組合せ最適化を体系的に知って Pythonで実行してみよう

(株) 構造計画研究所 斉藤努

1 はじめに

これから、誰でも**組合せ最適化**が使えるようにご案内したいと思います。

組合せ最適化を使うコツは、全体像を理解することです。どんな要素があるのか、それらがどのような関係にあるのかを知ることにより、解きたい問題についてアプローチできるようになります。

Python を使うことにより、簡単に、様々な最適化ができます。実際のコードを見ながら、あとで確認しましょう。最初に、身近な最適化の例から見ていきます。

> 先日、あなたが実家に帰ると、お土産に野菜を持って帰るよう言われました。東京の野菜は高いので、「儲かった」と喜びましたが、量が多すぎます。せいぜい 5kg しか持って帰れないとします。(宅配便は使えないことにしてください) また、野菜は切ったりすると傷むので、そのまま持って帰ることにします。

あなたは「1kg あたりの販売価格の高いものから選んでいこう」と考えました。実は、これは (そこそこいい方法ですが) 最適な方法ではありません。本当に最適な方法は、どうすればわかるでしょうか？

数理最適化を使えば、このような問題を解くことができます。

数理最適化では、数式を使って、問題を**数理モデル**で表します。先ほどの例は、

$$
\begin{array}{lll}
\text{最大化} & \sum_i p_i x_i & p_i\text{:販売価格} \\
& \sum_i w_i x_i \leq 5 & w_i\text{:重さ} \\
& \forall x_i \in \{0,1\} & x_i\text{:持って帰るかどうか}
\end{array}
$$

という数理モデルになります。数理最適化は、**連続最適化**と**組合せ最適化**に分けられます。

連続最適化: 連続要素のみ
組合せ最適化: 連続以外の離散要素を含む

今回は、組合せ最適化に焦点を当ててご説明します。

このように数理モデルを表すことを**定式化**するといいます。

2 定式化

レッスン 1

組合せ最適化では、数理モデルを定式化する。

定式化をするには、3 つの要素を決める必要があります。

1. 何を決めたいのか？ 「持って帰る野菜を決めたいです」
2. どうなるとうれしいのか？ 「持って帰る野菜の販売価格の合計が高くなるとうれしいです」
3. 守らないといけないことは？ 「持って帰る野菜を 5kg 以下にします」

この 3 つをそれぞれ、**変数、目的関数、制約条件**とよびます。先ほどの例で見てみましょう。

目的関数: $\sum_i p_i x_i$ → 最大
制約条件: $\sum_i w_i x_i \leq 5$
変数: $\forall x_i \in \{0, 1\}$

定式化できるようにするためには、慣れが必要です。今回は、いくつかの例を後で見ることにします。詳しく勉強する場合は、書籍「今日から使える!組合せ最適化」などを参考にしてください。

数理モデルから最適な答えを探すのは、(賢い人が作ってくれた) ツールを使います。このツールのことを最適化ソルバー、略して**ソルバー**とよびます。つまり、問題を（誰でもわかるように）表すだけで、ソルバーを使えば解 (答えのこと) を出すことができます。

最近では、このソルバーが使いやすく、性能がたいへんよくなってきました。みんなが、最適化が試せるようになってきているのです。

3 標準問題

先ほどの例の他に、どんな例があるでしょうか？あなたは実家に帰るときに、web で乗り換え駅を探索したかもしれません。

あなたの家の最寄り駅から、実家の最寄り駅までの最短路 (あるいは最安路) を探したい。

これも最適化問題です。最適化問題は、いろいろなところにたくさんあります。でも、問題ごとに定式化するのは、大変ですよね。実は、別々の問題でも、同じ定式化になることがよくあります。そうすると、わざわざ定式化する必要がなくなり、便利ですね。世の中によくある問題を**標準問題**とよぶことにします。

レッスン 2

標準問題を理解する。

関連する標準問題を集めて、標準問題クラスという枠組みを作ります。標準問題クラスは、7 つあります。

標準問題クラス	標準問題
グラフ・ネットワーク問題	最小全域木問題
	最短路問題
	最大流問題
	最小費用流問題
経路問題	運搬経路問題
	巡回セールスマン問題
集合被覆・分割問題	集合被覆問題
	集合分割問題
切出し・詰込み問題	ナップサック問題
	n 次元パッキング問題
割当・マッチング問題	一般化割当問題
	最大マッチング問題
配置問題	施設配置問題
スケジューリング問題	ジョブショップ問題
	スケジューリング問題

URL

野菜の選び方は**ナップサック問題**、乗り換え駅探索は、**最短路問題**といいます。標準問題は、よく研究もされているので、多くの場合、効率的な解法があります。あるいは、定式化がされているので、すぐ解くことができます。あとで、やってみましょう。

4　数理問題

最近、私がやっているコンテナの仕事のお話しをします。

世界中の人たちが、いろいろなものを安く買えるのはコンテナ輸送のおかげです。中国などで生産したものを日本やアメリカやヨーロッパに、大量に安く運べるからです。でも、空のコンテナが、どんどんたまります。また中国に戻さないといけません。いつ、どこからどこに戻すかを決めるのが、**最小費用流問題**になります。

ところが、最小費用流問題で表せない制約条件もあります。1 つが、カボタージュとよばれるものです。カボタージュというのは、国内のみの輸送を自国業者のみに規制することです。

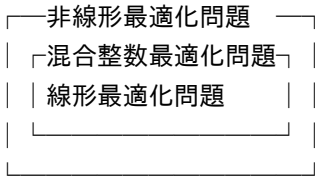
標準問題で対応できない場合は、数理モデルをそのまま扱います。数理モデルで表された問題を**数理問題**とよびます。標準問題は、数理問題として見ることもできます。

レッスン 3

数理問題を理解する。

数理問題には、何があるのでしょうか？

数理問題の違いは、解法 (アルゴリズム) の違いによります。この解法の種類によって、解きやすさが全く違います。また、数理問題ごとにソルバーも異なることがあります。なるべく、簡単に解ける数理問題に持って行けるかが、重要になります。数理問題は、親子関係になります。親から見ていきましょう。



(非線形最適化問題でもある)

(非線形最適化問題でも混合整数最適化問題でもある)

全ての最適化問題は、非線形最適化問題です。なぜ、わざわざ非線形最適化というのでしょうか？それは、線形最適化問題を非線形最適化として扱うのは非効率的だからです。非線形最適化という言葉には、非線形な目的関数か非線形な制約条件があることを暗黙に示唆しています。

線形最適化問題

数理問題で最も解きやすいのは、線形最適化問題です。最小費用流問題も線形最適化問題です。変数が**連続変数**で、目的関数と制約条件が線形 (1 次式) で表される問題です。これは、かなり大きな問題でも解けるようになってきました。

混合整数最適化問題

目的関数と制約条件は線形ですが、**離散変数**も許した問題を混合整数最適化問題とよびます。ナップサック問題や最短路問題も混合整数最適化問題です。

実務でよく現れる問題で、私の扱っている問題の多くは、これになります。難しい問題ですが、最近では、定式化や問題の規模次第で解けるようになってきました。

「混合整数最適化問題が解けるようになってきた」これこそが、最適化のしきいが下がってきた要因といえるでしょう。数理最適化初心者にとって、1 つの目標が、簡単な混合整数最適化問題の定式化することになるでしょう。

ただし、実務で、混合整数最適化問題を扱うのは難しく、いろいろ工夫が必要になることもあります。

非線形最適化問題

非線形最適化は、さらに難しい問題です。しかし、小規模であれば、現状でも解けるようになってきました。また、非線形最適化の中でも非線形凸最適化は、大規模なものでも扱えますが、ここでは省略します。

この 3 種類の例は後ほど見てみます。

5　ビジネスでの実例

事例	標準問題
空箱の輸送コスト最適化	最小費用流
ビークル間連携配送最適化	運搬経路
船舶スケジューリング最適化	集合被覆
店舗シフトスケジューリング	スケジューリング
3 次元パッキング最適化	n 次元パッキング
避難施設配置最適化	施設配置
大規模データベース配置	一般化割当
バス仕業作成最適化	最大マッチング

6　Python による実行例

レッスン 4

実際にやってみよう。

ナップサック問題

最短路問題

最小費用流問題

最小費用流問題 (定式化)

ナップサック問題 (定式化)

ポートフォリオ最適化問題