9 Game Playing

Syllabus

Overview, MiniMax, Alpha-Beta Cut-off, Refinements, Iterative deepening.

Contents

9.1 What is Game ? - The Adverserial Search

GTU : Summer-12, 14, 15, 18, Winter-12, 15, 17

- **In adversarial search problem environment is multiagent, competitive where in the agent's goals are in coflict.** Adversarial search problems are commonly known as **games.**
- Mathematical game theory a branch of economics views any multiagent environment as a game, provided that, impact of each agent on the others is "significant" regardless of whether the agent are cooperative or competitive.
- The term game means a sort of conflict in which n individuals or groups (known as players) participate.
- Game theory denotes games of strategy.
- **John Von Neumann** is acknowledged as **father of game theory.** Neumann defined game theory in 1928 and established the mathermatical framework for all subsequent theoretical developments.
- Game theory allows decision-makers (players) to cope with other decision-maker (players) who have different purposes in mind. In other words, players determine their own strategies in terms of the strategies and goal of their opponent.
- Games are integral attribute of human beings. Games engage the intellectual faculties of humans.
- If computers are to mimic people they should be able to play games.

Game playing has close relation to intelligence and it has well-defined states and rules.

9.2 Applications of Game Theory

- Applications of game theory are wide-ranging. **Von Neunmann** and **Morgenstern** indicated the utility of game theory by linking with ecomomic behavior.

1. Economic models

- For markets of various commodities with differing numbers of buyers and sellers, fluctuating values of supply and demand, seasonal and cyclical variations, analysis of conflicts of interest in maximizing profits and promoting the widest distribution of goods and services.

2. Social sciences

- The n-person game theory has interesting uses in studying the distribution of power in legislative procedures, problems of majority rule, individual and group decision making.

3. Epidemiologists

- Make use of game theory, with respect to immunization procedures and methods of testing a vaccine or other medication.

4. Military strategists

- Turn to game theorey to study conflicts of interest resolved through "battles" where the outcome or payoff of a war game is either victory or defeat.

9.3 Definition of Game

1. A game **has at least two** players. **Solitaire is not considered a game by game theory.**

   The term "solitaire" is used for single-player games of concentration.

2. An instance of a game begins with a player, choosing from a set of specified (game rules) alternatives. **This choice is called a move.**

3. After first move, the new situation determines which player to make next move and alternatives available to that player.

   i) In many board games, the next move is by other player.

   ii) In many multi-player card games, the player making next move depends on who dealt, who took last trick, won last hand, etc.

4. The moves made by a player may or may not be known to other players. Games in which all moves of all players are known to everyone are called **games of perfect information.** For example,

   i) Most board games are games of perfect information.

   ii) Most card games are not games of perfect information.

5. Every instance of the game must end.

6. When an instance of a game ends, each player receives a payoff. A **payoff** is a value associated with each player's final situation. A zero-sum game is one, in which elements of payoff matrix sum to zero. In typical zero-sum game :

   i) Win = 1 point,

   ii) Draw = 0 point,

   iii) Loss = – 1 point

9.4 Game Theory

- Game theory does not prescribe a way or say how to play a game. Game theory is a set of ideas and techniques for analyzing conflict situations between two or more parties. The outcomes are determined by their decisions.

General Game Theorem

- In every two players, game like zero sum, non-random, perfect knowledge game there exists a perfect strategy guaranteed to at least result in a tie game.

The frequently used terms in game theory : -

- The term "**game**" means a sort of conflict in which n individuals or groups (known as players) participate.
- A list of "**rules**" stipulates the conditions under which the game begins.
- A game is said to have "**perfect information**" if all moves are known to each of the palyers involved.
- A "**strategy**" is a list of the optimal choices for each player at every stage of a given game.
- A "**move**" is the way in which game progresses from one stage to another, beginning with an initial state of the game to the final state.
- The total number of moves constitute the entirety of the game.
- The **payoff** or **outcome**, referes to what happens at the end of a game.
- **Minimax : The least good of all good outcomes.**
- **Maximin : The least bad of all bad outcomes.**

- **The important and basic game theory theorem is the mini-max theorem. This theorem says,**
  "If a minimax of one player corresponds to a maximin of the other player, then that outcome is the best that both players can hope for."

9.5 Relevance of Game Theory and Game Playing

How relevant the game theory is to mathematics, computer science and economics is shown in the figure below :

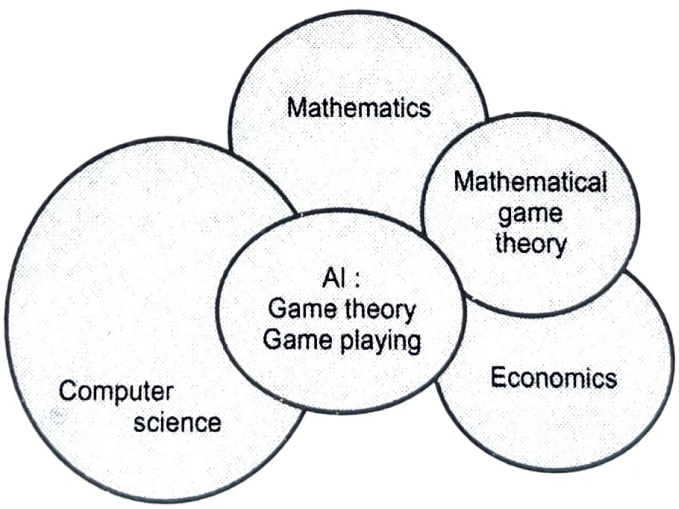

Fig. 9.5.1 Game theory and its relevance to other fields

9.6 Game Playing

Characteristics of Game Playing : -

1. There is always an "unpredictable" opponent
   - Opponent introduces uncertainty.
   - Opponent also wants to win.
2. Time limit

Games are always played in time constraint environment. Therefore is needs to handle time efficiently.

9.7 Types of Games

1. Based on chance

i) Deterministic (not involving chance)

For example -

Chess, Checkers, Tic-tac-toe

ii) Non-deterministic (can involve chance)

For example -

Backgammon, Monopoly.

2. Based on information

i) Perfect information -

Here all moves of all players are known to everyone.

For example -

Chess, Checker, Tic-tac-toe.

ii) Imperfect information -

Here all moves are not known to everyone.

For example -

Bridge, Pocker, Scrabble.

3. General zero-sum games

Players must choose their strategies simultaneously, neither knowing what the other player is going to do.

For example -

If you play a single game of chess with someone, one person will lose and one person will win. The win (+1) added to the loss (– 1) equals zero.

4. Constant - sum game

Here the algebraic sum of the outcomes are always constant, though not necessarily zero.

It is strategically equivalent to zero-sum games.

5. Non-zero - sum game

Here the algebraic sum of the outcomes are not constant. In these games, the sum of the payoffs are not the same for all outcomes.

They are not always completely solvable but provide insights into important areas of inter-dependent choice.

In these games, one player's losses do not always equal another player's gains.

The non-zero-sum games are of two types : -

i) Negative sum games (Competitive) -

Here nobody really wins, rather everybody loses.

Example - A war or a strike.

ii) Positive sum games (Co-operative) -

Here all players have one goal that they contribute together.

Example - An educational game, building blocks, or a science exhibit.

6. N-person game

It involves more than two players.

Analysis of such games is more complex than zero-sum games.

Conflicts of interest are less obvious.

9.8 Formal Representation of a Game as a Problem

9.8.1 A Game is Essentially a Kind of a Search Problem !

Game is formally defined with following four components : -

1. The initial state, which includes the board position and identifies the player to move.
2. A successor function, which returns a list of (move, state) pairs, each indicating a legal move and the resulting state.
3. A terminal test, which determines when the game is over. States where the game has ended are called as **terminal states.**
4. A utility function (also called an objective function or payoff function), which gives a numeric value for the terminal states. In chess, the outcome is a win, loss

or draw, with values +1, – 1, or 0. Some game have a wider variety of possible outcomes; the payoffs in backgammon range from + 192 to – 192.

9.8.2 Mixed Strategies

- A players's strategy in a game is a complete plan of action for whatever situation might arise. It is a complete algorithm for playing the game, telling a player what to do for every possible situation throughout the game.
- **A pure strategy** provides a complete definition of how a player will play a game. In perticular, it determines the move a player will make for any situation they could face. A player's strategy set, is the set of pure strategies available to that player.
- **A mixed strategy** is an assignment of a probabiblity to each pure strategy. This allows for a player to randomly select a pure strategy. Since probabilities are contineous, there are infinitely many mixed strategies available to a player, even if their strategy set is finite.
- A mixed stretegy for a player is a probability distribution, on the set of his pure strategies.

9.8.3 The Game Tree

The initial state and the legal moves for each side, define the game tree for the game.

- **Description of the game tree** [Refer Fig. 9.8.1 on next page].

1. **Root node -**

Represents board configuration and decision, required as to what is the best single next move.

If my turn to move, then the root is labeled a MAX node indicating it is my turn;

Otherwise it is labeled a MIN, node to indicate it is my opponent's turn.

2. **Arcs -**

Represent the possible legal moves for the player that the arcs emanate from.

3. At each level, the tree has nodes that are all MAX or all MIN.

4. Since moves alternate, the nodes at level 'i' are of opposite kind from those at level i + 1.

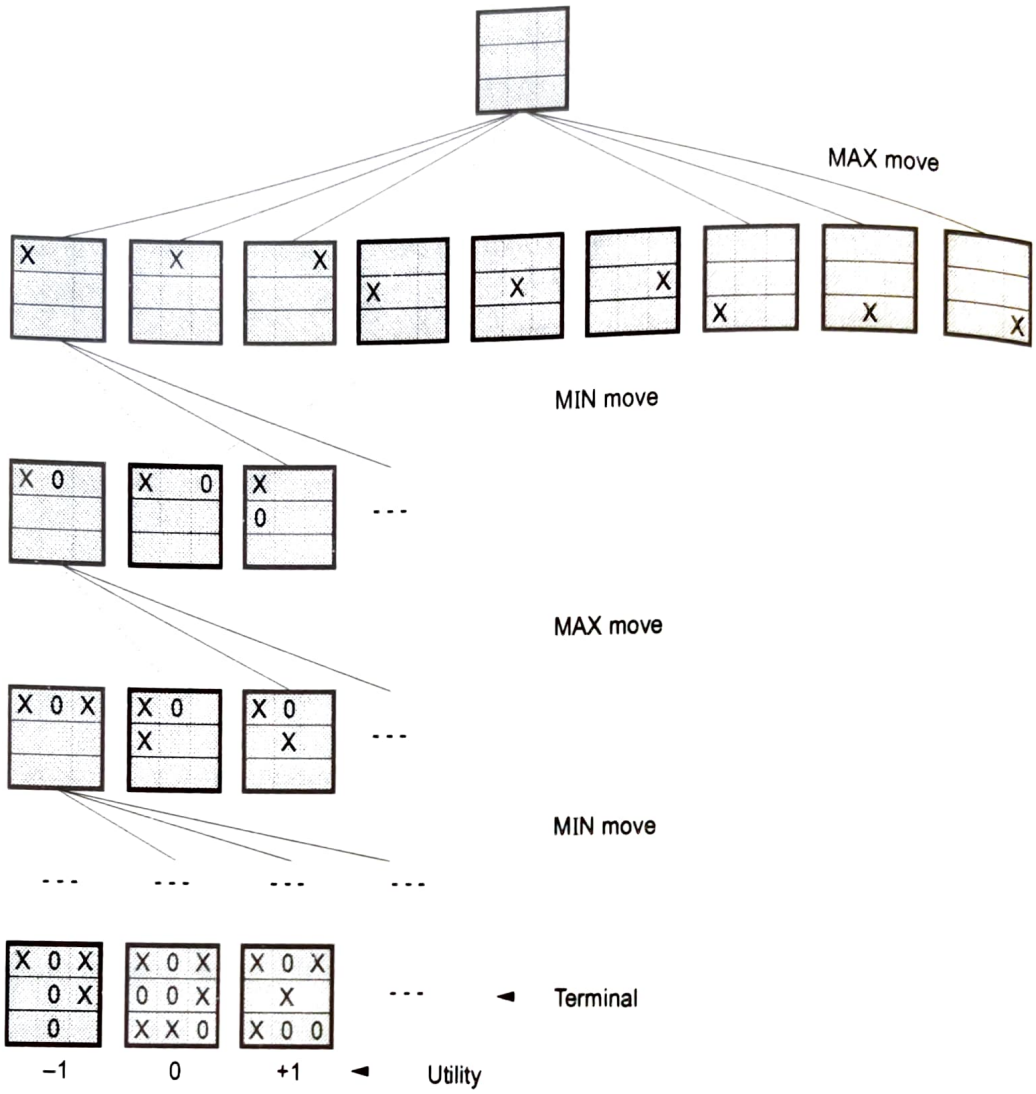

Fig. 9.8.1 Example of game tree

9.9 Game Strategy

9.9.1 Concept of Strategy

A player's strategy in a game, is a complete plan of action for whatever situation might arise. It is the complete description of how one will behave under every possible circumstances. You need to analyze the game mathematically and create a table with "outcomes" listed for each strategy.

A Two Player Strategy Table.

Players Strategies	Player A Strategy 1	Player A Strategy 2	Player A Strategy 3	etc.
Player B strategy 1	Tie	A wins	B wins	...
Player B strategy 2	B wins	Tie	A wins	...

Player B strategy 3	A wins	B Wins	Tie	...
etc.	...	...	...	...

1. Above is a partial tree for the game of tic-tac-toe.
2. The top node (root) is the initial state and MAX (player 1) moves first, placing X in an empty square.
3. The rest of the search tree shows alternate moves for MIN (player 2) and MAX.
4. Terminal states are assign utilities according to the rules of game.

9.9.2 Optimal Strategies

It is a techniques which always leads to superior solution than any other strategy, as opponent is playing in perfect manner.

Roughly speaking, an optimal strategy leads to outcomes that are at least as good as any other strategy when one is playing an infallible opponent.

Mini-Max Value

The minimax value for a given game tree is determined by optimal strategy by evaluating minimax value of each node.

Mini-Max Theorem

Players adopt those strategies which will maximize their gains, while minimizing their losses. Therefore the solution is, the best each player can do for him/herself in the face of opposition of the other player.

Determining Optimal Strategy

1. Given a game tree the optimal strategy can be determined by examining the minimax value of each node.
2. The minimax value of a node is the utility (for player called MAX) of being in the corresponding state, assuming that both players play optimally from this stage to the end of the game.
3. The minimax value of a terminal state is just its utility.
4. Given a choice, MAX will perfer to move to a state of maximum value, whereas MIN prefers a state of minimum value.

9.10 Mini-Max Algorithm

GTU : Winter-12,15,18,19, Sumemr-14,18,19,20

The minimax algorithm computes the minimax decision from the current state. It is used as a searching technique in game problems. The minimax algorithm performs a complete depth-first exploration of the game-tree.

9.10.1 The Algorithm

1. The start node is MAX (player 1) node with current board configuration.
2. Expand nodes down (play) to some depth of look-ahead in the game.
3. Apply evaluation function at each of the leaf nodes.
4. "Back up" values for each non-leaf nodes until computed for the root node.
5. At MIN (player 2) nodes, the backed up value is the minimum of the values associated with its children.
6. At MAX nodes, the backed up value is the maximum of the values associated with its children.

Note :

The process of "backing up" values gives the optimal strategy, that is, both players assume that your opponent is using the same static evaluation function as you are.

9.10.2 Properties of Mini-Max

1. Minimax provides a complete solution for finite tree.
2. Minimax provides optimal strategy against an optimal opponent.
3. The time complexity is O (b^m).
4. The space complexity is O(bm) for depth-first exploration where algorithm generates all successor at once, or O(m) for an algorithm that generate successors one at a time.

9.10.3 Problem Associated with Mini-Max

This algorithm explores whole search space. If we have a game with huge search spaces it will take long time.

In such cases then exact solution is completely infeasible.

9.10.4 Game Playing with Mini-Max-Tic-Tac-Toe [Noughts and Crosses] Example

1. Assume that two players named MIN and MAX who are playing the game.
2. MAX is playing first.
3. As you can see initially MAX has nine possbile moves.
4. Play alternates between MAX and MIN until we reach leaf nodes corresponding to terminal states such that one player has 3 in a row or all the squares are filled.
5. The number on each leaf node indicates the utility value of the terminal state from the point of view of MAX.
6. High values are assumed to be good for MAX and bad for MIN.
7. It is MAX's job to make use of search tree to determine best move.
8. Static evaluation. Criteria - '+ 1' for a win, '0' for draw.

9.10.5 Example to Show How Mini-Max Algorithm Works

Consider game of tic-tac-toe with the initial state -

0	0	X
X		0
		X

Step 1 -

Start : MAX (player 1) moves (MAX is making X)

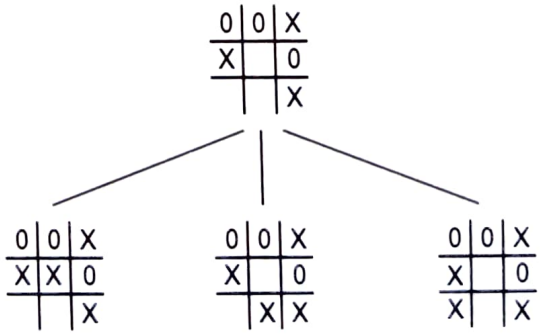

Fig. 9.10.1 Max playing (X)

Step 2 -

Next : MIN (player 2) moves.

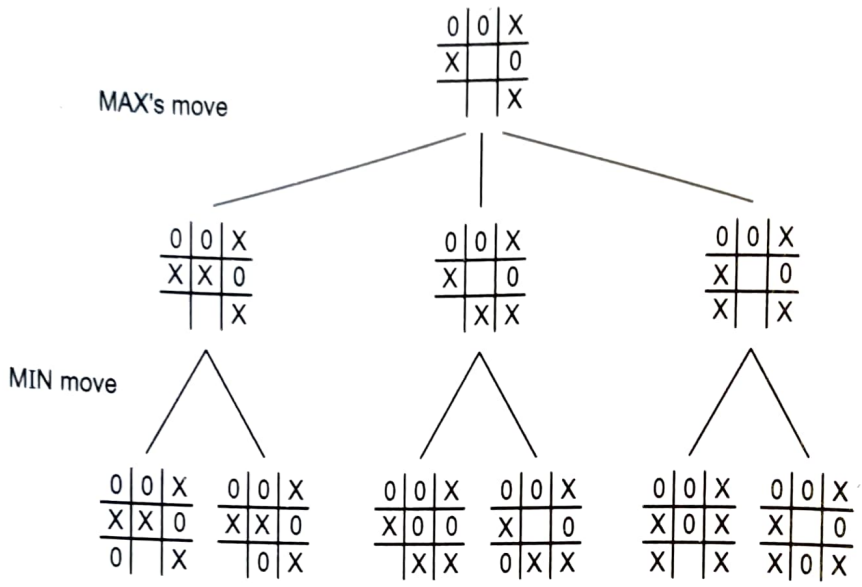

Fig. 9.10.2 Min playing (0)

Step 3 -

Again : MAX's moves

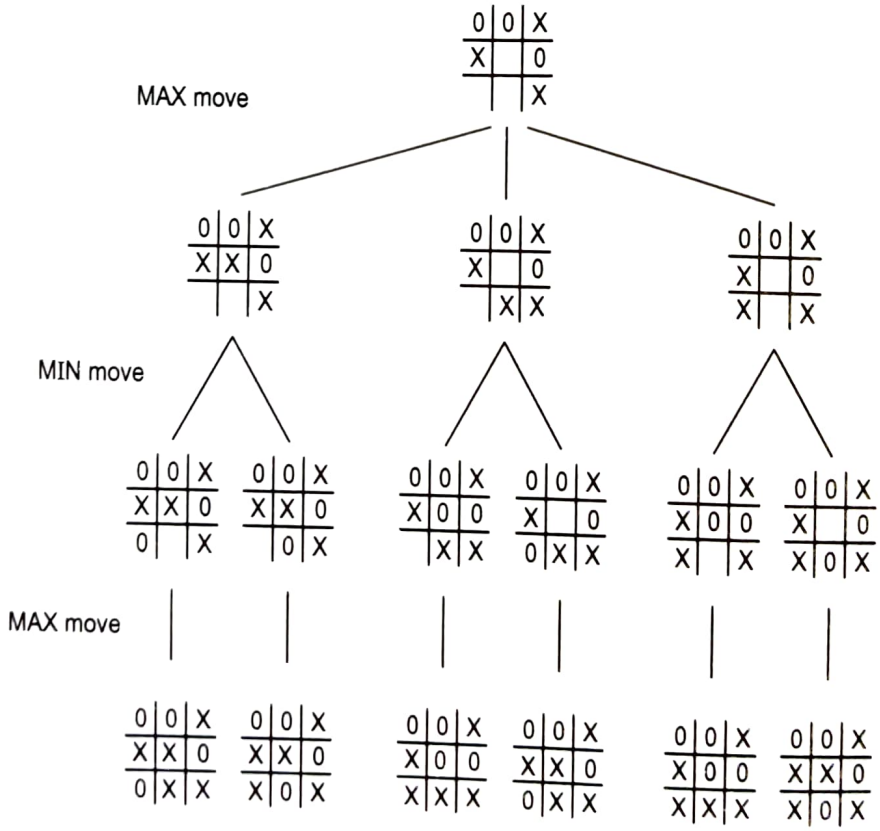

Fig. 9.10.3 Max playing (X)

Step 4 -

'+ 1' for a win, '0' for a draw. Criteria '+ 1' for a win, '0' for a draw.

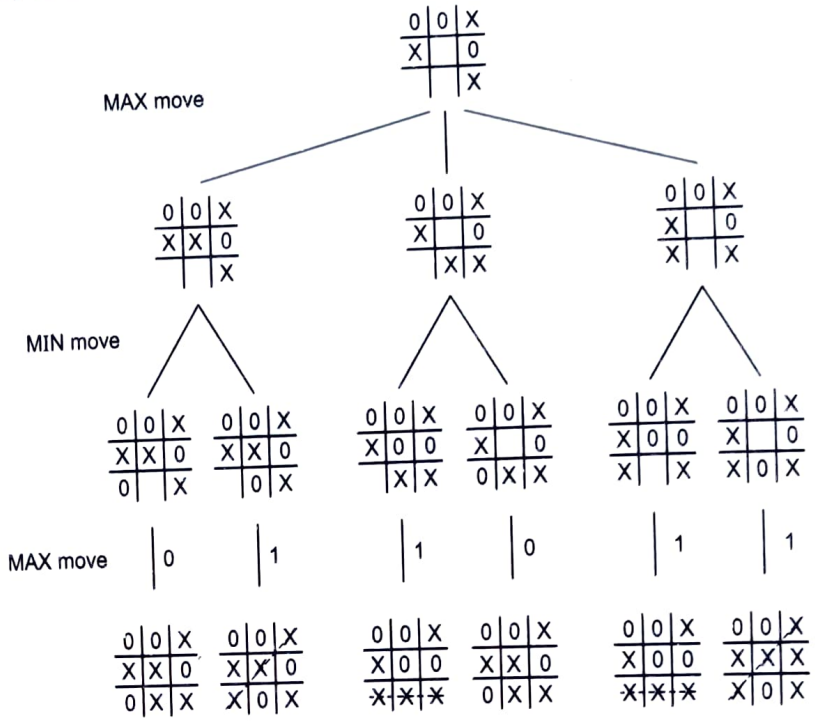

Fig. 9.10.4 Min playing (0)

Step 5 -

Level by level, on the basis of opponents turn UP : One level

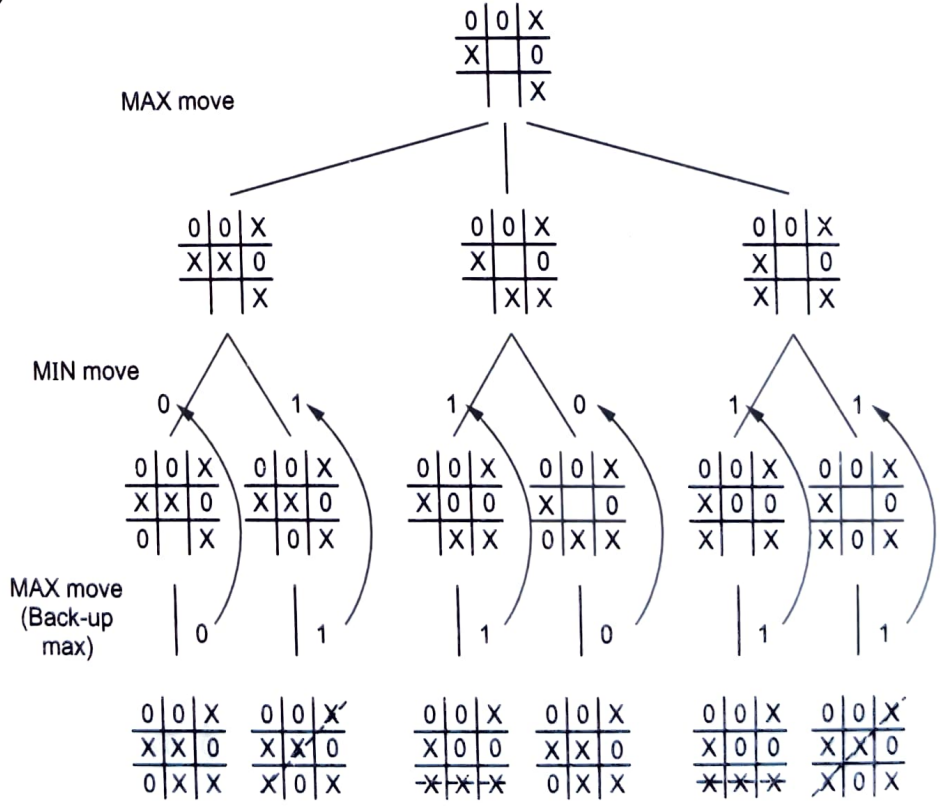

Fig. 9.10.5 One level up

Step 6 -

UP : Two levels

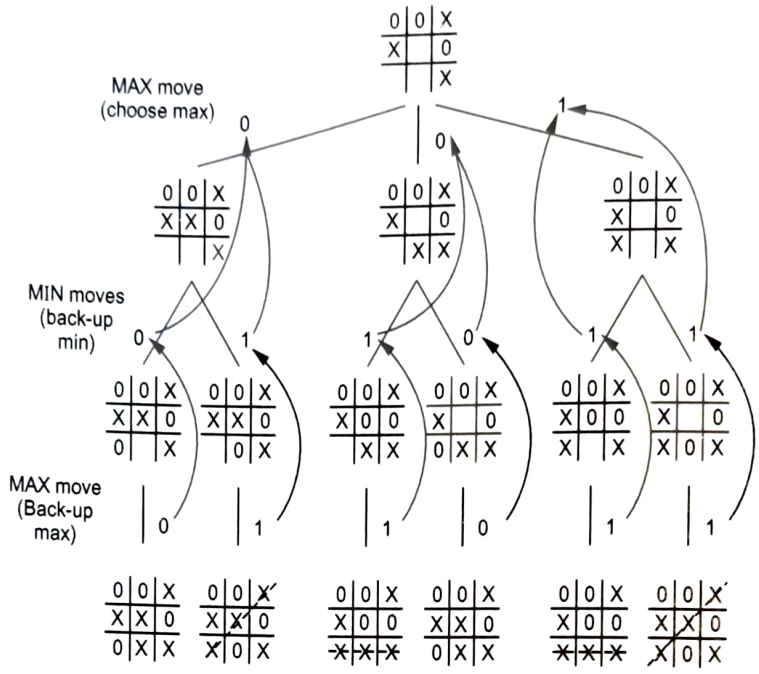

Fig. 9.10.6 One level up

Step 7 - Choose best move which is maximum

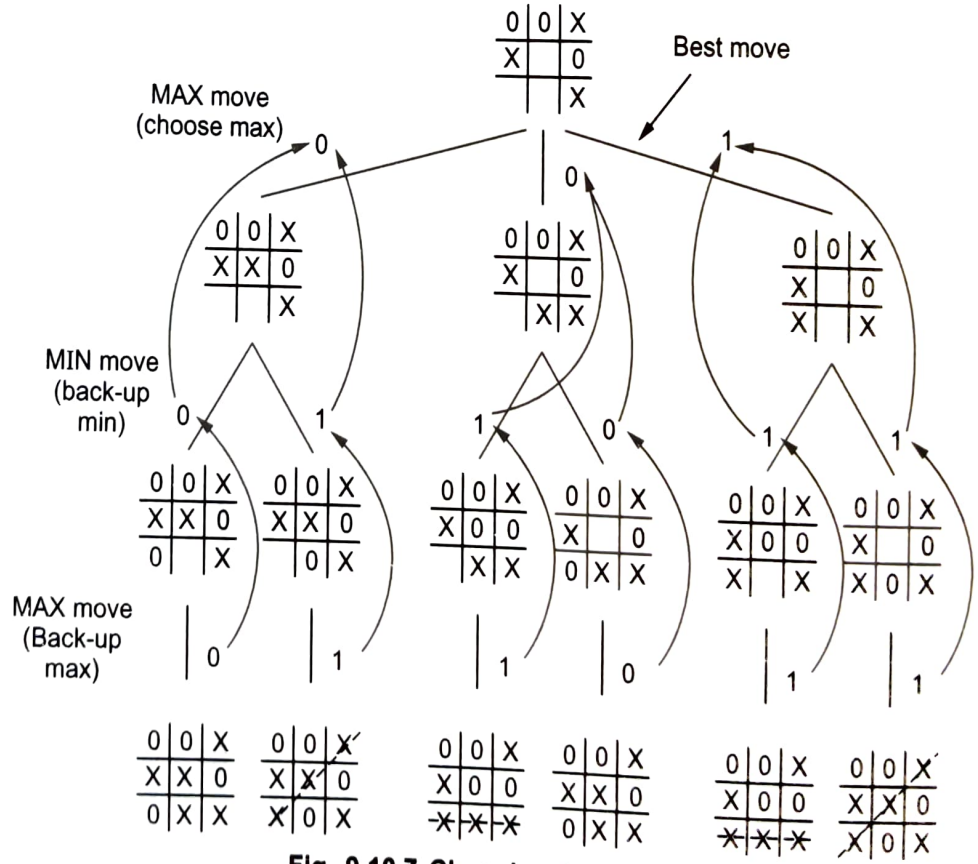

Fig. 9.10.7 Choosing best move

9.10.6 The 'made-up' Games and Concept of 'ply'

- For study purpose we can form a game a tree which is constructed up to certain level (i.e. upto certain depth). This is called as a **made-up game.** The term ply refers to depth of the tree.
- For example - ply 4 is level at depth 4 **below the root node.**

Example

A two ply game tree, for game of tic-tac-toe.

1. Assume that two players named MIN and MAX who are playing the game.
2. MAX is playing first.
3. The possible moves for MAX at the root node are labeled a_1, a_2 and a_3.
4. The possible replies to a_1 for MIN are b_1, b_2, b_3 and so on.
5. This particular game ends after one move each by MAX and MIN.
6. In game parlance, we say that this tree is one which, moves deep consisting of two-half-moves, each of which is called a **ply.**

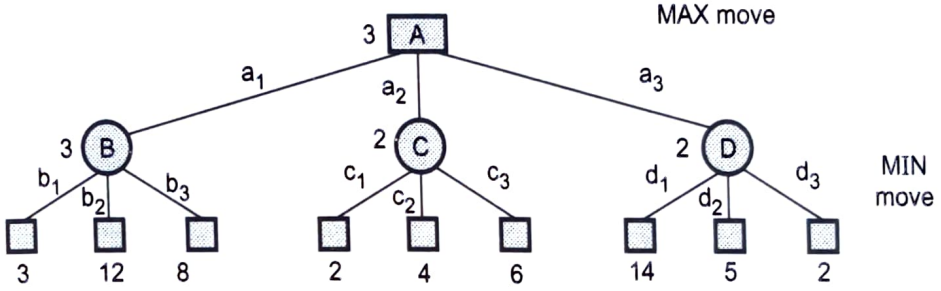

Fig. 9.10.8 Game tree

Example

A three-ply game tree, for game of tic-tac-toe

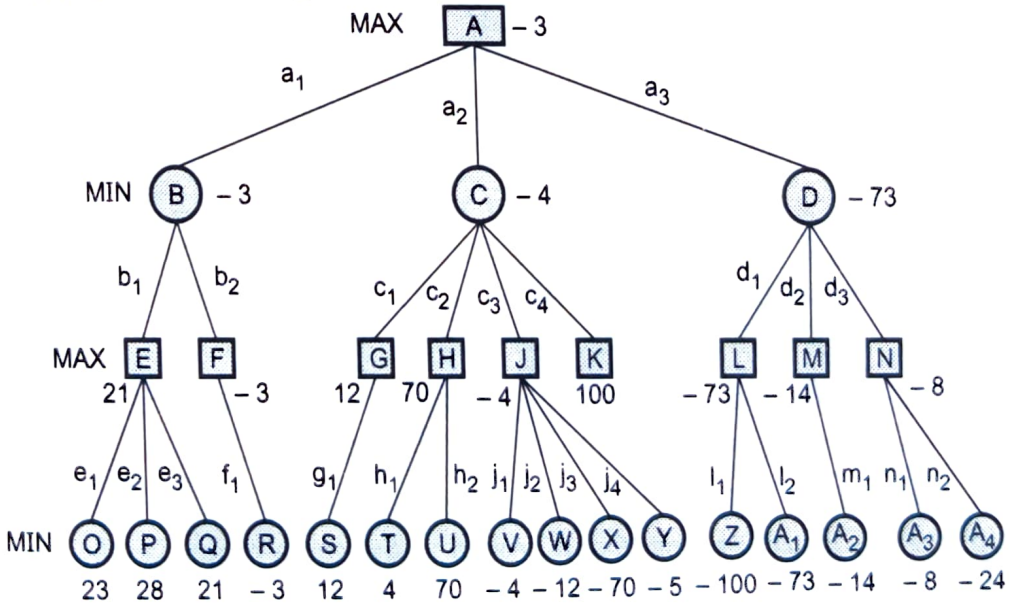

Fig. 9.10.9 Three ply game tree

9.10.7 Mini-Max Algorithm for Playing Multiplayer Games

1. There are many multiplayer games like cricket, football, etc.
2. The multiplayer game can be played with minimax concept.
3. Here the single value for each node is replaced with a vector of values. For example, in a three-player game with players A, B, and C a vector (V_A, V_B, V_C) is associated with each node.
4. For terminal states, this vector gives the utility of the state from each player's viewpoint. (In two player, zero-sum games, the two-element vector can be reduced to a single value because the values are always opposite.
5. In this implementation UTILITY function return a vector of utilities.
6. For non-terminal states we can calculate utility function value as explain below -

Consider following diagram,

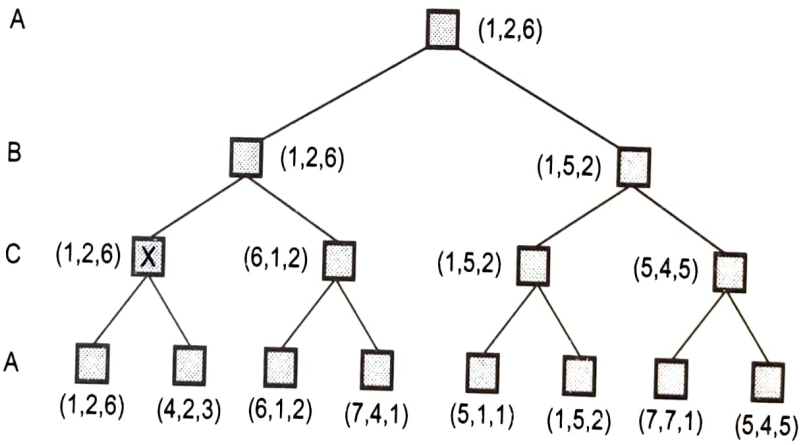

Fig. 9.10.10 Utility function value calculation

7. For non-terminal states vector values should be calculated as explained.

   Consider the node marked X in the game tree shown in above diagram. In this state, player C choose what to do. The two choices lead to terminal states with utility vector ($V_A = 1$, $V_B = 2$, $V_C = 6$) and ($V_A = 4$, $V_B = 2$, $V_C = 3$). Since 6 is bigger than 3, C should choose the first move. This means that if state X is reached subsequent play will lead to a terminal state with utilities ($V_A = 1$, $V_B = 2$, $V_C = 6$). Hence the backed-up value of X is this vector.
8. In general, the backed-up value of a node 'n' is the utility vector of whichever successor has the highest value for the player choosing at 'n'.
9. Multiplayer games usually involve alliance, whether formal or informal, among the players. Alliances are made and broken as the game proceeds.

9.11 Alpha-Beta Pruning

GTU : Summer-12,18, Winter-17,18,19

9.11.1 Motivation for α - β Pruning

1. The problem with minimax algorithm search is that the number of game states it has to examine is exponential in the number of moves.
2. α-β proposes to compute the correct minimax algorithm decision without looking at every node in the game tree.

α-β Pruning Example :

Step 1 :

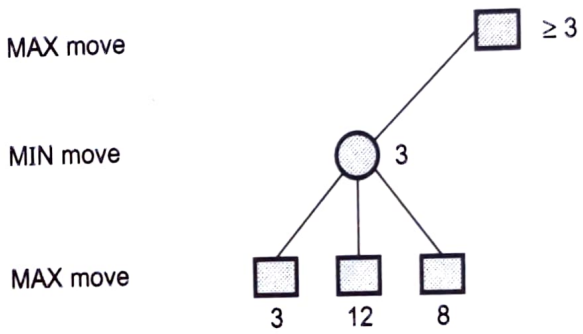

Fig. 9.11.1 Pruning example (i)

Step 2 :

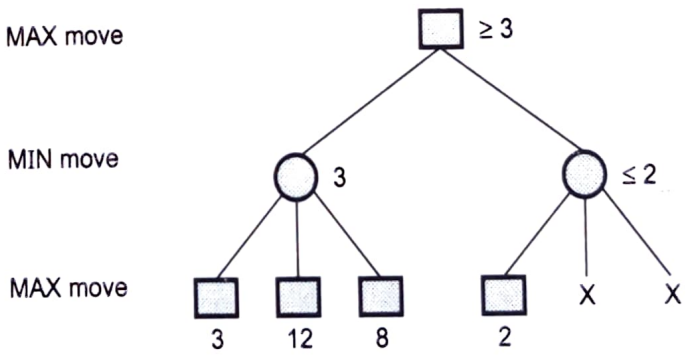

Fig. 9.11.2 Pruning example (ii)

Step 3 :

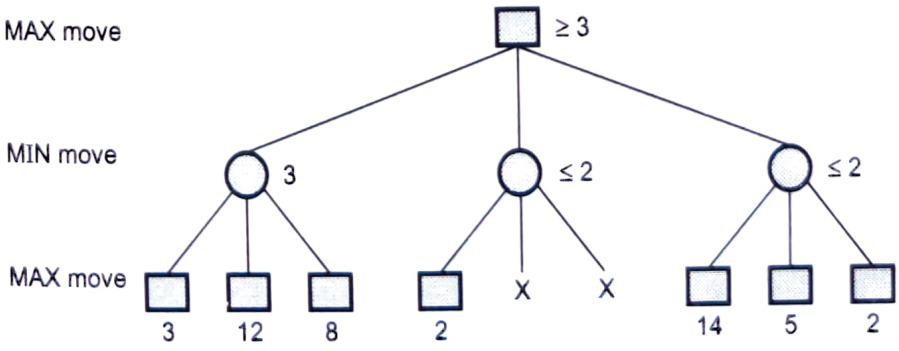

Fig. 9.11.3 Pruning example (iii)

9.11.2 Steps in Alpha-Beta Pruning

1. MAX player cuts off search when he knows MIN-player can force a provably bad outcome.
2. MIN player cuts of search when he knows MAX-player can force provably good (for MAX) outcome.
3. Applying an alpha-cutoff means we stop search of a particular branch because we see that we already have a better opportunity elsewhere.
4. Applying beta-cutoff means we stop search of a particular branch because we see that the opponent already has a better opportunity elsewhere.
5. Applying both forms is alpha-beta pruning.

9.11.3 Alpha Cutoff

It may be found that, in the current branch, the opponent can achieve a state with a lower value for us than one achievable in another branch. So the current branch is one that we will certainly not move the game. Search of this branch can be safely terminated.

For example -

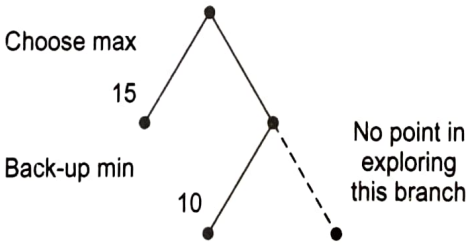

Fig. 9.11.4 Alpha cutoff

9.11.4 Beta-Cutoff

It is just the reverse of alpha-cutoff.

It may also be found, that in the current branch, we would be able to achieve a state which has a higher value for us than one of the opponent can hold us to in another branch. The current branch can be identified as one that the opponent will certainly not move the game. So search in this branch can be safely terminated.

For example - (Refer Fig. 9.11.5)

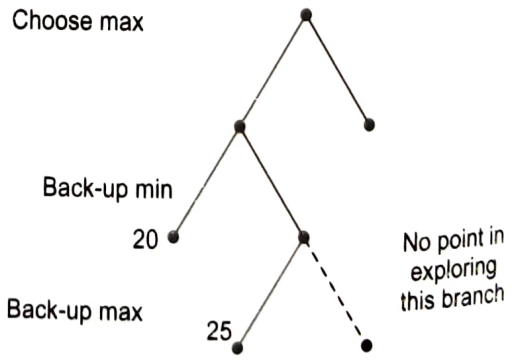

Fig. 9.11.5 Beta cutoff

9.11.5 Algorithm of Alpha-Beta Pruning

/*

alpha is the best score for max along the path to state.

beta is the best score for min along the path to state.

*/

If the level is the top level, let alpha = – infinity, beta = + infinity.

If depth has reached the search limit apply static evaluation function to state and return result.

If player is max :

Until all of state's children are examined with ALPHA-BETA or until alpha is equal to or greater than beta :

Call ALPHA-BETA (child, min, depth + 1, alpha, beta);

note result

Compare the value reported with alpha; if reported value is larger reset alpha to the new value.

Report alpha

If player is min :

Until all of state's children are examined with ALPHA-BETA or until alpha is equal to or greater than beta :

Call ALPHA-BETA, (child, max, depth + 1, alpha, beta);

note result.

Compare the value reported with beta; if reported value is smaller, reset beta to the new value.

Report beta.

9.11.6 Example of Alpha-Beta Pruning (Upto 3rd Ply)

1. In a game tree, each node represents a board position where one of the player gets to choose a move.
2. For example, look at node C in Fig. 9.11.6, as well as look at its left child.
3. We realize that if the players reach node C, the minimizer can limit the utility to 2. But the maximizer can get utility 6 by going to node B instead, so he would never let the game reach C. Therefore we don't even have to look at C's other children.

4. Initially at the root of the tree there is no gurantee about what values the maximizer and minimizer can achieve. So beta is set to ∞ and alpha to – ∞.
5. Then as we move down the tree, each node starts with beta and alpha values passed down from its parent.
6. If it's a maximizer node, then alpha is increased if a child value is greater than the current alpha value. Similarly, at a minimizer node, beta may be dereased. This procedure is shown in Fig. 9.11.6.

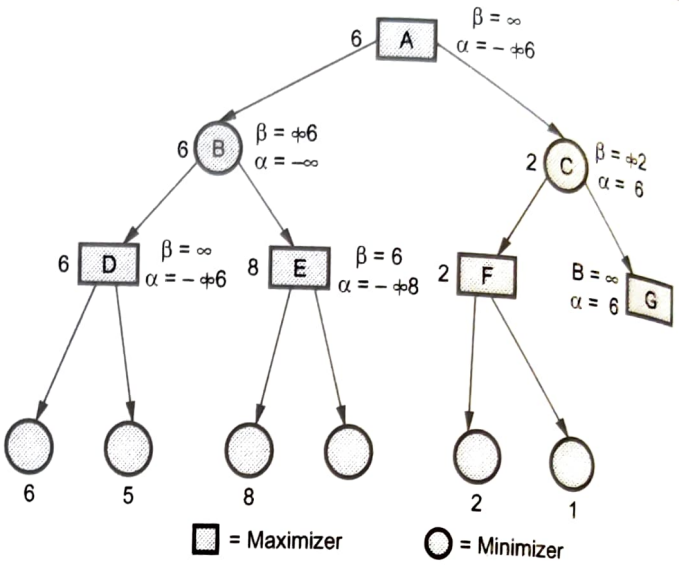

Fig. 9.11.6 Alpha beta pruning

7. At each node, the alpha and beta values may be updated as we iterate over the node's children. At node E, when alpha is updated to a value of 8, it ends up exceeding beta.
8. This is a point where alpha-beta pruning is required we know the minimizer would never let the game reach this node, so we don't have to look at its remaining children.
9. In fact, pruning happens exactly when alpha becomes greater than or equal to beta - that is, when the alpha and beta lines hit each other in the node value.

9.12 Refinements and Heuristic for Cutting Off Search

9.12.1 Evaluation Functions

1. An evaluation function returns an estimate of the expected utility of the game from a given position, just as the heuristic functions return an estimate of the distance to the goal.
2. It should be clear that the performance of a game-playing program is dependant on quality of its evaluation function. An inaccurate evaluation function will guide an agent toward positions that turn out to be lost.

3. Designing evaluation function :

a) The evaluation function should order the terminal states in the same way as the true utility function; otherwise, an agent using it might select suboptimal moves even if it can see ahead all the way to the end of the game.

b) The computation must not take too long !

c) For nonterminal states, the evaluation function should be strongly correlated with the actual chances of winning.

9.12.2 Imperfect and Real-time Decisions

- The minimax algorithm generates the entire game search space, whereas the alpha-beta algorithm allow us to prune large parts of it. However, alpha-beta still has to search all the way to terminal states for at least a portion of the search space. This depth is usually not practical, because moves must be made in a reasonable amount of time-typically a few minutes at most.
- For making search faster we can make a heuristic evaluation function that can be applied to states in the search. It will effectively turn nonterminal nodes into terminal leaves. In other words, the suggestion is to alter minimax or alpha-beta in two ways; the utility function is replaced by a heuristic evaluation function, which gives an estimate of the positions utility, and the terminal test is replaced by a cutoff test that decides when to apply heuristic function.

9.12.3 Cutting Off Search

1. Cutting off search is simple approach for searching faster.
2. The cutting off search is applied to limit the depth.

   If Cutoff-Test (s, depth) then return E(S) problem in cutting off search. (Where E is evaluation function).
3. Cutoff test might be applied in adverse condition.
4. It may stop search before allowable time.
5. Iterative deepening go until time is elapsed.
6. **Quiescent position -**

   a) If a position looks "dynamic", don't even bother to evaluate it.

   b) Instead, do a small secondary search until things calm down. For example - After capturing a piece, things look good, but this would be misleading if opponent was about to capture, right back.

   c) In general such factors are called **continuation heuristics.**

   d) A **quiescent position** is a position which is unlikely to exhibit wild swings (huge changes) in value in near future.

   e) Consider following example -

Here [Refer Fig. 9.12.1] now since the tree is further explored, the values, which is passed to A, is 6. Thus the situation calms down. This is called as **waiting for quiescence.** This helps in avoiding the **horizon effect of a drastic change of values.**

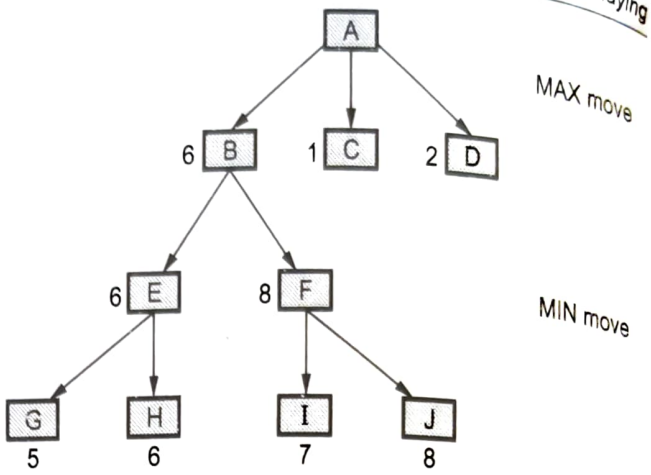

Fig. 9.12.1 Quiescent position example

7. The horizon effect -

A potential problem that arises in game tree search (of a fixed depth) is the horizon effect, which occurs when there is a drastic change in value, immediately beyond the place where the algorithm stops searching. Consider the tree show in Fig. 9.12.2 (a). It has nodes A, B, C and D. At this level since it is a maximizing ply, the value which will passed up at A is 5.

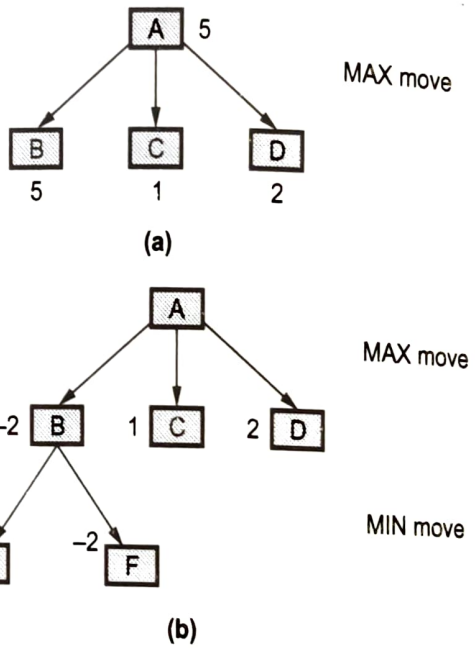

Fig. 9.12.2 Horizon effect example

Suppose node B is examined one more level as shown in Fig. 9.12.2 (b), then we see that because of a minimizing ply, value at B is – 2 and hence the value passed to A is 2. This results in a drastic change in the situation. There are **two proposed solutions** to this problem

a) Singular extension -

Which expands a move that is clearly better than all other moves. Its cost is higher with branching factor 1.

b) Secondary search -

One proposed solution is to examine the search space, beyond the apparently best one, to see that if something is looming just over the horizon. In that case we can revert to second-best move. Obviously then the second-best move has the same problem, and there isn't time to search beyond all possible acceptable moves.

9.12.4 Forward Pruning

1. It is another method for searching faster.
2. In this, some moves at a given node are pruned immediately without further consideration. Clearly, most humans playing chess only consider a few moves from each position (at least consciously).
3. Unfortunately, the approach is rather dangerous because there is no guarantee that the best move will not be pruned away.
4. This can be disasterous if applied near the root, because it may happen that, often the program will miss some "obvious" moves.
5. Forward pruning can be used safely in special siturations. For example, when two moves are symmetric or otherwise equivalent, only one of them need be considered - or for nodes that are deep in the search tree.

9.13 Games with Chance

Games with a certain element of chance are often more interesting than those without chance, and many games involve rolling dice, tossing a coin or something similar.

- In the real world many situation infront of us are unpredictable. It is also observed in many games.

  Example : Dice rolling, backgammon.

- Some time, in the game, imperfect information is available.

  Example : Cards, dominose, etc.

- In game with chance, we can introduce probabilities to our search diagrams and calculate minimax solutions as in the normal games.
- We add 1 more level in game tree i.e. the level of chance nodes.
- Chance nodes have as many successors as outcomes of the random element.
- Minimax with element of chance

  1) di (i = 1, ..., n) – Outcomes from the chance nodes.

  2) P (d_i) - Probability of d_i;

  3) S (N, d_i) - Moves from position N for outcome d_i;

  4) If N is MAX : Utility (N) = $\sum_{i=1}^{n} P(d_i) \max_{s \in S(N, di)}$ utility (s)

  5) If N is MIN :

  Utility (N) = $\sum_{i=1}^{n} P(d_i) \min_{s \in S(N, d_i)}$ utility (s)

- The utility is not computed by using just the terminal values. Therefore values assigned to win, loss and draw affect the choice of moves.
- Time complexity increases (n outcomes from the chance nodes) to $O(b^d n^d)$.
- Alpha-Beta pruning is more complicated in this game trees.

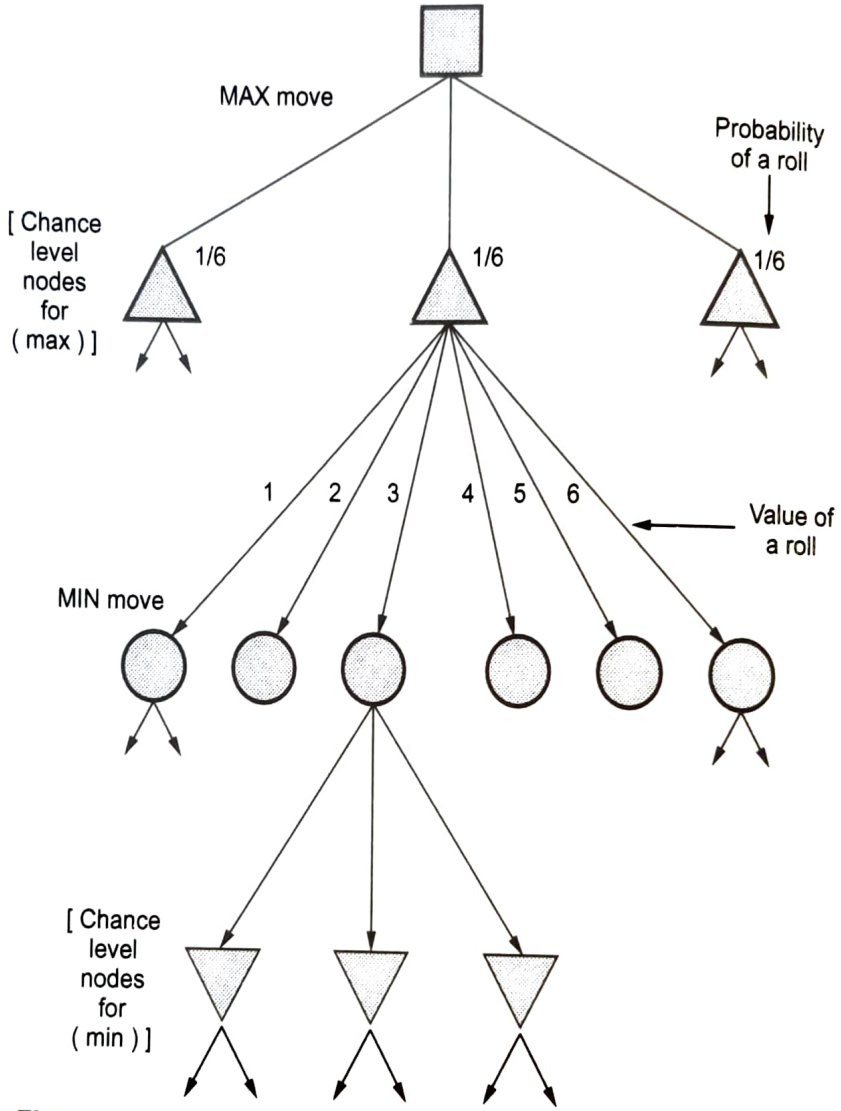

Fig. 9.13.1 A game tree for a backgammon (Game of chance)

Answer in Brief

1. *How can minimix also be extended for game of chance ?* **(Refer section 9.13)**
2. *Explain perfect decisions in game playing. Give example.* **(Refer section 9.9)**
3. *Explain minimax algorithms and how it works for game of tic-tac-toe.* **(Refer section 9.10)**
4. *Explain minimax search procedure with example upto 3rd ply.* **(Refer section 9.10)**
5. *Explain minimax procedure for game playing. Is this DFS or BFS ? How it can be modified to be used by a program playing three-four player game ?* **(Refer section 9.10)**
6. *How minimax procedure can be modified to play multipalyer game ?* **(Refer section 9.10)**

7. *Describe alpha-beta pruning using example. Show game tree upto 3rd ply.* **(Refer section 9.11)**
8. *Describe minimax procedure and alpha-beta pruning.* **(Refer section 9.11)**
9. *Explain alpha-beta pruning algorithm with example.* **(Refer section 9.11)**
10. *Write a short note on quiescent position and secondary search.* **(Refer sections 9.11 and 9.12)**
11. *Explain horizon effect.* **(Refer sections 9.11 and 9.12)**
12. *Explain minimax search algorithm for two player game.* **(Refer section 9.10)**
13. *Also explain alpha-beta pruning.* **(Refer section 9.10)**
14. *Explain the nature and scope of artificial intelligence. Why game playing problems are considered as AI problems ?* **(Refer sections 9.1 and 9.8)**
15. *Is the minimax procedure a depth-first or breadth-first search procedure ?* **(Refer section 9.10)**
16. *Why does the search in game-playing programs always proceed forward from the current position rather than backward from a goal state ?* **(Refer section 9.10)**

9.14 University Questions with Answers

Summer - 12

Q.1 *Explain the Alpha-Beta cutoff procedure in game playing.* **(Refer section 9.11)** [7]

Winter - 12

Q.2 *Explain minmax procedure for game playing with any example.* **(Refer section 9.10)** [7]

Summer - 14

Q.3 *Consider the game tree of Fig. 1 in which the static scores are from first player's point of view. Suppose the first player is maximizing player. Applying mini-max search, show the backed-up values in the tree. What move will the MAX choose ? If the nodes are expanded from left to right, what nodes would not be visited using alpha-beta pruning.* **(Refer section 9.10)** [3]

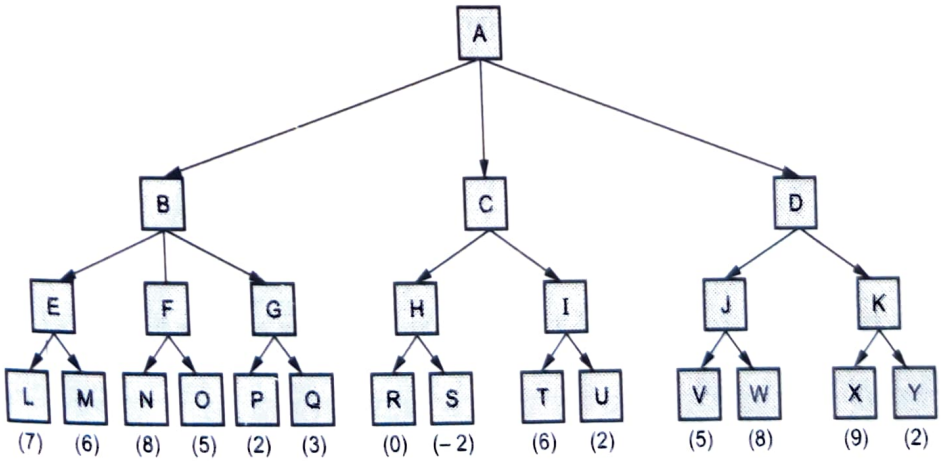

Fig. 1 Game Tree

Winter - 15

Q.4 *Explain Min-Max search procedure with an example.* **(Refer section 9.10)** [7]

Winter - 17

Q.5 *Explain alpha-beta cut off search with an example. State a case when to do alpha pruning.* **(Refer section 9.11)** [7]

Q.6 *Discuss min-max search method.* **(Refer section 9.11)** [3]

Summer - 18

Q.7 *Consider the game tree given in Fig. 2, in which the evaluation function values are shown below each leaf node for the max player. Assume that the root node corresponds to the minimizing player. Assume that the search always visits children left-to-right. Compare the backed-up values computed b the minimax algorithm by writing values at the appropriate nodes in the tree given.* **(Refer section 9.10)** [3]

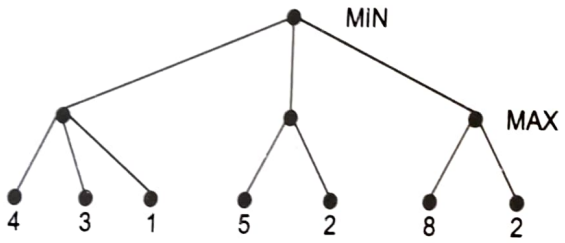

Fig. 2

Q.8 *For the game tree given in Fig. 2, which nodes will not be examined by the alpha-alpha pruning algorithm ? Show the process of alpha-beta pruning to justify your answer.* **(Refer sections 9.10 and 9.11)** [7]

Winter - 18

Q.9 *Discuss alpha-beta cutoffs procedure in game playing.* **(Refer section 9.11)** [4]

Q.10 *Explain min max procedure in game playing.* **(Refer section 9.10)** [7]

Summer - 19

Q.11 *Discuss min-max search method with an example.* **(Refer section 9.10)** [7]

Winter - 19

Q.12 *Explain the MiniMax search procedure for game playing using suitable example. What is the significance of Alpha and Beta cut-offs ?* [7]

(Refer sections 9.10 and 9.11)

Summer - 20

Q.13 *Simulate the working of Tic-tac-toe problem with Minimax technique.* [7]

(Refer section 9.10.5)

❑❑❑