Homework 2

(due - 23:59 Mar 31, 2022)

Please submit your homework as a PDF file.

Questions: (Solve [1], [2], [3], [4] and either [5] or [6]. Total 100 points.)

20170783

Donggyu Lee

[1] For each following descriptions of Alpha-Beta Pruning, choose your answers. (10 points)

① 'Alpha' means a lower bound on the value that a maximizing node may ultimately be assigned. (True/False)

② 'Beta' means an upper bound on the value that a minimizing node may ultimately be assigned. (True/False)

③ Alpha-beta pruning is a heuristic technique to improve the minimax algorithm. (True/False)

④ At maximizing levels, prune search if values less than 'beta' are discovered. (True/False)

⑤ At minimizing levels, prune search if values greater than 'alpha' are discovered. (True/False)

[2] This question considers pruning in games with chance nodes. Figure 5.19 shows the complete game tree for a trivial game. Assume that the leaf nodes are to be evaluated in left-to-right order, and that before a leaf node is evaluated, we know nothing about its value—the range of possible values is $-\infty$ to ∞. (20 points)

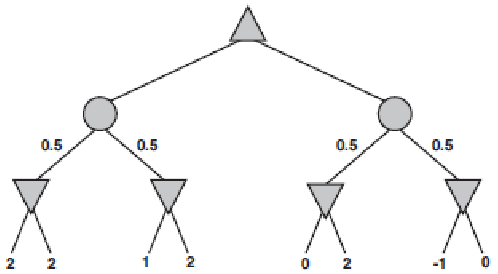
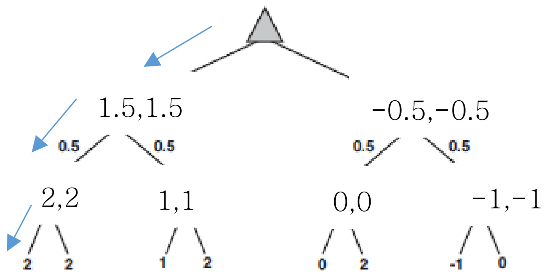


Figure 5.19 The complete game tree for a trivial game with chance nodes.

a. Copy the figure, mark the value of all the internal nodes, and indicate the best move at the root with an arrow.



b. Given the values of the first six leaves, do we need to evaluate the seventh and eighth leaves? Given the values of the first seven leaves, do we need to evaluate the eighth leaf? Explain your answers.

A. Yes on 7^{th}, No on 8^{th} because -1 is smaller than 0, and 0 is larger than -1.

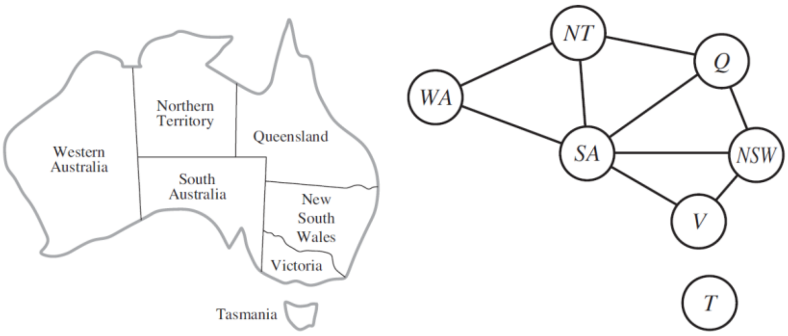
c. Suppose the leaf node values are known to lie between –2 and 2 inclusive. After the first two leaves are evaluated, what is the value range for the left-hand chance node?

A. [0,2]

d. Circle all the leaves that need not be evaluated under the assumption in (c).

A. 4^{th} (2)

[3] How many solutions are there for the map-coloring problem in Figure 6.1? How many solutions if four colors are allowed? Two colors? (10 points)



A. 3 Colors: 3(SA) * 2(NT) * 3(T) = 18

B. 4 Colors: 4(SA) * 3(WA) * 2(NT) * 2(Q) * 2(NSW) * 2(V) * 4(T) = 768

C. 2 Colors: 0

[4] Solve the cryptarithmetic problem following figure. Use backtracking with forward checking and the Minimum Remaining Value Heuristic to choose a variable and the Least Constraining Value heuristic to assign values to the selected variables for this problem. Write the solution procedures and answer. (20 points)

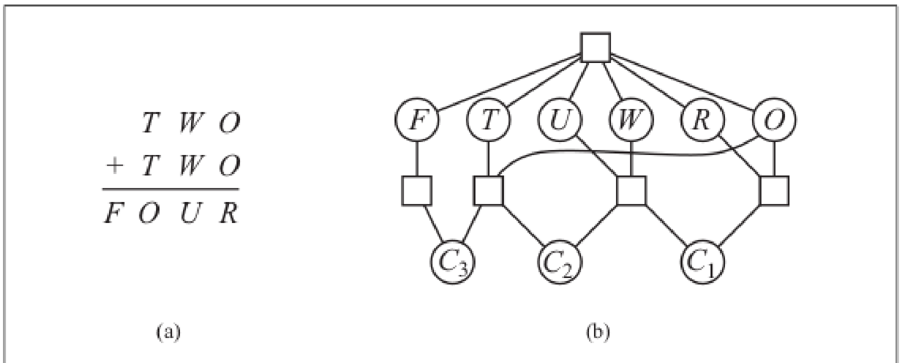


Figure 6.2 (a) A cryptarithmetic problem. Each letter stands for a distinct digit; the aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct, with the added restriction that no leading zeroes are allowed. (b) The constraint hypergraph for the cryptarithmetic problem, showing the *Alldiff* constraint (square box at the top) as well as the column addition constraints (four square boxes in the middle). The variables C_1, C_2, and C_3 represent the carry digits for the three columns.

Variables : {F, T, U, W, R, O, C1, C2, C3}

Domains : {0,1,2,3,4,5,6,7,8,9} for F, …, O; {0,1} for C1, C2, C3

Constraints:

Alldiff(F,T,U,W,R,O)

O + O = R + 10 ∗ C1

C1 + W + W = U + 10 ∗ C2

C2 + T + T = O + 10 ∗ C3

F = C3, T!=0, F!=0

A.

1. F = C3 = 1
2. C2(# domain = 2) = 1(if C2=1, W could be 5,6,7,8,9. If C2=0, W could be 1,2,3,4)
3. C1(# domain = 2) = 1 (same as C2).
4. T(# domain = 5) = 5 -> fail(O should bigger than 4)
5. T =6 -> fail(O should bigger than 4)
6. T = 7
7. O = 5 (1(C2) + 2T = 10(F) + O)
8. R = 0 (2O = 10(C1) +R)
9. W(# domain = 3) = 6
10. U =3 (1(C1) + 2W =10(C2) +U)

[5] Consider the problem of placing *k* knights on an *n*×*n* chessboard such that no two knights are attacking each other, where *k* is given and $k \le n^2$. (40 points)

a. Choose a CSP formulation. In your formulation, what are the variables?

   A. Each knight (X = {X1, X2, … Xk})

b. What are the possible values of each variable?

   A. (1~n, 1~n)

c. What sets of variables are constrained, and how?

   A. (Xi[0] +2, Xi[1] +1) != (Xj[0], Xj[1])
   B. (Xi[0] +2, Xi[1] -1) != (Xj[0], Xj[1])
   C. (Xi[0] -2, Xi[1] +1) != (Xj[0], Xj[1])
   D. (Xi[0] -2, Xi[1] -1) != (Xj[0], Xj[1])
   E. (Xi[0] +1, Xi[1] +2) != (Xj[0], Xj[1])
   F. (Xi[0] +1, Xi[1] -2) != (Xj[0], Xj[1])
   G. (Xi[0] -1, Xi[1] +2) != (Xj[0], Xj[1])
   H. (Xi[0] -1, Xi[1] -2) != (Xj[0], Xj[1])

d. Now consider the problem of putting *as many knights as possible* on the board without any attacks. Explain how to solve this with local search by defining appropriate ACTIONS and RESULT functions and a sensible objective function.

   A.

```
Function Result(){
    Int number_of_kinght =0
    While(Action){
            number_of_kinght++
    }
    Return number_of_knight
}

Function Action (variable){
    For(Int i =0, i < n, i++){
            tuple possible_location = Check(i, variable)
            if (possible_location) break
            else Return false
    }
    Add new knight to possible location
    Return True
}
List No_zones = [(2,1),(2,-1),(-2,1),(-2,-1),(1,2),(1,-2),(-1,2),(-1,-2)]
Function Check(i, variable){
    Tuple vertical, horizontal = (-i,-i)
    Tuple point = (variable[0], variable[1])
```

```
        For(vertical, vertical<i, i++){
                For(horizontal, horizontal<i, i++){
                        For (int j =0, j<No_zones.length; j++){
                                If ((point[0]+horizontal + No_zones[j][0], point[1] + vertical +
                    No_zones[j][1])) is not taken && point itself is not taken)
                                Return (point[0] + No_zones[j][0], point[1] + No_zones[j][1])
                        }
                }
        }
        Return false
    }
```

[6] Consider the problem of constructing (not solving) crossword puzzles: fitting words into a rectangular grid. The grid, which is given as part of the problem, specifies which squares are blank and which are shaded. Assume that a list of words (i.e., a dictionary) is provided and that the task is to fill in the blank squares by using any subset of the list.

Formulate this problem precisely in two ways: (40 points)

a. As a general search problem. Choose an appropriate search algorithm and specify a heuristic function. Is it better to fill in blanks one letter at a time or one word at a time?
    - BFS, Words.

b. As a constraint satisfaction problem. Should the variables be words or letters?
    - Letters.

Which formulation do you think will be better? Why?

A. General search problem with words. Starting from first letter to last letter is too waste of time.

(Posted on 2022/03/24)
