理由として、以下の様な背景がある

1. 構造化モデルの最適解は、サブモデルの最適解と何らかの関係があるはず．　この構造をうまく利用して、効率的に解を探索できる可能性がある
2. マルチプラントモデルのように、サブモデルのそれぞれが実際の組織と対応する場合には、モデルの分解手順が、分散計画経済と対応している

モデルの分解手法

Section4.1 のマルチプラントを例に挙げると、以下の2つの分解方法が考えられる

1. **decomposition by allocation**
    - リソースの配分割合を見つける問題と、各工場内での最適化問題に分解
    - Rosen (1964)
2. **decomposition by pricing**
    - リソースの **内部的な価格(internal price)** という概念を取り入れて、適切な価格を見つける問題と、各工場内での最適化問題に分解
    - Dantzig-Wolfe 分解 (Dantzig (1963))

[再掲] マルチプラントモデル

Section 4.1 で取り扱ったモデルは以下の通り

$$
\begin{aligned}
&\text{Maximize} && \text{Profit} && 10x_1 + 15x_2 + 10x_3 + 15x_4 \\
&\text{subject to} && \text{Raw} && 4x_1 + 4x_2 + 4x_3 + 4x_4 \le 120, \\
& && \text{Grinding A} && 4x_1 + 2x_2 \le 80, \\
& && \text{Polishing A} && 2x_1 + 5x_2 \le 60, \\
& && \text{Grinding B} && 5x_3 + 3x_4 \le 60, \\
& && \text{Polishing B} && 5x_3 + 6x_4 \le 75, \\
& && && x_1, x_2, x_3, x_4 \ge 0.
\end{aligned}
$$

内部価格 (internal price)

Section 4.1 の問題に、内部価格 (internal price) の概念を導入する. 原材料 1kg あたりの価格を p だと思うと、製品1単位に、4kgの原材料が必要なため、最大化したい利益はそれぞれ以下のとおり

- **Profit A**: $(10-4p)x_1+(15-4p)x_2$
- **Profit B**: $(10-4p)x_3+(15-4p)x_4$

Dantiz-Wolfe 分解では、p が小さいと、原材料をたくさん使おうとするし、 p が大きいと、あまり使わないようになる. ぎりぎり原材料を使い切るような p を見つけようとする

疑問

工場が3つあった場合でも変数1つの追加で十分なんだろうか?

モーダル定式化 (modal formulation)

サブモデルAの実行可能領域は以下のとおり. 全体の問題の実行可能解は、当然この領域に含まれているので、各頂点の線形結合であらわせる. サブモデルBも同様

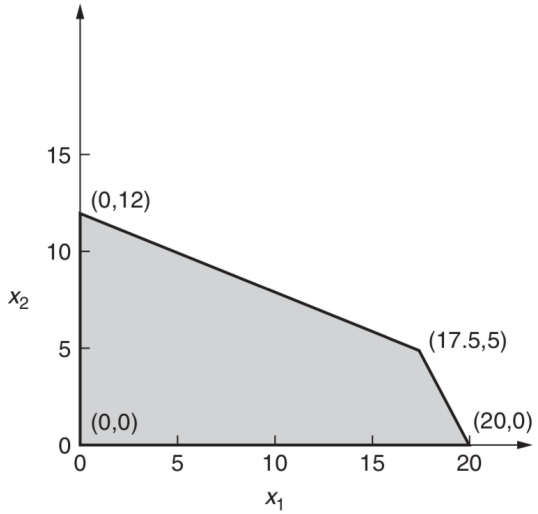

Figure 4.6

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \lambda_{11} \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \lambda_{12} \begin{pmatrix} 20 \\ 0 \end{pmatrix} + \lambda_{13} \begin{pmatrix} 17.5 \\ 5 \end{pmatrix} + \lambda_{14} \begin{pmatrix} 0 \\ 12 \end{pmatrix},$$
$$\lambda_{11} + \lambda_{12} + \lambda_{13} + \lambda_{14} = 1.$$

マスターモデル (master model)

前式をx_iに代入したものを **マスターモデル (master model)** もしくは主問題と呼ぶ

- λ_{ij} の係数を **proposal** と呼び、proposal の適切な混合具合を求める
- 制約数は、共通行の数 + サブモデル数で、基本的に元のモデルよりも少ない
- 変数の数は、各サブモデルの頂点数なので、(各サブモデルの制約の組み合わせになるので)一般に非常に多い. ただし、現実的にはほとんどの変数は0になる.

$$
\begin{array}{lrrrrrrrrrrrrr}
\text{Maximize} & & & & & & & & & & & & & \\
\text{Profit} & & & 200\lambda_{12} & + & 250\lambda_{13} & + & 180\lambda_{14} & + & 120\lambda_{22} & + & 165\lambda_{23} & + & 187.5\lambda_{24} \\
\text{subject to} & & & & & & & & & & & & & \\
\text{Raw} & & & 80\lambda_{12} & + & 90\lambda_{13} & + & 48\lambda_{14} & + & 48\lambda_{22} & + & 56\lambda_{23} & + & 50\lambda_{24} \leq 120 \\
\text{conv 1} & \lambda_{11} & + & \lambda_{12} & + & \lambda_{13} & + & \lambda_{14} & & & & & & = 1, \\
\text{conv 2} & & & & & & & & \lambda_{21} & + & \lambda_{22} & + & \lambda_{23} & + \lambda_{24} = 1.
\end{array}
$$

解法(列生成法)

ほとんどの変数 λ_{ij} が 0 にできることを利用して、 マスターモデルを解く.

1. 適当な proposal を残して、それ以外を 0 だと思った マスターモデル(restricted master model, 制限付きマスターモデル) を解く
2. その結果から、内部価格 p を求める
3. 内部価格 p のもとでサブモデルをとき、最適解に対応する頂点を得る
4. 得られた頂点を制限付きマスターモデルに追加する
5. 頂点が追加されなくなるまで、上記を繰り返す

疑問 : テキストでは λ_{ij} ではなく、ほとんどの proposal が最適解で 0 になる. と書かれているが、 proposal 自体は定数では?

初期化

初期値として、$\lambda_{11}, \lambda_{13}, \lambda_{21}, \lambda_{24}$ に対応する proposal だけを残した制限付きマスターモデルを書き下して、最適解を求める.

$$
\begin{array}{lrrrrrrr}
\text{Maximize} & \text{Profit} & & 250\lambda_{13} & & +187.5\lambda_{24} & & \\
\text{subject to} & \text{Raw} & & 90\lambda_{13} & + & 50\lambda_{24} & \leq & 120, \\
& \text{conv 1} & \lambda_{11} + & \lambda_{13} & & & = & 1, \\
& \text{conv 2} & & & \lambda_{21} + & \lambda_{24} & = & 1.
\end{array}
$$

制約付きマスターモデルを解く

Raw 制約の shadow price を求める.

- shadow price は、制約の右辺を変化させた時の目的関数の変化量
  - 今回の場合、最適解では $\lambda_{24} = 1$ でこれ以上増やせないので、 右辺の増加分 λ_{13} を増やすことになる. よって、$250/90 \approx 2.78$
- サブモデルの支店からすると、 shadow price はまさに内部価格 p に相当する

サブモデルを解く

内部価格 p を、制限付きマスターモデルで得られた shadow price に設定して、サブモデルを解く

- Profit A: $-1.12x_1 + 3.18x_2$
  - 最適解: $(x_1, x_2) = (17.5, 5)$ で、 λ_{13} に対応
- Profit B: $-1.12x_1 + 3.18x_2$
  - 最適解: $(x_3, x_4) = (0, 12.5)$ で、 λ_{24} に対応

制限付きマスターモデルに頂点を追加して解く

$\lambda_{11}, \lambda_{13}, \lambda_{21}, \lambda_{24}$ に、 λ_{13} を追加して、制限付きマスターモデルを解く (サブモデル B から得られた λ_{24} はすでに含まれているので追加しない)

$$
\begin{array}{lll}
\text{Maximize} & \text{Profit} & 250\lambda_{13} + 180\lambda_{14} + 187.5\lambda_{24} \\
\text{subject to} & \text{Raw} & 90\lambda_{13} + 48\lambda_{14} + 50\lambda_{24} \leq 120, \\
& \text{conv 1} & \lambda_{11} + \lambda_{13} + \lambda_{14} = 1, \\
& \text{conv 2} & \lambda_{21} + \lambda_{24} = 1.
\end{array}
$$

- 最適解: $\lambda_{13} = 0.52$、$\lambda_{14} = 0.48$、$\lambda_{24} = 1.$
- Raw制約の shadow price: 1.67

サブモデルを解く

新たに得られた内部価格 $p = 1.67$ を用いて、再度サブモデルAを解く

- サブモデルAの最適解は $(x_1, x_2) = (17.5, 5)$ で、新たな proposal は得られない
- サブモデルBの最適解は $(x_3, x_4) = (0, 12.5)$ で、新たな proposal は得られない

よって、制限付きマスターモデルはこれ以上更新されない.

そこで、現時点の最適解 λ_{ij} から x_i をもとめると、$x_1 = 9.17$、$x_2 = 8.33$、$x_4 = 12.5$ (直接解いた場合と同じになる)

疑問: この停止条件で、本当に最適解が得られていることがどのように保証されているか

制限付きマスターモデルとサブモデル

以下の2種類のモデルを利用した

1. サブモデル
    - 各部分問題の詳細が含まれる
2. 制限付きマスターモデル
    - 組織全体を表すモデル
    - 全体モデルと違い、サブモデルの詳細には関知しない
    - 共通制約がすべて含まれる

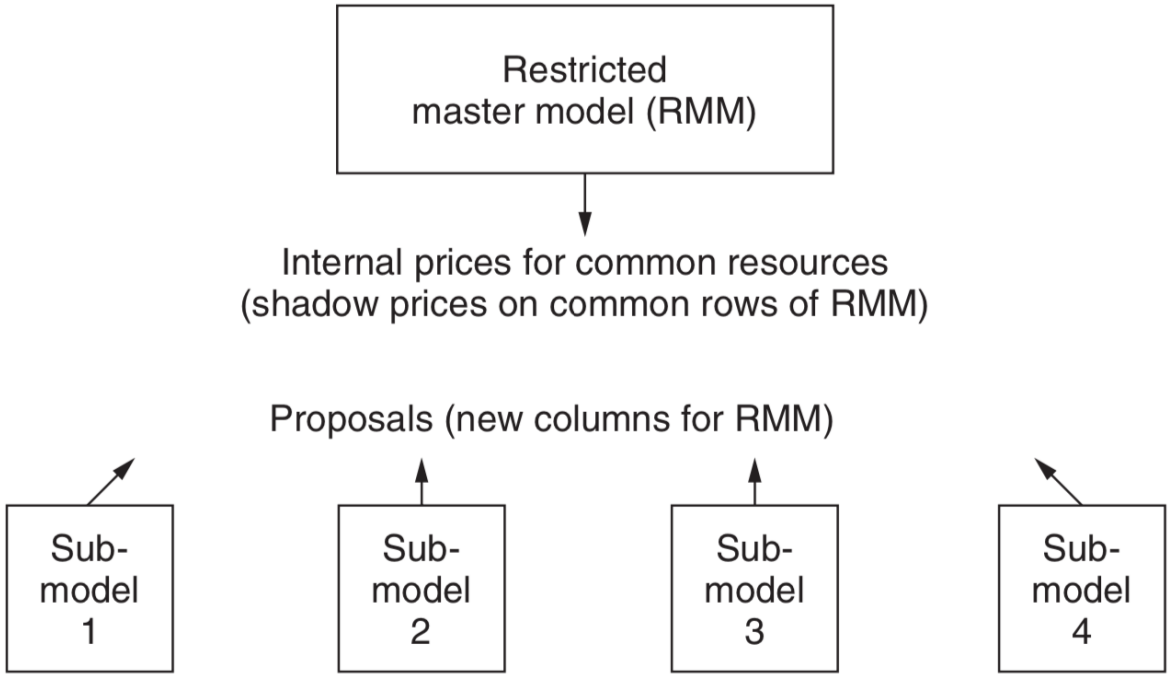

Figure 4.9

モデルを分解する利点と現状

1. 分解により、巨大な全体モデルを直接解く必要がなくなる
2. 制限付きマスターモデルは、各サブモデルの詳細に関与しないため、例えば、各工場で独自のアルゴリズムで最適化することもできる
3. 分散計画経済的な観点では、個別に最適化することで全体として最適な解がだせることを示しているので、研究対象となっている
4. 計算上は、分解しないで直接解いたほうが早いことがおおく、成功例は限定的
    - おなじモデルが頻繁に再利用される場合、やモデルが非常に大きくて共通行が少ない場合は試す価値がある可能性がある
    - Beale et al (1965)、 Ho and Loute (1981)、Lasdon (1970)が参考になる
5. 複数台のコンピューターをつなげて、マスターモデルをヘッドオフィスで、サブモデルを各工場で、ということもできるのかもしれない（すでにやっていそう）