レポート提出票

科目名: 情報工学実験2

実験テーマ: 実験テーマ1 数理計画法

実施日: 2020年 10月 19日

学籍番号: 4619055

氏名: 辰川力駆

共同実験者:

1　実験の要旨

本実験では実際に最適化問題を定式化をしながら解の検討をすることにより、最適化問題への理解を深める。

2　実験の目的

最適化問題を解くための数理的手法は数理計画法と呼ばれる。その数理計画法を現実問題に適用する際に役立つ基礎力を養うことを目的とする。

3　実験の原理 (理論)

3.1　用語の説明

最適化問題は、数学的には、解集合と呼ばれる集合 S と目的関数と呼ばれる関数 $f : S \to \boldsymbol{R}$ が与えられたとき、$f(\boldsymbol{x})$ を最小にする $\boldsymbol{x}$ を求める問題であり、これを、

$$\text{Minimize } f(\boldsymbol{x}) \text{ subject to } \boldsymbol{x} \in S$$

とかく。最適化問題の答え、つまり、$\boldsymbol{S}$ の任意の $\boldsymbol{x}$ に対し、$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ を満たす $\boldsymbol{x}^*$ を最適解と呼ぶ。また、$\boldsymbol{x}^*$ が局所的に最小なら、つまり、

$$\exists \epsilon > 0,\ \forall \boldsymbol{x} \in N_\epsilon(\boldsymbol{x}^*),\ f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$$

が成り立つなら、$\boldsymbol{x}^*$ を局所的最適解と呼ぶ。ただし、$N_\epsilon(\boldsymbol{x}^*) = \{\boldsymbol{x}; ||\boldsymbol{x}^* - \boldsymbol{x}|| < \epsilon\}$ である。局所的最適解と区別するために最適解を大域的最適解と呼ぶことも多い。大域的最適解は局所的最適解になっているが、逆は成り立たない。

3.2　定式化の作り方

定式化は、大雑把に言うと、言葉で説明された最適化問題に数式での表現を与えることである。簡単な例を元に示す。

例 1

庭造り。長さ $10[m]$ のフェンスがあり、このフェンスで囲われた長方形の庭を造りたいとする。庭の面積を最大にするにはどうすれば良いか？ この例で制御できる数量は (値) は、庭の縦の長さ $x_1[m]$ と横の長さ $x_2[m]$ であり、最適化問題の文脈では、決定変数と呼ばれる。また、

決定変数が満たすべき条件は制約条件と呼ばれる。まとめると、この問題は、$\boldsymbol{x} = {}^t(x_1, x_2)$ として、

$$f(\boldsymbol{x}) = x_1 x_2,\ S = \{\boldsymbol{x}; x_1, x_2 \in \mathbb{R},\ x_1, x_2 \geq 0,\ 2x_1 + 2x_2 = 10\}$$

とすれば、「Maximize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in S$ 」とかける。

例 2

野菜ジュース。栄養素 N_1, N_2, N_3 を補うために野菜 V_1 と V_2 を使ったジュースを毎朝飲むことにした。これらの野菜の評価と各要素の含有量は表 1 にまとめた。

また、栄養素 A_i は一日に $b_i[mg]$ だけ必要とする $(i = 1, 2, 3)$。費用を最小にするには各野菜をどれだけ購入すれば良いか？ ただし、野菜はグラム単位で購入できるものとする。

表 1: 例 2 の各栄養素の含有量

	単価 [円/g]	栄養素 $A_1[mg/g]$	栄養素 $A_2[mg/g]$	栄養素 $A_3[mg/g]$
野菜 V_1	c_1	a_{11}	a_{12}	a_{13}
野菜 V_2	c_2	a_{21}	a_{22}	a_{23}

例 2 の問題は、制御条件と目的関数が線形式で、さらに、決定変数が連続値を取る連続変数となる最適化問題として定式化される。このような問題は一般に線形計画問題と呼ばれる。

例 3

お菓子。遠足に持っていくお菓子を 300 円の予算内で買いたい。お菓子の候補として以下の 5 つまで絞った (「嬉しさ」は本人の主観)。

表 2: 例 3 のお菓子の候補

	1:チョコ	2:クッキー	3:ポテチ	4:グミ	5:ガム
嬉しさ	10	5	7	6	3
価格 [円]	140	80	130	70	30

ただし、同じお菓子を 2 つ以上は買わないこととする。この問題の定式化では決定変数は (連続値ではなく) 離散値にするとうまくいくことがわかる。具体的には、この問題は、

$$x_i = \begin{cases} 1 & i \text{ 番目のお菓子を購入するとき} \\ 0 & i \text{ 番目のお菓子を購入しないとき} \end{cases}$$

と解釈するものと約束し、$\boldsymbol{x} = {}^t(x_1, x_2, x_3, x_4, x_5)$ として、

$$\begin{aligned} f(\boldsymbol{x}) &= 10x_1 + 5x_2 + 7x_3 + 6x_4 + 3x_5, \\ S &= \{\boldsymbol{x}; x_1, x_2, x_3, x_4, x_5 \in \{0, 1\},\ 140x_1 + 80x_2 + 130x_3 + 70x_4 + 30x_5 \leq 300\} \end{aligned}$$

とすれば、「Maximize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in S$ 」とかける。

例 3 のように離散値をとる変数を整数変数、特に 0 か 1 の値をとる変数をバイナリ変数と呼ぶ。バイナリ変数は「ON/OFF」や「取る/取らない」等を表現できる極めて有用な変数である。整数変数を扱う最適化問題は整数最適化問題と呼ばれる。

整数変数と連続変数が混在する最適化問題はしばしば混合整数計画問題と呼ばれる。以下の最後の例は、混合整数計画問題として定式化するのが自然な例である。

例 4

工場での生産計画。ある工場で 2 つの製品 A_1 と A_2 がそれぞれ機械 M_1 と M_2 によって独立に製造されているとする。機械 M_i の生産効率は $e_i[kg/\text{時間}]$ で、セットアップに $s_i[W]$ と、単位時間の稼働に $t_i[W/\text{時間}]$ だけ電力を消費するとする $(i = 1, 2)$。一方、工場の 1 日の総消費電力の上限は $U^E[W]$ で、また、各機械の稼働時間の上限はともに U^L 時間とする。いま、製品 A_1 と A_2 の単価がそれぞれ p_1 と $p_2[\text{万円}/kg]$ であるとすれば、1 日の総利益を最大にするには、各機械の稼働時間をいくつに設定すればよいか？ ただし、機械を全く稼働させなければセットアップにかかる電力は生じないものとし、製品 A_1 と A_2 は (理想的であるが) 製造した分だけ売れるものとする。

4　実験方法

5　実験結果

6　検討・考察

7　まとめ

参考文献

[1] 東京理科大学工学部情報工学科 情報工学実験 2 2020 年度東京理科大学工学部情報工学科出版

A 付録

ソースコード 1: saiteki.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
template <class T>
inline bool chmax(T &a, T b)
{
    if (a < b)
    {
        a = b;
        return 1;
    }
    return 0;
}
template <class T>
inline bool chmin(T &a, T b)
{
    if (a > b)
    {
        a = b;
        return 1;
    }
    return 0;
}

using pll = pair<long long, long long>;
pll sub(long long h, long long w)
{
    if (h % 2 == 0 || w % 2 == 0)
        return {h * w / 2, h * w / 2};
    if (h > w)
        swap(h, w);
    return {h * (w + 1) / 2, h * (w - 1) / 2};
}

int main()
{
    long long H, W;
    cin >> H >> W;
    long long res = H * W;

    for (long long h = 1; h < H; ++h)
    {
        vector<long long> a(3);
        a[0] = h * W;
        auto p = sub(H - h, W);
        a[1] = p.first, a[2] = p.second;
        sort(a.begin(), a.end());
        chmin(res, a.back() - a[0]);
    }
```

```
49          for (long long w = 1; w < W; ++w)
50          {
51              vector<long long> a(3);
52              a[0] = H * w;
53              auto p = sub(H, W - w);
54              a[1] = p.first, a[2] = p.second;
55              sort(a.begin(), a.end());
56              chmin(res, a.back() - a[0]);
57          }
58          cout << res << endl;
59      }
```