Artificial Intelligence 82 (1996) 237-257

Artificial Intelligence

Searching game trees under a partial order [*]

Pallab Dasgupta, P.P. Chakrabarti [*], S.C. DeSarkar

Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India 721302

Received January 1994; revised November 1994

Abstract

The problem of partial order game tree search arises from game playing situations where multiple, conflicting and non-commensurate criteria dictate the merit of a position of the game. In partial order game trees, the outcomes evaluated at the tip nodes are vectors, where each dimension of the vector represents a distinct criterion of merit. This leads to an interesting variant of the game tree searching problem where corresponding to every game playing strategy of a player, several outcomes are possible depending on the individual priorities of the opponent. In this paper, we identify the necessary and sufficient conditions for a *set* of outcomes to be inferior to another set of outcomes for *every* strategy. Using an algebra called *Dominance Algebra* on sets of outcomes, we describe a bottom-up approach to find the non-inferior sets of outcomes at the root node. We also identify shallow and deep pruning conditions for partial order game trees and present a partial order search algorithm on lines similar to the α-β pruning algorithm for conventional game trees.

1. Introduction

Current game tree searching methods assume that a given position of the game can be evaluated as a single numerical value which indicates the merit of that position. In normal two-player games, a MIN-MAX value [4] is defined, which indicates the best alternative available to a player. Depth-first algorithms like *α-β pruning* [1] and best-first algorithms like *SSS** [6] are known to efficiently determine this MIN-MAX value. These studies have also been extended to multiplayer games [2].

In this paper, we study a variant of the game tree search problem, where the costs evaluated at the tip nodes of the game tree are vectors. The objective of such a framework

[*] Originally submitted as Research Note.

[*] Corresponding author. This author acknowledges the support of the Indian National Science Academy for partial support of this work.

is to model game playing situations where the merit of a position of the game is dictated by multiple non-commensurate criteria. In such game playing situations, a position of the game can be better than another in some criteria of merit and worse in some of the other criteria. The individual priorities of each player decides which position would be most desirable for it.

It is well known that the presence of an adversary can lead to game playing situations in optimization problems. Such an adversary may appear as an actual competitor (as in competitive markets) or as random factors that emerge during problem solving and affect the final outcome. For example, the problem of searching for a maximum capacity path in a flow tree has been modeled as a game tree search problem [4] when the edge capacities are random. Papadimitriou and Yannakakis [3] have described an extension of the Traveling Salesperson Problem (TSP) called Canadian TSP that can be modeled as a game playing problem.

Optimization problems involving multiple, conflicting and non-commensurate objectives have motivated a new search model introduced by Stewart and White [5]. In this model, the cost structure is vector-valued, where each dimension of the cost vector represents a distinct criterion to be optimized. A partial order called *dominance* is used to eliminate clearly inferior solutions and obtain the set of non-inferior solutions in the search space. In this paper, we extend the same framework for modeling game playing situations involving multiple non-commensurate criteria.

Multiobjective game playing situations may be modeled by game trees where the tip nodes have vector-valued costs. Each dimension of the cost vector represents a distinct criterion of merit. The partial order game tree search problem is to find the non-inferior options of a player by using the following partial order.

Definition 1.1 (*Dominance*). Let $y^1 \equiv \langle y_1^1, y_2^1, \ldots, y_K^1 \rangle$ and $y^2 \equiv \langle y_1^2, y_2^2, \ldots, y_K^2 \rangle$ be two K-dimensional vectors. Then y^1 dominates y^2 iff:

$$y_i^1 \geqslant y_i^2 \qquad \forall i \quad 1 \leqslant i \leqslant K \quad \text{and} \quad y^1 \neq y^2.$$

A vector $y \in Y$ is said to be "non-dominated" in Y if there does not exist another vector $y' \in Y$ such that y' dominates y.

The definition of dominance is similar to that used by Stewart and White [5] to define the partial order for searching multiobjective OR-graphs.

In conventional game trees, there exists a total order on the values (outcomes) evaluated at the tip nodes. Given a set of outcomes at a MAX-node, player-1 chooses the maximum one. Likewise, the minimum outcome is chosen by player-2 at MIN-nodes. In partial order game trees, we can eliminate the clearly inferior outcomes using the *dominance* relation, but the choice of the desired outcome from the remaining non-dominated set of outcomes will depend on the individual preferences of the player. We call the preferences of a player combined with the resulting selection procedure the *strategy* of the player.

It is easy to see that if the individual strategies of both players are known, then by applying the strategy of player-1 at MAX-nodes and the strategy of its opponent (player-2) at the MIN-nodes, we can solve the problem using conventional game tree search strategies. If the individual strategies of the players are not known, then for each strategy of player-1, several outcomes are possible depending on the strategy adopted by the opponent. Corresponding to each move from a node, a set of outcomes is possible for every strategy of the player. The objective of the partial order game tree search problem is to find the *non-inferior* sets of outcomes through each move from the root node of the game tree. Player-1 can then apply its own strategy on these sets of outcomes and select the move that returns the best set of outcomes based on its strategy. We shall show that even if the strategy of either player is known, it cannot be used in the interior nodes of the game tree, and hence it is necessary to find every non-inferior set of outcomes at the root node.

In this paper, we analyze the partial order game tree search problem and identify the necessary and sufficient conditions for a set of outcomes to be inferior to another set of outcomes for *every* strategy. For convenience of representation, we use an algebra called *Dominance Algebra* on the sets of outcomes to describe the non-inferior sets of outcomes and a bottom-up approach to determine the non-inferior sets of outcomes at the nodes of the game tree. Finally, we identify shallow and deep pruning conditions for partial order game trees and on the basis of those pruning conditions, we present a partial order search algorithm on lines similar to the α-β pruning algorithm for conventional game trees.

The paper is organized as follows. In Section 2 we describe the partial order game tree search problem. The use of *Dominance Algebra* is described in Section 3. In Section 4 we describe a brute force method to determine the non-inferior sets of outcomes for a player. The shallow and deep pruning conditions, as well as the search algorithm which uses the pruning conditions are described in Section 5.

2. The partial order game tree search problem

In the conventional game tree search problem, the values at the tip nodes of the game tree are members of a totally ordered set. Given the MIN-MAX values [4] of the children of a node, it is possible to determine the MIN-MAX value of the parent simply by using the total order to decide which value is the best. Using a bottom-up approach it is possible to determine the MIN-MAX value corresponding to each move at the root node of the game tree.

In the partial order game tree search problem, we only have a partial order on the the vector-valued outcomes. In the cases where the partial order can determine the better outcome, the choice is obvious. For example, in Fig. 1(a) it is obvious that player-1 at the MAX-node P will select the move with outcome $(11,5)$ since $(11,5)$ dominates both $(9,4)$ and $(7,3)$. Likewise, in Fig. 1(b) player-2 will select the move with outcome $(7,3)$ from node Q. In such situations, it is possible to decide the best move without any knowledge of the individual preferences of the players. On the other hand, consider the situation in Fig. 2(a). At node Q, player-2 has a choice between the outcomes $(11,5)$ and $(5,7)$. Since none dominates the other, the decision will be based on the preferences of player-2.

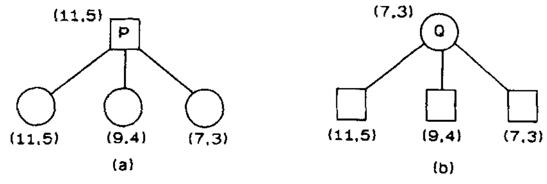

Fig. 1. Cases illustrating simple dominance.

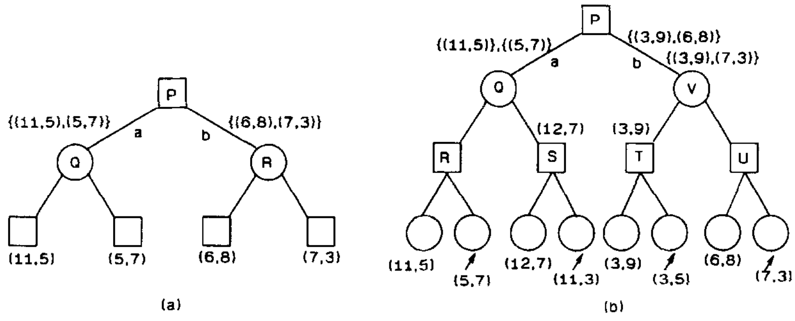

Fig. 2. Examples showing multiple sets of outcomes.

At node Q of Fig. 2(a), if the preferences of player-2 are not known, then it is not possible for player-1 to decide which outcome will be selected. Therefore, corresponding to move a at node P, we have a set of possible outcomes $S_a = \{(11,5),(5,7)\}$. In a similar manner, corresponding to move b at node P, we have a set of outcomes $S_b = \{(6,8),(7,3)\}$. Thus, at node P, player-1 has to choose between the *sets of outcomes* S_a and S_b, and accordingly take either move a or move b. This choice will depend on the preferences of player-1. The question which arises at this point is: *how does a player use its preferences to choose between such sets of outcomes?*

Thus the problem of the player is to choose between two sets of outcomes (such as S_a and S_b) with the knowledge that the preferences of the opponent (which are not known to this player) will decide the final outcome from the selected set. Let $\vec{x}$ denote the worst outcome in a given set based on the preferences of the player. Then the best that the given set can *guarantee* for the player is $\vec{x}$. This is similar to the familiar notion of MIN in conventional game tree search. Therefore, to compare two sets of outcomes based on individual preferences, we compare the worst outcome from each set based on those preferences. In case of a tie, we compare the other outcomes. The complete procedure is as follows:

```
**Compare**($S_1, S_2, \phi$)
To compare sets of outcomes $S_1$ and $S_2$ on the basis of preferences $\phi$
  1. If only $S_1$ is empty, then declare $S_2$ as better.
     Likewise, if only $S_2$ is empty, then declare $S_1$ as better.
     If both $S_1$ and $S_2$ are empty then
        select $S_1$ or $S_2$ randomly and declare it to be better.
```

```
2. Let $\vec{x}_1$ be the worst outcome in $S_1$ and
     $\vec{x}_2$ be the worst outcome in $S_2$ based on $\phi$.
3. If $\vec{x}_1$ and $\vec{x}_2$ are of equal preference then
     3.1 Drop all outcomes from $S_1$ and $S_2$ that are of
          equal preference to $\vec{x}_1$
     3.2 Goto [Step 1]
4. If $\vec{x}_1$ is better than $\vec{x}_2$ based on $\phi$, then declare $S_1$ as better
     else declare $S_2$ as better.
```

As an example, if the preferences of player-1 are such that $(5,7)$ is preferred over $(11,5)$ and $(11,5)$ is preferred over $(13,3)$, then the set $\{(13,3),(5,7)\}$ is preferred over the set $\{(13,3),(11,5)\}$. Also the set $\{(13,3),(5,7)\}$ is preferred over the set $\{(13,3)\}$, since depending on the preferences of the opponent there is a possibility of reaching the better outcome $(5,7)$ from the former set.

It should also be noted that two outcomes may have equal preference. Thus, the preferences of a player do not exactly induce a total order on the set of vector-valued outcomes; rather, they induce a many-to-one mapping from the set of outcomes to a totally ordered set which preserves the partial order imposed by the dominance relation.

It is easy to see that the way in which a player will behave can be described by its individual preferences and a procedure for comparing sets of outcomes. Throughout this paper we assume that the players use the procedure *Compare* to compare the sets of outcomes. We define the strategy of a player as follows.

Definition 2.1 (*Strategy*). The strategy of a player is a selection mechanism based on the procedure *Compare* and the individual preferences of the player which is consistent with the partial order imposed by the dominance relation. Thus if $\vec{x}$ is an outcome, which dominates an outcome $\vec{y}$, then for all strategies of player-1, $\vec{x}$ is better than $\vec{y}$ and for all strategies of player-2, $\vec{y}$ is better than $\vec{x}$.

Initially we analyze the situation that arises when the strategy of neither player is known. Later we shall show that even if the strategy of either of the players is known, the search problem remains the same.

In Fig. 2(a), corresponding to each move at node P we had only one set of outcomes. The following example illustrates that corresponding to a move, it is possible to have multiple sets of outcomes.

Example 2.2. Consider the game tree in Fig. 2(b). At node T player-1 will obviously select the outcome $\{(3,9)\}$. At node U player-1 can choose either $(6,8)$ or $(7,3)$ depending on its strategy. This strategy is not known to player-2; therefore, at node V, it has to choose between the sets $\{(6,8),(7,3)\}$ and $\{(3,9)\}$. Player-2 may choose either set depending on its own strategy.

Now suppose the strategy of player-1 is such that it prefers $(6,8)$ over $(7,3)$. If the game reaches node U, then it will select $(6,8)$. Therefore, corresponding to this strategy of player-1, move b (at node P) presents the set of outcomes $\{(3,9),(6,8)\}$ such that the opponent's strategy decides whether $(3,9)$ or $(6,8)$ will be reached. On the other hand, if player-1 prefers $(7,3)$ over $(6,8)$, then corresponding to that strategy, move

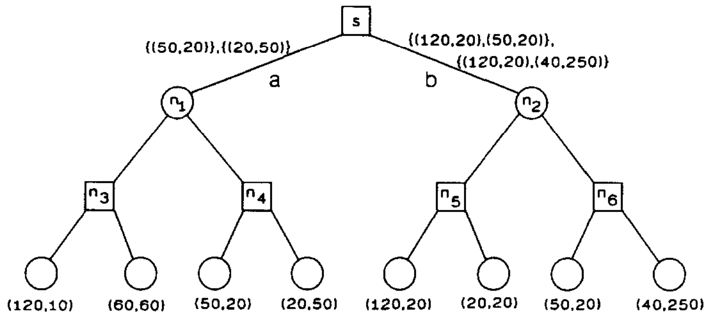

Fig. 3. Game tree illustrating that a strategy should not be used at interior nodes.

b presents the set of outcomes $\{(3,9),(7,3)\}$. Thus, if move b is selected at node P, then the outcome will either belong to the set $\{(3,9),(6,8)\}$ or the set $\{(3,9),(7,3)\}$ depending on the strategy adopted by player-1.

In a similar situation at node Q, player-2 has to choose between the sets $\{(12,7)\}$ and $\{(11,5),(5,7)\}$. In this case, since both $(11,5)$ and $(5,7)$ are dominated by $(12,7)$, player-2 will always select the move to node R. Therefore, corresponding to move a, we have two singleton sets, namely $\{(11,5)\}$ and $\{(5,7)\}$.

At node P of Fig. 2(b), we can provide player-1 with four sets of outcomes to choose from, namely $\{(11,5)\}$, $\{(5,7)\}$, $\{(3,9),(6,8)\}$ and $\{(3,9),(7,3)\}$. Are all these sets candidates for selection? Let us compare the sets $\{(11,5)\}$ and $\{(3,9),(7,3)\}$. Since $(11,5)$ dominates $(7,3)$, it is easy to see (through the procedure *Compare*) that there can be no strategy for player-1 that prefers the set $\{(3,9),(7,3)\}$. This set is therefore an inferior set of outcomes.

Example 2.2 shows that even if the individual strategies of the players are not known, certain sets of outcomes can be clearly discarded. The objective of partial order game tree search is to discard such sets of outcomes and provide the player at the root node with the non-inferior sets of outcomes.

Before giving the formal definition of the problem, it is necessary to make one further observation. Suppose that each player knows its own strategy but not that of its opponent. In the game tree representation, this means that only the strategy of player-1 is known. Can we use the strategy at the interior MAX-nodes of the game tree to prune away several sets of outcomes? The following example shows that the answer is negative in general.

Example 2.3. Consider the game tree in Fig. 3. Let the strategy of player-1 be such that given any two outcomes, the outcome which is greater in the first dimension is preferred. If two outcomes are equal in the first dimension, then the outcome with the larger second dimension is preferred. Let us first analyze the problem using the strategy of player-1 at the interior nodes. At node n_3 player-1 will choose the outcome $(120,10)$, and at node n_4 it will select $(50,20)$ based on its strategy. At node n_1, it is not known whether player-2 will select $(120,10)$ or $(50,20)$, and so corresponding to move a

at node P, we have the set of outcomes $\{(120,10),(50,20)\}$. Likewise, player-1 will select $(120,20)$ at node n_5 and $(50,20)$ at node n_6. At node n_2, player-2 will definitely select $(50,20)$ since it is dominated by $(120,20)$. Thus corresponding to move b we have the set $\{(50,20)\}$. If we compare the sets corresponding to move a and b on the basis of the strategy of player-1, we find that the set $\{(120,10),(50,20)\}$ is preferred over the set $\{(50,20)\}$, and so, move a appears to be better.

In this analysis, we have overlooked one vital point, that is, the opponent does not know the strategy of player-1. Therefore, from the point of view of player-2, player-1 can select either of the outcomes at nodes n_3 and n_4. Therefore, at node n_1, player-2 has to choose between the sets $\{(120,10),(60,60)\}$ and $\{(50,20),(20,50)\}$. Since both $(50,20)$ and $(20,50)$ are better than $(60,60)$ for every strategy of player-2, therefore player-2 will always select the set $\{(50,20),(20,50)\}$ and take the move to node n_4. Thus, corresponding to move a, we have two sets of outcomes, namely $\{(50,20)\}$ and $\{(20,50)\}$. For the given strategy of player-1, the set $\{(50,20)\}$ is preferred.

Using a similar reasoning, at node n_2, player-2 will have to select between the sets $\{(120,20)\}$ and $\{(50,20),(40,250)\}$. Player-2 can select either set depending on its strategy, therefore corresponding to move b, we have two sets of outcomes, namely $\{(120,20),(50,20)\}$ and $\{(120,20),(40,250)\}$. For the given strategy of player-1, the set $\{(120,20),(50,20)\}$ is preferred. By comparing this set with the set $\{(50,20)\}$ (which was preferred through move a) we find that actually move b is better for player-1.

The above example shows that unless the strategies of both players are known, we cannot apply the individual strategy of either player at an internal node while determining the sets of outcomes. Therefore, we can eliminate only those sets of outcomes that are inferior with respect to every strategy. In other words, we have to find the non-inferior *packets* of outcomes, where a *packet* of outcomes is defined as follows.

Definition 2.4 (*Packet of outcomes*). At a node n a set of outcomes P corresponding to a move a will be called a *packet* for a player iff it has the following two properties:

(1) There exists a strategy of the player, such that after taking move a, irrespective of the strategy adopted by the opponent, the final outcome will be better than or equal to an outcome in that set in every dimension.

(2) For every outcome $\vec{x}$ in P, there exists an opponent strategy such that if move a is taken, then the final outcome is $\vec{x}$.

From the previous discussion it follows that there may be several packets corresponding to a single move. The set of packets at a node is the union of the set of packets corresponding to every move at that node.

The second property of a *packet* ensures that redundant outcomes (that is, outcomes which will never be reached) are not included in a *packet*. For example in Fig. 3, the set of outcomes $\{(120,10),(50,20)\}$ corresponding to move a at node s satisfies the first property, but is not a packet for player-1 since the opponent at node n_1 will always select the move to n_4 (see Example 2.3), and therefore there is no opponent strategy to reach the outcome $(120,10)$.

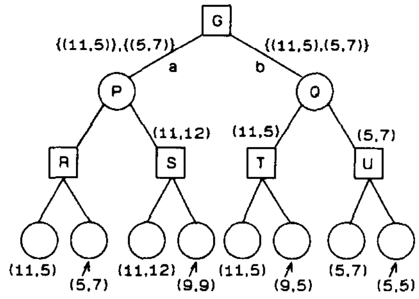

Fig. 4. Dominance by freedom of choice.

If we can find the entire set of packets at the root node of the game tree, then what we have is the sets of outcomes corresponding to every strategy of the player at the root node. Out of the entire set of packets, some packets may be inferior to other packets for every strategy of the player. What are the necessary and sufficient conditions for a packet to be inferior among the set of packets at a node? We identify two conditions as follows.

Clear Dominance. A packet P is inferior to a packet P' for a player if:

(1) each outcome in P' is either equal to some outcome in P or is strictly better than some outcome in P, and

(2) there exists at least one outcome in P' which is strictly better than some outcome in P.

In such cases we say that P' dominates P by *clear dominance.*

Dominance by Freedom of Choice. A packet P is inferior to a set of packets $P_1, \ldots, P_J$ if each packet P_i contains fewer outcomes than P, and the union of them yields P. In such cases we say that $P_1, \ldots, P_J$ dominates P by *freedom of choice.*

Before proving that these are the necessary and sufficient conditions for a packet to be inferior, we illustrate the idea of *dominance by freedom of choice* through an example. Note that the idea of *clear dominance* has already been illustrated in Example 2.2 and Example 2.3.

Example 2.5. Consider the game tree in Fig. 4. At node P, player-2 will always select the set $\{(11,5),(5,7)\}$ from the two sets $\{(11,5),(5,7)\}$ and $\{(11,12)\}$. Thus corresponding to move a at node G, we have two sets of outcomes, namely $\{(11,5)\}$ and $\{(5,7)\}$. At the nodes T and U, player-1 will always select $(11,5)$ and $(5,7)$ respectively. Thus, at node Q, player-2 can either select $(11,5)$ or $(5,7)$ depending on its strategy. Therefore, corresponding to move b we have the set of outcomes $\{(11,5),(5,7)\}$.

If the strategy of player-1 is such that $(11,5)$ is preferred over $(5,7)$, then the set $\{(11,5)\}$ will be preferred over $\{(11,5),(5,7)\}$ based on the procedure *Compare.* On the other hand, if the strategy of player-1 is such that $(5,7)$ is preferred over $(11,5)$, then $\{(5,7)\}$ will be preferred over $\{(11,5),(5,7)\}$. Thus, whatever be the strategy of player-1, the set $\{(11,5),(5,7)\}$ will never be selected. The sets $\{(11,5)\}$ and $\{(5,7)\}$ together provide more freedom of choice than the union $\{(11,5),(5,7)\}$ and therefore dominate the set $\{(11,5),(5,7)\}$.

Theorem 2.6. *The conditions of* clear dominance *and* dominance by freedom of choice *are sufficient conditions for a packet to be inferior among a set of packets.*

Proof. Suppose a packet P' dominates a packet P by *clear dominance*. Consider a strategy ST of the player. Let $\vec{x}$ be the worst outcome in P' based on ST. From the definition of clear dominance, there exists some outcome $\vec{y}$ in P which is either equal to $\vec{x}$ or worse for every strategy. If $\vec{y}$ is worse than $\vec{x}$ then P is obviously inferior. If $\vec{x}$ is equal to $\vec{y}$, we drop them from P and P', and use the same reasoning on the next worst outcome. Since there exists at least one outcome in P' which is strictly better than an outcome in P, it follows that P is inferior to P' for every strategy ST.

Suppose the set of packets $P_1, \ldots, P_J$ dominates a packet P by *freedom of choice*. Consider a strategy ST of the player. Let $\vec{x}$ be the best outcome in P on the basis of ST. From the definition of dominance by freedom of choice, there exists a packet P_i which is a subset of P and contains $\vec{x}$. Let us compare P_i and P on the basis of ST. If the worst outcome in P does not belong to P_i, then P is inferior. Otherwise, we drop that outcome from both sets and consider the next worst outcome and so on. Since P_i contains the best outcome in P and contains fewer outcomes than P, we find that P is inferior to P_i for the strategy ST. It follows that P is inferior to at least one of the of packets $P_1, \ldots, P_J$ for every strategy. □

Theorem 2.7. *For a packet to be inferior among a set of packets S, either of the conditions* clear dominance *or* dominance by freedom of choice *is necessary.*

Proof. Let us consider a packet P which is neither dominated by *clear dominance*, nor by *freedom of choice*. Let P_u denote the union of all packets in S which are subsets of P. Let P' denote the set of outcomes in P that are not in P_u. Since P is not dominated by freedom of choice, P' is non-empty. We now consider a strategy as follows:

(1) Every outcome in P_u has equal priority. Every outcome in P' has equal priority. Outcomes from P' are preferred over outcomes from P_u.

(2) Every outcome that does not dominate an outcome in P, or is not equal to an outcome in P, has a lower priority than every outcome in P.

Since P' is non-empty and its outcomes are preferred over those in P_u, it follows that P is preferred over all packets which are subsets of P. P is not dominated by clear dominance; therefore every packet that is not a subset of P must contain an outcome which neither dominates nor is equal to an outcome in P. Therefore, P is preferred over such packets as well. It follows that P is the most preferred packet based on the above strategy, and therefore not an inferior packet. □

The game tree search problem studied in this paper may now be defined as follows.

The Partial Order Game Tree Search Problem.

- *Given*: A game tree where the values at the tip nodes are K-dimensional vectors.
- *To find*: The set of non-inferior packets at the root node of the game tree.

To address the above problem, we will first analyze the problem of finding the minimal set of packets at a node n when the set of packets for the player at node n is given. For this purpose we shall use an algebra called *Dominance Algebra*. Subsequently, we shall address the problem of determining the set of packets at a node using partial order game tree search.

3. Dominance Algebra

Given the set of packets at a node, we have to identify the set of non-inferior packets (that is, those that may be selected by the player at that node). For convenience of representation we use an algebra to describe the type of operations that take place at the MAX- and MIN-nodes.

At a given node n, we have a set of packets $P_1, \ldots, P_m$ for the player who makes the move at that node. If n is a MAX-node, then we denote the options of player-1 at node n by a *MAX-expression* F_n as follows:

$$F_n = P_1 +_{\max} P_2 +_{\max} \cdots +_{\max} P_m.$$

The operator $+_{\max}$ is a commutative operator. Following the definition of packet dominance, we define another property of the $+_{\max}$ operator through the following *MAX-absorption law*.

MAX-absorption Law. A packet P_i at a MAX-node n can be absorbed under the following two situations:

- *Clear Dominance.* There exists a packet P_j at node n such that each outcome in P_j dominates (or is equal to) some outcome in P_i, and at least one outcome in P_j dominates an outcome in P_i. If this condition holds then:

  $$P_j +_{\max} P_i = P_j.$$

  In other words, P_i is absorbed in the MAX-expression F_n at node n.
- *Dominance by Freedom of Choice.* There exists a set of packets $P'_1, \ldots, P'_j$ at node n such that each of them contains fewer outcomes than P_i, and the union of them yields P_i. If this condition holds then:

  $$(P'_1 +_{\max} \cdots +_{\max} P'_j) +_{\max} P_i = P'_1 +_{\max} \cdots +_{\max} P'_j.$$

  In other words, P_i is absorbed in the MAX-expression F_n at node n.

Thus the $+_{\max}$ operator is actually the dominance operator over packets of outcomes. Using the MAX-absorption law, we can obtain a minimal MAX-expression at node n. A MAX-expression is actually a *set* of packets over which we can apply the MAX-absorption law to eliminate dominated packets. Therefore, throughout this paper we may use statements such as "a packet belongs to a MAX-expression" or "a packet is in a MAX-expression".

We now prove that given the set of packets at node n, application of the MAX-absorption law leads to a unique minimal MAX-expression F_n for that node. The following lemma shows that the sequence in which dominated packets are absorbed does not affect the final MAX-expression.

Lemma 3.1. *If F is a MAX-expression and P_1 and P_2 are packets such that $F +_{max} P_1 +_{max} P_2 = F$ (through some sequence of application of MAX-absorption law) then $F +_{max} P_1 = F$ and $F +_{max} P_2 = F$.*

Proof. It is easy to see that, if $F +_{\max} P_1 \neq F$ and $F +_{\max} P_2 \neq F$, then $F +_{\max} P_1 +_{\max} P_2$ cannot be equal to F. Without loss of generality, let $F +_{\max} P_1 = F$. (There is no loss of generality because $+_{\max}$ is commutative.) Now, if P_1 is not instrumental in the absorption of P_2, then the result follows trivially. Let us consider the cases where P_1 is used in the absorption of P_2. If P_1 is used in the absorption of P_2, then each outcome in P_1 must either dominate or be equal to some outcome in P_2.

The MAX-expression F can absorb P_1 through *clear dominance* or through *freedom of choice*. We analyze both situations:

- *Clear Dominance.* If F absorbs P_1 through clear dominance, then there exists a packet P_i in the MAX-expression F, such that P_i clearly dominates P_1. Then each outcome in P_i either dominates or is equal to some outcome in P_1, and at least one outcome in P_i dominates an outcome in P_1. Since each outcome in P_1 in turn either dominates or is equal to some outcome in P_2, we find that P_i absorbs P_2 through clear dominance.
- *Dominance by Freedom of Choice.* If F absorbs P_1 through freedom of choice then we can have two cases:
  (1) If P_1 absorbs P_2 through clear dominance, then there exists an outcome $\vec{x}$ in P_1 that dominates an outcome in P_2. Each outcome in P_1 must belong to some packet P_i in the MAX-expression F (from the definition of absorption by freedom of choice). Let P_i be the packet containing $\vec{x}$. Since every outcome in P_i is equal to some outcome in P_1, we find that P_i absorbs P_2 through clear dominance.
  (2) Since F absorbs P_1 through freedom of choice, there exist packets $P'_1, \dots, P'_j$ in the MAX-expression F, such that the set of packets $S_1 = \{P'_1, \dots, P'_j\}$ dominates P_1 by freedom of choice. Now if P_1 is used to absorb P_2 by freedom of choice then there exists a set S_2 of packets in the MAX-expression F, such that $S_2 \cup \{P_1\}$ absorbs P_2. It is easy to see then that $S_2 \cup S_1$ can absorb P_2 by freedom of choice. Therefore F absorbs P_2 by freedom of choice. □

The lemma effectively states that if P_1 is instrumental in the absorption of P_2, and P_1 is absorbed by F before the absorption of P_2 then F will also absorb P_2.

Theorem 3.2. *Given the set of packets for player-1 at a MAX-node n, application of the MAX-absorption law leads to a unique minimal MAX-expression for node n.*

Proof. If a packet is instrumental in the absorption of another packet then Lemma 3.1 shows that even if the former packet is absorbed earlier, the latter packet will still be absorbed. Thus the absorption of the dominated packets is independent of the order in which the packets are absorbed. The result follows. □

Given the set of packets for player-1 at a MAX-node we can individually test each packet to see whether it is absorbed by the other packets. Theorem 3.2 shows that this will lead to an unique minimal set of packets at the node. It may be easily shown that a similar analysis may be applied to the packets for player-2 (the opponent) at a MIN-node. If n is a MIN-node, then we denote the options of player-2 at node n by a *MIN-expression* F_n as follows:

$$F_n = P_1 +_{\min} P_2 +_{\min} \cdots +_{\min} P_m,$$

where $P_1, \ldots, P_m$ are packets for player-2 at node n. The operator $+_{\min}$ is a commutative operator similar to $+_{\max}$ except that it obeys the following *MIN-absorption law*.

MIN-absorption Law. A packet P_i at a MIN-node n can be absorbed under the following two situations:

- *Clear Dominance.* There exists a packet P_j at node n such that each outcome in P_j is dominated by (or is equal to) some outcome in P_i, and at least one outcome in P_j is dominated by an outcome in P_i. If this condition holds then:

$$P_j +_{\min} P_i = P_j.$$

- *Dominance by Freedom of Choice.* There exists a set of packets $P'_1, \ldots, P'_J$ at node n such that each of them contains fewer outcomes than P_i, and the union of them yields P_i. If this condition holds then:

$$(P'_1 +_{\min} \cdots +_{\min} P'_J) +_{\min} P_i = P'_1 +_{\min} \cdots +_{\min} P'_J.$$

Theorem 3.3. *Given the set of packets for player-2 at a MIN-node n, application of the MIN-absorption law leads to a unique minimal MIN-expression for node n.*

Proof. On lines similar to the proof of Theorem 3.2. □

Theorem 3.2 and Theorem 3.3 effectively prove that the $+_{\max}$ and $+_{\min}$ operators are associative. Dominance Algebra consists of sets (packets) of K-dimensional vectors and the two commutative and associative operators $+_{\max}$ and $+_{\min}$. Given the MAX-expression at a MAX-node (or a MIN-expression at a MIN-node), we can use the MAX-absorption law (MIN-absorption law) to obtain the minimal MAX-expression (MIN-expression) at that node.

4. Finding the packets

The question that has been answered so far is how to determine the minimal set of packets at a node when the entire set of packets at that node is given. Let us now address the problem of identifying the set of packets at a node. In particular, we address the following problems:

(1) How to find the minimal set of packets for player-1 at a given MAX-node n when the minimal set of packets for player-2 at the child MIN-nodes is given.
(2) How to find the minimal set of packets for player-2 at a given MIN-node n when the minimal set of packets for player-1 at the child MAX-nodes is given.

For the first problem, we define a function called *MIN-to-MAX* that converts a given MIN-expression to a MAX-expression. The function is based on the following result.

Lemma 4.1. *Let $\{P_1, \ldots, P_m\}$ be the minimal set of packets for player-2 at a MIN-node n. If we construct a set S of outcomes by selecting one outcome from each P_i, $1 \leqslant i \leqslant m$, then S is a packet for the parent MAX-node of n.*

Proof. We show that there exists a strategy for player-1 such that the final outcome either dominates or is equal to some outcome in S. If player-2 does not make a mistake, then it will select one of the packets $P_1, \ldots, P_m$. Without loss of generality, let us assume that player-2 selects the packet P_i. Then from the definition of a packet there exists a strategy for player-1 to reach any desired outcome from P_i. The result follows because, by the construction of S, one outcome in P_i belongs to S.

If player-2 makes a mistake, then it will select a packet P' that can be absorbed by $P_1 +_{\min} \cdots +_{\min} P_m$. Then, *every* outcome in those packets that are instrumental in the absorption of P' will be either dominated by or equal to some outcome in P'. Thus, if player-2 selects P', then there exists a strategy for player-1 to reach an outcome from P' which is either better than some outcome in S or equal to it. It follows that S is a packet for player-1. □

```
MIN-to-MAX(F: MIN-expression)
(1) Let $F = P_1 +_{\min} P_2 +_{\min} \cdots +_{\min} P_m$.
(2) Construct all possible sets of outcomes by selecting one outcome from each
    packet $P_i$, $1 \leqslant i \leqslant m$. Let these sets be $S_1, S_2, \ldots, S_J$.
(3) Using the MAX-absorption law, minimize the MAX-expression:
        $F' = S_1 +_{\max} S_2 +_{\max} \cdots +_{\max} S_J$.
(4) Return the MAX-expression $F'$.
```

Lemma 4.2. *If n_i is the ith child of the MAX-node n and F_{n_i} is the MIN-expression corresponding to the MIN-node n_i, then the set of packets for player-1 at node n through the child n_i can be represented by the MAX-expression returned by MIN-to-MAX(F_{n_i}).*

Proof. From Lemma 4.1 it follows that the set of packets constructed by the function *MIN-to-MAX* are packets for player-1 at node n. Once the game reaches node n_i, there exists a strategy for player-2 to ensure that the game reaches either some outcome from the selected packet at node n_i, or an outcome that is dominated by some outcome from the selected packet. Therefore, through node n_i player-1 can (at best) reach only those outcomes that are present in the packets at node n_i. The result follows. □

Theorem 4.3. *If F_{n_i} denotes the MIN-expression of the ith child of the MAX-node n, then the MAX-expression of the node n is:*

$$F_n = \mathit{MIN\text{-}to\text{-}MAX}(F_{n_1}) +_{max} \mathit{MIN\text{-}to\text{-}MAX}(F_{n_2}) +_{max} \cdots +_{max} \mathit{MIN\text{-}to\text{-}MAX}(F_{n_m}),$$

where m denotes the number of children of node n.

Proof. Follows from Lemma 4.2. □

The above analysis shows that using the *MIN-to-MAX* function, we can construct the set of packets (for player-1) at a MAX-node when the set of packets (for player-2) is given for each child MIN-node. The set of packets at a MIN-node can be constructed in a very similar fashion using the following *MAX-to-MIN* function.

```
MAX-to-MIN(F: MAX-expression)
(1) Let $F = P_1 +_{\max} P_2 +_{\max} \cdots +_{\max} P_m$.
(2) Construct all possible sets of outcomes by selecting one outcome from each
    packet $P_i$, $1 \leqslant i \leqslant m$. Let these sets be $S_1, S_2, \ldots, S_J$.
(3) Using the MIN-absorption law, minimize the MIN-expression:
        $F' = S_1 +_{\min} S_2 +_{\min} \cdots +_{\min} S_J$.
(4) Return the MIN-expression $F'$.
```

Theorem 4.4. *If F_{n_i} denotes the MAX-expression of the ith child of the MIN-node n, then the MIN-expression of the node n is:*

$$F_n = \mathit{MAX\text{-}to\text{-}MIN}(F_{n_1}) +_{min} \mathit{MAX\text{-}to\text{-}MIN}(F_{n_2}) +_{min} \cdots +_{min} \mathit{MAX\text{-}to\text{-}MIN}(F_{n_m}),$$

where m denotes the number of children of node n.

Proof. On lines similar to the proof of Theorem 4.3. □

At the leaf nodes of the game tree, the values themselves are the packets. Using the *MAX-to-MIN* and *MIN-to-MAX* functions and the absorption laws, we can now compute (in a bottom-up manner) the minimal set of packets at each node of the game tree (and ultimately those at the root node). Let us consider an example.

Example 4.5. Consider the game tree in Fig. 5. Let F_i denote the MAX-expression at the MAX-node i and the MIN-expression at the MIN-node i. It is easy to see that $F_4 = \{(5,4)\} +_{\min} \{(3,10)\}$ and $F_5 = \{(11,5)\} +_{\min} \{(5,9)\}$. Therefore, *MIN-to-MAX*(F_4) is $\{(5,4),(3,10)\}$ and *MIN-to-MAX*(F_5) is $\{(11,5),(5,9)\}$. The MAX-expression F_3 can be computed as:

$$F_3 = \{(5,4),(3,10)\} +_{\max} \{(11,5),(5,9)\} = \{(11,5),(5,9)\}.$$

Likewise, it is easy to see that $F_7 = \{(12,7)\} +_{\min} \{(7,12)\}$ and $F_{15} = \{(13,6)\} +_{\min} \{(6,13)\}$. Therefore, *MIN-to-MAX*(F_7) is $\{(12,7),(7,12)\}$ and *MIN-to-MAX*(F_{15}) is $\{(13,6),(6,13)\}$. The MAX-expression F_6 can be computed as:

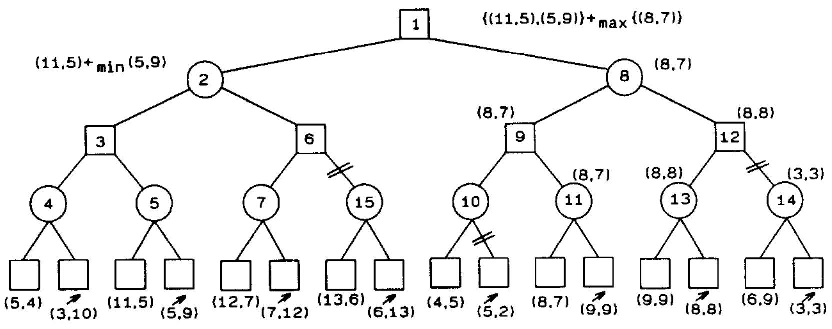

Fig. 5. Game tree showing partial order pruning.

$$F_6 = \{(12,7),(7,12)\} +_{\max} \{(13,6),(6,13)\}.$$

MAX-to-MIN(F_3) can be computed as $\{(11,5)\} +_{\min} \{(5,9)\}$. Also, *MAX-to-MIN*($F_6$) can be computed as:

$$\{(12,7),(13,6)\} +_{\min} \{(12,7),(6,13)\} +_{\min} \{(7,12),(13,6)\}$$
$$+_{\min} \{(7,12),(6,13)\}.$$

The MIN-expression F_2 can now be computed as:

$$\begin{aligned} F_2 &= \{(12,7),(13,6)\} +_{\min} \{(12,7),(6,13)\} +_{\min} \{(7,12),(13,6)\} \\ &\quad +_{\min} \{(7,12),(6,13)\} +_{\min} \{(11,5)\} +_{\min} \{(5,9)\} \\ &= \{(11,5)\} +_{\min} \{(5,9)\}. \end{aligned}$$

Looking at the other side of the game tree, the MIN-expression at node 10 is $F_{10} = \{(4,5)\} +_{\min} \{(5,2)\}$ and that at node 11 (using the MIN-absorption law) is $F_{11} = \{(8,7)\}$. Using, the *MIN-to-MAX* function on F_{10} and F_{11} we obtain the MAX-expression at node 9 as follows:

$$F_9 = \{(4,5),(5,2)\} +_{\max} \{(8,7)\} = \{(8,7)\}.$$

Similarly, using the MIN-absorption law we have $F_{13} = \{(8,8)\}$ and $F_{14} = \{(3,3)\}$. Therefore the MAX-expression at node 12 is:

$$F_{12} = \{(8,8)\} +_{\max} \{(3,3)\} = \{(8,8)\}.$$

The MIN-expression at node 8 can now be computed as:

$$F_8 = \{(8,7)\} +_{\min} \{(8,8)\} = \{(8,7)\}.$$

Now, *MIN-to-MAX*(F_2) is $\{(11,5),(5,9)\}$ and *MIN-to-MAX*(F_8) is $\{(8,7)\}$. Therefore, the MAX-expression at node 1 is:

$$F_1 = \{(11,5),(5,9)\} +_{\max} \{(8,7)\}.$$

Thus at the root node of the game tree of Fig. 5, player-1 has two packets to choose from, namely $\{(11,5),(5,9)\}$ and $\{(8,7)\}$.

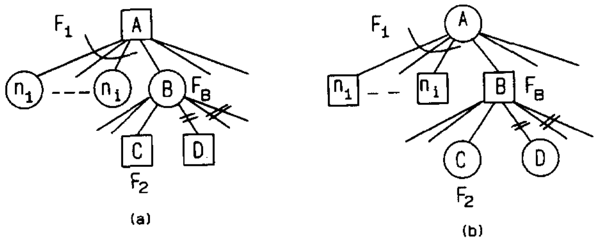

Fig. 6. Shallow pruning.

5. Partial order α-β pruning

In the previous section, we have seen how the set of non-dominated packets can be identified. However, the bottom-up approach is a brute force method that may require too much time and space. If we can find conditions under which certain branches of the game tree can be pruned, then we can apply techniques similar to α-β pruning to achieve our objective more efficiently.

We define two boolean functions *max_prune*(F_1, F_2) and *min_prune*(F_1, F_2) as follows:

Definition 5.1 (*Dominance test for pruning*).

- *max_prune*(F_1, F_2): If every packet in the MAX-expression F_2 is absorbed by the MAX-expression F_1 using clear dominance (of the MAX-absorption law), then *max_prune*(F_1, F_2) is true, else it is false.
- *min_prune*(F_1, F_2): If every packet in the MIN-expression F_2 is absorbed by the MIN-expression F_1 using clear dominance (of the MIN-absorption law), then *min_prune*(F_1, F_2) is true, else it is false.

Using the above functions, we define pruning conditions in partial order game trees which are somewhat analogous to α-β pruning in conventional game trees.

5.1. Shallow α-β pruning

The following lemmas help in identifying shallow pruning conditions.

Lemma 5.2. *Consider the game tree in Fig. 6(a). F_1 denotes the MAX-expression obtained at node A by collecting the packets (for player-1) backed up by the children $n_1, \ldots, n_i$ only. F_2 denotes the MAX-expression for the entire set of packets (for player-1) backed up at node C. F_B denotes the MIN-expression for the entire set of packets backed up (for player-2) at node B. If max_prune(F_1, F_2) is true, then the following equality holds.*

$$F_1 +_{max} \mathit{MIN\text{-}to\text{-}MAX}(F_B) = F_1.$$

Proof. Let us assume the contrary. Then there exists a packet P in *MIN-to-MAX*(F_B) which cannot be absorbed by F_1. This means that if player-1 takes the move to B, there exists a strategy of player-1 to reach an outcome in P or some outcome which dominates an outcome in P. Thus, even if player-2 takes the move to C, then there exists a subset P' of P, such that player-1 is able to reach some outcome in P' (or some outcome that dominates an outcome in P'). Therefore P' is either a packet at node C, or dominated by some packet at node C. In either case, P' can be absorbed by some packet P_i in F_1 using clear dominance. Since P' is a subset of P, it follows that P can also be absorbed by P_i using clear dominance. This contradicts our assumption that the packet P cannot be absorbed by F_1. The result follows. □

Lemma 5.2 shows that shallow pruning can be effected on the other successors of node B when *max_prune*(F_1, F_2) is true, where F_1 and F_2 are as described in the statement of the lemma. It may be noted that in Fig. 6(a), if F_1 absorbs a packet of F_2 using freedom of choice (in the MAX-absorption law), then the pruning is not possible. For example, let F_1 be $\{(11,5)\} +_{\max} \{(5,7)\}$ and F_2 be $\{(11,5),(5,7)\}$. Let the MAX-expression F_D at node D be $\{(2,9)\}$. Let there be no other children of node B (except nodes C and D). Then node B can back up the packet $\{(11,5),(5,7),(2,9)\}$ at node A, which cannot be absorbed by F_1. If node D would have been pruned, then this packet would not be backed up. This is the reason for our using only the clear dominance criteria (of the absorption laws) for defining the pruning conditions.

Reasoning in a similar way, we can define pruning conditions where the function *min_prune* is applicable.

Lemma 5.3. *Consider the game tree in Fig.* 6(b). F_1 *denotes the MIN-expression obtained at node A by collecting the packets (for player-*2*) backed up by the children* $n_1, \ldots, n_i$ *only.* F_2 *denotes the MIN-expression for the entire set of packets (for player-*2*) backed up at node C.* F_B *denotes the MAX-expression for the entire set of packets backed up (for player-*1*) at node B. If min_prune*(F_1, F_2) *is true, then the following equality holds.*

$$F_1 +_{min} \mathit{MAX\text{-}to\text{-}MIN}(F_B) = F_1.$$

Proof. On lines similar to the proof of Lemma 5.2. □

5.2. *Deep α-β pruning*

Deep pruning conditions can also be identified for partial order game trees as follows.

Lemma 5.4. *Consider a path in a game tree as shown in Fig.* 7(a). F_i *denotes the MAX-expression formed by collecting the set of packets (for player-*1*) backed up by those children of MAX-node* n_i *which are to the left of the child in the given path.* F_C *denotes the set of packets (for player-*1*) at node C. Let F denote the following MAX-expression:*

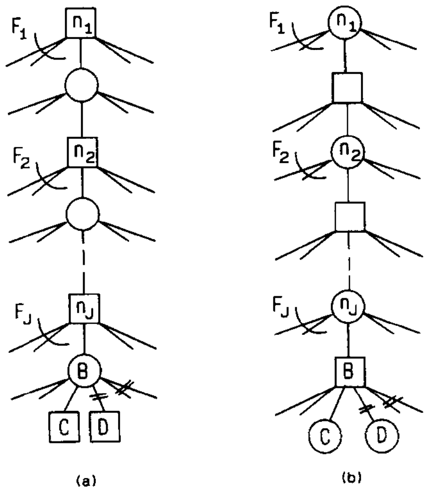

Fig. 7. Deep pruning.

$$F = F_1 +_{max} F_2 +_{max} \cdots +_{max} F_J.$$

*If max_prune(F, F_C) is true, then at node n_J the move to node B will never be selected by player-*1.

Proof. Let us assume the contrary, that is, player-1 selects the move to node B when the game reaches node n_J. Since player-2 can then select the move to node C, it is easy to see that each packet (for player-1) backed up by node B to n_J must contain a subset which is either a packet at node C or dominated by a packet at node C.

In the partial order game tree search problem, player-1 will have a set of packets of outcomes to choose from at the root node n_1. By definition, if the player-1 selects a packet, then there exists a strategy for it to ensure that the final outcome is either some outcome from that set or dominates some outcome from that set. Therefore, if the strategy of player-1 is such that it selects the packet P from the set of packets at the root node, then the game will go through those MAX-nodes n_i that has a non-dominated packet which is either some subset of P, or dominates some subset of P (and therefore, dominates P).

Therefore, if the game reaches node B, then player-1 must have selected a packet P at the root node such that there exists a packet P' (for player-1) backed up by node B to node n_J which is either a subset of P or dominates P. It follows that in each node n_i in the path to n_J, there exists a non-dominated packet P_i such that either P' is a subset of P_i, or P' dominates P_i.

Since P' is a packet backed up by node B, it contains a subset which is either a packet P_c at node C or dominated by a packet P_c at node C. Since *min_prune*(F, F_C) is true, there exists a packet P_i' at some node n_i which dominates P_c using clear dominance. P_i' will then also dominate P' and therefore, will dominate P_i. Thus P_i is a dominated packet at node n_i which brings us to a contradiction. The result follows. □

Using similar reasoning, we can prove the following lemma.

Lemma 5.5. *Consider a path in a game tree as shown in Fig.* 7(b). *F_i denotes the MIN-expression formed by collecting the set of packets (for player-2) backed up by those children of MIN-node n_i which are to the left of the child in the given path. F_C denotes the set of packets (for player-2) at node C. Let F denote the following MAX-expression:*

$$F = F_1 +_{min} F_2 +_{min} \cdots +_{min} F_J.$$

If min_prune(F, F_C) is true, then at node n_J, the move to node B will never be selected by player-1.

Proof. On lines similar to the proof of Lemma 5.4. □

Lemma 5.4 and Lemma 5.5 allows us to define *α-expressions* for MIN-nodes and *β-expressions* for MAX-nodes (using the idea of the α and β bounds in conventional game tree search).

Definition 5.6 (*α-expression*). The α-expression at a MIN-node B is the MAX-expression formed by collecting the packets (for player-1) currently backed up at all MAX ancestors of B.

Lemma 5.4 shows that the exploration of a MIN-node B can be terminated as soon as the MAX-expression backed up by any of its children is absorbed by the α-expression at B (using clear dominance).

Definition 5.7 (*β-expression*). The β-expression at a MAX-node B is the MIN-expression formed by collecting the packets (for player-2) currently backed up at all MIN-ancestors of B.

Lemma 5.5 shows that the exploration of a MAX-node B can be terminated as soon as the MIN-expression backed up by any of its children is absorbed by the β-expression at B (using clear dominance).

Using these results, we may now develop an algorithm on lines similar to the α-β pruning algorithm of conventional game tree search. Given heuristic vector estimates $e(n)$ at each terminal node n, the following procedure uses a technique similar to α-β pruning to determine the set of non-dominated packets for player-1. The call $F(s, -\vec{\infty}, +\vec{\infty})$ returns the MAX-expression at node s. $+\vec{\infty}$ is a packet containing a single K-dimensional outcome which has all dimensions equal to $+\infty$. Likewise $-\vec{\infty}$ is a packet containing a single K-dimensional outcome which has all dimensions equal to $-\infty$.

```
Procedure $F(n, \alpha, \beta)$
1. IF $n$ is a terminal node THEN Return $e(n)$
2. Generate successors $m_1, m_2, \ldots, m_b$ of $n$
```

```
2.1. Set $i \leftarrow 1$
2.2. IF $n$ is a MAX-node THEN
    2.2.1. Set $F_n \leftarrow -\vec{\infty}$
    2.2.2. Repeat the following steps until $i > b$
        2.2.2.1. Set $F_{m_i} \leftarrow F(m_i, \alpha, \beta)$
        2.2.2.2. IF $F_{m_i} \neq -\vec{\infty}$ THEN
                     Set $\alpha \leftarrow \alpha +_{\max} F_{m_i}$
                     Set $F_n \leftarrow F_n +_{\max} F_{m_i}$
                     IF $min\_prune(\beta, MAX\text{-}to\text{-}MIN(F_{m_i}))$ is true THEN
                        Return $-\vec{\infty}$
        2.2.2.3. Set $i \leftarrow i + 1$
    2.2.3. Return $F_n$
2.3. IF $n$ is a MIN-node THEN
    2.3.1. Set $F_n \leftarrow +\vec{\infty}$
    2.3.2. Repeat the following steps until $i > b$
        2.3.2.1. Set $F_{m_i} \leftarrow F(m_i, \alpha, \beta)$
        2.3.2.2. IF $F_{m_i} \neq -\vec{\infty}$ THEN
                     Set $\beta \leftarrow \beta +_{\min} MAX\text{-}to\text{-}MIN(F_{m_i})$
                     Set $F_n \leftarrow F_n +_{\min} MAX\text{-}to\text{-}MIN(F_{m_i})$
                     IF $max\_prune(\alpha, F_{m_i})$ is true THEN
                        Return $-\vec{\infty}$
        2.3.2.3. Set $i \leftarrow i + 1$
    2.3.3. Return $MIN\text{-}to\text{-}MAX(F_n)$
```

The working of the algorithm is illustrated on the game tree of Fig. 5. The sequence of nodes visited, and the corresponding values of α and β is as shown in Table 1. In MAX-nodes, $F(n)$ denotes the MAX-expression and in MIN-nodes, $F(n)$ denotes the MIN-expression. For convenience of writing, we have written packets containing a single outcome (a,b) as "(a,b)" instead of "$\{(a,b)\}$".

6. Conclusion

The problem of searching game trees under a total order has been well studied in the past. In this paper we have investigated the more general problem of searching a game tree under a partial order.

The contents of this paper focuses on the general issues in partial order game tree searching. The pruning conditions and basic search strategies can be enhanced to cater to the formulations of specific problems. For example, it is likely that if a problem involves many criteria, and each of them are modeled as separate dimensions of the cost vector, then very little pruning is possible as most outcomes become non-dominated. In order to enhance the pruning in such situations a judicious combination of some of the criteria may be called for. In the other extreme case, when all the criteria can be meaningfully combined, the proposed pruning conditions collapses to the pruning conditions of total order game tree search.

Table 1
Sequence of nodes visited by *procedure F* on the game tree of Fig. 5

n	$F(n)$	α	β
n_1, n_2, n_3	-	$-\vec{\infty}$	$+\vec{\infty}$
n_4	$(5,4) +_{\min} (3,10)$	$-\vec{\infty}$	$(5,4) +_{\min} (3,10)$
n_3	-	$\{(5,4),(3,10)\}$	$+\vec{\infty}$
n_5	$(11,5) +_{\min} (5,9)$	$\{(5,4),(3,10)\}$	$(11,5) +_{\min} (5,9)$
n_3	$\{(5,4),(3,10)\} +_{\max} \{(11,5),(5,9)\} = \{(11,5),(5,9)\}$	$\{(11,5),(5,9)\}$	$+\vec{\infty}$
n_2, n_6	-	$-\vec{\infty}$	$(11,5) +_{\min} (5,9)$
n_7	$(12,7) +_{\min} (7,12)$	$-\vec{\infty}$	$(11,5) +_{\min} (5,9) +_{\min} (12,7) +_{\min} (7,12) = (11,5) +_{\min} (5,9)$
n_6[a]	-	$\{(12,7),(7,12)\}$	$(11,5) +_{\min} (5,9)$
n_2	$(11,5) +_{\min} (5,9)$	$-\vec{\infty}$	$(11,5) +_{\min} (5,9)$
n_1, n_8, n_9	-	$\{(11,5),(5,9)\}$	$+\vec{\infty}$
n_{10}[a]	-	$\{(11,5),(5,9)\}$	$(4,5)$
n_9	-	$\{(11,5),(5,9)\}$	$+\vec{\infty}$
n_{11}	$(8,7) +_{\min} (9,9) = (8,7)$	$\{(11,5),(5,9)\}$	$(8,7)$
n_9	$(8,7)$	$\{(11,5),(5,9)\}$	$+\vec{\infty}$
n_8, n_{12}	-	$\{(11,5),(5,9)\}$	$(8,7)$
n_{13}	$(9,9) +_{\min} (8,8) = (8,8)$	$\{(11,5),(5,9)\}$	$(8,7) +_{\min} (8,8) = (8,7)$
n_{12}[a]	-	$\{(11,5),(5,9)\} +_{\max} (8,8)$	$(8,7)$
n_8	$(8,7)$	$\{(11,5),(5,9)\}$	$(8,7)$
n_1	$\{(11,5),(5,9)\} +_{\max} (8,7)$	$\{(11,5),(5,9)\} +_{\max} (8,7)$	$+\vec{\infty}$

[a] Pruning occurs at these nodes.

Acknowledgements

The authors would like to thank the reviewers for their comments, which helped in improving both the contents as well as the style of presentation of the paper.

References

[1] D.E. Knuth and R.W. Moore, An analysis of alpha-beta pruning, *Artif. Intell.* **6** (1975) 293-326.
[2] R.E. Korf, Multi-player alpha-beta pruning, *Artif. Intell.* **48** (1991) 99-111.
[3] C.H. Papadimitriou and M. Yannakakis, Shortest paths without a map, *Theoret. Comput. Sci.* **84** (1991) 127-150.
[4] J. Pearl, *Heuristics* (Addison-Wesley, Reading, MA 1984).
[5] B.S. Stewart and C.C. White, Multiobjective *A**, *J. ACM* **38** (1991) 775-814.
[6] G.C. Stockman, A minimax algorithm better than alpha-beta?, *Artif. Intell.* **12** (1979) 179-196.