Generating and Solving Imperfect Information Games

Daphne Koller
University of California
Berkeley, CA 94720
daphne@cs berkeley edu

Avi Pfeffer
University of California
Berkeley, CA 94720
ap@cs berkeley edu

Abstract

Work on game playing in AI has typically ignored games of *imperfect information* such as poker In this paper we present a framework for dealing with such games We point out several important issues that arise only in the context of imperfect information games particularly the insufficiency ot a simple game tree model to represent (he players information state and the need for randomization in the players optimal strategics We describe *Gala* an implemented system that provides the user with a very natural and expressive language for describing games From a game description Gala creates an augmented game tree with *information sets* which can be used by various algorithms in order to find optimal strategies for that game In particular Gala implements the first practical algorithm for finding optimal randomized strategies in two player imper fect information competitive games [Koller *el al* 1994] The running time of this algorithm is *palvno mial* in the size ot the game tree whereas previous algorithms were exponential We present experimental results showing that this algorithm is also efficient in practice and can therefore form the basis for a game playing system

1 Introduction

The idea of getting a computer to play a game has been around since the earliest days of computing The fundamental idea is as follows When it is ihe computer s turn lo move U creates some part of the game tree starting at the current position evaluates the leaves of this partial tree using a heuristic evaluation function and then does a minimax search of this tree lo determine the optimal move al lhc root This same simple idea is still the core of most game-playing programs This paradigm has been successfully applied lo a large class of games in particular chess checkers othello backgammon andgotRussellandNorvig 1994 Ch 5] There have been far fewer successful programs that play games such as poker or bridge We claim that this is not an accident These games fall into two fundamentally different classes and the techniques thai apply to one do not usually apply to the other

The essential difference lies in the information that is avail able to the players In games such as chess or even backgam mon, the current state of the game is fully accessible to both players The only uncertainty is about future moves In games such as poker (he players have *imperfect information* they have only partial knowledge about the current state of the game This can result in complex chains of reasoning such as Since I have two aces showing but she raised men she is either bluffing or she has a good hand but then if I raise a lot she may realize that 1 have al least a third ace, so she might fold so mavbc I should underbid bul It should be fairly obvious that the standard techniques are inadequate lor solving such games no variant of the minimax algorithm duplicates the type of complex reasoning we just described

In *game theory* [von Neumann and Morgenslern 1947] on the other hand virtually all of the work has focused on games with imperfect information Game theory is mostly intended lo deal with games derived from real life, and particularly from economic applications In real life one rarely has perfect information The insights developed hy game theorists for such games also apply to the imperfect information games encountered in AI applications

It is well known in game theory lhal the notion of a *strai eg* is necessarily different for games with imperfect mforma lion In pcrlccl inlormalion games the optimal move for each player is clearly defined al every stage there is a right move thai is *di* feast as good as *any* other move But in imperfect information games the situation is not as straightforward In the simple game of scissors paper stone any deterministic strategy is a losing one as soon as it is revealed lo the olher players Intuitively in games where there is an information gap it is usually lo my advantage lo keep my opponent in the dark The only way to do thai is by using *randomized strategies* Once randomized strategics are allowed ihe exis lence of optimal strategies in imperfect information games can be proved In particular this means lhal lhcrc exists an optimal randomized strategy lor poker in much the same way as there exists an optimal deterministic strategy for chess Kuhn [19*>0l has shown for a simplified poker game lhal the optimal strategy does indeed use randomization

The optimahly of a strategy has two consequences the player cannot do better than this strategy if playing against a good opponent and lurlhermore the player docs not do worse even if his strategy is revealed IO his opponent i c lhe opponent gains no advantage lrom figuring out the player s strategy This last feature is particularly important in the context of game-playing programs since they arc vulnerable lo ihis form of aitack sometimes the code is accessible and in general since they always play the same way thei rstrategy can be deduced by intensive testing Given these important benefits of randomized strategies in imperfect information games u is somewhat surprising that none of the AI papers that deal wim these games (e g [Blair era/ 1993 Gordon 1993 Smith and Nau 1993]) utilize such strategics

In this work we attempt to solve the computational problem associated with imperfect information games Given a concise description of a game compute optimal strategies for thai game Two issues in particular must be addressed First how do we specify imperfect information games' Describing the dynamics of the players information states in a concise fashion is a nontnvial knowledge representation task Second given a game tree with the appropriate structure how do we find optimal strategies for it?

We present an implemented system called *Gala* that ad dresses both these computational issues Gala consists of four components The first is a knowledge representation language that allows a clear and concise specification of imperfect in formation games As our examples show the description of a game in Gala is very similar to and not much longer than a natural language description of the rules of the game The second component of the system generates game trees from a game description in the language These game trees are aug menled with *information* tefv a standard concept Irom game theory thai captures the information slates of the players

The third component of the system addresses the issue *of* finding good strategies for such games Obviously the stan dard minimax type algorithms cannot produce randomized strategies The game theoretic paradigm for solving games is based on taking the entire game tree and transforming it into a matrix (called the *normal* or *strategic form* of the game) Various techniques such as linear programming can then be applied lo this matrix in order lo construct optimal strategies Unfortunately this matrix is typically exponential in the size of the game tree making the entire approach impractical for most games

In recent work Koller, Megiddo and von Stengel [1994] present an alternative approach lo dealing with imperfect in formation games They deline a new representation called the *sequence form* whose size is li*near in* the size of the game tree They show that many of the standard algorithms can be adapted to find optimal strategies using this representation This results in exponentially faster algorithms for solving a large class of games In particular they present an effective polynomial time algorithm for solving two player fullv com pctilivc games (such as poker) We have implemented this algorithm as part of the Gala system and tesled it on large examples of several games The results arc encouraging suggesting that in practice ihe running lime of the algorithm is a small polynomial in the size of the game tree

The final component of GalapresenLs theoplimal strategics in a way thai is comprehensible lo the user For any decision point in the game it lells the user which actions should be played with which probability The system also provides other information such as one player s beliefs about the stale of anolher agent or the expected value of a branch in the tree This functionality makes Gala a useful tool lor game theory researchers and educators as well as for users who wish lo use Gala as a game-theory based decision support system Finally Gala can also play the game according lo the computed strategy making il a basis for a computer game-playing system for imperfecl information games

2 Some basic game theory

Game Ihcory is ihe strategic analysis of interactive situations Several aspects of a situation are modeled explicitly the players involved the alternative actions that can be taken by each player at various limes, the dynamics of the situation ihe information available to players and die outcomes at the end Given such a model game theory provides the tools to formally analyze the strategic interaction and recommend rational strategies to the players

The standard representation of a game in computer science is a tree in which each node is a possible stale of the game, and each edge is an action available to a player that takes the game to new stale At each node there is a single player whose turn it is to choose an action The set of edges leading out of a node arc the choices available lo that player The player may be chance or nature in which case the edges represent random events The leaves of the tree specify a payoff for each player This representation is inadequate for games with imperfect inlormation because it docs not specify the information states ol the players A player cannol distinguish between states of the game in which she has ihe same information Thus any decision taken by the player must be the same at all such nodes To encode ihis constraint the game tree is augmented with *information sets* An information sel contains a set of nodes that arc indistinguishable lo a player at the time she has lomakc a decision

Figure 1 presents part of the game tree for a simplified variant of poker described by Kuhn [1950] The game has two players and a deck containing the Ihree cards 1 2 and 3 Each player antes one dollar and is dealt one card The figure shows the part of the game tree corresponding lo the deals (2,1) (2,3) and (I 3) The game has three rounds In Ihe first round the first player can either bet an additional dollar or pass Alter hearing the first player s bet the second player decides whether to bet or pass If player 1 passes and player 2 bets player 1 gets one more opportunity lo decide whclhcr or nol to bel If both bet or both pass the player with the highest card takes the pot If one player bets and the other passes then the betting player wins one dollar Lei (t *d)* denote the hands dealt lo the two players Initially, player 1 only knows his own card so for each possible *c* he has one information set 0_e containing two nodes, each node corresponds to the two possibilities for player 2 s hand In her turn player 2 knows *d* as well as player 1 s action at the first round Hence she has iwo information sets for each *d—\%* and I_d—corresponding lo player 1 s previous action Finally player 1 has an information set U'_c at the third round

Given a game tree augmented with information sets, one can define the notion of *strategy* *A deterministic strategy* like a conditional plan in AI is a very explicit 'how-to-play manual that tells the player what to do at every possible point in the game In the poker example, such a manual for player 1 would contain an entry If I hold a 3 and I passed on the first round, and my opponent bets then bet 1 " In general a deterministic strategy for player specifies a move at each of her information sets Since the player cannot distinguish between nodes in die same information set the strategy cannot dictate different actions at those nodes

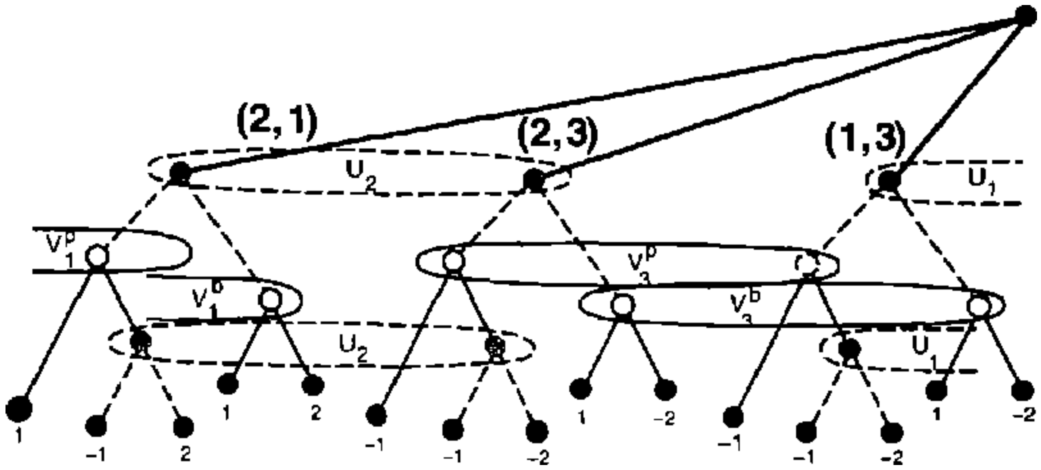

Figure I A partial game tree for simphlified poker, containing three of the six possible deals A move to the left corresponds to a pass a move to the right to a be [The information sets are drawn as ellipses some of them extend into other parts of the tree

Deterministic strategies arc adequate for games with perfect information where the players always know the current stale of the game In those games the information sets of both players are always single nodes and a deterministic strategy s, for player ? is a function from those nodes at which n is her turn to move to possible moves al that node The fact that deterministic strategics suffice for such games is the basis for the standard *mint max* algorithm (and Us variants) used for games such ai chess In such games called *zero sum games* there arc two players whose payoffs always sum to zero so that one player wins precisely what the other loses As shown by Zermelo [1913] the strategies produced by the mimmax algorithm are optimal in a verv strong sense Player *i* can not do better than to play the resulting strategy if the other player is rational Furthermore she can publicly announce her intention to do so without adversely affecting her pay offs A generalized version of the minimax algorithm shows the existence of optimal deterministic strategics for general games of perfect information The resulting strategy com bination (AI S„) has the important property of being in *equilibrium* for any *J* player *i* cannot pick a better strategy than 6, if the other players arc all playing their strategy s_j This is a minimal property Ihal wc want of a solution to a game Without it we are drawn back into the web of second guessing that characterizes imperfect information games (If she plays the orthodox strategy then I should do Y but she will figure out that this is better for me so she II actually do i but then)

It should be fairly obvious Ihal deterministic strategies will in general not have dicse properties in games with imperfect information Deterministic strategies are predictable and pre dictable play gives the opponent information The opponent can then find a strategy calculated to take advantage of this information thereby making the original strategy suboptimal Unpredictable play on the other hand, maintains Ihe informa Uon gap Therefore players in imperteci information games should use *mndonuzed strategies*

Randomized strategies are a natural extension of delcrmin istic strategies Where a deterministic strategy chooses a move at each information set a randomized strategy (formally called a *behavior strategy)* specifies a probability distribution over Lhe moves at each information set In our poker example a randomized strategy m_1 lor player 1 can be described by *defining* the probability of *betting* al each information *set* L'_e and t''_c $c = 1\ 2\ 3$ A combination of randomized strategies *fi]* μ_n one for each player induces a probability distn buLion on the leaves of the tree thereby allowing us to define the *exptdidpu\off* $h_i(\mu_1 \quad \mu_n)$ for each player i

In his Nobel prize winning theorem Nash showed that the use of randomized strategies allows us to duplicate the sue cessful behavior that we gel from deterministic strategies in the perfect information case In general games there is always a combination μ_1 , μ_n of randomized strategies that is in equilibrium for any *i* and any strategy *fi[,*

$$h_i(\mu_1 \quad \mu_n) \geq h_i(\mu_1, \quad \mu'_i \quad \mu_n)$$

That is no player gains an advantage by diverging from the equilibrium solution so long as Ihe other players stick (o II

Just as in Ihe case of perfect information games the equilibrium strategies arc particularly compelling when the game is zero-sum Then as shown by von Neumann fvon Neumann and Morgenstern 19471 any equilibrium strategy is optimal against a rational player More precisely the equilibrium p a i μ_1, μ_2 e precisely those where *fi* is Ihe strategy that maximızes $\max_{\mu'_1} \min_{\mu'} h_1(\mu'_1 \ \mu'_2)$ and *ftj* is the strategy that maximizes $\max_{\mu'} \min_{\mu'_1} h_2(\mu'_1, \mu'_2)$ (which since *hi* = —/i] is precisely $\min_{\mu'} \max_{\mu'_1} h_1(\mu'_1 \ \mu'_2)$) Intuitively, *fi* is the optimal defensive strategy Tor player 1 K provides the best worst-case nayotf It is these strategies that we will be most conceded with finding

3 Gala *a* game description language

As we mentioned Ihe first component of Gala is a knowledge representation language for describing games This is a Prolog based language, thai uses the power of a declarative representation to allow clear and concise specification of games The idea of a declarative language to specify games was proposed by Pell 11992] who utilizes it to specify

```
game(blind_tic_tac_toe
 |players   [a  b]
  objects   |grid_board   array| $size    $size ))
  params   [size]
  flow   [take_turns(mark unless(full) until(win)))
  mark   (choose( $player   (X  Y  Mark)
            (empty(X  Y)  member(Mark  [x  o])))
          reveal( $opponent   (X  Y))
          place((X  Y)  Mark))
  full   (\+(empty(_  _))
           outcome(draw))
  win   (straight_line(_  _  length = 3  contains(Mark)) ->
           outcome(wins( $player )))])
```

Figure 2 A Gala description of blind tic tac-ioc

symmetric chess like games—a class of Iwo-playcr perfecl-information board games Our language is much more gen eral and can be used lo represenl a very wide class of games in particular one-player two-player ajid multi player games games where the outcomes arc arbitrary payoffs and game*, with either perfect or imperfect information As we will show the expressive power of Gala allows for clear and concise game descriptions that arc generally of similar length to natural language representations of the rules of the game

To illustrate some of the features of Gala Figure 2 presents an example of a complete description for blind tic lac-loe an imperfect information version of standard tic-tac toe The player*, lake turns placing marks in squares, but in his turn a player can choose to mark either an x or an o he reveals lo his opponent the square in which he makes the mark but nol the type of mark used As usual Ihe goal is to complete a line of three squares with the same mark

A game description in Gala is a list of features each one describing some asped of the game For example players (a b] indicates that the game is to be played between two players named a and b

The Gala language has several layers the lower ones pro vide basic primitives while the higher layers use those primi lives to provide more complex functionality The lowesl layer provides the fundamental primitives for defining the structure of a game The choose (P Layer Mo Je Constraint) primitive describes the possible moves available lo player at a given point in the game It allows player lo make any move Move satisfying constraint This last argument can be an arbitrary segment of Prolog code In our example Move consists of a square specified by its coordinates x and Y and a mark Mark constraint requires Ihat the square be empty and that Mark be either *x* or o The first argumenl lo choose can also be nature in which case one of a number of events is chosen at random By default these random events have uniform probability but a different probability distribution may be specified The outcome primitive describes the outcome of the game at the end of a particular sequence of moves This will often be a list of payoffs one for each player but as the example demonstrates Gala allows Other possibilities The reveal (Player Fact) primitive describes the dynamics of the players information stales It adds Fact lo player s information state The information added can be simple or an arbitrary Prolog ex pression In blind tic tac-loe a player chooses both a square and a mark but reveals to his opponent only the mark

At a somewhat higher level the flow feature describes the course of the game The game can be divided into phases some may lake place just once while others can be repeated until a goal is reached In blind tic tac-toc for example the players take turns executing the sequence of actions specified in the mark feature, until the condition specified in the full or the win feature is satisfied The unless condition is tested before the turn Gala also allows gameflow to be nested recursively Each phase can be described by its own series of features which may include flow The flow of bridge for example can be described as follows

```
flow   (play_phase(bidding)  play_phase(take_tricks))
phase(bidding
 [flow   (take_turns(bid  until(contract_reached)])
phase(take_tricks
 {flow   (p'ay_ro unds(trick  13)
```

In order to allow a natural specification of the game, Gala provides a separate representation for the *game state* where relevant information about the current state of the game is stored In blind tic-tac-loe the game state contains the currenl board position This information is accessed, for example, by choose in order lo determine which moves are possible only those squares that are empty are legal moves The game state is maintained by modifying it appropriately e g , by the place operation when the players make their moves Much of the functionality in the higher levels of the Gala language is devoted lo accessing and manipulating the game state

The intermediate levels of Gala provide a shorthand for concepts that occur ubiquitously in games These include lo cations and their contents pieces and their movement patterns and resources that change hands such as money In blind tic lac toe the statements that deal with the contents of squares are an instance of locations and their contents other examples of functionality supported by this level are move(queen(white) [d 1) (d B)) and pay(gambler pot Bet)

On a more abstract level we have observed Ihat certain structures and combinations appear in virtually all games While ihese are usually sets of one sort or another they come in many flavors For example, a flush in poker is a set of five cards sharing a common property a straight on the other hand is a sequence of cards in which successive elements bear a relation to one another a full house is a partition into equivalence classes based on rank in which the classes are of a specific size A word in Scrabble and a 21 in Blackjack are another type of combination a collection of objects bearing no particular relationship lo each other but forming an interesting group in totality

The Prolog language provides a few predicates that describe sets and subsets We have supplemented these with various predicates thai make it easy to describe many of the combinations occunng in games For example, chain(predicate set) determines whether sat is a sequence in which succes sive elements are related by predicate partition (Relation set classes) partitions set into equivalence classes based on Relation For a more elaborate example, consider the fol lowingcode which concisely tests for all types of poker hand except flushes and straights

```
detailed_partition(match_ran)(  Hand  Classes  Ranks  Sizes)
 associate  (Sizes  Type
   [ ( [4  1]   four_of_a_kind>   ( [3  2]   full_houae)
     ([3  1  1]   three_o£_a_kind)   ([2  2  1]   two_pairs)
     ([2  1  1  11   one_pair)   ( [ 1 1 1 1 1 ]   nothing)])
```

The predicate detailed_partition takes two inputs a set—in this case Hand—and an equivalence relation—in this case match-rank, which relates two cards if they have the same rank It partitions the set into equivalence classes and produces three outputs a list ciassed of the equivalence classes

In decreasing order of size a corresponding list of the defin ing property of the equivalence classes in this case the Ranks present in the hand and a list sizes of the sizes of the different classes In this example if Hand is [9♡ 6♣ 9♠ 6♡ 6◊] then Classes Would be [[6♣ 6♡ 6◊] [9♡ 9♠]] Ranks would be [6 9] and sizes would be [3 2] In poker, sizes contains the relevant structure of the hand and it is used to classify the hand using an association list The above hand for example is immediately classified as a full house

The high level modules of Gala build on the intermediate levels to provide more specifie functionality thai is common to a certain class of games such as boards that form a grid playmgcards dice and so on In the blind tic-tac toe example we declare a grid-board object This makes a whole range of predicates available that depend on the board being rectilinear The straight line predicate is an example it tests for a straight line of three squares containing the same mark This predicate is defined in terms of chain In general high level predicates are typically very easy to define in terms of the intermediate level concepts so that adding a module for a new class of games requires little effort

A useful feature of Gala is that it allows some parameters of the game to be left unspecified in the game description and provided when the game is played In blind tic-lac toe the board size is such a parameter This makes it very easy to encode a large class of games in a singlt program These parameters can actually be code-containing features Thus, it is possible to provide the movement patterns of pieces in a game at runtime This allows a simple interface between Gala and Pell s *Metagame* program [Pell 1992] which generates symmetric chess like games randomly

Given a description of a game in the Gala language Gala generates the corresponding game tree with information sets as described in Section 2 The tree is defined by the choose reveal and outcome primitives The Gala interpreter plays the game and constructs the game tree as it encounters these operations When it encounters a choose primitive a node is added lo the tree and an edge is added for every option available to the player The interpreter then explores each branch of the tree corresponding to each ol the options If the first argument lo choose is a player, the system also adds the node to the appropriate information set of that player the one that contains all the nodes where the player has the same information slate The information slate consists of all facts revealed lo the player by the reveal primitive the list of choices available to the player and all decisions previously taken by the player If the first argument to choose is random, then the node is marked as a chance node and the probability of each random choice is recorded When the interpreter encounters the outcome primitive it adds a leaf to the tree and backtracks to explore other branches

4 Solving imperfect information games

How do we find equilibrium strategies in imperfect informa tion games? This is in general a very difficult problem Consider the poker example from Section 2 There we specified a strategy for each of the players using six numbers When trying to solve a game we need *lo find* an appropriate set of numbers that satisfies the properties we want That is we want to treat the parameters of the strategy as variables and solve for them The general computational problem is

Maximize$_x$ min$_y$ $h(x, y)$
subject lo x represents a strategy for player 1 (*)
y represents a strategy for player 2

where $h(x\ y)$ denotes the expected payoff to player 1 if the strategies corresponding to x y arc played

It turns out that the heart of the problem is finding an appropriate set of variables for representing the strategy The first atlempl is lo use the move probabilities in the behavior strategy In the poker example we would then have $x = \{x_c, x'_c \quad c = 1\ 2\ 3\}$ representing player 1 s strategy, and $y = \{y_d^b\ y_d^c \quad d = 1\ 2\ 3\}$ representing player 2 s strategy The problem is that this payoff is a nonlinear function of the x s and y s In order to avoid this problem which would force us louse nonlinear optimization techniques the standard solution algorithms in game theory do not use game trees and behavior strategics as their primary representation Rather they operate on an alternative representation called the *normal form* In the two player case the normal form is a matrix A whose rows are all the deterministic strategics of the first player and whose columns are all the deterministic strategies of the second The entry in the zth row and jth column is the expected payoff lo the players when player 1 plays strategy s_1^i and player 2 plays strategy s_2^j A randomized strategy can now be viewed as a probability distribution over all the deterministic strategies Hence x is simply a probability distribution over rows it has a variable x_i for each row such that $x_i \geq 0$ for all i and $\sum_i x_i = 1$ If player 1 plays a- and player 2 plays y then lhe expected payoff of the game is simply $x^T Ay$ Under this representation of strategies $(\star)$ takes aparticularly simple form It is then fairly easy lo show that that appropriate vectors s and y can be found from A using standard linear programming methods

For non zero-sum games the normal form also forms the basis lor essentially all solution algorithms Gala provides access to the normal form algorithms using an interface to the GAMBIT system developed by McKelvey and Turocy [McK eclvey, 1992] GAMBIT provides a toolkit for solving various classes of games including games with more than two players and games where the interests of the players arc not strictly opposing Since Gala allows a clear and compact specifi cation of such games the combined system provides both a represenlation language and solution algorithms for games describing multi agent interactions

Unfortunately the normal form algorithms arc practical only for very small games The reason is that the normal form is typically *exponential* in the size of the game tree This is easy lo see A determinitic strategy must specify an action at each information set The total number of possible strategies is therefore exponential in the number of information sets which is usually closely related to the size of Lhe game tree Consider our poker example generalized lo a deck with k cards For each card c player 1 must decide whether to pass or bet and if he has the option whether lo pass or bet at the third round There are three courses of action for each c so the total number of possible strategies is 3^K Player 2 on the other hand, must decide on her action for each card d and each of the two actions possible for the first player in the first round The number of different decisions is therefore $2k$ so the total number of deterministic strategies is $2^{2k} = 4^k$ Since the normal form has a row for each strategy of one player and a column for each strategy of the other it is also exponential in k while the size of the game tree is only 9k + 1 In general the normal form conversion is typically exponential in terms of both time and space

This problem makes the standard solution algorithms an unrealistic option for many games Due to the large branching factor in many games even the approach of incrementally solving subtrees would not suffice to solve this problem (This approach also encounters other difficulties in the context of imperfect information games see Section 6) Recently a new approach to solving imperfect information games was developed by Koller Megiddo and von Stengel [1994] This approach uses a conversion to an alternative form called the *sequence form,* which allows it to avoid the exponential blowup associated with the normal form We will describe the main ideas briefly here for more details see [Koller *et al* 1994]

The sequence form is based on a different representation of the strategic variables Rather than representing probabilities of individual moves (as in the non linear representation above) or probabilities ol full deterministic strategies (as in the normal form) the variables represent the *realization weight* of different *sequences* of moves Essentially a sequence for a player corresponds to a path down the tree but it isolates the moves under that player's direct control ignoring chance moves and the decisions of the other players In our poker game for example player 1 would have $4k + 1$ sequences In addition to the empty sequence (which corresponds to the root of the game) he has four sequences for each card *c* [bet on *c]* (in which case there is no third round) [pass on c], [pass on c, bet in the last round] and [pass on *c* pass in the last round] Player 2 also has $4k + 1$ sequences the empty sequence and for each card *d* the four sequences [bet on *d* alter seeing a pass] [pass on *d* after seeing a pass] [bet on *d* after seeing a bet] [bet on *d* after seeing a bet] Given a randomized strategy the realization weight of a sequence for a player is the producL of the probabilities of the player s moves encoded in the sequence Essentially the realization weight of the sequence corresponding to a path down the tree is a conditional probability the probability that this path is taken given that the other players and nature all cooperate to make this possible The probability that a path is actually taken in a game is therefore the product oi the realization weights of all the players sequences on that path times the probability ot all the chance moves on the path

The *sequence form* of a two player game consists ol a payoff matrix *A* and a linear system ol constraints for each player In a two player game the zth row ot A corresponds to a sequence a_1 lor player I and the jth column to a sequence σ_2 for player 2 The entry a_{ij} is the weighted sum of the payoff al the leaves that are reached by this pair of sequences (they are weighted by the probabilities of the chance moves on the path) If a pair of sequences is not consistent with any path to a leaf the malnx entry is zero So lor example the matrix entry for the pair of sequences [bet on 2] and [pass on 1 after seeing a bet] is 1 The matrix entry for the pair [bet on 2] and [pass on 1 after seeing a pass] is 0, since this pair is not consistent with any leaf

We now solve (*) using realization weights as our strategic variables We will have a variable x_{σ_1} for each sequence a] of player 1, and a variable y_{σ_2} for each sequence a_2 of player 2 Using the analysis above we can show that the expected payoff of the game $h(x, y)$ is $x^T Ay$ This is precusely analogous to the expression we obtained for the norma] form It remains only to specify constraints on x and *y* guaranteeing that they represent strategies For the norma] form these constraints simply asserted that these vectors represent probability distributions In this case, the constraints are denved from the following fact If is the sequence for player: leading to an information set al which player i has to move ' and m_1 , m* are the possible moves at that information set then we must have that $x_\sigma = x_{\sigma m_1} + \quad + x_{\sigma m_k}$ The only other constraints are that the realization weight of the empty sequence is 1 (because the root of the game is reached in any play of the game) and that $x_\sigma \geq 0$ for all *r*

Note that the sequence form is at most linear in the size of the game tree since there is at most one sequence for each node in the game tree, and one constraint for each information set Furthermore, it can be generated very easily by a single pass over the game tree The format of the sequence form resembles that of the normal form jn many ways and it appears thai many normal form solution algorithms can be converted to work for the sequence form The work of [Koller *et al* 1994] focuses on the two playercase They provide sequence form variants for ihe best normal form algorithms for solving both zero-sum and general two player games The resulL which is of most interest lo us is the following

Theorem 4 1 *The optimal strategies of a two player zero sum game are the solutions of a linear program each of whose dimensions is linear in the size of the game tree*

The matrix of the linear program mentioned in the theorem is essentially the sequence form The resulting matrix can then be solved bv *any* standard linear programming algorithm such as the *simplex: algorithm* which is known to work well in practice We can also use a different linear programming algorithm whose worst-case running time is guaranteed lo be polynomial Hence this theorem is the basis for an efficient polynomial time algorithm for finding optimal solutions lo two player zero sum games

5 Experimental results

The sequence-form algorithm for two-player zero sum games has been fully implemented as part of the Gala system The system generates the sequence form creates the appropriate linear program and solves it using the standard optimization librarv of CPLEX We compared this algorithm to the traditional normal form algorithm by using GAMBiT lo convert the game trees generated by Gala to the normal form, and CPLEX lo solve the resulting linear program We experimented with two games the simplified poker game described in Section 2 increasing the number of cards in the deck and an *inspection game* which has received significant attention in the game theory community as a model of on site inspections for arms control treaties lAvenhaus et *al* 1995] The resulting running times are shown in Figure 3 They are as one would expect in a comparison between a polynomial and exponential algorithm

These results arc continued for the sequence form in Figure 4 (It was impossible to obtain normal-form results for the larger games) There we also show the division of time be iween generating the sequence form and solving the resul ting

This formulauon requires that the players never forget their own moves or information they once had This implies that there is at most one sequence *o* leading lo this information set

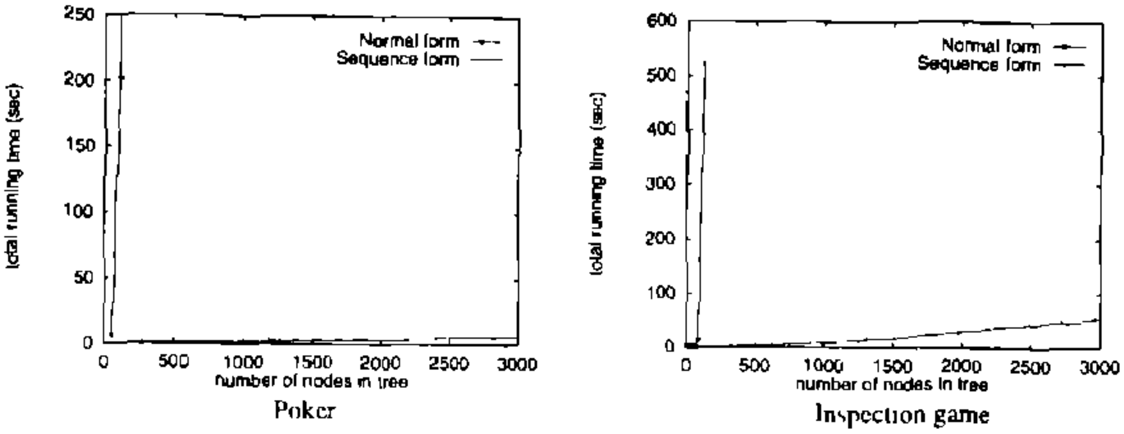

Figure 3 Normal form vs sequence form running time

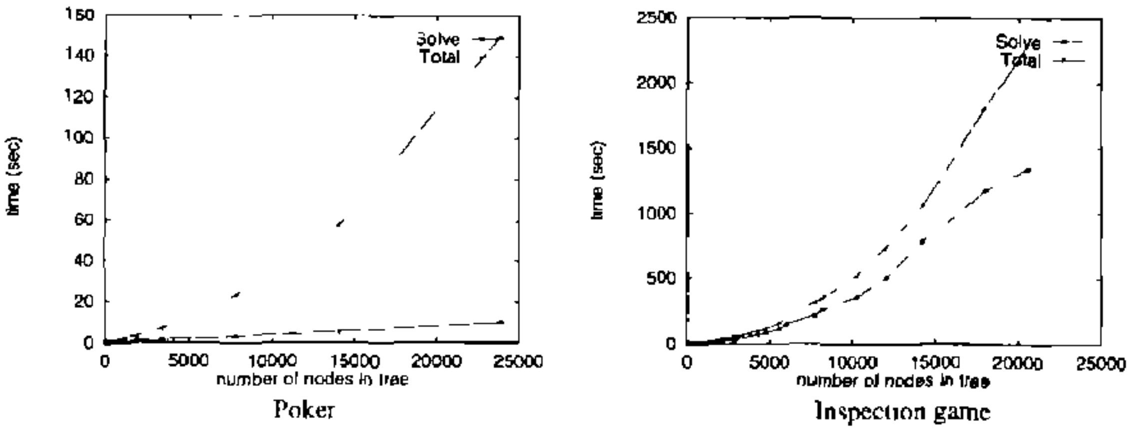

Figure 4 Time for generating and solving the sequence form

linear program Tor the poker games we can see that generating the sequence form lakes the bulk of the time Solving even the largest of these games lakes less than 10 seconds Tins leads us to believe Ihal these techniques can be made to run considerably faster by optimizing the sequence form generator Finally note that the algorithm is much faster for poker games than for the inspection games In the lull paper we explain these results and define certain characteristics of a game Ihal lend to have a significant effect on the running time of the sequence-form algorithm

As we remarked above the final component of the Gala system reads in the strategies computed by this algorithm and interprets them in a way that is meaningful with respect to the game In particular it allows the strategies to be examined by the user who can then use them as part of ihe decision making process We have discovered that examining these strategies often yields interesting insights about the game Figure 5 shows the strategies for both players in an eight card simplified poker Consider the probability that the gambler bets in Ihe first round n is fairly high on a 1 somewhat lower on a 2 0 on the middle cards and then goes up for the high cards The behavior for the low cards corresponds to bluffing a characteristic Ihal one lends to associate with the psychological makeup of human players Similarly after seeing a pass in the first round the dealer bets on low cards with very high probability Psychologically we interpret this as an attempt lo discourage the gambler from changing his mind and belling on the final round In more complex games we sec other examples where human behavior (eg underbidding) is game-theoretically optimal

6 Discussion

As in the case of perfect information games game trees for full-fledged games are often enormous Although we expect to solve games with hundreds of thousands of nodes in the near future full-scale poker is much larger than thai and it is unlikely we will be able to solve it completely Of course chess-playing programs are very successful in spite of the fact Ihat we currently cannot solve full-scale chess Can we apply the standard game-playing techniques to imperfect information games' We believe that the answer is yes but the issue is noninvial Even the concepl of a subtree' is not well defined in such games For one thing the program cannot simply crcalc the subtree starting at the current state since it does not know precisely which node of the game tree is the actual slate ol the game it knows only that the node is one of those in a certain information set In addition information sets belonging to other players may cross the subtree boundary' as was the case in Figure 1 It is not obvious how to deal with these problems We hope lo address this issue in future work Another approach that may well prove fruitful is based on the observation that there is a lot of regularity in the strategies

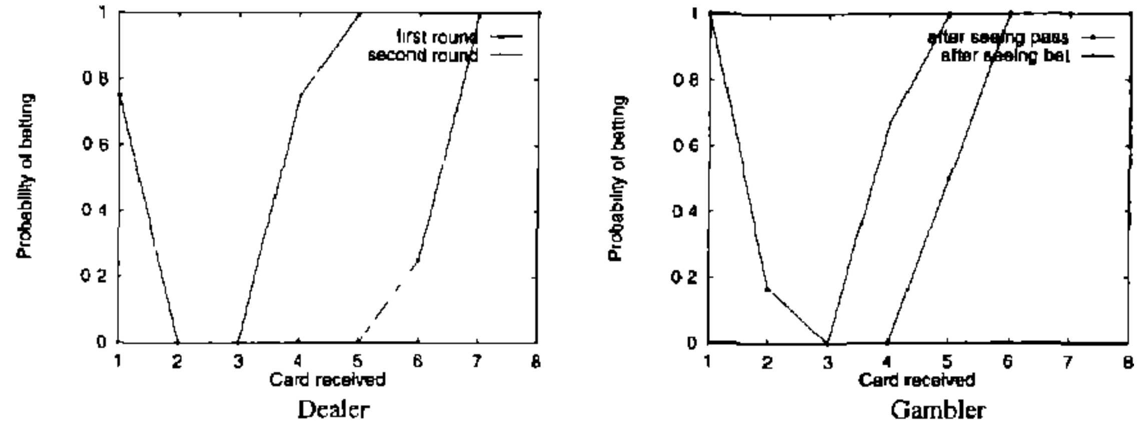

Figure 5 Strategies for 8 card poker

for small poker games the player often behaves the same tor a variety of different hands This suggests thai in order lo solve large games we could abstract away some features of the game, and solve the resulting simplified game completely For the game ot poker we could abstract by partitioning the set of possible deals into clusters and then solve the abstracted game Our experimental results indicate that the resulting strategies would be very close to optimal

Most of the techniques we discussed in this paper also apply to more general classes of games Gala prov ;des the function-alily for specifying arbitrary multi-player games Currently these can only be solved using the traditional (normal-form) algorithms accessed through our GAMBIT interface and these are practical only for small games However the sequence form can be used to represent any perfect recall game and the results of iKoller et al 1994] indicate that many of the stan dard techniques could carry over from the normal form to the sequence form We hope lo use the sequence form approach for more general games and show that the resulting expo nenlial reduction in complexity indeed occurs in practice If so the resulting system may allow an analysis of multi-player games a class of games that have been largely overlooked Perhaps more importantly the system could also be used to solve games that model multi-agent interactions in real life

We believe that the Gala system facilitates future research into these and other questions Its ability to easily specify games of different types and lo generate many variants of each game allows any new approach lo be extensively tested We intend lo make this system available through a WWW sile (http "www cs berkeley edu' "daphne gala') in the hope that it will provide the foundation for other work on imperfect information games

Acknowledgements

We are deeply grateful to Richard McKelvey and Ted Turocy for going out of their way to ensure that theGAMBIT functional-ity we needed for our experiments was ready on time We also thank the International Computer Science Institute at Berke-ley for providing us access lo the CPLEX system We also wish to thank Nimrod Megiddo Barney Pell Stuart Russell John Tomlin and Bernhard von Stengel for useful discussions

References

fAvenhaus et al 1995] R Avcnhaus B von Stengel and S Zamir Inspection games In Handbook of Game Theory Vol 3 to appear North Holland 1995

[Blamr et al 1993] J R S Blair D Mutchler and C Liu Games with imperfect information In Working Notes AAAI Fall Symposium on Games Planning and Learning 1993

[Gordon 1993] S Gordon A comparison between prob abilistic search and weighted heuristics in a game with incomplete information In Working Notes AAAI Fall Sym posium on Games Planning and Learning, 1993

[Koller et al 1994] D Koller N Megiddo and B von Sten gel Fast algorithms for finding randomized strategies in game trees In Proceedings of the 26th Annual ACM Sym posium on the Theon of Computing, pages 750-759 1994

[Kuhn, 1950] HW Kuhn A simplified two-person poker In Contributions to the Theon' of Games I pages 97-103 Princeton University Press 1950

[McKelvey 1992] RD McKelvey GAMBIT Interactive Ex tensive Form Game Program California Institute of Tech nology 1992

[Pell, 1992] B Pell Melagame in symmetric chess-like games In Heuristic Programming in Artificial Intelligence ? — The Third Computer Olympiad Ellis Horwood 1992

[Russell and Norvig 1994] S J Russell and P Norvig Ar tificial Intelligence A Modern Approach Prentice Hall 1994

[Smith and Nau 1993] SJJ Smith and D S Nau Strategic planning for imperfect-information games In Working Notes AAA1 Fall Symposium on Games Planning and Learning 1993

[von Neumann and Morgenstern, 1947] J von Neumann and O Morgenstern The Theon of Games and Economic Behavior Pnnceion University Press, 2nd edition, 1947

[Zermelo 1913] E Zermelo Uber eine Anwendung der Mengenlehreauf dieTheonedes Schachspiels In Proceed tngs of the Fifth International Congress of Mathematicians II pages 501-504 Cambridge University Press 1913