組み合わせ最適化

目次

1　はじめに

本レポートでは, 組み合わせ最適化問題の代表例である最短路問題について述べる. なお, 実験で用いたプログラムは付録に記し, 実験で用いたインスタンスは別ファイルで添付する. ファイル名は「N_(ノード数)_P_(枝数).txt」となっている. N_20_P_100.txt にインスタンスの説明を記述している.

1.1　最短路問題

節点集合 $V = \{1, 2, \ldots, n\}$ と有向枝集合 $E \subseteq V \times V$ から成るグラフ $G = (V, E)$ と各枝 $(u, v) \in E$ の長さ $d(u, v)$ が与えられたとき, $(v_i, v_{i+1}) \in E, i = 0, 1, \ldots, l-1$ を満たす節点の列 $P = \langle u = v_0, v_1, \ldots, v_l = v \rangle$ を始点 u から節点 v への有向路 P と呼び, 路に含まれる枝の長さの総和を路の長さ $l(P)$ と言う. グラフ G において, 点 u から点 v への有向路 P の中で長さ $l(P)$ が最小となるものを探すことを考える. ここで, 枝重みは一般には負の場合も考える. 特に, $l(C)$ が負である有向閉路 C が存在することもある. この場合, 有向路 P を単純有向路としておかないと, $l(P)$ はいくらでも小さくなってしまう. そのため, 以下では単純有向路のみ考えることとする. 点 u から点 v への単純有向路 P の中で長さ $l(P)$ を最小にするものを点 u から点 v への最短路と呼び, その長さを $\mathrm{dist}(u, v)$ で表す. 与えられた 2 点 $u, v \in V$ に対して, 点 u から点 v への最短路を求める問題は, 負の枝重みを許す場合は NP-困難となる.

1.2　分枝限定法

離散問題を解く際, 問題の決定変数をひとつずつ固定して小規模な多数の問題に分割し, その全てを解くことで等価的に元の問題を解くことができる. この分割を分枝操作という. 分枝操作で作られる部分問題をすべて解く場合は非常に大きな計算量を必要とするが,

- 部分問題の最適値
- 部分問題が実行不可能であること
- 部分問題の最適解が元の問題の最適解にならないこと

のいずれかが分かれば, その部分問題をそれ以上調べる必要は無くなる. こうして探索を打ち切ることを限定操作と呼ぶ. 分枝操作と限定操作を組み合わせた方法を分枝限定法と呼ぶ.

2　課題 1

グラフ $G = (V, E)$, 節点 $s, t \in V$ 及び各枝 $(u, v) \in E$ の長さ $d(u, v)$ が与えられたとき, 点 s から点 t へ至る最短路を求める分枝アルゴリズムを考える. 素朴に考えると, 点 s から点 t に至るすべての路を出力し, そのコストが最小となるものを選べば良い. このアルゴリズムの擬似コードを Algorithm1 に示す. なお, 変数 prov は暫定解を表すグローバル変数であり, 十分大きい値で初期化されているものとする. 最後に出力された集合が最短路である.

Algorithm 1 pathall(x, F, len)

入力: グラフ G 上の節点 x, 点 s から点 x へ至る有向路に含まれる節点集合 F, 点 s から点 x まで F を通って到達した際の重み len

出力: グラフ G 上で点 s から点 t へ至る有向路で通る F を含む節点集合

```
for すべての有向枝 (x, y) ∈ E, y ∉ F に対して do
  if y = t then
    if prov > len then
      prov ←len
      return F ∪ {y}
    end if
  else
    pathall(y, F ∪ {y}, len + d(x, y))
  end if
end for
```

3 課題 2

課題 1 で設計したアルゴリズムを実装し, グラフの接点数及び枝数に対して実行時間及び探索したノード数を調べる. 実験結果を表 1 および図 1 に示す. なお, 節点数は 20 とし, 枝はランダムな節点を繋ぎ, 重みは [-10,1000] からランダムな整数が選ばれるとしてグラフを生成した. 枝数が小さいと終点に到達できない場合があり, 結果が安定しないため, 枝数 100 以上での結果を記載する.

表 1: 全探索の結果

枝数	実行時間 [msec]	探索ノード数
100	393	1481782
110	160	619687
120	275	1029249
130	4658	17883845
140	16484	60452241
150	11783	43699212
160	70016	257566813

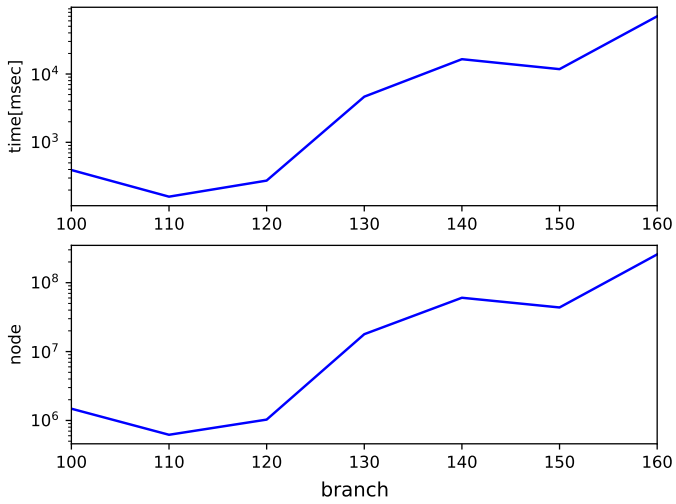

図 1: 実行時間と探索ノード数

3.1 結果の考察

図 1 を見ると, 多少のばらつきはあるものの, 探索ノード数や実行時間が枝数の増加に従って増大していることがわかる. 辺の数が小さいときにばらつきが生まれるのは, 到達不可能な点があったり, 他の点への有向枝を持たない「行き止まり」の点が多くなるためであると考えられる. そのため, 辺の数が多くなるにつれてばらつきは小さくなり, 実行時間や探索ノード数は爆発的に増大すると考えられる.

4 課題 3

課題 1 のアルゴリズムに限定操作を導入する. ここでは, 始点から現在の点までの長さに未探索の負重みの枝の長さを加えた値を部分問題の下界とし, 下界が暫定解を上回った場合は探索を打ち切る. 正当性については, この下界値は現在の点から終点に至ったときに取り得る最小の長さであることから明らかである. この限定操作アルゴリズムの擬似コードを Algorithm2 に示す. prov は Algorithm1 と同じで, lower_bound の初期値は負重みの枝の長さの総和である. 変数 lower_bound は下界値を直接表すものではないことに注意する. 最後に出力された集合が最短路である.

Algorithm 2 branch_bound(x, F, len, lower_bound)

入力: グラフ G 上の節点 x, 点 s から点 x へ至る有向路に含まれる節点集合 F, 点 s から点 x まで F を通って到達した際の重み len

出力: グラフ G 上で点 s から点 t へ至る有向路で通る F を含む節点集合

```
1: for すべての有向枝 (x, y) ∈ E, y ∉ F に対して do
2:   if y = t then
3:     if prov > len then
4:       prov ←len
5:       return F ∪ {y}
6:     end if
7:   else
8:     if d(x, y) < 0 then
9:       lower_bound ← lower_bound - d(x, y)
10:    end if
11:    branch_bound(y, F ∪ {y}, len + d(x, y), lower_bound)
12:    if d(x, y) < 0 then
13:      lower_bound ← lower_bound + d(x, y)
14:    end if
15:  end if
16: end for
```

5 課題 4

課題 3 で設計したアルゴリズムを実装し, 実行時間と探索ノード数を枝刈りなしのアルゴリズムと比較する. 結果を表 2 および図 2 に示す.

表 2: 枝刈りの結果

枝数	実行時間 [msec]	探索ノード数
100	0	205
110	0	92
120	0	149
130	0	362
140	0	317
150	0	320
160	0	171

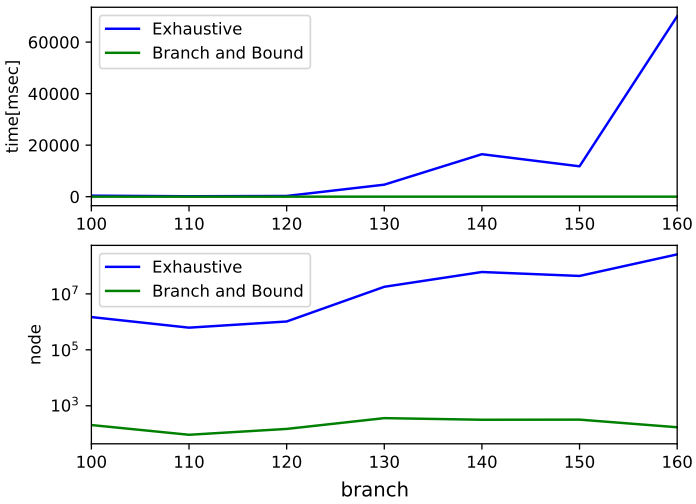

図 2: 実行時間と探索ノード数

5.1 結果の考察

表 2 および図 2 を見ると, 分枝限定法では枝数が増加しても実行時間は 1ms 未満に留まり, 探索ノード数も全探索の 1/1000 から 1/100000 程度になっている. 以上のことから, 負の重みを持つ辺が少なく, またその絶対値が相対的に小さいときには, 課題 3 で考えた限定操作が有効であることがわかる. ただし, たとえば 1 本を除いて重みの絶対値が 1000 以下で, 1 本の枝の重みが-1000000000 など非常に小さい場合には課題 3 の限定操作は効果が薄くなる. 実際, N_20_P_150.txt の 1 行目の重みを-1000000000 とした bad_P_150.txt では, 探索ノード数は全探索, 分枝限定ともに 43699212 で変化しなかった. グラフの特徴に適した限定操作を適用することが重要であることがわかる.

6 付録

実験で用いたプログラムを記す.

コード 1: 課題 2

```
#include<iostream>
#include<cmath>
#include<vector>
#include<fstream>
#include<ostream>
#include<sstream>
#include<bitset>
#include<time.h>
#include<sys/time.h>
using namespace std;

#define t 19
const int inf = 1000000;
int prov = inf;
unsigned long long count = 0;

void pathall(int v, bitset<20> bset, int len, vector<vector<int>> &mat){
    if (v==t){
        if (prov > len){
            prov = len;
        }
        // cout << bset << endl;
        return;
    }
    count++;
    for (int i=0; i<20; i++){
        if (mat.at(v).at(i) == inf){
            continue;
        }
        if (bset.test(i)){
            continue;
        }
        bset.set(i); //通った点を追加
        pathall(i, bset, len+mat.at(v).at(i), mat);
        bset.reset(i); //終わったら戻す
    }
}

void calc(int m){
    int u, v, w;
    count = 0;
    prov = inf;
    vector<vector<int>> mat(20, vector<int>(20));

    ostringstream oss;
    oss << "N_20" << "_P_" << m << ".txt";
    string str = oss.str();
    ifstream in(str);
    cin.rdbuf(in.rdbuf());
    oss.clear();
```

```
51
52      for (int i=0; i<20; i++){
53          for (int j=0; j<20; j++){
54              mat.at(i).at(j) = inf;
55          }
56      }
57      for (int i=0; i<m; i++){
58          cin >> u >> v >> w;
59          mat.at(u).at(v) = w;
60      }
61
62      clock_t start = clock();
63      pathall(0, 1, 0, mat);
64      clock_t end = clock();
65      long double time = (double) (end - start)*1000.0/CLOCKS_PER_SEC;
66      cout << m << "␣" << time << "␣" << count << "␣" << prov << endl;
67  }
68
69  int main(void){
70      for (int i=1; i<17; i++){
71          calc(i*10);
72      }
73      return 0;
74  }
```

コード 2: 課題 4

```
1   #include<iostream>
2   #include<cmath>
3   #include<vector>
4   #include<fstream>
5   #include<ostream>
6   #include<sstream>
7   #include<bitset>
8   #include<time.h>
9   #include<sys/time.h>
10  using namespace std;
11
12  #define t 19
13  const int inf = 1000000;
14  int prov = inf;
15  unsigned long long count = 0;
16
17  void pathall(int v, bitset<20> bset, int len, vector<vector<int>> &mat, int lower_bound){
18      if (v==t){
19          if (prov > len){
20              prov = len;
21          }
22          // cout << bset << endl;
23          return;
24      }
25      if (len + lower_bound > prov){
26          return;
27      }
28      count++;
29      for (int i=0; i<20; i++){
30          if (mat.at(v).at(i) == inf){
```

```
31              continue;
32          }
33          if (bset.test(i)){
34              continue;
35          }
36          bset.set(i); //通った点を追加
37          if (mat.at(v).at(i) < 0){
38              lower_bound -= mat.at(v).at(i);
39          }
40          pathall(i, bset, len+mat.at(v).at(i), mat, lower_bound);
41          bset.reset(i); //終わったら戻す
42          if (mat.at(v).at(i) < 0){
43              lower_bound += mat.at(v).at(i);
44          }
45      }
46  }
47
48  void calc(int m){
49      int u, v, w;
50      count = 0;
51      prov = inf;
52      int lower_bound = 0;
53      vector<vector<int>> mat(20, vector<int>(20));
54      ostringstream oss;
55      oss << "N_20" << "_P_" << m << ".txt";
56      string str = oss.str();
57      ifstream in(str);
58      cin.rdbuf(in.rdbuf());
59      oss.clear();
60
61      for (int i=0; i<20; i++){ //初期化
62          for (int j=0; j<20; j++){
63              mat.at(i).at(j) = inf;
64          }
65      }
66      for (int i=0; i<m; i++){ //枝の入力
67          cin >> u >> v >> w;
68          mat.at(u).at(v) = w;
69          if (w < 0){
70              lower_bound += w;
71          }
72      }
73
74      clock_t start = clock(); //時間計測
75      pathall(0, 1, 0, mat, lower_bound);
76      clock_t end = clock();
77      long double time = (double) (end - start)*1000.0/CLOCKS_PER_SEC;
78      cout << m << "␣" << time << "␣" << count << "␣" << prov << endl;
79  }
80
81  int main(void){
82      for (int i=1; i<17; i++){
83          calc(i*10);
84      }
85      return 0;
86  }
```

7　参考文献

参考文献

[1] 実験演習ワーキンググループ (2022),「数理工学実験テキスト_2022 年版」