

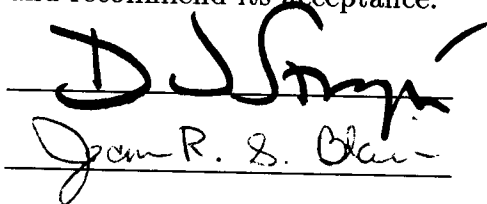
To the Graduate Council:

I am submitting herewith a thesis written by Cheng Liu entitled "A recursive algorithm for an imperfect information game with perfect recall". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.



Dr. David C. Mutchler, Major Professor

We have read this thesis
and recommend its acceptance:



Accepted for the Council:



Associate Vice Chancellor
and Dean of the Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature

Cheng Hsiao

Date

5/28/92

**A RECURSIVE ALGORITHM
FOR AN IMPERFECT INFORMATION
GAME WITH PERFECT RECALL**

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Cheng Liu

August 1992

In memory of my beloved father

ACKNOWLEDGEMENTS

Grateful thanks and appreciation are expressed to my advisor, Dr. David Mutchler, for his continued help to develop, improve, and complete this thesis and for his understanding, patience, and cooperation throughout the course of this research. Sincere thanks and appreciation also go to the other members of my thesis committee, Dr. Jean Blair and Dr. David Straight for their suggestions in preparation of the manuscript.

ABSTRACT

The traditional method to find an optimal strategy for a general n -player game is to enumerate all pure strategies and use a linear programming technique to find the strategy with highest payoff. Unfortunately, the number of pure strategies in even a small game is too large to be enumerated. In this work we developed a recursive algorithm to find an optimal solution in a one player game with perfect recall. This algorithm evaluates each leaf in the game tree only once. This significantly reduces the time compared to the traditional method. This algorithm can also be used in an n -player game with perfect recall to check if a pure strategy is an equilibrium point.

Contents

1	Introduction	1
2	Background of Game Theory	6
2.1	Games in Extensive Form	7
2.2	Strategies and Payoffs	11
2.3	Equilibrium Point in a Game	13
3	An Algorithm for Imperfect Information Games	15
3.1	Corresponding One Person Game	15
3.2	Frontier Set	16
3.3	An Evaluation Function for Frontier Set	18
3.4	Exactness and Recursive Exactness	23
3.5	An Algorithm For the Corresponding One Person Game	30
3.6	The Efficiency of the Algorithm	35
4	Conclusion	47
4.1	Summary	47
4.2	Future Study	48
	Bibliography	50
	Appendix	54

Chapter 1

Introduction

The traditional method to find an optimal strategy for a general n -player game is to enumerate all pure strategies and use a linear programming technique to find the strategy with highest payoff. Unfortunately, the number of pure strategies in even a small game is too large to be enumerated. Furthermore, finding an optimal strategy in general, even for a one player game is NP-hard[Blair, Mutchler, and Liu, 1992]. However, some games have a property called perfect recall, meaning that a player can always remember all his previous moves. For example, bridge can be treated as a four-player game with perfect recall or a two-team game without perfect recall. In this work we review the literature of games and suggest a recursive algorithm to find an optimal solution in a one player game with perfect recall. This algorithm evaluates each leaf in the game tree only once, and thus significantly reduces the time compared to the traditional method. This algorithm can also be used in an n -player game with perfect recall to check whether a pure strategy is an equilibrium point.

In the remainder of this chapter we give an informal description of a game and its solution (a strategy) to motivate what lies ahead. Chapter 2 gives the formal definitions. A game is played by one or more players. A player chooses one move from several legal alternatives, one at a time. Each player makes a move in turn. If a game is finite then a play in the game eventually reaches

an ending point where there are no more legal moves left. Each player obtains a payoff according to the position of the ending point in the game tree. There are two forms in which to represent a game: tree structured form (also called extensive form [Kuhn, 1953]) and normalized form (also called matrix form). A game represented in extensive form can always be transformed into normalized form. In a two-player game with normalized form, there is a payoff matrix A and the entry $a_{i,j}$ is the payoff if player 1 uses strategy i and player 2 uses strategy j . "The normalized form is better suited to the derivation of general theorems while the extensive form exposes the characteristic differences between games" [Kuhn, 1952, p193].

Many games have random moves such as shuffling a deck of cards or throwing dice. We can image that there is a chance player whose responsibility is doing the chance move. The chance player does not have a payoff and ending point.

There are two ways to play the game. One way is to let players play the game and observe the outcome after the game is over. Another is to let each player announce his strategy before the game starts. An umpire then collects a strategy from each player and decides payoffs for players from a payoff matrix. Those pre-announced strategies are called pure strategies. We use the following example to explain pure strategy.

A pure strategy can be treated as a questionnaire with multiple choices. The questions in the questionnaire may look like:

- 1. You make the first move, what is your action? 1. 2. or 3.
- 2. Your opponent just made a move to ..., what is your action? 1. 2. 3. or 4.
- 3. Given the situation ..., what is your action? 3, 4, 6.
- 4. ...

A questionnaire lists all possible situations that could be encountered in a game and also lists all legal moves under each such situation. Each player answers all questions in a questionnaire and gives it to an umpire. The umpire examines the questionnaires from all players and decides what the play in the game will be. From the play and the rules of the game, the umpire then assigns a payoff for each player.

In some games the move of a player (including the chance player) is notified to all players, for example chess and backgammon. This is called a perfect information game. In some games the move of a player (including the chance player) is unknown to other players. For example, in bridge the move of a chance player at the beginning of a game (shuffling a deck of cards) is unknown to all other players. This is called an imperfect information game. Furthermore a player may remember all his previous moves. This is called perfect recall.

Does there exist a strategy that always leads a player to a position with highest payoff? The payoffs of a play in a game depend on the strategy of the player and also on strategies of other players. Since a player does not know which strategies other players are using, he has to make decisions by guessing other players strategies. Some times it is difficult to predict the other players' behavior. That is, an opponent may choose a good move, a random move or even a bad move. One of the decisions a player might make is to choose a strategy which guarantees a highest payoff in the worst case. That is, a player may not want to take a risk and therefore may assume that the other players are good players. If all players make the same assumption and if there is a set of strategies such that no player can obtain a higher payoff by changing his strategy, then this set of strategies is called an equilibrium point in the game.

Some classes of games are guaranteed to have an equilibrium point, regardless of the payoffs. Some games have no equilibrium point. Some games

have equilibrium points only in pure strategies, while other games have equilibrium points in mixed strategies. (A mixed strategy is one where a player has to randomly choose among several pure strategies to achieve an equilibrium point.) Intuitively, a perfect information game should have equilibrium points in pure strategies because after a player makes a move this move is notified to other players. A player knows exactly his opponents' moves so he does not have to use random strategies to guess his opponents' move. Indeed, Kuhn [1953] proved that every perfect information game has at least one equilibrium point in pure strategies. It seems difficult to have equilibrium points in pure strategies for an imperfect information game because a player does not know his opponents' moves so he may have to use random strategies to gain his highest payoff. Dalkey [1953] proved that for a game with any payoff values, there exists at least one equilibrium point in pure strategies if and only if the game has effectively perfect information (i.e. a player knows at least as much as his opponent previously knew). How about an imperfect information game with special payoff values? The easiest example is a game with constant payoffs. It does not matter what players know, any set of pure (or mixed) strategies is an equilibrium point.

In this work, we are dealing with imperfect information games in extensive form with perfect recall and pure strategies. If a player can magically guess all his opponents' moves then it is equivalent to a one person game, and we called it a corresponding one person game. We develop here a recursive algorithm to find a pure strategy for a corresponding one person imperfect information game with perfect recall. This algorithm evaluates each leaf exact once. We also modify this algorithm to check whether a set of pure strategies in an n -player game is an equilibrium point.

Chapter 2 discusses the background of game theory and gives formal definition of terms used in game theory. Chapter 3 develops a recursive algorithm

for a corresponding one person imperfect information game with perfect recall. Chapter 4 contains the conclusion and some discussion of possibilities for future study.

Chapter 2

Background of Game Theory

This chapter gives the formal definitions of games in extensive form and terminologies used in game theory. The general idea of a game is that it starts from a given position and follows a sequence of moves. At each move either a player chooses an alternative from several possibilities or a die is thrown as a chance (random) move. Examples of games are chess, bridge, and backgammon. There is no chance move in chess except for the determination of who shall play first. Shuffling a deck of cards in bridge is the chance move at the starting position. Throwing a die in a backgammon is a chance move during the game.

There is an important difference between chess (or backgammon) and bridge. In a chess (or backgammon) game each player has perfect knowledge of moves while in a bridge game each player has only partial information about the moves. In a game of imperfect information, each player makes a decision under the condition that he may not be able to determine the exact position of the game.

At the end of a game, there is a payoff to each player. In chess, it is win, loss, or draw. In bridge or backgammon it is a winning or losing number. The following section makes a formal definition of games.

2.1 Games in Extensive Form

There are three elements in a general game: (1) moves (personal or chance), (2) a possible lack of knowledge, and (3) a payoff function. We shall follow with minor modifications the definition of games in extensive form given by Kuhn (1953).

Definition 2.1.1 A game tree K is a rooted, directed, finite tree. Throughout, the symbol K will refer to a game tree.

Before giving the definition of a game, it is necessary to define the several terms which are associated with the game tree.

Definition 2.1.2 The alternatives at a vertex $x \in K$ are the arcs in K with x as the head. Those vertices in K which possess alternatives (internal nodes) are called moves. The number of alternatives at x is denoted $m(x)$. The vertices in K which have no alternative are called leaves. Throughout, the symbol W denotes the set of leaves in K .

Definition 2.1.3 A node y follows node x , denoted $y > x$, if $x \neq y$ and there is a directed path from x to y . A set of nodes Y follows set X of nodes, denoted $Y > X$, if $\exists x \in X$ and $\exists y \in Y$ such that $y > x$. A node y is a descendant of a node x , if either $x = y$ or $y > x$. We let $D(X)$ denote the set of all descendants of nodes in X .

Definition 2.1.4 A play in a game tree K is a path from the root to a leaf.

Definition 2.1.5 A n -person game $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ in extensive form is defined by

- a game tree K whose root is called the starting position of K .

- a player partition $\mathcal{P} = \{P_i\}_{i=0}^n$ which partitions the moves into $n + 1$ sets. The moves in P_0 are called chance moves. For $i = 1, \dots, n$, the moves in P_i are called personal moves for player i .
- an information partition $\mathcal{U} = \{U_i\}_{i=1}^n$ where each U_i partitions P_i . Each U_i is thus a set of sets of personal moves for player i . For any U_i each set in U_i is called an information set. An information set has additional restrictions:
 - No information set intersects the same play more than once.
 - For any information set u , $\forall$ moves $x, y \in u$, we have $m(x) = m(y)$.
- a collection of probability distributions, one per chance move, as specified by the function $p: P_0 \times K \rightarrow R$, where for any fixed chance move x , $p(x, \cdot)$ is a probability distribution on the children of x . That is, $p(x, y)$ gives the probability of child y being selected by the chance “player” when playing the game at move x .
- a pay-off function $h: W \rightarrow \mathcal{R}^n$.

Note the two restrictions on information sets. First, no information set contains two moves in the same play. This implies that the player always remembers the number of moves he made during a game. Then he can always distinguish two moves if those two moves are in the same play. The second restriction above means that if two moves have a different number of alternatives then the players can locate their position in the game by the difference.

An example game is the chess game. The chess game is a two person game in which the two players alternately make their moves. Assume that player one makes the first move, then the root of the game tree is a personal move for player one. According to the rules of the chess game, player one has twenty legal moves. That is eight possible ways of moving one of the pawns forward one step, eight

possible ways to move one of the pawns forward two steps and four possible ways to move one of knights. The game tree in the chess game has a root and the root has twenty arcs out of the root. Each arc points to a node and these twenty nodes are personal moves for player two. Since player two has also twenty legal alternatives according to the rules of the chess game, then the third move of the play (the second move of player one) has a total of four hundred moves. During the game, these two players make decisions alternately and the sequence of these decisions is a play. Since chess is a finite game, the play eventually reaches a position in which no legal move is left for either player. This position is called a leaf. A win, loss, or draw is assigned to a leaf and this is called a payoff.

The chess game has special properties: no chance move is involved and each player knows his and the other player's moves. Under this circumstance the player knows exactly his position in the game tree. This is called a perfect information game.

Definition 2.1.6 *Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game. Γ is called a perfect information game if for every player i and every information set $u \in U_i$, u contains only one node. Γ is called an imperfect information game if it is not a perfect information game.*

Although backgammon contains chance moves in a game tree, it is a perfect information game because the outcome from the chance move is informed to all players. An example of imperfect information games is the bridge game. The bridge game is a four player game. Those players are named West, North, East, and South. Each player gets thirteen cards out of a deck of fifty-two cards which has been randomly shuffled by an umpire. Each player sees only his own 13 cards. In the game tree of bridge the root is a chance move and it represents shuffling the cards. There are $\binom{52}{13} \times \binom{39}{13} \times \binom{26}{13}$ arcs out of the root. Each arc points to

a node. Assume that player West makes the first move (lead) in the game; then all these nodes are personal moves for player West. Those personal moves are grouped into personal information sets. Since West can locate 13 cards out of 52, there are $\binom{52}{13}$ information sets for his first move. Each information set contains $\binom{39}{13} \times \binom{26}{13}$ moves because player West has to guess the distribution of the other 39 missing cards. Player North makes the second move. Player North shows his cards to all players and player South makes the decisions for player North. There are $\binom{52}{13} \times \binom{39}{13} \times \binom{26}{13} \times 13$ personal moves for player North because there are 13 alternatives for each player West personal move. Those personal moves for player North are also grouped into information sets in which each information set contains $\binom{25}{13}$ moves, because North can locate 27 cards out of 52 (1 card from player West, 13 cards from South and 13 cards from himself). Player East makes the third move where there are $\binom{52}{13} \times \binom{39}{13} \times \binom{26}{13} \times 13 \times 13$ moves because player North also has 13 alternatives. Those personal moves for player East are also grouped into information sets in which each information set contains $\binom{25}{13}$ moves, because East can locate 27 cards out of 52 (1 card from West, 13 cards from North and 13 cards from himself). To end the first round, player South makes the final move. There are $\binom{52}{13} \times \binom{39}{13} \times \binom{26}{13} \times 13 \times 13 \times 13$ moves because player East also has 13 alternatives. Following the same logic, each information set for player South contains $\binom{24}{12}$ moves because North can locate 28 cards out of 52 (1 card from West, 13 cards from North, 1 card from East, and 13 cards from himself). It is clear that the game tree in bridge is extremely large.

Sometimes in a game, a player may forget his previous moves. If a player always remembers his previous move then this game is called perfect recall for that player. We need the following definitions to formally define a game with perfect recall.

Definition 2.1.7 Let X be a subset of information set u . Then the children of X are $Y_1, \dots, Y_{m(u)}$, where

$$Y_i = \{y \mid y \text{ is the } i\text{th child of a node in } X\}$$

Each such Y_i is a child of X .

Definition 2.1.8 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game. Γ is called perfect recall for player i if for any two information sets $u, v \in U_i$, if $u > v$ then $\exists Y$ such that Y is a child of v and $u \subseteq D(Y)$.

The bridge game can be treated as four player game as mentioned before. It can also be treated as two player game. That is, a player plays two hands of the cards and has to forget what he knows about one hand when he plays the other hand. Bridge treated as a four player game is perfect recall for each player, treated as a two player game is not perfect recall.

2.2 Strategies and Payoffs

To determine payoffs for each player after a play in a game, one might naturally let the players play the game and observe the final outcome. There is another way to calculate payoffs for each player in a game even without the players aid during a game. To accomplish this, each player announces in advance his decisions in every possible situation which might occur in a game. An umpire then collects all those decisions and determines the payoff for each player in the game. Those decisions are called pure strategies.

Definition 2.2.1 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game. A pure strategy for player i is a function π_i : player i moves of $K \rightarrow$ nodes of K . If player i chooses the child $y \in K$ at move $x \in K$ then $\pi_i(x) = y$. We extend the definition of π_i to

sets of moves as follows. Let $X \subseteq u \in U_i$ and we define $\pi_i(X) = \{\pi_i(x) \mid x \in X\}$. Furthermore, a pure strategy n -tuple π is $\pi = (\pi_0, \dots, \pi_n)$.

Since each player announces his pure strategies in advance, the umpire does not have to consider the entire game tree to assign the payoff for each player. Only a certain set of nodes can be reached by using a pure strategy n -tuple π in a general n -player game. The node in this set are said to be possible under π .

Definition 2.2.2 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game and π a pure strategy n -tuple for Γ . The set possible under π denoted $Poss \pi$ is defined recursively by:

- the root is in $Poss \pi$.
- if x is in $Poss \pi$ and x is a chance node then each child of x is in $Poss \pi$.
- if x is in $Poss \pi$ and x is a personal move then $\pi(x)$ is in $Poss \pi$.

Consider a game with no chance moves such as the chess game. A pure strategy n -tuple π determines a single play and only one leaf is in $Poss \pi$. The payoff for each player from the pure strategy n -tuple π is determined by the payoff function at that leaf.

If a game has chance moves, for a pure strategy n -tuple π , $Poss \pi$ may contains several leaves. The payoff for each player from the pure strategy π are determined by the expected payoff for those leaves contained in $Poss \pi$.

Definition 2.2.3 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game and π a pure strategy n -tuple for Γ . Let w be a leaf in K , and $\bar{w}$ be a play from root of K to w . Then the probability of w , denoted $p(w)$, is

$$p(w) = \prod_{\substack{x, y \in \bar{w} \\ x \text{ a chance move}}} p(x, y)$$

The probability of w under π , denoted $p_\pi(w)$, is

$$p_\pi(w) = \begin{cases} p(w) & \text{if } w \in \text{Poss } \pi \\ 0 & \text{otherwise} \end{cases}$$

Definition 2.2.4 The expected payoff $H_i(\pi)$ to player i is defined by

$$H_i(\pi) = \sum_{w \in W} p_\pi(w) h_i(w)$$

2.3 Equilibrium Point in a Game

Assume that all players predetermine their pure strategies π . If player i has options to change his pure strategy under the assumption that other players follow their predetermined pure strategies, then player i may be able to replace his pure strategy by a better one in order to increase his payoff. The following defines this concept.

Definition 2.3.1 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game and π be an n -tuple pure strategy for Γ . The strategy $\pi \mid \rho_i$ is the same as π except that ρ_i replaces π_i .

Occasionally, a pure strategy n -tuple π in an n -player game forms a situation in which player i can not increase his payoff by replacing his pure strategy n -tuple π_i with any other pure strategy. If this happens to every player then no player is motivated to change his pure strategy. This is called an equilibrium point.

Definition 2.3.2 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game and π be an n -tuple pure strategy for Γ . Strategy π is called an equilibrium point if for $i = 1, \dots, n$

$$H_i(\pi) \geq H_i(\pi \mid \rho_i)$$

for all pure strategies ρ_i .

It is well known that an n -player game with perfect information always has an equilibrium point in pure strategies[Kuhn, 1953]. An imperfect information game may or may not have equilibrium points under pure strategies. In a game without equilibrium points under pure strategies, players may choose not one pure strategy but several pure strategies randomly. This is called a mixed strategy or behavior strategy. We will not discuss these two types of strategies in this work. Interested readers are referred to Kuhn [1953].

Chapter 3

An Algorithm for Imperfect Information Games

In this chapter we define a *corresponding one person game* in which every player except one announces his pure strategy before the game starts. In this game, the remaining player determines his optimal strategy under the assumption that the other players use their announced pure strategies. Furthermore, we assume that this player has perfect recall. A recursive algorithm to find an optimal pure strategy for the corresponding one person game is developed here. We then modify this algorithm to check whether a pure strategy in an n -player game with perfect recall is an equilibrium point.

3.1 Corresponding One Person Game

In a general n -player game Γ , if every player predetermines a pure strategy n -tuple π and every player uses his predetermined pure strategy except one player, say i , this game corresponds to a one person game. This game is called the corresponding one person game with respect to player i and pure strategy π . When the player i and pure strategy π are clear from the context, we refer simply to a corresponding one person game.

In a corresponding one person game with respect to player i and n -tuple

pure strategy π , only player i has decisions to be made. From these characteristics we next define the successor of moves.

Definition 3.1.1 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game and π a pure strategy n -tuple for Γ . For $x \in K$, the successor of x with respect to player i and n -tuple π , denoted $s_{i,\pi}(x)$, is given by

$$s_{i,\pi}(x) = \begin{cases} \{x\} & \text{if } x \text{ is a leaf} \\ \{x\} & \text{if } x \text{ is a player } i \text{ node} \\ \{y \mid y \text{ is a child of } x\} & \text{if } x \text{ is a chance node} \\ \{\pi(x)\} & \text{if } x \text{ is a player } j \text{ node, where } j \neq i \end{cases}$$

We next define the successor of a set of moves in this game.

Definition 3.1.2 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game and π a pure strategy n -tuple for Γ . For any set X of nodes in K , the successor of X with respect to player i and n -tuple π , denoted $S_{i,\pi}(X)$, is given by

$$S_{i,\pi}(X) = \bigcup_{x \in X} s_{i,\pi}(x)$$

Where player i and pure strategy n -tuple π are clear from context, we sometimes refer simply to the successor of x or X .

3.2 Frontier Set

In any general game, decisions are made at all information sets. A player attempts to choose a child at each information set so that this child hopefully leads the player to a winning position or largest payoff value. In a perfect information game there is only one move in every information set u and then each child of u contains always one node. But in an imperfect information game, a child of an information set could contain leaves, nodes from a single information set, nodes from several information sets, or any combination of the above. We next introduce the concept of frontier sets.

Definition 3.2.1 A frontier set X in a game tree K is a set of nodes in K such that no two nodes in X lie on the same play.

A frontier set in the corresponding one person game with respect to player i and pure strategy π has the following properties.

Definition 3.2.2 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game and π a pure strategy n -tuple for Γ . A frontier set X , with respect to player i and n -tuple π , is said to be:

- singleton, with information set u , if $u \in U_i$ and $X \subseteq u$. [That is, all members of X are in a single information set for player i .]
- reduced if X is not singleton and $X = S_{i,\pi}(X)$.
- unreduced otherwise.

Where player i and pure strategy n -tuple π are clear from context, we sometimes say simply that X is singleton, reduced, or unreduced. It is clear that:

- A set containing only leaves is reduced.
- An information set for player i , or any subset of an information set for player i , is singleton.
- Any set containing chance moves or player j moves is unreduced.

Definition 3.2.3 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be a game and X a frontier set in K . The information set partition of X , denoted info-sets (X) , is a partition of the personal moves of X into its information sets. That is, info-sets (X) is the set $\{X_1, \dots, X_k\}$ such that:

- $\bigcup_{i=1}^k X_i = X - \{\text{leaves in } X\} - \{\text{chance nodes in } X\}$. [That is, info-sets (X) consists of sets that together include all and only the non-leaves and non-chance nodes in X .]

- If $x \in X_i$ and $y \in X_j$, where $i \neq j$, then x and y belong to different information sets in Γ .
- If $x \in X_i$ and $y \in X_i$, then x and y belong to the same information set in Γ .

3.3 An Evaluation Function for Frontier Set

In a corresponding one person game, player i chooses a child at each information set and hopefully the child will lead to a winning position. It will be convenient to define an evaluation function and assign a value to a child. Since a child is a frontier set in a corresponding one person game, next we define an evaluation function $H_{i,\pi} : \text{subsets of nodes of } K \rightarrow R$ for frontier sets.

Lemma 3.3.1 *Let X be a frontier set.*

- If X is singleton, then every child of X is a frontier set.
- If X is unreduced with respect to player i and pure-strategy n -tuple π , then $S_{i,\pi}(X)$ is a frontier set.
- If X is reduced with respect to player i and pure-strategy n -tuple π , then every Y in info-sets (X) is a frontier set.

Proof: Follows directly from the definitions. $\square$

Definition 3.3.1 *Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game. Let π be any pure-strategy n -tuple for Γ . Let X be any frontier set in K . Then the payoff for player i from X given π , denoted $H_{i,\pi}(X)$, is given recursively by*

$$H_{i,\pi}(X) = \begin{cases} H_{i,\pi}(\pi_i(X)) & \text{if } X \text{ is singleton} \\ H_{i,\pi}(S_{i,\pi}(X)) & \text{if } X \text{ is unreduced} \\ \sum_{\substack{x \in X \\ x \text{ a leaf}}} p(x)h_i(x) + \sum_{Y \in \text{info-sets}(X)} H_{i,\pi}(Y) & \text{if } X \text{ is reduced} \end{cases}$$

where each condition in the above is with respect to player i and n -tuple π .

The above recursive evaluation function $H_{i,\pi}$ of a corresponding one person game is closely related to the expected payoff function H_i in a general n -player game. To make this point clear, we next define an alternative evaluation function $H_{i,\pi}^{alt}$: subsets of nodes of $K \rightarrow R$.

Definition 3.3.2 Recall that W denotes the set of leaves in a game tree K . Let X be a frontier set in K . Then:

$$W_X = \{w \in W \mid \exists x \in X \text{ such that } w \text{ is descendant of } x\}$$

Definition 3.3.3 Let $\Gamma = (K, \mathcal{U}, \mathcal{P}, h, p)$ be an n -player game. Let π be any pure strategy n -tuple for Γ . Let X be any frontier set in K . Then the payoff for player i from X given π , denoted $H_{i,\pi}^{alt}(X)$, is given by

$$H_{i,\pi}^{alt}(X) = \sum_{w \in W_X} p_\pi(w) h_i(w)$$

Unfortunately, the expected payoff function for player i in a general n -player game, $H_i(\pi)$, does not equal to the evaluation function for the frontier set X , $H_{i,\pi}(X)$, or the alternative evaluation function, $H_{i,\pi}^{alt}(X)$, defined above. The following definitions and lemmas relate these quantities under certain conditions.

In a general n -player game, a pure strategy n -tuple π defines a set of moves which can be reached by the pure strategy π and is called possible under π . We extend the same concept in a corresponding one person game and call it possible under π for player i .

Definition 3.3.4 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game, and π a pure strategy n -tuple for Γ . The set possible under π for player i denoted $Poss_i \pi$ is defined recursively as follows:

- the root of K is in $Poss_i \pi$.

- if x is in $Poss_i \pi$ and x is a chance node or a personal move for player i then each child of x is in $Poss_i \pi$.
- if x is in $Poss_i \pi$ and x is a personal move for player j and $j \neq i$ then $\pi_j(x)$ is in $Poss_i \pi$.

An information set $u \in U_i$ is relevant for player i when the other players use π iff there exists an x in u such that $x \in Poss_i \pi$.

We need the following lemmas to show that for a player i and pure strategy n -tuple π , the payoffs of a frontier set X in a game tree K from the evaluation function $H_{i,\pi}$ and the alternative evaluation function $H_{i,\pi}^{alt}$ are the same if every node in X is possible under π for player i .

Lemma 3.3.2 For a player i and a pure strategy n -tuple π in an n -player game Γ , for a frontier set $X \in K$,

- if $X \subseteq Poss \pi$ then $S_{i,\pi}(X) \subseteq Poss \pi$.
- if $X \subseteq Poss_i \pi$ then $S_{i,\pi}(X) \subseteq Poss_i \pi$.

Proof: Since $S_{i,\pi}(X) = \cup_{x \in X} s_{i,\pi}(x)$, we need to show that $\forall x$, if $x \in Poss \pi$ then $s_{i,\pi}(x) \subseteq Poss \pi$.

- Case a: x is a leaf or player i node. Since here $s_{i,\pi}(x) = \{x\}$, we have $s_{i,\pi}(x) \subseteq Poss \pi$.
- Case b: x is a chance node. Since here $s_{i,\pi}(x) = \{y \mid y \text{ is a child of } x\}$, we have $s_{i,\pi}(x) \subseteq Poss \pi$, by definition of $Poss \pi$.
- Case c: x is player j node where $j \neq i$. Since here $s_{i,\pi}(x) = \{\pi_j(x)\}$, we have $s_{i,\pi}(x) \subseteq Poss \pi$ by definition of $Poss \pi$.

We can use the same logic to prove $Poss_i \pi$. $\square$

The above lemma states that if a frontier set is possible then all successors of the frontier set are also possible. Now we can prove the lemma that $H_{i,\pi}(X) = H_{i,\pi}^{alt}(X)$.

Lemma 3.3.3 *Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game and π a pure strategy n -tuple for Γ . For a frontier set $X \subseteq \text{nodes of } K$, if $X \subseteq Poss \pi$ then*

$$H_{i,\pi}(X) = H_{i,\pi}^{alt}(X)$$

Proof: This proof is an induction on the maximum depth of the recursion in the calculation of $H_{i,\pi}(X)$.

Basis step: Let the maximum depth of the recursion in the calculation of $H_{i,\pi}(X)$ be 0. Then frontier set X has to be reduced and $\text{info-set}(X) = \emptyset$. Also if $x \in X$ then x is a leaf.

$$\begin{aligned} H_{i,\pi}(X) &= \sum_{\substack{x \in X \\ x \text{ a leaf}}} p(x)h_i(x) && \text{by definition of } H_{i,\pi} \\ &= \sum_{x \in W_X} p(x)h_i(x) && \text{by definition of } W_X \\ &= \sum_{x \in W_X} p_\pi(x)h_i(x) && \text{because } X \subseteq Poss \pi \end{aligned}$$

Inductive step: Assume that for any frontier set X for which the maximum depth of the recursion in the calculation of $H_{i,\pi}(X)$ is less than k , $k \geq 1$,

$$H_{i,\pi}(X) = H_{i,\pi}^{alt}(X) = \sum_{x \in W_X} p_\pi(w)h_i(w)$$

Consider the frontier set X for which the maximum depth of the recursion in the calculation of $H_{i,\pi}(X)$ is k .

Case i: X is singleton.

$$\begin{aligned}
H_{i,\pi}(X) &= H_{i,\pi}(\pi_i(X)) && \text{by definition of } H_{i,\pi} \\
&= \sum_{w \in W_{\pi_i(X)}} p_\pi(w) h_i(w) && \text{by induction hypothesis} \\
&&& \text{(applicable because } X \subseteq \text{Poss } \pi \\
&&& \text{implies } \pi_i(X) \subseteq \text{Poss } \pi) \\
&= \sum_{w \in W_{\pi_i(X)}} p_\pi(w) h_i(w) \\
&+ \sum_{w \in W_X - W_{\pi_i(X)}} p_\pi(w) h_i(w) && \text{because } p_\pi(w) = 0 \ \forall w \in W_X - W_{\pi_i(X)} \\
&= \sum_{w \in W_X} p_\pi(w) h_i(w) \\
&= H_{i,\pi}^{alt}(X) && \text{by the definition of } H_{i,\pi}^{alt}
\end{aligned}$$

Case ii: X is unreduced.

$$\begin{aligned}
H_{i,\pi}(X) &= H_{i,\pi}(S_{i,\pi}(X)) && \text{by definition of } H_{i,\pi} \\
&= \sum_{x \in W_{S_{i,\pi}(X)}} p_\pi(x) h_i(x) && \text{induction hypothesis} \\
&&& \text{(applicable because by lemma 3.3.2} \\
&&& S_{i,\pi}(X) \subseteq \text{Poss } \pi \text{ and} \\
&&& \text{the maximum depth of the recursion in} \\
&&& \text{the calculation of } H_{i,\pi}(S_{i,\pi}(X)) \\
&&& \text{is less than } k) \\
&= \sum_{w \in W_{S_{i,\pi}(X)}} p_\pi(w) h_i(w) \\
&+ \sum_{w \in W_X - W_{S_{i,\pi}(X)}} p_\pi(w) h_i(w) && \text{because } p_\pi(w) = 0 \ \forall w \in W_X - W_{S_{i,\pi}(X)} \\
&= \sum_{x \in X} H_{i,\pi}(S_{i,\pi}(X)) \\
&= H_{i,\pi}^{alt}(X) && \text{by the definition of } H_{i,\pi}^{alt}
\end{aligned}$$

Case iii: X is reduced. Let

$$\begin{aligned}
X_l &= \{x \in X \mid x \text{ is a leaf}\} \\
X_{nl} &= \{x \in X \mid x \text{ is not a leaf}\}
\end{aligned}$$

$$H_{i,\pi}(X) = \sum_{\substack{x \in X \\ x \text{ a leaf}}} p(x) h_i(x) + \sum_{Y \in \text{info-sets}(X)} H_{i,\pi}(Y)$$

The first term,

$$\begin{aligned}
\sum_{\substack{x \in X \\ x \text{ a leaf}}} p(x) h_i(x) &= \sum_{x \in X_l} p(x) h_i(x) && \text{a leaf is his own root} \\
&= \sum_{w \in W_{X_l}} p(w) h_i(w) && \text{by definition of } W_X \\
&= \sum_{w \in W_{X_l}} p_\pi(w) h_i(w) && \text{because } X \subseteq \text{Poss } \pi
\end{aligned}$$

By the definition of *info-set*, if $X \subseteq \text{Poss } \pi$ then $\forall Y \in \text{info-set}(X)$, we have

$Y \subseteq Poss \pi$. Since the maximum depth of the recursion in the calculation of $H_{i,\pi}(Y)$ is less than k , we can apply the induction hypothesis on the second term,

$$\begin{aligned} H_{i,\pi}(Y) &= \sum_{w \in W_Y} p_\pi(w) h_i(w) \\ \sum_{Y \in info-set(X)} H_{i,\pi}(Y) &= \sum_{Y \in info-set(X)} \sum_{w \in W_Y} p_\pi(w) h_i(w) \\ &= \sum_{w \in W_{X_{nl}}} p_\pi(w) h_i(w) \end{aligned}$$

Therefore,

$$\begin{aligned} H_{i,\pi}(X) &= \sum_{w \in W_{X_i}} p_\pi(w) h_i(w) + \sum_{w \in W_{X_{nl}}} p_\pi(w) h_i(w) \\ &= \sum_{w \in W_X} p_\pi(w) h_i(w) & W_X = W_{X_i} \cup W_{X_{nl}} \\ &= H_{i,\pi}^{alt}(X) & \text{by definition of } H_{i,\pi}^{alt}(X) \end{aligned}$$

□

Finally, we can show that in a general n -player game, the payoff for player i from the root given pure strategy n -tuple π is equivalent to the value of the game for player i using the pure strategy π .

Lemma 3.3.4 *For any pure strategy n -tuple π in an n -player game Γ with game tree K ,*

$$H_{i,\pi}(\{\text{root of } K\}) = H_i(\pi)$$

Proof: This is a special case of lemma 3.3.3. Let x_0 denote the root of K and W denote the leaves of K , then $W_{\{x_0\}} = W$.

$$H_{i,\pi}(\{x_0\}) = \sum_{w \in W} p_\pi(w) h_i(w) = H_i(\pi)$$

□

3.4 Exactness and Recursive Exactness

In a general n -player game, a player makes decisions at each information set based in part on the payoffs of children of an information set. A personal information set may contain several moves and every move has a child. Since a player can not distinguish the difference between moves in an information set, the player

should consider all children of the information set. The payoff of a child for the information set is an expected value among those children. Unfortunately, not all moves in an information set are possible under π for player i . That is several moves in the information set are irrelevant to player i . We define terms exact and recursively exact for a frontier set with moves which are only relevant for player i and a pure strategy n -tuple π .

The difficulty in handling an imperfect information game is brought about by the possible moves in an information set depending on the other players' pure strategy π . A different pure strategy π' generates a different set of possible moves in the information set. We define the term u^π to be the set of possible moves from information set u under pure strategy π .

Definition 3.4.1 *Let u be an information set for player i in an n -player game Γ and π be a pure strategy n -tuple for game Γ . We define*

$$u^\pi = u \cap \text{Poss}_i \pi$$

If we know a pure strategy π , then we can locate the moves which are possible under π in an information set. We have the following definition.

Definition 3.4.2 *Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game. Let π be a pure-strategy n -tuple for Γ . Let X be a frontier set in K . We say that X , with respect to player i and n -tuple π , is exact if:*

- *Case 1: X is singleton, with associated information set u . Here we require that $X = u^\pi$*
- *Case 2: X is unreduced. Here we require that $S_{i,\pi}(X)$ is exact with respect to player i and n -tuple π .*
- *Case 3: X is reduced. Here we require that each Y in info-sets (X) is exact with respect to player i and n -tuple π .*

If a frontier set X is exact under π for player i and also the children of X are exact, then we can define a recursive property for exact. The recursively exact is a global version of exact.

Definition 3.4.3 *For any frontier set X and pure strategy n -tuple π in an n -player game Γ , we say that X , with respect to player i and n -tuple π , is recursively exact if:*

- *Case 1: X is singleton, with associated information set u . Here we require that:*
 - *X is exact (i.e. $X = u^\pi$), and*
 - *each child Y of X is recursively exact.*
- *Case 2: X is unreduced. Here we require that $S_{i,\pi}(X)$ is recursively exact.*
- *Case 3: X is reduced. Here we require that each Y in info-sets (X) is recursively exact.*

where the phrases singleton, unreduced, reduced, exact, and recursively exact are all with respect to player i and n -tuple π .

Lemma 3.4.1 *Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game. Let π be a pure-strategy n -tuple for Γ . Let X be a frontier set in K and Y be a child of X . If Γ has perfect recall for player i , and X is singleton and exact then Y is exact.*

Proof. If X is a frontier set and Y is a child of X then Y is also a frontier set by the lemma 3.3.1. Since X is singleton and exact then $\exists u \in U_i$ such that $X \subseteq u$ and $X = u^\pi$. By the definition of $Poss_i \pi$ and the fact that X is a subset of player i information set, we know that $Y \subseteq Poss_i \pi$. Let $Y^0 = Y$ and $Y^n = S_{i,\pi}(Y^{n-1})$, $n \geq 1$. Since the game is finite then $\exists k$ such that Y^k is reduced or singleton and

$Y^{k-1}, \dots, Y^0$ are unreduced. Since $Y^0 \subseteq \text{Poss}_i \pi$, from lemma 3.3.2 $Y^n \subseteq \text{Poss}_i \pi$ for all $1 \leq n \leq k$.

We need to show that for any $Z \in \text{info-set}(Y^k)$, Z is exact. Since Z is singleton by the definition of *info-set*, we know that $\exists v \in U_i$ such that $Z \subseteq v$. We need to show $Z = v^\pi$. We already showed that $Z \subseteq \text{Poss}_i \pi$ (since $Y^k \subseteq \text{Poss}_i \pi$). Next we need to show that $v^\pi \subseteq Z$.

Since Y^0 is a child of X and since Z is not empty, there is an element of Z that is a descendent of an element of X , and hence, $v > u$. By perfect recall, $\exists$ child Y' such that $v \subseteq D(Y')$. For any $y \in v^\pi$, we have $y \in D(Y')$. Since Y' is a child of u , and $y \in D(Y')$, $\exists$ an ancestor x of y such that $x \in u$. Since $y \in \text{Poss}_i \pi$, we have $x \in \text{Poss}_i \pi$. That is, $x \in u^\pi$. Since X is exact, $X = u^\pi$, that is, $x \in X$.

We have $x \in X$, and from x some child $x' \in Y^0$ such that y is a descendent of x' . (It is possible that $y = x'$.) Also, y is a player i move. If every node on path from x' to y , except y itself, is a chance node or a player j node ($j \neq i$), then $y \in Y^k$. In that case, since $y \in v$ and $Z = Y^k \cap v$, we have $y \in Z$ and have finished the proof.

All that remains is to rule out the possibility of a player i node other than y itself being on the path from x' to y . Suppose there were such a node, call it y' , contained in an information set v' . Since Z is nonempty, it contains a node, call it z . Since $z \in Z$, it is also in Y^k , hence there is a path from some node in X to Z such that z is the first player i node on the path (after X). Since there is a path from the node y' in v' to y , and $y \in v$, we have $v > v'$. By definition of perfect recall, $z \in D(v')$. But z is the first player i node on its path from X , and v' is a player i information set, so z must be in v' itself. But the information set containing Z is (by definition) v , so $v' = v$. But $y' \in v'$, and $y \in v$, y follows y' but $y \neq y'$, and since $v = v'$, we have two nodes on same play in a single information

set. This violates the definition of an information set. $\square$

Intuitively, if we need to construct the recursive relationship of exact, we need a game with perfect recall. With perfect recall a player can make decisions based on the current status. But without perfect recall the player has to guess his previous moves when he makes decisions on the current status. The following lemma explores this concept.

Lemma 3.4.2 *For any frontier set X and pure strategy n -tuple π in an n -player game Γ , if Γ has perfect recall for player i , then X is exact if and only if X is recursively exact.*

Proof: Sufficiency. The proof is an induction on the maximum depth of the recursion in the calculation of $S_{i,\pi}(X)$. The limitation of perfect recall is needed only for necessity.

Basis step. Let the maximum depth of the recursion in the calculation of $S_{i,\pi}(X)$ be 0. Then frontier set X must be reduced and *info-set* (X) is the empty set. Also if $x \in X$ then x is a leaf. X is exact by the definition of exactness.

Inductive step. Assume that for any frontier set X for which the maximum depth of the recursion in calculation of $S_{i,\pi}(X)$ is less than k , $k \geq 1$, then the following is true. If X is recursively exact then X is exact.

Consider the frontier set X for which the maximum depth of the recursion in the calculation of $S_{i,\pi}(X)$ is k and X is recursively exact.

- Case i: X is singleton. X is exact by definition of recursive exactness.
- Case ii: X is unreduced. We know that $S_{i,\pi}(X)$ is recursively exact. We have X has maximum depth of the recursion in calculation of $S_{i,\pi}(X)$ less than k . We can apply the induction hypothesis that $S_{i,\pi}(X)$ is exact. Then X is exact by the definition of exactness.

- Case iii: X is reduced. We know that each Y in *info-set* (X) is recursively exact. We have Y has maximum depth of the recursion in calculation of $S_{i,\pi}(Y)$ less than k . We can apply the induction hypothesis that $S_{i,\pi}(Y)$ is exact. Then X is exact by the definition of exactness.

Necessity. The proof is an induction on the maximum depth of recursion in the calculation of $S_{i,\pi}(X)$.

Basis step: Let the maximum depth of the recursion in the calculation of $S_{i,\pi}(X)$ be 0. Then frontier set X must be reduced and *info-set* (X) is the empty set. Also if $x \in X$ then x is a leaf. X is recursively exact by definition of recursive exactness.

Inductive step. Assume that for any frontier set X for which the maximum depth of the recursion in calculation of $S_{i,\pi}(X)$ is less than k , $k \geq 1$, then the following is true. If X is exact then X is recursively exact.

Consider the frontier set X for which the maximum depth of recursion in the calculation of $S_{i,\pi}(X)$ is k , and X is exact.

- Case i: X is singleton. Let Y be any child of X . Since player i has perfect recall, X is singleton, and X is exact, then from lemma 3.4.1 Y is exact. Since Y is exact and Y has the maximum depth of recursion in calculation of $S_{i,\pi}(Y)$ less than k then we can apply the induction hypothesis that Y is recursively exact. Since X is exact and each child Y of X is recursively exact then X is recursively exact by the definition of recursively exact.
- Case ii: X is unreduced. We know that $S_{i,\pi}(X)$ is exact. Since $S_{i,\pi}(X)$ has the maximum depth of recursion in calculation of $S_{i,\pi}(S_{i,\pi}(X))$ less than k then we can apply the induction hypothesis that $S_{i,\pi}(X)$ is recursively exact, so X is recursively exact.

- Case iii: X is reduced. We know that each Y in $\text{info-set}(X)$ is exact. Since Y is singleton then from the same logic of Case 1, Y is recursively exact. Therefore, X is recursively exact by the definition of recursively exact.

□

Finally, we can prove that in any n -player game, the root of the game tree is recursively exact. The proof is essentially the same as the proof of lemma 3.4.1.

Lemma 3.4.3 *For any pure strategy n -tuple π in any n -player game Γ with game tree K , if Γ has perfect recall for player i , then $\{\text{root of } K\}$ is recursively exact.*

Proof: Let x_0 denote the root of the game tree K and $X = \{x_0\}$. Since Γ has perfect recall, from lemma 3.4.2 we only need to show that X is exact.

- Case i: X is singleton. x_0 is a personal move. Let u denote the personal information set which contains x_0 . Then u contains only one move because there is only one move in one play in an information set by the definition of games. Since $x_0 \in \text{Poss}_i \pi$ then $X = u^\pi$. Hence x_0 is exact by definition of exactness.

- Case ii: X is unreduced.

Let $X^0 = X$ and $X^n = S_{i,\pi}(X^{n-1})$, $n \geq 1$. Since the game is finite then $\exists k$ such that X^k is reduced or singleton and $X^{k-1}, \dots, X^0$ are unreduced. Since $X^0 \subseteq \text{Poss}_i \pi$, from lemma 3.3.2 $X^n \subseteq \text{Poss}_i \pi$ for all $1 \leq n \leq k$.

We need to show that for any $Z \in \text{info-set}(X^k)$, Z is exact. Since Z is singleton by the definition of info-set , we know that $\exists v \in U_i$ such that $Z \subseteq v$. We need to show $Z = v^\pi$. We already showed that $Z \subseteq \text{Poss}_i \pi$ (since $X^k \subseteq \text{Poss}_i \pi$). Next we need to show that $v^\pi \subseteq Z$.

For any $y \in v^\pi$, if there is no player i node except y itself on the path from x_0 to y , then $y \in Z$ and we are done.

All that remains is to rule out the possibility of a player i node other than y itself on the path from x_0 to y . Suppose there were such a node, call it y' , contained in an information set v' . Since Z is nonempty, it contains a node, call it z . Since $z \in Z$, it is also in X^k , hence there is a path from the root x_0 to $z \in Z$ such that z is the first player i node on the path (after x_0). Since there is a path from the node $y' \in v'$ to y , and $y \in v$, we have $v > v'$. By definition of perfect recall, $z \in D(v')$. But z is the first player i node on its path from X , and v' is a player i information set, so z must be in v' itself. But the information set containing Z is (by definition) v , so $v' = v$. But $y' \in v'$, and $y \in v$, y follows y' but $y \neq y'$, and since $v = v'$, we have two nodes on same play in a single information set. This violates the definition of an information set.

- Case iii: X is reduced. x_0 is a leaf and x_0 is exact by the definition of exactness.

□

3.5 An Algorithm For the Corresponding One Person Game

In this section we propose a recursive algorithm to find a pure strategy for player i in a corresponding one person game under π . We prove that if the game has perfect recall then the pure strategy obtained from the algorithm is optimal.

The following two definitions describe the recursive algorithm.

Definition 3.5.1 *Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game. Let π be any pure strategy n -tuple for Γ . Let X be any frontier set in K . Then the optimal payoff for*

player i from X given π , denoted $V_{i,\pi}(X)$, is given recursively by

$$V_{i,\pi}(X) = \begin{cases} \max\{V_{i,\pi}(Y) \mid Y \text{ is a child of } X\} & \text{if } X \text{ is singleton} \\ V_{i,\pi}(S_{i,\pi}(X)) & \text{if } X \text{ is unreduced} \\ \sum_{\substack{x \in X \\ x \text{ a leaf}}} p(x)h(x) + \sum_{Y \in \text{info-sets}(X)} V_{i,\pi}(Y) & \text{if } X \text{ is reduced} \end{cases}$$

where each condition in the above is with respect to player i and n -tuple π .

Definition 3.5.2 Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game. Let π be any pure strategy n -tuple for Γ . An optimal strategy for player i given π , denoted $\rho_{i,\pi}^*$, is as follows. Let $u \in \mathcal{U}_i$ be any information set that is relevant to player i when the other players use π . Let $X = u^\pi$. Then $\rho_{i,\pi}^*(u)$ is any child Y of X that maximizes $V_{i,\pi}(Y)$.

The following lemma proves that the algorithm defined above and the evaluation function defined in the previous section generate the same payoffs for a frontier set if the game has perfect recall and the frontier set is exact.

Lemma 3.5.1 For any frontier set X and pure strategy n -tuple π in an n -player game Γ with perfect recall, if X is exact, then

$$\begin{aligned} H_{i,\pi|\rho_{i,\pi}^*}(X) &= V_{i,\pi|\rho_{i,\pi}^*}(X) \\ &= V_{i,\pi}(X) \end{aligned}$$

Proof. To show that

$$V_{i,\pi|\rho_{i,\pi}^*}(X) = V_{i,\pi}(X)$$

is trivial because $V_{i,\pi}$ is independent of the i^{th} component of π .

The proof of

$$H_{i,\pi|\rho_{i,\pi}^*}(X) = V_{i,\pi|\rho_{i,\pi}^*}(X)$$

is an induction on the maximum depth of the recursion in the calculation of $H_{i,\pi|\rho_{i,\pi}^*}(X)$.

Basis step. Let the maximum depth of the recursion in the calculation of $H_{i,\pi|\rho_{i,\pi}^*}(X)$ be 0. Then frontier set X must be reduced and $\text{info-set}(X)$ is $\emptyset$. Also if $x \in X$ then x is a leaf. Therefore,

$$H_{i,\pi|\rho_{i,\pi}^*}(X) = V_{i,\pi|\rho_{i,\pi}^*}(X)$$

by definition of $H_{i,\pi}$ and $V_{i,\pi}$.

Inductive step: Assume that for any frontier set X for which the maximum depth of the recursion in the calculation of $H_{i,\pi|\rho_{i,\pi}^*}(X)$ is less than k , the following is true.

$$H_{i,\pi|\rho_{i,\pi}^*}(X) = V_{i,\pi|\rho_{i,\pi}^*}(X)$$

Consider a frontier set X for which the maximum depth of the recursion in the calculation of $H_{i,\pi|\rho_{i,\pi}^*}$ is k .

Case i: X is singleton. Since X is exact and Γ has perfect recall then from lemma 3.4.2 X is recursively exact. $\rho_{i,\pi}^*(X)$ is recursively exact by the definition of recursively exactness. Then $\rho_{i,\pi}^*(X)$ is exact from the lemma 3.4.2.

$$\begin{aligned} H_{i,\pi|\rho_{i,\pi}^*}(X) &= H_{i,\pi|\rho_{i,\pi}^*}(\rho_{i,\pi}^*(X)) && \text{definition of } H_{i,\pi} \\ &= V_{i,\pi|\rho_{i,\pi}^*}(\rho_{i,\pi}^*(X)) && \text{induction hypothesis} \\ &= \max\{V_{i,\pi|\rho_{i,\pi}^*}(Y) \mid Y \text{ is a child of } X\} && \text{definition of } \rho_{i,\pi}^* \\ &&& \text{and } X \text{ is exact} \\ &= V_{i,\pi|\rho_{i,\pi}^*}(X) && \text{definition of } V_{i,\pi} \end{aligned}$$

Case ii: X is unreduced. Since Γ has perfect recall and X is exact then $S_{i,\pi}(X)$ is exact by lemma 3.4.1.

$$\begin{aligned} H_{i,\pi|\rho_{i,\pi}^*}(X) &= H_{i,\pi|\rho_{i,\pi}^*}(S_{i,\pi}(X)) && \text{definition of } H_{i,\pi} \\ &= V_{i,\pi|\rho_{i,\pi}^*}(S_{i,\pi}(X)) && \text{induction hypothesis} \\ &= V_{i,\pi|\rho_{i,\pi}^*}(X) && \text{definition of } V_{i,\pi} \end{aligned}$$

Case iii: X is reduced. Since Γ has perfect recall and X is exact then each Y in

$info-set(X)$ is exact from the lemma 3.4.1.

$$\begin{aligned}
H_{i,\pi|\rho_{i,\pi}^*}(X) &= \sum_{\substack{x \in X \\ x \text{ a leaf}}} p(x)h_i(x) + \sum_{Y \in info-sets(X)} H_{i,\pi|\rho_{i,\pi}^*}(Y) && \text{definition of } H_{i,\pi} \\
&= \sum_{\substack{x \in X \\ x \text{ a leaf}}} p(x)h_i(x) + \sum_{Y \in info-sets(X)} V_{i,\pi|\rho_{i,\pi}^*}(Y) && \text{induction hypothesis} \\
&= V_{i,\pi|\rho_{i,\pi}^*}(X) && \text{definition of } V_{i,\pi}
\end{aligned}$$

□

The following lemma proves that in a game with perfect recall if a frontier set X is exact then the algorithm $\rho_{i,\pi}^*(X)$ chooses the alternative which generates the best payoff among all children of X .

Lemma 3.5.2 *For any frontier set X and pure strategy n -tuple π in an n -player game Γ with perfect recall, and for any pure strategy ρ_i for player i , if X is exact, then*

$$H_{i,\pi|\rho_i}(X) \leq H_{i,\pi|\rho_{i,\pi}^*}(X)$$

Proof. This proof is an induction on the maximum depth of the recursion in the calculation of $H_{i,\pi|\rho_{i,\pi}^*}(X)$.

Basis step. Let the maximum depth of the recursion in the calculation of $H_{i,\pi|\rho_{i,\pi}^*}(X)$ be 0. Then frontier set X must be reduced and $info-set(X)$ is $\emptyset$. Also if $x \in X$ then x is a leaf. Therefore,

$$\begin{aligned}
H_{i,\pi|\rho_i}(X) &= \sum_{\substack{x \in X \\ x \text{ a leaf}}} p(x)h_i(x) \\
&= H_{i,\pi|\rho_{i,\pi}^*}(X)
\end{aligned}$$

Inductive step: Assume that for any frontier set X for which the maximum depth of the recursion in the calculation of $H_{i,\pi|\rho_{i,\pi}^*}(X)$ is less than k , the following is true.

$$H_{i,\pi|\rho_i}(X) \leq H_{i,\pi|\rho_{i,\pi}^*}(X)$$

Consider a frontier set X for which the maximum depth of the recursion in the calculation of $H_{i,\pi|\rho_i^*}$ is k .

Case i: X is singleton. Since Γ has perfect recall and X is exact then $\rho_i(X)$ is exact.

$$\begin{aligned}
H_{i,\pi|\rho_i}(X) &= H_{i,\pi|\rho_i}(\rho_i(X)) && \text{definition of } H_{i,\pi} \\
&\leq H_{i,\pi|\rho_{i,\pi}^*}(\rho_i(X)) && \text{induction hypothesis} \\
&= V_{i,\pi|\rho_i}(\rho_i(X)) && \text{lemma 3.5.1} \\
&\leq V_{i,\pi|\rho_{i,\pi}^*}(\rho_{i,\pi}^*(X)) && \text{definition of } \rho_{i,\pi}^* \text{ and } X \text{ is exact} \\
&= H_{i,\pi|\rho_{i,\pi}^*}(\rho_{i,\pi}^*(X)) && \text{lemma 3.5.1} \\
&= H_{i,\pi|\rho_{i,\pi}^*}(X) && \text{definition of } H_{i,\pi}
\end{aligned}$$

Case ii: X is unreduced. Since Γ has perfect recall and X is exact and $\rho_i(X)$ is a child of X then $\rho_i(X)$ is exact.

$$\begin{aligned}
H_{i,\pi|\rho_i}(X) &= H_{i,\pi}(S_{i,\pi}(X)) && \text{definition of } H_{i,\pi} \\
&\leq H_{i,\pi|\rho_{i,\pi}^*}(S_{i,\pi}(X)) && \text{induction hypothesis} \\
&= H_{i,\pi|\rho_{i,\pi}^*}(X) && \text{definition of } H_{i,\pi}
\end{aligned}$$

Case iii: X is reduced. Since Γ has perfect recall and X is exact then for each Y in $\text{info-set}(X)$, Y is exact.

$$\begin{aligned}
H_{i,\pi|\rho_i}(X) &= \sum_{\substack{x \in X \\ x \text{ a leaf}}} p(x)h_i(x) + \sum_{Y \in \text{info-sets}(X)} H_{i,\pi}(Y) && \text{definition of } H_{i,\pi} \\
&\leq \sum_{\substack{x \in X \\ x \text{ a leaf}}} p(x)h_i(x) + \sum_{Y \in \text{info-sets}(X)} H_{i,\pi|\rho_{i,\pi}^*}(Y) && \text{induction hypothesis} \\
&= H_{i,\pi|\rho_{i,\pi}^*}(X) && \text{definition of } H_{i,\pi}
\end{aligned}$$

□

Finally, we can prove that the algorithm finds the optimal pure strategy for player i given π in a corresponding one person game.

Theorem 3.5.1 *Let $\Gamma(K, \mathcal{U}, h, \mathcal{P})$ be an n -player game with perfect recall for player i . Let π be a pure strategy n -tuple for Γ . Then for any pure strategy ρ_i for player i ,*

$$H_i(\pi \mid \rho_i) \leq H_i(\pi \mid \rho_{i,\pi}^*) = V_{i,\pi}(\{\text{root of } K\})$$

Proof: Let x_0 be the root of K . From lemma 3.4.3 $\{x_0\}$ is exact.

$$\begin{aligned} V_{i,\pi}(\{x_0\}) &= H_{i,\pi|\rho_{i,\pi}^*}(\{x_0\}) && \text{lemma 3.5.1} \\ &= H_i(\pi|\rho_{i,\pi}^*) && \text{lemma 3.3.4} \end{aligned}$$

For all i and ρ_i ,

$$\begin{aligned} H_i(\pi|\rho_i) &= H_{i,\pi|\rho_i}(\{x_0\}) && \text{lemma 3.3.4} \\ &\leq H_{i,\pi|\rho_{i,\pi}^*}(\{x_0\}) && \text{lemma 3.5.2} \\ &= H_i(\pi|\rho_{i,\pi}^*) && \text{lemma 3.3.4} \end{aligned}$$

□

Corollary 3.5.1 *Let $\Gamma(K, \mathcal{U}, h, \mathcal{P})$ be an n -player game with perfect recall for player i . Let π be a pure strategy n -tuple for Γ . Then if $\pi_i = \rho_{i,\pi}^*$ for all player i then π is an equilibrium point.*

Proof.

Since $\rho_{i,\pi}^* = \pi_i$ for all i then for any ρ_i for player i ,

$$\begin{aligned} H_i(\pi) &= H_i(\pi|\pi_i) \\ &= H_i(\pi|\rho_{i,\pi}^*) \\ &\geq H_i(\pi|\rho_i) && \text{theorem 3.5.1} \end{aligned}$$

3.6 The Efficiency of the Algorithm

In this section we develop a recursive algorithm and also discuss the efficiency of the algorithm. The following algorithm is constructed from definitions 3.5.1 and 3.5.2.

Algorithm 3.6.1 */* algorithm V */*

```
int Optimal[number of information sets];
```

```
/* save the optimal alternative for each information set */
```

```
function V (X : frontier set)
```

```
{
```

```

double          maxChildValue;
double          curChildValue;
double          sum;
information set u;

switch (Property(X)):
{
case SINGLETON:
    maxChildValue = -LARGE;
    for each child Y of X do
    {
        curChildValue = V (Y);
        if (curChildValue > maxChildValue)
            maxChildValue = curChildValue;
    }
    u = InfoSet (X);
    Optimal[u] = Order (X, Y);
    return (maxChildValue);
    break;
case UNREDUCED:
    return (V (Succ (X)));
    break;
case REDUCED:
    sum = 0.0;
    for each leaf w in X do
        sum = sum + p(w)h(w);
    for each element Z in info-set (X)

```

```

        sum = sum + V (Z);
    return (sum);
    break;
}
}

function Order (X, Y : frontier set)
{
    if Y is ith child of X then return i.
}

function Succ (X : frontier set)
{
    return the successors of X.
}

function InfoSet (X: frontier set)
{
    return the information set which contains X.
}

function Property (X: frontier set)
{
    return the property of X which is
    one of the following, SINGLETON, UNREDUCED, or REDUCED.
}

```

See Appendix A for a trace on an example.

It is worth noting that the calculations of $\rho_{i,\pi}^*$ are different between definition 3.5.2 and algorithm 3.6.1. Let $\Gamma = (\mathcal{K}, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game with perfect recall, X be a frontier set of a game tree K , and π be a pure strategy of Γ . The optimal pure strategy for player i at any information set $u \in U_i$ in a corresponding one person game, $\rho_{i,\pi}^*(u)$, is any child Y of X that maximizes $V_{i,\pi}(Y)$ (definition 3.5.2). Hence the decision at an information set u is independent of its parents. But in algorithm 3.6.1, the strategy $\rho_{i,\pi}^*(u)$ is defined recursively starting with the root of the game tree K . Therefore the decision at an information set u is related with its parents. The following proposition proves that the two methods generate the same result as long as the tie is broken by the same way.

Proposition 3.6.1 *Let $\Gamma = (\mathcal{K}, \mathcal{P}, \mathcal{U}, h, p)$ be an n -player game with perfect recall. Let π be a pure strategy of Γ . Let $\rho_{i,\pi}^*(u)$ denote the pure strategy generated from definition 3.5.2 for player i at information set u when all other players use π where tie is broken in favor of leftmost child. Let $\rho_{i,\pi}'(u)$ denote the pure strategy generated from algorithm 3.6.1 for player i at information set u when all other players use π when children are presented left to right. Then for all $u \in U_i$, $\rho_{i,\pi}^*(u) = \rho_{i,\pi}'(u)$.*

Proof: For any information set u , by definition 3.5.2 there exists a frontier set X such that $X = u^\pi$ and $\rho_{i,\pi}^*(u)$ is some child Y of X that maximizes $V_{i,\pi}(Y)$. Let X' be a frontier set such that $X' \subseteq u$ and X' is generated by the algorithm 3.6.1. We only need to show that $X = X'$.

Since Γ has perfect recall, from lemma 3.4.3 the root of K is recursively exact. By examining the algorithm we know that if a frontier set X is recursively exact then,

- Case i. X is singleton. Y is a child of X , Y is recursively exact by the definition of recursively exact.

- Case ii. X is unreduced. $Succ(X)$ is recursively exact by the definition of recursively exact.
- Case iii. X is reduced. Z is an element of *info-set* (X). Z is recursively exact by the definition of recursively exact.

From the above argument we know that if X is recursively exact then a frontier set X' generated immediately by the algorithm is also recursively exact. Since the root of K is recursively exact, every frontier set generated from algorithm 3.6.1 is recursively exact. Therefore X' is recursively exact and exact by the definition of recursively exact. Then $X' = u^\pi = X$. $\square$

To check the efficiency of the algorithm, the following proposition proves that the algorithm evaluates every leaf at most once.

Proposition 3.6.2 *Let x_0 be the root of a game tree K . The algorithm $V(\{x_0\})$, algorithm 3.6.1, evaluates each leaf in the game tree K at most once.*

Proof: The proof is an induction on the maximum depth of recursion in calculation of $V(X)$. Let X be a frontier set in K . Recall that W_X denotes leaves under the subtrees which have roots in X .

Basis step: Let the maximum depth of the recursion in the calculation of $V(X)$ be 0. Then frontier set X must be reduced and *info-set*(X) is empty set. Also if $x \in X$ then x is a leaf. Since all leaves are disjoint, by examining the algorithm, we know that each leaf has been evaluated at most once.

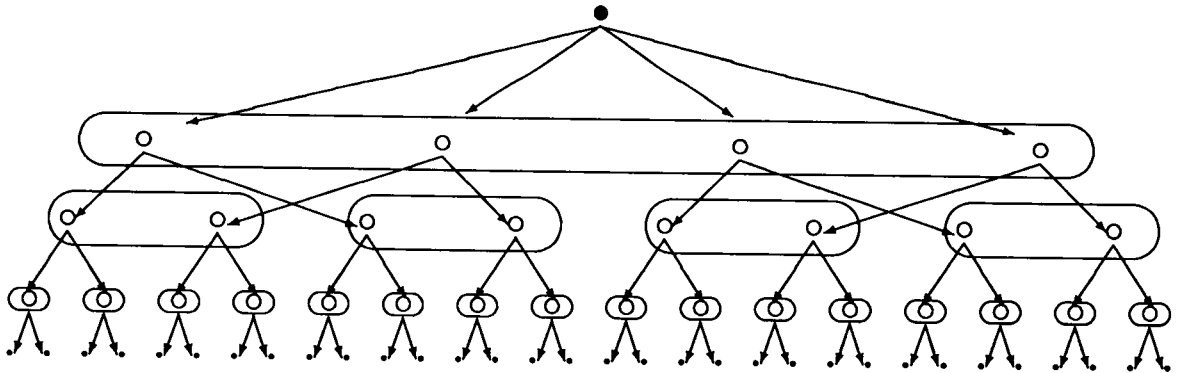
Inductive step: Assume that for any frontier set X for which the maximum depth of the recursion in calculation of $V(X)$ is less than k , $k \geq 1$, then the algorithm $V(X)$ evaluates each leaf in W_X at most once.

Consider a frontier set X for which the maximum depth of recursion in the calculation of $V(X)$ is k .

- Case i: X is singleton. Let Y_i be the i th child of X . Since Y_i has the maximum depth of recursion in calculation of $V(Y_i)$ less than k then we can apply the induction hypothesis that $V(Y_i)$ evaluates each leaf in W_{Y_i} at most once. We know $W_X = \cup_i W_{Y_i}$ and all W_{Y_i} are disjoint sets. Thus $V(X)$ evaluates each leaf in W_X at most once.
- Case ii: X is unreduced. Since $Succ(X)$ has the maximum depth of recursion in calculation of $V(Succ(X))$ less than k , we can apply the induction hypothesis. Hence, $V(Succ(X))$ evaluates each leaf in $W_{Succ(X)}$ at most once. It follows that $V(X)$ evaluates each leaf in W_X at most once.
- Case iii: X is reduced. We know $X = \{\text{leaves in } X\} \cup \text{info-set}(X)$. Let Z be any element of $\text{info-set}(X)$. Since Z is singleton, we can use the same argument in Case i and then $V(Z)$ evaluates each leaf in W_Z at most once. Since all Z are disjoint sets and all leaves in X are disjoint from all Z s, $V(X)$ evaluates each leaf in W_X at most once.

□.

The following is an example of a game which shows the efficiency of the algorithm 3.6.1 comparing to the traditional approach. The root of the game tree is a chance node. Each node has branching factor 2 except the chance node.



Proposition 3.6.3 *Let $\Gamma = (K, \mathcal{P}, \mathcal{U}, h, p)$ be a one person game. The game tree K has height k , $k \geq 2$. The root has N alternatives where $N = 2^{k-2}$. All other personal moves have 2 alternatives. The information sets are grouped together on the same height (i.e. no information set contains two nodes on different heights) where the number of nodes in each information set at depth d is $N/2^{d-1}$. The following are true.*

- *The number of intermediate nodes at depth d , $1 \leq d < k$, is 2^{k+d-3} .*
- *The number of leaves is 2^{2k-3} .*
- *The number of information sets at depth d , $1 \leq d < k$, is 4^{d-1} .*
- *The total number of information sets is $\frac{(4^{k-1}-1)}{3}$.*

Proof: Let the height k of the game tree K be any number where $k \geq 2$. The proof of the number of intermediate nodes at depth d is by induction on the depth d .

Basis step: Let the depth be 1. The number of nodes in the game tree K at depth 1 is $2^{k+1-3} = 2^{k-2} = N$.

Inductive step: Assume that for all depths $d < n \leq k$, the number of nodes in the game tree K at depth d is 2^{k+d-3} .

Consider the depth $n \leq k$. Since each node has two alternatives, and from the inductive hypotheses the number of nodes at depth $(n-1)$ is $2^{k+n-1-3}$, the number of nodes at depth n is $2^{k+n-1-3} \times 2 = 2^{k+n-3}$. This completes the proof of the number of intermediate nodes.

Since the leaves are at the depth k , the number of leaves is $2^{k+k-3} = 2^{2k-3}$. The number of information sets at depth d , $1 \leq d < k$, is the number of moves at depth d divided by the number of moves in each information set at depth d .

That is,

$$\frac{2^{k+d-3}}{\binom{N}{2^{d-1}}} = \frac{2^{k+d-3}}{2^{k-2-d+1}} = 2^{2d-2} = 2^{2(d-1)} = 4^{d-1}$$

The total number of information sets is

$$\sum_{d=1}^{k-1} 4^{d-1} = \frac{4^{k-1} - 1}{3}$$

□

For any $k \geq 2$, there are game trees of the form in proposition 3.6.3 that have perfect recall. The $k = 4$ tree shown above gives an example of such a tree.

Proposition 3.6.4 *Let the game Γ be defined as in proposition 3.6.3, but with perfect recall. Let k be the height of the game tree. The algorithm 3.6.1 calls the recursive function, V , 4^k times.*

Proof: Let t_n be the total number of recursive calls to V when algorithm is called on a singleton set at height n in the game tree.

When the height of the game tree k is 1, it is a singleton information set. This information set has two children which are only leaves. Therefore $t_1 = 2$. There are two children for every singleton frontier set. For each child Z there are two singleton frontier sets in $\text{info-set}(Z)$. Therefore the recurrence can be written,

$$\begin{aligned} t_n &= 2 \times (1 + 2 + 2 \times t_{n-1}) \quad n \geq 2 \\ &= 6 + 4t_{n-1} \end{aligned}$$

Therefore the number of the recursive calls to V is given by

$$t_n = 6 + 4 t_{n-1} \quad n \geq 2$$

subject to $t_1 = 2$.

The recurrence can be written

$$t_{n+1} - 5 t_n + 4 t_{n-1} = 0$$

The characteristic equation is therefore

$$(x - 4)(x - 1) = 0$$

The roots of this equation are 1 and 4, so the general solution of the recurrence is

$$t_n = c_1 + c_2 4^n$$

We know that $t_1 = 2$, and $t_2 = 6 + 4 \times 2 = 14$. Then

$$\begin{aligned} t_1 &= c_1 + c_2 4 = 2 \\ t_2 &= c_1 + c_2 16 = 14 \end{aligned}$$

Then

$$c_2 = 1, c_1 = -2$$

We finally have

$$t_k = 4^k - 2$$

Since t_k is the number of calls at a singleton frontier set, we have another two recursive function calls, one at the root and one at the singleton information set with depth 1. That is,

$$t_k = 4^k - 2 + 2 = 4^k$$

□

The above algorithm evaluates each leaf only once which is 2^{2k-3} . But the number of information sets is $\frac{4^{k-1}-1}{3} \geq 4^{k-2}$ where $k \geq 2$. A naive way to represent a pure strategy is to assign choices at all information sets. Then the number of pure strategies in a game tree with height k is greater than 2^{4^k-2} where $k \geq 2$. This is a quite large number when k becomes large. However, many pure strategies are redundant. For example, if a player chooses the left child at the first information set (depth 1) then the right two children of the first information set are not reachable after that move. We call those unreachable information sets non-sensible and those information sets which can be reached sensible. The

pure strategies defined only at the sensible information sets are called sensible pure strategies. It is easy to verify that the number of sensible information sets at depth k in the above example game is 2^{k-1} . Therefore, the total number of sensible information sets is $2^k - 1$ and the total number of sensible pure strategies is 2^{2^k-1} .

The following table compares the naive pure strategies, sensible pure strategies, number of leaves, and number of recursive function called in the game defined above. This shows that the algorithm significantly reduces the computing time compare to the traditional method.

height k	# of total info. sets	# of pure strategies	# of sensible info. sets	# of sensible strategies	# of leaves	# of func. calls
2	1	2	1	2	2	4
3	5	32	3	8	8	16
4	21	2097152	7	128	32	64
5	85	3.9×10^{25}	15	32768	128	256
6	341	4.4×10^{102}	31	2.1×10^9	512	1024
7	1365	8.1×10^{410}	63	9.2×10^{18}	2048	4096

We can extend the algorithm to check whether an n -tuple pure strategy of an n -player game is equilibrium point.

Algorithm 3.6.2 */* A program to check whether a pure strategy is optimal */*

```
int Optimal = FALSE;
```

```
function CheckOptimal (pi : pure strategy)
```

```
{
```

```
/* Optimal's value upon exit is the answer */
```

```

    for each player i
        V (i, {root of K}))

}

function V (i : player, X : frontier set)
{
    double          maxChildValue;
    double          curChildValue;
    double          sum;
    information set u;

    switch (Property(i, X)):
    {
    case SINGLETON:
        maxChildValue = -LARGE;
        for each child Y of X do
        {
            curChildValue = V (Y);
            if (curChildValue > maxChildValue)
            {
                maxChildValue = curChildValue;
                u = InfoSet (X);
                optimal = Order (X, Y);
            }
        }
    }

    if (optimal != pi[u])

```

```

    {
        Optimal = FALSE;
        exit;
    }
    return (maxChildValue);
    break;
case UNREDUCED:
    return (V (Succ (i, X)));
    break;
case REDUCED:
    sum = 0.0;
    for each leaf w in X do
        sum = sum + p(w)h(w);
        for each element Z in info-set (X)
            sum = sum + V (Z);
    return (sum);
    break;
}
}

```

Chapter 4

Conclusion

4.1 Summary

The traditional method to find equilibrium points in an imperfect information game is to enumerate all pure strategies and to form a linear programming problem. This problem can be solved by a linear programming technique. But the number of pure strategies in a game is already tremendous. To use a linear programming technique makes the problem more difficult. Sometimes an imperfect information game has equilibrium points in pure strategies. If equilibrium points exist in both pure and mixed strategies, it is more desirable to find pure strategies because they are simple and no randomization is involved. The algorithm we suggested here can find an equilibrium point in pure strategies in a one person game with perfect recall. For a general n -player game, it may not have an equilibrium point in pure strategies. For a corresponding one player game a player may not always have a right guess about other players' moves. Then this algorithm can be used as a heuristic technique to find a solution.

The difficulty to find an equilibrium point in pure strategy in a game with imperfect information is that a player has to guess which nodes in an information set are possible by guessing other player's moves. In a one person game without perfect recall, the player has to guess what he had done in previous move. Then

the decision at an information set not only depends on his parent but also depends on his ancestors. Therefore it is difficult to generate a recursive algorithm for an imperfect information game without perfect recall. A one player game with perfect recall makes the recursive algorithm possible because the player remembers all his previous moves and then the decision at an information set can be separated from his ancestors.

In chapter 2 we reviewed the formal definitions of games given by Kuhn[1953]. In chapter 3 we developed the concept of frontier set. One of the differences between perfect information game and imperfect information game is that in a perfect information game a child has only one node but in an imperfect information game a child is a set of nodes. We call this a frontier set. We also developed an evaluation function for frontier sets. The value we assigned to each child (a frontier set) is the value of the child return to his parent. And then we developed an algorithm which generates pure strategies at each information set based on the largest payoffs among his children. The difficulty in this algorithm is how to decide which moves in an information set are possible. We developed the concept of exactness and recursive exactness and proved that the root of a game tree with perfect recall is always exact. Finally, we extended this algorithm to check whether a pure strategy in an n -player game with perfect recall is an equilibrium point.

4.2 Future Study

We already mentioned before that Dalkey had proved that for any payoff function a game has at least one equilibrium point in pure strategies if and only if the game has effectively perfect information. But some games may have special payoff structures. For example, a game may have constant payoff function. Then any pure strategy in this game is an equilibrium point. It is also obvious that if a

subset of a game tree has constant payoff function then any pure strategy in the subgame defined by the subtree is an equilibrium point. In the bridge game the number of leaves in the game tree is enormous but the values of payoff are between 0...13. Therefore, many subtrees have constant payoffs. For example, a hand with four Aces, four Kings, four Queens, and one jack generates a subgame for no trump with constant payoff function. Furthermore, suppose there is an information set in which a child leads to a higher payoff than any other child despite how the other players play. Then it is logical to choose that child for that move. Under those assumptions it is possible to develop a recursive algorithm to reduce the game tree from bottom up and then use a linear programming technique to solve a smaller problem.

BIBLIOGRAPHY

Bibliography

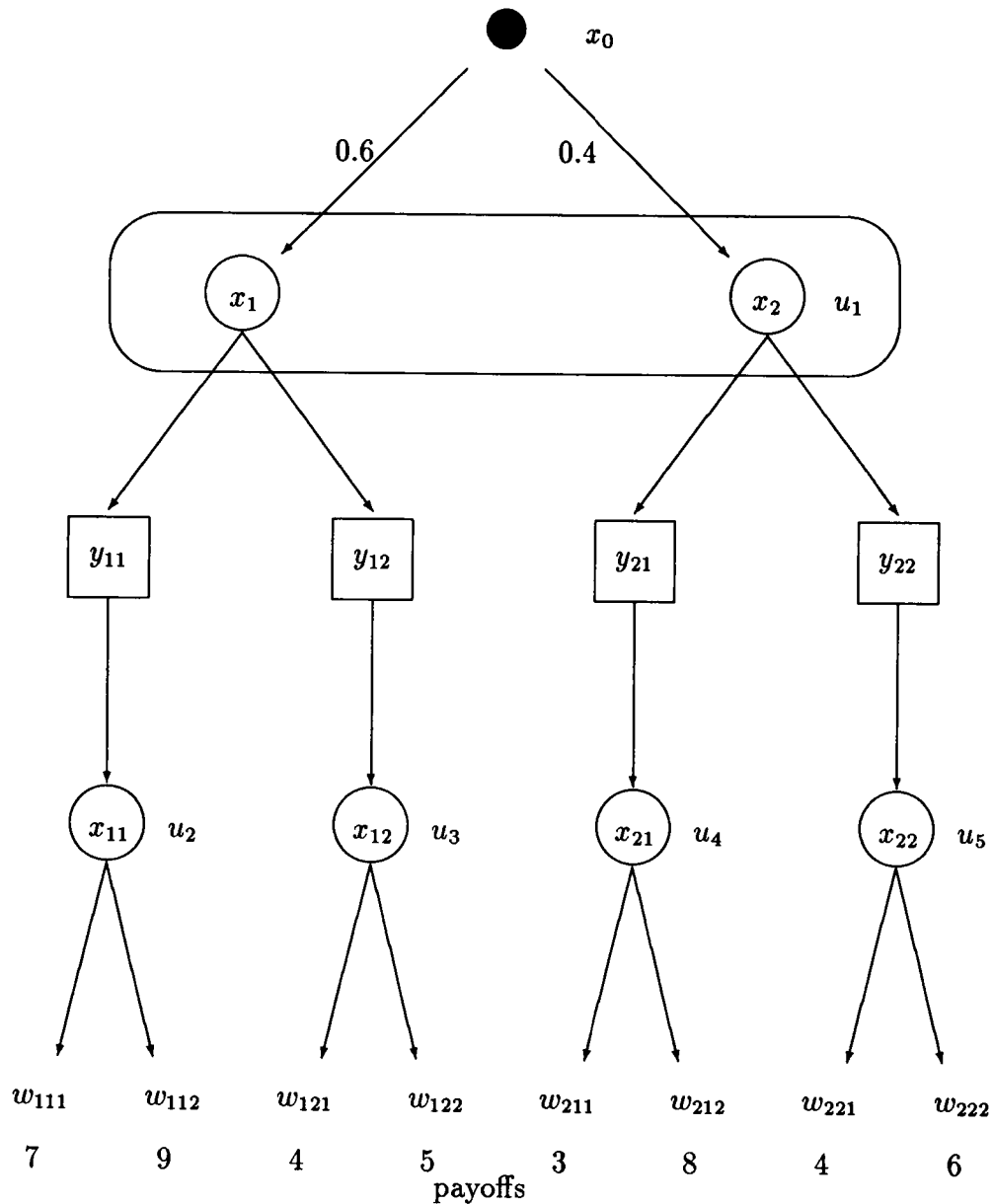
- [1] J. Blair, D. Mutchler, and C. Liu. Technical report in preparing, Department of Computer Science, University of Tennessee, Knoxville, 1992.
- [2] G. Brassard and P. Bratley. *Algorithmics: Theory and Practice*. Prentice Hall, New Jersey, 1988.
- [3] N. Dalkey. Equivalence of information patterns and essentially determinate games. In H. W. Kuhn and A. W. Tucker, editors, *Annals of Mathematics Studies, Number 28, : Contributions to the theory of games, Volume II*, pages 217–243, Princeton University Press, 1953.
- [4] H. W. Kuhn. Extensive games and the problem of information. In H. W. Kuhn and A. W. Tucker, editors, *Annals of Mathematics Studies, Number 28, : Contributions to the theory of games, Volume II*, pages 189–216, Princeton University Press, 1953.
- [5] R. D. Luce and H. Raiffa. *Games and Decisions*. John Wiley and Sons, New York, 1957.
- [6] A. Okada. Complete inflation and perfect recall in extensive games. *International Journal of Game Theory*, 16(2):85–91, 1987.
- [7] G. L. Thompson. Bridge and signaling. In H. W. Kuhn and A. W. Tucker, editors, *Annals of Mathematics Studies, Number 28, : Contributions to the theory of games, Volume II*, pages 279–289, Princeton University Press, 1953.

- [8] G. L. Thompson. Signaling strategies in n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Annals of Mathematics Studies, Number 28, : Contributions to the theory of games, Volume II*, pages 267–277, Princeton University Press, 1953.
- [9] N. N. Vorob'ev. *Game Theory : Lectures for Economists and Systems Scientists*. Springer-Verlag, New York, 1985.

APPENDIX

Appendix

In this section we trace an example for the algorithm describe in chapter 3.



player 1 personal move



player 2 personal move



chance node

w

leaves

u_i

information sets

payoffs

```

01  V ( $\{x_0\}$ )
01   $\{x_0\}$  is unreduced
01  Succ (X) is  $\{x_1, x_2\}$ 
02      V ( $\{x_1, x_2\}$ )
02       $\{x_1, x_2\}$  is singleton
02      maxChildValue = -LARGE
02      /* left child is  $\{y_1, y_3\}$ , right child is  $\{y_2, y_4\}$  */
02      /* calculate the left child */
02      curChildValue = V  $\{y_1, y_3\}$ 
03           $\{y_1, y_3\}$  is unreduced
03          Succ (X) is  $\{x_{11}, x_{21}\}$ 
04              V  $\{x_{11}, x_{21}\}$ 
04               $\{x_{11}, x_{21}\}$  is reduced
04              sum = 0.0
04              /* there is no leaf in  $\{x_{11}, x_{21}\}$  */
04              /*  $\{x_{11}\}$  is the first element of info-set  $\{x_{11}, x_{21}\}$  */
04              /* calculate the first element */
04              sum = sum + V  $\{x_{11}\}$ 
05                   $\{x_{11}\}$  is singleton
05                  maxChildValue = -LARGE
05                  /* left child is  $\{l_{111}\}$ , right child is  $\{l_{112}\}$  */
05                  /* calculate left child  $\{l_{111}\}$  */
05                  curChildValue = V ( $\{l_{111}\}$ )
06                       $\{l_{111}\}$  is reduced
06                      sum = 0.0
06                      /*  $l_{111}$  is the only leaf */
06                      sum = 0.0 + p ( $l_{111}$ ) h ( $l_{111}$ )

```

```

06          = 0.0 + 0.6 × 7 = 4.2
06          /* the info-set $\{l_{111}\}$  is empty set */
06          return (4.2)
05      curChildValue = 4.2
05      (4.2 > -LARGE)
05      maxChildValue = 4.2
05      /* calculate the right child */
05      curChildValue = V ( $\{l_{112}\}$ )
06           $\{l_{112}\}$  is reduced
06          sum = 0.0
06          /*  $l_{112}$  is the only leaf */
06          sum = 0.0 + p ( $l_{112}$ ) h ( $l_{112}$ )
06              = 0.0 + 0.6 × 9 = 5.4
06          /* the info-set $\{l_{112}\}$  is empty set */
06          return (5.4)
05      curChildValue = 5.4
05      (5.4 > 4.2)
05      maxChildValue = 5.4
05      /* u is the second information set */
05      u = 2
05      Optimal [2] = Right Child
05      return (5.4)
04      sum = 0.0 + 5.4 = 5.4
04      /*  $\{x_{21}\}$  is the second element of  $\{x_{11}, x_{21}\}$  */
04      /* calculate the second element */
04      sum = 5.4 + V  $\{x_{21}\}$ 
05           $\{x_{21}\}$  is singleton

```

```

05      maxChildValue = -LAEGE
05      /* left child is  $\{l_{211}\}$ , right child is  $\{l_{212}\}$  */
05      /* calculate the left child  $\{l_{111}\}$  */
05      curChildValue = V ( $\{l_{211}\}$ )
06           $\{l_{211}\}$  is reduced
06          sum = 0.0
06          /*  $l_{211}$  is the only leaf */
06          sum = 0.0 + p ( $l_{211}$ ) h ( $l_{211}$ )
06              = 0.0 + 0.4  $\times$  3 = 1.2
06          /* the info-set $\{l_{211}\}$  is empty set */
06          return (1.2)
05      curChildValue = 1.2
05      (1.2 > -LARGE)
05      maxChildValue = 1.2
05      /* calculate the right child  $\{l_{212}\}$  */
05      curChildValue = V ( $\{l_{212}\}$ )
06           $\{l_{212}\}$  is reduced
06          sum = 0.0
06          /*  $l_{212}$  is the only leaf */
06          sum = 0.0 + p ( $l_{212}$ ) h ( $l_{212}$ )
06              = 0.0 + 0.4  $\times$  8 = 3.2
06          /* the info-set $\{l_{212}\}$  is empty set */
06          return (3.2)
05      curChildValue = 3.2
05      (3.2 > 1.2)
05      maxChildValue = 3.2
05      /* u is the 4'th information set */

```

```

05             u = 4
05             Optimal [4] = Right Child
05             return (3.2)
04             sum = 5.4 + 3.2 = 8.6
04             return (8.6)
03         return (8.6)
02     curChildValue = 8.6
02     (8.6 > -LARGE)
02     maxChildValue = 8.6
02     /* calculate the right child */
02     curChildValue = V {y2, y4}
03         {y2, y4} is unreduced
03         Succ ({y2, y4}) is {x12, x22}
04             V {x12, x22}
04             {x12, x22} is reduced
04             sum = 0.0
04             /* there is no leaf in {x12, x22} */
04             /* {x12} is the first element of info-set {x12, x22} */
04             /* calculate the first element */
04             sum = sum + V {x12}
05                 {x12} is singleton
05                 maxChildValue = -LARGE
05                 /* left child is {l121}, right child is {l122} */
05                 /* calculate left child {l121} */
05                 curChildValue = V ({l121})
06                     {l121} is reduced
06                     sum = 0.0

```

```

06          /*  $l_{121}$  is the only leaf */
06          sum = 0.0 + p ( $l_{121}$ ) h ( $l_{121}$ )
06              = 0.0 + 0.6  $\times$  4 = 2.4
06          /* the info-set $\{l_{121}\}$  is empty set */
06          return (2.4)
05      curChildValue = 2.4
05      (2.4 > -LARGE)
05      maxChildValue = 2.4
05      /* calculate the right child */
05      curChildValue = V ( $\{l_{122}\}$ )
06           $\{l_{122}\}$  is reduced
06          sum = 0.0
06          /*  $l_{122}$  is the only leaf */
06          sum = 0.0 + p ( $l_{122}$ ) h ( $l_{122}$ )
06              = 0.0 + 0.6  $\times$  5 = 3.0
06          /* the info-set $\{l_{122}\}$  is empty set */
06          return (3.0)
05      curChildValue = 3.0
05      (3.0 > 2.4)
05      maxChildValue = 3.0
05      /* u is the 3'rd information set */
05      u = 3
05      Optimal [3] = Right Child
05      return (3.0)
04      sum = 0.0 + 3.0 = 3.0
04      /*  $\{x_{22}\}$  is the second element of  $\{x_{21}, x_{22}\}$  */
04      /* calculate the second element */

```

```

04      sum = 3.0 + V { $x_{22}$ }
05      { $x_{22}$ } is singleton
05      maxChildValue = -LAEGE
05      /* left child is { $l_{221}$ }, right child is { $l_{222}$ } */
05      /* calculate the left child { $l_{221}$ } */
05      curChildValue = V ({ $l_{221}$ })
06      { $l_{221}$ } is reduced
06      sum = 0.0
06      /*  $l_{221}$  is the only leaf */
06      sum = 0.0 + p ( $l_{221}$ ) h ( $l_{221}$ )
06      = 0.0 + 0.4  $\times$  4 = 1.6
06      /* the info-set{ $l_{221}$ } is empty set */
06      return (1.6)
05      curChildValue = 1.6
05      (1.6 > -LARGE)
05      maxChildValue = 1.6
05      /* calculate the right child { $l_{222}$ } */
05      curChildValue = V ({ $l_{222}$ })
06      { $l_{222}$ } is reduced
06      sum = 0.0
06      /*  $l_{222}$  is the only leaf */
06      sum = 0.0 + p ( $l_{222}$ ) h ( $l_{222}$ )
06      = 0.0 + 0.4  $\times$  6 = 2.4
06      /* the info-set{ $l_{222}$ } is empty set */
06      return (2.4)
05      curChildValue = 2.4
05      (2.4 > 1.6)

```

```

05             maxChildValue = 2.4
05             /* u is the 5'th information set */
05             u = 5
05             Optimal [5] = Right Child
05             return (2.4)
04             sum = 3.0 + 2.4 = 5.4
04             return (5.4)
03         curChildValue = 5.4
03         (5.4  $\not\geq$  8.6)
03         /* u is the first information set */
03         u = 1
03         Optimal [1] = Left Child
03         return (8.6)
02     return (8.6)
01 return (8.6)

```

VITA

Cheng Liu was born in Taiwan. He received a PhD degree in geography from University of Tennessee in 1986. While working at Federal Express Corporation, he decided to study computer science at Memphis State University. After two and half years of living in Memphis, he realized that Memphis is not the best place to live in the world. He started working for Oak Ridge National Laboratory in 1988 and continued his education at University of Tennessee. He received his MS in computer science in 1992. He plans to pursue a PhD in computer science, but his wife and two kids do not allow him to do so.