Problem 1. Non-zero-sum games

1.

(a).

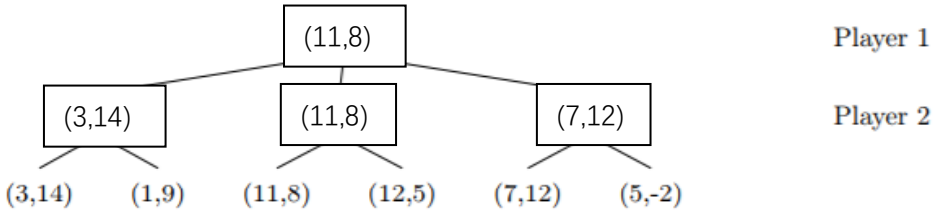


Figure 1: Non-zero sum game tree

(b).

The pruning condition is $U_B > k - \alpha$

For minimax game, each player wants to maximum his utility. For player B, he tries to minimize player A's score, which means that player B wants to maximize -U_A. The utility for player B is $U_B = -U_A$

We know that the minimizer node v should be $v < \alpha$ from alpha-beta pruning. Which can be written as $U_B = -U_A < \alpha \rightarrow U_B > -\alpha$

From condition $|U_A + U_B| \leq k \rightarrow -k - U_B \leq U_A \leq k - U_B$.

We want to prune if the upper bound is less than α, so that $k - U_B < \alpha$ and $-k - U_B < \alpha$ $\rightarrow U_B > k - \alpha$ and $U_B > -k - \alpha$

Where $-k - \alpha$ must less than $k - \alpha$ because $k \geq 0$ which $|U_A + U_B| \geq 0$

So that $U_B > k - \alpha$

Problem 2. Local Search

(a)

There will be 6 cases for delegates choosing the bowls. Suppose that delegates are a, b, c and the bowls are $1,2,3$

The choosing cases are $a1b2c3, a2b1c3, a2b3c1, a1b3c2, a3b1c2, a3b2c1$

For each case, denoted as c_k, k is from 1 to 6, the score function is below:

Suppose the function $C(i,n,j)$ be the delegate n choosing M&Ms, which i is the number of MMs in the bowl. Each time of choosing M&Ms, it has the color, red, blue, green and denoted as 1, 2, 3 for red, blue, and green. show as j. For each delegate, the delegate who prefer red MMs, denoted as 1, and so on.

The score function should be:

$$f(c_k) = \sum_{n=1}^{3} C(i,n,j) \qquad if\ n \neq j$$

$$Otherwise, f = 0$$

The score function should be the minimum of these 6 cases.

Which should be $score\ function = \min(f(c_1), f(c_2), f(c_3), f(c_4), f(c_5), f(c_6))$

Take the MMs from first bowl, if the color is green, put it in the second bowl, if the color is blue, put it in the third bowl. Do the same thing for second bowl, if the color is red, put it in first bowl, and if it is blue, put it in the third one. For the third bowl, if the color is red, put it

in first bowl, and if it is green, put in second bowl.

(b).
Suppose that there are 12 MMs, 4 red, 4 blue and 4 green. If the 3 lines are ordered originally like:

$$\mathrm{R\ R\ G\ G}$$
$$\mathrm{R\ R\ B\ B}$$
$$\mathrm{G\ G\ B\ B}$$

The score function of it is 6.
For every time swap the MMs, there is no increasing or decreasing of score function.
Until we have 3 lines of MMs in 3 separated colors, the score function is still 6. There is no increasing or decreasing. In this situation, it considered as the local maximum for the score function. In this way, we cannot use this score function to converge on the goal of bowls full of one M&M color.

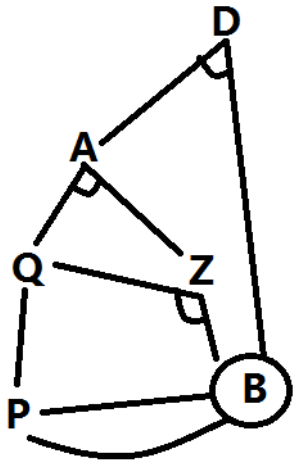
(c)
Every time inserting M&Ms from one line into an arbitrary location in another line, it creates the random probability, which will accept the situation of some little bad case than greedily choose of MM. it will allow some probability of jumping out of the local maxima and get the global maximum of this score function.

Problem 3. Propositional Logic

$$A \wedge B \Rightarrow D$$
$$Q \wedge -R \Rightarrow A$$
$$-Q \vee -B \vee -R$$
$$P \Rightarrow Q$$
$$P \Leftrightarrow B$$
$$B$$

Convert to horn form:
Assume $-\mathrm{R} = \mathrm{Z}$

$$\mathrm{A} \wedge \mathrm{B} \Rightarrow \mathrm{D}$$
$$\mathrm{Q} \wedge \mathrm{Z} \Rightarrow \mathrm{A}$$
$$\mathrm{Q} \wedge \mathrm{B} \Rightarrow \mathrm{Z}$$
$$\mathrm{P} \Rightarrow \mathrm{Q}$$
$$(\mathrm{P} \Rightarrow \mathrm{B}) \wedge (\mathrm{B} \Rightarrow \mathrm{P})$$
$$\mathrm{B}$$



(a)

Forward chaining:

B is true from knowledge base, so that P is true. P is true so that Q is true. Q and B are true so that Z is true. Q and Z are true so that A is true. A and B are true so that D is true.

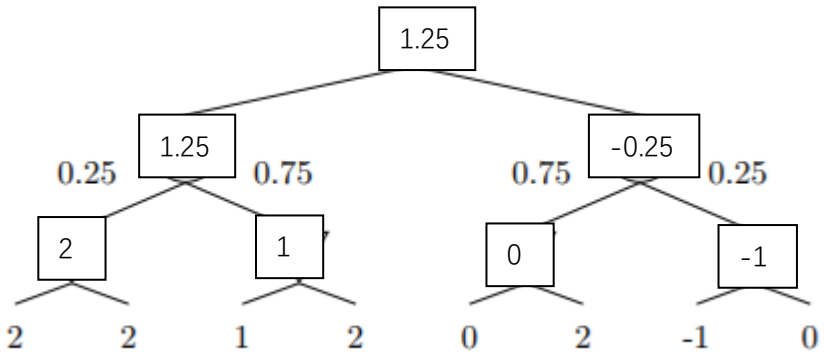
(b)

Backward chaining

Assume D is true. So that we need to know A and B are true. Assume A is true so that we need to know Q and Z are true. Assume Z is true so that we need to know Q and B are true. Assume Q is true so that we need to know P is true. Assume P is true so B is true. B is true is our knowledge base. Proved by backward chaining.

Problem 4. Uncertainty – Expectimax

(a).



(b).

The value of left branch is 1.25. We do need to evaluate the seventh and eighth leaves. if the leaf values are very large, we cannot ignore them because it might be larger than the chance of first 4 leaves which is 1.25. In this way, it will lead to a wrong root action. If we know that

the value of 7th node, which is -1, we know that the right branch value is at most -0.25. in this way, we do not need to know the 8th leaf value.

(c)

We need to know the first 5 leaf values. first 4 values will show that the left branch's value which is 1.25. Knowing the 5th leaf value is 0, whatever the 6th leaf node is, the minimum value of them is 0 or less than 0, and the chance of this branch is at most 0. If the 7th and 8th leaf nodes are all at most 2, the chance of them is 0.5, which means the right branch of this tree is at most 0.5. we do not need to know the last 3 leaf nodes.

(d).

Suppose we have n leaf nodes which are $\{y_1, y_2, \dots \dots y_n\}$.

For every leaf, it is positive linear transformation, which is $y_i = ax_i + b, a > 0$

Rewrite the leaf nodes which is $\{ax_1 + b, ax_2 + b, \dots \dots ax_n + b\}$

Subtract b in each element, we will get $\{ax_1, ax_2, \dots \dots ax_n\}$

Because $a > 0$, divide a in each element, we get $\{x_1, x_2, \dots \dots x_n\}$

It didn't change the order of the values and didn't change the decision for expectimax search. So that the that expectimax search decisions are insensitive to positive linear transformations of the leaf values in the tree