23/25

Comp560 Jiabing Song/Yingying Wu

HW2 Report

Part 1: Constraint Satisfaction Problems

Graduation Dinner (5 points)

5/5

a) Variables : {J1, J2, T, R, M} #J1 stands for Jasmine and J2 for Jason

Domains : {1,2,3,4,5}

Constraints : J2 < J1, diff(M,T) > 2, diff(J1,J2) > 1, diff(M,J2) > 1, diff(M,R) > 1

b) J1:{3,4,5}, J2:{1,2,3}, T:{1,2,4,5}, R:{1,2,3,4,5}, M:{1,4,5}

c) M, because M has the fewest legal values therefore it is the most constrained variable

d) J1:{4}, J2:{1,2}, T:{1,2}, M:{5}

e) J2 = 1, T = 2, R = 3, J1 = 4, M =5

T = 1, J2 = 2, R = 3, J1 = 4, M =5

Hide & Seek (8 points)

8/8

Variables:{(1,1), (1,2), (1,3)...}

Domains:{friend, tree, empty}

Constraints: Starting from a friend’s position, we seek all eight directions to find if the first object x we meet is a friend or a tree or empty.

left(x) ≠ {friend}

right(x) ≠ {friend}

up(x) ≠ {friend}

down(x) ≠ {friend}

upper-left(x) ≠ {friend}

upper-right(x) ≠ {friend}

lower-left(x) ≠ {friend}

lower-right(x) ≠ {friend}

We use the Minimun Remaining Value method to select which variable to improve next. We write a function to find all constraints on different friends. In each iteration, we first improve the most contrained one.

After improvement we’ll check the new total constraints. If it equals 0, that means the program has found the solution and it will break the iteration. If the total constraints remain the same, that means most constrained one is no longer able to improve. we randomly pick a improvable friend and improve it.

After that we check all friends to see if any of them can be improved. If there's no friend to improve while the total_constraints is not 0, that means we are stuck in local minima. We managed to solve the local minima by taking a step back. That is, we implement a function called random move which randomly pick a friend and reassigning a different position in his column. And then we repeat the whole process. During the process, there were several times that we meet some difficulties. At first, the algorithm would stop when the most constrained one cannot be improved instead of all friends in the forest cannot be improved. We solved this by a random improvement. Later we, as expected, stuck in local minima. We solved this by a random move.

<u>Initialization 1:</u>

The initial forest. Friends randomly in assigned in each column.

```
T T
      T
      T F   T F
          T
  T F   T   F
    T F   F T
T F T     T
F T T
Constraints of  7 0 :
[6, 1]
Constraints of  6 1 :
[7, 0]
Constraints of  4 2 :
[2, 4]
[5, 3]
Constraints of  5 3 :
[5, 5]
[4, 2]
Constraints of  2 4 :
[4, 2]
Constraints of  5 5 :
[5, 3]
[4, 6]
Constraints of  4 6 :
[5, 5]
Constraints of  2 7 :
```

Solution 1:

```
T T F
F     T   F
      T     T F
        F T
  T     T   F
    T F     T
T F T     T
  T T
Constraints of  1 0 :
Constraints of  6 1 :
Constraints of  0 2 :
Constraints of  5 3 :
Constraints of  3 4 :
Constraints of  1 5 :
Constraints of  4 6 :
Constraints of  2 7 :
iteration: 4
```

Initialization2

different tree position

```
T T           F
    F T   F
  F   T     T
          T
  T     T
F   T F     T
T   T   F T F
  T T
Constraints of  5 0 :
Constraints of  2 1 :
[1, 2]
Constraints of  1 2 :
[2, 1]
Constraints of  5 3 :
[6, 4]
Constraints of  6 4 :
[5, 3]
Constraints of  1 5 :
Constraints of  6 6 :
Constraints of  0 7 :
```

solution2:

```
T T F
      T       F
  F   T F   T
          T
  T     T F
F   T F     T
T   T     T F
  T T
Constraints of  5 0 :
Constraints of  2 1 :
Constraints of  0 2 :
Constraints of  5 3 :
Constraints of  2 4 :
Constraints of  4 5 :
Constraints of  6 6 :
Constraints of  1 7 :
iteration 8
```

initializatoin3

increase size of the grid, same number of trees

```
  T     F     F   T
      T     F
      T   F T
          T
F       T           T
            T T   F
T         T
    T F
  F T
  T
                    F
    F           F     F
Constraints of  4 0 :
[7, 3]
Constraints of  8 1 :
Constraints of  11 2 :
[11, 8]
Constraints of  7 3 :
[4, 0]
Constraints of  0 4 :
[0, 7]
Constraints of  2 5 :
[1, 6]
Constraints of  1 6 :
[0, 7]
[2, 5]
Constraints of  0 7 :
[0, 4]
[1, 6]
Constraints of  11 8 :
[11, 2]
[11, 11]
Constraints of  5 9 :
Constraints of  10 10 :
[11, 11]
Constraints of  11 11 :
[11, 8]
[10, 10]
```

solution3

```
F T           F   T
    F T             F
      T F   T
          T F
        T           T
            T T   F
T         T
    T F
  F T
  T             F
          F
                      F
Constraints of  0 0 :
Constraints of  8 1 :
Constraints of  1 2 :
Constraints of  7 3 :
Constraints of  2 4 :
Constraints of  10 5 :
Constraints of  3 6 :
Constraints of  0 7 :
Constraints of  9 8 :
Constraints of  5 9 :
Constraints of  1 10 :
Constraints of  11 11 :
iteration: 6
```

initialization4

more trees

```
  T F T
T     T   F F F
  T   T F   T
  F       T
F T   F T   T
            T T
T   T     T
    T T
Constraints of  4 0 :
[3, 1]
Constraints of  3 1 :
[4, 0]
Constraints of  0 2 :
Constraints of  4 3 :
[1, 6]
Constraints of  2 4 :
[1, 5]
Constraints of  1 5 :
[1, 6]
[2, 4]
Constraints of  1 6 :
[1, 5]
[1, 7]
[4, 3]
Constraints of  1 7 :
[1, 6]
```

solution4

```
F T F T   F
T     T       F
  T   T F   T
  F       T F
  T   F T   T
            T T
T   T     T
    T T
Constraints of  0 0 :
Constraints of  3 1 :
Constraints of  0 2 :
Constraints of  4 3 :
Constraints of  2 4 :
Constraints of  0 5 :
Constraints of  3 6 :
Constraints of  1 7 :
iteration: 3
```

<u>initialization5</u>

fewer trees

```
F     T
  F F T     F
  T   T F     F
          T
        T

    T
      F   F
Constraints of  0 0 :
[1, 1]
Constraints of  1 1 :
[1, 2]
[0, 0]
Constraints of  1 2 :
[1, 1]
Constraints of  7 3 :
[7, 5]
Constraints of  2 4 :
[2, 7]
Constraints of  7 5 :
[7, 3]
Constraints of  1 6 :
[2, 7]
Constraints of  2 7 :
[2, 4]
[1, 6]
```

solution5:

```
      T   F
F     T
  T F T F
          T
  F     T     F

    T       F
      F
Constraints of  1 0 :
Constraints of  4 1 :
Constraints of  2 2 :
Constraints of  7 3 :
Constraints of  2 4 :
Constraints of  0 5 :
Constraints of  6 6 :
Constraints of  4 7 :
iteration: 18
```

one of thelocal minima initialization before we came up with the solution:

```
  T F   F
T   T F       F
T           T
      T       T
  F         F
    T T T F
T         T T
F T T
```

Jiabing and Yingying demand bonus point for solving local minima! *Ice Bear mimic*

Part 2: Minimax +1 bonus

10/12 **Candy game**

Evaluation Function:

We agree that the evaluation utility should be determine by three factors: the score of the grid, whether it will be captured at that move and whether it can capture at the move. Thus, the evaluation function can be written as Eval(N) = w1 * the value of N + w2 * the value to eat the enemy if any + w3 * the value will be deducted if can be eaten(this one is usually negative), where N is a terminal node or a node at the depth limit. If the player is min, the evaluation utility is negative.

Then we agree that the evaluation priority should be first eat the opponent's grid, then prevent itself from eaten, the last is the score. So after we played with different weight values, we choose w1 = 0.2, w2 = 0.45, w3 = 0.35. We also played with weights like, 0.33, 0.33, 0.33 / 0.1, 0.7, 0.2 / 1,0,0 / 0,1,0 / 0,0,1...

-1 What was depth limit

Minimax:

Minimax is devided by three function: miniMax, mmMax and mmMin and a tree node class called mmNode, which includes the coordinate, score and player in the node. Minimax will make a list of available cells of the current gameboard and pass it down to mmMin/ mmMax.(depends on the player who's calling it) mmMax and mmMin will recursively called each other before reaching the depth limit to return maxNode or the minNode.

Alphabeta:

Our Alphabeta function takes a node, A and B as parameters. We first determine if the node is a terminal node or at the depth limit level. If so we return the evaluated value. Else we first check if it is a max player or a min player and then call the function on their successors recursively. For both players we pass down the alpha and beta value and update the A and B (which serve as the local alpha and beta). If A >= beta or alpha >= B we do the pruning.

-2 Unfortunately we don't have enought time to make alphabeta fully functional. It got stuck in some weird loop so we only include our result running minimax below.

1. Minimax vs minimax:

AlmondJoy.txt

```
Enter file name(with .txt):AlmondJo
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
blue:  0 1
green:  0 2
blue:  0 3
green:  0 4
blue:  0 5
green:  1 0
blue:  1 2
green:  1 3
blue:  1 4
green:  1 5
blue:  2 0
green:  2 1
blue:  2 3
green:  2 4
blue:  2 5
green:  3 0
blue:  3 1
green:  3 2
blue:  3 4
green:  3 5
blue:  4 0
green:  4 1
blue:  4 2
green:  4 3
blue:  4 5
green:  5 0
blue:  5 1
green:  5 2
blue:  5 3
green:  5 4
blue:  5 5
green:  1 1
blue:  4 4
green:  2 2
blue:  3 3
green:  0 0
========================
```

```
g g g b g b
g g g g b g
b g g b g b
g b b b b g
b g b b b b
g b g b b b
19
17
Blue wins!
```

average time: 0.000523606936137
total node expanded: 1296
average node per move: 72

Ayds.txt

```
Enter file name(with .txt):Ayd
99 1 99 1 99 1
1 99 1 99 1 99
99 1 99 1 99 1
1 99 1 99 1 99
99 1 99 1 99 1
1 99 1 99 1 99
blue:  0 1
green:  0 2
blue:  0 3
green:  0 4
blue:  0 5
green:  1 0
blue:  1 2
green:  1 3
blue:  1 4
green:  1 5
blue:  2 0
green:  2 1
blue:  2 3
green:  2 4
blue:  2 5
green:  3 0
blue:  3 1
green:  3 2
blue:  3 4
green:  3 5
blue:  4 0
green:  4 1
blue:  4 2
green:  4 3
blue:  4 5
green:  5 0
blue:  5 1
green:  5 2
blue:  5 3
green:  5 4
blue:  5 5
green:  1 1
blue:  4 4
green:  2 2
blue:  3 3
green:  0 0
========================
```

```
g g g b g b
g g g g b g
b g g b g b
g b b b b g
b g b b b b
g b g b b b
901
899
Blue wins!
```

average time: 0.000529249509176
total node expanded: 1296
average node per move: 72

Bit-O-Honey.txt

```
Enter file name(with .txt):Bit-O-Honey.txt
1 1 1 1 1 1
2 2 2 2 2 2
4 4 4 4 4 4
8 8 8 8 8 8
16 16 16 16 16 16
32 32 32 32 32 32
blue:  0 1
green:  0 2
blue:  0 3
green:  0 4
blue:  0 5
green:  1 0
blue:  1 2
green:  1 3
blue:  1 4
green:  1 5
blue:  2 0
green:  2 1
blue:  2 3
green:  2 4
blue:  2 5
green:  3 0
blue:  3 1
green:  3 2
blue:  3 4
green:  3 5
blue:  4 0
green:  4 1
blue:  4 2
green:  4 3
blue:  4 5
green:  5 0
blue:  5 1
green:  5 2
blue:  5 3
green:  5 4
blue:  1 1
green:  2 2
blue:  3 3
green:  5 5
blue:  4 4
green:  0 0
=========================
```

```
g b g b g b
b b g g b g
b g g b g b
g b b b b g
b g b b b b
g b g b b g
229
149
Blue wins!
```

average time: 0.000569144884745
total node expanded: 1296
average node per move: 72

Mounds.txt

```
Enter file name(with .txt):Mounds.txt
1 1 1 1 1 1
1 3 4 4 3 1
1 4 2 2 4 1
1 4 2 2 4 1
1 3 4 4 3 1
1 1 1 1 1 1
blue:  0 1
green:  0 2
blue:  0 3
green:  0 4
blue:  0 5
green:  1 0
blue:  1 2
green:  1 3
blue:  1 4
green:  1 5
blue:  2 0
green:  2 1
blue:  2 3
green:  2 4
blue:  2 5
green:  3 0
blue:  3 1
green:  3 2
blue:  3 4
green:  3 5
blue:  4 0
green:  4 1
blue:  4 2
green:  4 3
blue:  4 5
green:  5 0
blue:  5 1
green:  5 2
blue:  5 3
green:  5 4
blue:  5 5
green:  1 1
blue:  4 4
green:  2 2
blue:  3 3
green:  0 0
=========================
```

```
g g g b g b
g g g g b g
b g g b g b
g b b b b g
b g b b b b
g b g b b b
38
34
Blue wins!
```

average time: 0.00050863954756
total node expanded: 1296
average node per move: 72

ReesesPieces.txt

```
Enter file name(with .txt):ReesesPieces.txt
66 76 28 66 11 9
31 39 50 8 33 14
80 76 39 59 2 48
50 73 43 3 13 3
99 45 72 87 49 4
80 63 92 28 61 53
blue:  0 1
green:  0 2
blue:  0 3
green:  0 4
blue:  0 5
green:  1 0
blue:  1 2
green:  1 3
blue:  1 4
green:  1 5
blue:  2 0
green:  2 1
blue:  2 3
green:  2 4
blue:  2 5
green:  3 0
blue:  3 1
green:  3 2
blue:  3 4
green:  3 5
blue:  4 0
green:  4 1
blue:  4 2
green:  4 3
blue:  4 5
green:  5 0
blue:  5 1
green:  5 2
blue:  5 3
green:  5 4
blue:  5 5
green:  4 4
blue:  1 1
green:  3 3
blue:  2 2
green:  0 0
=========================
```

```
g b g b g b
b b b g b g
b b b b g b
g b b g g g
b g b g g g
g b g b g b
1037
616
Blue wins!
```

average time: 0.000501712163289
total node expanded: 1296
average node per move: 72

Bonus 2:

pyramid.txt (board size 8x8)

```
Enter file name(with .txt):pyramid.txt
1 1 1 1 1 1 1 1
1 2 2 2 2 2 2 1
1 2 4 4 4 4 2 1
1 2 4 8 8 4 2 1
1 2 4 8 8 4 2 1
1 2 4 4 4 4 2 1
1 2 2 2 2 2 2 1
1 1 1 1 1 1 1 1
blue:  0 1
green:  0 2
blue:  0 3
green:  0 4
blue:  0 5
green:  0 6
blue:  0 7
green:  1 0
blue:  1 2
green:  1 3
blue:  1 4
green:  1 5
blue:  1 6
green:  1 7
blue:  2 0
green:  2 1
blue:  2 3
green:  2 4
blue:  2 5
green:  2 6
blue:  2 7
green:  3 0
blue:  3 1
green:  3 2
blue:  3 4
green:  3 5
blue:  3 6
green:  3 7
blue:  4 0
green:  4 1
blue:  4 2
green:  4 3
blue:  4 5
green:  4 6
blue:  4 7
green:  5 0
blue:  5 1
green:  5 2
blue:  5 3
green:  5 4
blue:  5 6
green:  5 7
blue:  6 0
green:  6 1
blue:  6 2
green:  6 3
blue:  6 4
green:  6 5
blue:  6 7
green:  7 0
blue:  7 1
green:  7 2
blue:  7 3
green:  7 4
blue:  7 5
green:  7 6
blue:  7 7
green:  1 1
blue:  6 6
green:  2 2
blue:  5 5
green:  3 3
blue:  4 4
green:  0 0
======================
g g g b g b g b
g g g g b g b g
b g g g g b g b
g b g g b g b g
b g b b b b g b
g b g b b b b g
b g b g b b b b
g b g b g b b b
82
66
Blue wins!
```

average time: 0.0012518465519

total node expanded: 4096

average node per move: 227

superPyramid.txt (board size 10x10)

```
1 1 1 1 1 1 1 1 1 1
1 2 2 2 2 2 2 2 2 1
1 2 4 4 4 4 4 4 2 1
1 2 4 8 8 8 8 4 2 1
1 2 4 8 16 16 8 4 2 1
1 2 4 8 16 16 8 4 2 1
1 2 4 8 8 8 8 4 2 1
1 2 4 4 4 4 4 4 2 1
1 2 2 2 2 2 2 2 2 1
1 1 1 1 1 1 1 1 1 1
```

```
=========================
g g g b g b g b g b
g g g g b g b g b g
b g g g g b g b g b
g b g g g g b g b g
b g b g g b g b g b
g b g b b b b g b g
b g b g b b b b g b
g b g b g b b b b g
b g b g b g b b b b
g b g b g b g b b b
average time: 0.00249629020691
total node expanded: 10000
average node per move: 555
blue score: 182
green score: 150
Blue wins!
```

*Sequence of moves omitted

superAlmondJoy.txt (20x20)

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
========================
g g g b g b g b g b g b g b g b g b g b
g g g g b g b g b g b g b g b g b g b g
b g g g g b g b g b g b g b g b g b g b
g b g g g g b g b g b g b g b g b g b g
b g b g g g g b g b g b g b g b g b g b
g b g b g g g g b g b g b g b g b g b g
b g b g b g g g g b g b g b g b g b g b
g b g b g b g g g g b g b g b g b g b g
b g b g b g b g g g g b g b g b g b g b
g b g b g b g b g g b g b g b g b g b g
b g b g b g b g b b b b g b g b g b g b
g b g b g b g b g b b b b g b g b g b g
b g b g b g b g b g b b b b g b g b g b
g b g b g b g b g b g b b b b g b g b g
b g b g b g b g b g b g b b b b g b g b
g b g b g b g b g b g b g b b b b g b g
b g b g b g b g b g b g b g b b b b g b
g b g b g b g b g b g b g b g b b b b g
b g b g b g b g b g b g b g b g b b b b
g b g b g b g b g b g b g b g b g b b b
average time: 0.0262665903568
total node expanded: 160000
average node per move: 8888
blue score: 201
green score: 199
Blue wins!
```

havn't found upper bound of board size so far

Member Contribution:

We regard all codes and report as the result of our equally hard work. We solved problem one together. On problem two and three we discuss about the algorithms together and write functions seperately, but test and debug together.