Proceedings of the
5th Conference on Artificial
Intelligence
Pages 158-162

Philadelphia, PA
August 1986

An Algorithmic Solution of N-Person Games

Carol Luckhardt
Keki Irani

University of Michigan
EECS

ABSTRACT

Two-person, perfect information, constant sum games have been studied in Artificial Intelligence. This paper opens up the issue of playing n-person games and proposes a procedure for constant sum or non-constant sum games. It is proved that a procedure, max^n, locates and equilibrium point given the entire game tree. The minimax procedure for 2-persons games using look ahead finds a saddle point of approximations, while max^n finds an equilibrium point of the values of the evaluation function for n-person games using look ahead. *Max^n* is further analyzed with respect to some pruning schemes.

I *INTRODUCTION*

Game-playing is one of the first areas studied in Artificial Intelligence (AI) [Ric83]. Most of the work has been done with games that are 2-person, finite, constant sum (and therefore non-cooperative), perfect information and without a random process involved. For example, chess and checkers involve two people, have a finite number of strategies available to each player, pay the same total amount at the end of the game, each player knows the other player's moves, and there is no chance involved. The most famous game programs are the chess players such as Cray-Blitz, Chaos and Belle [Nel84]. This paper addresses n-person games, that is, games with more than two players, and describes a method of computer play for non-cooperative, non-constant sum games, and for cooperative games given a coalition structure. The approach has been to bring game theoretic results into the more pragamatic AI domain.

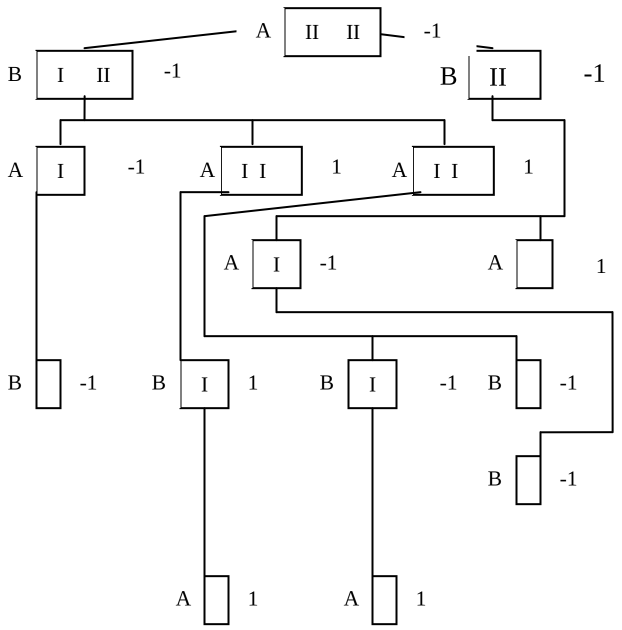
II *BACKGROUND*

Trees are often used as models of decision making in AI and in game theory. From the rules or definition of a game, the game tree representation can be specified for a n-person game by a tree where [Jon80]:

(1) the root node represents the initial state the game,
(2) a node is a state of the game with the player whose move it is attached to it,
(3) transitions represent possible moves a player can make to the next possible states,
(4) outcomes are the payoff assignments associated with each terminal node, which are n-tuples where the *i*th entry is paid to player i.

Because most games of interest have combinatorially explosive game trees, AI programs tend to analyze partial game trees in order to determine a best move. An *evaluation function* is a function which estimates what resulting value the game should have been when given a terminal node of a partial game tree. Then by the *look ahead* procedure, values are backed up from the terminal nodes to each node of the tree according to the *minimax* searching method [Ric83]:
(1) at the program's move, the node gets the maximum value of its children,
(2) at the opponent's move, the node gets the minimum value of its children.
The value that is backed up to the root node is the value of the game, and the move taken should be to a node that has that value as its backed up value. If the whole tree is available to be analyzed, there is a theorem from game theory called the *minimax theorem* [LuR57] that applies. It is for 2-person zero sum games. *Zero sum* means that the payoff values for each player add up to zero for any payoff vector. The theorem says there is a strategy that exists for each player that will guarantee that one gets at most v while the other loses at most v and the value of the game of the game is v. This set of strategies, one for each player, is called a *saddle point*.

For example, in the game of 2-2 Nim, initially there are 2 piles of 2 tokens. Players A and B alternate turns. Each player selects a pile and removes any number of tokens from that pile, taking at least one. The loser is the one that takes the last token.



The terminal node value of 1 corresponds to the vector (1,0) and -1 corresponds to (0,1). Since this is a 2-person zero sum game, the outcomes can be represented by one number. The value of 2-2 Nim is -1 which means that no matter what A does, B can always make a move that will lead to a win for B.

A technique from AI called *alpha-beta pruning* [Ric83] reduces the number of nodes that have to be visited when calculating the minimax values. For example, in the above game tree ordering, when doing a depth first search and backing up to B [I II], the left most child needs to be evaluated to get a -1 and then it is not necessary to look any further since this is the best that B can do. If a game tree has depth d and branching factor b, then in the best case of this pruning procedure, $2b^{d/2}$ nodes are evaluated rather than the complete $\boldsymbol{b^d}$ nodes [Win77].

III *N-PERSON GAMES*

Considering games with more than two players, one value will no longer suffice in representing the outcome. A vector is required for both constant and non-constant sum games. A *constant sum* game is one where the sum of the entries in an outcome vector is the same value for any terminal node. It no longer makes sense to evaluate for any terminal node. It no longer makes sense to evaluate the game based on any one player's payoff values.

Game theory solutions to non-cooperative games are usually a set of strategies for each player that are in some sense optimal, where the player can expect the best outcome given the constraints of the game and assuming the other players are attempting to maximize their own payoffs. A solution for an n-person, perfect information game is a vector which consists of a strategy for each player, $(s_1, \ldots, s_n)$. A strategy defines for the player what move to make for any possible game state for the player. Call the set of possible strategies for player i, P_i, and the payoff to player i, U_i. U_i is a real value function on a set of strategies, one for each player. The set $\{P_1, \ldots, P_n; U_1, \ldots, U_n\}$ is called the *normal form* of a game [Jon80]. An *equilibrium point* for $\{P_1, \ldots, P_n; U_1, \ldots, U_n\}$ is a strategy n-tuple $(s_1, \ldots, s_n)$, such that for all i=1, …,n and $s_i, s_i' \in P_i$, $U_i(s_1, \ldots, s_i', \ldots, s_n) \le U_i(s_1, \ldots, s_i, \ldots, s_n)$. The s_1's are called *equilibrium strategies*. For example, in the game represented by,

$$\begin{array}{ccc} & \beta_1 & \beta_2 \\ \alpha_1 & (-4,-4) & (1,-9) \\ \alpha_2 & (-9,1) & (-1,-1) \end{array}$$

Figure 2. 2X2 Game

where player A's strategies are the α_i's and B's are the β_i's , and (a,b) means pay a to the first player and b to the second player, (α_1, β_1) which corresponds to (-4,-4) is an equilibrium point. The equilibrium point has the property that no player can improve his or her own strategy if the other strategies are held fixed. A saddle point is an equilibrium point, while an equilibrium point may not be a saddle point.

These non-cooperative games with perfect information are always solvable in this sense according to the following theorem.

Theorem 1:

A finite n-person non-cooperative game which has perfect information possesses an equilibrium point in pure strategies [Jon80], page 63).

A *pure strategy* is a single α_i or β_i , as we have seen so far. The theorem just states the existence of an equilibrium point, not how to find one.

IV MAX^{N}

If we have rational players who are trying to maximize their own payoffs, the backed up values should be the maximum for each player at each player's turn. We call this procedure MAX^N . The $\max^n$ procedure, maxn(node) is recursively defined as follows:

(1) For a terminal node,
    Maxn(node) = payoff vector for node
(2) Given node is a move for player I, and
    $(v_{1,\ldots,} v_{nj})$ is
    maxn(j^{th} child of node), then
    maxn(node) = $(v_1, \ldots, v_n)$,
    which is the vector where $v_i = \max v_{ij}$

Calling the procedure with the root node finds the $\max^n$ value for the game and determines a strategy for each player, including a move for the first player. This procedure can be used with a look ahead where a terminal node in the definition above becomes a terminal node in the look ahead. For example, given the payoff vectors on the bottom row, by the procedure, A should take the first move represented here by the right child:

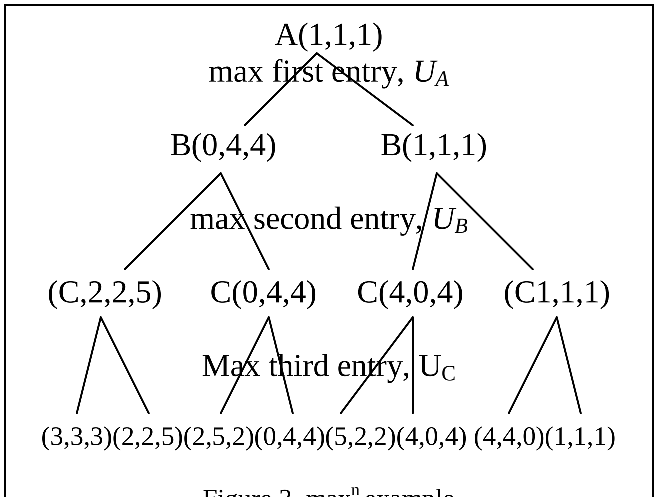


Figure 3. $\max^n$ example

Note that this procedure does not require that there be an order in the moves of the players going down the tree. There may be more than one equilibrium point. Where a tie occurs in the back up, each possible choice will lead to an equilibrium point, so it does not matter which move is selected.

Theorem 2:

Given a n-person, non-cooperative, perfect information game $\{P_1, ..., P_{n},; U_1, \ldots, U_n\}$, in tree form, $\max^n$ finds an equilibrium point for the game.

Proof:

Backing up values in the tree by applying the $\max^n$ procedure, with some tie-breaker, determines a strategy for each player which gives a strategy set $S=(s_{1, \ldots,} s_n)$, $s_i \in P_{i,}$ i=1,…,n. So, at each node for each player I, the strategy s_i gives the arc or move choice which maximizes the backed up value of U_i of the children nodes. In order to have an equilibrium point, we need to show that for all i,