Machine Learning of Bridge Bidding

Dan Emmons

April 1, 2009

Abstract

The goal of this project is to create an effective machine bidder in the card game of bridge. Factors like partial information and the multiplicity of the meanings of bids make this task difficult. This research proposes to overcome these problems with the use of Monte Carlo sampling method for overcoming the limitation of partial information and a tree structure of constraints paired with sets of actions to store the bidding system used by a partnership. With this tree structure a machine partnership trains by continually swapping their new bidding inclinations to learn new decision networks. The performance of bidders is evaluated by having them play against a control pair in both directions for each hand and converting the results to an average IMP gain per hand. The results of this project will not only demonstrate the feasibility of having a machine learn to bid in this manner, but also may develop new bidding conventions useful to human bridge players.

Contents

1 Introduction

Games such as bridge are frequently used as test of and a medium for developing intelligent decision-making algorithms. So far the bidding phase of a bridge hand as proved to be very difficult for machines to perform well for several reasons. Unlike many other games, bridge involves interactions with both cooperative and opponent agents. Also, only part of the total information is available to each agent while bidding, making it difficult to evaluate the effectiveness of given bid until the entire bidding sequence is complete. Perhaps the greatest challenge of all is that bids made by human players usually have more than one meaning. Bids often times suggest contracts to play, but just as often are made to provide the bidder's partner with more information about the bidder's hand. Sometimes they even direct the bidder's partner about what to do while defending. Because of the limited amount of information available to each bidder it can be difficult to determine which of the meanings was intended, or if multiple where intended. The limited information even makes it hard to decide what information would be most useful to give to one's partner about their hand (Graff). All these issues would seem to indicate that machine bridge bidding requires a very different category of solution from solutions to problems in other games.

The goal of this project is to create a machine bidding agent as skilled as possible. This will be done using a flexible bidding system framework that can evolve through co-training over the course of many hands. The important developmental points in this project will be designing this framework or choosing from among those already designed, choosing an effective method to overcome partial information, and finding an effective way for partnerships to continually exchange information about their bidding tendencies so that they might improve through co-training. Progress will be measured using the average change in IMPs scored per hand in simulated team matches. This project will not deal with the playing phase of a hand beyond what is required to evaluate the performance of bidding agents.

2 Background

Bridge bidding is far from being a solved problem. Exceptional bridge playing programs at best earn an average negative one IMP gain against experts, less than the average gain for skilled human players against experts. Compared to the average IMP gain of novices against experts of about negative two IMPs, this figure is not terrible, but it still indicates that the current state of the art in this subject has much room for improvement (Ginsberg). However, a few techniques have been shown to be more successful than others, and those techniques are the compounded to form the bidding system this project sets forth.

A great many computer bridge systems have adopted a Monte Carlo sampling method to overcome the limitation of partial information (Amit and Markovitch, Ginsberg). Using this method a bidder would generate a random sample of many hands that satisfy the constraints already placed upon each hand by the bidding and the known thirteen cards that must lie within the bidder's hand. Actions are evaluated based on these sample hands. Probabilistically, the bidder is evaluating based on what are close to the actual percentage chances of the scenario lying that way, so this method effectively allows the machine bidder to "imagine" the results of their actions based on the expected card distribution (Ando et all).

Double-dummy solving is generally considered to be a solved problem at the current state of the art in machine bridge bidding. Many techniques are known to be very effective in solving within large search trees within little time. The hash table, quick trick checking, and single suit analysis methods optimizations are at this point known to be helpful in solving quickly (Chang). These and other improvements have been adopted for the requisite double-dummy solver in this project in order to be able to measure the improvements in machine bidding agents at a more reasonable pace.

3 Development

3.1 Double-Dummy Solver Implementation

The development of a fast double-dummy solver was the first step taken in this project. This tool is necessary for two purposes. The first is evaluating the performance of bidding agents at the completion of an auction, and the second is for the bidders to analyze the final state of an auction and determine that states utility for themself.

3.1.1 Algorithm Overview

The algorithm used to count the number of tricks available in a contract is based on the standard mtd(f) algorithm, a type of adverserial search used to improve upon the minimax algorithm with alpha-beta pruning. The algorithm is given the remaining cards of the hand, the cards that have been played so far in the current trick, a goal trick count to reach, and the number of tricks taken so far. It will recursively call itself with many different cards from its hand having been played each time. The algorithm is initialized first with the goal number of tricks being 7, and subsequent calls are made that shift the goal to the midpoint of the range that is known to contain the goal.

If the algorithm is run on a hand position in which the opponents have taken too many tricks to allow the declarer to reach the goal, the algorithm returns to its caller that the goal number of tricks cannot be reached from the hand position. If the algorithm is run on a hand position in which the declarer has already reached his goal number of tricks the algorithm returns to its caller that the goal number of tricks can be reached from the hand position. If a defender sees just one position in which the declarer cannot make the goal number of tricks, he prematurely returns that the goal cannot be reached. If the declarer or his partners see a position in which they can reach the goal number of tricks, they prematurely return this result. This ability to easily form alpha-beta cutoffs due to the zero-window search of the algorithm is what gives it its speed gain.

3.1.2 Improvements on MTD(f)

This algorithm runs far too slowly to be able to evaluate a thirteen-trick hand, so many improvements and optimizations were made in order to facilitate this. The first optimization introduced was the use of a hash table. This hash table stores hand positions as keys and maps them to the known upper and lower bounds of tricks that the declarer can make from that position. This prevents positions that can be reached by many lines of play from being re-evaluated, which saves a very significant amount of time. It also in some cases prevents re-evaluation of positions when the algorithm is initiated with a different goal value.

The hash table improves speed significantly, but quickly depletes memory when evaluating a full thirteen-trick hand. The most memory-consuming part of an entry is the hand position, so finding a way to generate a unique numerical value for each hand to store in the hash table vastly reduces the memory cost per entry. An even better method is to take advantage of this value so that hands that will have the same trick result have a hash collision, making the second one adopt the trick score of the first without even evaluating it. The solution implemented uses a character array. The first four bytes hold the locations of the top four cards from each suit. Each pair of two bits holds the number of the player that holds the corresponding card. The next eight bytes hold the lengths of the suits of each player, using each set of four consecutive bits to hold the suit length for the corresponding player (although in almost every case only three bits are needed). The last byte contains the player who must play the next card. This character array is then converted into an integer by using the hash code function of strings. This function does permit some collisions but is erratic enough in its distribution of values across the hash table that it is impossible for two hands with the same associated value to be transformed into each other by any legal line of play, so unwanted collisions will never occur regardless.

In very simple but effective searching optimization is to search in a breadth-first manner when the declarer is only one trick away from either reaching or failing to reach is contract. This breadth-first searching is done with each player only checking the highest card they can play in each suit. Searching in this manner reduces the average search time by a factor of two, because it prevents the computer searching for an elaborate line of play to defeat a contract by more tricks in the future when there is a simple line of play to defeat it immediately.

The final optimization used to speed up the process of double-dummy solving is ordering card choices according to the likelihood that they will benefit the player. This allows an experienced developer to give the computer simple bridge rules he or she knows to be usually correct in his or her own experience with the game. These help search times by more frequently forcing cutoffs of the search prematurely by finding a line of play that either reaches the goal number of tricks or prevents reaching it, depending on the player that must play next. It is not necessary that these rules be correct in every situation because the computer will continue on with the search at these nodes if such a line is not found. Examples of these principles include covering the opposing team's honors with your own, playing low to a trick if you can't beat your partner's card, and if playing last to a trick take it with the minimum possible card.

3.1.3 Game Tree Pruning Techniques

When compounded with the above techniques, pruning the search tree dramatically reduces evaluation time by reducing the average branching factor of the search tree. Great care must be taken in this excercise not to prune nodes that can change the minimax value returned by the search. This is a tricky excercise in the game of bridge because unusual and counter-intuitive plays frequently generate the best results. For this reason, only two pruning techniques are used, both being very safe in regard to their ability to change the end trick score.

The first of these techniques is that a card play is not tried if another equal card has already been checked. An example of this would be if a player has both the jack and queen of diamonds; only one needs to be checked because the cards are for all intents and purposes equal-valued. Another example would be if a player had both the ten and the queen of diamonds and the jack had already been played. This technique can never change the end trick value because the moves that it avoid playing are easily proven to yield exactly the same result as at least one other move the player can make.

The other pruning technique is only checking cards that either can beat the current winning card for the trick or are the lowest play option available. This technique cannot be applied to the choices a player has available when they must lead, but in all other scenarios this has the effect of preventing a player from losing to a trick with an unnecessarily high card. For instance, if the seven of hearts is lead, the next player need only check the hearts above the seven and his lowest heart that he or she is holding for his or her next play. This technique, as counter-intuitive as it may seem, is actually capable of changing the end trick score in rarely-occuring situations where transportation is a key factor. However, by playing cards in the proper order this change can almost always be avoided, making this optimization so near to perfect that it is still advantageous to use it anyway for the large corresponding speed gain.

3.2 Structure of Bidding Hierarchy

The bidding hierarchy exists for two purposes. The first is to examine the a hand and the bidding history and determine which bids should be examined as possibilities for that agent to choose. The other purpose is exactly the reverse; it is used to examine a bid and the bidding history and determine what must be true of the hand of the bidder in order to make that bid.

The bidding hierarchy is implemented as a tree. Each node of the tree contains three things: a set of constraints, a set of actions, and a set of pointers to other nodes. The root node contains no constraints and all actions. When a search is made, the search begins at the root node and propogates down every pointer where the constraints of the target node are met by the hand. There are many priorities of pointers, and propogation down each successively lower priority can only occur at nodes where there is no propogation down any higher priorities. To find possible actions given a hand, the union of the sets of actions of the bottom-most nodes reached is taken. To find constraints given a bid, propogation occurs down all pointers and sets of constraints that could lead to nodes with that action are derived. Figure one shows an implementation of this hierarchy.

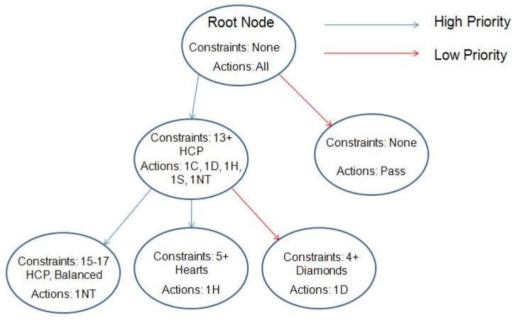

Figure 1: Bid Decision Hierarchy

3.3 Bidding Algorithm

It is vital to the success of the bidders that they be able to use two types of reasoning in their bidding. The first type is the ability to use some set of bidding conventions to form their own communicative language. This is crucial because it allows the agent to form more accuracte pictures of the hands of both its partner and other agents, letting it zero in on the contract that should be played. The second type of reasoning needs to be a substitute for common sense. This is crucial because it will enable the bidding agent to make tough calls when multiple conventions apply or no conventions apply, place the final contract, figure out when to double its opponents, and know when to make deceitful bids. The algorithm outlined in this project handles both of these types of reasoning to help the machine agents to bid to their fullest potential.

When an agent is asked for its bid, it begins by querying the bidding hierarchy to find all of its available bids. Other bids are not even considered because they would slow the search a great deal and they are nearly always worthless. The agent then examines all the constraints on the hands of other bidders so far due to the bidding, and generates a large pool of hands that fix its own cards. Each type of constraint has a function that examines a hand and returns a value representing how well that constraint is matched. The lowest value is zero for a perfect match, and the highest value differs by constraint. For each deal in the pool a linear combination sum of all the evaluations of these functions is computed, and the deals with the lowest such values are used as a representative sample of how the other hands will lie. The weights of the linear combination are one for constraints imposed by the opponents and two for the constraints imposed by the cooperative agent.

The reason for finding a representative sample of hands in this manner is threefold. First, it is possible that a strange sequence has occured that over-constrains the hand. In this case the agent does not want to spend forever looking for hands that match the bidding because none exist; this algorithm will simply yield a variety of close solutions. The second reason is that opponents may occasionally attempt to deceive the bidder by making false bids that will cause the opponent to grossly missrepresent the hand. With this algorithm the linear combination weights the constraints imposed by opponents less heavily than those imposed by the cooperative agent, who is presumably trying to bring to light the truth to the bidder in these situations. The final reason is that generating only hands that satisfy all the constraints, assuming such a set of hands exists, will typically change the expected values of attributes of the hands unrelated to the constraints, creating an unrepresentative sample. By generating generic hands and only taking the ones that match the scenario this algorithm avoids that pitfall and creates a sample with probability distributions that are similar to the corrent distributions.

Once the bids and samples are obtained, a lookahead is performed in the auction is performed for each sample assuming the hands of everybody to be the same as in the sample. The lookahead is performed using a probabilistic minimax algorithm. Alpha-beta pruning cannot be used in this instance of the minimax search because it is crucial to evaluate all branchs of the tree to get a representative sampling of how hands will likely appear; in other words, the search must return not just the best score, but must return all an expected value that is computed by multiplying all scores by a probability estimate for that score. When the terminal nodes are reached that signify the end of the auction, the value returned is the score earned for the declarer. When the bidder has performed this lookahead for each bid and sample pair, the value assigned to each bid is the average of the values obtained from searching all the samples after making that bid. The bid with the highest value is selected. This effectively serves the reasoning function that serves as a replacement for common sense, while the bidding hierarchy performs the function of the conventional reasoning that allows partners to communicate.

While this approach is tremendously powerful, it has one downside, the significance of which is yet to be determined. This method causes all players to play more conservatively than they otherwise might, which is sometimes a virtue but sometimes a hindrance. The reason for this is that as part of the nature of the lookahead algorithm used, the agent assumes that all other agents can see its cards. This assumption is highely unrealistic and causes the agent to handle competitive auctions diffirently than it otherwise would, because it assumes it will be doubled for every contract that it bids that does not make. One method to avoid this problem is to generate a new set of samples at every step in the lookahead, but this is far more time consuming. Because the most costly operation is double-dummy solving contracts by several orders of magnitude, the former implementation completes the task in approximately O(1) time and the later in O(a^n) time, where a is the number of samples generated per step for the later implementation and n is the depth of the look-ahead in the auction. A solution that has neither drawback has been sought for but not yet found.

4 Performance Analysis

4.1 Double Dummy Solver Tests

The double dummy solver has been tested by running it many times to obtain an average time for analysis after each adjustment to the code. Testing for correctness has been done by having skilled bridge players examine the hand until they have figured out the answers. This is very time consuming but is the only way to know for sure whether or not the program is correct. A program that was more frequently correct was always judged to be superior to a program that was faster.

Here is output data from a sample run of the double-dummy solving program. The values that it displays are correct. There are twenty output tricks scores, corresponding to the five different trump suits available (the fifth suit being the lack of trump) and the four possible declarers for each trump selection. The solving process took approximately one second overall in order to solve for all cases:

```
                North:
                Clubs: T 7 5 3 2
                Diamonds: J
                Hearts: A Q J T
                Spades: T 9 7

West:                           East:
Clubs: 6                        Clubs: A J 8
Diamonds: A K T 7 5             Diamonds: Q 9 8
Hearts: 9 8 4                   Hearts: 5 3
Spades: Q J 6 2                 Spades: A K 8 5 4

                South:
                Clubs: K Q 9 4
                Diamonds: 6 4 3 2
                Hearts: K 7 6 2
                Spades: 3

            Trick Counts by Declarer:
            (North, South, East, West)
                Clubs: 9 9 3 3
                Diamonds: 2 2 11 11
                Hearts: 7 7 3 3
                Spades: 0 0 11 11
                No Trump: 2 2 8 8
```

4.2 Bidding Agent Tests

Because of the initial lack of sophisticated agents for the bidding agents of this project to compete with in tests, the agents in their current rendition were tested against opponents that randomly choose among any of their valid bids. The victory for the reasoning agents was dramatic, as expected. Even with a completely empty bidding hierarchy, leaving the agents entirely up to their own reasoning capacity and making inferences about the hand of their partner exceedingly difficult, the agents earned an average IMP gain per hand of approximately twenty-one IMPs. The statistic was measured over eight deals. This conclusively shows that the reasoning abilities of the agents in this version is effective in some capacity, although it is difficult to accurately state the magnitude of this effectiveness because of the huge gap in skill between the reasoning agents and their opponents.

5 Results

The implemented bidding agents have performed spectacularly against random bidders, proving that the presented reasoning algorithm improves the quality of bidding. However, the bidding hierarchy used by the agents during this test contained only a root node that listed all actions as available, so the agents were not able to constrain each other's hands in any way other than assuming that the prior bids they had made where the correct expected value bids. Even with this huge detriment of not understanding each the bids of the cooperative bidding agent, the presented algorithm still managed to reason its way to a sizeable victory. With such a bidding tree consisting of just the root node, this algorithm frequently stumbles its way into the proper contract, sometimes get spectacularly lucky and sometimes performing at shamefully-low level.

One average it performs reasonably, but the auctions that it bids are always awkward, as in this deal where South gets rather lucky:

```
Dealer: West
Vulnerable: None

       North
       Clubs: A K 7 6
       Diamonds: J T 8 4
       Hearts: Q T 8 3
       Spades: 2

West                East
Clubs: 9 8 5 4      Clubs: J 2
Diamonds: 9 7 6     Diamonds: A Q 2
Hearts: J 2         Hearts: A K 9 7 6 4
Spades: 8 7 6 3     Spades: K 9

        South
        Clubs: Q T 3
        Diamonds: K 5 3
        Hearts: 5
        Spades: A Q J T 5 4

South   West    North   East
        Pass    Pass    Pass
2S      Pass    3H      Pass
3S      X       4C      Pass
4S      Pass    4NT     Pass
5C      Pass    5H      Pass
5S      X       Pass    Pass
Pass

5SX Nonvul - South
Making Exact
Score: 650
```

However, with the addition of some of the commonly used bidding conventions to the bidding tree, the program performs in a much more reliable manner. The mean score obtained by the program is increased dramatically, and the standard deviation is significantly reduced. The ability to make inferences about the other hands at the table based on these conventional bids is clearly valuable to the program. Here is a sample hand were the program gets a result of roughly the same magnitude as before, but in this case it is quite evident to those who know about bridge bidding that the bidding agents are extremely skillful.

```
Dealer: West
Vulnerable: E-W

       North
       Clubs: Q J 8 4 3
       Diamonds: K T 4
       Hearts: K J 9
       Spades: K 7

West                East
Clubs: T 6 2        Clubs: 5
Diamonds: A 5 3 2   Diamonds: 9 8 7 6
Hearts: 8 6 2       Hearts: Q 5 3
Spades: Q 6 2       Spades: A J 9 4 3

        South
        Clubs: A K 9 7
        Diamonds: Q J
        Hearts: A T 7 4
        Spades: T 8 5

South   West    North   East
        Pass    1D      Pass
1H      Pass    2C      Pass
3NT     Pass    4C      Pass
4H      Pass    Pass    Pass

4H Nonvul - South
Making Exact
Score: 420
```

Nearly all human players would not be able to match this result on this hand, as it is generally a bad idea to play with a trump suit where the partnership holds fewer than eight cards. However, the program gathered enough information to conclude that it was more likely that the partnership would have no spade stopper than it was that they would loss many tricks for playing in a short trump fit, and made the unorthodox choice of four hearts as its contract, which yields the best result possible against opponents of any reasonable skill level. In short, by using common bidding conventions the agents can perform at a very high level. The future addition of more such conventions would be a simple way to improve the agent's bidding quality further. A further step beyond that would be for the agent's to have the capability to train together to inductively produce their own conventions, automating this process and potentially yielding interesting conventions for use by human bridge players.

6 References

Amit, A., and Markovitch, S. (n.d.). Learning to Bid in Bridge. Retrieved October 21, 2008 from http://citeseerx.ist.psu.edu/viewdoc/

Ando, Takahisa; Sekiya, Yoshiyuki; and Uehara, Takao. (1998, October). Partnership Bidding for Computer Bridge. Retrieved November 21, 2008 from http://portal.acm.org.mutex.gmu

Chang, M. (1996, August 1). Building a Fast Double-Dummy Bridge Solver. Retrieved September 29, 2008 from http://citeseerx.ist.psu.edu/viewdoc/

Ginsberg, Matthew L. (n.d.). GIB: Steps towards an Expert-Level Bridge-Playing Program. Retrieved January 22, 2009 from http://citeseerx.ist.psu.edu/viewdoc/

Graff, Mark. (n.d.) KIBITZ: The Design of a Bridge-Bidding Program Utilizing Simulated Human Judgement. Retrieved December 20, 2008 from http://portal.acm.org.mutex.gmu.edu/

A Rules of Bridge

The card game bridge is exceptionally difficult to learn, although the rules are simple. The game uses a standard deck of playing cards, with an equal number of cards dealt randomly to each player. Bridge must be played with exactly four players; neither more nor fewer players are permissible. Each player is partnered with the player across from them and opponents with the other two players. A hand of bridge is played in two phases: the bidding and the playing. When playing the game bidding comes first, but it is necessary to have an understanding of the play before attempting to understand bidding.

The play consists of thirteen tricks, where each trick is a grouping of one card from each player. A trick commences with one player leading any card in their hand. Each subsequent player going around the table clockwise must play a card of the same suit from their hand. If they have no cards of the suit that was lead, they may play any card they wish. The player that wins the trick is the one who plays the highest card of the suit that was lead for the trick. There is also may or may not be a trump suit. If there is, the highest trump card played in the trick wins irrespective of whether trump was the lead suit or not. The player who leads for the next trick is the winner of the previous trick.

In the bidding, sometimes referred to as the auction, each partnership must decide which trump suit it wishes to play in and how many tricks they think they can win between the two of them. The first person to bid is the dealer, and the bid moves clockwise around the table. On your turn you may bid a contract, pass, double, or redouble. A contract consists of a number and either a suit or no-trump. If a contract is bid it must be higher than all previous contracts. A contract is defined to be higher than another contract if either its number is higher or the numbers are equal and its suit is higher. The order of suits from lowest to highest is clubs, diamonds, hearts, spades, and lastly, no-trump. The suit represents what you wish to be the trump suit, and the number is the number of tricks the player thinks that him and his partner can take together minus six. The minimum bid is one, corresponding to seven tricks, and the maximum bid is seven, corresponding to all thirteen tricks. A double can only be bid if the previous contract was bid by the opponents. The double increases the number of points the opponents score if they make their contract but increases the number that the doubling partnership gets if they fail to make their contract. A redouble increases these two values even more. A redouble may only be bid if the previous contract was bid by the redoubling partnership and the contract has already been doubled. The bidding sequence ends after three consecutive passes, unless the first three bids are passes, in which case the fourth player must pass to end the bidding.

If everybody passes both sides receive score zero for the hand. Otherwise the partnership that made the last contract must play the hand and take that many tricks. The first person of that partnership to have bid the suit that ended up as trump is called the declarer. Their partner is called the dummy. The other two players are called the defenders. The person to the left of the declarer leads for the first trick. Immediately following this lead the dummy lays down all their cards so that everybody can see, and the declarer plays both hands attempting to take as many tricks as possible. The defenders try to prevent this.

There is no penalty for taking more tricks than were bid; indeed, taking more tricks will always earn a better score for the declarer. However, in some cases it will reduce the declarer's score to bid too low, as bonuses can be earned if contracts of certain levels are bid and made. The declarer's partner receives the same score as the declarer. The defenders receive the negative of the score that the declarer receives.

B Standard IMP Scoring Method

The method chosen to be used in this project to evaluate the performance of bidding agents is the standard IMP scoring method of Swiss teams bridge games. The reason that this selection was made has to do with how the agents select their next bid. They choose the bid that they believe gives them the best expected score value. This strategy is well-suited for the bridge IMP scoring method that favors trying to make large point gains over a small set of opponents, but not necessarily advantageous for the matchpoints scoring method which favors trying to obtain a point score that is only minimally better than the score of every opponent from a large set of opponents. Therefore the IMP scoring method is more representative of how well the bidders accomplished their aim. On a more practical note, nearly every computer bridge tournament is scored with the IMP method, so adopting it in this project allows it to more easily face competition.

When executed with human players the IMP method requires eight players rather than four. The players divide themselves into two teams of four players, with each team consisting of two partnerships of two players. The two partnerships of a team go to opposite tables to prevent them from overhearing the events occuring at the other table. At one table two partnerships from opposite teams play a deal and record their score result before passing it to the table containing the other two partnerships. The partnerships of the other table play the exact same board, ensuring before they begin that they seat themselves so as to give each partnership the hands that the partnership of the opposing team played at the other table. When they complete their play they add their score result to the score obtained by the partnership of the same team at the other table. This sum will be zero if the events of the hand transpired the same way at the other table, but deviations almost always result in one team receiving a positive sum and the other a negative sum of the same magnitude. The sum is converted to an IMP score via a standard lookup table used by the ACBL which awards more IMPs for larger sums. This IMP score effectively removes the luck associated with the differing strengths of hands that random dealing provides.

Computers, on the other hand, are capable of executing this scoring method with only four bidding agents. For a computer, obtaining the two scores to be summed is as trivial as playing a deal, then having all players rotate left one seat, then playing the deal again. This can be done because the computer player will not remember the events of the previous play, so the players may play again as if the deal is completely new to them.

In this project I use the average IMP gain per hand when playing against a control bidding system as a measure of the skill of the bidders. Some established values for this measure exist that allow the bidders to be compared against the strength of over bidding systems or human bridge players. For instance, it is known that experts typically receive an average IMP gain of two IMPs per hand when playing against novices and that the standard deviation of IMP swings on one board is typically around five and a half (Ginsberg).