Proceedings of AAAI Symposium on Computer Game Playing, AAAI Spring Symposium, Stanford University, March 1988

A COOPERATIVE N-PERSON GAME SOLUTION

Carol A. Luckhardt

Southwest Research Institute
San Antonio, TX 78284

ABSRTRACT

Game playing in Artificial Intelligence (AI) has resulted in effective algorithms for playing two-person, zero sum, perfect information, non-cooperative games. Game theory has been too theoretical to apply directly to the play of games with more than two players. In this presentation, a function for representing a game is suggested, the stability of a coalition is defined, and an algorithm is developed for play of an n-person, perfect information, cooperative game by a computer. This work is based on the max^n evaluation for n-person, perfect information, non-cooperative games [LuI86].

INTRODUCTION

From the rules or definition of a game, the game tree representation can be specified for an n-person game be a tree where [Jon80]:

(1) The root node represents the initial state of the game;
(2) A node is a state of the game with the player whose move is attached to it;
(3) Transitions represent possible moves a player can make to the next possible states; and
(4) Outcomes are the payoff assignments associated with each terminal node, which and 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 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; and
(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.

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 play (s_1. . . . . s_n). A strategy defines for the player what move to make for any possible game states for the player. Call the set of possible categories for player i, P_i, and the payoff to player i, U_i. U_i is a real valued function on a set of strategies, one for each player. The set {P_1. . . . . .P_n; U_1. . . . . U_n} is called the *normal form* of a game [Jon80].

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_{1j}, . . . , v_{nj}) is maxn(j^{th} child of node), then maxn(node) = (v_1, . . . v_n), which is the vector where $v_i = \max_j v_{ij}$.

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. Fro example, given the payoff vectors on the bottom row, by the procedure, A should take the move represented in Figure 1 by the left child:

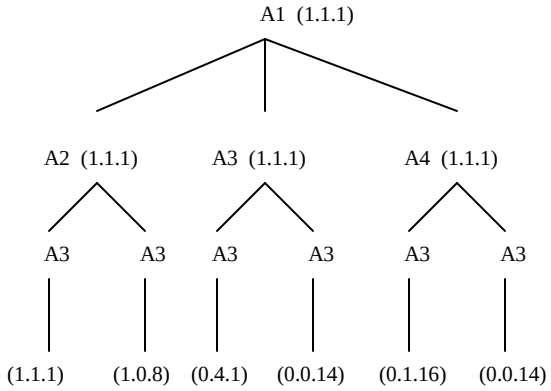

Figure 1. Max^n example

Given a non-cooperative n-person game in the tree for, the vector values associated with each terminal node represent the payoff to each of the n players if that node is the result of playing the game. For a node or game situation, the max^n procedure determines a strategy for each player, a move for the current player to use, and an equilibrium point for the game for the current node [LuI86]. The procedure backs up from the terminal nodes the vectors that give the player pf the parent node the most payoff. This is repeated up the tree until the root node has a vector value associated with it and a child to choose for the move to make from the root node.

A COOPERATIVE GAME FUNCTION

A cooperative game is one in which communication and coalition formation is allowed between players. A *coalition* is a subset of the n players such that a binding agreement exists between the players. The coalition can be treated as one player with a strategy, which is collectively determined. When is a player's turn who is in the coalition, it is the coalition's move. A *coalition structure* on an n-person game {P_1, . . . , P_n; U_1, . . . , U_n} is a partition of {1, . . . , n}.

For a cooperative game, a set of players, which is a coalition if they cooperating with each other, can be treated as one player in the max^n procedure by summoning the payoffs of the players. Max^n can be applied to the structure such that each coalition represents a player. Applying max^n to all possible coalition structures, the minimum payoff in the resulting payoff vectors for all structures containing that coalition. This includes a minimum payoff for each individual player. By subtracting the sum of the individual minimum payoffs of the players that make up the coalition from the minimum payoff for a coalition, the profit for the coalition is calculated. This profit is notated by m© for coalition c. These concepts are defined formally below.

Definition 1:

Let M(c|S) be the payoff to coalition C ϵ S when the max^n procedure is applied to a game with coalition structure S.

$M(c \mid S) = \sum U_i$ (S1……,Sn) where the S_i are the strategies determined by max^n.

Definition 2:

The minimum expected payoff (mep) for a coalition is given by

Mep(c) = min M(c|S) where c ϵ S.

Definition 3:

For any coalition c, let

$M(c) = mep(c) - \sum mep(i)$.

An interesting property of this function, which will be proved next, is that for disjoint coalitions a and b, m(ab) ≥ m(a) + m(b), where ab is the union of a and b [also, if a function f(x) satisfies this inequality, then there exists a game such that f(c) = m(c) for every coalition, which implies that the inequality is the only restriction on m(c)]. A game can be represented by this m-function.

Lemma 1:*

M[ab|(ab)US] >= M[a|(a)(b)US] + M[b|(a)(b)US]

where S is a substructure containing all other players besides those in coalitions a and b.

Proof:

Let a consist of players ($a_1, \ldots, a_i$) and b consists of players ($b_1, \ldots, b_j$). Max^n on (a)(b)∪S determines strategies α, β r for a, b, and S respectively. Max^n on (ab)US finds the best that a and b can do together , that is, for $\delta \in P_a \times P_b$ where $P_a = P_{a1} \times \ldots \times P_{ai}$ and $P_b \times \ldots \times P_{bj}$.

$M[ab|(ab)US] = \max [U_a (\delta, r) + U_b(\delta, r)]$

$\geq U_a(\alpha, \beta r) + U_b(\alpha, \beta r)$

$\geq M[ab|(ab)US] + M[b|(a)(b)US]$

where $U_a(v) = U_{a1}(v)b + \ldots\ldots + U_{ai}(v)b$ and $U_b(v) = U_{b1}(v) + \ldots + U_{bj}(v)$ for a vector v which consists of strategies for all the players.

Theorem 1:

For disjoint coalitions a and b,

$$mep(ab) \geq mep(a) + mep(b).$$

Proof:

For coalition structure S where $ab \in S$, $mep(ab) = \min_s M(ab|S) = M(ab|S')$ for some structure $S' = \{ab\} \cup T$. Assume that the theorem is false. Then $mep(ab) < mep(a) + mep(b)$. Since mep(a) and mep(b) are minimums, letting $R = \{a\}\{b\} \cup T$, the structure with a and b playing separately, then

$$M(a|R) \geq mep(a) \text{ and } M(b|R) \geq mep(b).$$

So, $mep(ab) < M(a|R) + M(b|R)$ and $M(ab|S') < M(a|R) + M(b|R)$. But $S' = \{ab\} \cup T$ and $R = \{a\}\{b\} \cup T$, which contradicts Lemma 1. This proves the theorem.

From this theorem we know that for all coalition structures S. This means that the grand coalition achieves the most overall payoff, and there may be other structures that also achieve this maximum sum of payoffs. We also can deduce that is $mep(1, \ldots, n) =$ mep(i), then mep(ab) = mep(a) + mep(b) for all coalitions a and b where a . By subtracting indiviual mep(i) values for i in a and b from the inequality in Theorem 1, $m(ab) \geq m(a) + m(b)$. Also, $m(1, \ldots n) \geq \sum m(c)$ for any structure S.

EARNINGS AND STABILITY

Given a game in m-function form, a division of the profits to each player is defined based on increasing coalition size and is called the earnings for each player. We use Definition 4 in a recursive definition for E(i,c) to simplify the notation.

Definition 4:

For player i and coalition c, let f(i,c) = E(i,c), for all coalitions c' where $i \in c'$.

Definition 5:

The earnings for player i in coalition c are given recursively as,

$$E(i.c) = \begin{cases} \text{If } |c| = 1. \quad 0 \\ \text{if } |c| > 1. f(I,c) + m(c) - \dfrac{\sum f(j,c)}{|c|} \end{cases}$$

for the example game tree.

Coalition	mep(c)	m(c)	E(I . c)
a	1	0	(0)
b	1	0	(0)
c	1	0	(0)
ab	4	2	(1,1)
ac	6	4	(2,2)
bc	8	6	(3,3)
abc	14	11	(3,4,4)

If, for each player in the coalition, the player's earnings are the maximum earnings over all coalitions which contain that player, the coalition is defined to be *stable*. A structure is stable if each coalition in the structure is stable. In the example, the coalition and structure (a,b,c) is stable. Some properties of these functions are: (1) if a structure is stable for a game, then any coarser structure is also stable; (2) every game has at least one stable coalition; and (3) if a coalition is stable, there is no coalition of equal or larger size which is a threat to the stability of the coalition. These properties are discussed in greater detail below.

In order to analyze the process of calculating earnings, the following definitions will be useful.

Definition 6:

The level of a coalition, c, is equal to the cardinality of the coalition, |c| =

The total profit P(i) can be thought of as being accumulater at various levels of coalitions. From one level l to the next, l + 1, either a player's profit is increased by the maximum equally distributed coalition profit or it is not increased at all, but it does not decrease. This monotone non-decreasing function is defined next.

Definition 7:

The profit at level l for player a is defined recursively as

$$P_1(a) = \max \left\{ P_1\text{- }1(a), \max E[a.c_l(a)] \right\}$$

Where $P_1(a) = 0$.

Before a solution algorithm is defined, some concepts are proven in order to support a procedure based on stability.

Definition 8:

A coalition c_m is c –stable if, for all players $i \in c_m$. P (i) = $E(i,c_m)$ where $m \leq$.

Theorem 2:

For any n-person game, for each level = 1…..n, there exists a coalition that is c –stable.

Proof:

For k=1 this is trivially true, since $P_1(i)=0$ for all players. Assume it is true for k, so there is a coalition c that is c_k-stable. We need to show that there is a coalition c_{k+1}-stable. We know the $E(i,c) = P_k(i)$ for $i \in c$.

Case 1: $m(c_{k+1}) - \sum P_k(i) \leq 0$ for all c_{k+1} . This difference is the profit at level k + 1 that is distributed to $i \in c_{k+1}$, so c must be c_{k+1}-stable.

Case 2: There exists a c´ at the k + 1 level such that $m(c´) - \sum P_k(i) > 0$ and is maximum over all c_{k+1}. So, each player $i \in c´$ receives a profit above $P_k(i)$ that is more than in any other c_{k+1}. Then each of the players $i \in c´$ gets $P_{k+1}(i)$ and c´ is c_{k+1}-stable.

Corollary 1:

For any n-person game there is at least one stable coalition.

Proof:

Take Theorem 2 with = n.

Theorem 3 states that for a stable coalition t and any coalition u where u has at least as many players as t, t u = v and v is not empty, then the players in u cannot offer to the players in v more than they can get in t without going below what the players in u – v could get in a smaller coalition. Therefore, there is no coalition of equal or greater size than that of a stable coalition, which is a threat to the stability.

We use the following conventions

Definition 9:

For coalitions c´ and c where , let the earnings for coalition c´ be E(c´,c) = E(i,c) and the profit at level for coalition c be

Theorem 3:

It t is stable, then for

Proof:

Case 1: u > t

The inequality can be rewritten as

$E(v,t) + E(t - v,t) - P(t - v) \geq E(v,u) + E(u - v,u) - P_{u-1}(u - v)$.

Since t is stable,

$E(v,t) = P(i) \geq E(v,u)$.

Therefore, what needs to be shown is

$E(t - v,t) - P_{u-1}(t - v) \geq E(u - v,u) - P_{u-1}(t-v)$.

Since t is stable and P is increasing, $E(t - v,t) = P_n(t - v) \geq P_{u-1}(t - v)$. Therefore, $E(t - v.t) - P_{t!-1}(t - v) \geq 0$. If $E(u - v.u) - P_{u!-1}(u - v) \leq 0$, then the inequality is true. By definition.

$$E(u - v.u) = P_{u!-1}(u - v) + |u - v| \cdot \frac{m(u) - P_{u!-1}(u)}{|u|}$$

If $A = \frac{m(u) - P_{u!-1}(u)}{|u|} \leq 0$, then $E(u - v.u) \leq P_{u!-1}(u - v)$. if $A > 0$,

then $E(v,u) = P_{u!-1}(v) + |v|. A$

But $E(v,t) = P_n(v) = P_v = P_{u!-1}(v)$ since t is stable and $|t| \leq |u| - 1$, so that $E(v,t) \leq E(v,u)$ which is a contradiction since t is stable. So, $A \leq 0$ and $E(u - v.u) - P_{u!-1}(u - v) \leq 0$ and the theorem is true for this case.

Case 2 : $|u| = |t| = k$.

For $i \in v$, since t is stable, $E(i, t) \geq E(i, u)$. By the definition of earnings and substituting k and k -1 where possible.

$$P_{k-1}(i) + \frac{m(t) - P_{k-1}(t)}{k} \geq P_{k-1}(i) + \frac{m(u) - P_{k-1}(u)}{k}$$

Therefore, $m(t) - P_{k-1}(t) \geq m(u) - P_{k-1}(u)$. Adding $P_{k-1}(v)$ to both sides gives $m(t) - [P_{k-1}(t) - P_{k-1}(v)] \geq m(u) - [P_{k-1}(u) - P_{k-1}(v)]$.

Since $P_l(a-b) = P_l(a) - P_l(b)$ from the definition of P_l.

$m(t) - P_{k-1}(t - v) \geq m(u) - P_{k-1}(u - v)$.

Therefore $m(t) - P_{t-1}(t - v) \geq m(u) - P_{u-1}(u - v)$.

From this sense of stability a solution algorithm is now defined for cooperative n-person games. The algorithm begins with a game in m-function form and produces a coalition structure and payoff vector.

Solution Algorithm:

(1) If a stable structure exists, then any minimum solution structure and the corresponding payoff vector comprise the solution.

(2) If there is not a stable structure, then take all stable coalitions and form a substructure of these players as follows. If a subset of the stable coalitions is a partition of another stable coalition, then delete the largest coalition from the stable coalition set for the rest of the calculation.

   a. If a stable coalition c has no players in common with the other stable coalitions, then the substructure contains c with its respective payoffs.

   b. For each set of coalitions $\{c_1, \ldots, c_m\}$ that has any players in common, the union of these players is a new coalition c in the substructure with the folloing pay:

   Let $c^1 = \bigcap_{j=1}^{m} c_j$ and $c^2 = \bigcup_{j=1}^{m} c_j - c^1$

   For $i \in c^1$, i receives P(i) = max E(i, c), otherwise a player i receives

$$\left[m(c) - \sum_{j \in c1} P(j) \right] \cdot \frac{r_i}{\sum_{j \in c1} r(j)}$$

Where $r_k = P(k) \cdot \frac{v_k}{m}$ where v_k is the number of stable conditions in which k is contained. So $1 \leq v_k < m$.

(3) If all players are not yet acconted for, apply the above two steps to the subgame with the remaining players, calculation new payoff values. Repeat this until all players have payoffs.

SOLUTION ALGORITHM

(4) If the resulting combined substructures do not have a sum of payoffs equal to m(1, ….., n) then let the final solution structure be {1,….,n} with additional payoff evenly distributed among the players. Otherwise, the solution is the combined substructures and associated payoffs defined above.

For the example, the solution algorithm gives {(a, b, c); (2,4,4)} since (a,b,c) is stable.

Coalition	m(c)	E(I,c)
a	0	(0)
b	0	(0)
c	0	(0)
d	0	(0)
ab	3	(1.5,1.5)
ac	4	(2,2)
ad	5	(2.5, 2.5)
bc	6	(3,3)
bd	7	(3.5, 3.5)
cd	8	(4,4)
abc	10	(2.5, 3.5, 4)
abd	7	(1.5, 2.5, 3)
acd	9	(2, 3.5, 3.5)
bcd	8.5	(2.5, 3, 3)
abcd	12	(2,3, 3.5, 3.5)

(cd) and (abc) are the stable coalitions. The solution is the union of these two coalitions with payoffs calculated in 2b. {(abcd); (2.2.8, 4, 3.2)}.

CONCLUSIONS

The solution algorithm's largest asset is that it works and can be applied to the actual play of a game. This is not true of most other game theoretic solutions. This solution is supported by the properties of stability and by comparisons to game theoretic concepts. In its favor is that the solution from the algorithm is contained in many of the game theoretic solution sets such as the core and stable set, and sometimes in the bargaining set and kernel (see [LuR57] for definitions). The solution algorithm is definite and gives a precise solution; it is not necessary to search through a large solution space.

The solution algorithm is based on the assumption that each player will agree to an equal distribution of profit for the coalition. The algorithm can be directly applied to a game or a game with an evaluation function applied at any level of the game tree. The solution algorithm can be taken advantage of by a singular player, such as a computer, to decide on individual play and to make recommendations to other players. In calculating the solution, an overall or global point of view is adopted, with each player achieving his or her best possible result so that everyone might win.

The results in this work have implications for and can be applied to areas in AI, mathematics, economics, social psychology, and conflict resolution.

ACKNOWLEDGEMENTS

I thank Prof. Keki Irani for his guidance and suggestions with regard to this work.

REFERENCES

[Jon80] Jones, A.J. , Game Theory: Mathematical Models of Conflict, Ellis Horwood, west Sussex, England, 1980

[LuI86] Luckhardt, C.A. and Irani, K.B. "An Algorithmic Solution of N-Person Games" National Conference on Artificial Intelligence, Philadelphia, PA, August 1986.

[LuR57] Luce, R. and Raiffa, H., Games and Decisions, John Wiley & Sons, New York, 1957.

[RiC83] Rich, Elaine, Artificial Intelligence, McGraw Hill, US, 1983.

Proceedings of AAAI Symposium on Computer Game Playing, AAAI Spring Symposium, Stanford University, March 1988