Advances in computer bridge: techniques for a partial-information, communication-based game

by

Paul M. Bethe

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

January, 2021

Professor Ernest Davis

Dedication

To my family, for always inspiring me to work harder.

Acknowledgements

First of all, I would like to thank my advisor Ernie Davis for working with me for so many years, and helping me finish up after 10 years of effort. This thesis would not have happened without the many faceted support of my employer, both in time and money. Finally, I am grateful to my wife Lindsay for pushing me over the years to finish, even when life became much more complicated.

Abstract

Bridge is an imperfect information game with elements of competition against opponents as well as cooperation with a partner. Despite the application of many tenets of artificial intelligence, humans have yet to be consistently bested by the computer. This thesis explores AI shortcomings in both the play and bidding phases of the game. In the play, we explore weaknesses in the cutting edge Monte Carlo techniques and explore both inference and learning based solutions. In the bidding, we go beyond existing rule based systems and investigate deep reinforcement learning as a method to learn how to bid.

Table of Contents

Dedication iii

Acknowledgements iv

Abstract v

List of Figures x

List of Tables xii

1 Introduction 1

2 Background 3

2.1 Fundamentals of Bridge . . . 3

2.1.1 Overview . . . 3

2.1.2 Dealing . . . 4

2.1.3 Bidding . . . 4

2.1.4 Contract . . . 5

2.1.5 Play . . . 5

2.1.6 Claims . . . 6

2.1.7 Duplicate vs Rubber . . . 6

2.1.8 Scoring . . . 7

2.1.9 Diagrams and Nomenclature . . . 7

2.1.10 Play Techniques . . . 9

2.1.11 Bidding Methods . . . 14

2.2 Computer Games . . . 17

2.2.1 Simple Search . . . 17
2.2.2 Minimax in Games . . . 18
2.2.3 Alpha-Beta Pruning . . . 20
2.2.4 Transposition Tables . . . 21
2.2.5 Evaluation Functions . . . 21
2.2.6 Imperfect Information . . . 21
2.2.7 Monte Carlo . . . 22
2.2.8 Conformant Planning . . . 22
2.3 State of Bridge Play . . . 23
2.3.1 Partition Search . . . 23
2.3.2 Limitations of Monte Carlo . . . 24
2.3.3 General Bidding . . . 27
2.3.4 Bidding Systems and Meanings . . . 27
2.3.5 Recent Work . . . 27
2.3.6 Current Programs . . . 29
2.4 Advances with Machine Learning . . . 29
2.4.1 AlphaGo . . . 29
2.4.2 Poker . . . 30

3 Improving Move Selection using Monte Carlo 31
3.1 Expert Ability . . . 31
3.2 Monte Carlo Tree Search . . . 33
3.3 Finding Flaws . . . 34
3.3.1 Subtle Error . . . 34
3.3.2 Beginner Mistake . . . 37
3.3.3 Equal is not Equivalent . . . 39
3.3.4 Player Construction . . . 39
3.3.5 Frequency . . . 41
3.3.6 Desired Result . . . 42
3.4 Improvements . . . 44
3.4.1 Sub-simulation . . . 44
3.4.2 Learning . . . 44

3.4.3 Information Score . . . 45
3.4.4 Goal Modification . . . 46
3.5 Experimental Results . . . 46
3.5.1 Computational Performance . . . 47
3.5.2 Validation of Double-dummy Comparisons . . . 48
3.5.3 Information Score Results . . . 49
3.5.4 Using Variance Reduction . . . 49
3.6 Conclusions . . . 50
3.6.1 Future Work . . . 51

4 Uncontested Bidding **54**
4.1 How Humans Bid . . . 54
4.1.1 Skill Evaluation . . . 55
4.2 How Computers Currently Bid . . . 56
4.2.1 Skill Evaluation . . . 59
4.3 Reinforcement Learning . . . 59
4.3.1 Q-learning . . . 60
4.3.2 Deep Q-learning . . . 60
4.4 Learning to Bid Unopposed . . . 61
4.4.1 Definition . . . 61
4.4.2 Representation . . . 62
4.4.3 Rewards . . . 63
4.4.4 Model . . . 63
4.4.5 System to Learn . . . 64
4.4.6 Training Data . . . 65
4.4.7 Experimental Results . . . 65
4.4.8 Conclusions . . . 67
4.5 Future Work . . . 68
4.5.1 Auction Meanings . . . 68
4.5.2 Competitive Encoding . . . 68

5 Conclusions and Future Work **71**
5.1 Conclusions . . . 71

5.2 Future Work 71

Appendix: Scoring in Bridge **73**

Bibliography **80**

List of Figures

2.1 Sample two-hand diagram 7
2.2 Sample defensive diagram 8
2.3 Example double-dummy layout 9
2.4 Simplest finesse 10
2.5 Simple squeeze 11
2.6 Squeezed out of hearts 11
2.7 Squeezed out of spades 12
2.8 Simple safety play 12
2.9 Safety play needed 13
2.10 Safety play: lost trick 14
2.11 Example 2NT auction 15
2.12 Competitive auction 17
2.13 Tic-Tac-Toe with a winning position 19
2.14 Tic-Tac-Toe transposition 21
2.15 Example of strategy-fusion 24
2.16 Strategy-fusion example: after diamond pitch 25
2.17 Strategy-fusion example: North follows 26
2.18 Auctions not the same 28

3.1 Expert robot play 32
3.2 Expert robot squeeze 33
3.3 Subtle defensive error 34
3.4 End-position 35
3.5 Good defense 36
3.6 Incorrect spade pitch 37

3.7 Bad lead 38
3.8 Lead constraints 40
3.9 Random choices by side over 10K deals 42
3.10 Safety play 43
3.11 Double-dummy and Combined Defender compute times 47
3.12 MCTS per-path baseline and enhanced timings 48
3.13 Negative inferences 52
3.14 Negative inferences: alternate 53

4.1 Partscore bidding 55
4.2 Slam bidding 57
4.3 Uncontested auction 62
4.4 Uncontested bit representation 62
4.5 Single bid bit representation 69
4.6 Auction state bit representation 69
4.7 Example encoded auction 69
4.8 Example competitive auction bits 70

List of Tables

3.1 Random choices per deal over 10K 41
3.2 500-path MCTS choices at trick 4 43
3.3 Human defense expectation vs. double-dummy 49
3.4 Effect of the enhanced MCTS 49
3.5 Effect of using standard deviation 50

4.1 Human average IMPs per deal 56
4.2 Expected tricks actual hands 58
4.3 Expected tricks opposite strong NT 58
4.4 Best contracts opposite various hands 59
4.5 Conservative model openings 66
4.6 Conservative model responses 66

A.1 Table of slam bonuses 74
A.2 Table of doubled and redoubled multipliers 74
A.3 Scoring examples of making contracts 74
A.4 Scoring examples of failed contracts 75
A.5 Table of vulnerabilities 75
A.6 Example scores for a single board in 13 table duplicate 77
A.7 Example matchpointing for one direction 77
A.8 Partial IMP table 78
A.9 Sample 8 board match (T1=Table 1, T2=Table 2) 79

Chapter 1

Introduction

Bridge, Chess, Go: the great mind sports of the 20th and 21st centuries, games of strategy and skill that require a lifetime of study and play to master. Fascinated by these games, humans have produced countless articles, books, and newspaper columns devoted to the complexities of these games. It is therefore not surprising that computer scientists found an intriguing problem space to explore cutting edge artificial intelligence ("AI") with a tangible goal of "beating an expert". As a bonus, generalized solutions could often then be applied to an array of other problem spaces. A late addition to the group, Poker also became a game of great interest, for humans perhaps because of the large sums of money changing hands, for researchers more likely for the psychological aspects around bluffing as well as the application to certain types of economic problems.

Until the last years of the twentieth century, computing power was too limited to overcome the complexity of these games. The first breakthrough was only in 1997 when Deep Blue bested Garry Kasparov, a Grandmaster and World Chess Champion. This was the culmination of years of dedication at IBM by some of the top researchers, and it also offered new hope of mastering the other mind sports. And yet they resisted.

It was twenty more years until 2016 when International Go Champion Lee Sedol was bested by AlphaGo. In a result that shocked many, AlphaGo demonstrated the newly available power of combining cloud-computing with cutting edge machine learning models.

Then in 2019 Poker was the next to fall, as Carnegie Mellon's Pluribus out performed professional players in multiplayer no-limit Texas hold'em, just a year after its precursor Libratus was victorious in heads-up play.

So that leaves Bridge. At the turn of the century with significant advances in Bridge on

the sub-problem of double-dummy, there was hope that Bridge would soon follow. Yet, Bridge AI programs continue to suffer from significant weaknesses, particularly in the bidding phase. Artificial intelligence has yet to consistently beat humans; however, the successes of AlphaGo and Pluribus have encouraged researchers to try and tackle this problem.

In this dissertation, I will begin by recapping the history of Bridge AI and present the reader with a series of open problems in the two phases of the game, bidding and card play. Solutions will then be presented and explored with some experimental results.

Code for all the described algorithms will be published under
https://github.com/pmbethe09/Bridge.

Chapter 2

Background

In order to fully appreciate the problems still faced in building computer Bridge players, the reader must have a background in several areas: (i) the card play and bidding phases of Bridge as well as scoring; (ii) general search, adversarial game tree search including minimax and Alpha-Beta pruning; (iii) and from machine learning, Q-learning and deep neural networks. In this chapter we will briefly survey these topics, and then discuss their applications to computer Bridge.

2.1 Fundamentals of Bridge

This Section will provide a whirlwind tour of the basic rules and strategies of Bridge so the computer scientist may better understand the problems being tackled. Scoring is briefly covered in section 2.1.8 and then more thoroughly in the Appendix.

2.1.1 Overview

Bridge is a partnership card game played by four players, using a standard deck of 52 cards; 2 through 10, Jack, Queen, King, Ace in each of the four suits: clubs ♣, diamonds ♢, hearts ♡, spades ♠. The highest cards in each suit A, K, Q, J are referred to as *high-cards*. Compass directions North, East, South, and West are used to identify the players, with the order of the bidding and the play in that order, clockwise around the table. Partners are seated across from each other: either N and S, or E and W and are often referred to as a *pair* or *side*.

A game or match consists of one or more *deals*, a unit of play that has no carryover to the next deal except for overall match score[1]. This is similar to a multi-game chess match in which each

[1]except for Rubber which is covered in scoring

game starts with a blank-slate. In a single deal a.k.a. *hand* or *board*, all of the cards are dealt out (13 per player), and then the bidding proceeds clockwise from the dealer. Hand is also used to describe the cards held by a player for a particular deal.

Once the bidding completes, play commences consisting of 13 separate *tricks* where each player contributes a single card from their hand to a trick. Then a score is entered based on the number of tricks won by each side (see Section 2.1.5) and the players proceed to the next deal.

The goal of each deal is to maximize the score received by your side, which may be dependent on the *scoring format*, and as we will see the bidding plays a crucial role in achieving that goal.

Much more can be found in The Official Encyclopedia of Bridge [Francis et al., 1995], here we cover only the bare minimum needed to follow along.

2.1.2 Dealing

In "rubber Bridge" or home games, the cards are physically dealt out, often rotating more than one deck so as to speed up the time between deals. The player who does the dealing is called the *dealer* and is the first to bid in the *bidding* phase.

In "duplicate Bridge" all the hands to be played are dealt in advance and placed in a physical *board* (used as a synonym for deal) that keeps each player's hand separate; there is also a pre-defined numbering of boards in regards to the dealer and vulnerability. On board 1, North is the dealer, after which the dealer rotates clockwise via East, South, West and so on. The *vulnerability* on each deal is a concept that factors heavily into how a deal is scored. On a given board either side may be vulnerable or not, and this is predetermined in a rotation based on the board number.

When the bidding of a board is started, players take their cards from the slot marked for their direction. During the play of the hand each player plays and turns over cards directly in front of them so as to keep their hand of cards separate from the others, allowing the same cards to be returned to the board in the proper slot. This in turn allows the board to be sent to other tables where players can replay the identical deal.

2.1.3 Bidding

At their turn, a player takes a *call* by choosing from *Pass*, *Double*, *Redouble*, or any legal *bid*. Any player may always *Pass* which is the same as taking no bid; however, *Double* is allowed by a player only when the opponents hold the current highest bid and the contract has not yet been

doubled. *Redouble* is allowed only when the opponents have doubled, and can only be done once for a particular bid.

A legal bid must be *higher* than all previous bids regardless of who made them, noting that the first to bid is unconstrained. A bid consists of one of the four suits as a *trump* suit (covered in section 2.1.5), or Notrump ("NT") coupled with a level $[1-7]$. In order for a bid to be higher than the previous bid, it must either be a higher level or the same level with a higher denomination. The possible trump suit denominations are ranked from low to high: clubs ♣, diamonds ♢, hearts ♡, spades ♠, and Notrump which indicates that there is no trump suit. Thus 1♣ is the lowest possible bid and 7NT is the highest.

The bidding, also commonly referred to as the *auction*, starts with the *dealer* and continues with each player taking a call until there have been three consecutive *Passes*, with the exception that all four players must have the chance to take a call before the auction can end. If the first four calls of an auction result in all four players passing, then the deal is a *passout* and a score of 0 is entered and the players proceed to the next deal.

2.1.4 Contract

Except in case of a *passout*, the result of the auction is a *contract*, held by a particular partnership, which specifies the number of tricks they must win, and assigns either a trump suit or Notrump. The bidding process also determines which of the players in the team winning the contract is *declarer* (the first of the partnership to bid the denomination of the winning contract) and which is *dummy* (their partner); the opposing team is called the *defenders*. By convention, the number of tricks to be won is stated in terms of the number of tricks beyond a six-trick "book"; for example, a contract of 4♠ is a commitment to win 10 (4+6) out of the 13 available tricks with spades as trump.

2.1.5 Play

The player to the left of the declarer leads to the first trick ("the opening lead"), choosing any card in their hand. Thereafter, the winner of the previous trick leads any card they wish to begin the next trick. After the opening lead, the partner of the declarer (*the dummy*) lays their cards on the table for all to see, and the declarer is charged with playing cards from both hands to maximize their side's ability to take the most tricks. Thus the declarer knows the cards in their hand and in dummy's hand, but they do not know how the remaining 26 cards are divided between the

defenders. Each of the defenders knows their own hand and, after the opening lead dummy's, but the defenders do not know which cards are in their partner's or declarer's hand.

In a single trick, the players each play one card in sequence. The first player of a trick (the *leader*) may play any card in their hand. Each of the remaining players must play a card in the suit that has been led (by *following suit*) unless they are *void* (out of cards) in that suit. If they are void, they may *discard* or *pitch* any card in their hand and are said to *show-out* of the suit led. The winner is then determined as the highest card played in the suit led. Or, if there is a trump suit, any player void in the suit led may discard a trump, commonly known as *ruffing* or *trumping*, and in that case, the highest trump played determines the trick winner (as multiple players may ruff on the same trick). In a Notrump contract there is no trump suit, thus discards when void can never win a trick.

Following the last trick, the number of tricks won by the declaring side is compared to the contract, and a score is obtained (see Section 2.1.8).

2.1.6 Claims

At any point during the play, declarer will often speed up the game by claiming all or some of the remaining tricks. This is done by stating a *line of play*, which is an ordering of the cards in declarer and dummy which is guaranteed to succeed despite any unknowns in the distribution of the opponents' cards. If accepted, the deal is immediately over, the cards put away, and a score is entered combining the tricks played and the claimed tricks. In the rare case where a claim is not accepted, the *Director* is called to adjudicate (the Bridge equivalent of a referee).

Occasionally a defender may also claim if they are known to possess a certain number of winning cards in their own hand with the caveat that they cannot speak for their partner or require them to play any ordering of cards.

2.1.7 Duplicate vs Rubber

When making decisions in Bridge in both the bidding and the play, it is important to know the form of scoring in use, as that has significant consequences on expected-value calculations. The two major forms of contract scoring are *duplicate* where each hand is independently scored and *rubber* where there is some score carryover from previous hands. Each then has several variations on how contract scoring relates to match scoring. In modern times, almost all articles involving Bridge are about duplicate. Additionally, almost all of the available data on played hands is from duplicate

team International Matchpoint ("IMP") scoring, thus we ignore rubber for our experiments and focus on duplicate IMPs.

2.1.8 Scoring

The score for each hand is based on the contract and the number of tricks won by each pair. If declarer wins at least the number of tricks that were contracted for, then the pair will earn a positive score on the hand, and the defenders will earn the corresponding negative score. If declarer takes fewer tricks than what was contracted for, the defending pair will earn the positive score on the hand and the declaring pair will earn the corresponding negative score.

An important consideration is that contracts at a high enough level may earn additional scoring bonuses, which becomes part of the nuances of bidding. If a side bids 7 and thus contracts for all of the tricks this is called a *grand-slam* and if made earns the maximal bonus available. The next highest bonus is a a bid of 6, a *small-slam* or *slam*. A *game* bonus is awarded for a bid of at least $3NT$, 4♡, 4♠ and 5♣, 5♢. All lower contracts are called *partscores*. *Overtricks* are any tricks made in addition to the number required, the importance of which depends on the form of scoring.

More thorough coverage of scoring can be found in the Appendix.

2.1.9 Diagrams and Nomenclature

A typical diagram involves 1, 2 or 4 hands. Single hand diagrams ♠K4 ♡742 ♢AQJ6 ♣KQJ6 are most often used to explore bidding or lead problems.

In the play, as described in Section 2.1.5, each player can see the dummy and therefore has knowledge of 2 hands.

A diagram showing two paired compass directions (East and West as in Figure 2.1, or North and South) is used to tackle declarer play problems.

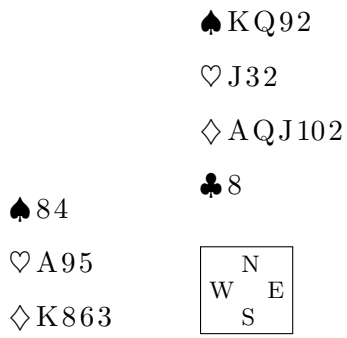
♠AK		♠432
♡AKQ	N W E S	♡432
♢AKQJ		♢5432
♣AKQJ		♣432

Figure 2.1: Sample two-hand diagram

Unseen cards are held by the opponents, in an unknown *distribution*, which is defined as the

number of cards each player holds in each suit. We will use these two-handed diagrams often to give examples. Sometimes these problems also include the bidding, as that makes high card holding and distributional inferences available.

A defensive diagram such as Figure 2.2 shows one defender plus the dummy, and often the auction as well.



South	*West*	*North*	*East*
2♡	P	4♡	P
P	P		

Figure 2.2: Sample defensive diagram

Finally, Figure 2.3 shows a double-dummy diagram where all 4 hands are given, either for compactness of a problem, or to examine exact search possibilities.

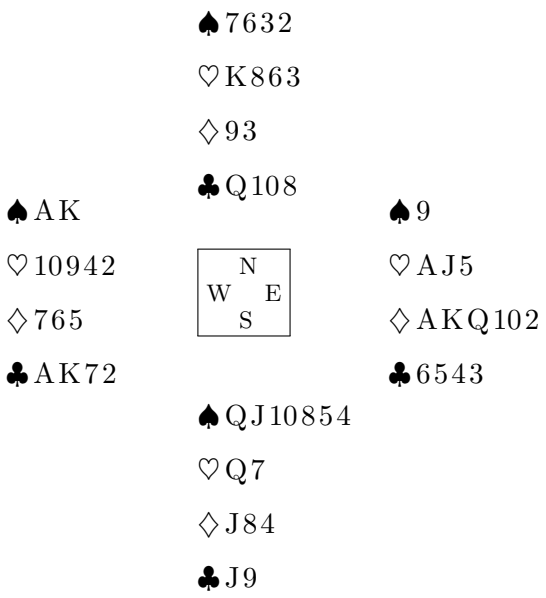


Figure 2.3: Example double-dummy layout

'-' versus '=' It is often useful to speak of an unknown distribution of cards in a suit through shorthand. For example, '3-0 clubs' describes either opponent holding all three missing clubs (and the other none), but with both options available. However, in '3=0 clubs with North', by using '=' the shorthand implies the specific layout of three clubs with North and none with South (so 3-0 is the combination of 3=0 and 0=3). This rule is also applied to player distributions where '-' is generic and '=' can be used in rank order using '♠ = ♡ = ♢ = ♣'. For example, '5-4-3-1' describes any hand with a five, four, three and one card suit; '5=4=3=1' indicates a hand with five spades, four hearts, three diamonds and a single club.

Similar to void, describing zero cards in a suit, a *singleton* is a term for describing exactly one card in a suit, where *doubleton* indicates exactly two cards in a suit.

2.1.10 Play Techniques

A *finesse* is a technical play by any player that attempts to take an extra trick conditional on a certain card being held by a specific opponent. For example, in Figure 2.4, West is on lead in a Notrump contract and leads any spade, with North then *playing small* (shorthand for playing any smaller card in the same suit or any card smaller than the 9). The opponents are already known to hold ♠$KT54$, but it is unknown as to how they are distributed.

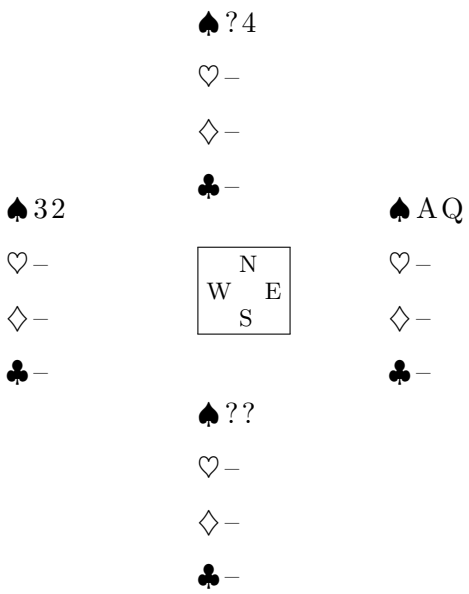


Figure 2.4: Simplest finesse

Of course, while the Ace will always win, in this example playing the Queen will also win whenever the King is with North, but lose when South holds it (as South gets to play the King after seeing the Queen played). If the finesse is successful, the declarer will then *cash* the Ace, playing a card which is guaranteed to take a trick, netting two tricks instead of just one.

A *squeeze* is an ordering of plays which forces one or both opponents to concede one or more additional tricks from the number that were previously available. This most often happens for the benefit of the declarer against the defense, but can on occasion occur the other way. Recognition and execution of a squeeze by a human is considered advanced to expert level play.

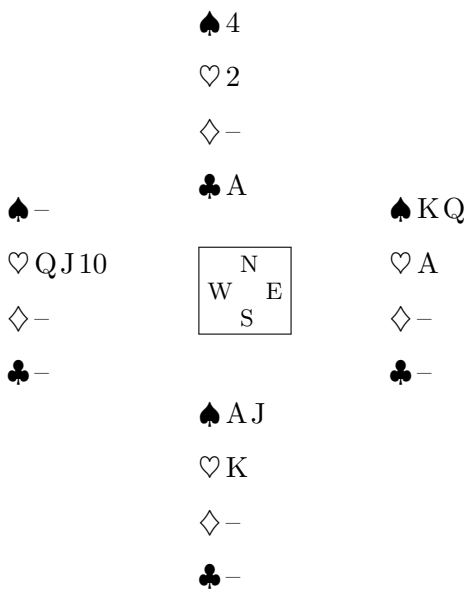


Figure 2.5: Simple squeeze

In Figure 2.5, North on lead in a Notrump contract can count two obvious winners via the ♣A and then South's ♠A. However, observe what happens when North starts with the ♣A, it turns out that East has no winning choice:

- If East discards the ♡A, dummy discards the ♠J and North as in Figure 2.6 can take South's ♡K and ♠A.

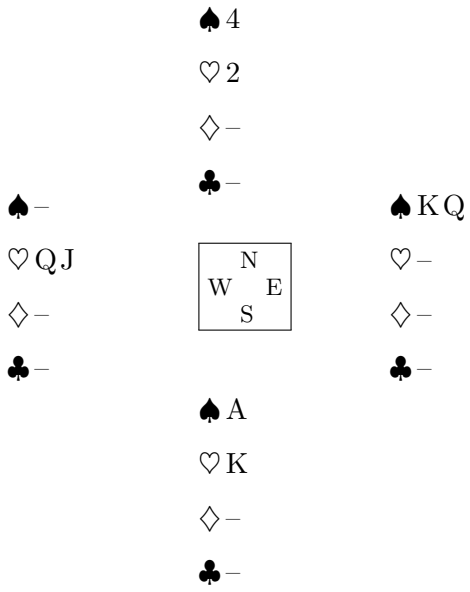


Figure 2.6: Squeezed out of hearts

- If instead East discards the ♠Q (or King), South discards the ♡K; North can now take dummy's ♠A then Jack as shown in Figure 2.7.

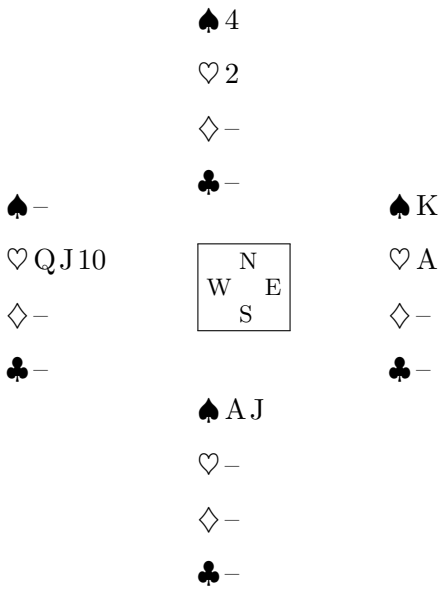


Figure 2.7: Squeezed out of spades

Where it seemed that NS would only take 2 tricks (♣A, ♠A), all 3 are taken by *squeezing* East. There are many different variations on squeezes, the one shown being a *simple positional* [Love, 1968].

A *safety play* is a technique used to guarantee that a certain number of tricks are taken, typically at the expense of a chance to win more tricks. For example, EW playing in a Notrump contract, and ignoring the other suits, declarer needs four tricks from clubs to make in Figure 2.8.

Figure 2.8: Simple safety play

A natural play might be to take the King and then Ace or vice-versa, which leads to 4 or 5 tricks whenever both hands have at least one club. However, look at what happens if the ♣K is taken and the layout is as Figure 2.9; the ♣QJ will take two tricks reducing declarer's tricks to 3.

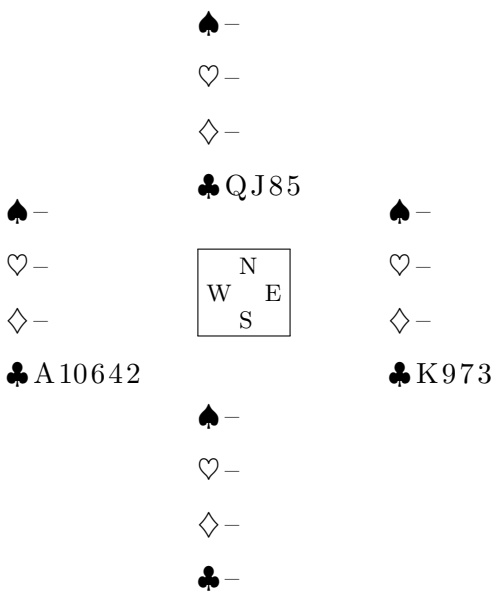


Figure 2.9: Safety play needed

Instead, since declarer wants to guarantee that they win four tricks, they can *safety play* by leading the ♣2 from West towards the other hand, then:

- If North discards another suit, East plays the King and with perfect information can play for 3 more tricks;
- If North plays the Queen or Jack, East plays the King and will take 3 or 4 more tricks; or
- If North plays low (the ♣5 or ♣8), East makes the key play of the ♣9.

It is the last option which is the crucial safety play: in normal layouts of 2-2 or 3-1 such as Figure 2.10, this will often lose a trick to the Queen or Jack that might have been avoided; but if the layout is as Figure 2.9, four tricks will be secured and the contract made.

On this hand the option is symmetric, the same play could be made by initially leading the ♣3 from East planning to play the ♣T if South follows with a low club.

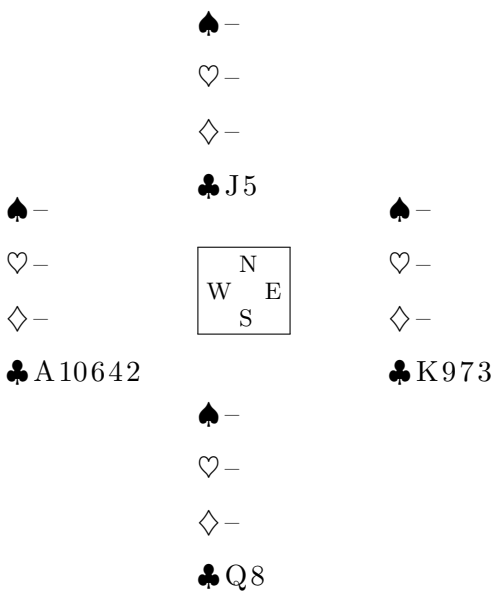


Figure 2.10: Safety play: lost trick

2.1.11 Bidding Methods

The goal of the bidding phase is to maximize your side's score on the deal, with some consideration for the competition's scoring format (see Appendix). This is where the cooperative nature of the game is most on display, as the bidding is typically an affair that occurs over several rounds, a conversation of sorts where a side tries to apply agreed upon rules for selecting bids on the way to the goal.

2.1.11.1 High Card Points

Using High Card Points ("HCP") is a common method (but not necessarily the best method) for evaluating the strength of a hand. Using values of A=4, K=3, Q=2, J=1 made popular by Goren [Francis et al., 1995], one can convert a hand into a number of points so as to apply a set of bidding rules.

2.1.11.2 General

An *opening bid* is the first non-Pass in an auction, and is less about suggesting this exact contract and more about conveying some information about the hand held in both distribution (defined in Section 2.1.9) and strength (the number of high-cards as an initial estimate of the number of tricks that the hand can take).

			North	*East*	*South*	*West*
♠Q9432		♠AK85	P	2NT	P	3♡[1]
♡7	N W E S	♡KJ63	P	3♠	P	3NT
♢97652		♢KQ	P	4♠	P	P
♣Q3		♣A52	P			

[1] Jacoby transfer to 3♠, showing 5+ spades

Figure 2.11: Example 2NT auction

In example 2.11, East opens 2 Notrump, showing 20-21 HCP and a somewhat balanced hand (typically no more than one 2-card suit). While this might be the final contract if everyone passes, the bid is selected to convey information to their partner and allow consideration of whether the side should bid to a game or even try for a slam (see 2.1.8). 3♡ is a *conventional* **Jacoby transfer** (we will discuss conventions next) which shows 5 or more spades and requests a response of 3♠ but also forbids a call of Pass[2]. This is a wide ranging bid whose specifics will be clarified via a subsequent call. 3♠ was bid as requested and then $3NT$ clarifies that West has exactly 5 spades and believes their side may be able to make game[3]. Finally East, aware of the trick-taking power of a 9-card spade fit chooses 4♠ as the contract rather than passing $3NT$.

This simple auction offered examples of:

1. An Opening Bid: 2NT - starting the conversation by describing the hand, unlikely to be the final contract;
2. A Conventional Response: 3♡ - continuing the conversation and requiring a forced 3♠ or limit set of responses;
3. A Choice of Games: 3NT - which completes the two-step description started with 3♡; and
4. A Decision Bid: P or 4♠ - information having been passed, a final choice to maximize expected score.

2.1.11.3 Conventions

While the goal of bidding is to reach the best contract, that does not mean that every bid should be made with the expectation of playing such a contract. As such there are many intermediate

[2]Forbids in the sense of partnership agreement, Pass of course is legal.

[3]the number of tricks for a game bonus

bids chosen to facilitate the passing of information. Some of those bids may be conventions.

A Bridge *convention* is an agreement about a call or a set of related calls which may or may not indicate anything about the suit bid, but rather convey some previously agreed meaning [Francis et al., 1995]. Conventions to be played must be agreed by partners before play begins and must be disclosed to their opponents. For some conventions the disclosure is done by providing a *convention card* indicating artificial agreements, as well as *alerting* the opponents before or during the auction that something unusual may have come up.

Some conventions and bids are barred from Bridge tournaments being regarded as destructive to the game. This is important to mention only in that computers which choose to invent their own system may run afoul of these kinds of restrictions if trying to play against human competition.

When a conventional bid occurs, an opponent at their turn to bid may ask questions of the partner of the player who made the conventional bid.

Examples of various common conventions in use:

- *Stayman*: a bid of 2♣ over partner's 1NT opener which is *forcing*, says nothing about clubs, and *asks* partner to bid 2♡ with 4 hearts, else 2♠ with 4 spades, and in all other cases 2♢.
- *Jacoby transfer*: when partner has opened 1NT (or 2NT), a bid of 2♢/2♡ (3 over 2NT) saying nothing about the suit bid, but showing the next suit up (♡/♠) and forcing partner to bid that next step. The responding hand may then pass, or otherwise continue the description of their hand. At the expense of a natural 2♢ bid, this allows for a wide range of responding hands to describe themselves in a conversation while keeping the bidding low.
- *2/1 game-forcing*: after an opening bid of 1 of a suit, a bid of 2 of a lower-ranked suit being *forcing* to game (defined in Section 2.1.8).

In the examples above the common word *forcing* was used, which indicates a bid that by agreement cannot be passed by their partner even though the rules allow it.

Opening lead and carding conventions are also routinely employed by defenders to communicate information about the cards in the suit led or sometimes the rest of their hand, but just as with bidding conventions, these agreements must be fully disclosed before and during the play.

2.1.11.4 Constructive vs Competitive

The auction in Figure 2.11 was an example of a *constructive* or *unopposed auction* as one side did all the bidding while the other stayed silent and passed.

A competitive auction on the other hand might look something like the auction in Figure 2.12. Note that X is a common abbreviation for *double*, and XX for *redouble*.

North	*East*	*South*	*West*
1♠	2♡	2♠	4♡
4♠	5♡	X	P
P	P		

Figure 2.12: Competitive auction

Where all of 4♠, 5♡ and *X* fall in the *decision bid* category with players trying to decide which action will maximize their side's score. Either bid may have included the possibility of a *save* where one side bids to a possibly unmake-able contract with the expectation that the penalty conceded, even when doubled, will be less than the other side's score for a making game.

Since a competitive auction can benefit the overcalling side, why aren't all auctions competitive? For every bid taken there is always a risk of being doubled and conceding a penalty, and that risk goes up the higher the bidding level and the more bids that have already taken place. So each bidder must weigh the risk/reward of bidding over passing when entering an auction.

2.2 Computer Games

A quick background in how computers play games.

2.2.1 Simple Search

When solving a problem such as the classic one of legally placing Queens on an N by N chessboard, labelling a map, or finding a route between two places, there are no opponents.

These operations are easy to implement effectively on a computer, as they only require a method of enumerating the states and a verification function to test if the result is still valid. Formally,

- I: initial state
- s: the current state
- G(s): goal state
- A(s): actions available in s

- T(s, a): applies a to s and provides a new s'

Where G tests if you've reached the *goal state* of the search (e.g. a route between two places), A provides the actions from a given state and T applies an action to a state to provide a new state. With this, any search such as this can guarantee a solution if one exists through a standard algorithm.

```
function SEARCH(s, G, A, T)
    if G(s) then
        return solution
    end if
    for all a in A(s) do
        if SEARCH(T(s, a), G, A, T) is solution then
            return solution
        end if
    end for
    return failure
end function
```

However, the number of possible moves which may be searched through is exponential in size, such that the search can take a long time. A set of good heuristics can give a better guess as to which move to try, thereby speeding up the search, as once a single solution is found, the search is over.

2.2.2 Minimax in Games

Unfortunately, when playing a game such as Chess or Bridge, a simple search is not enough, as for each move that one player makes, their opponent will always try to make the best counter-move. Thus for every good move that one player makes, one must back up to each previous opponent play and try all of the other options to see if those would have been better from the opponent's perspective.

Let us examine the easiest example, using Tic-Tac-Toe as in Figure 2.13:

?		O
?	X	
X	O	

Figure 2.13: Tic-Tac-Toe with a winning position

At X's turn, a human can clearly see that playing in the top or middle left box will deliver victory, because O is unable to block in two places. But how can a computer discover these two plays are the only winning ones? Unlike the N-queens search above, at each turn when considering X's move, the computer must also try each counter play for O, assuming that the opponent will play optimally.

Consider if X tried to play in the bottom right corner, if O plays anywhere but top left, X wins. A blind depth-first search would have its solution, but not an *adversarial search*. A search for a solution must backtrack to the last play for O and try every possible play. In this case, top left would be found as blocking the win, so the search must continue.

More formally, borrowing from [Russell and Norvig, 2009]:

- S_0: the initial state
- s: the current state
- Player(s): current player
- Actions(s): legal moves
- Result(s, a): provides the new state from applying a to s
- Terminal(s): indicates if the game is over
- Payoff(s): indicates the payoff for a terminal game

Where we wish to compute

$$Minimax(s) = \begin{cases} Payoff(s) & \textbf{if } Terminal(s) \\ \max(a) \forall a \in Minimax(Result(s,a)) & \textbf{if } Player(s) = MAX \\ \min(a) \forall a \in Minimax(Result(s,a)) & \textbf{if } Player(s) = MIN \end{cases}$$

For games like Go or Chess the $Payoff(s)$ is always $[+1, 0, 1]$ to indicate win, tie or draw. Whereas for Bridge it is wider ranged in the number of tricks out of thirteen taken by each side.

This type of search can also be represented by a minimax tree, where X's moves are *or nodes*, as only a single valid solution is needed at those, but O's moves are *and nodes*, as X's previous winning move must work for all of O's possible moves. X is considered the *maximizer* as they are trying to win, while O is the *minimizer*, as from X's perspective they are trying to force X to lose.

Each game position or *node* of the tree is evaluated using Terminal to see if it is a draw or win for either player. If so it is a *leaf* node, if not it is an *internal* node where all possible moves of the current player must be considered (Actions). When choosing a best move for a player, the goal is for all leaves after that selection to lead to the best possible outcome. Put another way, you cannot evaluate whether a move to an internal node is a good one until the chain of follow up moves ends in a win/lose/draw outcome.

This search space is much larger than a simple blind search. In general, verifying a successful plan requires presenting a tree with branches for every possible move for O; the problem is thus not in NP (the class of problems whose solution verification takes polynomial time). Instead, the unfortunate consequence of this is that none of the improvements on state-space search by using local search or other optimization algorithms are useful.

2.2.3 Alpha-Beta Pruning

Minimax provides an important framework for computing optimal decisions in an adversarial game, if only there was enough computing power. For example, chess has an average branching factor of about 35, so in a typical 40-move limit per-player game that yields 35^{80} nodes to evaluate, an impossible problem [Russell and Norvig, 2009].

Alpha-Beta pruning is an enhancement to *minimax* search which prunes nodes early when it discovers that there is no reason to continue searching [Knuth and Moore, 1975]. By eliminating whole sections of a minimax tree, the total number of nodes can be reduced to a more reasonable number for evaluation.

Alpha corresponds to a global known value that max can already achieve, where Beta is the value that min can confirm. If at any point at a max-node (*or-node*) the *maximizer* can play to a better outcome than Beta, the search is abandoned, as min will make a better choice earlier in the search tree. The same is true when evaluating *and-nodes*, if the *minimizer* has a play which will lead to a lower option than Alpha, the search is abandoned, as max will make a better choice

higher in the tree.

2.2.4 Transposition Tables

For Tic-Tac-Toe, Alpha-Beta pruning plus the addition of transposition tables makes it possible to quickly compute optimal moves. A *transposition table* takes advantage of symmetries in positions to eliminate redundant searches. For example, perhaps in the previous position in Figure 2.13, X was in fact evaluating potential second moves on a board of X in the center and O directly below. An alternate set of moves might reach Figure 2.14.

O		
	X	
	O	X

Figure 2.14: Tic-Tac-Toe transposition

However, it would be a waste of computation to continue any further as one can clearly see this is just a longitudinal transposition of the previous position in Figure 2.13.

2.2.5 Evaluation Functions

Transposition Tables and Alpha-Beta pruning allow Chess and Go to evaluate to additional depths, but this is still not enough to do complete searching of the game-tree. The general solution, is to employ an evaluation function which predicts the value or *goodness* of a game state since reaching a terminal state is not feasible. This is then combined with a limited-depth minimax tree search to predict the best move for a particular state.

2.2.6 Imperfect Information

Tic-Tac-Toe, Chess and Go, are all examples of a *perfect information* game. At any point a player knows the exact state of the game, both their pieces and their opponent's. The complexity of these games comes from both the branching factor and the *depth* of the game-tree searches necessary to consider a move. If a player considers the next three moves each for both players (known as 6-ply), the game may still be wide-open. Thus the complexity and skill of the best programs is in evaluating each new board based on a move.

Bridge has a much shallower search space, as most auctions involve a player taking only a few bids, and the card play always terminates after each player plays all 13 cards. Where Bridge differs from the others is information. Throughout the hand there are hidden pieces of information that must be considered: all three hands during the bidding, and then the two other hands during the play (assuming you are not dummy). This type of *imperfect information* or *partially observable* game requires additional machinery to attempt to select an optimal move.

2.2.7 Monte Carlo

One method of move selection in an imperfect information game is simply to enumerate all possible realities, or slightly better all logically possible *belief states* by eliminating inferentially impossible distributions. Once this set is available, apply an adversarial search to each perfect information problem, then compute the *expected value* of each move based on the outcomes in each belief state. Given a set of deals s occurring with probability P, we wish to compute

$$argmax \sum_{s} P(s) Minimax(Result(s, a))$$

Of course for any game of interest the total number of such states is too large, so instead a random subset of these belief states or *paths* must be drawn and then solved. This *Monte Carlo simulation* attempts to balance choosing enough possibilities to evaluate to produce a decent solution versus evaluating too many, which may use more time than is reasonable.

Monte Carlo Tree Search ("MCTS") is the application of this Monte Carlo technique to a problem where tree search is then applied to the perfect information problem (covered extensively in Chapter 3).

2.2.8 Conformant Planning

Conformant planning is the procedure of generating a plan under uncertainty which will succeed regardless of the true state of the world [Hoffmann and Brafman, 2006]. The initial problem is transformed from one with uncertainty to a state space search, where the current state is actually a set of *belief-states* about the possible worlds. This is somewhat similar to Ginsberg's Lattice approach, which will be covered later in Section 2.3.2.2.

DTAC [Bethe, 2010a] is a type of conformant planning algorithm which finds a *claim* in the game of Bridge by reducing the adversarial and incomplete information game to a simple depth-first

search with quick solution. This can also be useful for an Alpha-Beta search by quickly providing and reinforcing values of *alpha* and *beta* to assist in tree pruning.

In related work, Russell and Wolfe explored a new method of solving imperfect information games in Kriegspiel via their *belief-state AND-OR search* in [Russell and Wolfe, 2005]. This technique showed solid success on a limited database of checkmate positions for 3-ply and 5-ply searches.

It is similar to DTAC only in that it explores a method of securing a guaranteed plan to an adversarial game. Unlike DTAC, it uses belief-state space to do an exact search up to resource limitations.

2.3 State of Bridge Play

Since the first fully automated Bridge program, Bridge Baron, was published [Throop, 1983], there have been large strides in AI Bridge. Bridge has the three distinct types of problems which we have covered: bidding, single-dummy play (the normal imperfect-information problem), and double-dummy play (assuming perfect information). Each of these have different rules, objectives, and inferences that can be drawn, and research has tended to focus on only one aspect at a time.

Early work on Bridge tended to follow other fields that looked for rule-based systems as a way to encode Bridge player heuristics and planning techniques to solve the single-dummy play problem [Throop, 1983; Smith et al., 1998; Frank et al., 1998]. It is also believed that a basic database of bids was used in similar rule-based bidding engines.

2.3.1 Partition Search

Double-dummy is the Bridge term referring to a situation with perfect information, and as such, an exact solution, can be solved for using the previously discussed technique of adversarial search with Alpha-Beta pruning. However, a double-dummy solver with only Alpha-Beta was still not fast enough. Considerable work was done by Ginsberg to build on existing adversarial search pruning techniques, and by adding in his *partition-search* algorithm which significantly reduced the number of moves to consider in a lossless way [Ginsberg, 1999; Ginsberg, 1996], solutions to the double-dummy problem could now be computed very efficiently. Armed now with an exact solver for the perfect information problem, a Monte Carlo simulation where each path was represented by an exact distribution of the cards could be used for move selection. At each play the card selected is the one with the maximal expected value over the layouts that have been analyzed via the solver.

2.3.2 Limitations of Monte Carlo

The Monte Carlo algorithm has a number of shortcomings which can limit the effectiveness of the technique. Firstly, the choice of move is based on a statistical estimate and may fail to account for outlier distributions. More subtly, there is the problem of *strategy fusion* [Frank and Basin, 1998] which is the following: the algorithm estimates the value of a move in a state of partial knowledge as its expected value averaging over all complete knowledge elaborations of that state. But there are circumstances where that estimate is not valid, because it assumes that all future moves can be chosen correctly, whereas they may in fact depend on information that will not be known.

2.3.2.1 Strategy-fusion

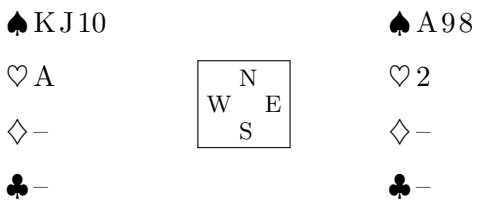


Figure 2.15: Example of strategy-fusion

In Figure 2.15, West is declarer, on lead to take the rest of the tricks. The opponents are known to hold 4 hearts, 1 diamond, and 3 spades including the queen. Due to earlier play (for example, South discarding on diamonds), North is known to hold the remaining diamond. If spades are 2-1, the queen will always fall, but if they are 3-0, which way should declarer guard for the *finesse*? Playing the King of spades first (and then the Jack) will only succeed when North holds all three spades, but playing to the Ace fails in that case. Conversely, playing to the Ace succeeds when South has all three spades, but fails when North turns out to have them.

A sound declarer would always make this contract without the need to guess, by making a "discovery play". By taking the Ace of hearts first, declarer learns what is needed for the next play.

- If North discards a diamond, then all four hearts are known to be in the South hand, and by inference the queen and all of the spades are in the North hand as shown in Figure 2.16. Declarer now playing with perfect information takes three spade tricks via the "marked" finesse through North;

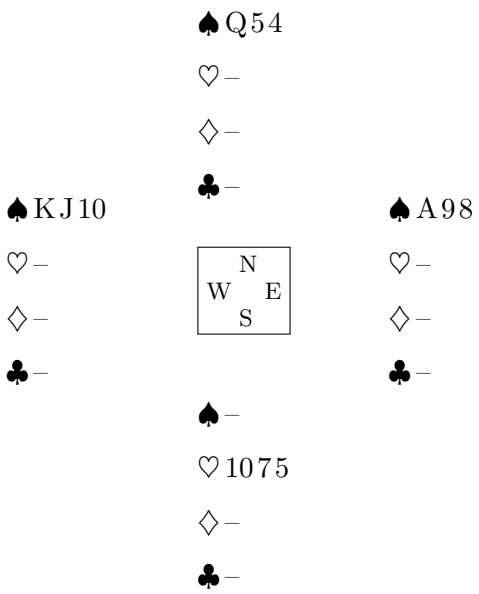


Figure 2.16: Strategy-fusion example: after diamond pitch

- Alternatively, if North follows to the first heart, then the North hand must contain $[0-2]$ spades. This is known by inference, as North is also known to hold at least one diamond, and started with only four cards. Declarer no longer needs to guard against North holding all 3 spades as the layout in Figure 2.16 is impossible. Accordingly they play the ♠T to dummy's Ace. If both players follow then the Queen will fall on the next round under the King. If instead North has revealed the position by discarding, declarer is able to take the "marked" finesse coming back as the position is known to be Figure 2.17.

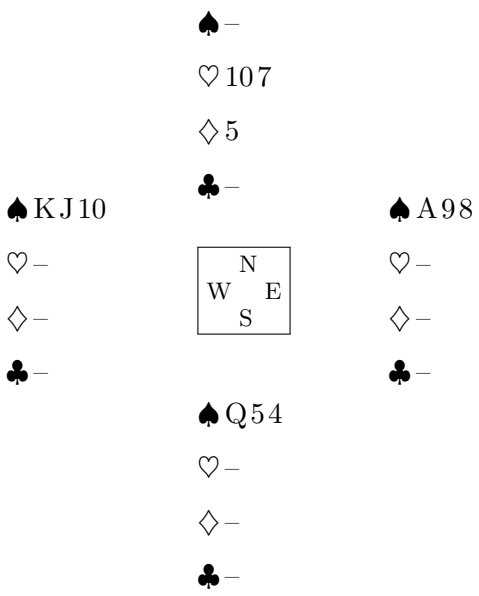


Figure 2.17: Strategy-fusion example: North follows

From the perspective of a Monte Carlo approach, either choice of $\spadesuit T$, $\heartsuit A$ always succeeds. In 3=0 splits where North has the queen (Figure 2.16) the finesse is taken immediately, as the double-dummy solver "knows" it will work. In other cases, the solver plays the Ace and also takes all the tricks. When taking an expectation over possible leads, the King of spades fails when the queen is with South, but the other two leads of the Ace of hearts and the Ten of spades work equally (taking the Ten and Jack as equals). So one of the two "winning" options is chosen at random, and 50% of the time, Monte Carlo has chosen the inferior line of $\spadesuit T$, which may fail in 11% of the actual distributions.

2.3.2.2 Lattice Methods, and Squeaky Wheel Optimization

In an attempt to overcome *strategy-fusion*, Ginsberg employed a lattice method for solving *binary decision diagrams* [Nielsen, 2001]. This technique worked and was found to be useful for small end-positions, but as the number of cards remaining increased, the time increased as well (to over 1 minute for 8-card positions). However, Ginsberg modified this technique using *achievable sets* and a modification to the selection order called *squeaky wheel optimization* [Joslin and Clements, 1999], which created an approximation of the problem that was solvable in reasonable time. Profiling this technique against GIB's Monte Carlo engine showed a small but significant improvement in the play. This is the method that is currently used by GIB, but it is still an approximation.

2.3.3 General Bidding

As previously suggested, one can informally divide bids into several categories including opening bids, intermediate bids used for information passing and gathering, and final or decision bids.

Analysis of existing programs show that they perform fine at opening bids, as they simply involve matching a set of criteria with a database of rules to apply.

Existing AI also displays strength at making decision bids, particularly in competitive auctions, as enough information has been passed that a Monte Carlo simulation can be very effective in selecting a winning bid.

It is the set of in-between bids, the planning and conversational bids where problems still exist. This topic will be expanded upon later.

2.3.4 Bidding Systems and Meanings

In Chess or Go when a move is made, the player need not explain why they chose the move nor what it means. In Bridge card play there may be agreements about what card is led from length or touching honors as well as certain other types of signals, but the declarations on these are fairly straightforward.

Bidding however, requires a much more complete level of disclosure. Other than in an unusual auction, most bids are expected to be explainable through a standard set of terminology. Whether an opening 1♡ bid shows a Standard American style 5 or more hearts and 11-21 HCP or something completely different, the opponents must receive all agreed upon information even if one or both opponents plans to pass. Thus it is important that for any machine learning algorithms to be usable in real competition against computer or human, semantic knowledge of the meaning of each bid must be made available.

2.3.5 Recent Work

Recent research on Bridge has primarily looked at improving bidding through Machine Learning ("ML"). Thanks to AlphaGo [Silver et al., 2016], an interest has been renewed in this area with several recent papers that explore similar methods of deep reinforcement learning applied to Bridge.

Yeh et al. (2018) investigated using a deep Q-network to learn constructive bidding. They tried two tracks: (1) strongly hinting opening-bids that conform to a human system; and (ii) allowing the model to drive all bids. They explored convergence speeds for several human-designed systems

including Standard American Yellow Card ("SAYC"), Precision and Natural-5542. The authors note the importance of balancing exploration vs exploitation for achieving best performance, and report very encouraging results when compared to Wbridge5. Importantly they also address the need to produce semantic information as bid meanings must be explainable to opponents. Training data consisted of 100,000 randomly generated deals plus another 40,000 for validation and testing. Unfortunately the code, developed in MATLAB, is unpublished so we were unable to exactly verify their results.

Rong, Qin and An (2019) followed on that work, to investigate the more challenging problem of competitive bidding. They worked to train two networks: one which took as input the bidder's cards and the auction history to output probabilities of each other player's cards; the second augmented the bidder's cards and auction history with the estimate of partner's hand produced by the first network. This second input state was modeled as a 318-bit map (9 per bid) plus an additional 52 bits each for the bidder and partner's hand and 2 more for the vulnerability. Training data of one million deals was obtained from the Vugraph Project [Sarantakos, 2020] and augmented with double-dummy analysis. Strong results for the trained model were reported; however, there is no published code available for verification.

Gong, Jiang and Tian (2019) also attempted to learn all aspects of bidding; however, their 175-bit representation of the bidding history does not seem to differentiate between which partner has made a double or redouble. For example, the auctions in Figure 2.18 would be encoded identically to each other, yet they are two very different auctions which must be differentiated. Training data consisted of 2.5 million randomly generated deals augmented with double-dummy analysis. As with the other recent work, no code was made available for inspection.

West	*North*	*East*	*South*
1♢	P	P	X
P	P	1H	

(a) Long auction

West	*North*	*East*	*South*
1♢	X	1H	

(b) Short auction

Figure 2.18: Auctions not the same

Legras et al (2018) went in a different direction, using Inductive Logic Programming to model a simplified problem of determining whether to open the bidding or not when given a limited hand. They were able to engage four expert bridge players to label a few thousand hands which qualified

for the experiment, split into several different data sets. The authors report some success for this sub-problem and we hope to see continued success as the problem size expands.

There were also a few papers attempting to revisit issues with the play of the hand including [Cazenave and Ventos, 2019] which explored a new search algorithm in place of MCTS and [Ventos et al., 2017] which explored how better random seed selection for WBridge5 could improve results in the play. The latter is a concept that will be explored in Chapter 3 although with a different proposed solution.

Each of these works showed good promise towards the goal of building an AI to the equivalent of an expert Bridge player, and we hope to build on these in the coming chapters.

A survey of research prior to 2010 is covered in [Bethe, 2010b].

2.3.6 Current Programs

As with Chess, Bridge saw many attempts at commercial software, with perhaps Bridge Baron [Throop, 1983] having the most early success. GiB is the commercially available program that came out of Ginsburg's previously mentioned work; it is available as an opponent on BridgeBase.com, but it no longer actively competes. Over the last ten years, several programs including Jack, WBridge5 and SharkBridge have been crowned champions of the World Computer Bridge Championships ("WCBC"), these programs are rumored to be written using a customization of the Monte Carlo and double-dummy techniques first proposed by Ginsberg [Russell and Norvig, 2009]. However, exact details of these programs are proprietary, and were therefore not available for review and analysis.

2.4 Advances with Machine Learning

2.4.1 AlphaGo

AlphaGo [Silver et al., 2016] demonstrated that for a discrete state-space search, deep Q-learning was an effective way of learning an evaluation function for a possible Go-board after a series of min-max moves. This prompted researchers for several games, including Bridge, to investigate if the same techniques could be used (as highlighted in Section 2.3.5).

AlphaGo was a large undertaking employing close to 50 engineers and researchers over several years. The great breakthrough of AlphaGo was the use of deep reinforcement learning as a method of computing two evaluation functions. The *value network* is a neural network used to predict

the expected winner from a board state. The *policy network* then used that network to train a model for selecting the next move in concert with a limited-depth tree search. These networks were bootstrapped with a large corpus of previously played amateur games where a series of positions and the eventual winner were already known. After that, self-play was used to continually train and refine these models. By playing millions of games, these complex neural networks could extract learning features from the boards that were more likely to lead to a particular outcome.

The result of this investment was a 4-1 victory over one of the top-ranked players in the world, utilizing as many as 1900 CPUs and 280 GPUs during the match [Silver et al., 2016].

AlphaZero was the next iteration of these techniques skipping over bootstrapping and using an improved algorithm plus new Tensorflow processing unit hardware. Responding to criticism of the computing footprint used in AlphaGo, the new iteration required a much smaller footprint to play with (4 TPUs and a single 44-core CPU). The AlphaZero model was extended to other games, convincingly defeating the top programs in both Chess and Shogi in 2017 [Silver et al., 2018].

2.4.2 Poker

Another game now bested by AI is no-limit Texas Hold'em. Libratus [Brown and Sandholm, 2018] thoroughly defeated four professional players over a 20 day tournament of heads-up play. A primary challenge of no-limit as opposed to limited betting, is a state space that is much too big to solve. Building on breakthroughs in limited Hold'em using counterfactual regret minimization ("CFR"), Libratus combined a method to reduce the early stages of play to a solvable sub-game with a Monte Carlo CFR to overcome this.

Not long after, the solution was extended to cover multiplayer no-limit Texas Hold'em as Pluribus [Brown and Sandholm, 2019] defeated a group of human professionals in a similar type of competition.

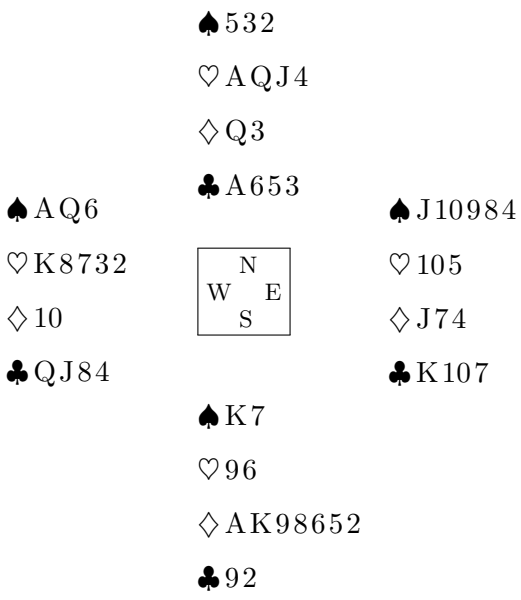
Chapter 3

Improving Move Selection using Monte Carlo

While there has been significant recent research in the unsolved problem of Bridge bidding, there has been little on the play phase of the game. In this Chapter we re-investigate how computers play the cards using Monte Carlo Tree Search ("MCTS"), and the blind spots that creates. Then we propose two modifications to move selection which experimentally showed a significant improvement of 0.05 tricks per deal played. More specifically, in several cases where random tie-breaking leads to incorrect play, our system can often find the correct move.

3.1 Expert Ability

Thanks to the breakthroughs which allowed for a fast perfect-information *double-dummy* solver [Ginsberg, 1996], computer Bridge bots have repeatedly shown impressive expert-level play; consequently, recent research has almost exclusively focused on the bidding phase. Several recent examples of top-level play can be found in World Computer Bridge Championships ("WCBC") articles such as [Levy, 2017].



West	*North*	*East*	*South*
1♡	P	1♠	2♢
2♠	X	P	3♡
P	4NT	P	5♢
P	6♢	P	P
P			

Figure 3.1: Expert robot play

In the 20th WCBC round-robin, Figure 3.1, Wbridge5 South reached 6♢ after West opened 1♡, East responded 1♠, and West raised to 2♠. By convention, the lead of the ♣Q shows the Jack but denies the King, so applying inferences from the bidding, West almost certainly held the ♡K and the ♠A. Declarer thus won the Ace, played ♢Q to the Ace, finessed the heart Queen and again found the winning play of a diamond to the 9, thereafter running diamonds to reach the five-card end-position in Figure 3.2.

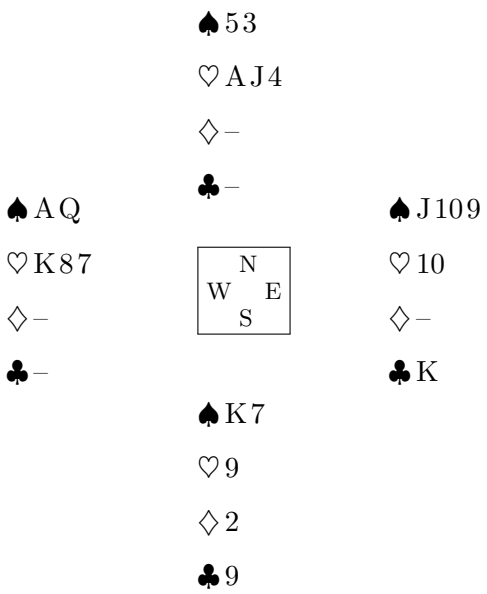


Figure 3.2: Expert robot squeeze

The last diamond squeezed West, who could not hold both two spades and three hearts. Impressive play that any expert would be proud of, MCTS provided the highest probability sequence of plays that also turned out to be correct. This and many other such hands can be found in International Computer Games Association journals and other write-ups of these tournaments, and without a deeper analysis, one could be convinced that robots are expert players needing no further improvements.

3.2 Monte Carlo Tree Search

To achieve these results, Bridge AI employs a variation of MCTS. The Monte Carlo portion involves converting from hidden to perfect information by dealing out the unknown cards in a way that is consistent with inferences available from the bidding and the play. The Tree Search is a double-dummy solver that uses the standard minimax, alpha-beta, and transposition table techniques, plus partition search to reduce branching factor. This search can provide the number of tricks that will be taken for each card under consideration. If unlimited time and/or resources were available, all possible distributions of the unknown cards would be examined. However, that is infeasible, and so the theory of MCTS is that by dealing enough paths one can produce a set which is representative of the complete space. In practice that must also balance with the time constraints allowed for the play of each card, so a balancing algorithm will draw and solve paths until either a set number of

paths or a time-limit is reached, then produce the best result available so far.

We previously covered:

$$argmax \sum_{s} P(s) Minimax(Result(s, a))$$

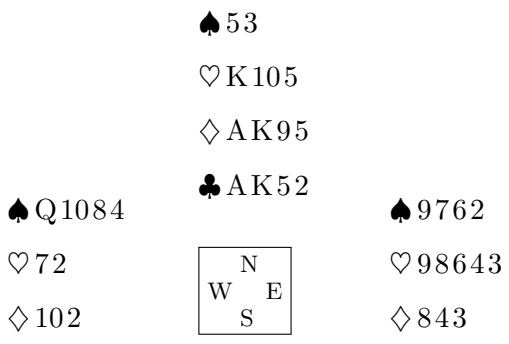
as the standard equation to compute the optimal move in MCTS where *Result* is the number of tricks taken, but Bridge introduces at least two problems to that general solution:

- How are ties broken among equal probability choices?
- What is the goal for which minimax is optimizing?

As we will see these are still somewhat open problems in need of solution.

3.3 Finding Flaws

3.3.1 Subtle Error



South	*West*	*North*	*East*
2NT	P	7NT	P
P	P		

Figure 3.3: Subtle defensive error

Figure 3.3 shows 7NT on the lead of the ♣J. Expert play might proceed with three rounds of clubs and the ♠A looking for an easy 13^{th} trick, failing that three rounds of hearts and four rounds of

diamonds ending in the dummy. Figure 3.4 shows the position when the fourth round of diamonds is led, the discards having revealed that East started 4=5=3=1 (and thus West 4=2=2=5).

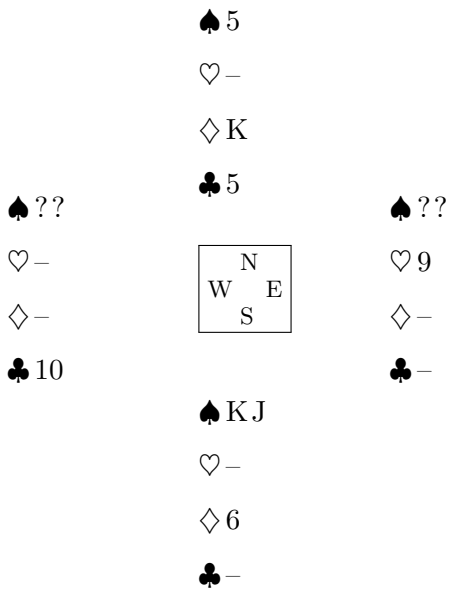


Figure 3.4: End-position

Expert defense in Figure 3.4 is for East to pitch a heart and West to pitch a low spade, into the position in Figure 3.5. When the ♣5 was not a winner a spade would be led by North, East would play low, and it is now a 50/50 guess as to which hand has the ♠Q, as declarer does not know which opponent holds the ♠Q and which holds the ♠9.

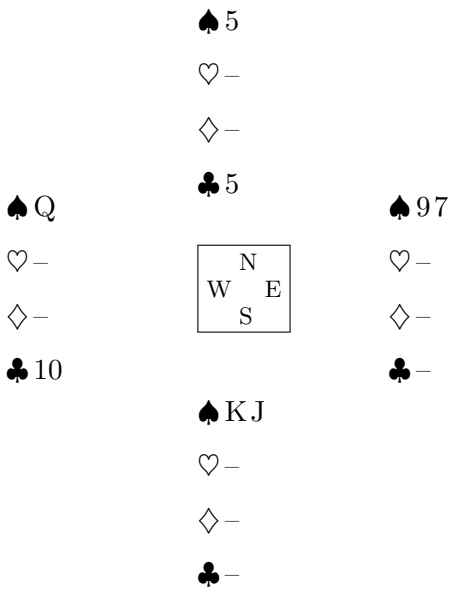


Figure 3.5: Good defense

Except... when defending this hand, the *robots*[1] are prone to several mistakes. When the $\diamondsuit K$ was led in Figure 3.4, West was squeezed and technically had no winning play. Thus for some robots, it was equally likely that the $\spadesuit T$, $\clubsuit T$ or $\spadesuit Q$ were pitched. As discussed above, the $\spadesuit T$ leads to a 50/50 guess by declarer; however, either of the latter two plays gives away information allowing declarer to take the remaining tricks without guess.

Similarly and more subtly, on the last diamond trick, East had the uninteresting choices of ♠97 or ♡9 and also picked randomly when in fact a spade was fatal. After a spade pitch from West and a low spade pitch from East, as in Figure 3.6, declarer has no more guessing to do as West is known to have exactly one spade and one heart while East has one spade and one club; so the Queen will fall under the King, regardless of which player holds it.

[1] tested and verified with all of Wbridge5, Jackbridge, Bridge Baron, GiB

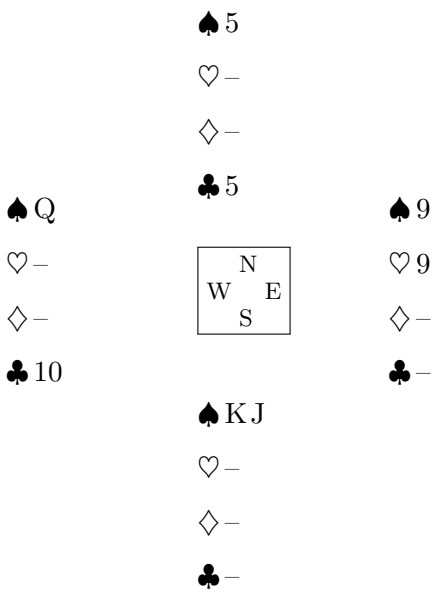


Figure 3.6: Incorrect spade pitch

What is going on here? The same MCTS that led to outstanding play in the previous section caused the downfall of the robots on this hand. Thanks to the bidding and the play, while declarer has some unknowns, East and West are playing with perfect information. Thus East and West know that all plays they make are equally likely to fail and therefore do not matter. Since in these cases, the standard MCTS tie-breaker for equal-probability moves of random selection fails to always produce optimal play.

In fact, the robots sometimes play this correctly and sometimes incorrectly, depending on the choices made randomly. This sensitivity to random number choices is similar to that of [Ventos et al., 2017]; however, we will propose quite a different solution.

3.3.2 Beginner Mistake

As discussed often on community forums[2], perhaps the most frustrating thing a bot can do is to make a mistake that even a beginner would not.

[2]e.g. https://bridgewinners.com/article/view/playing-with-and-against-the-gibs

	♠A10965 ♡AQJ ♢QJ ♣K73	
♠73 ♡972 ♢10932 ♣QJ104	N W E S	♠K ♡8643 ♢87654 ♣986
	♠QJ842 ♡K105 ♢AK ♣A52	

North	*East*	*South*	*West*
1♠	P	$2NT^1$	P
$3NT^2$	P	$4NT^3$	P
$5♡^4$	P	6♠	P
P	P		

[1] Game-forcing, 4+ spades

[2] 17+ HCP, balanced

[3] Asking: how many do you have of [♠A, ♠K, ♡A, ♢A, ♣A]

[4] Answer: exactly 2

Figure 3.7: Bad lead

Thanks to the bidding, East can predict that the NS hands look similar to the ones shown here. East can infer that the opponents hold all four aces from the fact that NS have bid a slam in spades without the King. The aces will split two and two between North and South as conveyed by the 5♡ response in the auction. Therefore, MCTS may predict that all choices of opening lead are equal and select the ♠K. This is a documented flaw in MCTS caused by attempting to compute a strategy by averaging the results of a series of perfect information solutions. In each double-dummy outcome the opponent plays with perfect information, which here means playing the ♠A and catching the ♠K.

In actual play, leading the ♠K is the only choice that would allow declarer to make the slam. On any other lead, declarer, whether human or robot, would likely win some other suit lead in

dummy and proceed to play the ♠Q. When West plays the ♠3, MCTS will show the same thing that a human declarer would know from the probabilities: that West is more likely to hold King or K7 than East is to hold exactly the King (and it doesn't matter what you do with East holding K7). The highest percentage play of finessing will lose, resulting in a failed slam.

A first thought might be to program a heuristic to prevent leading a singleton King or King of trumps, but counter-examples can be found where that would be the winning lead.

3.3.3 Equal is not Equivalent

The core issue on these two hands is the reality that a seemingly equal move under MCTS may not be equivalent in terms of the information revealed to the opponents. This is particularly acute in cases of asymmetric information as in these two deals, where auction and/or play created an imbalance between the two sides. What this suggests is that doing an expected value calculation by simply computing the average expected tricks of a choice is inferior under certain conditions.

3.3.4 Player Construction

This raises a pair of questions: what is the frequency of these random-of-equal choices and what effect do they have on the number of tricks taken by each side? Note that equivalent cards are already excluded from this analysis; for example, if a player has to choose from the ♠432 they are equivalent and this is not considered part of this experiment.

Answering these questions proved somewhat difficult, as a full MCTS player had to be constructed and instrumented to record the outcomes of using different random seeds for decision making. This player needed to:

- Have a method for solving double-dummy problems;
- Have a method for doing constrained dealing;
- Have a database for extracting meaning from auctions; and
- Be able to play Bridge against itself or other robots like GiB/Wbridge5.

A double dummy solver was easy as an open source library DDS was available in C++ [Haglund and Hein, 2020] and easily adaptable. Similarly, the Dealer program [Uijterwaal, 2020] offered the ability to propose complex constraints for dealing a set of hands, including HCP, specific cards, or lengths in a suit and relative lengths between suits. However, the rest of our MCTS player needed

construction from scratch, the most difficult part of which required an engine for interpreting auction and play meanings so as to make constrained dealing inferences.

The overall algorithm to play a card was:

Algorithm 1 Baseline MCTS

```
function ChooseCard(hand, auction, play, samples)
    constraints ← InterpretAuction(hand, auction, play)
    for all deal in Dealer(constraints, samples) do
        for all card, tricks in DDS(deal) do
            append tricks to results[card]
        end for
    end for
    avg[card] ← Average(results[card])
    return card with max(avg[card])                ▷ Ties broken randomly
end function
```

In our default setting, the DEALER function generated 100 random deals consistent with the proposed constraints. We used the DDS and Dealer software discussed above, but InterpretAuction was original code to produce constraints from the auction history. It consisted of two parts:

- A database which stored meanings for bids such as a 1♠ or 2♠ opening or a 2♠ response to a 1♠ opening; and
- A matching algorithm which determined what a bid was doing in sequence such as "opening" or "responding to 1♠".

North	*East*	*South*	*West*
1♡	P	2♡	P
P	P		

Figure 3.8: Lead constraints

For example, on lead as East holding ♠ K Q 6 4 ♡ 10 ♢ 10 7 6 ♣ A 8 6 3 2 after the auction in Figure 3.8, the DEALER function requires producing 100 deals consistent with the bidding. Our InterpretAuction algorithm would match these bids and might produce the following input for the dealer program:

```
predeal east SKQ64, HT, DT76, CA8632
condition hearts(north) >=5 && (hcp(north) > 10 && hcp(north) < 17)
  && hearts(south) >= 3 && (hcp(south) >= 5 && hcp(south) <= 9)
produce 100 action printoneline
```

Once all of these components were available, some engineering in Java and C++ was required to bring it all together in a program instrumented to be able to count every interesting occurrence and explain each decision being made.

Our complete MCTS player was then tested in a series of 64 board matches against GiB and found to be of comparable strength.

3.3.5 Frequency

The effort of constructing the player was worth it, as the results validated the work.

Side	Random draws per deal
declaring side	4.1
defense	3.8

Table 3.1: Random choices per deal over 10K

Shown in Table 3.1, a random sample of 10,000 deals demonstrated that there were multiple situations for every deal requiring a random draw to choose between cards deemed equal (tolerance of 0.01 tricks) by MCTS. Additionally, there was no clear correlation by side or to the trick in which the choice might occur (Figure 3.9). Our hypothesis for this is that late in the hand one side will take most of the tricks and with flexibility in the order chosen, while the other side may often be out of a suit played with equal choice of discards under MCTS that will not impact the play[3].

[3] As mentioned in Section 3.3.4, adjacent cards ♠432 or equivalent cards such as ♠753 if the 64 have already been played are already eliminated from this analysis.

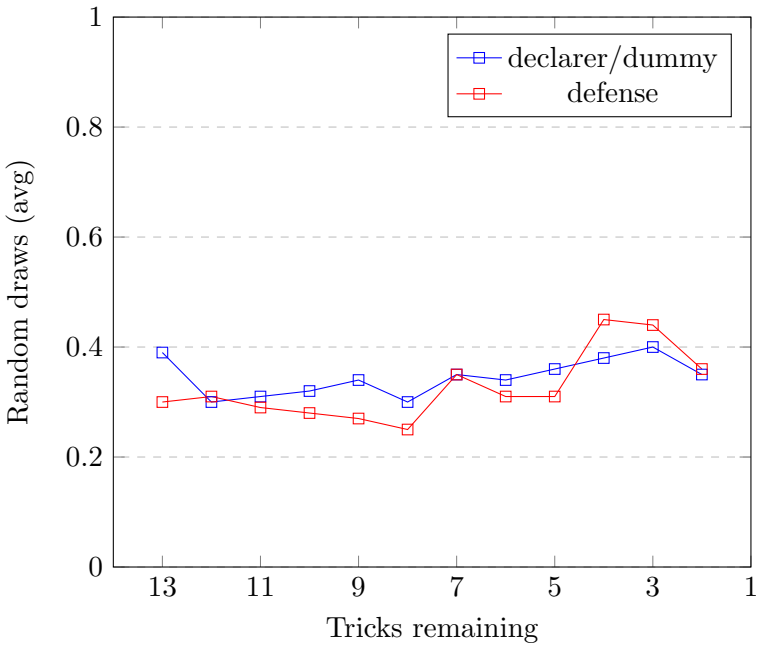


Figure 3.9: Random choices by side over 10K deals

Next, we measured how often a choice between cards that in, algorithm 1, would be judged to be equal, and thus made randomly, would in fact alter the number of tricks taken. In 10,000 randomized trials we computed the difference between upper and lower bounds of tricks taken on the same deal and then averaged across all of the deals. The result of this experiment was a net effect of ± 0.15 tricks per deal. Since a single trick may make or break a contract, this is a very noticeable deficiency in the base algorithm.

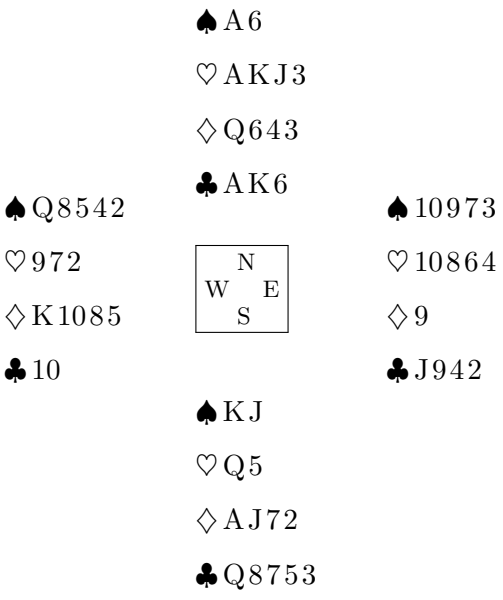
3.3.6 Desired Result

In many other games like Chess, the outcome of a game is win, lose or draw, but in Bridge the result and therefore the goal may vary based on the form of scoring (see Appendix).

In Figure 3.10 after a spade lead to the King, North plays the ♣AK and discovers the bad club break; however, a 100% play exists to guarantee the contract. A diamond to the Ace and a low diamond will either:

- Find diamonds 3-2 or a singleton King, taking 3♢ 3♣ 4♡ and 2♠;
- Find East with 4 or more diamonds to the King and vulnerable to a club-diamond squeeze;
- Force West to take the diamond King leading to 3♢ 3♣ 4♡ and 2♠; or

- Let West holding 4 or more diamonds duck the diamond King, at which point after the ♢Q wins, a club can be lost without repercussion leading to 2♢ 4♣ 4♡ and 2♠.



North	*East*	*South*	*West*
2NT	P	4NT	P
6NT	P	P	P

Figure 3.10: Safety play

This is an example of a safety play, as discussed in Section 2.1.10, where one sacrifices the chance at overtricks to increase the probability of making the contract (or here guarantee). However, this is not the move selected by the robots based on the probabilities in Table 3.2.

Cards	Avg tricks	Standard dev.
♡Q	9.21	± 0.4
low♣	9.21	± 0.4
low♢	9.0	± 0.0

Table 3.2: 500-path MCTS choices at trick 4

Here base MCTS selects the move with the highest average number of tricks from each of the sampled paths, but that is not the right choice if the form of scoring is IMPs (see Appendix). There is a premium in IMP scoring for making your contract, thus the marginal utility of each trick is not the same and needs to be accounted for when selecting moves.

3.4 Improvements

The data clearly indicating a real problem for random selection, a number of solutions were considered as alternate methods of tie-breaking. Additionally, we propose modifying the goal of declarer making (or the defense beating) the current contract to take into account the variance of move choices.

3.4.1 Sub-simulation

One possible solution might be to run an inner MCTS. For each deal of the outer Monte Carlo simulation, run another MCTS from the opponents' perspective, using their cards and auction and play knowledge to simulate their belief-states and then compute results for all of the plays being considered.

This approach was examined in general by [Goodman, 2019] for Hanabi, a Japanese cooperative card game, and found that the problems of strategy fusion and non-locality (discussed in 2.3.2.1) still persist. These issues arise as information known only by the current player *leaks* into the opponent's predicted strategy.

We also find two flaws for the case of Bridge and our example problems. The first problem is computational: double-dummy searches taking approximately 50ms per path for the first trick and 30ms for the second trick (Figure 3.11) are prohibitive for any reasonably sized sub-simulation. At 100 paths in the base algorithm, this is 5s for the initial round of play and acceptable. In fact more paths could be considered, but another factor of 100 or more for a sub-simulation would be unacceptable. Second and perhaps more important here, is that it doesn't solve the first problem presented in Section 3.3.1. In that case in the sub-simulation, declarer would always correctly guess the location of the $\spadesuit Q$ (in Figures 3.4-3.5) so it provides no help in determining which card to play.

3.4.2 Learning

Modern AI demands that Machine Learning be considered for any new problem, but here it runs into issues. Unlike the limited success of the full double-dummy ML solver in [Mossakowski and Mandziuk, 2009], here we wish only to pick the best among equals given our knowledge of the current hand and the trick in progress.

Unfortunately, we lack an algorithm that can determine whether one choice is better than another. For example, in the hand in Figure 3.4, no choice of play can prevent the declarer from

making the correct guess. Due to this, we are unable to provide a *goodness* score for reinforcement purposes of an unsupervised model.

To use supervised learning would require a labeled training set, but no such corpus exists. The creation of such a data set would require significant work in identifying situations in hands where a choice among equals exists, and then a human would have to analyze and provide a label. The time required to create one large enough is considered prohibitive, as analysis might take a minute or two per deal and we are not sure how large a corpus would be needed for training and testing of an accurate model.

3.4.3 Information Score

Unable to use sub-simulation or learning to solve this problem, we instead adapt a modified version of *combined defender* search first introduced in [Bethe, 2010a] for claiming, and use it to estimate the information available to the opponent after the completion of the trick being played. The combined defender search computes the number of tricks known to be available given all information at hand but otherwise assuming the worst possible outcome for any possible distribution. This provides the lower bound on the number of tricks declarer can take at any given point. By defining the combined defensive assets as:

$$CD = [\spadesuit = (H, L, M); \heartsuit = (H, L, M); \diamondsuit = (H, L, M); \clubsuit = (H, L, M)]$$

Where:

- H is the highest card the opponents have in that suit.
- L is the max length that either opponent could have in a suit.
- M is the min length that either opponent has in a suit.

This construct allows a linear-time accept or reject of a proposed line of play to take a certain number of tricks; thus we can now perform a state-space search over what is now a complete information problem reducing the total time required (as opposed to an inner-MCTS).

The key to success here is that we do not leak hidden information in computing an estimate of the opposition's choices. This has similarities to how [Cowling et al., 2012] worked to eliminate information leaks in a set of hidden information card games, except here the concepts are applied to Bridge.

For our modified version, for each play considered by the current player, a one-trick adversarial search is performed where one side is maximizer and the other minimizer, except that *Result* is the combined defender search tricks rather double-dummy tricks. We then keep those numbers as a secondary score associated with the choice of a particular card.

So where previously we recorded only the number of tricks taken on a path, we now also calculate and keep the opponents' lower bound on tricks as the information score in:

$Result(s, a) = (tricks, information\_score)$

Now when determining which card to play, we use the *information score* tricks provided by the combined defender search to break ties within a tolerance of 0.01 tricks to the maximum.

On the surface, this algorithm is a clear improvement as it leads to correct play in both of the problem examples: against 7NT from Section 3.3.1, pitching either a spade by East or $\spadesuit Q/\clubsuit T$ by West increases by one the combined defender claimable tricks by the opponent and so is rejected by our algorithm in favor of best play; similarly against 6$\spadesuit$ in Section 3.3.2, leading the $\spadesuit K$ is rejected by our algorithm in the tiebreaker as it gives away a trick in *combined defender* search.

3.4.4 Goal Modification

Based on observing the commercially available bots playing the hand in Figure 3.10 and not finding the safety play, we also propose a modification to the goal that $argmax$ should be selecting from[4] for optimal play. Instead of selecting $\max(tricks)$, when more than one action satisfies the goal number of tricks we compute the sample standard deviation (σ) and instead choose $\max(tricks - \sigma)$, breaking ties as above using the information score. The choice of σ as opposed to 1.5 or 2 times σ is arbitrary. Future experiments might examine the effect of choosing other confidence boundaries for these computations.

Using this metric for move selection, our enhanced MCTS now plays correctly and makes the contract via safety play in the hand from Figure 3.10.

3.5 Experimental Results

In section 3.4.3 we showed that the information score algorithm is an improvement over base MCTS by selecting the optimal card for those specific hands. We now performed additional experiments to understand what value this algorithm would have over a larger set of deals, and what if any

[4]Note that the software builds used in WCBC play may already include this optimization

performance costs exist.

3.5.1 Computational Performance

Now that we have a prospective algorithm, we investigated both the performance cost of this additional calculation, and the effectiveness in terms of achieved results. Figure 3.11 shows that while a single combined defender analysis is always faster than double-dummy, for the first few tricks there is a noticeable cost for the combined defender. Unfortunately, while only one double-dummy search is required for a single Monte Carlo path, our new algorithm requires a combined defender analysis for the remaining layout after every possible trick.

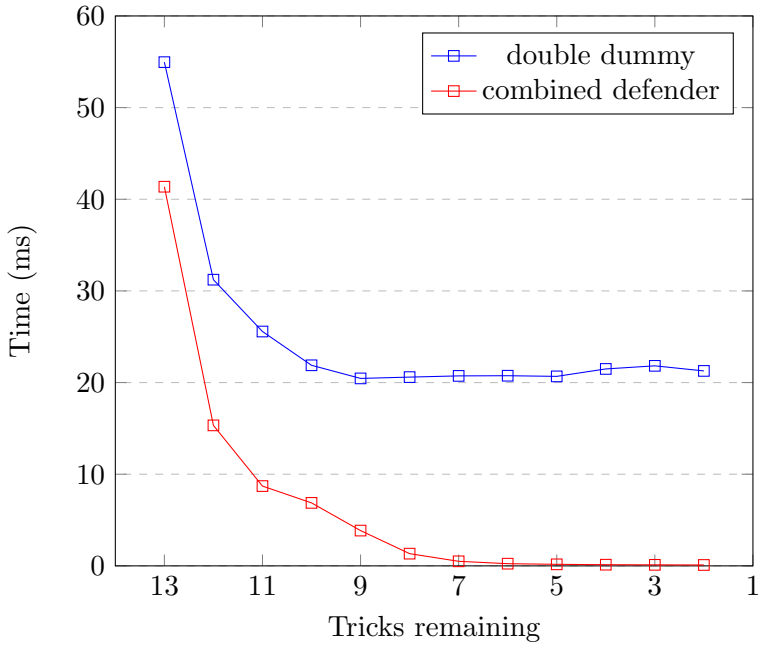


Figure 3.11: Double-dummy and Combined Defender compute times

For example, in Figure 3.7 computing the information score when considering each of the possible first tricks would run approximately 12 different paths which at 15ms per computation (12-tricks remaining) would cost 0.18 seconds per path, or an additional 18s for 100 paths. In Figure 3.12 we show the significant additional cost to the MCTS algorithm when enhanced with combined-defender search. We observed just under 2 minutes on average for a single player to play the cards to a deal, although this can vary depending on early branching factors in the tree search.

For perspective, the highest level Bridge tournaments allow approximately 8-9 minutes total per deal split among all players. Other types of play usually offer less time for board completion,

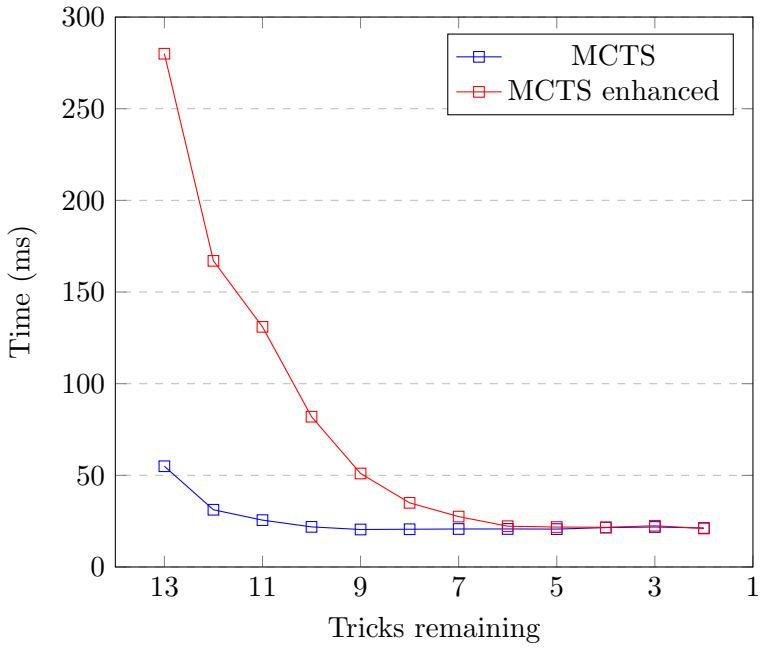


Figure 3.12: MCTS per-path baseline and enhanced timings

and when a human is playing with or against a robot they expect a fairly responsive system. When allowing all players to play and including the bidding, that puts enhanced MCTS right on the time limits for the highest level play, but likely too slow for other types of tournaments.

However, it is worth noting that the code used for the combined defender is ten years old and was written with a goal of sub-second performance to be performed once per trick. With the new purpose of running several times per path, additional performance is required and likely achievable through additional caching and parallelism.

3.5.2 Validation of Double-dummy Comparisons

To gain some comparative insight, we compute the expected tricks per deal that human defenders take versus the double-dummy expectation.

In Table 3.3 we present an analysis of approximately 70,000 deals available from the Vugraph Project [Sarantakos, 2020] to see how human defenders fared against the expected double-dummy result of the contract they were defending. For comparison in the table we include numbers from [Pavlicek, 2014] which cover a different set of similar strength competitions.

This demonstrates that actual results are close enough to double-dummy to provide a good baseline for comparison. Also of note is reinforcement of the concept of *declarer's edge* in play,

Source	Differential
Vugraph	-0.09
Pavlicek	-0.07

Table 3.3: Human defense expectation vs. double-dummy

where declarer tends to outperform the defense by a small margin (although this inverts at the slam level). A more extensive analysis of actual versus double dummy results is available in [Bird and Anthias, 2011].

3.5.3 Information Score Results

Unfortunately, we do not have a corpus of human vs computer deals, so the best we can do is look at robot vs. robot double-dummy comparisons before and after our proposed modifications. Table 3.4 shows a similar edge for declarer, and then a noticeable improvement was observed with the combined defender modification over the same set of deals. However, it should not be compared to Table 3.3 in absolute terms, as we do not know the relative strength of the robot declarer versus humans.

Source	Differential
MCTS base	-0.14
MCTS enhanced	-0.09

Table 3.4: Effect of the enhanced MCTS

We already showed that random move selection occurs several times per hand. These results suggest a significant improvement over the baseline of random move selection through the method of breaking near-ties (tolerance of 0.01) via the combined defender's information score.

3.5.4 Using Variance Reduction

While improvements over random move selection could be quantified in terms of a tricks per hand metric, our next investigation was to modify our goal from taking the most tricks on a deal, to one of making or beating the contract. To this end we needed to count how often such a decision was made, and then change our success metric in terms of a change of goal met or failed.

We looked for three types of occurrences to consider move modification, using terms G for the goal, H for the highest expected tricks and S for the safest play (H is always greater than S) and σ with subscripts for the standard deviation:

- A safety play approximation with an expected value of at least goal and a standard deviation of zero: $S >= G$ and $\sigma_S \approx 0$;
- A safest play, where $H > S$ yet $S - \sigma_S >= G > H - \sigma_H$; and
- A safer play where $H > S$ but $S - \sigma_S > H - \sigma_H >= G$.

Our findings in Table 3.5 show that these plays do happen with some frequency, and that this goal modification does lead to improved results at IMPs. It is important to note that these numbers are dependent on the simulation parameters; that is another MCTS engine might yield different numbers based on constraints extracted from bidding and play. Since our calculation uses just a sampling of the possible distributions, it is possible that there are outlier cases not covered in our sample that would change the computations.

Type	Per deal occurrence
Safety	0.10
Safest	0.04
Safer	0.25

Table 3.5: Effect of using standard deviation

When the example hand in Figure 3.10 is played by our baseline MCTS and GiB, the contract both succeeds and fails depending on the random seeds in use by both declarer and defense. That is, similar to Figure 3.7, there are subtle defensive mistakes available to unlucky random draws such as West pitching a low diamond on the second round of clubs. Our enhanced MCTS using safer move selection always makes this hand as declarer.

3.6 Conclusions

We explored an improvement to random move selection among equals with some success, and found a small number of hands where a modified goal was profitable. There was a computational cost to this; however, it was noted that performance improvements may exist in the DTAC library that

was used. Additionally, the algorithm, as coded and tested, computed the information score on each path unconditionally, storing it in case needed. We later realized that for a fixed number of paths, the memory cost for which is small, first the number of double-dummy tricks could be computed and then the information score would only be computed if needed to break a near-tie. This on-demand algorithm would require reserving a time allowance for a potential tie-breaker, if using a fixed-time algorithm which attempts to compute additional paths until a time limit is reached.

3.6.1 Future Work

However, our approach was somewhat simplistic, as we did not take a deeper look at how to bring all of these aspects into a single metric for computation. For example, the information score was used only to break near-ties, should the tolerance for ties be softened? Similarly how should the first and maybe second standard deviations be incorporated into these computations?

We might in future work model $Result(s, a) = d * double\_dummy + i * information\_score - s * \sigma$ and then attempt to find optimal weights for d, i, s by playing and replaying a set of deals.

We investigated information score as an estimate of how defender plays affected the information available to the declarer on the next trick. The same analysis could be used by declarer for themselves. For example, in the strategy-fusion example of Section 2.3.2.1, the correct line could be achieved if declarer computed the combined defender score after each possible trick. That is after playing the $\heartsuit A$ declarer can claim all three tricks whereas after playing a spade to the Ace, the same might not be true.

Our algorithm produced definite constraints and then gave equal weight to each possible deal produced by the dealer. This incorrectly assumes that all players follow their partnership rules at all times. In practice players sometimes stretch boundaries. Therefore, it would be interesting to weight positive and negative inferences and use those weights in computing averages and confidence intervals. This could even include learning specific tendencies of human opponents by ingesting previously played deals.

Also missing from our analysis were certain types of negative inferences based on the non-occurrence of some earlier event. For example, if a defender had a play that would have defeated the contract on a certain distribution of the cards but did something else, one could eliminate that distribution at some probability.

An example of such is Figure 3.13 sourced from [Stewart, 2015]. Against 4♠ West leads the ♡2

	♠J94	Vul: EW
	♡108	
	♢A874	
	♣KQ105	
♠85		♠Q63
♡2	N W E S	♡A7643
♢Q10952		♢J3
♣J6432		♣A97
	♠AK1072	
	♡KQJ95	
	♢K6	
	♣8	

South	*West*	*North*	*East*
1♠	P	2♣	P
2♡	P	2♠	P
4♠	P	P	P

Figure 3.13: Negative inferences

to East's ♡A. The ♡3 is returned and West ruffs with the ♠5, the ♣3 is returned to the King and East's ♣A, and East returns the ♢J.

Having lost three tricks already, South must take the remaining 10 with the only possible loser coming in spades. Declarer must determine if the spades are split as shown in Figure 3.13 requiring a finesse, or if West's original spade holding was Qxx as in Figure 3.14 in which case the ♠A then ♠K is the correct sequence.

Probabilistically the heart distribution increases the chances of West holding the ♠Q, and this is borne out by robot play as both GiB and our MCTS went down on this hand playing the ♠A then ♠K. MCTS believed that the failing line was .3 tricks stronger than taking the finesse with no difference in standard deviation.

However, assuming East and West are competent players, a strong inference exists to place the ♠Q with East. When in with the ♣A East would have played another heart for West to ruff with the Queen while South was forced to follow and North was unable to overruff. Missing the ♠Q this would be a clear play for East to defeat the contract; therefore, the only logical conclusion is that

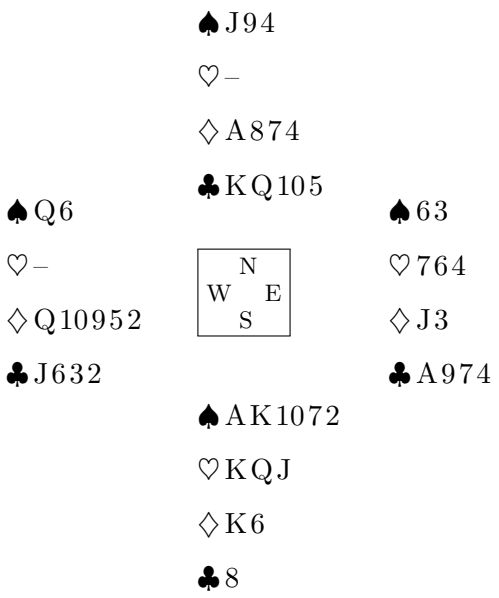


Figure 3.14: Negative inferences: alternate

they knew West would not be able to ruff higher than dummy's Jack.

An enhancement therefore exists for play algorithms that would take into consideration that on all of the layouts where West had the ♠Q, the hand would have been beaten earlier, thus reducing or eliminating any probability associated with such hands in the Monte Carlo simulation.

Chapter 4

Uncontested Bidding

An *uncontested* or *constructive* auction is one where one side does all the bidding while the opponents stay silent by passing throughout. Using 70,000 deals sourced from the Vugraph Project [Sarantakos, 2020] of national and international play from 1996-2010, an analysis showed that almost 46% of deals are uncontested. This suggests that it is a reasonable endeavor to study the sub-problem of partnership bidding free from opponent interference.

Definition: for the purposes of evaluating different results in constructive situations, we define *constructive par* as the maximum duplicate score the bidding side could achieve ignoring the opponents' scores. This is different from the standard definition of *par* (see Appendix) as we seek to eliminate random positive IMP rewards that reflect a lack of opponent bidding. For example, if we bid to 3♡ making and on the given deal the opponents could make 3♠, we should not be rewarded with a positive IMP score as in our experiment the opponents will not be allowed to bid. Or if we bid to 4♠ making 10 tricks, and the par result was the opponents saving in 5 clubs or diamonds, that should not be considered a better result than some other deal where we also make 4♠ and that is also the par result.

4.1 How Humans Bid

Bidding methods combine the application of rules, judgment and insight in various situations with the goal of maximizing the expected score. The Laws of the game and the conditions of contest for a particular event or tournament determine the framework for what is legal, but leave many decisions up to the individual partnerships. As such there are many different bidding systems and conventions between partners which affect the meanings of certain bids and therefore judgments

about such bids. Importantly, whatever agreements are in place must be disclosed to the opponents. While we will discuss how these variations might be modeled, for the purposes of this chapter we will generally assume an agreed upon system such as Standard American [ACBL, 2020].

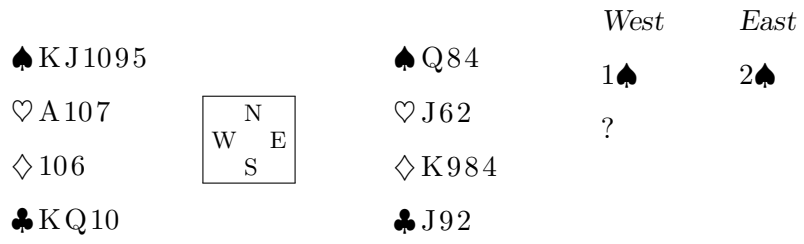


Figure 4.1: Partscore bidding

Opening and responding bids tends to follow a prescribed script, often easily when only a single choice matches the given hand, but other times requiring some judgment when at a boundary between two or three options. For example, in the auction in Figure 4.1, West's opening bid of 1♠ shows 5 or more spades, and opening hand values (typically 11 or 12 HCP or more depending on agreements[1]). In this case, it is a strict application of rules, with no other option fitting and therefore no judgment required. Change the $\heartsuit A$ to the $\heartsuit Q$, and now with a bare minimum of 11 HCP and no Aces (a secondary criteria often used by humans) the choices of 1♠ or *Pass* would both need consideration. East's response in auction 4.1 is an application of responder's rules for partner's opening of 1♠, where 6-9 HCP and 3 or 4 spades raises to 2♠. Again, this hand falls squarely in the middle of one choice, so no judgment would be needed, but change the $\diamondsuit K$ to the $\diamondsuit J$, and now many would pass with 5 HCP, while others might still choose to raise to 2♠.

At West's next bid they must consider whether to be satisfied with a partscore in spades, or whether to try for a game bonus if the side's combined assets might make one. Here the bids typically have an agreed upon meaning, but West must use their judgment whether to *Pass*, try for game via one of the bids between $[2NT, .., 3♠]$, or just bid the game of 4♠.

4.1.1 Skill Evaluation

How well do humans bid without competition? Using only the auction, we examined the uncontested deals referenced above from the Vugraph Project to see how close to a constructive par was achieved. For each result we computed a score using double-dummy tricks for the actual contract minus the constructive par contract converted to IMPs (see Appendix) which would be in range

[1]High Card Points, see 2.1.11.1

$[-24, 0]$ with 0 a perfect score.

Negative scores were achieved in many ways:

- An underbid across a bonus boundary, e.g. contracting for a partscore when enough tricks were available to make a game bonus;
- Any overbid of more tricks than were available, resulting in a minus duplicate score; and
- An underbid by trump suit, that is a making contract worth 20 or more points less than the top spot e.g. 2♣ making for +90 when 3♠ was makeable for +140.

With the two underbid categories occurring on 17% of the deals, and overbidding another 14%.

Metric	IMPs per deal	Range
Constructive par + double-dummy	-3.77	$[-24, 0]$
Par + double-dummy	-3.04	$[-24, 24]$
Par + actual tricks	-2.72	$[-24, 24]$

Table 4.1: Human average IMPs per deal

Our observed result for constructive par using double-dummy tricks, par using double-dummy and par using actual tricks are in Table 4.1. This matches our expectations for these metrics: constructive par eliminates all positive IMP scores by eliminating opponent results; par comparisons will include some positive IMP results where the opponents failed to reach a lower scoring higher bid such as a save (see Appendix); and declarer will sometimes take more tricks than expected by double-dummy using actual play and thus increasing IMPs.

4.2 How Computers Currently Bid

There is little published work indicating how the commercial robots bid, but it is generally believed that they follow similar methods to those proposed in [Ginsberg, 2002]. A database is kept for matching hands to bids such as the choices of 1♠ and 2♠ in Figure 4.1. For later auction bids the database can be consulted for the meaning of certain bids, but this is where simulation can be valuable. Using the same tools of MCTS discussed in the previous chapter, hands can be drawn consistent with the current auction and then analyzed via double-dummy.

For example, West's second bid in Figure 4.1 might be guided by such a simulation using partner's 2♠ bid as well as the opponents' failure to bid. If 10 or more tricks seem likely[2] or unlikely, 4♠ or *Pass* may be chosen respectively.

Unsurprisingly computers have no problems matching a bid to a pre-defined database for opening and responding bids, and are also very good at decision bids where an MCTS simulation can answer the question of what to do next.

However, computers struggle in the in-between space where game (or slam) is possible, but more information is needed. A human might make a *game-try* by selecting one of the bids in $[2NT, .., 3♠]$ using previous partnership agreements on whether this shows length, strength, shortness, or asks for help[3]. The hope would be for one partner to convey enough information through a bid for the other partner to make a better decision between game or not; however, the introduced risk of this bid is now playing in 3♠ instead of 2♠ when the game-try is rejected. Reaching the higher contract of 3♠ after a declined game-try may result in an unsatisfying score if only 8 tricks are taken.

			West	*East*
♠AK94		♠Q	$1NT$	$2♢_1$
♡76	N W E S	♡AJ953	$2♡$	?
♢AK76		♢QJ95		
♣Q73		♣AK10		

[1] 5+ hearts, forces a 2♡ response

Figure 4.2: Slam bidding

Similar to game-bidding, slam-bidding often involves several rounds of bidding and exploration past the initial bids, where the hope is to pass enough information to enable a partnership to better judge if a slam should be bid. Consider the example in Figure 4.2 where computers have struggled. Looking at both hands across a 100-path Monte Carlo in Table 4.2, 6♢ rates to be successful almost all of the time, while $6NT$ may sometimes succeed and 6♡ may rarely make.

[2] more in appendix on scoring

[3] SAYC uses length

Suit	Tricks
♣	10.53 ± 0.61
♢	11.98 ± 0.14
♡	10.65 ± 0.61
♠	10.12 ± 0.74
NT	11.34 ± 0.47

Table 4.2: Expected tricks actual hands

The first 3 bids followed the predefined script: 1NT showing 15-17 HCP and a balanced hand (at most 1 doubleton), 2♢ was a Jacoby transfer showing 5 or more hearts in many types of hands, 2♡ was a forced response to await further information. When given to several of the commercial programs, this opening portion was mostly identical with some variations based on the methods in use. But what happened next differed wildly in 10 replays of the same hand:

- BridgeBaron in 10/10 bid $1NT$-2♢-2♡-6♡
- Jackbridge in 9/10 bid $1NT$-2♢-2♡-3♢-$3NT$-$6NT$
- Wbridge5 in 10/10 bid $1NT$-3♡-$3NT$-$Pass$ as it was playing a different system
- GiB showed the most promise in 9/10 bidding 1N-2♢-2♡-3♢-4♢-6♢

The difficulty here for the East hand is that the expected number of tricks can vary wildly depending on West's actual hand as shown in Table 4.3.

Suit	Tricks
♣	10.28 ± 1.43
♢	11.81 ± 0.98
♡	11.91 ± 0.83
♠	9.31 ± 2.0
NT	11.89 ± 0.99

Table 4.3: Expected tricks opposite strong NT

While 6♢ appears to be the top spot with the actual West hand in Figure 4.2, with any of the hands in Table 4.4 there is a different and clear best contract that the bidding would hope to

uncover and reach.

West hand	Best contract
♠$AK94$ ♡KQ ♢$A76$ ♣8763	6NT
♠$J934$ ♡$KQ8$ ♢$AK6$ ♣$QJ3$	6♡
♠$A93$ ♡$Q8$ ♢$AK6$ ♣$QJ953$	6♣
♠$K943$ ♡$Q8$ ♢$AK6$ ♣$QJ53$	3NT

Table 4.4: Best contracts opposite various hands

It is likely that each of the bots used a simulation before the final bid, picking what seemed to be the most likely contract given the incomplete information available at the time.

This is where humans use judgment and intuition in ways that are extremely hard to program as they don't fall into the simple rule or simulation-based decision making categories.

4.2.1 Skill Evaluation

Similar to the tabulation for humans shown in Table 4.1, we sought to evaluate the strength of the existing bidding engines. However, only limited data was available for this analysis from WCBC play so we computed our own. Even though it is not considered as strong a bidder as other top programs, we used GiB as it provides a command-line interface. We setup four robots to bid random deals until we had collected 10,000 with constructive auctions.

GiB performed at -4.15 IMPs per deal versus constructive par, which at 0.4 IMPs worse than the result for humans in Table 4.1 is considered significant. While analyzed on different data sets, we believe the size of the robot data was large enough to be generally representative and comparable to the human number.

4.3 Reinforcement Learning

Reinforcement Learning [Sutton et al., 1998] is an iterative algorithm for determining which actions for a given state maximize a reward (or reduce a penalty). At the core is a strategy function $F(s) \rightarrow a$ which indicates an action to take given a current state and then an updater $U(F, s, a, r) \rightarrow F'$ which takes the reward from applying the action to the given state and produces a new strategy. A *model-based* algorithm is one which predicts the expected reward from taking a specific action through a model of the transition probabilities. This contrasts with a

model-free algorithm that samples how well a predicted action was rewarded in order to update the *policy* for future predictions.

4.3.1 Q-learning

Quality learning or *Q-learning* [Watkins and Dayan, 1992] is a model-free technique used when a problem presents a discrete set of states and actions. A *value function* $Q[S, A]$ is maintained indicating for each state s the expected value of doing each action a. The standard machinery for this is the Bellman equation [Bellman, 1952], a recursive formulation of the short-term reward r plus a discounted (β) long-term reward for a given state-action combination.

$$Q(s, a) = r(s, a) + \beta \max Q(s', a')$$

Learning the optimal action for a given state is done by repeated iterations of applying the latest policy to the current state, periodically updating the policy based on observed rewards.

4.3.2 Deep Q-learning

When a problem, despite being discrete, becomes too complex to keep a complete table $Q[S, A]$, researchers have introduced neural-networks as an approximation of the Q-function in a *deep Q network* ("DQN") [Mnih et al., 2013].

The salient feature of a DQN is how the model automatically extracts features through back-propagation, removing the need for any human-engineered features. A DQN can be vulnerable to overfitting, so this is tackled using two main techniques:

- *Experience replay* collects data in mini-batches, and only updates the network after a certain batch size is reached. By ensuring more variation in the data this reduces the chance of overfitting.
- A *target network* approach maintains two separate networks, one which is used for Q-value lookups and another which stores the network updates from experience replay. After a large number of batches, the two networks are synchronized. This is used to stabilize the training and lessen the problem of chasing a moving target.

A common choice for training a DQN is the use of a Huber loss function [Huber, 1992]. This function reports mean squared error when the error is small, but for large errors switches to absolute error to reduce noisy outliers:

$$L(\delta) = \begin{cases} \frac{\delta^2}{2} & \textbf{if } |\delta| \leq 1 \\ |\delta| - \frac{1}{2} & \textbf{otherwise} \end{cases}$$

The Adam learning algorithm [Kingma and Ba, 2014] has become a popular replacement for the classic gradient descent for problems with non-stationary objectives and noisy gradients. The algorithm combines two algorithms: Adaptive Gradient ("AdaGrad") [Duchi et al., 2011] and Root Mean Square Propagation ("RMSProp") [Tieleman and Hinton, 2012]. It does so by calculating an exponential moving average of the gradient and gradient squared, and adapting the RMSProp learning rates to use both the first and second moments.

We used all of these techniques when building our model described in the next Section.

4.4 Learning to Bid Unopposed

It is clear from the game and slam decision hands just presented that a new approach to computer-based bidding is needed. In view of the success of AlphaGo and similar programs, it seems reasonable to try and use reinforcement learning to model Bridge bidding judgment.

4.4.1 Definition

To model a constructive auction, we need consider only the actions of two cooperating partners with a shared goal of reaching the optimal contract. We can define an auction A as a series of bids b_i selected from the set of bids $B = [1\clubsuit, 1\diamondsuit, ..., 7\spadesuit, 7NT]$, remembering that for a bid to be legal it must either be at a higher level than the previous, or the same level but with a higher denomination ranking low to high $[\clubsuit, \diamondsuit, \heartsuit, \spadesuit, NT]$. In the constructive auction instead of the first bid, the player may select $Pass$ instead and the auction continues with all bids still legal. For all subsequent choices a player may select a legal bid continuing the auction, or $Pass$ which ends the auction. The calls of double and redouble, used only in competitive auctions do not need to be considered or represented in this sub-problem.

Except in the case of two passes (a passout), when the auction ends the final contract is the last bid made. The intermediate bids do not matter except for determining the declarer, that is in an auction such as Figure 4.3, the final contract of $4\spadesuit$ would be declared by West as the first to bid in spades even though East made the final bid.

West	*East*
1♣	1♡
1♠	4♠
Pass	

Figure 4.3: Uncontested auction

It is also important to note that we are unable to reinforce the model and evaluate the goodness of an individual bid as the auction cannot be scored until it completes. However, this should not prove to be an issue as this is common to reinforcement learning problems.

4.4.2 Representation

To represent the directly observable state, we need 3 parts:

- the cards we hold
- the current auction
- the vulnerability

The cards we hold can be represented as 13 selected of a 52-bit map of $[\clubsuit 2, ..., \clubsuit A, \diamondsuit 2, ..., \diamondsuit A, \heartsuit 2, ..., \heartsuit A, \spadesuit 2, ..., \spadesuit A]$. The current auction can have a bit to represent an initial pass, plus 35-bits representing the bids that may have happened. We do not need to represent the final pass as that leads to a terminal state. We also need a final bit indicating if our side is vulnerable or not (this can affect the scoring and therefore the reward/penalty for a contract).

Vul(1)	P(1)	bids(35)	13-card hand(52)

Figure 4.4: Uncontested bit representation

This 89-bit vector will be the input state to our model, and is a lossless representation of our knowledge. The output vector of actions will be *Pass* plus the 35 bids, with only legal bids considered.

4.4.3 Rewards

The goal of the constructive auction is to reach the optimal contract for each hand based on the scoring format (see appendix). It is not enough to consider simply the duplicate score, as the match/tournament scoring format should be an important consideration in the reward structure. Continuing our evaluation trend, we choose IMPs both because team play is considered the highest level of Bridge competition, but also because it is easier.

Our reward for any particular auction will then be the score for the final contract using double-dummy tricks minus the constructive par and converted to IMPs. This IMP score will therefore be in range [0, -24] since the best we can do is reach constructive par. In addition, to both punish overbidding and lightly model opponents who would start doubling if we frequently overbid by two tricks or more, for scoring purposes we will double any contract which goes down two or more.

Since both partners can be assumed to be playing the identical system, our hope is that the model can infer a hidden expectation of partner's holdings. Thus for this experiment, we chose not to model partner's expected holding, nor to provide such an estimate either learned or rule based as an input to the model.

4.4.4 Model

We used a Deep Q-learning model, made available and easy to use by OpenSpiel [Lanctot et al., 2019], which builds on top of the machine learning libraries provided by the Tensorflow project [Abadi et al., 2015]. OpenSpiel provides a framework for any game to plug in a model which provides: legal actions in a given state, a state transition function for applying a chosen action to a state, and a rewards function for scoring a terminal state. At the driver layer many different models including DQN are available for use with complex configuration parameters including: layers, target update speed, optimizer, loss function.

Having built a game module for uncontested bidding, we trained a policy network using a four-layer DQN with 1024 nodes per layer, the input state as described and the output probability for each of the 36 actions. In the Bellman equation (see Section 4.3), a discount factor of one ($\beta = 1$) was used to represent that there is no penalty for longer versus shorter auctions. We used the Adam optimizer and Huber loss with network updates every 200 steps. As common with these sorts of models, a number of training attempts were required in an attempt to manually tune the ϵ hyperparameter, which is the probability that the model output will be ignored in favor of random

exploration of any legal action. We found that starting ϵ at 0.1 and decaying to 0.05 was good for this problem.

4.4.5 System to Learn

In any AI system built to outplay humans, an important consideration are the human-engineered constraints put in place. For example, the now famous "move 37" from game two of the AlphaGo versus Lee Sedol match was originally dismissed by human commentators who initially labeled it a mistake, only to later realize its sheer brilliance. Had AlphaGo been limited to only moves to those played by humans in similar situations, this creativity would not have been found.

However, while tempting to give the model freedom to select any bid given any hand, any useful computer bidder should be compatible with both another robot or a human. So we settled on a hybrid approach which constrained only opening bids, but afterwards allowed anything.

For opening bids only, we enforced certain selection restrictions utilizing a rules-engine:

- A hand with less than 10 HCP was required to Pass;
- A hand with 10-13 HCP would be offered Pass plus the matching rules-based option; and
- A hand with greater than 13 HCP would not be allowed to Pass.

These limits were chosen as a stretch of the boundaries under which a human might operate. Most humans open at 11 or 12 HCP, and very few if any human players would pass a 13 HCP hand, thus our choice of a 10-13 boundary where we allowed the model to be aggressive or conservative. We used standard rules, and if multiple bids matched such as 1♠ and $1NT$ we left it to the model to choose. The following rules were used, where *balanced* refers to any hand with at most one 2-card suit (5-3-3-2, 4-4-3-2, 4-3-3-3):

- $2NT$: 20-21 HCP balanced (see above);
- $1NT$: 15-17 HCP balanced (see above);
- 1♠ : 5 or more spades, no longer suit;
- 1♡ : 5 or more hearts, no longer minor, more ♡ than ♠;
- 1♢ : not a 1♠ or 1♡ opener, more ♢ than ♣; and
- 1♣ : not a 1♠, 1♡ or 1♢ opener.

For all actions after the opening bid, the model could consider all legal bids plus pass as output from our DQN.

In this experiment, we did not take into account that players vary their opening bidding strategy depending on whether prior players passed. For example, in actual competition, the second bidder would typically take a more conservative bidding approach if dealer chose to pass; the third bidder might be more aggressive in their bid in order to distract the next opponent if the first two players passed; and after three passes, the fourth player has the option to pass and collect a score of zero.

4.4.6 Training Data

Our initial training data included one million randomly generated deals with precomputed double-dummy analysis. However, this ran into a local-minima of sorts. In 16% of deals the bidding side could not make anything (we arbitrarily bid as EW). Additionally, in 50% of deals only a partscore was available.

Our initial trained model, other than when forced to open, found that a strategy of always passing would often get the top reward of zero or a small loss to a making partscore.

Based on this issue, we felt that that it was appropriate to exclude almost all deals where nothing could be made. In this way a strategy of always passing will always incur a penalty and thus we believed would encourage the model to do some bidding. Thus we updated our training algorithm to bid as NS on any deal where EW could not make anything.

Each deal from the updated training set was bid once, with updates to the target network occurring every 200 deals.

4.4.7 Experimental Results

After re-training the model with our updated data, we then evaluated against a corpus of 10,000 randomly generated hands where we similarly matched the training data in choosing a bidding side as the one that could make something.

Our model scored an average of -4.45 IMPs against constructive par per deal. While it did show that we were able to model bidding using reinforcement learning, disappointingly this result was slightly worse than GiB and much worse than our human target. While we had hoped that hands such as the example would result in the ability for our model to learn cooperative communication, instead we found a an extremely conservative bidder taking advantage of the lack of opponents to Pass at a much higher rate than humans.

Shown in Figure 4.5, a series of hands which a human would open but were usually passed using our trained model.

Hand	Human choice	Model	Score
♠KQ642 ♡J2 ♢K87 ♣A54	1♠	Pass	-4
♠832 ♡AKT75 ♢A65 ♣Q8	1♡	Pass	-2
♠AK9865 ♡3 ♢A9 ♣9842	1♠	Pass	-9
♠984 ♡8 ♢AK32 ♣AQ832	1♣	1♣	-1

Table 4.5: Conservative model openings

Similarly, Figure 4.6 lists a series of hands facing an opening bid on which a human would respond more often than not but passed using our trained model. Presumably bidding introduced the risk of a higher negative score, where stopping in the lowest possible partscore when game was unlikely was likely to score better.

Hand	Opening Bid	Human response	Model	Score
♠QJ972 ♡A ♢QT64 ♣T53	1♢	1♠	Pass	-3
♠84 ♡AQ97 ♢J976 ♣T43	1♠	1NT	Pass	0
♠K63 ♡A965 ♢QT65 ♣J9	1♣	1♡	Pass	-1
♠Q63 ♡Q2 ♢Q97 ♣QT852	1♠	2♠	Pass	0
♠JT76543 ♡AJ42 ♢− ♣T3	1♣	1♠	Pass	-14
♠KQ85 ♡9 ♢AJ42 ♣AJ93	1♠	2NT forcing or 4♡ splinter	4♠	0

Table 4.6: Conservative model responses

The last hand in Figure 4.6 shows that the model didn't always Pass; it was able to learn the bonus reward for successfully bidding game. However, on this hand a standard choice would be to show the spade fit and strength via a conventional bid, allowing for the chance that the opening hand is more than a minimum such that enough tricks for slam might be available. On other layouts of the cards, 12 or 13 tricks might have available in spades, instead our model quickly claims what turned out to be the top reward of game without further exploration.

As another example with ♠ Q 6 3 ♡ Q 2 ♢ Q 9 7 ♣ Q 10 8 5 2 a human would raise to 2♠ (as opposed to passing), as the bid has a competitive value in forcing the next opponent's bids to come at a high level. Without competition, the model found no value in these sorts of low level raises since

such a bid could lower the reward.

These conservative leanings reflect what we found via double-dummy analysis, namely that the majority of hands make only a partscore, and only a very small portion of hands can make a slam. The model seems to have discovered that while biding may turn a score of -2 or -3 into a 0, it could also get too high and quickly score -5 or worse. For example, if 7 tricks are available in spades and Notrump and 9 in hearts opposite a 1♠ opener passing scores -2, where a bidding sequence to anything other than 2♡ or 3♡, such as 2NT down one would score -5.

Of specific interest, our model does no better than other programs on the hand in Figure 4.2, bidding 1NT - 3NT.

4.4.8 Conclusions

The constructive bidding problem provided the opportunity to study two-player cooperation using reinforcement learning. We showed that an unsupervised system could be trained to produce reasonable results that are not far off of handcrafted systems developed over many years. Our model took advantage of the lack of competition and found that on a normal data set an extremely conservative approach could be only a bit worse than existing rules-based engines. This is not a perfect comparison, as the human and robot data were extracted from auctions that had the potential to be competitive but ended constructive, whereas in our experiments the auctions were forced as constructive. In an actual competition, this system's results would be dependent on the deals being played: more deals making only partscore would likely fare well as the model is not susceptible to overbidding, whereas the opposite if faced with a higher than normal number of deals making game or slam.

A similar work [Yeh et al., 2018] was published before this one tackling the same problem; however, we became aware of their work too late to incorporate any of their ideas into our system design. Yeh et al. also use Deep Q-learning with the standard RMSProp (see Section 4.3.2) for optimizer, but with a newly defined *penetrative* Bellman equation they believed was more suited to the problem of Bridge Bidding. They observed that there are typically less than 6 bids between the partners in a constructive auction and that the discount factor $\gamma = 1$, and thus performed a recursive simplification of the Bellman equation which they named "penetrative". Their work offers more in-depth coverage of tuning hyperparameters and also considers allowing both a structured and wide-open system, whereas we only present our best values and then focus on the ramifications of such a trained model.

It is our belief that in order to be explainable in competition with humans, the path forward both for competitive and constructive bidding is more of a hybrid system that balances model freedom with suggested bids as extracted from a rules-based system throughout the auction.

4.5 Future Work

4.5.1 Auction Meanings

For the purposes of our experiments, we assumed the computer bidder knew the meanings of all bids, and thus these understandings did not need to be inputs to the model. Rong et al. (2019) used one network to estimate partner's hand to provide as input to the network which selects a bid; further research in this direction appears to be worthwhile. Rather than training different networks for each different system variation, the model should take advantage of any available rule-based knowledge about partner's bids and include them as inputs to the model. Thus the trained model becomes more generic and can be applied to a wider range of bidding systems.

4.5.2 Competitive Encoding

While we were unable to satisfactorily train a model for competitive bidding in time for this work, we did create an extremely efficient lossless encoding which may be of some use to future work.

As before, we can encode our hand as 13 of 52 bits selected in a bit-map. Vulnerability now requires 2 bits to represent the four possibilities of [none, us, them, both]. To encode a lossless representation of the auction is a bit harder. We know that any bid can only happen once, and must occur in-between bids which are lower and higher; however, the number of passes and whether any doubles or redoubles happened are important to capture.

We then note that after a bid is made, there are six possible legal sequences which include a *Double*, we combine these and incorporate two other possibilities where the bid was made but without doubling, or the bid was not made:

0. Bid did not occur
1. No doubling
2. *Double*
3. *Double Redouble*

4. *Double Pass Pass Redouble*

5. *Pass Pass Double*

6. *Pass Pass Double Redouble*

7. *Pass Pass Double Pass Pass Redouble*

These 8 possibilities fit into 3-bits, after which there can be 0, 1, or 2 passes before the next bid which fits into 2.

Passes(2)	Bid+risk(3)

Figure 4.5: Single bid bit representation

We also need 2 leading bits to indicate if 0, 1, 2, or 3 passes proceeded the first bid of the auction. Bringing it all together to represent the current state needed to select a bid we form a 231-bit string.

Vul(2)	P(2)	13-card hand(52)	bids(5*35=175)

Figure 4.6: Auction state bit representation

Note that this is always relative to the current bidder.

South	*West*	*North*	*East*
P	1♣	X	1♠
2♡	X	3♡	P
?			

Figure 4.7: Example encoded auction

As an example ignoring the first 56 bits and using hex32, auction 4.7 would encode as in Figure 4.8:

1	2	0	0	1	0	0	0	2	0	0	0	0	9
P	1CX			1S				2HX					3H P

Figure 4.8: Example competitive auction bits

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Having built an MCTS card player comparable to commercial software, we showed how the information score algorithm could be used as an estimate of the opponents' information on a hand and to enhance the base algorithm. By using this we were able to make a noticeable improvement in move selection but at a performance cost which will need to be addressed. We also showed additional improvement for IMP scoring using confidence boundaries such as σ to make safer plays.

In the bidding phase, we trained a model for the uncontested bidding problem using deep Q-learning. We showed that reinforcement learning could be used for constructive bidding in place of either rules-based or simulation-based bidding algorithms at only a small additional loss in IMPs per deal versus constructive par. Our results were different than similar bidding studies, as we used different models both for inputs and learning and a modified measuring system.

Although unable to successfully train a model for competitive bidding, we proposed an efficient lossless encoding which may be useful in future work.

5.2 Future Work

Recent work on bidding has examined cooperative and competitive learning by agents, such that testing of trained models is done again by agents.

How well would these models do if playing with or against a human opponent? Can they explain what their partner's bids mean? This is a requirement in actual play.

What can they learn about certain human opponents or partnerships? Some may be aggressive

or conservative in bidding, or perhaps regularly violate agreements: e.g. opening $1NT$ with 14 HCP despite a 15-17 agreement. To our knowledge these types of inferences have yet to be incorporated into any such programs, but would be useful for any Bridge AI program.

The next stage in bidding systems will likely be a hybrid of existing rules-based and simulation-based systems, with a reinforcement learning trained model.

While there have been improvements on all fronts, we will not know how significant an improvement they are taken as a whole unless an effort is made to create an end-to-end bridge player that incorporates all of the recent advances. Once such a player has been created, it should first be entered into the WCBC to see how it fares against the established commercial robots.

After that, similar to the AlphaGo match against Lee Sedol, an exhibition match should be played of 90 boards over 2 days pitting a team of four robots against four international level human players.

Appendix: Scoring in Bridge

Duplicate Scoring

Scoring in duplicate is two-part. First, based on the deal result of contract and number of tricks taken, a *score* is achieved using the following formulae. If the total number of tricks taken by declarer is greater than or equal to the *contract_tricks* plus 6, the contract is said to be *made* or *making*, and the score is as follows:

$tricks = total\_declarer\_tricks - 6$

$score = tricks * trump\_value + bonus$

Where the *minor* suits of clubs and diamonds are each worth 20 points per trick, the *major* suits of hearts and spades are worth 30 points per trick, and Notrump is worth 40 for the first trick, 30 for each additional trick. If the $contract\_tricks * trump\_value < 100$ the contract is a *partscore*, for which the making bonus is 50. When the $contract\_tricks * trump\_value >= 100$ (3NT, 4♡, 4♠, 5♣, 5♢)the contract is a *game* and earns a game bonus of 300 when not-vulnerable, and 500 when vulnerable. *Overtricks* are any tricks made in addition to the number contracted, e.g. taking 11 tricks in 3♠ you are said to have "made with 2 overtricks". Note the important distinction of bidding 3♠ and taking 10 tricks from bidding 4♠ and making the same. In the latter case the contract earns a game bonus if made since contract tricks of 4 times spades worth 30 is greater than 100 ($4 * 30 = 120$) while the former is not.

When $contract\_tricks = 6$ a side has contracted for all but one of the tricks and this is called a *small-slam* or just *slam*. If a side bids 7 and thus contracts for all of the tricks this is called a *grand-slam*. Making a slam or grand-slam earns an additional bonus added to the game bonus and base score as per Table A.1.

	not Vulnerable	Vulnerable
Small	500	750
Grand	1000	1500

Table A.1: Table of slam bonuses

To score a making doubled or redoubled contract an augmented formula is used which separates contract tricks from any overtricks taken.

$score = insult + overtricks * overtrick\_value + contract\_tricks * trump\_value * trick\_multiplier + bonus$

	insult	non-Vul overtrick value	Vul overtrick value	trick multiplier
Doubled	50	100	200	2
Redoubled	100	200	400	4

Table A.2: Table of doubled and redoubled multipliers

The slam bonus award is unchanged, but the game bonus is now awarded when $contract\_tricks * trump\_value * trick\_multiplier >= 100$. So lower level contracts, for example, 2♠X and $1NTXX$ receive a game bonus when made.

Contract	Tricks	not Vulnerable	Vulnerable
4♠	10	420	620
3♠	9	140	140
3♡	10	170	170
2♠ X	8	470	670
1♣ XX	8	430	630
2NT	9	150	150
3NT	9	430	630

Table A.3: Scoring examples of making contracts

Note: "X" is shorthand for a doubled contract, and "XX" for one that is redoubled.

When the number of tricks taken is less than the contract tricks, the contract is said to "have gone down". This can include the number of *under-tricks*, e.g. "four spades down 2" means that

8 tricks were taken (2 less than the required amount).

When a contract fails, the opponents of the declaring side receive a score as follows based solely on the number of under-tricks, the vulnerability and any doubling as follows in Table A.4.

	not Vulnerable			Vulnerable		
under tricks		doubled	redoubled		doubled	redoubled
1	50	100	200	100	200	400
2	100	300	600	200	500	1000
3	150	500	1000	300	800	1600
4	200	800	1600	400	1100	2200
each additional	+50	+300	+600	+100	+300	+600

Table A.4: Scoring examples of failed contracts

Vulnerability

In the previous section different scores occur for many contract types due to a mentioned but undefined notion of *vulnerability*. In duplicate bridge this is simply a rotating concept based on the board number, whereby either or both sides may or may not be vulnerable.

This pattern is repeated for each subsequent group of 16 boards, remembering that North is the dealer on board 1,5,9,... East on 2,6,10,... and so forth.

Vulnerable	Board numbers
Neither	1, 8, 11, 14
NS	2, 5, 12, 15
EW	3, 6, 9, 16
Both	4, 7, 10, 13

Table A.5: Table of vulnerabilities

This concept can be important to factor in when bidding as it can change the payoff function in certain risk/reward situations.

Par Result

On each deal there exists an equilibrium score such that neither side can make any further improvements without reducing their net score. The term for this result is *par* and can be used as a baseline for evaluations of an achieved score, or a bid to contract.

Par can be either the best positive result of one side, or in some cases a negative result for a side that is both a higher bid than the other sides best positive result but yet smaller than that score. It is assumed whenever a side has bid higher than they can make that they are doubled. Double-dummy analysis is used to provide the number of tricks available in each contract/declarer combination.

For example, if one side can make 7 tricks in NT, 8 tricks in clubs, and nine tricks in hearts, then the par result would be +140 in 3♡. Whereas if with both sides non-vulnerable, NS can make 10 tricks in hearts, while EW can make 9 tricks in clubs, the par result would be 5♣*X* down 2 for -300, which is a better result for EW than -420 against 4♡

Matchpoint Pairs Scoring

Once a score for a deal is achieved by all tables it can be converted to points.

In *Pairs scoring* a board is played at several tables by different combinations of players and a pair's result on a deal is converted to *Matchpoints* ("MPs") as follows: 1 point for every score beaten and 0.5 points for every score tied. This is done for NS pairs taken against the scores from the NS perspective and then again for EW pairs from theirs. So for a given playing of a deal there is a fixed number of points available to be split by the opponents equal to the number of comparisons that will be made.

Round	NS	EW	Contract	Tricks	NS Score
1	1	1	3♡ N	10	+170
2	13	12	4♡ N	10	+420
3	12	10	4♡ N	10	+420
4	11	8	3♡ N	9	+140
5	10	6	3♡ N	9	+140
6	9	4	4♡ N	9	-50
7	8	2	3♡ N	10	+170
8	7	13	3♡ N	10	+170
9	6	11	3♡ N	9	+140
10	5	9	4♡ N	10	+420
11	4	7	4♡ N	9	-50
12	3	5	3♡ N	9	+140
13	2	3	4♡ N	10	+420

Table A.6: Example scores for a single board in 13 table duplicate

In the example deal in Table A.6 which was played 13 times, scores are shown for one direction first by round, then grouped by score in Table A.7 for *matchpointing*[1]. In each case the EW opponents scored 12 minus that number, and the total points given out is $tables * (tables - 1)$. Each board is scored as such and then for each pair all the scores are added up to achieve a final ranking amongst pairs.

Score	Occurrences	Matchpoints
+420	4	10.5
+170	3	7
+140	4	3.5
-50	2	0.5

Table A.7: Example matchpointing for one direction

Suppose the scores of -50 are due to either a perfect but hard to find play by the defense, or a small but costly mistake by the declarer with the combined probability of 25 %.

[1]computing matchpoint scores for each result

Knowing all of this, should you bid 3 or 4♡? An expected-value calculation yields:

$game : 10.5 * .75 + 0.5 * .25 = 8$

$partscore : 7 * .75 + 3.5 * .25 = 6.125$

The correct decision is to bid the 4♡ game. For this particular set of scores, the break-even probability is around 46 % bidding game if higher, and not if lower. However, of course at the time a deal is played you do not know what will happen, so a player is forced to predict how a choice might fare both in the contracts reached and the probability of making a certain number of tricks.

Teams Scoring

In *Teams scoring* a match is played between two teams using a series of duplicated boards that are played at two tables. At one table Team A plays in the NS direction and at the other table they play EW, while Team B sits against them at both places. To convert score to points, for each board teammates simply add up their scores and then convert the net to *International Matchpoints* ("IMPs") a sample of which is in Table A.8. The IMPs scored for each board and then totaled up for each side to obtain a match result.

Difference	IMPs
20-40	1
50-80	2
90-120	3
130-160	4
170-210	5
220-260	6
270-310	7
320-360	8
370-420	9
430-490	10
500-590	11
600-740	12
750-890	13

Table A.8: Partial IMP table

The highest number of IMPs on a single board is 24 and is awarded for a net score of 4000 or more [Francis et al., 1995]. Of note is that the table is non-linear in that it gets harder and harder to score the next IMP.

Board	T1 Contract	T1 NS Score	A IMPs	B IMPs	T2 Contract	T2 NS Score
1	3N S +2	460	-	-	3N S +2	460
2	2♡ S +2	170	-	10	4♡ S =	620
3	2♠ S +1	140	-	-	2N N +1	150
4	4♠ S -1	-100	-	-	3♠ S -1	-100
5	6♡ N =	1430	13	-	4♡ S +2	680
6	2♠ W =	-110	-	12	4♠ W X -2	500
7	5♢ E +1	-620	13	-	6N W =	-1440
8	4♡ W =	-420	-	-	4♡ W =	-420
		Total	26	22		

Table A.9: Sample 8 board match (T1=Table 1, T2=Table 2)

Referring to the sample match in Table A.9, suppose on board 5 where Team A NS at Table 1 bid and make 6♡, the player making a decision whether to explore for slam considered that: the opponents would likely not be in slam, the probability of making slam is P. If slam makes as here, A wins 13 IMPs, but if it goes down, -100 combined with 650 from the other table results in 13 IMPs to the B side. Thus, in this over-simplistic analysis one should bid slam whenever $P > 0.5$.

Now consider board 2, where Team B NS were rewarded with 10 IMPS for bidding to 4♡ when their counterparts did not. Suppose only 9 tricks had been available, then 140 and -100 would result in 6 IMPs to A. An over simplified analysis suggests that the payoff for bidding a making game is very high, and thus at IMP scoring, games should be bid even when the perceived chances of making are greater than 38%. However, this fails to take into account many other outcomes: you bid a doomed game and the opponents double collecting a large penalty, your counterparts are also thinking as you and you both end up in the same game contract, the lead and play may be different and produce a different outcome.

Bibliography

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[ACBL, 2020] ACBL (2020). *Standard American*. http://web2.acbl.org/documentlibrary/play/SP3 (bk) single pages.pdf.

[Bellman, 1952] Bellman, R. (1952). On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716.

[Bethe, 2010a] Bethe, P. M. (2010a). Dtac: A method for planning to claim in bridge. Master's thesis, New York University.

[Bethe, 2010b] Bethe, P. M. (2010b). The state of automated bridge play. *NYU Report*.

[Bird and Anthias, 2011] Bird, D. and Anthias, T. (2011). *Winning Notrump Leads*. Master Point Press.

[Brown and Sandholm, 2018] Brown, N. and Sandholm, T. (2018). Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424.

[Brown and Sandholm, 2019] Brown, N. and Sandholm, T. (2019). Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890.

[Cazenave and Ventos, 2019] Cazenave, T. and Ventos, V. (2019). The {\alpha}{\mu} search algorithm for the game of bridge. *arXiv preprint arXiv:1911.07960*.

[Cowling et al., 2012] Cowling, P. I., Powley, E. J., and Whitehouse, D. (2012). Information set monte carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):120–143.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

[Francis et al., 1995] Francis, H. G., Truscott, A. F., and Francis, D. A. (1995). *The Official Encyclopedia of Bridge*. Amer Contract Bridge League.

[Frank and Basin, 1998] Frank, I. and Basin, D. (1998). Search in games with incomplete information: A case study using bridge card play. *Artificial Intelligence*, 100(1-2):87–123.

[Frank et al., 1998] Frank, I., Basin, D., and Matsubara, H. (1998). Finding optimal strategies for imperfect information games. In *Proceedings Of The National Conference On Artificial Intelligence*, pages 500–508. John Wiley & Sons Ltd.

[Ginsberg, 1996] Ginsberg, M. (1996). Partition search. In *Proceedings Of The National Conference On Artificial Intelligence*, pages 228–233.

[Ginsberg, 1999] Ginsberg, M. (1999). GIB: Steps toward an expert-level bridge-playing program. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 584–593. Citeseer.

[Ginsberg, 2002] Ginsberg, M. (2002). GIB: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research*, 14:313–368.

[Gong et al., 2019] Gong, Q., Jiang, Y., and Tian, Y. (2019). Simple is better: Training an end-to-end contract bridge bidding agent without human knowledge.

[Goodman, 2019] Goodman, J. (2019). Re-determinizing information set monte carlo tree search in hanabi. *arXiv preprint arXiv:1902.06075*.

[Haglund and Hein, 2020] Haglund, B. and Hein, S. (2020). *DDS*. github.com/dds-bridge/dds.

[Hoffmann and Brafman, 2006] Hoffmann, J. and Brafman, R. (2006). Conformant planning via heuristic forward search: A new approach. *Artificial Intelligence*, 170(6-7):507–541.

[Huber, 1992] Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer.

[Joslin and Clements, 1999] Joslin, D. and Clements, D. (1999). Squeaky Wheel. *Optimization. Journal of Artificial Intelligence Research*, 10(5):353–373.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Knuth and Moore, 1975] Knuth, D. E. and Moore, R. W. (1975). An analysis of alpha-beta pruning. *Artificial intelligence*, 6(4):293–326.

[Lanctot et al., 2019] Lanctot, M., Lockhart, E., Lespiau, J.-B., Zambaldi, V., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., Hennes, D., Morrill, D., Muller, P., Ewalds, T., Faulkner, R., Kramár, J., Vylder, B. D., Saeta, B., Bradbury, J., Ding, D., Borgeaud, S., Lai, M., Schrittwieser, J., Anthony, T., Hughes, E., Danihelka, I., and Ryan-Davis, J. (2019). OpenSpiel: A framework for reinforcement learning in games. *CoRR*, abs/1908.09453.

[Legras et al., 2018] Legras, S., Rouveirol, C., and Ventos, V. (2018). The game of bridge: a challenge for ilp. In *International Conference on Inductive Logic Programming*, pages 72–87. Springer.

[Levy, 2017] Levy, A. (2017). Wcbc 2016: The 20th world computer-bridge championship. *J. Int. Comput. Games Assoc.*, 39(3-4):233–238.

[Love, 1968] Love, C. E. (1968). *Bridge Squeezes Complete: Or, Winning End Play Strategy*. Dover Publications.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[Mossakowski and Mandziuk, 2009] Mossakowski, K. and Mandziuk, J. (2009). Learning without human expertise: a case study of the double dummy bridge problem. *IEEE transactions on neural networks*, 20(2):278–299.

[Nielsen, 2001] Nielsen, J. (2001). BuDDy-A Binary Decision Diagram Package. *IT Univ. Copenhagen (ITU), Copenhagen, Denmark: Tech. Rep.*

[Pavlicek, 2014] Pavlicek, R. (2014). Actual play vs. Double-Dummy.

[Rong et al., 2019] Rong, J., Qin, T., and An, B. (2019). Competitive bridge bidding with deep neural networks. *arXiv preprint arXiv:1903.00900*.

[Russell and Norvig, 2009] Russell, S. and Norvig, P. (2009). *Artificial intelligence: a modern approach*. Prentice Hall, 3rd edition.

[Russell and Wolfe, 2005] Russell, S. and Wolfe, J. (2005). Efficient belief-state AND-OR search, with application to Kriegspiel. In *International Joint Conference on Artificial Intelligence*, volume 19, page 278. Citeseer.

[Sarantakos, 2020] Sarantakos, N. (2020). *Vugraph Project*. sarantakos.com/bridge/vugraph.html.

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

[Silver et al., 2018] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

[Smith et al., 1998] Smith, S., Nau, D., and Throop, T. (1998). Computer bridge: A big win for AI planning. *AI Magazine*, 19(2):93–106.

[Stewart, 2015] Stewart, F. (2015). Bridge: January 8, 2015. *The Mercury News*.

[Sutton et al., 1998] Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

[Throop, 1983] Throop, T. (1983). *Computer bridge*. Hayden.

[Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

[Uijterwaal, 2020] Uijterwaal, H. (2020). *Dealer*. henku.home.xs4all.nl/software/dealer.html.

[Ventos et al., 2017] Ventos, V., Costel, Y., Teytaud, O., and Ventos, S. T. (2017). Boosting a bridge artificial intelligence. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1280–1287. IEEE.

[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

[Yeh et al., 2018] Yeh, C.-K., Hsieh, C.-Y., and Lin, H.-T. (2018). Automatic bridge bidding using deep reinforcement learning. *IEEE Transactions on Games*, 10(4):365–377.