知能プログラミング演習II 課題1

グループ 02
29114073 末永彩羽

2019 年 10 月 7 日

提出物　rep1

グループ　グループ 02

メンバー

学生番号	氏名	貢献度比率
29114001	愛甲拓海	
29114002	青木侑省	
29114073	末永彩羽	
29114156	近藤拓海	
29119032	古川翔也	

1　課題の説明

課題 1-1 Search.java の状態空間におけるパラメータ（コストや評価値）を様々に変化させて実行し，各探索手法の違いを説明せよ．

課題 1-2 グループでの進捗管理や成果物共有などについて，工夫した点や使ったツールについて考察せよ．

2　課題 1-1

> Search.java の状態空間におけるパラメータ（コストや評価値）を様々に変化させて実行し，各探索手法の違いを説明せよ．

グループのメンバーそれぞれが、違ったパラメータを用いることにより、様々な観点から考察する。

2.1　手法

Search.java を改良し、コストや評価値をランダムな値でセットされるようにした。また、コストだけや評価だけを変化させるというオプションもつけた。具体的には、以下である。

1. デフォルト値

2. コスト・評価値共にランダム値

3. コストはデフォルト値、評価値のみランダム値

4. 評価値はデフォルト値、コストのみランダム値

他のメンバーがランダム値をセットする部分の実装を行ってくれたので、私はランダム値にする部分を決めるオプションを加えた。

2.2 実装

まず、プログラムに含まれるクラスは以下の 2 つ。

- Search クラス: 経路探索を行うクラス．Search.java に含まれる。
- Node クラス: グラフ構造のクラス。Search.java に含まれる。

元の Search.java から、実行する際に用いる引数を二つに増やし、ランダム値にするかどうかを選べるようになっている。以下はランダム値に設定する際に用いるメソッドであり、この部分は他のメンバーが実装した。

Listing 1: random メソッド

```
1    private int random(boolean rd, int def){
2        return rd ? (int)(Math.random()*10):def;
3        }
```

このメソッドからランダム値が生成されるが、オプションである第二引数の値によって、rd の真偽の切り替えを行うことで、どの部分でこの生成するかを制御した。

Listing 2: makeStateSpace メソッド

```
1    private void makeStateSpace(int rand) {
2        boolean rd = false;
3        node = new Node[10];第二引数がのとき全ランダム値、のとき評価値のみランダ
             ム値
4
5        //12
6        if(rand == 1 || rand == 2)
7        {
8        rd = true;
9        }
10
11       // 状態空間の生成
12       node[0] = new Node("L.A.Airport", 0);
13       node[1] = new Node("UCLA", random(rd, 7));
14       node[2] = new Node("Hoolywood", random(rd, 4));以下省略
15       //
```

2.3 実行例

Search クラスに引数 1 と 2 を指定した実行結果を以下に示す。

```
clf14073@cse:~/eclipse-workspace/Search > java Search 1 2
L.A.Airport(h:0)
  children:UCLA:1
  children:Hoolywood:3
UCLA(h:9)
  children:Hoolywood:1
  children:Downtown:6
Hoolywood(h:2)
  children:Anaheim:6
  children:Downtown:6
  children:Pasadena:3
Anaheim(h:2)
  children:GrandCanyon:5
  children:Pasadena:2
  children:DisneyLand:4
GrandCanyon(h:4)
  children:DisneyLand:2
  children:Las Vegas:1
SanDiego(h:0)
  children:UCLA:1
Downtown(h:5)
  children:SanDiego:7
  children:Pasadena:2
Pasadena(h:3)
  children:DisneyLand:1
  children:Las Vegas:7
DisneyLand(h:1)
  children:Las Vegas:5
Las Vegas(h:0)

Breadth First Search
STEP:0
OPEN:[L.A.Airport(h:0)]
CLOSED:[]
STEP:1
OPEN:[UCLA(h:9), Hoolywood(h:2)]
CLOSED:[L.A.Airport(h:0)]
STEP:2
OPEN:[Hoolywood(h:2), Downtown(h:5)]
CLOSED:[L.A.Airport(h:0), UCLA(h:9)]
STEP:3
OPEN:[Downtown(h:5), Anaheim(h:2), Pasadena(h:3)]
CLOSED:[L.A.Airport(h:0), UCLA(h:9), Hoolywood(h:2)]
STEP:4
OPEN:[Anaheim(h:2), Pasadena(h:3), SanDiego(h:0)]
CLOSED:[L.A.Airport(h:0), UCLA(h:9), Hoolywood(h:2), Downtown(h:5)]
STEP:5
```

```
48 OPEN:[Pasadena(h:3), SanDiego(h:0), GrandCanyon(h:4), DisneyLand(h:1)]
49 CLOSED:[L.A.Airport(h:0), UCLA(h:9), Hoolywood(h:2), Downtown(h:5), Anaheim(h
       :2)]
50 STEP:6
51 OPEN:[Las Vegas(h:0), SanDiego(h:0), GrandCanyon(h:4), DisneyLand(h:1)]
52 CLOSED:[L.A.Airport(h:0), UCLA(h:9), Hoolywood(h:2), Downtown(h:5), Anaheim(h
       :2), Pasadena(h:3)]
53 *** Solution ***
54 Las Vegas(h:0) <- Pasadena(h:3) <- Hoolywood(h:2) <- L.A.Airport(h:0)処理時間
55 :1556800 ステップ数:6以下省略
56 //
```

引数を 1 と 2 としているので、探索方法は幅優先探索、ランダム値は評価値のみとなる。初めに生成されたグラフ構造が出力され、次に探索のステップと結果が表示される。デフォルトでは第一引数で指定した探索方法のみ行われてプログラムが終了するが、考察を行う利便上、すべての探索法を一括で行うので、続けて違う探索法での結果が表示されていく。

2.4 実行結果

実行結果をまとめた表を以下に示す．6 つの探索方法について、評価値とコストをデフォルト値とランダム値の組み合わせ 4 パターンで実行した。ランダム値が含まれるオプションは、実行するたびグラフが変化するので、5 回実行したうちの平均の値を取っている。また、100 ステップ数以内に探索が終わらないものは、解なしとし、平均からは省いてある。

表 1: デフォルト値

探索手法	実行時間	ステップ数	コスト
幅優先探索	1542800	6	13
深さ優先探索	1501000	5	16
分枝限定法	1949300	7	11
山登り法	6308700	無限	解なし
最良優先探索	1254700	5	13
A-star アルゴリズム	2272200	7	11

表 2: 全てランダム値

探索手法	実行時間 (ns)	ステップ数	コスト
幅優先探索	2807320	6	12.8
深さ優先探索	3259720	5	18.6
分枝限定法	4488680	6.2	9.2
山登り法	4002520	2.75	15.5
最良優先探索	3177240	4.4	16
A-star アルゴリズム	2235660	4.4	8.8

表 3: 評価値のみランダム値

探索手法	実行時間	ステップ数	コスト
幅優先探索	1746660	6	13
深さ優先探索	1340780	5	16
分枝限定法	1999580	7	11
山登り法	1548940	2.75	16.75
最良優先探索	1323860	4.4	15.6
A-star アルゴリズム	2606160	7.6	11

表 4: コストのみランダム値

探索手法	実行時間	ステップ数	コスト
幅優先探索	1830080	6	10.2
深さ優先探索	2845160	5	14.8
分枝限定法	4185420	6.2	9.8
山登り法	14829500	無限	解なし
最良優先探索	1884100	5	10.2
A-star アルゴリズム	3215800	4.8	9.8

以上の表をグラフで可視化したものが以下の図 1 である。

2.5 考察

以上の結果から、探索手法やパラメータによって、探索時間やステップ数、コストに違いが出てくることがわかる。

まず、探索手法による違いを見ていく。表 1 より、このグラフでは分枝限定法と A-star アルゴリズムで最適解を見つけることができた。ただし、実行時間やステップ数で比べるとその二つはより多くの過程を必要としていることがわかる。一般に比較されやすい幅優先探索と深さ優先探索では、実行時間やステップ数で違いは見られないが、コストにおいて両者に違いが出ている。このグラフでは、幅優先探索でより良い解を発見できる。山登り法のみ、無限ループに陥ってしまい解が発見できなかった。これは、より良いコストのルートを欲張り法で選んでいくことで、閉じた部分グラフを探索し続けてしまうことが原因にある。

これらの特徴は、パラメータを変えても見られる。図 1 で全体を比較すると、グラフの形はそれほど大きく異ならない。これは、グラフの形そのものが、探索手法に大きく影響を与えているといえる。

つづいて、パラメータによる違いを詳細に見ていく。実行時間を見てみると、図 1 の右半分と左半分、つまり、コストを変えたか変えていないかで違いが出ている。コストをランダムで設定した方は、デフォルト値よりも実行時間が長くなっている。

一方、評価値をランダム値にすると、幅優先探索、深さ優先探索、分枝限定法では変えないときと比べて違いがみられない。山登り法では解が見つかるようになり、最良優先探索や A-star アルゴリズムでは、ステップ数や解のコストにわずかながら変化がみられる。これは、後者三つの手法

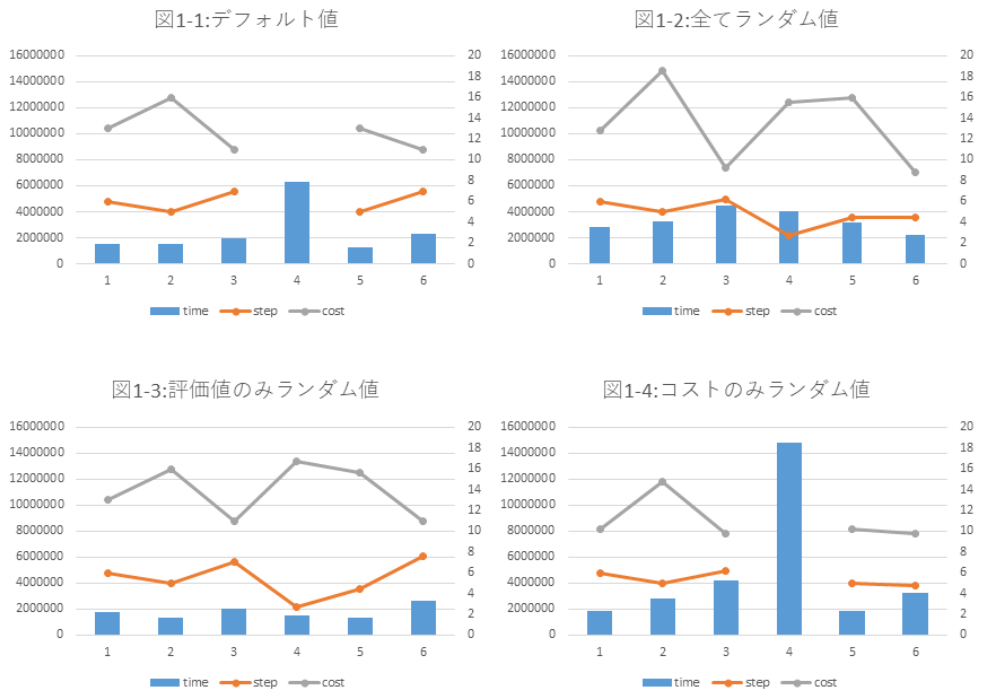

図 1: 実行結果まとめ

が評価値を用いたアルゴリズムである点から納得がいく。A-star アルゴリズムでステップ数の増加の原因は、あべこべな評価値での計算の多さによるものだと考えられる。

3 課題 1-2

グループでの進捗管理や成果物共有などについて，工夫した点や使ったツールについて考察せよ．

課題 1-2 は実装を伴わない課題であるため、考察のみ記す。

3.1 考察

私たちのグループでは、成果物の共有を GitHub、個々の相談や連絡を LINE で行うことにした。理由としては、GitHub では進捗管理がしやすく、グループでも使ったことのある人が一人いたためである。しかし、GitHub だけではメッセージのやり取りは難しいため、全員がすでにアカウントを取得しており、使い方の慣れている LINE を併用して用いることにした。

4 感想

GitHub の使い方に苦労している。便利であるために複雑で、理解して使えるようになるのに時間がかかりそうだ。他の班の友達にもいろいろと聞いて、なんとか慣れていきたい。探索手法の比較では、評価値のパラメータをあべこべにしても、思ったよりも最適解に近い値を導くことができていたことに驚いた。

参考文献

[1] 今さら聞けない！GitHub の使い方【超初心者向け】, TECHACADEMY, https://techacademy.jp/magazine/6235 （2019 年 10 月 13 日アクセス）.