Deceptive Play by Computer Bridge Program

Noriyuki Kobayashi and Takao Uehara

Tokyo University of Technology, 1401-1, Katakura-cho, Hachioji-shi, Tokyo, Japan
noriyuki@ue.it.teu.ac.jp uehara@cc.teu.ac.jp

Abstract. Contract bridge is an imperfect information game. The Monte Carlo simulation is the most popular technique for modern computer bridge programs. The unseen cards are dealt at random, biasing the deal so that it is consistent with the bidding and play thus far. Many deals are generated as samples of Monte Carlo simulation. There are no deals inconsistent with the visible player's own hand. The double dummy result of every move in each deal is evaluated and the best card is selected by averaging over a large number of samples. This paper proposes an algorithm to find a deceptive play that is one of the techniques used by top players. An opponent is modeled as an agent who guesses the player's hand by rule-based knowledge. The deals guessed by the agent may be inconsistent with the player's real hand. The proposed algorithm finds a play line that is the best for the opponent in the deals guessed by the agent and also good enough for the player in the deals consistent with the player's hand. The algorithm is implemented and tested for some typical examples.

Keywords: Computer bridge, Deceptive play, Imperfect information game, Agent, Hypothetical reasoning, Constraint logic programming

1 Introduction

Bridge programs generally have attempted to duplicate human bridge-playing methodology. For example, Nygate and Sterling's PYTHON [1] was a program that could recognize a squeeze (one of the top players' techniques) based on the knowledge in Love's book [2]. Since Ginsberg introduced the partition search technique to GIB in 1998 [3], the brute-force search-based approach has been so successful that few knowledge-based approaches were developed. This paper proposes an algorithm that is the combination of a knowledge-based opponent model and a search-based approach to find a deceptive play (another top player's technique).

The authors proposed an agent model for a bidder [4]. Since bridge auction is a task of imperfect information, each agent has hypothetical reasoning ability and generates images of the other players' hands by abduction from a bidding sequence. The same architecture of the agent is useful as the opponent model in the playing stage of bridge. The knowledge on human playing methodology is built in the agent for guessing the other players' hands from the observed play.

The Monte Carlo card selection algorithm [3] constructs a set of deals D consistent with both the bidding and play of the deal thus far. Every deal in the set D is usually consistent with the player P's own hand that is visible. The best card is selected by evaluating the double dummy result of every move

in each deal. It is impossible to find a deceptive play by this algorithm because opponents can see P's hand in double dummy bridge. Our algorithm to find a deceptive play constructs a set of deals D' guessed by each opponent's agent A. A deal in the set D' may be inconsistent with the player P's own hand. The proposed algorithm searches a play line that is the best for opponents in D' and also good enough for the player's side in D.

The Monte Carlo card selection algorithm is introduced in Section 2. The extension of the algorithm to search a deceptive play is explained in Section 3. The opponent model is described in Section 4. The implementation and experimental result are shown in Section 5 and 6 respectively.

2 Monte Carlo cardplay algorithm

The algorithm shown in [3] is introduced. GIB (Ginsberg's computer bridge program) has a high performance (partition search) engine to analyze bridge's perfect-information variant (double dummy bridge), where all of the cards are visible and each side attempts to take as many tricks as possible. GIB uses Monte Carlo simulation for cardplay (and bidding). The unseen cards are dealt at random, biasing the deal so that it is consistent with both bidding and with the cards played thus far. Then GIB analyzes the deal in double dummy and decides which of the possible plays is the strongest. Averaging over a large number of such Monte Carlo samples would allow us to deal with the imperfect nature of bridge information.

[Algorithm1: Monte Carlo card selection algorithm] To select a move from a candidate set M of such moves:

Step 1. Construct a set D of deals consistent with both the bidding and play of the deal thus far.

Step 2. For each move m ε M and each deal d ε D, evaluate the double dummy result of making the move m in the deal d. Denote the score obtained by making this move s(m,d).

Step 3. Return that m for which Σ_d s(m,d) is maximal.

3 Algorithm to find a deceptive play

At the step 1 of Algorithm 1 a deal d ε D is generated assuming that the player P's own hand is seen, that is, every d ε D is consistent with the real P's hand. In our algorithm to find a deceptive play it is assumed that P's hand is unseen by the other player. A set D' of deals constructed from the standpoint of the opponent A is used for evaluation in step 2 of our algorithm.

The authors proposed an agent model for a bidder [4]. The agent has hypothetical reasoning ability and generates images of the other players' hands by abduction from observed facts based on the rule-based knowledge. Knowledge for generating hypothetical explanation from observed play history is added to the agent. This new agent model is used for generating a deal d' ε D' from the standpoint of the opponent A. Fig.1 is a brief illustration of our idea.

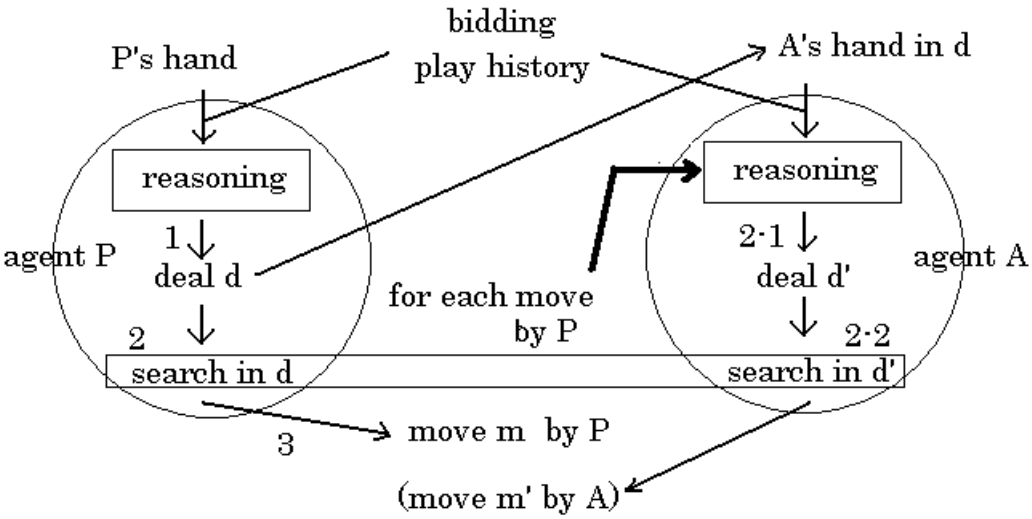

Fig.1 Selection of move by player P based on deal by opponnent A

At step 2 of Algorithm 1 the double dummy result of making the move m ε M in the deal d ε D is evaluated. In our algorithm we assume that opponent A's selection of play line depends on the double dummy result of making the move m' ε M' in the deal d' ε D' where M' is a candidate set of A's moves.

At step 3 of Algorithm 1 the player P selects the move with the highest total score for D. It is the same as that in our algorithm. If the selected move has a low total score for D', then the move is the deceptive play we are looking for.

[Algorithm2: algorithm to find a deceptive play] To select a move from a candidate set M of the player P's moves:

Step 1. Construct a set D of deals consistent with both the bidding and play of the deal thus far from the standpoint of player P.

Step 2. For each move m ε M and each deal d ε D, evaluate the result of making the move m in the deal d and selecting the play line according to the step 2-1 to 2-3.

Step 2-1. Construct a set D' of deals consistent with both the bidding and play including m from the standpoint of the opponent A (the next player).

Step 2-2. For each move m' ε M' and each deal d' ε D', evaluate the double dummy result of making the move m' after m in the deal d'. Denote the score obtained by making these moves s([m,m'],d'). Also evaluate the double dummy result of making the move m' after m where player P's side selects each move assuming the deal d and the opponent A's side selects each move assuming the deal d'. Denote the score obtained by making these moves s([m,m'],[d,d']).

Step 2-3. Select that m' for which $\Sigma_{d'}$ s([m,m'],d') is minimal. Denote the score s([m,m'],[d,d']) obtained by making this move s(m,d).

Step 3. Return that m for which Σ_d s(m,d) is maximal.

4 Opponent model

We assume that opponent players know nothing about a deceptive play. The architecture of an agent with knowledge and hypothetical reasoning ability [4] is used as the framework of opponent model. Knowledge for cardplay is added to that of the bidding system described in [4]. Knowledge of the bidding system is used for both selecting a bid and guessing the hand from the bid. Knowledge of cardplay is not always used for selecting a card. It is used for guessing the player's hand from the card played.

Knowledge of cardplay is divided into 3 types: rules of the game, conventions and heuristics. An example of rules of the game is "Play a card of the same suit as that of the first card of the trick, if it is possible." It is able to deduce that the player has no card in the suit if he played a different suit.

Examples of conventions on opening leads are as follows:

"Lead the King if one has the Ace and the King of a certain suit",

"Lead the King if one has the King and the Queen of a certain suit".

When West leads King of hearts, the agent of South generates the hypothesis that West has Ace or Queen of hearts by abduction based on the knowledge [5].

Examples of heuristics are as follows:

"Play the weakest card if one cannot win the trick as the last player of the trick",

"Play the weakest winning card if one can win the trick as the last player of the trick".

Suppose that a small spade was led by North and a small spade from East, Jack of spades from South and Ace of spades from West followed. In this case the agent of North who has the King of spades generates the hypothesis that West does not have Queen of spades. This type of knowledge in the opponent model is essential for deceptive play.

5 Implementation

Our experimental program is written in constraint logic programming language ECLiPSe. 208 variables representing each player's possession of 52 cards are introduced. The number of each suit and points in a hand are a function of these variables. Goren's bidding system is described as rule-based knowledge. Knowledge of cardplay is divided into 3 types: rules of the game, conventions and heuristics. Each player is implemented as an agent with knowledge and a reasoning mechanism.

Steps to construct a set of deals D consistent with both the bidding and play thus far (in the step 1 of Algorithm 2) are as follows.

Step 1. Construct basic constraints such that each player has 13 cards and each card belongs to only one player.

Step 2. Set the value of each variable according to the fact that each card on the table originally belonged to the player who played the card.

Step 3. Construct constraints by deduction from play history based on rules of the game.

Step 4. Set the value of each variable according to the fact that cards in a player's own visible hand belongs to the player.

Step 5. Construct constraints by hypothetical reasoning from bidding (Discard the constraint when it is inconsistent with existing constraints).

Step 6. Construct constraints by hypothetical reasoning from play history based on conventions and heuristics (Discard the constraint when it is inconsistent with existing constraints).

Step 7. Repeat the random generation of a deal that is consistent with those constructed constraints.

Steps 4 and 6 above must be modified as follows in order to construct a set D' of deals (in the step 2-1 of Algorithm 2).

(Modification of step 4) Set value of each variable according to the hypothesis that cards in the opponent (the next player) A's hand in deal d belong to A.

(Modification of step 6) Construct constraints by hypothetical reasoning from play history including m based on conventions and heuristics (Discard the constraint when it is inconsistent with existing constraints.

Evaluation of the double dummy result of making the move m' after m is implemented based on the minimax search. The player P's side selects each move assuming the deal d and the opponent A's side selects each move assuming the deal d'. Fig.2 shows how to update values where it is assumed that the player P is in tern to play at the root of the game-tree and the type of the root is max..

```
update(MaxMin,Card,Value, [Card1,Value1],[Card1,Value1]):-
  MaxMin=max, Value =< Value1;   MaxMin=min, Value >= Value1.
update(MaxMin,Card,Value,[Card1,Value1],[Card,Value]):-
  MaxMin=max,Value > Value1;   MaxMin=min, Value < Value1.
```

(a) well-known algorithm

```
update2(max,Card,[RValue,IValue],(RRecord,IRecord), (RNewRecord,INewRecord)):-
    update(max,Card,RValue,RRecord,RNewRecord),
    update(max,Card,IValue,IRecord,INewRecord).
update2(min,Card,[RValue,IValue],(RRecord,IRecord), (RNewRecord,INewRecord)):-
    update(min,Card,IValue,IRecord,INewRecord),
% update in the real world d depends on the update in the imaginary world d'
    IRecord=[ _ , IRecordValue],
    (IValue > IRecordValue,   RNewRecord=RRecord;
     IValue < IRecordValue,   RNewRecord = [Card, RValue];
     IValue = IRecordValue,   update(min,Card,RValue,RRecord,RNewRecord)).
```

(b) proposed algorithm

Fig.2 Selection of cards and update of values at a node of a game-tree

update(MaxMin,Card,Value,Record,NewRecord) in Fig.2(a) is the usual update method of a minimax search algorithm[6] where MaxMin is the type of node (max or min) in the game-tree. The Value of a candidate Card is compared with the Record and the selected one is recorded as the

NewRecord.

update2(MaxMin,Card,Values,Records,NewRecords) in Fig2(b) is the proposed update method. The list Values consist of RValue and Ivalue. The RValue is the evaluation of the Card based on the double dummy result in deal d generated from the standpoint of the player P. The IValue is the evaluation of the Card based on double dummy result in deal d' generated from the standpoint of opponent A. The list Records (NewRecords) consist of RRecord (RNewRecord) selected by P and the IRecord (InewREcord) selected by A. At each max node the player P selects a card according to the result in d but the opponent A thinks that P selects a card according to the result in d'. At each min node the situation is more complex. Opponent A selects a card according to the result in d', so player P must evaluate the value assuming A's selection of the card.

6 Example

[Example 1] What card should West play in the example in Fig.3? (The basic idea is in [7])

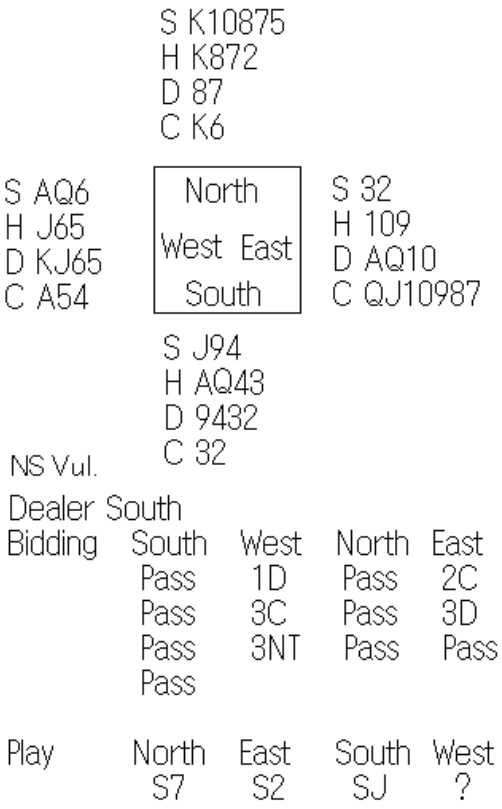

Fig.3 Deal, bidding and play history of example 1

The deal d in Fig. 4(a) is generated from the standpoint of the player P (West) where West's hand is assumed to be visible. The deal d'_1 in Fig. 4(b) is generated from the standpoint of opponent A (North) after West played Queen of spades and led a small club, where North's hand is the same as that in deal d and West's hand is assumed to be invisible. d'_2 in Fig. 4(c) is generated from the standpoint of opponent A (North) after West played Ace of spades and led a small club under the same assumption as d'_1.

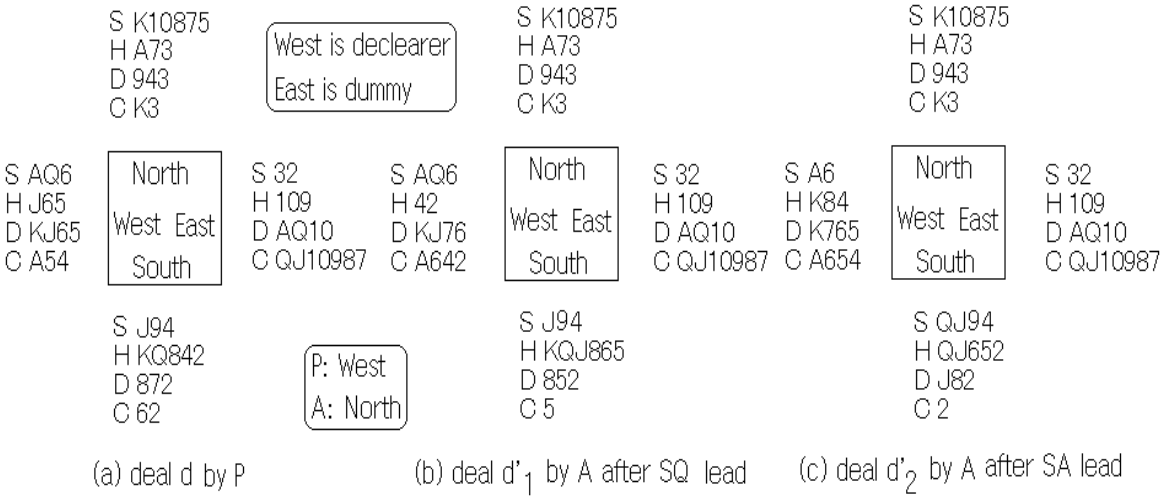

Fig.4 Deals generated by the player P and opponent's agent A

Fig.5 is a part of an application of our algorithm (Steps 2-2, 2-3, and 3) to this example. In this experiment the depth of the search is 3 tricks (or deeper if control is on an opponent's side).

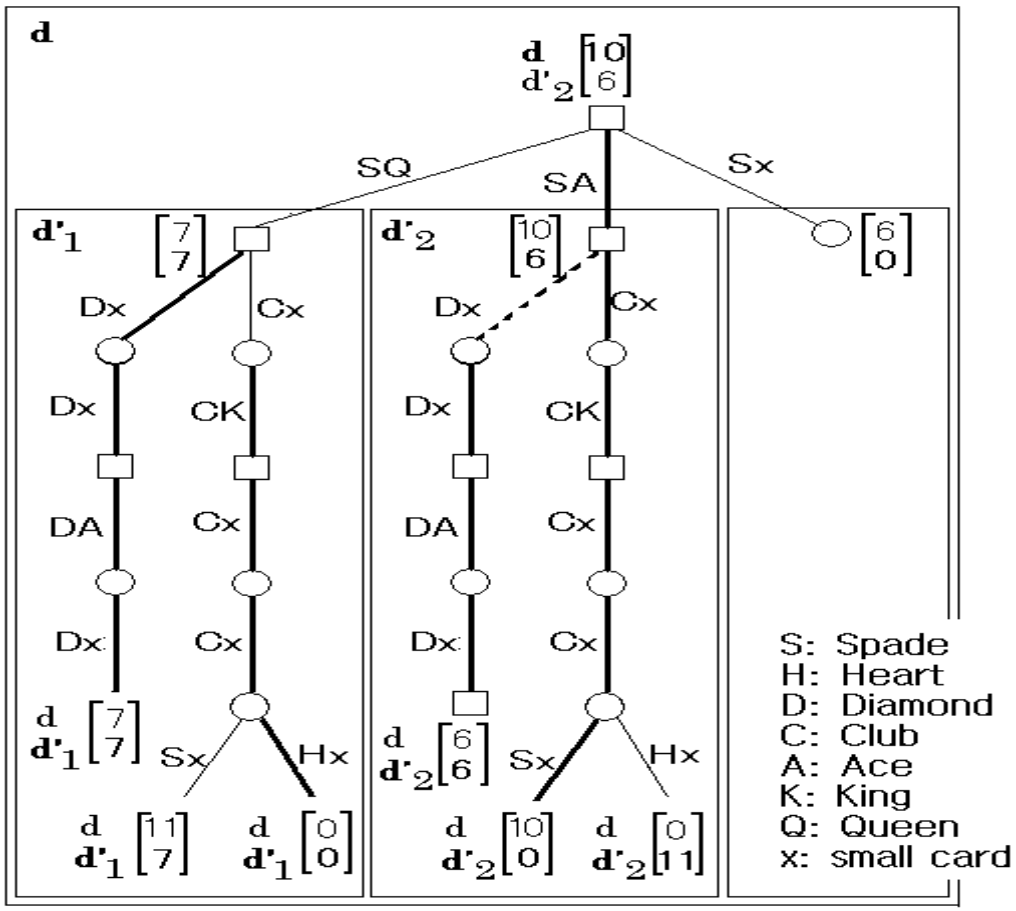

Fig.5 Game-tree search to find a deceptive play

At the leaf of the game-tree, payoff is 0 if the contract fails, otherwise payoff is the number of

tricks the declarer can get by cashing the top cards. As the result of this experiment the player P (West) selects Ace of spades expecting 10 tricks. P guesses that the opponent A (North) may expect 3 down (6 tricks) for this play.

Case 1 (After Queen of spades is played):

Opponent A (North) leads a heart when he gets a trick by King of clubs according to the double dummy result (0: defeat the contract) in deal d'_1 (where no stopper is in hearts). The lead of a heart defeats the contract also in deal d.

Case 2 (After Ace of spades is played):

Opponent A (North) leads a spade when he gets a trick by King of clubs according to the double dummy result (0: defeat the contract) in deal d'_2 (where no more stopper in spade). The lead of a spade gives the declarer 10 tricks in deal d.

In the case of GIB (Version 5.1.7) West played Queen of spades and North sometimes led a heart after North got control by taking King of clubs. When West of our program played Ace of spades North of GIB often took King of spades and led a small spade after North got control. In this case GIB plays in a manner similar to that of our opponent model.

7 Conclusions

The algorithm to find a deceptive play, one of the top players' techniques, is proposed. An opponent model is implemented and the algorithm is tested for typical examples successfully. It takes more than several minutes on our 800MHz computer even for very small D and D'. It is necessary to introduce various techniques, such as the zero-window search, in order to get better performance. We are planning to use this program only when we cannot find any play making the contract as the declarer (or defeating the contract as a defender). Examples of deceptive play by a defender are also tested.

8 Acknowledgements

The authors thank PARC of Imperial College for giving the license to use the constraint logic programming language ECLiPSe.

References

1. Sterling, L. and Nygate, Y.: PYTHON: An expert squeezer, Journal of Logic Programming, 8, 21-40 (1990).

2. Love E. R.: Bridge squeezes complete, Dover (1959).

3. Ginsberg M. L.: GIB:Imperfect information in a computationally challenging game, Journal of Artificial Intelligence Research (2001).

4. Ando T. and Uehara T.: Reasoning by agents in computer bridge bidding, Computers and Games 2000 (2000).

5. Uehara T.: Application of abduction to computer bridge, Systems and Computers in Japan, Vol.26, No.9, pp.23-33 (1995).

6. Sterling, L. and Shapiro E.: The art of Prolog, The MIT Press (1994).

7. Gardner, N. and Mollo V.: Card play technique: Or the art of being luckey, Batsford Books (2001).

8. Kelsey H.: The tricky game: Deceptive play to winning bridge, Devyn Press (1982).