数理工学実験
組合せ最適化

滝本亘（学籍番号 1029-33-1175）

2023 年 1 月 22 日

1　課題 1

1.1　はじめに

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設計し、その擬似コードを書け。

1.2　原理・方法の詳細

分枝限定法

分枝限定法は、問題をいくつかの小規模な問題に分割し、その全てを解くことで等価的に元の問題を解くという考え方に基づいている。小規模な問題への分割は、例えば、ある一つの決定変数の値を固定し（グラフ探索の場合、枝や節点を決め)、それぞれの場合を個別に考察することによって実現できる。このように問題を分割する操作を分枝操作という。分枝操作を繰り返し行うことで、すべての場合を列挙することができるが、その過程は生成木と呼ばれる根付き木を用いて表現できる。生成木において、、根は元の問題に対応し、その二つの子はそれぞれある変数の値を 0 または 1 に固定した 2 つの場合に対応する。その他の節点も同様である。よって、内部の節点は根からその節点への路に対応していくつかの変数の値を 0 か 1 に固定した問題に対応し、葉は全ての変数の値が定まった解に対応する。一部の変数の値が固定された問題を部分問題と呼び、この例の場合、部分問題も元の問題と等しい構造である。

1.3　擬似コード

Algorithm 1 shortest(x, F, c)

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ている節点集合 F,s から F を経て x に至るまでの長さ c

出力: 点 s から点 t へ至る最短路 $pass$,$pass$ の長さ min

```
for 全ての有向枝 (x, y) ∈ E, y ∉ F について do
    if y = t then
        if c + d(x, y) ≤ min then
            min = c + d(x, y)
            passF ∪ {y}
        end if
    else
        shortest(y, F ∪ {y}, c + d(x, y))
    end if
end for
```

2　課題 2

2.1　はじめに

課題 1 で設定したアルゴリズムを実装し、グラフの節点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、表とグラフにまとめよ。

2.2　実験方法

コード 1: 課題 2

```
import time

A=[
    [0,1,3],
    [0,8,4],
    [0,7,5],
    [0,0,6],
    [0,3,8],
    [0,5,9],
    [0,4,12],
    [0,1,13],
    [0,1,16],
    [1,9,3],
    [1,4,7],
    [1,9,8],
    [1,1,9],
    [1,2,13],
    [1,0,15],
    [1,5,18],
    [2,0,3],
    [2,8,6],
    [2,7,7],
    [2,2,9],
    [2,2,10],
    [2,3,12],
    [2,8,14],
    [2,7,17],
    [2,8,18],
    [2,1,19],
    [3,5,4],
    [3,7,6],
    [3,8,9],
    [3,7,11],
    [3,4,15],
    [3,2,17],
    [3,4,19],
    [4,3,5],
    [4,4,6],
    [4,7,9],
    [4,9,10],
    [4,5,17],
    [4,2,18],
    [5,5,6],
    [5,4,10],
    [5,5,12],
    [5,2,13],
    [5,6,18],
    [6,8,7],
    [6,0,10],
    [6,9,12],
    [6,5,13],
    [6,9,15],
```

```
        [6,7,17],
        [6,3,18],
        [7,9,8],
        [7,8,9],
        [7,8,11],
        [7,9,13],
        [7,3,15],
        [7,2,16],
        [7,0,17],
        [7,1,18],
        [8,6,9],
        [8,4,10],
        [8,5,12],
        [8,0,13],
        [8,0,14],
        [8,3,17],
        [8,3,18],
        [8,7,19],
        [9,8,10],
        [9,7,11],
        [9,7,13],
        [9,2,14],
        [9,6,15],
        [9,2,16],
        [9,5,17],
        [9,1,19],
        [10,5,11],
        [10,3,15],
        [10,9,18],
        [11,9,13],
        [11,6,14],
        [11,9,15],
        [11,1,16],
        [11,6,17],
        [11,6,18],
        [11,5,19],
        [12,3,14],
        [12,9,15],
        [12,7,16],
        [12,4,18],
        [13,7,14],
        [13,1,16],
        [13,7,17],
        [13,9,18],
        [13,3,19],
        [14,9,16],
        [14,2,17],
        [15,6,19],
        [16,7,17],
        [16,7,19],
        [17,8,18]
]

a=0
def shortest(x,F,c):
    global a
    global min
    global path

    for i in range(len(A)):

        if A[i][0]==x:

            if A[i][2] in F:
                continue

            else:
                a+=1
```

```
121                     Y=F.copy()
122                     Y.append(A[i][2])
123                     d=c
124                     d+=A[i][1]
125
126                     if A[i][2]==t:
127                         try:
128                             if d<=min:
129                                 min=d
130                                 path=[Y,min]
131                             else:
132                                 continue
133                         except:
134                             min=d
135                             path=[Y,min]
136
137                     else:
138                         shortest(A[i][2],Y,d)
139
140             else:
141                 continue
142
143         return path
144
145     t1=time.time()
146
147     s=0
148     t=19
149     F=[s]
150     c=0
151     path=[]
152     print(shortest(s,F,c))
153
154     t2=time.time()
155
156     print(t2-t1)
157     print(a)
```

2.3 結果

表 1: 課題 2

節点数	枝数	ノード数	実行時間
19	100	3960	0.018923161
18	92	3151	0.010963833
17	80	1999	0.005386095
16	69	1109	0.002947552
15	62	663	0.002079072
14	54	505	0.000961533
13	48	282	0.000572333
12	40	171	0.000500052
11	35	152	0.000306938
10	31	86	0.000170159
9	25	47	0.0000700545
8	18	28	0.0000483775
7	15	20	0.00000998259
6	12	13	0.00000999212
5	7	6	0.00000998974
4	5	3	0
3	3	1	0

実行時間は 100 回の平均を取っている。

節点数、枝数が少ないほど、ノード数も少なくなり、実行時間は短くなる。また節点数 14 あたりから、実行時間が 0.0 となる桁落ちが発生するため、枝数が多いのに実行時間が短いという現象が見られる。

3　課題 3

3.1　はじめに

課題 1 で設定したアルゴリズムに関して限定操作を導入し、その妥当性を説明および証明せよ。限定操作別の擬似コードを書き、最後に全体の限定操作アルゴリズムの擬似コードを書け。

3.2　原理・方法の詳細

3.2.1　分枝限定法 2

分枝限定法で、実際に全ての葉を列挙するわけではなく、その一部分だけを生成する。生成木のうち実際に生成された節点のみからなるものを探索木という。ある部分問題（節点）に対して、

(i) その最適値

(ii) その部分問題が実行不可能であること

(iii) その部分問題の最適解が元の問題の最適解とはならないこと

のいずれかが分かれば、その部分問題はこれ以上調べる必要はなく、その下の節点（子孫と呼ぶ）の探索を省略できる（終端するという）。これを限定操作と呼ぶ。探索の過程で、まだ終端されていない部分問題を活性であるといい、全ての部分問題が終端されたとき、つまり活性部分問題がなくなったとき、分枝限定法は厳密な最適解を与える（もしくは実行可能解が存在しないことを証明する）。

限定操作の代表例である下界値テストについて簡単に述べる。近似解法等を用いて実行可能解を一つでも得ることができれば、その目的関数値は最適値の上界値となる。また、探索が生成木の葉に到達したときにも実行可能解が得られる場合がある。これまでの探索で得られている最良の上界値を暫定値という。一方、制約を緩めた緩和問題を解くなどして得られた最適値は、元の問題の最適値の下界値となる。このようにして得られた上下界値を用い、「ある部分問題の下界値が暫定値以上であれば、その子孫を調べる必要はない」ことが結論でき。その部分問題の子孫を全て調べても、暫定値よりもよい目的関数値を持つ解は見つからないからである。以上が下界値テストである。（最大化問題の場合は上界と下界の役割が入れ替わる。）

なお、計算時間の都合などにより活性部分問題が残っている状態で探索を終了し、その時点での暫定値を出力する方法もしばしば用いられる。このとき、出力した値が最適である保証はなく、分枝限定法を近似解法として用いることになるが、この場合でも、活性部分問題の下界値の最小値が元の問題の下界値となり、出力値の精度を測る指標としてこれを用いることができる。

3.3　擬似コード

4　課題 4

4.1　はじめに

限定操作を実装し、適当なインスタンスで課題 2 の実装と実行時間の及び探索したノード数を比較せよ。結果を表とグラフにまとめよ。