Reinforcement Learning Systems
A Study using Bidding in the Game of Bridge

Kalle Prorok
Department of Computing Science*
Umeå University
SE-901 87 UMEÅ
Sweden
Licentiate Thesis 2001

November 12, 2003

Abstract

In this thesis we study the possibilities for a computer to learn the bidding part of the card game Bridge.

It contains a quick overview of machine learning methods and specially the biologically inspired Reinforcement Learning (RL) method, based on reward, where we used Q-learning as an algorithm for RL. Bridge is introduced quickly, analyzed and used as an example of a multi learners environment. The Bridge bidding systems learnt first are useful but below the level of an average club-player and only without opponent's interference. A second try with more detailed information and a better evaluator function resulted in more powerful bidding and also invention of, more or less, new ideas.

The essence of how to coordinate learning is studied in a very small and simple game, the Sum-game, and we found difficulties for the Q-learning algorithm although an evolutionary algorithm (EA) approach seemed to work well. However, it is a slightly unfair comparison because our implementation of the EA reformulates the multi-learner (agent) game into a global, single population, search.

Some new ideas of how to improve the learning, performance and structure are also presented.

The licentiate work is accompanied by a separate report in the robotics control field.

*Now at the dept of Applied Physics and Electronics, Umeå University

Contents

1 Preface **8**

2 Abbreviations **9**

3 Symbols **9**

4 Introduction and Background **10**
- 4.1 What is learning? . . . 10
  - 4.1.1 Feedback from the Teacher/Environment . . . 11
  - 4.1.2 Machine Learning(ML) . . . 11
  - 4.1.3 Operant conditioning . . . 11
  - 4.1.4 Learning by doing . . . 12
  - 4.1.5 Reinforcement learning(RL) . . . 12
  - 4.1.6 Problems during learning . . . 12
- 4.2 Why bridge? . . . 13
  - 4.2.1 The rules of bridge . . . 14
  - 4.2.2 General . . . 17
  - 4.2.3 Bidding system . . . 18
  - 4.2.4 Scoring . . . 20
  - 4.2.5 Available software . . . 22
- 4.3 The goal . . . 24

5 Definitions, Methods & Theories **26**
- 5.1 Reinforcement Learning . . . 26
  - 5.1.1 Introduction . . . 26
  - 5.1.2 Reward . . . 27
  - 5.1.3 Discounted return . . . 27
  - 5.1.4 Markov property . . . 27
  - 5.1.5 Policy . . . 28
  - 5.1.6 Value function . . . 28
  - 5.1.7 Episodic and Continuing tasks . . . 28
  - 5.1.8 Temporal difference (TD) . . . 29
  - 5.1.9 Bellman equation . . . 29
  - 5.1.10 Dynamic Programming(DP) . . . 30
  - 5.1.11 Iterative solution . . . 30
  - 5.1.12 Q-learning . . . 31
  - 5.1.13 Hidden states . . . 31
  - 5.1.14 Applications with Reinforcement Learning . . . 31
  - 5.1.15 Conclusions . . . 32
- 5.2 Artificial Neural Nets (ANN) . . . 32
- 5.3 Genetic Algorithms (GA) . . . 34
- 5.4 Control systems . . . 36
  - 5.4.1 Adaptive Control . . . 37
- 5.5 Game Theory . . . 38

5.6 Multi Agent Systems(MAS) . . . 40
5.6.1 Multi Agent Learning . . . 41

6 Related work **43**
6.1 GARIC . . . 43
6.2 GIB . . . 43
6.3 Game-tree search applied on Bridge . . . 44
6.4 Computer Bidders . . . 44
6.5 Joint-Action Learning . . . 44

7 Overview of my work **49**
7.1 Before my Ph.D-studies . . . 49
7.1.1 Early Bridge Software and Hardware . . . 49
7.1.2 Experiments with Adaptive Control . . . 49
7.2 During my Ph.D-studies . . . 50
7.2.1 A double dummy solver . . . 51
7.2.2 A Control Approach to Reinforcement Learning . . . 53
7.2.3 The man in the box . . . 53
7.2.4 Multi Agent Cooperative Reinforcement Learning . . . 53
7.2.5 Robotic Soccer . . . 53
7.2.6 Joint-Action Learning . . . 54
7.2.7 Large-scale Q-learning . . . 54
7.2.8 The Sum Game . . . 54
7.2.9 Genetic Bridge . . . 54
7.2.10 Boom-Tip Control . . . 54

8 Cooperative Reinforcement Learning in Multiple Agents **55**
8.1 Introduction . . . 55
8.2 Reinforcement Learning . . . 55
8.3 Multiagent Coordination . . . 55
8.4 The Application Task . . . 56
8.5 Representation levels of a hand . . . 57
8.5.1 Trick-taking abilities . . . 59
8.5.2 Scoring . . . 60
8.6 The Learning Task . . . 60
8.6.1 Some Trade-offs and Problems . . . 62
8.6.2 Allowed systems . . . 62
8.7 Results . . . 62
8.7.1 Tested tricks . . . 70
8.7.2 Example set . . . 73
8.7.3 Implementation Performance Optimizations . . . 73
8.7.4 Deal statistics . . . 74
8.7.5 Problems encountered . . . 75
8.7.6 Q - windup . . . 77
8.8 Future . . . 77
8.8.1 Genetic learning . . . 78

8.8.2 Locker-Room agreements . . . 78
8.8.3 Multidimensional representation . . . 78
8.8.4 Bid sequences . . . 78
8.8.5 First round defined . . . 79
8.8.6 Heuristics/Information utilization . . . 79
8.8.7 High gamma to encourage bidding . . . 79
8.8.8 Hierarchical learning of concepts . . . 80
8.8.9 Bayesian Belief Networks . . . 80
8.8.10 1-Agent learning . . . 80
8.8.11 Variations of Q-learning . . . 80
8.8.12 Transforming into a single-agent problem . . . 80

9 Learning bidding systems by using large-scale Q-tables 81
9.1 Introduction . . . 81
9.2 Special properties . . . 86
9.3 Experiments . . . 86
9.4 Results . . . 87
9.4.1 Selecting learning rate, exploration and initial Q-value . . . 87
9.5 Conclusions . . . 101

10 The Sum-game 102
10.1 What is learnt? . . . 102
10.2 Some similar games . . . 103
10.2.1 Listening . . . 104
10.2.2 Thinkforyourselfing . . . 104
10.2.3 Combining . . . 104
10.3 Results . . . 104

11 Acknowledgments and Epilogue 106

A Fuzzy Logic 107
A.1 Fuzzy control . . . 107

List of Figures

1 A typical deal in schematic form. It is dealt by player North who then starts the bidding with East next in turn. North is void in (have no) spades but has a weak hand and passes. East has a strong hand (many honors) and normally opens the bidding by giving a bid in his longest suit, 1 ♠ here. South has an average hand but not enough to enter the bidding with a bid other than Pass. . . . 15
2 The scheme of the bidding with the hands in Fig 1. Three passes end the auction. . . . 16
3 A table with the pairs tournament protocol (paper) from the 7th deal. The resulting topscore points are 2 to pair 6 and 1. . . . 21

4 The International Match-Point scale. It is often used in Teams tournaments to improve the importance of high-level contracts. . 22
5 The result of a match. There is usually 30 VP to distribute between the two teams. A typical match is 24 boards and if the difference is 25 IMPs the match result will be $20 - 10$ VP. 23
6 The reinforcement learning cycle. The Agent senses the state and a reinforcement. It tries to estimate whether this state is good. . 26
7 A simple grid-world. Four actions are possible in each cell; north, south, east and west. Actions off the grid leaves the location unchanged but rewards -1. A move into A locates in a and rewards +5. There are 9 states. 29
8 Co-convergence into the optimal values by interacting policy improvement with policy evaluation. The processes converge because the (thick) lines are not orthogonal. 30
9 A processing node (element). The output y is a function of the sum $(w_1 \cdot x_1 + w_2 \cdot x_2)$. A sigmoidal function is often used. Learning is done by adjusting the weigths w_i. 32
10 A simple artificial neural net with 4 inputs and one output. There are 3 processing nodes in the hidden layer. 33
11 The genetic process. In this case the child gets the first part from its father and the second from its mother. There is also a random change (mutation) close to the middle. The black boxes at the end supplies storage for the fitness value for this particular individual. 34
12 A schematic view of a controlled system. 37
13 Adaptive control. 38
14 The Prisoner's Dilemma. If the thief cooperates with the police and his partner does not, he will go free. The thieves cannot communicate with each other. 39
15 The partnership N-S are able to select a bid first (in clubs) and then E-W decides. On this hand it is assumed N-S are able to take 7 tricks with clubs as trumps and E-W 10 with spades. . . . 40
16 A screenshot of GIB during the bidding. 43
17 A small part of the GIB-database, showing how the Ace-asking bid Blackwood(4 NT) is described. The *numbers* indicates priorities. 45
18 The policies for player a (above) and player b as a function of time. The learning was started at $t = 400$. The first action selected was a3 and b3 ($r = 5$) at about $t = 800$ and then the hesitating combinations a2/3, b2/3 ($r = 0..6$) were selected around time 1000 to 1400 and finally stabilized into a2, b2 ($r = 7$). . . . 47
19 The combined value (Eq 6) and the reinforcement as a function of time. They found their optimal actions a1,b1 ($r = 11$) at $t = 1050$. Before that mainly a3,b3 was selected, resulting in poor reinforcement values. 48

20 The finesse. A two-card situation; when south starts the play with a small, then west plays low or the king and depending on what card west plays north takes the ace or finesses the queen in the hope of west having the king (which is the situation here). The success rate of a finesse is a priori 50%. 51
21 The search-tree for the simple case with just two cards. The numbers at the bottom indicates the number of tricks taken by the North-South. 52
22 A typical hand 57
23 The evaluation of part of the test set. The hands, the learnt bidding, the score and the highest optimal contract (within parentheses) 63
24 Some of the opening bids. The seven columns denotes increasing strength classes (See table 7 at page 70). An asterisk "Alert" denotes a bid which does not carry a natural or – for the opponents – expected meaning, for example an artificial bid only asking partner to describe his hand. 64
25 Some of the responses to the opening bid of 1 spade. The average score (scaled *10 to avoid round-off errors in the integer arithmetic) by selecting different bids are shown to the right for the particular hand with 1 spade, 5 hearts, 2 diamonds and 5 clubs and the 7 different strengths. The top 1s-line means pass - leave the contract at 1s. 68
26 Rebids with a hand opened 1 NT and which got an answer of 3 ♡. When having a strong hand with ♡ support, 4 ♡ is bid but P or 3NT otherwise. The dots indicate hands normally not opened 1 NT. 69
27 Testing different learning rates. East hand reduces its learning rate and West increases. H denotes High, M medium and L low change-rate. The score was the difference to the optimal for a test set (not used as a reward-value). 71
28 The distribution of strength (hcp). Statistics from the 10 million hands with at most five cards in any suit. 75
29 A contour plot of how the strength of E disturbs the strength distribution of W. A high point-count in E reduces (of course) the point-count in W. 76
30 The strength distribution of the combined hands when E has 16 h.c.p. There is a 54% probability they will have at least 25 together. 77
31 With $q_0 = 0$ ($q_0 = +200$ is similar). The smooth curve is the evaluation of the training set and the rugged curve the test set. 200 epochs (right end), each 10 miljon tries, were run, repeated three times, and averaged into these curves of the achieved score. 88
32 Runs with $q_0 = -200$. The smooth curve is for the training set and the rugged curve the test-set. X-axis is epochs and y is the score. 89

33 A high vaule of the exploration and the learning rate improves fast learning. High exploration seems to be most important. q_0 is -200. 90
34 A rerun with a higher learning rate. 91
35 A 5-day CPU run of learning. Lr is 20/1024 , $\varepsilon = 0.050$ and $q_0 = -200$.The maximum training case has a score of 104.89, max test-score is 121.7 and average on the last ten is 94.65. . . . 92
36 The scores are actually decreasing step-wise when the learning rate reduces. 94
37 With a high decay of the learning rate, the performance reduces drastically. 95

List of Tables

1 High card points 57
2 Distributional points 58
3 Number of shapes 59
4 Estimated number of tricks with given strength and trump-length. 59
5 Distributed percentages of resulting number of tricks with a given strength and trump-length. 61
6 The average number of tricks in No trump with a given strength according to Hillyard. 61
7 Classifying the strengths. 70
8 The shapes of the simulated hands. 76
9 The average and standard deviation of the last ten test evaluations, averaged over three runs. q0 is zero. 88
10 The average of the last ten test evaluations, averaged over three runs. q0 is -200. 88
11 Integer round-off error 93

1 Preface

As soon as computers entered, people tried to make use of them when playing games ([Sha50], [Tur50]). In the bridge area there have been some trials but the game is more complex than many other games. A lot of attempts to play bridge have been done. There are also plenty of training "tools" for improving human play and bidding by doing statistical analyses and for practicing. Not until recently (2000) has a computer bridge player [Gin96a]become competitive with humans. Its card-play is almost always technically correct but there is still plenty of room for improvements in the bidding and tactical areas.

The term reinforcement learning(RL) seems to be coined by Minsky in the 1960's. This term also turned up independently in control theory ([Min61] and [WF65] as cited in [Lan97]) but only recently recognized by each other [SBW91]. One of the advantages with applying RL to a game (like bridge) compared to many other areas are the quick evaluation of the results compared to robotics (with slow hardware) and repeatability compared to many psychological or biological experiments.

The term "system" in the title indicates several learners, trying to cooperate. This thesis will be accompanied by the papers and some notes regarding the control part.

2 Abbreviations

ANN	Artificial Neural Net(work)s
DP	Dynamic Programming
EP	Evolutionary Programming
EV	Expected Value
FL	Fuzzy Logic
GA	Genetic Algorithms
GIB	Ginsberg Intelligent Bridge-player
GPI	Generalized Policy Iteration
IBM®	International Business Machines
IL	Independent Learner
JAL	Joint-Action Learner
LTC	Losing Trick Count
MA	Multi Agent
MAL	Multi-Agent Learning
MARL	Multi-Agent Reinforcement Learning
MDP	Markov Decision Process
ML	Machine Learning
NT	No trump
POMDP	Partially Observable MDP
Pr	Probability
PTS	Periodic Team Synchronization
RL	Reinforcement Learning
SBF	Svenska BridgeFörbundet
SOM	Self-Organizing Maps
STURE	Self TUning REgulator
TD	Temporal Difference (learning)
WBF	World Bridge Federation

3 Symbols

Symbol	**Description**
a	action
ε	a small number, exploration rate
γ	forgetting factor
λ	eligibility
π	policy
Q	quality value
r	reinforcement signal
s	state
w	weight in a processing node

4 Introduction and Background

This thesis touches upon several major topics but the red thread[1] in it is Reinforcement Learning which means that the computer (-software) will try to learn from environmental rewards of two kinds *pleasure* or *pain*. It is usually a scalar signal, sometimes even a binary signal like 0 for OK(good) and -1 for failure(bad). This is particularly valuable in those cases when we know whether the result of some actions are good but we do not know how to achieve those results. One of the examples discussed in the literature is the driving of a car. We can measure the lap time and give it as a feedback reward, but we do not know how to make the optimal curve-taking under certain conditions. The second example is in the card-game of bridge; it is fairly easy to see if a final contract is good or bad, but difficult to program the bidding to reach such a (good) contract. A comparison perhaps can be done with a dinner; it is easier to judge if a meal is tasty than to find out how to cook a dinner to make it a good one. The bridge application has also some other interesting features like learning to cooperate with a learning partner and developing some kind of common language, that is having two persons transferring information between them and coming to a correct decision.

The very basic idea of Reinforcement Learning is to have a table (or function) with situations (states) and map the actions to be taken when in those situations. The table stores the value (expected average rewards) for the actual actions tried historically in a certain situation and normally selects the action with the highest expected average reward in the long run when in those situations again. Sometimes (i.e. if a random number is below a certain threshold ε; a Roulette wheel selection) it goes for an *exploratory action* to try to learn more. This makes such systems a little bit dangerous for critical systems but by setting the reward on a very narrow range of values around a stable point or by having a safety controller it may also work for critical systems.

My hope is that these experiments with bridge will be applicable on other domains like network management (information transfer), compression algorithms (finding descriptions of i.e. a face automatically), electronic commerce (how to perform an auction) and robotics (learning to interact). Another area might be language learning; how did we invent a language "in the beginning".

The licentiate thesis is also complemented by the author with a report in the area of robotic control.

4.1 What is learning?

Many definitions of learning exists; some of them are:

"Gain knowledge or skill by study, experience or being taught" [Oxford Advanced Learner's Dictionary]

"Learning is defined as any relatively permanent change in behavior resulting from past experience, and a learning system is characterized by its ability

[1] From the Greek story about finding the Minotaur in the Knossos palace, Crete.

to improve its behavior with time, in some sense towards an ultimate goal" (Narendra&Thathachar[NT74], as quoted by Borga [Bor98]).

Both those definitions are suitable for this thesis.

4.1.1 Feedback from the Teacher/Environment

There are many ways to provide feedback to a learner. These can be classified as follows:

- Supervised: instructive, shows desired response for each stimuli.
- Unsupervised: no feed-back. Learns representations. One example is the self-organizing maps (SOM) by Kohonen.([Koh82] as cited in [Hay99]).
- Reinforcement: the environment provides evaluative, scalar reward or punishment feedback. It is useful when there is no teacher who knows the answer. The learner can solve difficult problems, maybe not yet solved by humans, "Trial and success"[SB98].
- Collaborative filtering: biases constraints from others i.e. if you buy a book maybe you will be told that other people who bought this book also bought these other books. (This is used in the Internet based bookstore www.Amazon.com)

4.1.2 Machine Learning(ML)

This research area involves the mechanisms in machines which make them improve their own behavior. Typically these machines are computers and the mechanisms are programs or algorithms. Other possible interpretations are machines which learn or train people, biological learning processes simulated in a machine, or how people can teach machines to work better. In this thesis we consider the first interpretation.

Machine learning methods are advantageous when preprogramming is not possible as in the following situations:

- unknown environments;
- varying environments;
- it is too complex or much to program.

4.1.3 Operant conditioning

Reinforcement Learning is biologically inspired. One source is operant conditioning: Association between a stimulus and a response is established and this association is strengthened or weakened depending on the outcome of the response . By 1911 Thorndike [Tho11] had formalized this notion into a 'law' of psychology - *the law of effect*. In full it reads:

> "Of several responses made to the same situation those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections to the situation weakened, so that, when it recurs, they will be less likely to occur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond."

4.1.4 Learning by doing

"These processes of learning and doing are inevitably intertwined; we learn as we do and we do as well as we have learned". (Brooks [Bro86] as quoted by Borga, [Bor98]). There is also a difference between knowing and understanding where at least the latter improves by doing.

4.1.5 Reinforcement learning(RL)

A typical task is learning to bicycle. Parents just observe and failure hurts! The pleasure centre in the brain feels comfortable or satisfied when success is achieved. One question that might arise is whether positive (praise) or negative (abuse) feedback is best for learning. Although failures seems to be remembered better, a positive attitude surely improves the fun part of the learning. Modern dog training is nowadays performed in a positive way because it is easier to provide the feedback in direct connection to the action.

4.1.6 Problems during learning

One of the problems in RL is the *Exploration/Exploitation dilemma*: Should I use my knowledge (exploit) or try to get new (explore)? One popular example is the one-armed bandit task: You are playing by inserting coins in a one-armed bandit game machine known to be good but should you try to use another machine, ("explore"), it may be somewhat better but probably is worse. How often and for how long should these exploration phases be? The problem is partly solved in the statistical literature, even for the multiple case; N-armed bandits.

Another problem is *perceptual aliasing*; we do not measure or know everything (hidden information) and thus do not have all information to make the perfect decision, this is also applicable if some of the measurements are distorted or not possible to achieve.

When a task is successfully solved or improved, the *Credit assignment* problem asks who should get the credit and why it succeed.

Which of the actions deserves credit for the result?

There is also a temporal aspect of this; one example: why is the export of goods increasing? Maybe it is because of a decision made a long time ago. Does

Sweden export a lot of music because of the teaching of music during the sixties or because of our modern IT-industry?

4.2 Why bridge?

Blocks world [Win77] has been used as a small-scale model system for decades in the Artificial Intelligence community. Today's problems are of a different kind; cooperation, stochastic and/or non-linear systems, emergent properties etc. My suggestion is to use the international card-game bridge as a new model world with some interesting properties from the real world but still on a reasonably small scale. Some of the things that can be studied using the bridge platform are:

- Collaboration/Cooperation within pairs and teams;
- Competition between pairs, teams and countries;
- Communication - information transfer with limited bandwidth;
- Insecure domain - not all information is visible for all parts;
- Stochastic domains - there is a random factor (how the cards were dealt);
- The role of concentration; mental training;
- Planning (of bidding and play);
- Emergent properties; single suit versus multi-suit plays, end-plays;
- Democratic (descriptive) or Captain (enquiry) based information methods;
- Tactics, Strategy, Risk taking, selecting "the only chance" or safety play "guarding" depending on the situation;
- Constructive versus destructive ways of doing things;
- Psychological aspects - to be unpredictable, when to fool partner or opponent;
- Philosophical aspects - try to find out what partner or opponents think;
- Memory; a powerful, complex, bidding systems versus an easy to remember, understand and use bidding system;
- Getting information - what and when;
- Reducing amount of information to opponents who may take advantage;
- Learning from mistakes and successes.

The game is described in 4.2.1 and can be learnt from [AM94] with the standard bidding in Sweden presented in [Nil78]. Bridge is international with similar rules all over the world and there are at least one million players in the world. Bridge is also an event in the Olympic Games. In my opinion bridge is more complex compared to i.e. Chess which is an open game (no hidden information or stochasticity) and with no partnership to take into account, now performed well by computers using more or less brute-force search methods (IBM Deep Thought II) and databases for openings and end-plays. There are bridge programs performing well and it is also a matter of time before the computer beats most human bridge players. The main reason for this is the technical advantage a computer has (remembering all cards during the play and the bidding system, possibilities of statistical simulations and calculations making it possible to play well and invent conventions "on-the-fly" etc.) compensating for the lack of intuition and other psychological aspects by making fewer mistakes. Today's situation (September 2001) is that Card Play is very good but bidding lacks behind. The reason for this is that entering human judgement as rules in a bidding system database is very boring and tedious. To reduce the size, some compact descriptions are used but are prone to errors (Fig. 17). Methods found during this thesis work (i.e. Chapter 6.5, 8.6 and 8.8) might be used to improve future programs with automatically learned judgement from a huge amount of experiences.

4.2.1 The rules of bridge

Bridge is a card game played with a deck of 52 cards. The deck is composed of 4 suits (spades ♠, hearts ♡, diamonds ♢, clubs ♣). Each suit contains 13 cards with the Ace as the highest value and then the King, Queen, Jack, Ten, 9, ..., 2. The first five (the honors) are often abbreviated to A, K, Q, J, T. The game begins with some shuffling[2] of the deck and then the cards are dealt to the four players, often denoted North, South, East and West (N, S, E, W). N and S play together in a pair, and are called "partners" against E and W. A typical deal is presented in Fig. 1 and the same deal is shown in a screen-shot from GIB in Fig. 16. Before the card play begins, an auction (bidding) takes place. The pair winning the bidding (i.e. making the highest bid) wins the *contract* to make at least a certain number of *tricks*. The player who mentioned the suit first in the pair becomes the *declarer* and the other, his partner, is denoted the *dummy*. The other two players form a team called the *defenders*. The defender on declarer's left side starts the play by *leading*[3] a card. The dummy puts all his/her cards on the table (face up) making it easier to plan the play for all players.

Definition 1 *A card from each player, played clockwise is called a trick. The winner of the trick is the one, having the card with the highest rank in the suit*

[2] Curiosity: research has shown that seven shuffles are normally enough to get almost random result.

[3] Showing and playing by putting it on the table.

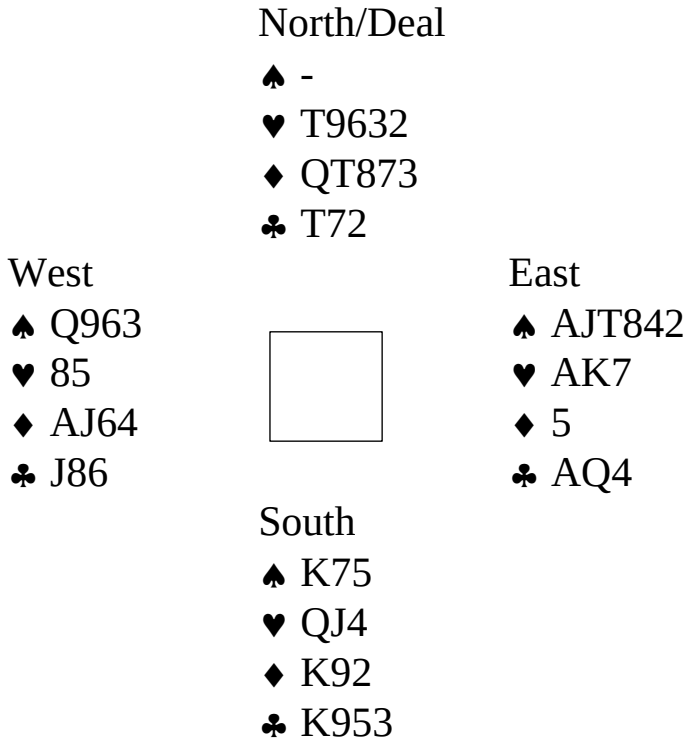


Figure 1: A typical deal in schematic form. It is dealt by player North who then starts the bidding with East next in turn. North is void in (have no) spades but has a weak hand and passes. East has a strong hand (many honors) and normally opens the bidding by giving a bid in his longest suit, 1 ♠ here. South has an average hand but not enough to enter the bidding with a bid other than Pass.

played by the first hand (the leader). In trump contracts the highest card in trump, if played, wins the trick instead.

The declarer decides which card is to be played from dummy. Each player contributes with a card on the table in a clockwise fashion and the winner of the trick plays the first card in the next trick. A trump-suit is looked for in the bidding to prevent opponents from taking a lot of tricks in a long suit of theirs. If no common trump suit is found, a game without trumps can be played; No Trump (NT), which is ranked higher in the bidding than the suits which are ranked ♣ (lowest), ♢, ♡ to ♠ (highest). Players have to *follow suit* except when they do not have any cards (left) in this suit - in that case they can discard any card or try to win the trick by playing trumps; *ruffing*.

During these two phases, the bidding and the play, there is cooperation in pairs playing against each other with four players two pairs at each table. The pairs communicate via bidding where the bids are limited to a few (15) "words", but many sequences "sentences/dialogues" are possible. Opponents are kept informed[4] about the *meaning* (semantics) of the bid or sequence. This can be seen as a mini-language with a few words like 4, 1, Spades(♠), Double, Pass etc. and these words can be combined via simple grammatical rules into short "sentences" like 3 Spades. The sentences are put in a sequence "conversation" called the bidding. It can look (within a pair) like E:1 ♠ - W:2 ♠, E:4 ♠ - W:Pass with semantic meanings like "I have a hand with spades as the longest suit and above average number of Aces, Kings and Queens" - "I have support for your spades but not such a good hand", "I have some extra strength" - "I have nothing to add". The first non-pass bid is denoted *opening*. A full bidding scheme is presented in Fig. 2.The normal goal of the bidding is to agree upon a

N	E	S	W
P	1♠	P	2♠
P	4♠	P	P
P			

Figure 2: The scheme of the bidding with the hands in Fig 1. Three passes end the auction.

suit with at least eight cards in common and at an appropriate level. Sometimes a suit with 7 cards in common will or have to do or a NT contract without a trump suit might be the best selection.

Three Passes finish the bidding and the winning pair handles the final contract (4 spades) and the card play, i.e. taking tricks, can begin. After the play,

[4]If they ask. It is often good to avoid questions when the opponents seems to have forgot their own agreements and ask at the end of the, hopefully, catastrophic bidding.

the pairs are given points according to the *result*. In tournament play the points are compared to the other pairs playing a *duplicate* version of the same deal, *a board*, at other tables almost compensating the factor of luck with "good cards" because all tables have to do their best with the same, given, cards as stored in the board.

4.2.2 General

Bridge is a model world with insecurity and stochastic components. A player's concentration varies during the game(s). A reasonable program is of great value for practising/training and during development of new methods for bidding.

Bidding The bidding is a competition between the pairs to reach the optimal contract. The meaning of the bids are collected in a document called "The Bidding System" and the contents vary a lot from pair to pair. Some of the bids are natural which means that they show the suit mentioned and others are conventional which means that they either ask partner for information or show something particular not connected to the natural meaning of the bid. The popular Stayman convention; 2 ♣ (clubs) asks for the number of hearts and spades in partner's hand after an 1 NT[5]-opening. There is a huge set of popular conventions to use and these may be combined rather freely making the bidding systems quite different.

Definition 2 *A non-pass bid when the other pair already has given a non-pass bid is called an intervention.*

Definition 3 *To deliberately make a bid, normally at a high level, with insufficient values is called "to make a preemptive bid" (or "to preempt"). Taking the risk of paying a penalty for not making the contract can be outweighed by reducing opponents' possibilities of exchanging information and thus forcing them to guess.*

Definition 4 *To make a bid when knowing that the contract probably will go down, but probably at a lesser expense than letting opponents' contract, be played and made is called to (make a) sacrifice.*

In our work we have studied unopposed bidding which means the pair is free to bid on their own without disturbing interventions from the opponents. There is no need for preemptive bids and sacrificing or to keep the importance of reducing the amount of information transferred to the opponents in mind. By having these limitations we are able to learn better in shorter time and requiring less memory but this of course limits the practical usefulness. Maybe we will have the opportunity to make the computer learn bidding systems with all four players later on.

[5] NT means No Trumps; suggests a play without any special suit as trump. Then it is not possible to ruff. In Sweden we call it Sang from the French sans atout.

4.2.3 Bidding system

A collection of rules regarding what to bid with a certain hand is called a bidding system. First you have to define the main routes to take, like the opening bids, general approach and, secondly, you may add and combine or invent *conventions*[6], which may be conventional[7][8] or natural. A partner knowing these rules may now make deductions of your bids like, "aha! he shows at least 5 spades by his bid and I have 3 of them; we have found a suit with at least 8 cards between us, that will make an adequate trump suit for us! Which bid should I select to inform him about this happy news and how should I show my strength to make him select the correct level (number of tricks to take)?"

Popular systems are the British Acol (very natural), The Strong club (opening 1 club shows a strong hand), the French Canapé (opening second best suit), the Swedish 2-way 1♣ (1 club shows strong or weak and balanced) and the Standard American (Yellow Card)[9] five card majors[10] (1 heart/1 spade shows at least five cards in the bid suit). Which the "best" bidding system is debatable, even how to define it, but the performance of systems may be compared when used in tournaments or in simulation. Of course the selection of system depends on the expected strength of the opponents; against weaker opponents it is sometimes better to select a system which forces them to use judgement, which they maybe lack. The allowed system complexity, at least the specification of openings, is limited in most short or low and ordinary level tournaments. The actual situation has also different demands; if leading by a safe margin you might go for safety or go for gambling if being behind, then there is nothing to lose. Bridge is a high-variance game which makes it possible for beginners to win a short tournament against better/more experienced opponents. This makes it highly interesting but also frustrating (if you are a good player). A real beginner can not win against experienced players over, say, 12+ boards.

Play Analyses of the play can be made in different ways; find the best play
- · on the actual board,
- · on average, given some information,
- · without information about the bidding or players,
- · taking psychological aspects into consideration
- · by "Rule of Thumb"

On the actual board In newspapers there are often problems of the kind "is it possible to make 4♠ against best play from the defenders". These

[6] Agreements about the meanings of the bids; such agreements have to be made known(alerted) to the opponents and they may ask about the meaning if they want.

[7] These are bids with a special, sometimes artificial, meaning, and often not promising length in the bidden suit.

[8] It is often good to avoid questions about the meaning when the opponents seems to have forgot their own agreements and ask at the end of the, hopefully, catastrophic bidding.

[9] Curiosity: Bill Gates played SAYC at the Nationals.[FL00]

[10] majors are hearts and spades. Clubs and diamonds are called the minors.

types of problems are called *Double Dummy,* and all four hands can be seen. A computerized example is given in Section 7.2.1. Such problems can often be solved within seconds with the help of bridge software, e.g. GIB.

On average During normal play the opponents' cards are not visible and a bridge player makes clever assumptions about reasonable distributions and honor positions. One way of solving this way of play will be to distribute opponents cards according to the background knowledge of bidding, player style etc. if given, and play a certain card and repeat this selection on reasonable variations of the distribution of opponents cards.

The result from this calculation will be the normal way of playing a hand as it is done at the table (a priori). Sometimes (for tactical reasons), a different way of play is used to increase the possibility of gaining a lot of points if the board happens to be unusually distributed. Safety play is achieved by choosing a way of play to guarantee a number of tricks. Sometimes the only chance is to assume (hope for) a special distribution of opponents cards or honors to make the contract. These choices are made upon beliefs of what the players do at other tables in a tournament and depending on the current situation.

The result from the analysis should be multi-dimensional; by playing a certain card there is a number (percentage) of deals resulting not only in t tricks but also in $t-1,\ t+1$ tricks etc. making it possible to choose a way of play from a tactical standpoint. See tables on pages 59 and 60.

Psychological aspects Logically there seems to be no difference in playing the Queen from QJT as opposed to the Ten but from the opponents point of view, the Queen might have been played from Q8 or something similar and it would be essential to incite opponents to cover with the King. It is a rule of thumb among bridge players to play the highest card to get coverage and play a low (the Ten) to make opponents play low. Sometimes this might be a clue as to who has the King and opens for the possibility to drop a singleton King over the Ace.

To simulate such situations we have to implement a bridge player program making hypotheses about opponents beliefs and feelings and also about what type or amount of information is transferred by playing a particular card.

Rule of Thumb By counting the number of trumps and high-cards, an estimate of the expected number of tricks to be taken can be made. This is the approach used in our first learning method because it is a quick table (at page 59) look-up instead of analyzing the play of individual cards. A more realistic evaluation function would generate stochastic results with different number of tricks (page 60) according to the hands. In our second stage trial (page 81), we used a large database with actual results.

Risk taking Some players or situations demand for higher risks, an example is a preemptive bid with a holding of cards that are either better or worse

than expected. This can force the opponents into bad contracts but also our own partnership into bad results. If you (think you) are a better pair than the opponents you should probably avoid taking high risks and instead wait for the opponents to make mistakes.

Scientific or non-scientific bidding methods If you have a fairly complicated "scientific" system, you will be able to find out partners holding and select the most probable optimal contract. If you have a less complicated system you are left in the dark, receiving correspondingly less information about your partner's cards, and you will have to guess more/better to arrive in the best contract. On the other hand opponents *also* have to guess. You may be lucky, finding a better contract than the "scientific" bidders, simply lucky with the actual deal/distribution of the cards or even having sleepy opponents allowing your sub-optimum contract to make. It is also possible that the "scientific" bidders forget their agreements or spend all their "brain-power" on the bidding and fail in the play. Anyhow it is not obvious which line to take in the bidding system design.

4.2.4 Scoring

Modern, tournament, scoring in bridge is done the following way; the suits are put into two groups, minors (♣ and ♢) and majors (♡ and ♠). For each trick above 6 taken in a minor-suit contract you earn 20, for each in a major it will be 30. In notrump (NT) there will be 40 for the first and 30 for the next one(s). You are supposed to take at least the number of tricks determined/decided in the bidding (i.e. the contract), otherwise the opponents get 50 for each undertrick. If you have bid to a contract where the sum of the points is at least 100 (i.e. 5♣, 4♠, 3 NT or higher) you have contracted for *Game* and if you make it you will receive 300 extra as a bonus, lower contracts give a part-score bonus of 50. Some examples: 4 ♠ bid and made gives $4 * 30 + 300 = 420$ but 3 ♠ with an overtrick gives only $4 * 30 + 50 = 170, 4$ ♠ down one gives 50 to the opponents but 3 ♠ exactly made gives $3 * 30 + 50 = 140$. In real bridge there is actually a variation added; vulnerability; when vulnerable the game bonus will be 500 and penalties are scored -100 for each undertrick. This makes for more tactics if one pair is vulnerable (have to be cautious) and the other not (not so dangerous to take risks). Actually there is another factor, dimension, abbreviated as X, the double, which increases the stakes: both penalties (if you fail to make the contract) and points (if you do make it) become higher. Then there is the Redouble (XX),...and the slams; Small Slam (6 of any denomination) and Grand Slam, (7 of anything).

There are several methods of "using" the actual score;

a) Rubber bridge. The goal is simply to score as many points as possible. Since the deck has no memory[11], it is often good to play for safety. Sometimes

[11] The chance of getting good cards is still the same even if you already have had good cards for a while.

people play for money at this kind of game, which is most usual in homes or at smaller clubs (in particular in the UK and its former colonies).

b) Pairs tournaments (Duplicate). The game is divided into "rounds" of a few (e.g. 2) boards. After each round pairs move according to a scheme to play another round against other opponents with other boards. The calculation of the results is based on *ranking* of the scores in a descending order. The size of the difference between two is not important. If a result for N-S is +110, then it doesn't matter if the next better result is +120,+600 or even +1400, it just happens to be the next better result! This method incites trying to outdo your opponents by small means as well as by large ones. In this type it is important to find a contract giving a high number of points per trick (the minors, ♣ and ♢, are normally avoided).

Sometimes the boards are duplicated so all tables play an identical board simultaneously "Barometer tournament" and the results will be available some minutes after a board is played. Typically 24 boards are played in an evening. Pairs tournaments is popular but luck plays a part here too, not so much in terms of getting good or bad cards, but more depending on gambling-style (i.e. in the hope of scoring 10 points more by playing in a less safe notrump contract rather than in a suit).

N			NS	EW	Contract		NS	EW
W	7	E	5	6	4♠	N +420	0	2
S			1	2	3NT	N +430	2	0

Figure 3: A table with the pairs tournament protocol (paper) from the 7th deal. The resulting topscore points are 2 to pair 6 and 1.

An example (Fig. 3): Pair 5 (N-S) meets pair 6 (E-W) on board 7 and N-S plays 4 ♠, just made, giving a score of 420 to pair 5. The same board is also played by pair 1 (N-S) and 2 (E-W) but here the pair one bids a hazardous 3NT and also makes 10 tricks after an unlucky lead by pair 2, scoring $40 + 2 * 30 + 30 + 300 = 430$. The pairs-calculation gives 2 topscore-points to pair 6 and 1 and zero to pair 5 and 2.

The prizes in Sweden are often of little economic value but the pride factor can be high.

c) Teams. Four players are playing against another 4-player team with two players from each team at each table, sitting N-S at one table and E-W at the other. After some boards have been played they are exchanged for other boards from the second table. The results can be compared with the other team's performance on the same boards immediately afterwards. This type of play is normally preferred by the elite. A more or less logarithmic scale (Fig.

4), International Match-Points (IMP) is used and, in contrast to the scoring at Pairs, here the size of the difference between two results are of paramount importance. This results in, i.e., playing to safely make one's contracts instead of taking a risk to make an overtrick (since the overtrick is of a, point-wise, very small value compared to the Game bonus) The IMPs are summed and the results presented in Victory points (VP), which vary with the number of boards played (Fig. 5). If the previous board (7, Fig. 3) was played in a teams tournament the difference between the teams would be $430 - 420 = 10$ and a difference of just 10 gives 0 imp, no point to any team but if one team just bid 3♠ the result will be $430 - 170 = 260$; 6 IMPs to the other team. If there are 3 boards like that and the rest even in a 24 board match the total IMPs result would have been $3 * 6 = 18$ and $18 - 12$ in VP, a tiny win. Note that if the pair above did bid 3♠, the resulting topscore (0) would have been the same. The prize is often honour and, perhaps, fame. Exception is tournaments like the Cavendish where price-money of US$ million+ is played for...

Points	Imp	Points	Imp
0 - 10	0	750 - 890	13
20 - 40	1	900 - 1090	14
50 - 80	2	1100 - 1290	15
90 - 120	3	1300 - 1490	16
130 - 160	4	1500 - 1740	17
170 - 210	5	1750 - 1990	18
220 - 260	6	2000 - 2240	19
270 - 310	7	2250 - 2490	20
320 - 360	8	2500 - 2990	21
370 - 420	9	3000 - 3490	22
430 - 490	10	3500 - 3990	23
500 - 590	11	4000 - more	24
600 - 740	12		

Figure 4: The International Match-Point scale. It is often used in Teams tournaments to improve the importance of high-level contracts.

More about bridge is presented in section 8.4.

4.2.5 Available software

There are some computer programs around (reviewed in [Fra98]) able to play bridge but most only at a low (beginners) level. This will obviously change in the near future. One candidate is AI-researcher professor Matthew L. Ginsberg's GIB which he thinks will be better than humans by the year 2003[12]. The

[12] By now we know it is not true.

THE WBF IMP-VP CONVERSION SCALE

VPs	Number of boards										
	8	*12*	*14*	*16*	*20*	*24*	*28*	*32*	*36*	*40*	*48*
15-15	0-1	0-1	0-2	0-2	0-2	0-3	0-3	0-3	0-3	0-3	0-4
16-14	2-5	2-6	3-7	3-7	3-8	4-9	4-10	4-10	4-11	4-11	5-12
17-13	6-8	7-9	8-10	8-11	9-12	10-14	11-15	11-16	12-17	12-18	13-20
18-12	9-11	10-12	11-14	12-15	13-16	15-19	16-20	17-22	18-23	19-25	21-28
19-11	12-14	13-16	15-18	16-19	17-21	20-24	21-25	23-28	24-29	26-32	29-36
20-10	15-17	17-20	19-22	20-23	22-26	25-29	26-31	29-34	30-36	33-39	37-44
21-9	18-20	21-24	23-26	24-27	27-31	30-34	32-37	35-40	37-43	40-46	45-52
22-8	21-23	25-28	27-30	28-31	32-36	35-39	38-43	41-46	44-50	47-53	53-60
23-7	24-26	29-32	31-34	32-36	37-41	40-45	44-49	47-52	51-57	54-60	61-68
24-6	27-29	33-36	35-38	37-41	42-47	46-51	50-55	53-58	58-64	61-68	69-76
25-5	30-33	37-40	39-43	42-46	48-53	52-57	56-61	59-65	65-71	69-76	77-84
25-4	34-37	41-45	44-48	47-52	54-59	58-64	62-68	66-73	72-79	77-84	85-93
25-3	38-41	46-50	49-54	53-58	60-65	65-71	69-76	74-82	80-88	85-93	94-102
25-2	42-45	51-55	55-60	59-64	66-72	72-79	77-85	83-91	89-97	94-102	103-112
25-1	46-50	56-61	61-66	65-71	73-79	80-87	86-94	92-100	98-106	103-112	113-123
25-0	51+	62+	67+	72+	80+	88+	95+	101+	107+	113+	124+

Figure 5: The result of a match. There is usually 30 VP to distribute between the two teams. A typical match is 24 boards and if the difference is 25 IMPs the match result will be $20 - 10$ VP.

software was reviewed by the author of this thesis[Pro99]. Standard GIB uses Borel simulation.

Algorithm 5 *([Gin99], Borel simulation) To select a bid from a candidate set B, given a database Z that suggest bids in various situations:*

1. Construct a set D of deals consistent with the bidding thus far.

2. For each bid $b \in B$ and each deal $d \in D$, use the database Z to project how the auction will continue if the bid b is made. (If no bid is suggested by the database, the player in question is assumed to pass.) Compute the Double Dummy result of the eventual contract, denoting it $s(b, d)$.

3. Return that b for which $\sum_d s(b, d)$ is maximal.

The improved GIB; GIBson, was released in August 2000 and it is able to make declarer play on a very high level including single dummy solutions. This improvement also makes the bidding better because it simulates the play to find the best or most probable contract. There are dozens of other programs for playing, generating deals[And99] etc. Ian Frank has made an end-play analyzer/planner FINESSE which seems to work well but too slowly for practical use (written in interpreted PROLOG) and he has also written an excellent doctoral's thesis "Search and Planning Under Incomplete Information" [Fra98]. PYTHON [SN90] as cited in [BGP93] is an end-play analyzer for finding the best play but not on how to reach these positions.

The Bridge Champion Zia Mahmood has offered one million pounds to the designer of a computer system capable of defeating him but withdrew his offer after the last match against GIB which was a narrow win for him. Regarding this subject Brent Manley [Man94] wrote: "It is generally accepted that writing a computer bridge program that plays bridge well is nearly impossible" (cited from [Fra98]). Christina Erskine[Ers92] is more optimistic; "One day, someone will program bridge to play more like an expert (and a very long, fascinating exercise in artificial intelligence it will be, too) and put all those chess programs in their place" (also from [Fra98]).

4.3 The goal

The goal of my current research work is neither good bridge playing or bidding software, nor to create a development system for the game but to see how the learning processes might work on this, for me, familiar subject. It would still be interesting to find an ideal bidding system. By comparing different learning algorithms, parameters, assumptions and representations I hope to find one method, outperforming the other's and perhaps be able to create a bidding system comparable with manually constructed ones.

This thesis also provides an overview of machine learning methods and their advantages and drawbacks helping the reader learn and select the best method on a given problem and also be aware of potential problems. I am also glad to have the opportunity to teach computer scientists some bridge (maybe creating a new platform for algorithm development) and teach some bridge players

computer science. I believe the results will be applicable to lots of problems in the Game Theory area including economical applications like auctions, price setting and other strategic decisions but also in other areas like cooperative and competitive learning systems in widespread areas. The Nobel Prize in economy 2001 (related to asymmetric information) is actually quite close to this area.

5 Definitions, Methods & Theories

Here is a summary of methods used in the applications. This can also be seen as an overview of machine learning techniques.

5.1 Reinforcement Learning

There is an excellent book about the subject [SB98] that is recommended reading. Here just a few topics used in the thesis will be introduced briefly. We did use Q-learning, presented in section 5.1.12.

5.1.1 Introduction

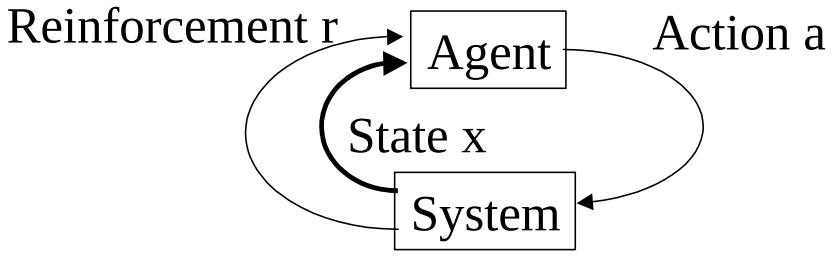


Figure 6: The reinforcement learning cycle. The Agent senses the state and a reinforcement. It tries to estimate whether this state is good.

Reinforcement learning is the method in which there is only a weak scalar feedback signal, the reinforcement (or reward), success or failure in the binary case, but it contains no direction or time information. It is up to the learner to try to find out the reason for the failure and how to avoid it (Fig. 6). Finding temporal information is also an important part because the failure might be a result of an action taken a long time ago.

Definition 6 *An agent[Fer99] is a physical or virtual entity*
a) which is capable of acting in an environment
b) which can communicate directly with other agents,
c) which is driven by a set of tendencies (in the form of individual objectives or of a satisfaction/survival function which it tries to optimize),
d) which possesses resources of its own
e) which is capable of perceiving its environment (but to a limited extent),
f) which has only a partial representation of this environment (and perhaps none at all),
g) which possesses skills and can offer services,
h) which may be able to reproduce itself,

i) whose behavior tends towards satisfying its objectives, taking account of the resources and skills available to it and depending on its perception, its representations and the communications it receives.

Agents are capable of acting and not just reasoning and are more or less autonomous.
Reinforcement learning can be used on systems where there are goals to fulfill like keeping a formula-1 car on the road or cutting a tree log within a length window. We are not mainly interested in keeping the car in the middle of the road as this would be suboptimal, but in allowing the controller to find the optimal path through a curve. This obviously improves the maximum possible speed. Similarly the control of the feeding of a tree log is not a matter of following a controlled stop-ramp, the goal is to cut at the right place and nothing else! Maybe this type of control can be used for robotics where the goal might be to find and grasp an object and not to follow a certain path.

5.1.2 Reward

The reward defines the goal in the reinforcement learning problem. It maps each state into a single number, indicating the desirability to be in that state. The learner's objective is to maximize the total reward in the long run. Rewards r_t are normally given directly by the environment at time t.

5.1.3 Discounted return

To avoid infinite sums of rewards and also to control the behavior of the agent, future rewards are scaled down with a parameter γ , called the discount rate, typically in the range 0.8 (myopic) to 1. This makes a reward r_t today more worth than a reward r_{t+1} tomorrow. The agent selects actions a_t to maximize the expected discounted return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{1}$$

5.1.4 Markov property

If all relevant information is stored in the state, the state signal is said to be Markov. Typically a checkers position is Markov because it summarizes all previous moves into a position. A learning task that satisfies the Markov property is called a Markov Decision Process (MDP) and, if the state and action spaces are finite, a finite MDP. Then, given any states s and actions a, the probability P of each possible next state s' at time $t + 1$ is

$$P^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$$

and the expected value of the next reward R is

$$R^a_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}$$

5.1.5 Policy

A policy defines the agent's way of behaving, a mapping from perceived states s to actions a to be taken when in those states. Policies are often denoted with the letter π; $\pi(s, a)$. Policies may be stochastic, meaning that one out of several different actions can be selected in the same state with some probabilities. There is at least one policy that is better than or equal to all other policies. This is an optimal policy π^*. Part of the policies is the selection of exploratory actions. This can be done by selecting such actions with a small fraction ϵ and methods are for instance ϵ-greedy or soft-max-selections (Boltzmann, see equation 8 at page 72). $\pi(s, a)$ is the probability of selecting action a in state s. If the policy-space is large or open, π can be implemented as a function instead of being stored in a table.

5.1.6 Value function

The Value $V(s)$ of a state is the prediction of the sum of future rewards by being in that state. The values are estimated and reestimated from sequences of observations. It is important to note that the value of a state is dependent on the policy (See 5.1.5) i.e. the border squares in Fig. 7 have a low value (are dangerous to be in, easy to fall of the edge) in the random policy (select action by random) but higher in a better policy. An example from car driving; suppose you are going with a very old (or young) man, then it is probably much more dangerous to be close to the border of the road than if you were going with an experienced race driver.

The state-value function V for MDPs in a state s can be defined as

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\},$$

where R_t is the discounted return (Eq 1) and s_t is the state at time t. The action-value function Q (for quality) separates the return for each action a ,

$$Q^{\pi}(s, a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\},$$

making it easier to maintain action selections by keeping the expected returns separated for each action.

5.1.7 Episodic and Continuing tasks

If the agent-environment interaction naturally breaks down into a sequence of separate episodes it is called an *episodic task* otherwise an *continuing task*.

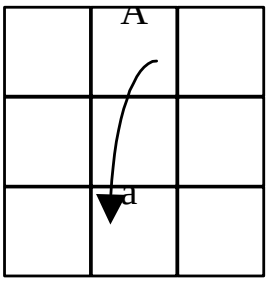


Figure 7: A simple grid-world. Four actions are possible in each cell; north, south, east and west. Actions off the grid leaves the location unchanged but rewards -1. A move into A locates in a and rewards +5. There are 9 states.

5.1.8 Temporal difference (TD)

The value of a later state s_{t+1} may influence the value in the earlier (whence temporal) state s_t with the following update rule

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{2}$$

where α is a learning-rate parameter (typically in the range 0.01..0.1). This equation (2) is called $TD(0)$. It can be modified into $TD(\lambda)$ (Section 5.1.12).

5.1.9 Bellman equation

Value functions used in RL and dynamic programming satisfy particular recursive relationships. For any policy π and state s the following applies:

$$\begin{aligned}
V^\pi(s) &= E_\pi\{R_t \mid s_t = s\} && (3)\\
&= E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s\right\}\\
&= E_\pi\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_t = s\right\}\\
&= \sum_a \pi(s,a)\sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_{t+1} = s'\right\}\right]\\
&= \sum_a \pi(s,a)\sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^\pi(s')] && (4)
\end{aligned}$$

The equation is called the *Bellman optimally equation* for V^π[Bel57] and expresses a relationship between the value of a state and the values of its successor states. The equation averages over all, weighted by probability of occurring, possible resulting states and their discounted value of the reward.

Definition 7 *The optimal state-value function is defined as*

$$V^*(s) = \max_{\pi} V^{\pi}(s), \ \ forall \ s \in S. \tag{5}$$

There is at least one policy that is better than or equal to all other policies.

5.1.10 Dynamic Programming(DP)

Given a perfect model of the environment as a MDP, the optimal policies can be computed by turning Bellman equations into update rules. Continuous states and actions normally have to be quantized into finite spaces.

5.1.11 Iterative solution

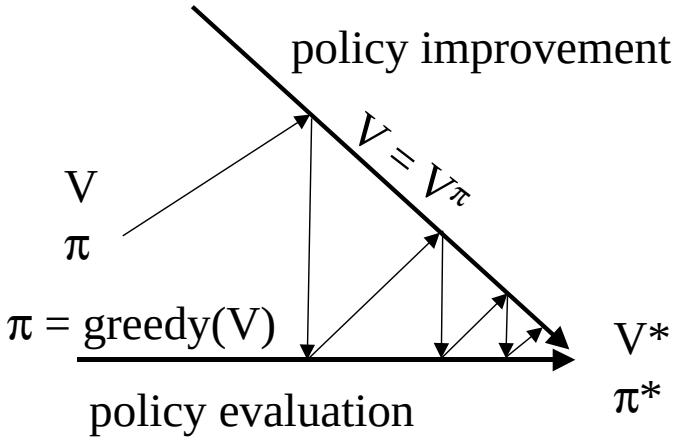


Figure 8: Co-convergence into the optimal values by interacting policy improvement with policy evaluation. The processes converge because the (thick) lines are not orthogonal.

The Bellman equation can be used as *iteration steps* to find optimal policies. This will reduce the calculation by solving a set of linear equations by iteration. Suppose we start with the random policy (selecting an action by random) and calculate the values of each state. Now we can select a better policy by selecting actions leading into states with higher values and with this new policy the values of the states are improved. By combining policy evaluation and policy improvement (Fig. 8) we invent generalized policy iteration (GPI). The algorithm is guaranteed to find the optimal policy in polynomial time ([SB98], p107) and in practice within a few simple iterations instead of solving huge sets of equations. This is also called Asynchronous Dynamic Programming(ADP).

Since many problems have many states, maybe 10^{20}, one has to settle for approximate solutions by using functional representations instead of tables but in this case an iterative method still works but there is no "set of linear equations".

Definition 8 *An estimation method involving averaging over random samples is called a Monte-Carlo method.*

5.1.12 Q-learning

One of the breakthroughs in RL was the development of an TD control algorithm known as Q-learning[WD92]. Define Q^π as the action-value function for policy π. Q separates the averages for each action taken in a state and is a Monte Carlo (MC) method because it involves averaging over random samples of actual returns. $Q(s, a)$ may be a parameterized function if the state and/or action space is large. The advantage with Q-learning is to learn the optimal policy without actually following it, so called off-policy learning.

Algorithm 9 *Q-learning.*
Initialize Q(s,a) arbitrarily
Repeat (for each episode):
Initialize s
Repeat (for each step of episode):
Choose a from s using policy derived from Q (e.g. ϵ-greedy)
Take action a, observe r, s'
$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s.a)]$
$s \leftarrow s'$
until s is terminal

The algorithm can be described as "keep track of the average result from each state with the different actions separated. When arriving in a new state, update the previous state with the highest value of the possible actions although this action may not be selected due to i.e. exploratory moves". There is a corresponding on-policy learning method, Sarsa, that learns the best behavior given the exploratory moves. Simply remove the Max operation in the equation above to obtain Sarsa.

Eligibility traces The learning may speed up by extending the methods mentioned above into multiple steps. A backward weight λ (trace decay) is used to propagate rewards backward to visited states during the episode.

5.1.13 Hidden states

If the agent does not perceive exactly what state it is in we have a partially observable Markov decision process (POMDP). These processes can often be solved by using groups of states and jumping between those groups.

5.1.14 Applications with Reinforcement Learning

Backgammon Gerald Tesauro implemented a computerized backgammon player[Tes94]. It became a World Champion and invented new unintuitional moves now played by experts. It was based on an ANN (see next subsection) to

evaluate the value of a state. The state space is around 10^{400} but most states are not so common in normal play. After about 3 million games (about a week of CPU-time) with itself it performs excellently but it has only learnt in those areas in the state space occurring during normal play.

Some Control problems Some of the classical ones are *mountain car*; how to run an underpowered car uphill, *acrobot (acrobatic robot)*; a stick-person trying to swing up, *inverted pendulum*; balancing a vertical pole via a moveable cart and avoiding running out of the track.

Formula-1 *car race* game; trying to optimize curve taking in a simulator environment [Woy97]. The first attempt was carried out using a reinforcement of 0 for staying on the road and -1 if outside; a GARIC based controller (see Sec. 6.1) calculating and simulating for four hours resulted in a controller just telling the car to stand still!

Elevator control; [CB96] Minimizing the waiting time for people calling for the elevator by making the elevator learn how to behave. The RL algorithm performs better than the most popular algorithms.

5.1.15 Conclusions

Reinforcement Learning is a powerful learning method. Some of the problems with the method are risk/cost of failure and slow learning because the situations have to appear many times.

5.2 Artificial Neural Nets (ANN)

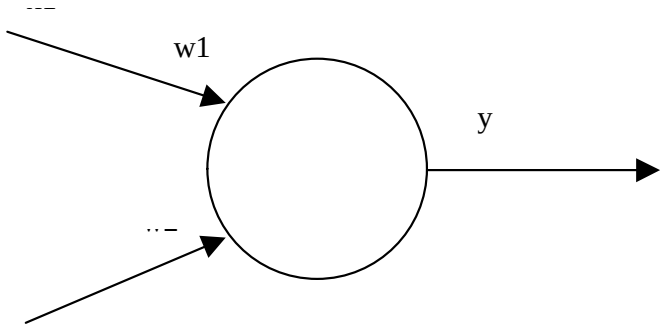


Figure 9: A processing node (element). The output y is a function of the sum $(w_1 \cdot x_1 + w_2 \cdot x_2)$. A sigmoidal function is often used. Learning is done by adjusting the weigths w_i.

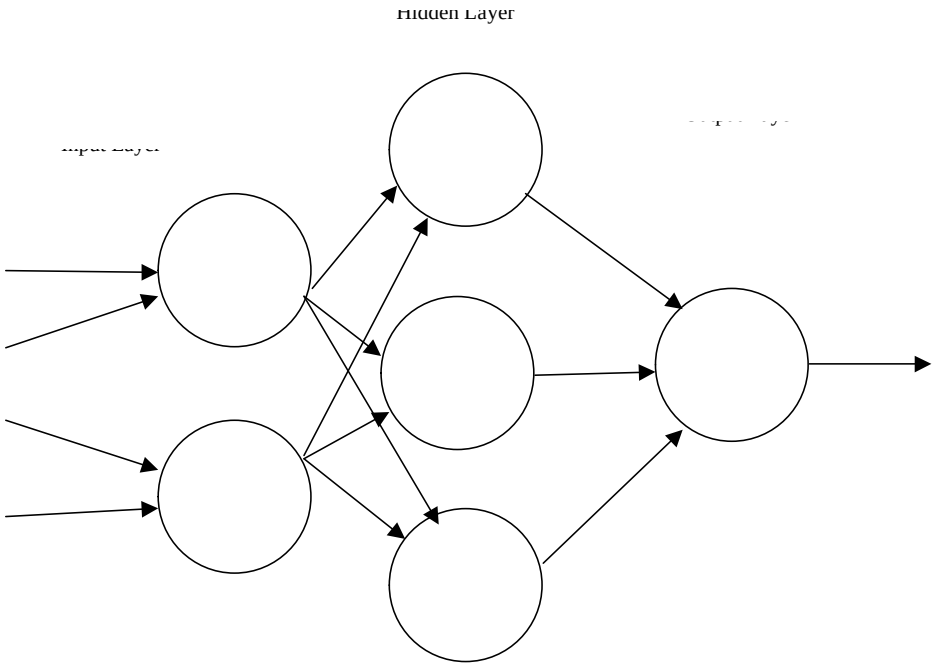


Figure 10: A simple artificial neural net with 4 inputs and one output. There are 3 processing nodes in the hidden layer.

By connecting nodes (Fig. 10) with weighted inputs and detect the output, the weights (Fig. 9) can be adjusted to the desired outputs, "be taught". Normally these nodes ("neurons") are connected in layers. They can approximate many functions (universal approximators) and be used to interpolate between states or actions to reduce the memory requirements by offering a parameterized function instead of large tables. The most popular learning rule is error back-propagation. Sigmoidal functions are often used as output functions and sometimes radial basis functions are used to simulate nearest neighbor behavior. One new idea in the area is pulsed neural networks [ME99] where the input and output are *spike events* and the timing of the individual spikes are important. Suppose we have a firing rate of 10 spikes/sec. In a normal ANN this is a number '10'. In a pulsed net the distribution of the spikes during this second is important; do they come evenly distributed or clustered 'burst-wise'? This is more like the human nervous system where time-spatial relations have to be represented when, for example, grasping and moving an object and trying to estimate the friction forces to be able to keep the object without dropping or destroying it.

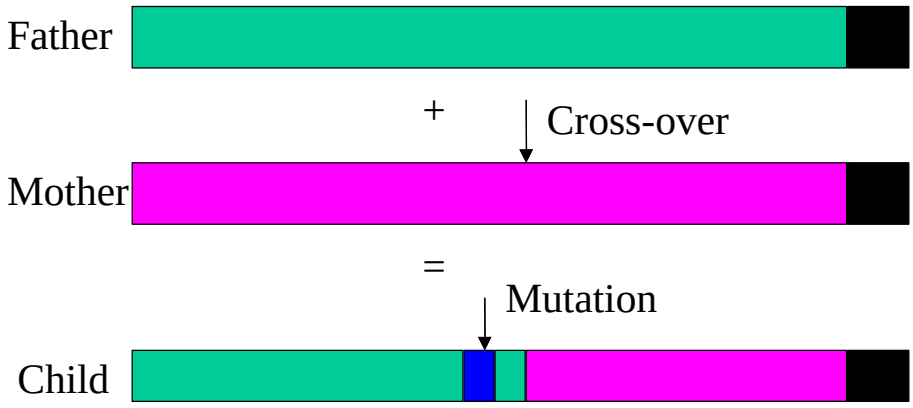


Figure 11: The genetic process. In this case the child gets the first part from its father and the second from its mother. There is also a random change (mutation) close to the middle. The black boxes at the end supplies storage for the fitness value for this particular individual.

5.3 Genetic Algorithms (GA)

By encoding a strategy into a string (vector) "chromosome" of genes and evaluate the fitness function for this string in many individuals (phenotypes), a good strategy can be found. Offsprings are generated by combining parts (Fig. 11) of the strings from parents with high fitness resulting in offspring that may or may not be better (having higher fitness). The parents are then often removed from the population. We try to maximize the fitness, "survival of the fittest" in order to reach a near-optimal solution.

The main advantage with GA is the easy implementation of many applications including ours. The disadvantage is often slow convergence into near optimum. One of the books about GA:s is [Mic99] and the algorithms are compared with traditional methods in [MF00].

Algorithm 10 *Evolution program(EP)[Mic99].*
$t := 0$
initialize P(t)
evaluate P(t)
while (not termination-condition) do
    $t := t + 1$
    select P(t) from P(t − 1)
    alter P(t)
    evaluate P(t)
end

P denotes population (a vector with fairly many individuals) in the algorithm above. These algorithms (GA is a special case of EP) use recombination and

mutation to speed up the search and are normally better than pure random search. It is important to have a large set of individuals and keep the generations mixed, genetic diversity, to avoid local (sub)optima. This method, although not so effective, does not require derivatives, which makes it especially suitable for discrete systems like the bridge-bidding; multiple bidding systems can be generated and evaluated and, as the next generation, parts of good systems are combined into better (and worse) systems. Other suitable methods in integer optimization are branch-and-bound and graph-theoretical methods.

The solutions to a genetic algorithm are not guaranteed to be optimal and some termination condition are always used to stop evaluating individuals when a certain level of fitness is reached or some time has elapsed. Michalewicz describes it this way[MF00]: "A method to get an approximate solution to a problem instead of having an exact solution to an approximated problem". This is because many real problems are not solvable by conventional algorithms without making some simplifications or assumptions. Genetic algorithms and ANNs are easy to implement and naturally parallelize into several processors to improve the results achieved within a certain time frame. The encoding and applied operators (crossover, mutation) are of course of great importance[Mic99]. Premature convergence is one of the typical problems, meaning the whole population has more or less the same genes. This can be reduced by methods like incest prevention, island models, religions or crowding models. It is also important to note that nearby genes have a greater chance to join together into the next generation due to the crossover selection. Roulette wheel selections of partners with probability of selection proportional to the fitness requires some care; the fitness has to be positive and the range is important because otherwise the selection will be random or the opposite; the same individual will be selected (almost) all the times. For numerical optimizations with floating point numbers there are specialized operators like non-uniform mutation and arithmetical crossover instead of simple binary operators. Some[Sch99] have even used machine instructions to create random programs performing tasks like inventing curiosity or enjoying themselves. Nordin has made robots learn (how) to walk and even develop 3D vision with EP[NW03].

Example 11 *Car driving. Assume a vector "chromosome" with actions and a length as the fitness value. The index(positions) of the vector can correspond to situations (states) like position on the road, speed, distance to next curve etc and values (the actions) corresponds to what to do when in a particular situation; brake sharply, turn left, increase speed a little etc. Assume now we have several such vectors, a generation, with random entries and a simulator (or real cars, which is expensive). Let each of the individuals control a car with a discouraging result; all cars drive off the road after only a few meters. This length, "fitness", is stored together with the vector. But some of them (those with "greatest" lengths) managed to stay a little bit longer on the road (although not for long) and becomes parents for the next generation who inherits parts from their parents; someone inherits right-turning abilities from the father and braking abilities from the mother resulting in better children (and worse; neither*

being able to turn nor brake). After many generations (Darwin:Survival of the Fittest) we (may) have an Emerson Fittipaldi (or Ronnie Pettersson or Michael Schumacher).

A nontechnical, humorous comparison between optimization methods can be found on the Internet[Sar94]:

"Training an ANN is a form of numerical optimization, which can be likened to a kangaroo searching for the top of Mt. Everest. Everest is the _global optimum_, the highest mountain in the world, but the top of any other really tall mountain such as K2 (a good _local optimum_) would be satisfactory. On the other hand, the top of a small hill like Chapel Hill, NC, (a bad local optimum) would not be acceptable".

...< parts removed>

"Notice that in all the methods discussed so far [ANN and Newton etc], the kangaroo can hope at best to find the top of a mountain close to where she starts. In other words, these are _local ascent_ methods. There is no guarantee that this mountain will be Everest, or even a very high mountain. Many methods exist to try to find the global optimum.

In simulated annealing, the kangaroo is drunk and hops around randomly for a long time. However, she gradually sobers up and the more sober she is, the more likely she is to hop uphill.

In a random multistart method, lots of kangaroos are parachuted into the Himalayas at random places. You hope that at least one of them will find Everest.

A genetic algorithm begins like a random multistart. However, these kangaroos do not know that they are supposed to be looking for the top of a mountain. Every few years, you shoot the kangaroos at low altitudes and hope the ones that are left will be fruitful, multiply, and ascend. Current research suggests that fleas may be more effective than kangaroos in genetic algorithms, since their faster rate of reproduction more than compensates for their shorter hops".

5.4 Control systems

The intention of Control Systems (Fig. 12) is to keep the controlled system in some desired states, like keeping the temperature in a room between 19 and 21 $^{\circ}C$ by adjusting the heaters. One problem with controllers is the risk for oscillating behavior, instability, due to large or old/delayed feedback. Some other problems involve non-linearity, friction and backlash. Reinforcement learning may be used to learn to control a process but one of the problems is the need to fail in order to learn, which can be disastrous. Both Bertsekas[Ber76] and Åström[ÅW84] has written books in the areas of Dynamic programming and Stochastic Control. They indicate the close relationship between learning and control. A controller is often designed from a model of the process to be controlled and requirements for step-responses and security margins.

The most popular controller structure is the Proportional-Integrative-Derivative (PID) controller where the control output is the sum of the error between the de-

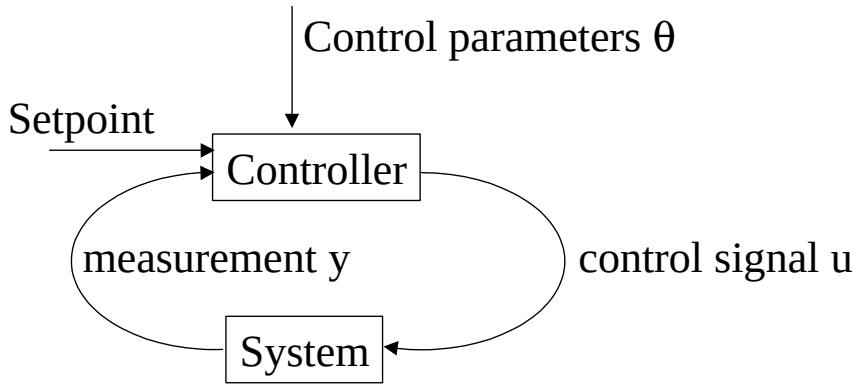


Figure 12: A schematic view of a controlled system.

sired value (set-point) of a thing to be controlled and the actual value multiplied by a constant parameter K_P and constants K_D , K_I multiplying the derivative (difference between) and integral (sum) of previous errors. The constants' values are often selected by using rules[13] of thumb. These rules says you should increase some parameters until a instability occurs (self-oscillation) and then reduce the parameters to half of their value (a slightly more complex method is used in reality).

An improvement can be made by writing a recursive version where anti-windup is a main feature; avoiding the summing up of errors outside of the controllers ability due to limited control signal (like $0 - 5$ Volts is allowed, not -12000 to $+12000$ Volts which maybe was calculated by the PID-formula). Bumpless transfer from manual mode is also easily achieved with such a controller. Modern controllers use a state-space matrix approach but seems to be rather uncommon, maybe due to the mathematical modelling necessary to have such systems. It is important to include a model of human behavior in such a controller; this was dramatically illustrated by the JAS-fighter crash in the middle of Stockholm 1993.

5.4.1 Adaptive Control

By having the controller listen to the system while controlling, it is (sometimes) possible to build a mathematical model of the controlled system and when you have a model you can calculate the optimal controller [Åst95](Fig. 13). The problem here is the bias-variance dilemma; if you are successful in your controlling, the system will have a constant output destroying the possibility of achieving a good mathematical model. This results in model-gliding into modelling the noise of the sensors or something else and the controller will then behave strangely. Paradoxically, by adding random noise to the control signal

[13] The Ziegler-Nichols approach is popular.

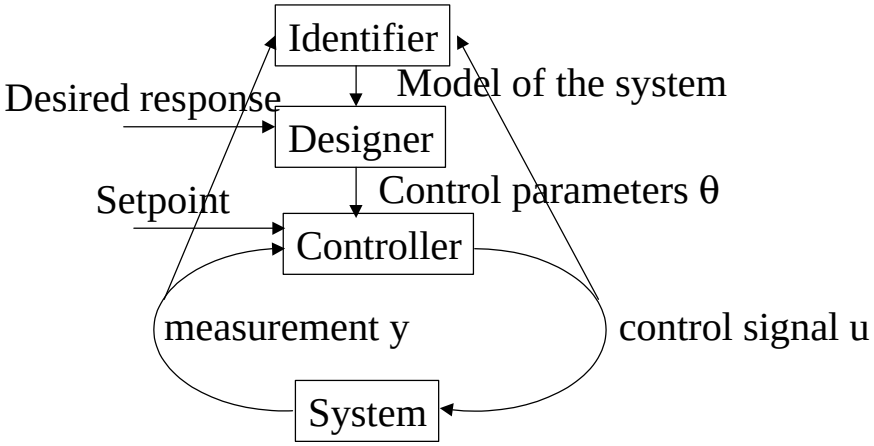


Figure 13: Adaptive control.

we ensure excitation of the system and manage to keep the model in place. There are also problems with adaptive control in terms of the large amount of mathematics used (typically a set of 20x20 matrices has to be solved with a Ricatti-equation), requiring expensive computers and deep numerical analysis knowledge. There are also difficulties in managing non-linear systems and multi-dimensional control. A company (First Control Systems AB, Västerås Sweden) has invented a patented relay method to induce excitation in systems during a tuning phase with small steps causing responses.

5.5 Game Theory

There are many types of games, e.g. children's games. However, social and economic interactions, (such as negotiations of a labor agreement), two newspapers setting their advertising rates, auction for radio frequencies, court cases etc. can be seen as games. These are all situations where a decision maker must take into account the actions of others, this interdependency is the essence of a game.

Definition 12 *In a static game[BF98], no information of the opponents' choice is available and there is a lack of interest in future interactions. A static game can be described by its strategic form; lists of players and their pure strategies and lists of payoffs normally presented in a payoff table. The moves are made "simultaneously".*

Each player must form some *belief* about what the opponent will do. An example of a static game is paper-rock-scissors. A strategy is *strictly dominant* if it is best whatever the opponent do. One popular example with a strictly

		Thief #2	
		Hang tough	*Confess*
Thief #1	*Hang tough*	(1 year, 1 year)	(20 years, freedom)
	Confess	(freedom,20 years)	(10 years, 10 years)

Figure 14: The Prisoner's Dilemma. If the thief cooperates with the police and his partner does not, he will go free. The thieves cannot communicate with each other.

dominant strategy (confess) is the Prisoner's Dilemma (Fig. 14). A "confess" will result in a lower punishment, no matter what the other thief does.

If both parties can agree to adopt a (better) strategy (hang tough) we have a *cooperative game*.

At Nash equilibrium (after prof John C. Nash, awarded with the Nobel Prize) each player's (self-confirming) strategy is the **best response** to the belief that the other player(s) will adapt their Nash equilibrium strategies. A more detailed description is available in [BF98].

Definition 13 *In a dynamic game[BF98], the players move in a fixed sequence. The players moving later in a game do so knowing the moves others have made before them. Those who move earlier must take this into account in devising their optimal strategy.*

One example of a dynamic game is bridge. If the players of the game in Fig.15 has perfect information, the optimum bid will be 5 ♣ by N-S and P by E-W (solved by doing a game tree) but in an **imperfect information** game N-S can force E-W into a guess (by bidding 4 ♣ or otherwise) whether they can take 10 tricks (as denoted in the table) or 9 or even 11 tricks.

Definition 14 *A game in which players meet in strategic interaction repeatedly are referred to as* ***repeated*** *games.*

Definition 15 ***Zero-sum*** *games are games where one earns the same amount as the other loses (i.e. bridge).*

Gerald Tesauro has made some interesting work[Tes00] regarding multi-agent Q-learning. Price wars appears when two competing sellers offer identical products to a large population of customers, not knowing everything. The same theme (asymmetric information) actually re-appears in the Nobel Prize in economy[14] 2001 by Spence, Akerlof[15] and Stiglitz.

[14] Originally not a genuine Nobel Prize, it was introduced by the Swedish Riksbanken 1969 and named "Bank of Sweden Prize in Economic Sciences in Memory of Alfred Nobel".

[15] He coined the term "Lemon" for a used car with hidden defects in "The Market for Lemons: Quality Uncertainty and the Market Mechanism", published in Quarterly Journal of Economics in 1970.

	N-S bid		
	1♣	4♣	5♣
NS-Score	+70	-150	-200

	E-W bid		
	P	2♠	4♠	5♠
NS-Score	?	-170	-420	+50

Figure 15: The partnership N-S are able to select a bid first (in clubs) and then E-W decides. On this hand it is assumed N-S are able to take 7 tricks with clubs as trumps and E-W 10 with spades.

5.6 Multi Agent Systems(MAS)

By having a set of agents we have a multi-agent system (MAS);

Definition 16 *Multi-Agent System, MAS[Fer99]. The term MAS is applied to a system comprising the following elements:*

1) An environment, E, that is, a space which generally has a volume.

2) A set of objects, O. These objects are situated, that is to say, it is possible at a given moment to associate any object with a position in E. These objects are passive, that is, they can be perceived, created, destroyed and modified by the agents.

3) An assembly of agents, A, which are specific objects ($A \subseteq O$), representing the active entities of the system

4) An assembly of relations, R, which link objects (and thus agents) to each other.

5) An assembly of operations, Op, making it possible for the agents of A to perceive, produce, consume, transform and manipulate objects from O.

6) Operators with the task of representing the application of these operations and the reaction of the world to this attempt at modification, which we shall call the laws of the universe.

By the way, the author actually made his Master Thesis[Pro84] by using calculating parallel agents.

$$A = in.B + err.C$$

where A,B and C are agents and "in" is an input signal to agent A and if received the calculation is continued by agent B (with a similar formula). This type of agents does not have much similarity with today's agents and definitions.

5.6.1 Multi Agent Learning

This is the case when there are multiple learning agents (defined in 5.6.1) which try to fulfill a goal.

Typical domains for those agents are:

- collaborative, where a group of agents share a common goal, and
- adversary, in which there are agents with competing goals.

RL is a powerful method for solving Markov Decision problems (MDPs) (See page 27) and the RL algorithm Q-learning (See 5.1.12) is guaranteed to converge in a single agent system if the state-space is small enough so that lookup table representation can be used [WD92]. But convergence is not yet shown for MAS. Some of the problems arising in real-world problems when multiple agents act and learn are that they are not fully Markov in nature - resulting in non-stationarity and/or not fully observable problems because the other agents also changes their behavior.

Some other problems are related to the necessity of coordination and can sometimes be solved by communication, convention or learning[Joh99].

The lure of multiagent systems There are a lot of advantages and pitfalls with multiagent systems. Some of the advantages are collected by Sandip Sen [Sen00]:

1. Enables coherent distributed applications
2. Utilization of distributed resources
3. Fault tolerant systems/design
4. Modular design
5. Incremental system development

Pitfalls of Multiagent Systems

1. Lack of globally consistent knowledge
2. Conflicting goals
3. Dynamic/Open environment (lie/cheat, new agents come and go)
4. Reliance on accurate communication

5. Unintended global side effects of local decisions; one example from the traffic area: get home as quickly as possible will result in chaos unless rules were followed.

Manuela Veloso pointed out (in a personal conversation with the author) the difficulties in adversarial games; suppose two teams meet in a match and the result is 3-1. A year later they meet again after lots of training and practice but the result is again 3-1. Have they learnt anything? It is obviously difficult to find out if the learning agents increase their performance when they are playing against other learning agents.

Example 17 *Iterated Prisoner's Dilemma ([SC85]). The Prisoner's Dilemma (Fig. 14) can be repeated multiple times, making the agents learn the best behavior (Hang tough). Agents with longer memory fared best.*

Example 18 *Visiting the pub. Suppose there is a choice of which evening to visit a pub. Each agent decides to go on a certain day and, if unlucky, the pub will be crowded or, alternatively, almost empty some days. The agents have to select a behavior to achieve global optimum. (This problem is called Arthur's bar problem [Art94] as cited in [WT00].)*

Periodic Team Synchronization (PTS) domains contains limited communication periods. One example is football (soccer) play where there are predetermined Multiagent Protocols; *locker-room agreements* before/between the matches. During the match they act autonomously with limited communication possibilities.

Layered learning [Sto00] is a machine learning paradigm defined as the following four principles:

1. A mapping from inputs to outputs is not tractably learnable. There is limited communication and noisy environments. Instead learning is done by breaking down a problem into several task layers. At each layer, a concept needs to be acquired.

2. A bottom-up, hierarchical task decompositions is given. Not automated, the layers are defined by the machine learning opportunities (algorithms) in the domain.

3. Machine learning(ML) exploits data to train and/or adapt. Learning occurs separately at each level. The learning can be On-line or Off-line and then frozen for future use.

4. The output of learning in one layer feeds into the next layer. The goal of learning is to generalize the target mapping from the training examples which provide the correct outputs for only a portion of the input space.

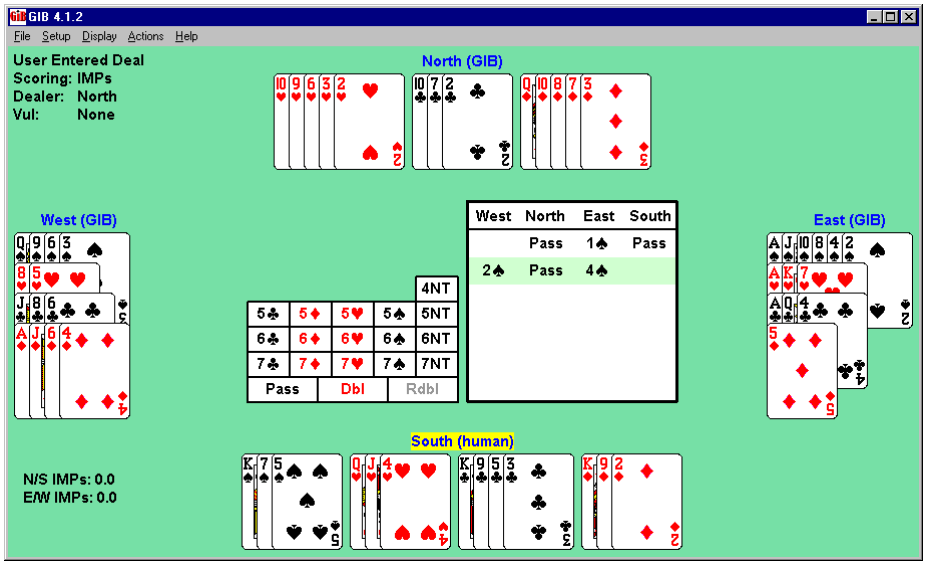


Figure 16: A screenshot of GIB during the bidding.

6 Related work

6.1 GARIC

In GARIC, Generalized Approximate Reasoning-based Intelligent Control [BK92], Reinforcement Learning (RL) is combined with Fuzzy Logic (FL) and Artificial Neural Nets (ANN). The fuzzy rules are added manually, giving the control problem a structure but the parameters are learnt through reinforcement learning.

6.2 GIB

Matthew Ginsberg has produced, among other things, the following papers in the bridge area;

- A step towards an Expert-Level Bridge-Playing program [Gin99] - describes the algorithms behind GIB (Fig 16).
- How computers will play Bridge [Gin96a] - shows how computers can invent bidding systems "on-the-fly" by simulating inside each player and achieving same results if the same random seed is used.
- Do computers need common sense - to reduce problems to a smaller instance (note on the internet).

- An analysis of the law[16] of total tricks [Gin96b] - shows it is correct to about 70%.

6.3 Game-tree search applied on Bridge

Ian Frank has written an excellent doctoral thesis[Fra98] where he shows that simply repeating and averaging plays with a double-dummy solver does not work; Strategy fusion and Non-locality are some of the problems occurring and resulting in suboptimal play when trying to extend a double-dummy solver into a single-dummy solver. Some algorithms like $vm\_\alpha\beta$ are suggested.

Dynamic Backtracking [Gin93], Partition search [Gin96c] and Transposition Tables are some of the areas investigated by Ginsberg and used to reduce the effective branching factor down to 1.27.

6.4 Computer Bidders

Most programs basically use a rule-based approach. The first one was by Carley[Car62] with four (!) bidding rules with a performance like "The ability level is about that of a person who has played dozen or so hands of bridge and has little interest in the game". Wasserman (cited from [Fra98]) divides the rules into classes that handle different types of bids, such as opening bids, responding bids or conventional bids. The classes are organized by procedures into sequences."Slightly more skillful than the average duplicate Bridge player at competitive bidding". MacLeod [Mac91](cited from [Fra98]) tries to build a picture of cards a player holds but cannot represent disjunction; when a bid has more than one possible interpretation. Staniers [Sta77](cited from [Fra98]) introduces look-ahead search as an element of Bayesian planning and Lindelöf's COBRA [Lin83] uses quantitative adjustments. Lindelöf claims that COBRA bidding is of world expert standard. Ginsberg has invented a relatively short but cumbersome(Fig. 17) way of representing bidding systems.

GIB uses Terrorist's Moscito (Major-Oriented Strong Club with Intrepid two Openers) because it is difficult to find a good defense. It is complicated to remember for a human person but easy for a computer because the conditions are well-defined. Actually Moscito seems to be (temporarily?) abandoned (since 2001) due to redesign of the database language.

6.5 Joint-Action Learning

Caroline Claus and Craig Boutilier have written an excellent paper [CB98] in which they describe Multiagents learning and playing in repeated static games. The actions are taken concurrently as in the paper-rock-scissors game. They call the classical Q-learners *Independent Learners* (ILs) and the agents who learns their own actions in conjunction with other agents *Joint Action Learners*

[16] A rule that says "the total number of tricks for both pairs is the same as the total number of trumphs". Invented/Discovered by Jean Rene Vernes [Ver69] and Popularised by Larry Cohen and Marty Bergen in [Coh92].

```
1*#08115 "  {5{[CDHS]:"          !.!
%0% %.% <0x1083> *2153*#08116 "   5[+#b]"             !DOPI!
%6% %:H[^:]*~8&[^:]*~8&% *2160*#08117 "   5[++#b]"
!DOPI!                            %7%
%:H[^:]*~8&[^:]*~8&[^:]*~8&% *2159*#08118 "   5[+++#b]"
!DOPI!                            %8% %:H~8&.~8&.~8&.~8&%
*2156*#08119 " 4N:$7"               !.!
%0% %.% *1501*#08120 "  (5N|[67][CDHSN]):"   !.!
%0% %.% <0x1083> *2153*#08121 "   X"                  !DOPI!
%3% %.% *2154*#08122 "   P"                  !DOPI!
%4% %:H[^:]*~8&% *2155*#08123 "   X"                  !DOPI!
%5% %:H[^:]*~8&[^:]*~8&% *2154*#08124 "   P"
!DOPI!                            %6%
%:H[^:]*~8&[^:]*~8&[^:]*~8&% *2155*#00126 "
.*{{{{{{{{{{{{[CDHS]<#j=3><#k=6>:P:4N"        !.!
%0% %.% *2037*#00127 "  :(P|X):"             !.!
%0% %.%#00128 "   @ACE_BLACKWOOD~"    !BLACKWOOD!
%1% %.% *1501*#00129 "   @ACE_RKCB~"         !RKCB!
%1% %.%#00130 " [123]{{{{{{{{{{{{.:P:5N$12"      !GSF!
%0% %.% *2229*#00131 "  :,:"                 !.!
%0% %.%#00132 "   7#a"                !.!
%5% %:H[^:]*~9>#a% *2034*#00133 "   7#a"                !.!
%5% %:H[^:]*~6=#a% *2034*#00134 "   6#a"                !.!
```

Figure 17: A small part of the GIB-database, showing how the Ace-asking bid Blackwood(4 NT) is described. The *numbers* indicates priorities.

(JALs). The strategies are denoted $\pi(a^i)$, designating the probability of agent i selecting action a. A strategy π is *deterministic* if $\pi(a^i) = 1$ for some a^i.

Definition 19 *It is called Fictitious play if each agent i keeps a count* $C^{\,j}_{a^j}$ *of the number of times agent j has used action* a^j *in the past.*

By doing this it can calculate relative frequencies of each of j:s moves as an indication of j:s strategy. For each agent j, i assumes j plays action a^j with probability:

$$Pr^j_{a^j} = \frac{C^j_{a^j}}{\sum_{b^j} C^j_{b^j}}.$$

By using stateless Q-learning;

$$Q(a) \leftarrow Q(a) + \alpha(r - Q(a)),$$

where α is the learning rate, an agent gets an estimate of the value of taking action a. The actions can be their individual or both's actions combined into a *joint-action* pair.

Definition 20 *If the learner learns Q-values without regarding actions performed by other agents it is called independent (IL).*

Definition 21 *If it learns Q-values with actions combined with the actions performed by other agents it is called joint-action learner (JAL).*

Example 22 *If the agents can select action a0 and a1, an IL has 2 actions to consider (its own a0 and a1) but a JAL has four action-pairs in its table; <a0,a0>, <a0,a1>, <a1,a0> and <a1,a1>.*

To determine the relative value of individual actions, an JAL maintains beliefs about the others strategies. The expected value (EV) of the individual action a^i is

$$EV(a^i) = \sum_{a^{-i}} Q(a^{-i} \cup \{a^i\}) \prod_{j \neq i} \{Pr^i_{a^{-i}[j]}\}.$$

Those are the Q values multiplied by the probabilities of having the other part of the combined action selected by the other agent(s). The term a^{-i} denotes all actions except action i. The EV is used instead of Q for action selection. Both sides calculate the EV-matrix and both will know the optimal combination of actions.

Some problems might arise; when to select such an action and what to do if several joint-actions have the same values (multiple equilibria)? The combined method uses a $\rho-$weighted sum of the Q values and the EV:

$$Comb(a^i) = \rho MaxQ(a^i) + (1 - \rho)EV(a^i) \tag{6}$$

to avoid such problems.

Example 23 *The climbing game. Suppose we have a game-matrix with the following contents:*

	b1	b2	b3
a1	11	-30	0
a2	-30	7	6
a3	0	0	5

. (7)

The optimal, combined actions are a1 for player a and b1 for player b but the problem is how they should find this combination of actions.

a) Independent learners:

A simulation was done and the result is shown in figure 18

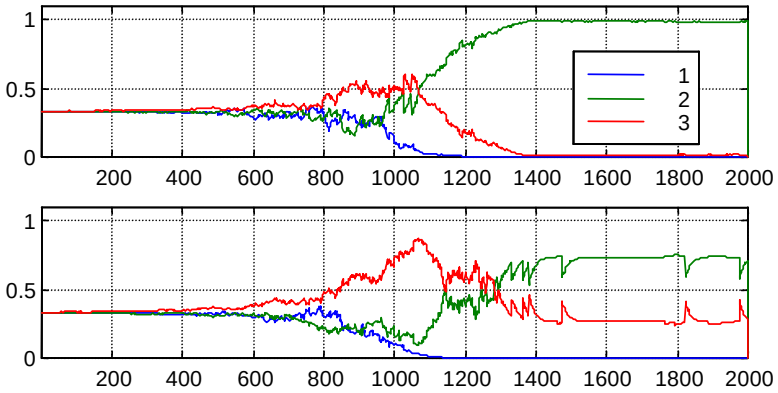


Figure 18: The policies for player a (above) and player b as a function of time. The learning was started at $t = 400$. The first action selected was a3 and b3 ($r = 5$) at about $t = 800$ and then the hesitating combinations a2/3, b2/3 ($r = 0..6$) were selected around time 1000 to 1400 and finally stabilized into a2, b2 ($r = 7$).

They found the following Q-values after a while:

Q-values for player a: $-5.4 \quad 6.9 \quad 2.4$,

Q-values for player b: $-10.3 \quad 6.9 \quad 6.0$

and did not succeed in finding the best action(-pair). The selected actions are a2 (6.9) and b2 (also 6.9). It is slightly less than seven because b also selects b3 (6.0) quite often.

b) Joint-Action

By using compound-moves $<a_i,b_j>$ *in the climbing game, the optimum was found (Fig 19).*

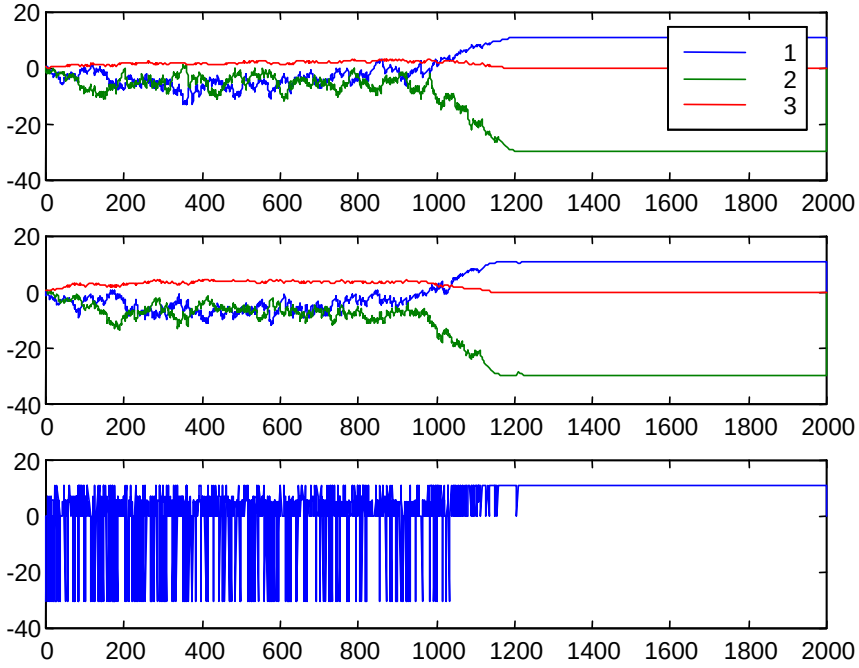


Figure 19: The combined value (Eq 6) and the reinforcement as a function of time. They found their optimal actions a1,b1 ($r = 11$) at $t = 1050$. Before that mainly a3,b3 was selected, resulting in poor reinforcement values.

7 Overview of my work

Here I present some of the work I have done some time ago which I think is related to this thesis' title and then some of the current research. There will also be a separate document, describing the control part.

7.1 Before my Ph.D-studies

7.1.1 Early Bridge Software and Hardware

Since 1975 I have written several popular programs shortly described in this note.

In the middle of the 70's the personal computer had still not arrived but there were programmable calculators and minicomputers. In order to become a proficient bridge player it is necessary to train and practise, for example the bidding. Since I am, at times, lazy I used the calculator to generate random deals for practising with my partner. This way we avoided the work and time-delay in shuffling cards, dealing them out and sorting the hands only to find out that the bidding often was trivial (like Pass-Pass). A PC-based program "givar" is still used by people in Sweden (mainly in the cities of Örebro, Uppsala; Skellefteå and Umeå) and perhaps Norway.

The tournaments were calculated by hand and in the late 70's with mini-calculators with paper-rolls to find the errors. It was natural to do the scoring on PCs when they arrived (around 1984). Today's (2000) bridge is heavily computerized with the tournament scores entered from protocols. The results, including the deals, are handed out after the tournament and entered into the bridgefederation's database. Internet games are also performed with players physically located all around the world.

I found out that the administration was still old-fashioned because of this entering of scores manually into a centralized computer, so I developed a distributed computer system based on smart terminals (from TeleFrang first and then custom made) where the users entered the information immediately at the table. There was a lot of other advantages as avoidance of Guide-cards (shows where to go next round) and no necessity of pair-numbers; a name worked fine and it was also possible to arrange other new types of tournaments. It was successfully tested by the Swedish Ladies team in Linköping 1993 and the idea was recently commercialized by Jannersten Förlag AB as Reporter®.

7.1.2 Experiments with Adaptive Control

One of my first projects was designing a timber dry chamber controller. The problem with such controllers is the variability of operating conditions; sometimes the timber is very damp and sometimes dry, seasonal variations of the weather etc. A traditional PID-controller can not manage all these conditions and has to be retuned; not an easy task for a novice on a saw mill. In order to cope with such complications I implemented an adaptive controller but found out that it is very difficult to do research alone in a small company.

In my first job as a computer and control-engineer at K Lidström AB I implemented an adaptive controller for the forest industry; a wood-drying chamber. In this chamber it was important to keep the temperature and moisture according to values from a drying scheme. The problem with conventional controllers is that they have to be adjusted depending on seasonal variations or even during a drying cycle (about 3 weeks); at the beginning it is a slow system with plenty of water in the timber. However, at the end the system has a quite quick step-response when most of the water is gone and problems with over-temperature arise. The reason for these over-temperatures is e.g. when it is dry outside (during winter), we could not ventilate to get the temperature down because then the humidity get too low and the logs crack. The people running these chambers did not have knowledge of control systems and it was difficult and slow to adjust the controllers. At that time (1985) there were few conventional adaptive controllers (I remember only Asea Novatune) and they were expensive as were computer chips and memory. Intel had a processor (8052AH-Basic) with a built-in Basic interpreter which made it possible to implement high level calculations (we did not find any suitable C-compilers for single-chip computers).

The Identifier part A recursive least-square identifier method[Lju81] was used;

$$\begin{aligned}
\widehat{\Theta}(t) &= \widehat{\Theta}(t-1) + K(t)[y^T(t) - \phi^T(t)\widehat{\Theta}(t)] \\
K(t) &= P(t)\phi(t) = \frac{P(t-1)\phi(t)}{\lambda + \phi^T(t)P(t-1)\phi(t)} \\
P(t) &= [P(t-1) - \frac{P(t-1)\phi(t)\phi^T(t)P(t-1)}{\lambda + \phi^T(t)P(t-1)\phi(t)}]/\lambda
\end{aligned}$$

The recursiveness made it possible to get a new estimate after each new sample without too heavy calculations. The $\widehat{\Theta}(t)$ was the parameter vector (we modelled our system as a discrete linear, fifth order system "ARMAX") and because we knew the parameters well, we could use a low start value of the covariancematrix, P(0); the value of the identity matrix I. The $\phi(t)$ was the old measurements (we used 20 samples) of input and output-values to/from the system and the $y(t)$ was the last measurement of the output (a 2 element vector in our case). The λ was a forgetting factor; old values were weighted with a lower value to make the model adapt to the new circumstances. A value of 0.995 was used which gave a rather slow adaptation. When the model ($\widehat{\Theta}(t)$) was estimated, control parameters with desired properties were calculated.

7.2 During my Ph.D-studies

First there is a short overview of my Ph.D work and then some of the topics as actual articles follows.

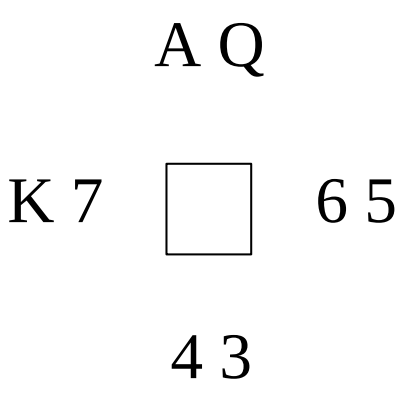


Figure 20: The finesse. A two-card situation; when south starts the play with a small, then west plays low or the king and depending on what card west plays north takes the ace or finesses the queen in the hope of west having the king (which is the situation here). The success rate of a finesse is a priori 50%.

7.2.1 A double dummy solver

This work was my preparation for a workshop in Linköping, Sweden 1997, discussing object-oriented programming in C++ on supercomputers. I think it would have been very difficult to implement this in Fortran and more complex to extend if written in a non-object-oriented language.

A program for analyzing a simple deal was constructed and run on an IBM SP2 parallel computer, gently placed at my disposal by the High Performance Computer Center North (HPC2N). The program took around 18 cpu-hours in total, using the minimax algorithm with $\alpha - \beta$ pruning. Six nodes were utilized but no dynamic load balancing was used. The dealt cards were split into sequences like 678, representing a single choice disregarding the difference in the tactical factor of playing the six or the eight. Only the last 10 out of 13 tricks were analyzed. The search-tree for just a two card deal is shown in Fig.20.

A hand consists of thirteen cards and the average number of possible leading cards is probably about 8 (high or low from each suit). Then the next player has typically three cards to choose from, possibly forming a sequence, say a later branch factor of two which gives $8 * 3 * 2 * 2 = 96$ on the first trick. The branch factor in the following tricks will be slightly reduced due to the rule of following suit. When a player has run out of a suit he will discard any other card (possibly ruffing with trumps) and the branch-factor then suddenly increases. As a rule of thumb one can estimate the average possible number of legal plays on a hand as $96^{13} \approx 10^{25}$. If this was the case for the SP task, it solved 96^{10} ways of play in 6 nodes resulting in 10^{14} lines of plays analyzed per second. This is not reasonable, so in this case the sequencing and $\alpha - \beta$ worked out

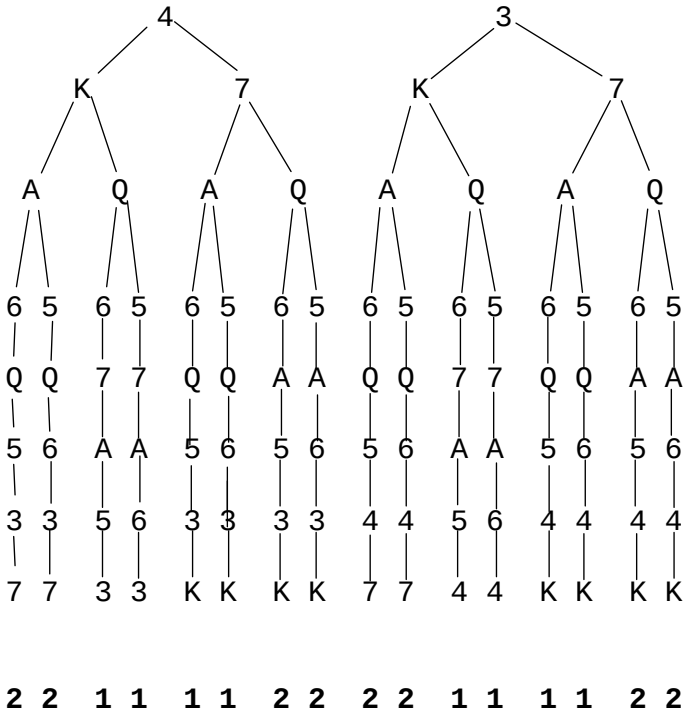


Figure 21: The search-tree for the simple case with just two cards. The numbers at the bottom indicates the number of tricks taken by the North-South.

well and the deal had fewer selections because there were plenty of sequences reducing the problem size. Frank([Fra98]) has simulated deals and found that most deals have around 10^{27} legal plays and very few have up to $10^{36}$and some as low as 10^{23}. It is interesting to see that humans are able to solve these kinds of problems in minutes.

One way of reducing the number of evaluations would be to store the calculations of state-values. Many states can be reached different ways like playing first the 3 and then the J or the reverse order perhaps resulting in the same state. By using this method we end up in a search *graph* instead of a search tree.

Suppose we remember the states with only 5 cards left on each player. There are $\binom{13}{5} = \frac{13!}{5!(13-5)!} = 1287$ combinations of choosing 5 cards out of 13 by each player resulting in $1287^4 \approx 10^{12}$ possible states but some (many?) of them might be impossible to reach due to the follow suit rule.

Another way of reducing the search is by using *transposition tables;* by storing only the relative ranks of the remaining cards, even fewer variations on end plays will then be available[Gin96c].

7.2.2 A Control Approach to Reinforcement Learning

A study of applying reinforcement learning methods to control problems was done and presented as a poster at Reglermöte 98' in Lund, Sweden [Dag98, In swedish]. Genetic algorithms were combined with Fuzzy logic and Artificial Neural Nets in a GARIC-like approach in a modelled car-driving environment, this with some success, .

7.2.3 The man in the box

A review (in Swedish) of the GIB-software 1999 for the Swedish magazine Svensk Bridge[Pro99] It popularly reports the programs' strengths and weaknesses but mainly in a positive and entertaining (?) way.

7.2.4 Multi Agent Cooperative Reinforcement Learning

This presentation was made in Lugano, Switzerland 1999 with overhead slides. The article was written and extended later but mainly based on the results presented. The slides and the software are available via my old homepage; www.cs.umu.se/~kallep and the article is included later on in this thesis.

7.2.5 Robotic Soccer

Team Sweden participated in the Sony four-legged robot league in RoboCup 2001[Htt01]. I supervised a sub-group from Umeå and the Team won the informal European Championships in Paris but did not succeed in the real World Championships in Seattle, USA. We implemented[Car02] eighteen gait (walking) styles for the robots and made them move fastest of them all at the time.

A mistake, forgetting to replace the goalie's memorystick at half-time, reduced our chances of winning (the goalie joined the opponents..).

7.2.6 Joint-Action Learning

We have tried bridge-inspired game-matrices with 4 players and 16 actions each (one bid-round), and it seems to work well (scale up). However, Sandip Sen (personal conversation) mentioned that this is not always the case. The players were keeping track of action-quadruples and always found the optimal actions in a four-dimensional game matrix filled with random numbers. This method is not directly applicable to the bridge domain because the decisions are sequential in bridge but illustrates how to solve the difficulties for four learners to synchronize their actions to reach the global optimum. There are no published results.

7.2.7 Large-scale Q-learning

A long-run test with a powerful computer in an attempt to produce some useful bidding systems was done. Some practical issues like selecting parameters and how they behave in a multi-agent-learning scenario was studied. This work was presented at the American Association for Artificial Intelligence (AAAI) spring symposium 2002 at Stanford University, Palo Alto, California[TS02] and the article follows.

7.2.8 The Sum Game

A "mini version" of bridge was invented to simplify the understanding of why the learning was so slow. Standard Q-learning does not manage to learn some simple two-player games although GAs can solve them easy. The game is only published in this thesis as yet.

7.2.9 Genetic Bridge

A simple try with GA was made to solve bridge-bidding problems. Only five opening bids were tried and this method has to be investigated further before we know if it works in a satisfactory way. No results are published but the experiments looks promising.

7.2.10 Boom-Tip Control

A normal forestry hydraulic crane consist of booms with several joints and requires a lot of human skill to run with high productivity. The boom-tip moves in three dimensions (xyz) but the joints often work with four dimensions; three angles and a telescopic (transversal) dimension. An attempt to let the computer take care of the extra degrees of freedom (redundancy), hopefully reducing human stress and fatigue is made and seems to work reasonably well, at least in the lab. It will probably be easier to learn to run the crane.

Only traditional control completes the inverse kinematics based solution today but there seems to be an advantage with learning systems, making the crane-control adapt to environmental, machine and human behavior. A separate report on this issue will be available at the end of 2003. A presentation of some ideas regarding the storing of derivatives was made at the Swedish Workshop on Autonomous Robots (SWAR'02) in Örebro, Sweden, but this method was abandoned in favour of a heavily optimized on-line calculation. We had to implement integer based trigonometry routines to solve the inverse kinematic problems fast and i.e. the atan2(y,x) required 8 different cases depending on the size and sign of x and y and then the division y/x had to be calculated in three different ways to avoid overflow. An optimized utilization of the redundancy in the crane was also carried out.

8 Cooperative Reinforcement Learning in Multiple Agents

8.1 Introduction

To make agents learn simultaneously seems to be a complicated task, which it of course is. The typical problems arising are coordination problems, resulting in non-optimal behavior when stuck in non-global optima.

8.2 Reinforcement Learning

The environment responds with a score, indicating the result of the collected behavior performance in this case.

8.3 Multiagent Coordination

How should agents coordinate their efforts? Suppose you and your friend are trying to learn how to play football, with your friend standing closer to the goal than yourself. Assume you both are blind. The feedback from the environment is whether you score a goal or not. The best way of playing for a well trained couple is to kick the ball in your partner's direction, (s)he catches the ball and scores a goal. Now suppose, as a start, you kick the ball and send it in a random direction. Then you update a table with the direction-angles and average number of goals scored on the axes. I guess the direction with the highest average is the direct direction toward the goal and not the angle pointing at your friend because (s)he also does not know in which direction the goal is, and thus sabotage your tries to cooperate. On the other hand your friend will not learn unless (s)he will never learn unless given some passes and you will be stuck in this suboptimal strategy.

Trust and Chance to learn In this particular case the first player has to give the second player a reasonable chance to learn.

8.4 The Application Task

I have studied the card-game of Bridge for the following reasons;

- It is a modern "bricks world" with social interaction
- It is governed by internationally well-known rules
- It contains stochastic/insecure behavior
- The game has (almost) reasonable problem-size from a computer point-of-view

The application falls naturally into a four agent problem with the players as agents. A bridge play consists of two phases; the bidding and the playing of the cards. During the bidding a pair agrees upon a trump suit and number of tricks to be taken by making *bids* ending into a *final contract* by *Passing.* The bids are: Pass, $1\clubsuit, 1\diamondsuit, 1\heartsuit, 1\spadesuit, 1NT, 2\clubsuit, ..7\spadesuit, 7NT$ and should be given in a *strictly ascending* order except for $Pass$ which may be made anytime. Four consecutive passes (all pass, which means no play) or three passes after a bid conclude the auction. In our experiments we assume each opponent to pass whenever it is his/her turn to bid, making the problem much simpler than in real bridge. The numbers 1..7 means the number of tricks minus 6 to be taken. Bidding 7 means contracting to take all 13 tricks and is thus the highest possible level (Grand Slam). This means you have a *limited information exchange* because you have to make a contract at least at the increasing level bidden (to get a positive score).

Card playing will not be handled in this article but there are several implementations including this phase, GIB e.g. ([Gin99]). The bidding is by many players considered to be the most important part of the game and there have been many discussions on what bid to make under this or that circumstance. The goal of the bidding is to find a) a denomination and b) a suitable level to contract for. We often try to find a trump suit with at least eight cards in common or contract in "no trumps" (NT) with stoppers in all suits. Some of the levels give bonuses if you a) bid up to (at least) this level and b) succeed in making the contract. One example of bonuses is bidding to the 6-level ("small slam"). If the pair does not make their contract the opponents receive points. Furthermore it can even be lucrative to bid to a contract which doesn't make (*sacrifice*). This is true when the penalty paid for not making your contract is inferior to letting opponents score higher by being permitted to make contract of their own. In this study we have only studied offensive bidding, without opponents, due to the difficulties arising in coordination, simulation time and memory space limitations.

The pair normally have some agreements, *a system*, on how to bid on certain hands and also some *conventions* for specific purposes like the bid 4NT with the responses $5\clubsuit$ showing 0 aces, $5\diamondsuit$ one etc. The agreements are normally supplemented with a scale of how to estimate the strength of a hand and a normal way of doing this is to assign $4p$ to an ace, $3p$ to a King, $2p$ to a Queen

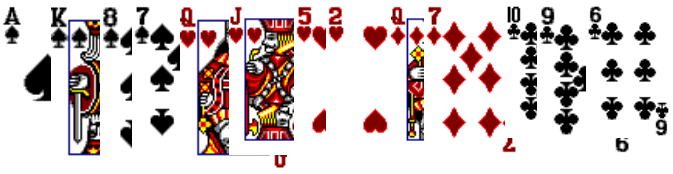


Figure 22: A typical hand

Card	Strength Value
Ace	4
King	3
Queen	2
Jack	1

Table 1: High card points

and $1p$ to the Jack (Knave) and some extra points for distributional values. As a rule of thumb a pair needs $33p$ between the two players to make a small slam.

The task can be categorized as a non-discounted episodic task.

8.5 Representation levels of a hand

A player sees its cards as presented in Fig. 22, but using that format in computer representation would, of course, be unnecessarily space-consuming. A shorter representation with almost the same meaning (losing the graphic "personality" of the cards) is:

$$\spadesuit AK87$$
$$\heartsuit QJ52$$
$$\diamondsuit Q7$$
$$\clubsuit T96$$

This is the standard way of presenting deals in e.g. books and newspapers. An even simpler representation is AK87, QJ52, Q7, T96 or AKxx, QJxx, Qx, T9x with the order of suits understood. The total number[17] of possible hands are $\binom{52}{13} = \frac{52!}{13!39!} = \frac{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48 \cdot 47 \cdot 46 \cdot 45 \cdot 44 \cdot 43 \cdot 42 \cdot 41 \cdot 40}{13 \cdot 12 \cdot 11 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2} = 100 \cdot 49 \cdot 47 \cdot 46 \cdot 43 \cdot 41 \cdot 2 \cdot 17 = 635013559600$. There are 8192 different (AK92, AK82, etc.) suit combinations, 560 shapes (5431, 5143..) but only 39 hand patterns (5431,5422, etc. disregarding suit-order) [And99]. There is a method to evaluate the strength of a hand, *the high card point* scale (h.c.p.)invented by the American Milton Work and presented in table 1.

The representation of a hand can be 4432 (4 spades, 4 hearts, 3 diamonds and 2 clubs), 12 h.c.p., where first the distribution is described and then its

[17] Often wrongly calculated in the literature due to round-off errors

Suit	Distributional value
no card (void)	3
one (singleton)	2
two (doubleton)	1

Table 2: Distributional points

high-card strength. A hand with AKQJ, AKQ, AKQ, AKQ contains 37 points which is the maximum. The number of possible hands in this representation is $560 * 38 = 21180$. The total number of h.c.p. in a deck is 40 and the average strength is exactly 10. Of course this scale is just meant for making a rough estimate of a hand. Most bridge players adjust (reevaluates) the value of the hand, maybe dynamically, during the bidding. This is because players receive new information from each bid, e.g. about fits (how well a player's card and his partner's work together). One add-on invented by Charles Goren is the method of evaluating distributional strength in *distributional points* (table 2).

The total value (h.c.p + distributional points) of our hand is $12 + 1$ (doubleton $\diamondsuit$) $= 13$ points. These distributional points are normally not added during a search for a no trump (NT) contract. Several other approaches are available, such as the first ever high-card evaluation, Ely Culbertson's Honor Trick and long-suit points or, the modern, losing trick count (LTC). Another approach is "The Law of Total Tricks" [Ver69] which says that the total tricks for the two sides equals the total number of trumps they hold in their respective (between the two of each pair) longest suit. This is not always true but can be of some use in competitive bidding. Thomas Andrews [And99] has made some analysis showing that the value of the Ace is underestimated in the 4321-scale.

These strengths are used to define a bidding system in terms of distribution and strength intervals. In our approach we reduce it further by just keeping the distribution of the cards (the shape) and the range of strength:

$$4432, medium$$

where medium might show 12-14 h.c.p.

We limit the number of possible distributions (by simply throwing away examples) down to 104 (at most 5 cards in any suit) and the number of strength-intervals are set to 7 to reduce the state-space into a reasonable size, covering about 79.4% (table 8 on page 76) of all hands. This is maybe not the optimal way of reducing the state space; maybe it is better to reduce it to the 7x104 most common hands[18] (shape and/or strength); a 6322 shape is more common than a 5440. Instead of reducing the number of shapes, the shapes may be grouped together, covering the full game of Bridge - instead of solving an approximate problem correct we can solve the correct problem approximately. The problem is to find this grouping. The total number of distributions (shapes) are 560.

[18] This is actually done later on in this thesis.

Max length	No of shapes
4	20
5	104
6	224
7	336
8	420
9	480
10	520
11	544
12	556
13	560

Table 3: Number of shapes

Points:	23	24	25	26	27	28	29	30	31	32	33	34
NT play	8		9		10			11			12	
8-card		9		10			11			12		13
9-card	9		10			11			12		13	

Table 4: Estimated number of tricks with given strength and trump-length.

8.5.1 Trick-taking abilities

The reason for having a point-count method such as outlined above is, of course, to estimate the number of tricks the two hands can take together. Table 4 (implemented with if-statements) is used as an evaluative function when trying to find the optimal contract. When the bidding has finished the number of tricks taken by the actual pair of hands is found via the table and then the resulting score is calculated. This is of course a very coarse approximation of real play results. Some additional corrections are made to make the estimates more realistic; if the pair does not have at least 4 cards in common in all suits in a NT contract, the number of tricks is reduced by one (which maybe is not a correct reduction).

A reasonable guess is that this function is correct in about 70% of the cases. Thomas Andrews [And99] has used GIB to estimate the number of tricks taken using similar point-count functions.

The table could also be made probabilistic (like no trump with 24 h.c.p. results in 9 tricks in 50% of the cases, 8 in 28% etc.) but it is not done here, although it is simple to implement. Some advanced playing methods like taking a finesse, executing a squeeze or a throw-in play makes these things hard to estimate but GIBson (a single-dummy solver, compared to GIB which is a double-dummy solver) or real players are better, but slower, at making estimates than this table. A better estimate is made by collecting real statistics from databases available from the Internet. A GIB-version from 1998 has played a lot of hands against itself and I wrote a program to collect some statistics. (if you are doing this yourself on a PC, do not forget to switch the order of bytes

in the 4 bytes integer from Motorola to Intel) The play is not very good but, on the other hand, nor is the defense. I counted the number of tricks taken in different contracts and indexed them on strength and number of tricks(Tr). Only pure high card points were used, no distributional values were added. 700,000 hands were used, each play takes GIB something in the order of a few minutes, resulting in a CPU-year of play in total (done in parallel). A human playing 100 deals/week (quite a lot) would need 134 years to get a corresponding experience. The entries in table 5 denote the relative frequencies in percentage of the actual result.

Some examples are; With 28 h.c.p. and 9 spades a small slam (6S) has $37 + 12 = 49\%$ chance to succeed but 6NT only $12 + 3 = 15\%$ chance according to GIBs simulation of real play.

Robin Hillyard [Hil01] has calculated the expected number of tricks at no trump, presented in table 6. It seems to agree with table 5 above. My guess is that these tables slightly overestimate the number of tricks achieved in real bridge because the analysis is based on playing optimally with open cards which is never the case in real life.

8.5.2 Scoring

After having determined the contract, the play takes over. Here the result is estimated by the table 4 on page 59 to save some time and effort. The result is positive if we make our contract (succeed in taking at least the promised number of tricks) or negative when failing to do so.

$$TricksToTake = LevelBid + 6$$

If the bidding reaches certain levels, bonuses are awarded – provided, of course, that the contract is made. A part-score (below game) is given a mere 50. A game bonus of 300 is awarded if contracts like 3NT,4♠, 4♡, 5♣ or 5♢ are reached and made. A Small Slam (6 of anything) gives a bonus of 700 and a Grand Slam gives even more. Each trick taken is worth 20 (♣♢), 30 (♡♠NT) or 40 (first NT) and the penalty is 50 for each undertrick. Real bridge scoring is slightly more complicated. Some examples: 2♠, 10 tricks = 50+4*30=170, 4♠, 10 tricks=300+4*30=420, 2♠, 8 tricks $= 50 + 2 * 30 = 110$, 4♠, 8 tricks $= -2 * 50 = -100$.

8.6 The Learning Task

We start by dealing two random hands until the pair of hands fulfill the limiting conditions (allowed shapes (distributions), point range in one hand, point range of the two hands together), then we classify each hand into the correct state (shape and strength) and select an opening bid according to the selection logic and Q-values in this state. Next, we select the other bids in a similar way until a pass is encountered. Then we evaluate the reached contract and the best possible contract. We calculate reinforcement and give it to the selected bids to update the Q-values. We add an extra, negative, reinforcement: blame the

NT play: High Card Points North and South Together,
all types of hands

Tr	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
6	32	28	21	14	8	5	3	2	1	1	1	0	0	0	0
7	24	30	31	27	19	13	8	5	3	2	2	1	1	0	0
8	10	17	25	31	32	26	19	12	7	4	2	1	1	0	0
9	2	5	9	16	25	29	29	22	14	7	3	1	0	0	0
10	0	1	2	5	10	18	26	32	30	23	13	5	2	0	0
11	0	0	0	1	2	5	11	19	29	36	35	27	15	7	2
12	0	0	0	0	0	1	3	6	12	22	34	46	54	50	38
13	0	0	0	0	0	0	1	1	3	5	10	18	28	41	61

trump play: Hands with 8 Spades together

Tr	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
6	5	2	1	0	0	0	0	0	0	0	0	0	0	0	0
7	20	12	7	3	1	1	0	0	0	0	0	0	0	0	0
8	36	33	25	16	9	5	2	1	0	0	0	0	0	0	0
9	27	33	36	36	30	21	14	8	3	1	1	0	1	0	0
10	10	15	23	31	37	40	36	30	22	13	7	3	1	0	0
11	2	3	7	11	18	25	35	40	42	40	33	23	15	8	1
12	0	1	1	2	4	7	12	18	27	36	45	50	48	39	34
13	0	0	0	0	0	1	1	3	6	9	15	23	36	53	65

trump play: Hands with 9 Spades together

Tr	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	4	1	1	0	0	0	0	0	0	0	0	0	0	0
8	22	17	11	6	3	2	1	0	0	0	0	0	0	0	0
9	36	34	32	24	15	11	6	2	1	0	0	0	0	0	0
10	24	30	34	38	37	32	25	18	11	5	2	0	1	0	0
11	8	13	18	23	31	37	42	41	38	32	19	11	10	2	0
12	1	3	4	7	12	16	22	32	37	45	50	49	45	36	29
13	0	0	0	1	1	2	4	7	12	18	29	40	44	62	71

Table 5: Distributed percentages of resulting number of tricks with a given strength and trump-length.

HCP	20	21	22	23	24	25	26	27	28	29	30
Tricks	6.1	6.7	7.2	7.6	8.2	8.7	9.1	9.7	10.1	10.6	11.1

Table 6: The average number of tricks in No trump with a given strength according to Hillyard.

one that first made a bid that was higher than the optimal (the other one just had to try to reduce the losses, although it might have been his fault in the first place). This is a non-discounted episodic task to learn.

The goal is to find a contract in notrump or in a suit with at least 7 cards together, at a correct level and to find the best (semantic) meaning of the bids. When the extended version will be ready (with four players) the following will also apply: Fight with the opponents to get a maximal score and try to make it difficult for them by bidding high quickly (preempting) in order to force them to guess.

8.6.1 Some Trade-offs and Problems

A trade-off between representation levels and precision contra training examples/time/space is needed. Explorations tend to destroy partner's confidence - we try to reduce exploration.It seems like the players have to learn simultaneously to coordinate. The initial values of Q are very important to make the start optimistic or pessimistic. We try to get a good start by pretraining - done by presetting some Q-values. Learning rates are important and influences performances and speeds of learning.

8.6.2 Allowed systems

There are usually restrictions applied to bidding systems to be used in tournaments. The rules are given by the bridge federations, Svenska Bridgeförbundet(SBF) and World Bridge Federation (WBF) respectively. This is so because it is difficult for a human opponent to find a reasonable defence within a few minutes to a highly artificial bidding system. In some circumstances it is possible to send a description of the highly artificial system together with a suggested defence to opponents some weeks before the tournament and make it possible to use such a system. We have not considered such issues in this thesis but they can be added i.e. by setting the reinforcement low if the learnt system does not comply with the rules.

8.7 Results

One of the best evaluations of 77 test-hands resulted in an average score of losing 88 points/deal compared to the optimal result (Fig. 23). Humans at a club level lose less; around 50 point/deal (my estimate; just a simple test was made with one person).

The resulting system (Examples in Fig. 24) can be classified as a natural one. Some new ideas came out like opening 3♠ with 18+ points.

A decision tree for the opening bids is generated by using Ross Quinlan's software See5 (we used the demo version; restricted to 200 cases, randomly selected from the generated bidding system):

```
pts > 15:
:...s > 3: 3S (34/7)
```

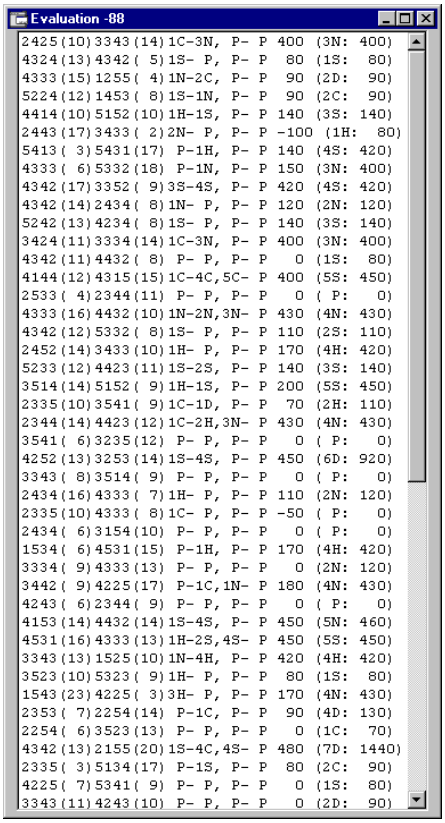


Figure 23: The evaluation of part of the test set. The hands, the learnt bidding, the score and the highest optimal contract (within parentheses)

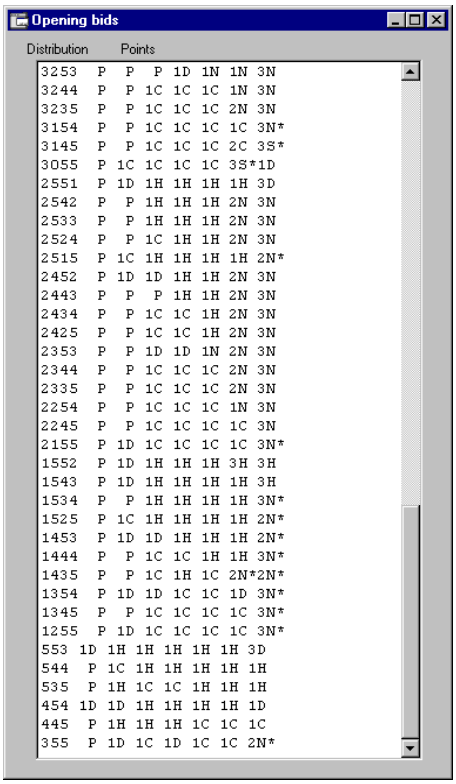


Figure 24: Some of the opening bids. The seven columns denotes increasing strength classes (See table 7 at page 70). An asterisk "Alert" denotes a bid which does not carry a natural or – for the opponents – expected meaning, for example an artificial bid only asking partner to describe his hand.

```
:   s <= 3:
:   :...pts > 18:
:       :...s <= 0: 1H (2/1)
:       :   s > 0: 3N (13/3)
:       pts <= 18:
:       :...h <= 2:
:           :...h <= 1: 2C (2/1)
:           :   h > 1: 1C (2/1)
:           h > 2:
:           :...h <= 4: 2N (6/1)
:               h > 4: 1H (4)
pts <= 15:
:...pts <= 8:
    :...c > 1: pass (45/4)
    :   c <= 1:
    :   :...pts <= 3: pass (4/1)
    :       pts > 3: 1D (7/2)
    pts > 8:
    :...h > 3:
        :...c > 4: 1C (3/1)
        :   c <= 4:
        :   :...s <= 4: 1H (26/1)
        :       s > 4:
        :       :...h <= 4: 1S (5)
        :           h > 4: 1H (4/1)
        h <= 3:
        :...c <= 3:
            :...s > 4: 1S (5)
            :   s <= 4:
            :   :...d <= 4: 1N (3/1)
            :       d > 4:
            :       :...pts <= 13: 1D (7/1)
            :           pts > 13: 1N (2)
            c > 3:
            :...h > 2:
                :...pts <= 13: 1C (4/1)
                :   pts > 13: 1N (3/1)
                h <= 2:
                :...s <= 3: 1C (8)
                    s > 3:
                    :...pts <= 11: 1C (5/1)
                        pts > 11: 1S (6/1)
```

The numbers within parenthesis indicate the number of cases in the data that are mapped to this leaf and the number of them that are classified incorrectly.

It can also be converted into a collection of rules, *a rule set*:

```
Rule 1: (32/1, lift 4.1) pts <= 3
         ->  class pass  [0.941]
Rule 2: (45/4, lift 3.9) pts <= 8 c > 1
         ->  class pass  [0.894]

Rule 3: (7/1, lift 7.4)  pts > 8 pts <= 11 h <= 2 c > 3
         ->  class 1C  [0.778]
Rule 4: (18/4, lift 7.1) pts > 8 pts <= 13 h <= 3 c > 3
         ->  class 1C  [0.750]
Rule 5: (13/3, lift 7.0) pts > 8 pts <= 15 c > 4
         ->  class 1C  [0.733]
Rule 6: (10/5, lift 4.8) pts <= 18 s <= 3 h > 1 h <= 2
         ->  class 1C  [0.500]

Rule 7: (7/1, lift 12.0) pts > 8 pts <= 13 h <= 3 d > 4 c <= 3
         ->  class 1D  [0.778]
Rule 8: (7/2, lift 10.3) pts > 3 pts <= 8 c <= 1
         ->  class 1D  [0.667]

Rule 9: (26/1, lift 5.0) pts > 8 pts <= 15 s <= 4 h > 3 c <= 4
         -> class 1H  [0.929]
Rule 10: (20/1, lift 4.9)   pts > 8 pts <= 15 h > 4
          -> class 1H  [0.909]
Rule 11: (4, lift 4.5)   pts > 15 pts <= 18 s <= 3 h > 4
          -> class 1H  [0.833]
Rule 12: (6/2, lift 3.4)   s <= 0
          -> class 1H  [0.625]

Rule 13: (14/1, lift 8.0) pts > 8 pts <= 15 s > 4 h <= 4
          ->  class 1S  [0.875]
Rule 14: (6/1, lift 6.8)  pts > 11 pts <= 15 s > 3 h <= 2 c > 3
          ->  class 1S  [0.750]
Rule 15: (11/5, lift 4.9) pts > 8 pts <= 15 s > 3 h <= 2 c > 3
          ->  class 1S  [0.538]

Rule 16: (3, lift 22.9)   pts > 13 pts <= 15 s <= 4 h <= 3 c <= 3
          ->  class 1N  [0.800]
Rule 17: (6/2, lift 17.9) pts > 13 pts <= 15 h > 2 h <= 3
          ->  class 1N  [0.625]
Rule 18: (3/1, lift 17.1) pts > 8 pts <= 15 s <= 4 h <= 3
                          d <= 4 c <= 3
          ->  class 1N  [0.600]
```

```
Rule 19: (2/1, lift 50.0) pts > 15 s <= 3 h <= 1
          ->  class 2C  [0.500]

Rule 20: (6/1, lift 21.4) pts > 15 pts <= 18 s <= 3 h > 2 h <= 4
          ->  class 2N  [0.750]

Rule 21: (34/7, lift 5.2) pts > 15 s > 3
          ->  class 3S  [0.778]

Rule 22: (13/3, lift 11.3) pts > 18 s > 0 s <= 3
          ->  class 3N  [0.733]

Default class: pass
```

The statistics (n, lift x) or (n/m, lift x) summarize the performance of the rule; n is the number of training cases covered by the rule and m, if it appears, shows how many of them do not belong to the class predicted by the rule. The rule's accuracy is estimated by the Laplace ratio (n-m+1)/(n+2). The lift x is the result of dividing the rule's estimated accuracy by the relative frequency of the predicted class in the training set. A value between 0 and 1 indicates the confidence with which this prediction is made. Some of the simplified results of this learning are:

- Do not open 1 NT with at least 4 ♡
- Open 3♠ with a strong hand(16+) with at least 4 ♠
- Pass with $0-8$ p
- Open with 9+p
- Open with ♣ even if having (more) ♡
- 1♢ shows $9-12$ with 5♢ (or 3-8 with 0-1 ♣)
- 1♡ shows $9-18$ with 4♡
- 1♠ shows $9-15$ with 5♠ or $9-15$ with 4 ♠ and 4 ♣
- 1 NT is $9-15$ balanced with less than 4 ♡ or $13-15$ with ♢
- 2♣ is a strong (16+) hand with the minors
- 2NT shows 16-18 with 3-4♡ but less than 4 ♠

Some examples of the rest of the bidding are presented in Fig. 25 and Fig. 26. Other runs resulted in systems where the opener either had a strong hand with at least four cards in the suit bid or a weak hand with shortage in the bid suit. Responder had to force the bidding by a jump response with a strong hand because the weak hand just passed on a normal (1 over 1 or, if necessary, 2 over 1) bid.

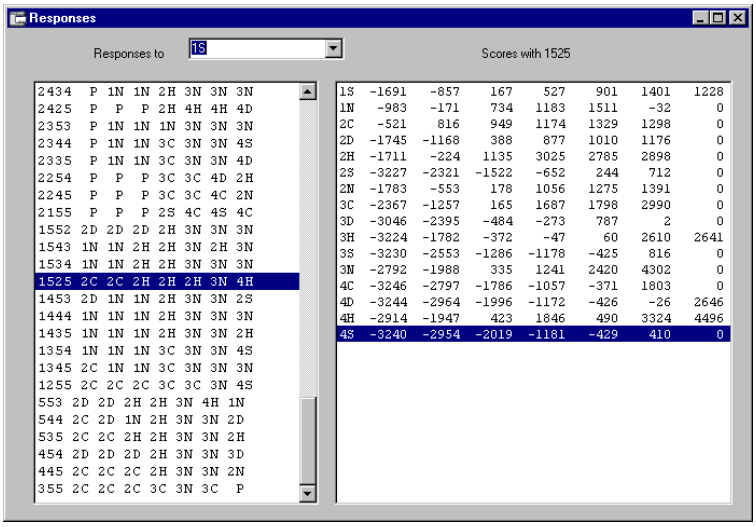


Figure 25: Some of the responses to the opening bid of 1 spade. The average score (scaled *10 to avoid round-off errors in the integer arithmetic) by selecting different bids are shown to the right for the particular hand with 1 spade, 5 hearts, 2 diamonds and 5 clubs and the 7 different strengths. The top 1s-line means pass - leave the contract at 1s.

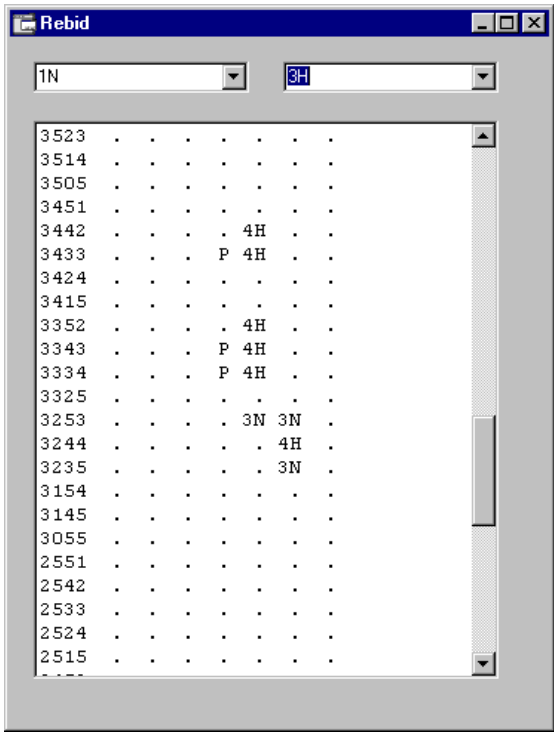


Figure 26: Rebids with a hand opened 1 NT and which got an answer of 3 ♡. When having a strong hand with ♡ support, 4 ♡ is bid but P or 3NT otherwise. The dots indicate hands normally not opened 1 NT.

Class	Strength	Fuzzy
1	$0-6$	very weak
2	$7-9$	weak
3	$10-11$	medium
4	$12-13$	weak opening
5	$14-16$	opening
6	$17-19$	strong
7	$20-$	very strong

Table 7: Classifying the strengths.

8.7.1 Tested tricks

A bridge hand consists of thirteen cards but we represent it as a four-digit number for the distribution and a two-digit number for the strength. Ex. 4135,12 means a hand with four spades, one heart, three diamonds and five clubs. 12 points is a strength slightly above average (the average being 10). Strength is further classified into seven ranges with limits set according to frequency in a heuristic way as shown in table 7.

Learning rates Q-learning with a finite MDP is guaranteed to converge to the optimal policy if all states are visited infinitely often and all actions tested while reducing the learning-rate slowly enough. Our problem here is that this is not a MDP because each agent is an environment for the other and has to learn the environment changes (not necessary in a MDP). The learning rate is in any case still a very important parameter to modify to this approximate MDP. Each bidding position (of the four; opener, responder, rebidder, second rebid) has its own learning rate. A high learning rate means that the learner quickly adapts to the new situation but will be poor on generalization (averaging) because it forgets so fast. A low learning rate has good averaging performance but it takes many trials before it adapts and old cases are remembered a long time after the other bid-positions maybe have changed its way of behavior. We have tested the following ways:

- Slowly decreasing by a factor
- Alternating the players rate (one high, the others low)
- Decrease one, increase some of the others
- Select a rate by random
- Sine-wave in a hope for inducing "learning cycles"

We have not tested some of the Automatic adjustment methods like Delta-bar-delta or Temporal Coherence [BS99] that tries to adjust the learning rates during the learning depending on the results.

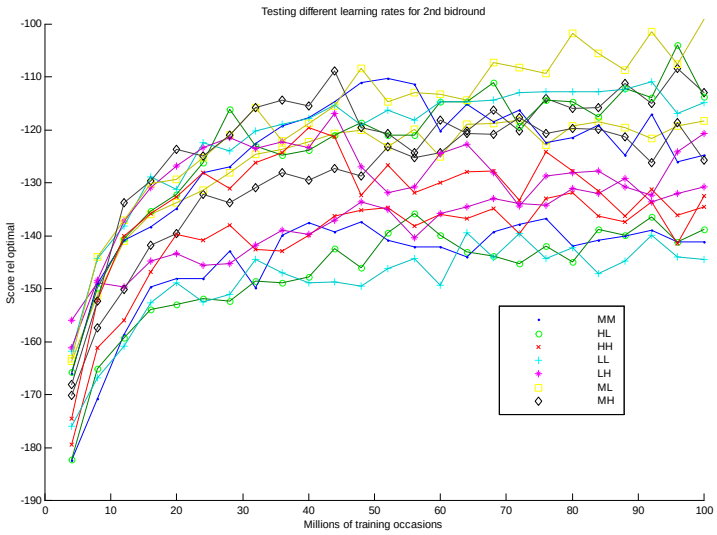


Figure 27: Testing different learning rates. East hand reduces its learning rate and West increases. H denotes High, M medium and L low change-rate. The score was the difference to the optimal for a test set (not used as a reward-value).

A test of letting the first hand (East) start with a high learning rate and then gradually decrease it while increasing the next hand was done in Fig. 27. The following values were used:

$$\begin{aligned} lrE &= 256 * k_e^n, n = 1..25 \\ lrW &= 0.01 * k_w^n, n = 1..25 \end{aligned}$$

and the k:s had the following value in the different cases:

Label	E	W
H	0.98	1.5
M	0.92	1.38
L	0.8	1.1

The case E Medium and W Low, although not significant, seems to work best. The presented values are averaged over ten runs but the single runs overlap considerably. The learning rates were represented as parts of 1024 which makes calculations quick; $0.01 * Q$ is calculated via $(102 * Q) >> 10$, where$>>$ 10 denotes shift right 10 positions (divide by 1024) which is a much quicker instruction for this computer than "multiply by 0.1".

Exploration We have tried several variants of how to perform exploration;

-*ε-greedy* selects the best (greediest) action (and randomly among them if several are equal) except when a random number is below a small number ε. Select any action in the latter case.

-*Boltzmann* methods slowly decrease the "temperature" T to reduce exploration as a function of time. The T is reduced with a factor at each time step. The probability p of selecting action a is set to:

$$p(a) = \frac{e^{Q_a/T}}{\sum e^{Q_i/T}} = \frac{e^k e^{Q_a/T}}{\sum e^k e^{Q_i/T}} = \frac{e^{k+Q_a/T}}{\sum e^{k+Q_i/T}}, \quad \text{set } k = -\max Q_i/T \tag{8}$$

where Q_i is the Q-value for action i and Q_a for the action a. The index for the sums go through all possible actions. The k is introduced to eliminate numerical problems because $k+Q/T$ has to be in the range -4 to 14 in our implementation (parts of a million integer scale). With a floating-point implementation it can be about -300 to 300, also a limited range. This method is kind of "one-shot" because no more exploration will be made when the temperature has dropped down.

- *"best choice"*: Set probability of selection proportional to Q-value, a "Roulette wheel" (linearly shifted to make all Q-values positive) approach. This does not seem to do better than the simple ε-greedy exploration, a fact that is also indicated in the literature[SB98].

Pre-training We tried to manually enter Q-values for a natural system for the opening bids and some responses but they will "unlearn" if they are not coordinated with learning rates by setting them to zero for the opener until the responder stabilizes.

The initial values of the Q-tables are important because they control the balance between exploration and exploitation, at least in the transient start-up phase.

Reinforcement signal What value of the reward r should the evaluator give the learners?

a) Calculate the difference between the actual score to the score achievable on this combination of hands

$$r = score - achievable \tag{9}$$

This works badly due to high exploration. Assuming a start value of Q with zero "optimistic start", almost all tried bids will start with negative scores (strange contracts) making them worse than hitherto untested alternatives. This will lead to a huge, time consuming exploration phase of all alternatives.

b) Difference as above but added 200 points.

$$r = score - achievable + 200 \tag{10}$$

A contract not worse than -200 is denoted better than the untested cases. A "Pass Always System" evolved, guaranteeing a score of 200. Maybe a long run will improve this policy?

c) Absolute score. Simply reinforce the learner by the actual score.

$$r = score * 10 \tag{11}$$

This method worked best, at least in our short (one to two day) runs. The score was scaled with 10 to avoid round-off errors when multiplied by the learning rates.

d) Heuristic reinforcement.

$$r' = r - 100 * isToHighBid \tag{12}$$

To avoid high-level bidding with insufficient strength cards there was an extra negative reinforcement on such bids that were above the optimal contract.

Discount/Decay How far-sighted should bidders be? By selecting different values for γ (Eq. 1), different behaviors were obtained. A low value makes learners prefer a short bidding sequence and result in guessing into 3NT more often than with a high-valued γ. Maybe a value higher than the normal limit of one will encourage a more informative, long sequence but this was not tested. Eligibility traces were not used.

8.7.2 Example set

We allowed max 5 cards per player in any suit and requiredat least 14 h.c.p. in common: if less the opponents will probably enter the bidding and our assumption of an undisturbed auction between two players is not relevant.

By limits like these, the learnt system will obviously change due to bias effects. Suppose we say that the common strength is at least 25 h.c.p., then the opener with 0 h.c.p. will know, from experience, that partner has at least 25 and the weak hand can still open the bidding without danger of getting too high.

Regarding On/Off-policy learning; while exploration will degrade the systems behavior, no exploration at all limits the learning abilities severely. We tried off-policy learning but this was not relevant because a hand could be worth a lot if it happens to have a strong partner but on the other hand very little opposite a weak one. The exploration problem is even worse with our cooperating agents because the trust decreases.

8.7.3 Implementation Performance Optimizations

The built-in random number generator `rand()` was replaced by an integer implementation from Bjarne Stroustrup's C++ book[Str97] resulting in a speed improvement (because of integers instead of floating-point calculations) of a

factor five when generating random numbers which is a substantial (about 50% originally) part of the execution time. The following formula was used:

$$\operatorname{rand} x := \operatorname{rand} x * 1103515245 + 12345$$

To generate random numbers in a range like 0..1023 the lowest bits were simply masked out to avoid divisions. The statistical properties might get bad by doing like this (actually not, see page 75) but we decided to go for speed. A maximum of 4 bids in a sequence was allowed to keep the Q-tables small enough. Only pass and the 15 following bids (in a given situation) were allowed. This is normally of little practical importance (no double-jump bids above trumps were allowed; however such bids are sometimes used by real players for conventional purposes, e.g. to show shortage "splinter").

The examples are precalculated and stored to reduce deal-generating time. Four million examples(deals) are generated and stored in memory for quick reference. A random sample was selected for learning. This was carried out via a separate file; (this is a bad idea because someone who wants to run the program has to download this large file). It would had been better to generate the deals and then reuse them. The Boltzmann equation was stored as an integer table with parameters in the range -40/10 to 140/10 and soft limits. The bridge scores were multiplied with ten to get an extra "decimal" in the range of a 16-bit integer. We tried to be aware of quantization effects with learning rates, scores etc.

The program was running multi-threaded under windows NT 4.0. The user interface was implemented as a separate thread making it possible to adjust parameters and see the results while running/learning. No special precautions were taken to avoid simultaneous changes to the same parameter. The programming environment used was the RAD[19] Borland C++ Builder v4.0 Pro. A portable Dell CPiA, 366 MHz with 256 MByte of memory was used for the calculation. A training example took about 5 microseconds to perform. One deal took about 10 microseconds to prepare but was stored for further reference multiple times. Total memory required was 186 MBytes.

8.7.4 Deal statistics

To see how many of the actual deals are used in the learning due to the limitations above, a generating session was run and the shape statistics were collected in Table 8.7.4.

Random number generators are sometimes tested by the "Poker-Test", here we invent a bridge-test by simulating deals with different shapes and compare them to theoretical values.

The theoretical values were calculated as follows:

$$\frac{\binom{13}{S}\binom{13}{H}\binom{13}{D}\binom{13}{C}}{Totalhands} \cdot permutations \tag{13}$$

[19] Rapid Application Development; speeds up MS-Windows programming.

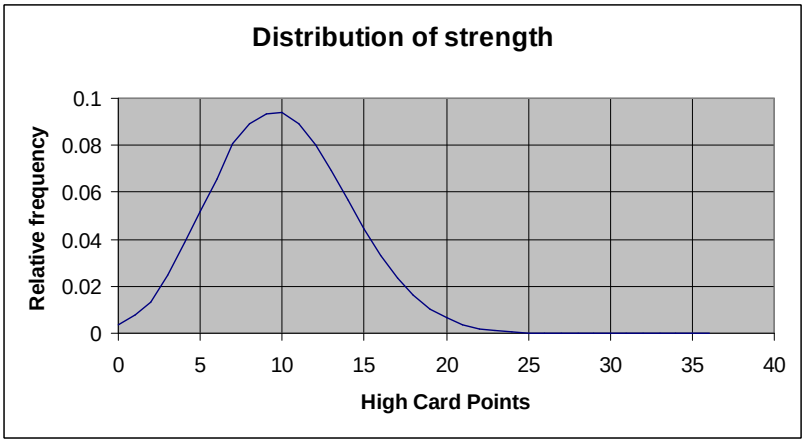


Figure 28: The distribution of strength (hcp). Statistics from the 10 million hands with at most five cards in any suit.

The theoretical values are close to the simulated ones so the random number generator is probably adequate. The total number of hands is about $6 \cdot 10^{11}$ as shown in section 8.5.

The 252 most common shapes with accumulated frequencies is found in Table 8 although only the 104 with maximum 5 card suits were used here.

8.7.5 Problems encountered

The absolute values of an integer is supposed to be an non-negative integer but the result of calling `abs()` with the largest negative integer results in a negative value due to the two-complement representation commonly used. This is because of asymmetries in the range of numbers; a `signed char` may represent -128 to +127 and there is simply no +128 when trying to evaluate `abs(-128)`. This error appeared in generating the assumed-to-be-positive random numbers and appears on average every 4 billionth call (ca. 4 hours run), resulting in a crash as a consequence and was obviously very hard to find.

Another thing was the evaluation of the Boltzmann soft-max. The exponential function e^x, requires the parameter x to be in about the range of -4 to 14. Our x was evaluated as a quote of an Q-value and a "temperature" T, resulting in poor performance when close to these borders. This problem was avoided by simply scaling the total expression into a proper range and having soft evaluations at the border values.

Shape	Rel.freq(%)/perm	Permutations	Total(%)	Theoretical
4333	2.6	4	10.56	10.536
4432	1.7	12	21.55	21.551
5332	1.3	12	15.50	15.517
5422	0.88	12	10.58	10.580
4441	0.75	4	2.997	2.993
5431	0.54	24	12.91	12.931
6322	0.47	12	5.639	5.642
6331	0.28	12	3.445	3.448
5521	0.26	12	3.185	3.174
6421	0.19	24	4.713	4.702
7222	0.12	4	0.514	0.513
5440	0.10	12	1.245	1.243
7321	0.078	24	1.804	1.881
5530	0.074	12	0.894	0.895
6511	0.059	12	0.708	0.705
6430	0.054	24	1.323	1.326
7411	0.032	12	0.392	0.392
6520	0.026	24	0.649	0.651
Sum		252 (of 560)	98.60	98.68

Table 8: The shapes of the simulated hands.

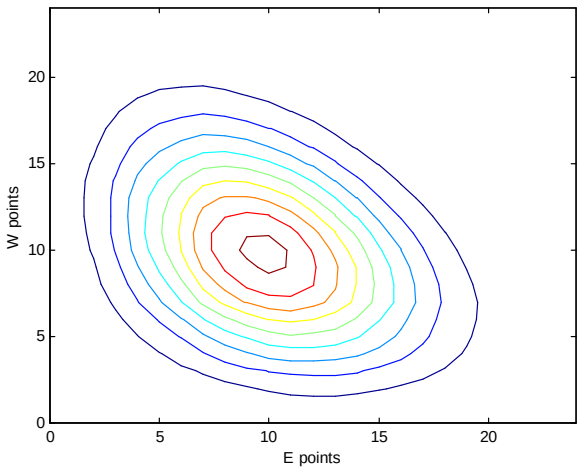


Figure 29: A contour plot of how the strength of E disturbs the strength distribution of W. A high point-count in E reduces (of course) the point-count in W.

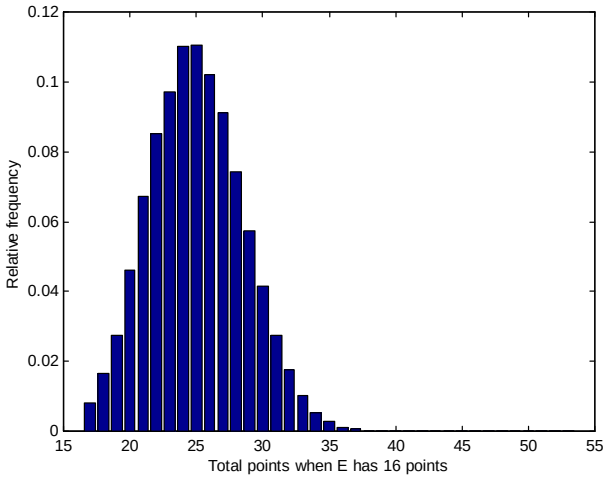


Figure 30: The strength distribution of the combined hands when E has 16 h.c.p. There is a 54% probability they will have at least 25 together.

8.7.6 Q - windup

We need a low learning rate to average different hand combinations and a high learning rate to quickly adapt to a new situation.

A bid that was very bad before might become good if partner begins to "understand" it. The problem is that it takes a lot of exploration before the Q-value rises enough to overcome its very low value, there has to be very many lucky trials before the new understanding is established. A suggestion for solving this problem is to reduce the allowed range for Q-values in an attempt to avoid "wind-up" (control theory term for the summation of the error in a PID-controller) but the idea was not tested in practise.

8.8 Future

An attempt to make the bidding work with 4 hands in a more unlimited version would be very exciting and of good use for the Bridge-playing community. This also adds a new dimension to the bidding with opponents who try to sabotage the bidding and the possibilities to exchange information. With the current approach such an extension would require 10+GB of primary memory and possibly thousands of years in order to learn something useful. The learning and representation have to be improved. One question that might be asked is: Why doesn't the responder take into consideration what hands a particular opening bid can be made with? The answer is: Because a particular bid can be selected by any hand due to the exploration possibility. This will be of no use and if some kind of probability calculus is performed it will also destroy the explo-

ration possibilities. By using the Genetic Approach, a template hand is found for a particular partner and this information can be used because it is constant during the practising.

8.8.1 Genetic learning

A preliminary test was done but with limited results. A better representation and specialized genetic operators have to be found. One possible way of doing this is to have a vector with (linear) indices of all possible distributions and strengths, as well as other parameters. Matching the values in these positions recommends the opening bid. Another approach will be to generate bidding rules like "if you have 5-6 spades and 0-3 hearts and 11-13 h.c.p. bid 1 Spade". The rules have to be generated in a way to make them non-contradictory to avoid difficulties or used in an ordered (ranked) fashion. The design of the basic logic also have to allow for multiple meaning cases like "if 11-13 balanced or 13+ with clubs bid 1 club" to find modern ways of system design.

8.8.2 Locker-Room agreements

By inventing a communication language, or something similar, the development of bidding systems during a training phase can be sped up by letting the bidders discuss meanings and agree on conventions using a more powerful language. This kind of communication is disallowed during real bridge bidding but is possible in "pauses" between boards. Some Meta-rules (like "the bid is to be taken as natural; showing the suit") can be used when non-discussed situations arise or some other learning can be done on both sides by starting with the same, agreed, random number seed.

8.8.3 Multidimensional representation

A 2-dimensional matrix is used in this case with a list of shapes as one axis and the strength-interval as the other. By using a 5 dimensional space with length of each suit on 4 axes and the strength as a fifth (and other things like LTC as sixth etc.), a nearest neighbor approach can be used. One example is a Support Vector machine (SVM) or a tree learning algorithm that reduce the state-space into regions and split or merge them according to results (kd-Q-learning Vollbrecht[Vol99]). The distance function has to be asymmetric because a 5431 hand is closer to a 5422 than to a 4432, weighting by the suit length may be applicable. The dimension (degrees of freedom) is actually one less because of the restriction of the sum into $\spadesuit + \heartsuit + \diamondsuit + \clubsuit = 13$. Instance-based learning can also be tried.

8.8.4 Bid sequences

Sixteen possible bids made four times in a row results in 65536 possible sequences. 104 shapes $\cdot$ 7 strength intervals equals 728 possible "hands" resulting in $728^2 \approx 530$ thousand combinations of two hands. Each sequence has to cover

on average $\frac{728^2}{16^4} \approx 8.1$ combinations but many combinations (about half of them) might be covered by the sequence pass-pass resulting in about 4 combinations in each sequence (except P-P). In each combination it is possible to find out the best final contract (by using the evaluator or some other source like some tournament results). Assign a (maybe set of) sequence(s) ending in the final contract to each of the combination and evaluate the resulting behavior. Modify the assignments with some RL or genetic approach.

Real bridge allows more than 16 bids, i.e. there are 36 bids for a start; pass, 1♣, 1♢, 1♡, 1♠, 1 NT, 2 ♣, ..., 7 NT. There are also a lot of ways to reach a certain contract; P is only reachable by the sequence P-P but 1♣ can be reached in two ways; P-1♣,P and 1♣-P, 1♢ can be reached in four ways (P-1♣, 1♢-P and P-1♢,P and 1♣-1♢, P and 1♢-P) and 3 NT in $2^{15} = 32768$ ways! By letting the opponents enter the bidding with Double, denoted by X, the possibilities are further increased (and decreased if they bid something else, e.g. a suit).

8.8.5 First round defined

By defining the opening bids for all hands (quite an easy task for most bidding systems) the continued bidding might be learnt. Probably the learnt system will be more or less natural and easy for a human being to learn. By making this definition the search state-space is heavily reduced and quicker learning will be achieved. Some programs have databases with full bidding but this is very tedious and cumbersome to enter. A learning module maybe works fine especially when two computer players play together and learn the same things, perhaps even during the bidding when a strange bidding situation occurs.

It is also possible to define some of the bid-situations and let the computer find the rest by learning.

8.8.6 Heuristics/Information utilization

By making the Q-tables available to the partner, he/she can estimate what kind of hands the opener may have and the probability of selecting a bid on a certain hand, or the other way around, given a bid, "what can opener have? / What contracts could be relevant?" This will probably result in trouble because opener may choose any bid on any hand as a exploratory "move" causing this method to make it harder to invent new ways of bidding once settled for a certain path.

8.8.7 High gamma to encourage bidding

By letting the gamma (γ) have values above 1, cooperation will perhaps be encouraged. Normally gamma is not allowed to have values above 1 but this task is an episodic task limiting the expected utility to a finite value. This idea will probably generate longer bidding sequences with more information exchange and possibly end up in a better contract. In real bridge this is counter-productive because opponents will also get more information about the hands. This will

simplify their defence during the bidding and play, by making it easier to find a good lead directing double, preemptive bid, opening lead or defensive play.

8.8.8 Hierarchical learning of concepts

By learning concepts like "balanced hand", "two-suiter" etc. the state space might be reduced and generalization abilities improved. The shorter "Open 1NT with a balanced hand" can be used instead of open 1NT with 4432,4342,4234 etc.

8.8.9 Bayesian Belief Networks

The approach to collect information into a net with probabilities seems good. Suppose we have 4 spades and 14 h.c.p., then a calculation can be made of the probabilities of partners distribution and strength. The likely result with those combinations resulting in the most probable contracts and receiving scores as expected values can be calculated. Learning would involve learning the structure and matching probabilities.

8.8.10 1-Agent learning

This idea is based on the assumption that the first agent just acts as mentioned earlier. The second agent tries to model the first one and anticipate its actions to maximize its reward.

8.8.11 Variations of Q-learning

There are many variations of the Q-learning theme that might work better for this case. Some examples are R-MAX ([BT01]) and average Q-learning ([GS01]). Some comparisons with minimax-Q.are made in [BV01]. Stochastic Game Theory for MARL is discussed in [BV00].

8.8.12 Transforming into a single-agent problem

By finding a reformulation of the problem into a single agent, normal Q-learning could be applied to solve it.

9 Learning bidding systems by using large-scale Q-tables

Some of the ideas from the previous chapter were implemented in a new program called "Reese", after the famous Terence Reese who died in 1996. He was known for his excellent books and simple and logical bidding style. Regarding our bidding program there was a hope to find a super-natural bidding system, but that was not to be. The program requires a quite powerful computer and at least 348 MByte of memory with the described configuration. The multi-tasking Microsoft Windows approach has been abandoned due to portability and other reasons like not being able to do long batch runs.

The experiments can be seen as what can be achieved with these methods in a complex two-learner task and also gives some hints of how to choose learning parameters.

9.1 Introduction

By using the huge database of GIB-results when playing against itself for 717.102 deals, we found a more reasonable evaluative function instead of the table 4 at page 59. One advantage is the greater number of dimensions added to the evaluative function; not only the sum of trumps and points were used but also how these trumps and points were distributed between the two; i.e. a combination of $12+12=24$ can be reasonable for bidding 3 NT but maybe not $23+2=25$[20] h.c.p., other differences such as: if trumps are divided 4-4 or 5-3, (or other shape-specific relations) are also taken into consideration. The database was generated by Matt Ginsberg, using an early version of GIB where neither the declarer or the opponents were so strong therefore; maybe the result is quite reasonable anyhow. A human being plays about 8 deals per hour and an eight hour work-day, five days a week would require 43 years to get a similar experience. Also given the improved computer power and memory available, more shapes can be used and a more detailed point-range classification can be made. We have used the 128 most common shapes which include all hands with at most a six-card suit except for the rare 5440, 5530, 6511, 6430 and 6520 shapes. This covers about 91% of the hands and, neglecting the inter-hand shape influence, $0.91 * 0.91 = 83\%$ of the hand-pairs. The strength was classified into the following 16 intervals (although this is easy to modify):

Class	0	1	2	3	4	5	6	7
h.c.p.	0-3	4-6	7-8	9	10	11	12	13
Class	8	9	10	11	12	13	14	15
h.c.p.	14	15	16-17	18-19	20-21	22-24	25-27	28-37

This leads to $128 * 16 = 2048$ different hands and $2048^2 = 4,194,304$ pairs (not all are possible; i.e. both hands with 25+ h.c.p.); obviously less[21] than

[20] The weak hand often lacks entrances.

[21] Some pairs appears several times.

17% (717,102/4,194,304) of the pairs are available in the database mentioned above.

To reduce memory requirements, learning time and simplify the restriction of learning into allowed bidding systems, the opening bids were defined by the user as a small two page table. Part of the definition of the opening bids in bidsyst.txt is:

```
Shape 0  4  7  9 10 11 12 13 14 15 16 18 20 22 25 28
6421  P           1S                      2S
6412  P           1S                      2S
6331  P 3S 2D        1S                   2S
6322  P 3S 2D        1S                      2S
6313  P 3S 2D        1S                   2S
6241  P           1S                      2S
6232  P 3S 2D        1S                      2S
6223  P 3S 2D        1S                      2S
6214  P           1S                      2S
6142  P           1S                      2S
6133  P 3S 2D        1S                   2S
6124  P           1S                      2S
5521  P           1S                      2S
5512  P           1S                      2S
5431  P              1S                      2S
5422  P                 1S                   2S
5413  P              1S                      2S
5341  P              1S                      2S
5332  P                 1S        1N    1S 2C 2N 2S
```

It can be represented as a decision tree or ruleset but this introduces errors (about 2% with the default settings in the See5 SW), is larger (five and ten pages respectively) and considerably more complex to read. The bidsyst.txt can be rearranged (preferably with a program) from this matrix representation into an input file with examples which look like this (points, number of Spades, Hearts, Diamonds, Clubs, Opening-bid):

```
0,6,4,2,1,pass
1,6,4,2,1,pass
2,6,4,2,1,pass
3,6,4,2,1,pass
4,6,4,2,1,pass
...
```

Each strength 0..37 is used to generate an example, resulting in $128 * 38 = 4864$ examples.

The See5, inductive-logic, software (available via http://www.rulequest.com) generates the following decision tree (only partly shown):

```
See5 [Release 1.13]     Wed Mar 14 15:57:11 2001

    Options:
        Generating rules
        Fuzzy thresholds
        Pruning confidence level 50%

Read 4864 cases (5 attributes) from openings.data

Decision tree:

pts in [20-38]:
:...s in [5-6]:
:   :...pts in [25-38]: 2S (390)
:   :   pts in [0-24]:
:   :   :...pts in [0-21]:
:   :       :...s in [1-5]:
:   :       :   :...d in [5-6]: 2S (4)
:   :       :   :   d in [1-4]:
:   :       :   :   :...h in [5-6]: 2S (4)
:   :       :   :       h in [1-4]:
:   :       :   :       :...h in [1-2]:
:   :       :   :           :...d in [1-2]:
:   :       :   :           :   :...d = 1: 2S (2)
:   :       :   :           :   :   d in [2-6]:
:   :       :   :           :   :   :...h = 1: 2S (2)
:   :       :   :           :   :       h in [2-6]: 1S (2)
:   :       :   :           :   d in [3-6]:
:   :       :   :           :   :...h = 1: 1S (4)
:   :       :   :           :       h in [2-6]:
:   :       :   :           :       :...d in [1-3]: 2C (2)
:   :       :   :           :           d in [4-6]: 1S (2)
:   :       :   :           h in [3-6]:
:   :       :   :           :...h in [4-6]: 1S (6)
... (4 more pages)
```

And the following set of rules (terms defined on page 8.7, my comments to the right of *):

```
Rule 1: (512, lift 3.9) pts in [0-3]
->  class pass  [0.998] * Pass with a very weak hand

Rule 2: (132/2, lift 3.8) pts in [0-11] s in [1-3] h in [1-4]
        d in [5-6]  d in [1-5]
->  class pass  [0.978] * Pass even up to average and 5 diamonds
```

```
Rule 3: (39, lift 3.8) pts in [0-12] s in [4-6] s in [1-4]
        h in [2-6] h in [1-2]    d in [1-3]
->  class pass  [0.976] * Pass with up to average and 4 spades

...

Rule 47: (20, lift 15.6) pts in [10-38] pts in [0-19]
                         s in [5-6] d in [5-6]
->  class 1S  [0.955] * open 1S with 10-19 hcp and 5-6 s and 5-6 d

Rule 48: (20, lift 15.6) pts in [10-38] pts in [0-19]
                         s in [5-6] h in [5-6]
->  class 1S  [0.955] * or hearts

Rule 49: (120/6, lift 15.4) pts in [10-38] pts in [0-19] s = 6
->  class 1S  [0.943]
...

Rule 128: (9, lift 245.7) pts in [4-38] pts in [0-6] s = 6
        d in [3-6] d in [1-3]
->  class 3S  [0.909] * Preempt 3S with 4-6 hcp,
                       6 s and 3 d (no 6430 allowed)

Rule 129: (26, lift 9.5) pts in [25-38] s in [3-6] s in [1-3]
        h in [5-6] h in [1-5] d in [2-6] d in [1-3]
->  class 3N  [0.964]
```

The program uses the original bidsyst.txt and it is doubtful if the tree or the rules can be useful for a human reader. At least some of the conditions should be joined to be more readable like:

```
pts in [10-38] pts in [0-19] s = 6 d in [3-6] d in [1-3]
```

and

```
h = 1  h in [1-5]
```

into
10-19 hcp 6x3x
and
1 h
or, even,
10-19 hcp, 6133

A variation of the most common system in Sweden is used as an example and defines openings at the one-level as four-card suits and 1 NT as (semi)balanced with $15-17$ high card points (h.c.p.). 2 clubs shows a strong hand with clubs or a balanced hand with $20-21$ or 28+ strength.

No opponents were taken into account except for the specified opening bids which included preemptive bids like 2 diamonds (showing a strong hand with diamonds or a weak hand with a six-card major "multi") and 3 in a suit showing a weak hand with (at least) a six-card suit.

The database contains a vector with the actual number of tricks taken by all hands as declarer and in any suit or notrump. We have assumed East to be declarer in all cases due to performance reasons. The maximum number of bids in a row was set to four (a pass was automatically added as a fifth bid if applicable) and the number of possible bids was limited to sixteen. The program was object-orientedly coded in the high performance language C++ and the library supplied random function was used. The exception mechanism was used to make it possible to catch some run-time errors (not range-errors) and report them to the standard error stream.

Learning rates were encoded as integers and shifted to the right ten steps, effectively scaling them down with a factor of 1024. Exploration rates were integer coded in per mille. A gamma-value of one was used and the initial values of the Q-tables for East and West respectively was a command line argument. The examples were separated in a training and a test set and training examples were selected in random order to avoid biasing effects. The desired number of examples were selected in order from the database except for unrepresentable uncommon shapes which were simply skipped. Exploration was not used during test-evaluations but used during training evaluations.

The score was calculated from the number of tricks taken and final contract. The vulnerability was "none" and dealer was always East (corresponding to board number $14 + n * 32$ in tournament play). Undoubled part-scores, games, Small slams and Grand slams were calculated and contracts going more than two down were assumed to be doubled. Sometimes more than 13 down (doubled..) was achieved by bidding into "funny" contract like 8 clubs or 9 spades[22] but these contracts were automatically avoided (due to the bad results..) after some training.

The Q-learning reinforcement rule

$$q := q + lr * (score - q) \tag{14}$$

was used for all four bidding situations (were applicable) and there was a learner in each of the implicitly given states; each hand-type combined with the bid-situation was a state. q is the average resulting score having this hand and following the policy, lr is the learning rate and $score$ is the resulting score if following the policy. Each hand-type had its own q-value for each bidding situation and bid. The bid with the highest q was selected in a particular situation unless exploration was actual on which any bid was selected with

[22] we allowed this to avoid expensive checks

equal probability. If several bids had the same q-value one of them was selected at random.

In practice this meant one out of 2048 q-openers were combined with another of 2048 q-responders and they had to learn together with their partners repeated multiple times depending on the database examples; hence multi-agent reinforcement learning.

9.2 Special properties

Because of the specified opening bid for each East hand there was a strength limitation dependent of the shape and opening bid. This can be utilized by having a straight scale minimum_opening_strength..minimum_opening_strength+15 instead of the strength classes but implies the restriction that an opening bid with a certain shape has to be limited to a range of sixteen points. This is often the case. By doing this we introduced higher precision in the openers rebid without consuming more memory. We have not limited the responder's possibilities of selecting bids except for the maximum deviation of 15 bid-steps above the opener's bid; 1 NT - 4 NT is allowed but not 1 NT - 5 C. If the responder's bids were defined, less memory would have been consumed and the learnt abilities would have been increased but some of the interesting results from these learning tasks is just which response bid to select on a particular hand opposing a particular opening-bid and normally no restrictions are applied from a rule point-of-view.

If a large number (one hundred million) of training steps are used during each epoch, a 32-bit long integer is not enough to sum all scores correctly if the average score is above 21; something a good bidder should break easily. Borland C++ V4 compiler supplies a 64 bit integer (__int64) which was used for statistical purposes and can be changed to a standard long (taking the risk of wrap-around or by reducing the epoch-length) or a double-precision floating-point sum (sacrificing speed).

The examples were binary stored in Motorola order in the GIB-database file and the bytes had to be re-ordered by the following formula to be useful on an Intel-x86-based machine:

```
fread(suitin, 4, 4, infile);
for(int j = 0; j < 4; j++)
   suit[j]=(( suitin[j] & 0xFF000000) >> 24) +
            (((suitin[j] & 0x00FF0000) >> 16) << 8) +
            (((suitin[j] & 0x0000FF00) >> 8) << 16) +
            (((suitin[j] & 0x000000FF)) << 24);
```

9.3 Experiments

We tried to find the best possible bidding system by making three experiments; first a survey run with different parameters, a second survey with some new, better, parameter values and then a longer run with the parameters estimated

to be best by the human observer. The parameters to be selected were the learning rate (common for both hands), exploration rate (also common) and initial values for the Q-tables.

In the first run we used learning rates 5, 10 and 20, exploration rate 5, 10 and 20 and Q-values -100 for east and -200, 0 and +200 for west. Each run was repeated three times with different random number seeds (13579, 5799 and 8642 respectively). This resulted in a total of $3*3*3*3 = 81$ cases. Ten million training examples were selected in each epoch and 200 epochs were run in each case. Each case took about 41 minutes and the total CPU-time was about 55 hours on a 1.5 GHz Intel Pentium IV running Windows 2000 and equipped with 512 MByte of memory. Each training session required one microsecond including evaluation.

Every 99th epoch, a bidding system was emitted into two files; responses.txt (115 kByte) and rebid.txt (525 kByte), the second response was not stored. The generation of the rebid-file took the opening restrictions into consideration to reduce the size. A log-file (6 kByte) with the time, average training-sum and average test-sum was generated and the standard output was redirected into a file (1.4 MByte) with the actual biddings from all the test-cases for future reference. About $2MB * 81 = 162MB$ disk space was used.

After this, a second run was made with 200 epochs with the learning rate and exploration fixed but three different initial strategies were used. Finally two long runs with 1000 epochs each, making it possible to evaluate the resulting bidding system, were made.

550.000 training examples and 200 test examples (positioned from 710.000 and forward) were used. 109.940 (16.7%) were skipped, reasonably close to the simplified 17% assumption mentioned above.

9.4 Results

Some tests were run with different combination of parameters.

9.4.1 Selecting learning rate, exploration and initial Q-value

The first run was made with a q_0 of 0. The results are plotted in Fig 31 and in Table 9. It seems as if the learning rate does not have a huge importance but the exploration rate has. The highest used value for both of them resulted in the best performance; an average value of 63.6 ± 3.1 with a learning rate of 0.019 and an exploration of 0.020, denoting a 20% risk of selecting a random bid. The training and test curves seems to follow each other but at a different level. The test curve is always below the training curve although no exploration is used. The training curve is also more smooth due to the large number of examples (10 million). The experiments were rerun with a q_0 of +200, meaning an untested alternative is worth 200 points. Because most bridge-results are in the -100..140 region this is the optimistic approach resulting in a (more) exhaustive search. The behavior in this case is similar to the behavior in the q_0 zero-case.

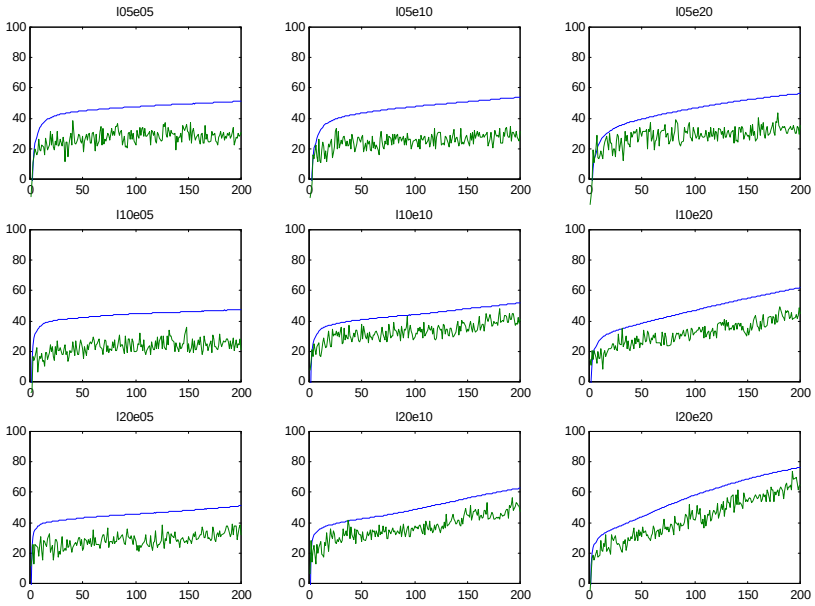


Figure 31: With $q_0 = 0$ ($q_0 = +200$ is similar). The smooth curve is the evaluation of the training set and the rugged curve the test set. 200 epochs (right end), each 10 miljon tries, were run, repeated three times, and averaged into these curves of the achieved score.

	$\varepsilon = \mathbf{0.005}$	$\varepsilon = \mathbf{0.010}$	$\varepsilon = \mathbf{0.020}$
$\mathbf{Lr = 5/1024}$	27.2 ± 8.7	27.6 ± 0.2	30.7 ± 9.9
$\mathbf{Lr = 10/1024}$	24.9 ± 1.8	40.4 ± 6.1	41.7 ± 3.7
$\mathbf{Lr = 20/1024}$	33.8 ± 3.6	50.1 ± 10.0	63.6 ± 3.1

Table 9: The average and standard deviation of the last ten test evaluations, averaged over three runs. q0 is zero.

	$\varepsilon = \mathbf{0.005}$	$\varepsilon = \mathbf{0.010}$	$\varepsilon = \mathbf{0.020}$
$\mathbf{Lr = 5/1024}$	29.3 ± 5.1	36.7 ± 13.6	51.4 ± 6.9
$\mathbf{Lr = 10/1024}$	28.4 ± 5.6	40.9 ± 6.2	54.6 ± 3.2
$\mathbf{Lr = 20/1024}$	27.2 ± 2.4	48.3 ± 8.3	68.7 ± 4.9

Table 10: The average of the last ten test evaluations, averaged over three runs. q0 is -200.

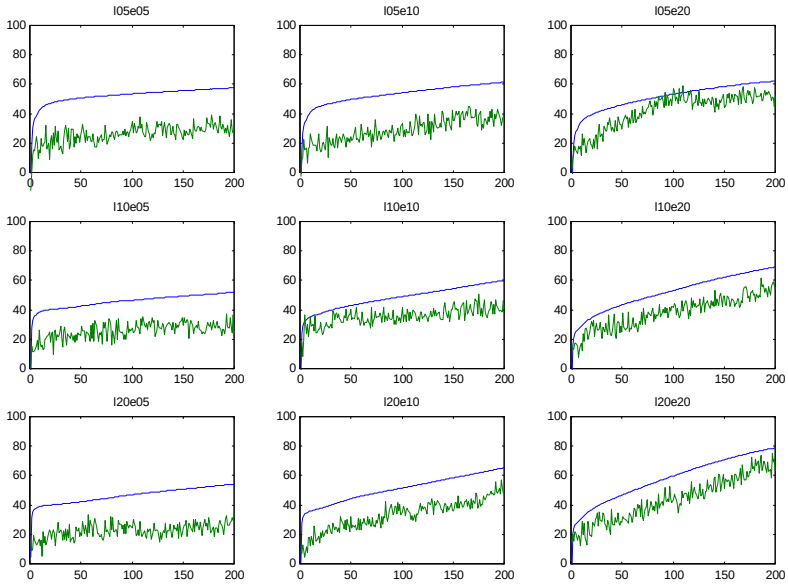


Figure 32: Runs with $q_0 = -200$. The smooth curve is for the training set and the rugged curve the test-set. X-axis is epochs and y is the score.

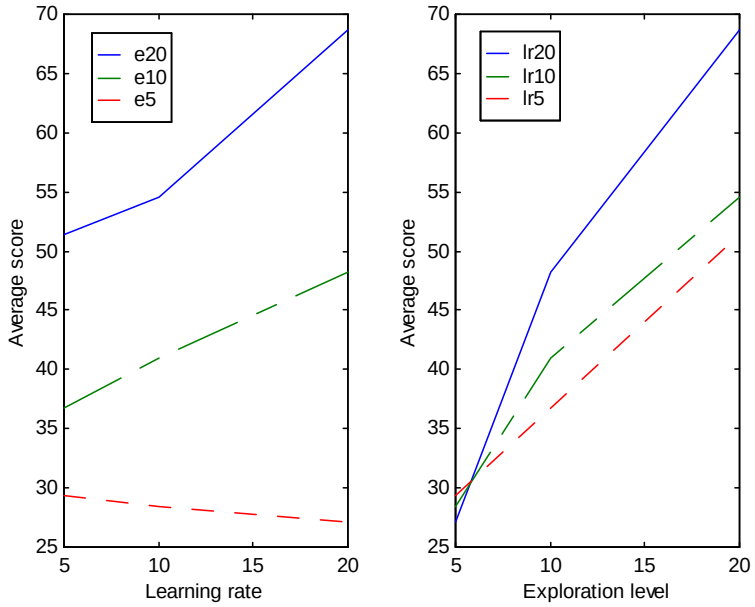


Figure 33: A high vaule of the exploration and the learning rate improves fast learning. High exploration seems to be most important. q_0 is -200.

A third run with $q_0 = -200$ was made with better results (Fig 32, table 10). This value (-200) resulted in less search of alternatives because a reasonable result like -100 was preferred to untested alternatives. This probably leads (guides) the learning through a kind of needle's eye into reasonable bidding systems and contracts. The best result (68.7 ± 4.9) was obtained with the highest parameter values; $Learning\_rate = 20/1024 \approx 0.019$ and $\varepsilon = 0.020$ as usual. The table is also graphically illustrated in Figure 33.

A rerun with a higher learning rate was carried out and shown in Figure 34. The plot shows the average of three runs. Now the best results were, instead, obtained with $q_0 = 200$. Obviously there is a combination effect on these parameters where high learning combines best with high q_0 and vice versa.

A long, single run with a lower learning rate but high exploration is shown in Figure 35. The levels seem, more or less, to stabilize around 100.

The single learner Q-learning theory says that convergence is assured if the learning rate is reduced slowly enough. An attempt to improve the average score is made by reducing the learning rate with a decay factor. In our cases, the learning seems to drop off at about epoch 150. Two experiments were run with the learning rate set to 20/1024 at this point; one with slow decay (Figure 36) and one with a high rate of decay (Figure 37). There are jumps in the performance and if viewed in detail those jumps are positioned according to

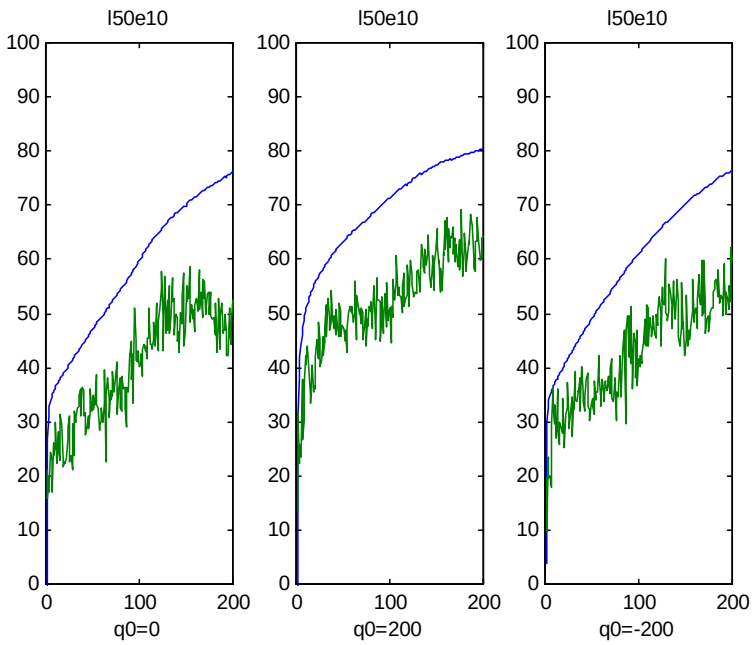


Figure 34: A rerun with a higher learning rate.

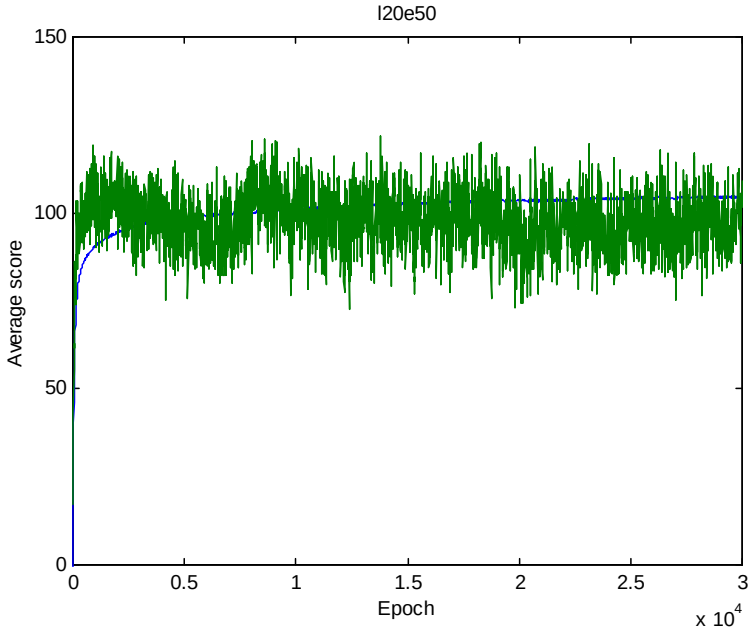


Figure 35: A 5-day CPU run of learning. Lr is 20/1024 , $\varepsilon = 0.050$ and $q_0 = -200$.The maximum training case has a score of 104.89, max test-score is 121.7 and average on the last ten is 94.65.

Epoch	Training-level	Lr	Difference
849	84.7	6/1024	170.7
951	83.4	5/1024	204.8
1074	79.1	4/1024	256
1221	67.3	3/1024	341.3
1413	45.7	2/1024	512
1683	35.6	1/1024	1024

Table 11: Integer round-off error

table 11 in the slow-decay case. When the learning rate steps down to small values, the required difference between score and q-value has to be at least one, when scaled with the learning rate according to Equation 14. The first noticeable effects appear when the learning rate is $6/1024 \approx 0.006$ and the required difference is then $1024/6 \approx 170.7$. In bridge terms this means that the learner is not able to see the difference between part-scores and concentrates on the large contracts like games and slams. When learning rate gets below 3/1024, the game-contract abilities are also lost. This problem with learning could be avoided or, at least, delayed by using floating-point representation but this consumes more memory (4 times in this case; $384 * 4 = 1536$ MByte) and requires more computing time. It can also be postponed by scaling up the score into a reasonable range. During early training scores like -3200 may appear but when bidding becomes reasonable such situations are rare (mostly during exploration). A normal score-range is then between -1400 and +1550, making a scale-factor of $32.767/1550 \approx 21$ applicable, allowing for lower learning rates. Thresholding mechanisms have to be added to avoid wrap-around.

Some typical examples of bidding from a reasonably good run are presented below. The parameters used were Lr=0.02 and exploration = 0.05 with q0 set to -200. This resulted in an average score of 87 on the training set and 100 on the test set after 562 epochs of training.

```
board East Hp West Hp  E  W  E  W   Score  My comment

  2. 2614 16 2353 15 1H  3N 4N 7H  1510 (Good but strange
  5. 2434  9 5323 14  P  1S 2C 2H   110 how 7H was found)
  6. 4522 10 1255 13  P  1C 1S 2C   110
  7. 3145  7 2434 15  P  1H 1N  P    90
 13. 1543  9 5332 16  P  1S 2C 3N   400
 16. 3514 12 3253 14 1H  3N  P      430
 23. 4225  9 5323 11  P   P           0 (missed part-score in sp)
 27. 4243  2 2641 22  P  3N  P     -100 (4H better)
 28. 2434 12 4243 12 1H  3N  P      -50
 29. 3613 13 3262 15 1H  3N  P      520 (missed a slam)
 31. 2614  4 5251  4 3H   P        -800 (opening bids defined
                                        by bidding system)
```

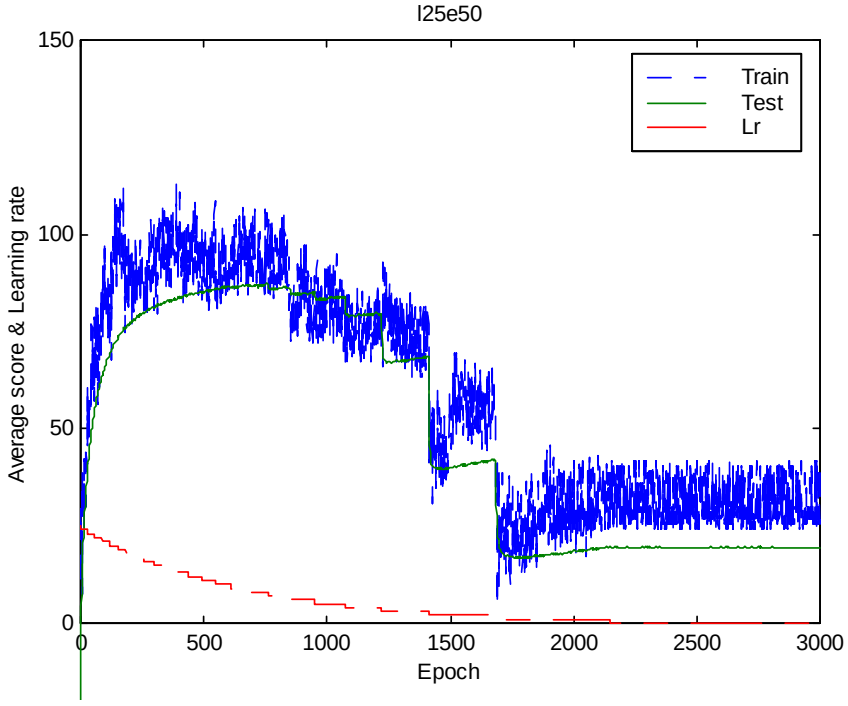


Figure 36: The scores are actually decreasing step-wise when the learning rate reduces.

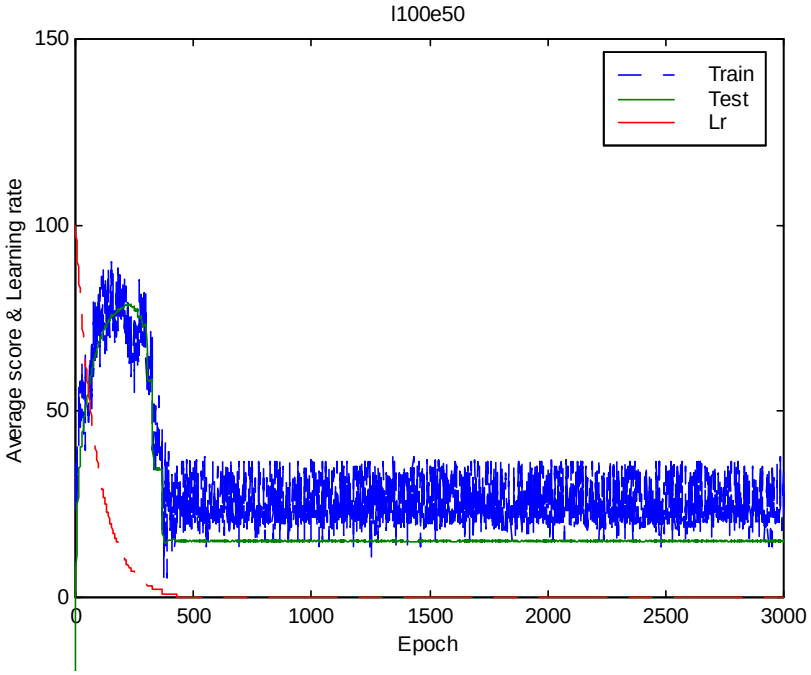


Figure 37: With a high decay of the learning rate, the performance reduces drastically.

```
 33. 5134  6 3622 11  P  1H 1N 2H   -50 (1N with 5 S?)
 34. 3514 11 4135 13  P  1C 1N  P   150
 37. 2353 13 4324  9 1D   P          70 (P with 9 hcp?)
 42. 3334 10 5512 13  P  1H 1S 4S   480 (smart bidding)
 52. 2623 10 2236  7 2D  2H  P      110 (2D shows weak H/S or
                           strong with D, 2H is a good bid)
 53. 2254 13 2632 10 1D  3N  P      400 (practical bridge,
                                         4H probably safer)
 54. 4342  2 3514 11  P  1C 1D 1H   -50
 56. 3163 12 5422 15 1D  3N  P      430 (4S safer?)
 60. 4144  0 3325 17  P  1N 2H 2S  -100 (2H with 0 hcp and 1 h?)
 64. 4441  9 5431 11  P  1S 1N 2S   110
 65. 3325  9 3244 16  P  1D 1N 3N   400

 73. 4252  8 1633 10  P  1H 1S 2H   140
 76. 2551  7 4153 16  P  1D 2D 2S   110
 77. 4324 18 3352 10 1C  1N  P      180 (P with 18?)
 78. 3154 13 4135  9 1D   P         130 (often P with avg.
                                         strength and support)
 79. 3163 17 1525 12 1D  3N  P      490 (missed 6D)
 80. 2542 13 6133 15 1H  3N  P      490 (often bids 3 NT with
                                         5-card majors)
 82. 1426 13 6241  9 1C  1N  P       90
 83. 2533 12 3433 12 1H  4H  P      420
 84. 3244 13 5332 13 1C  3N  P      460
 85. 2362 10 4432 15 3D   P         130 (found to P with 15)
 87. 2344  7 3136 18  P  2S 2N 3N   430
 88. 2434 11 1642 12  P  1H 4C 4H   450
 89. 4252 10 1444 15  P  1H 1S 3N   400
 90. 3253 12 4351 12  P  1D 1S  P   140
 91. 3433 20 4351  9 2C  3N  P      460
 95. 1363 13 3244 15 1D  3N  P      460
 97. 3352 16 2344 15 1N  3N  P      520 (to strong hand for 3N?)
 99. 2524  9 5422 12  P  1H  P      170

100. 1642 13 2263 10 1H  2D 2S 2N  -100 (what is 2S?)
102. 4423 10 3352 17  P  1N 2D 3N   460
105. 2614 14 4144 13 1H  3N  P      460
107. 5431  7 3343 14  P  1D 2H 2N   120
109. 4333 15 3415  2 1N  2C  P      -50
110. 1255 18 4432  7 1D   P         150
111. 3325 10 3451 12  P  1D 1S 1N    90
112. 5341 14 3235  8 1S   P         200
115. 2551 11 5233 13 1H  3N  P      460
117. 3433 10 3352 15  P  1D 1S 3N   400
119. 2623  9 3325  9 2D  2H  P      110
```

```
126. 3352 13 4342 15 1D  3N  P      430
127. 3343 11 3514 13  P  1H 1S 2H   200
129. 5332 13 5242  9 1S   P         170
131. 3415  8 1453 10  P  1D 1N 2D  -500 (catastrophe)
132. 4414 10 2542 12  P  1H 3S 4H   480 (what does 3S mean)
138. 1462 10 5215 18  P  3N 5C 6S   -50 (strange)

145. 5332 17 4315  5 1N  2C 2D 2S   170
146. 1426 12 3523 14 1C  3N  P      490
148. 3334 21 4153 10 2C  3S 3N 6S   980 (?)
149. 5314 12 4531 10 1S  4S  P      420
152. 3244 18 2344  9 1C   P         150 (3 N?)
153. 3442 13 3622  8 1H   P         170
154. 3352 16 3325  9 1N  3N  P      430
156. 4243 14 4135 13 1S  4C 4D 6S   980 (?)
158. 4315 15 2164  7 1C  2D 2H 3C   150
161. 3433 13 5233  9 1H  1S  P      170
162. 3433 13 2362 11 1H  3N  P      430
163. 2461  9 5125 12  P  1S 1N 2S   -50
165. 5332 14 2533 10 1S   P         170
169. 3433 13 3145 14 1H  3N  P      460
172. 2452 20 4225 11 2C  3H 4H 7C  1440 (?)
```

At a later position (epoch 1000) under the the same training conditions, a bidding system was emitted. Only a very small part of it (the response when partner has started with a Pass) is presented below It is significant because it indicates which of the suits to open and which strengts/point-counts are associated with a particular shape. Many cases are innovative and maybe unknown to most human bridge bidders. Some examples are: opening with only seven h.c.p. when having six spades, two hearts and a four card minor (club or diamond) suit, opening with the lower of adjacent suit if weak (then pass) or strong (continue bidding) and higher suit if medium strength, pass if balanced and less than fourteen h.c.p, open one or three NT with strong hands and any shape, two hearts and spades(stronger) shows a good hand with eight+ cards in the majors, open with nine h.c.p and unbalanced with six-card major or diamonds or when having a five-five shape,.. The question marks denotes situations which never occured during training.

```
Response to  P
         0  4  7  9 10 11 12 13 14 15 16 18 20 22 25 28
6421 :   P  P  P 1S 1S 1S 1S 1S 1S 1S 1S 2H 3C 2S  ?  ?
6412 :   P  P  P 1S 1S 1S 1S 1S 1S 1S 2C 3N 2C 3N 3N  ?
6331 :   P  P  P 1S 1S 1S 1S 1S 1S 1S 1S 3N 3N 3S  ?  ?
6322 :   P  P  P  P  P 1S 1S 1S 1S 1S 1S 1N 3N 3N 3N  ?
6313 :   P  P  P 1S 1S 1S 1S 1S 1S 1S 1S 1N 3N 3N  ?  ?
6241 :   P  P 1S 1S 1S 1S 1S 1S 1S 1S 1S 3N 3N 3N 2D  ?
```

```
6232 :   P  P  P  P 1S 1S 1S 1S 1S 1S 1N 1N 2D 3N 2S  ?
6223 :   P  P  P  P 1S 1S 1S 1S 1S 1S 1S 2S 3N 3S  ?  ?
6214 :   P  P 1S 1S 1S 1S 1S 1S 1S 1S 1N 3N 1S 3N  ?  ?
6142 :   P  P  P  P 1S 1S 1S 1S 1S 1S 1S 2S 3N  ?  ?  ?
6133 :   P  P  P 1S  P 1S 1S 1S 1S 1S 1N 3N 3N 2C  ?  ?
6124 :   P  P  P 1S 1S 1S 1S 1S 1S 1S 1S 3N 3N 3C  ?  ?

         0  4  7  9 10 11 12 13 14 15 16 18 20 22 25 28
5521 :   P  P  P  P 1H 1H 1H 1H 1H 1H 1H 3N 3N 3N  ?  ?
5512 :   P  P  P  P 1S 1S 1S 1H 1H 1H 1H 3S 3N 3S  ?  ?
5431 :   P  P  P  P 1S 1S 1S 1S 1S 1S 1S 1H 3N 3N 2C  ?
5422 :   P  P  P  P  P 1S 1S 1S 1S 1S 1S 1S 2S 3N 2N 3C
5413 :   P  P  P  P  P 1S 1S 1S 1S 1S 1S 1S 2H 2S  ?  ?
5341 :   P  P  P  P 1S 1S 1S 1S 1S 1S 1S 1S 3N 3N  ?  ?
5332 :   P  P  P  P  P  P 1S 1S 1S 1S 1S 1N 3N 3N 3N  ?
5323 :   P  P  P  P  P  P 1S 1S 1S 1S 1S 1N 3N 3N 3N 2H
5314 :   P  P  P  P 1S 1S 1S 1S 1S 1S 1S 3N 1C 2H 3N  ?
5251 :   P  P  P  P 1S 1D 1D 1D 1S 1S 1S 2D 3N 1N  ?  ?
5242 :   P  P  P  P  P  P 1S 1S 1S 1S 1S 1N 3N 3N 2D 3D
5233 :   P  P  P  P  P  P  P 1S 1S 1S 1S 1N 1N 3N 3S  ?
5224 :   P  P  P  P  P 1S 1C 1S 1C 1C 1S 1N 3N 3N 3N 3N
5215 :   P  P  P  P 1S 1S 1C 1C 1S 1S 1S 2H 3D 3N  ?  ?
5152 :   P  P  P 1D 1D 1S 1S 1S 1S 1S 1S 2D 2D 3N  ?  ?
5143 :   P  P  P  P  P 1S 1S 1S 1S 1S 1S 1N 3N 3S 3D  ?
5134 :   P  P  P  P 1S 1C 1S 1S 1S 1S 1S 1N 3N 3N 3C  ?
5125 :   P  P  P 1C 1C 1S 1S 1S 1S 1S 1S 1N 3C 3N  ?  ?

         0  4  7  9 10 11 12 13 14 15 16 18 20 22 25 28
4621 :   P  P  P 1H 1H 1H 1H 1H 1H 1H 1N 2H 3N 2H  ?  ?
4612 :   P  P  P 1H 1H 1H 1H 1H 1H 1H 1H 2H 2S 3C  ?  ?
4531 :   P  P  P  P  P 1H 1H 1H 1H 1H 1H 1H 3N 3H  ?  ?
4522 :   P  P  P  P  P  P 1H 1H 1H 1H 1H 1N 1N 3N 2H  ?
4513 :   P  P  P  P  P 1H 1H 1H 1H 1H 1H 1H 3N 2H 3H  ?
4441 :   P  P  P  P  P  P 1D 1H 1D 1H 1S 1H 2D 3N  ?  ?
4432 :   P  P  P  P  P  P  P  P 1S 1H 1H 1N 1N 3N 3N 3H
4423 :   P  P  P  P  P  P  P  P 1H 1H 1H 1N 2C 3N 3H  ?
4414 :   P  P  P  P  P  P 1C 1C 1C 1S 1C 1H 1H 3N 3N  ?
4351 :   P  P  P  P  P 1D 1D 1D 1D 1S 1D 3N 3N 3N  ?  ?
4342 :   P  P  P  P  P  P  P  P 1D 1D 1S 1N 1N 3N 3D  ?
4333 :   P  P  P  P  P  P  P  P 1S 1C 1N 1N 2C 3N 3N 2C
4324 :   P  P  P  P  P  P 1C 1C 1C 1C 1S 1N 2C 2C 3N 3N
4315 :   P  P  P  P  P 1C 1C 1C 1C 1C 1C 1S 3N 3S 3D  ?
4261 :   P  P  P 1D 1D 1D 1D 1D 1D 1D 1D 1S 3N 3D  ?  ?
4252 :   P  P  P  P 1D 1D  P 1D 1D 1D 1S 1N 3N 1S 3S  ?
4243 :   P  P  P  P  P  P  P 1D 1D 1D 1N 1S 1N 3C 3S  ?
```

4234 :	P	P	P	P	P	P	P	1C	1C	1S	1C	1N	3N	3N	3N	?
4225 :	P	P	P	P	P	P	P	1C	1C	1C	1C	2C	3N	3N	3D	?
4216 :	P	P	P	P	P	1C	1C	1C	1C	1S	1N	2S	2C	2H	?	?
4162 :	P	P	P	1D	P	P	1D	1D	1D	1S	1S	2D	3N	3N	?	?
4153 :	P	P	P	P	P	1D	1D	1D	1D	1D	1S	1S	3N	3N	3N	?
4144 :	P	P	P	P	P	P	1C	1C	1S	1C	1C	1N	1C	3N	3D	?
4135 :	P	P	P	P	P	1C	1C	1C	1C	1C	1C	3N	3N	3N	?	?
4126 :	P	P	P	P	P	1C	1C	1C	1C	1N	1C	2S	3S	3N	?	?

	0	4	7	9	10	11	12	13	14	15	16	18	20	22	25	28
3631 :	P	P	P	P	1H	1H	1H	1H	1H	1H	1H	2H	3N	3C	3N	?
3622 :	P	P	P	P	P	1H	1H	1H	1H	1H	1H	2C	3N	3N	3N	?
3613 :	P	P	P	P	1H	1H	1H	1H	1H	1H	1H	1H	2C	2D	?	?
3541 :	P	P	P	P	P	1H	1H	1H	1H	1H	1H	3D	2S	2H	2S	?
3532 :	P	P	P	P	P	P	P	P	1H	1H	1H	1N	3N	3N	2C	?
3523 :	P	P	P	P	P	P	P	1H	1H	1H	1H	1H	3N	2C	2H	3C
3514 :	P	P	P	P	P	1C	1H	1H	1C	1H	1H	1H	3N	3N	3C	?
3451 :	P	P	P	P	P	1D	1D	1D	1D	1D	1D	1N	2H	3D	?	?
3442 :	P	P	P	P	P	P	P	P	1D	1D	1H	1N	3N	3N	3N	?
3433 :	P	P	P	P	P	P	P	P	1C	1H	1H	1N	1N	3N	2H	3N
3424 :	P	P	P	P	P	P	P	1C	1C	1C	1H	1N	3N	3N	3N	?
3415 :	P	P	P	P	P	1C	1C	1C	1H	1C	1C	1H	1N	3N	?	?
3361 :	P	P	P	P	P	1D	1D	1D	1D	1D	1D	1N	1N	3N	3S	?
3352 :	P	P	P	P	P	P	1D	1D	1D	1D	1N	1D	3N	3N	2D	?
3343 :	P	P	P	P	P	P	P	P	1D	1D	1D	1N	3N	3N	3N	2S
3334 :	P	P	P	P	P	P	P	1C	1C	1C	1C	1N	3C	3N	3N	?
3325 :	P	P	P	P	P	P	1C	1C	1C	1C	1C	1N	3N	3N	3N	?
3316 :	P	P	P	P	1C	1C	1C	1C	1C	1N	1C	1C	3H	3N	?	?
3262 :	P	P	P	P	P	P	1D	1D	1D	1D	1N	3N	3N	3N	2N	3H
3253 :	P	P	P	P	P	P	P	1D	1D	1D	1D	1N	3N	2C	3N	?
3244 :	P	P	P	P	P	P	P	1C	1D	1C	1D	1N	3N	3N	3N	?
3235 :	P	P	P	P	P	P	P	1C	1C	1C	1C	1N	2C	3N	3D	?
3226 :	P	P	P	P	P	P	1C	1C	1C	1C	1N	1N	3N	3N	3H	?
3163 :	P	P	P	P	1D	1D	1D	1D	1D	1N	1D	2C	2N	3N	2D	?
3154 :	P	P	P	P	P	P	1D	1D	1C	1C	1C	1C	2C	3N	3N	?
3145 :	P	P	P	P	P	1C	1C	1C	1C	1C	1C	1N	3N	2C	2S	?
3136 :	P	P	P	P	P	1C	1C	1C	1C	1C	2C	2S	3N	3N	?	?

	0	4	7	9	10	11	12	13	14	15	16	18	20	22	25	28
2641 :	P	P	P	1H	1H	1H	1H	1H	1H	1H	1H	2H	2H	3N	2S	?
2632 :	P	P	P	P	P	1H	1D	1H	1H	1H	1H	1N	3N	3N	?	3N
2623 :	P	P	P	P	P	1H	1H	1H	1H	1H	1H	1H	3N	3N	2D	?
2614 :	P	P	P	1H	1H	1H	1H	1H	1H	1H	1C	1H	2C	2C	?	?
2551 :	P	P	P	P	1D	1H	1D	1D	1H	1D	1H	1H	3H	2H	?	?

```
2542 :   P  P  P  P  P  P 1H 1H 1H 1H 1H 1H 3N 3N 3N  ?
2533 :   P  P  P  P  P  P  P 1H 1H 1H 1H 1H 3N 3N 3N  ?
2524 :   P  P  P  P  P  P 1H 1H 1C 1H 1H 1H 3N 3N  ? 3D
2515 :   P  P  P 1C  P 1H 1C 1H 1H 1H 1H 1C 2C 3C  ?  ?
2461 :   P  P  P 1D  P 1D 1D 1D 1H 1D 1D 3N 3D 3D 3H  ?
2452 :   P  P  P  P  P  P 1D 1D 1D 1H 1D 2D 3N 3N 2D  ?
2443 :   P  P  P  P  P  P  P  P 1D 1H 1H 1N 3N 3N 3N  ?
2434 :   P  P  P  P  P  P  P 1C 1C 1H 1H 1N 3N 3N 3N  ?
2425 :   P  P  P  P  P  P 1C 1C 1C 1H 1C 1H 3N 3N 2H  ?
2416 :   P  P  P  P  P 1C 1C 1C 1C 1H 1H 3C 2C 3N  ?  ?
2362 :   P  P  P  P  P  P 1D 1D 1D 1D 1D 3N 3N 3N 3C 3D
2353 :   P  P  P  P  P  P  P 1D 1D 1D 1D 1N 3H 3N 2D  ?
2344 :   P  P  P  P  P  P  P 1C 1C 1D 1C 1N 3N 2C 3N  ?
2335 :   P  P  P  P  P  P  P 1C 1C 1C 1N 2C 3N 3N 3H  ?
2326 :   P  P  P  P  P  P 1C 1C 1C 1C 1N 3N 3N 3N 1N  ?
2263 :   P  P  P  P  P  P 1D 1D 1D 1N 1D 3D 2D 3N  ?  ?
2254 :   P  P  P  P  P  P  P 1C 1C 1D 1D 1C 3N 3N 3N 3N
2245 :   P  P  P  P  P  P  P 1C 1C 1C 1C 1N 3N 3N  ?  ?
2236 :   P  P  P  P  P  P 1C 1C 1C 1N 1C 1N 1N 3N 2N  ?
2164 :   P  P  P 1D  P 1C 1C 1D 1D 1D 1N 3N 3N 3D  ?  ?
2155 :   P  P  P  P  P 1C 1D 1D 1C 1D 1D 1D 2C 3N 3S  ?
2146 :   P  P  P  P 1C 1C 1C 1C 1D 1D 1N 2D 3N 3N  ?  ?


         0  4  7  9 10 11 12 13 14 15 16 18 20 22 25 28
1642 :   P  P  P 1H 1H 1H 1H 1H 1H 1H 1H 3N 3N 2C 2C  ?
1633 :   P  P  P 1H 1H 1H 1H 1H 1H 1H 1H 1H 3N 3N 1H  ?
1624 :   P  P  P 1H 1H 1C 1H 1H 1H 1H 1H 2C 2H 3D  ?  ?
1552 :   P  P  P  P 1H 1D 1D 1D 1H 1D 1D 1H 2D 3H 3H  ?
1543 :   P  P  P  P  P 1H 1H 1H 1H 1H 1H 1H 2D 3N 3D  ?
1534 :   P  P  P  P  P 1C 1C 1H 1H 1H 1C 1H 2D 3N  ?  ?
1525 :   P  P  P 1C 1C 1H 1H 1H 1H 1H 1H 1H 3N 3C  ?  ?
1462 :   P  P  P  P 1D 1D 1D 1D 1D 1D 1H 2D 3N 3C 3H  ?
1453 :   P  P  P  P  P 1D 1D 1D 1D 1D 1H 1N 3N 3N 2C  ?
1444 :   P  P  P  P  P  P  P 1C 1D 1H 1C 1N 1N 3N  ?  ?
1435 :   P  P  P  P  P  P 1C 1C 1C 1C 1H 1H 1C 3N 3N  ?
1426 :   P  P  P  P  P 1C 1C 1C 1C 1C 1C 3N 3N 3C 3S  ?
1363 :   P  P  P  P 1D 1D 1D 1D 1D 1D 1N 1N 1D 3N  ?  ?
1354 :   P  P  P  P 1D  P 1D 1C 1D 1C 1C 2H 2D 3N 3N  ?
1345 :   P  P  P  P  P 1C 1C 1C 1C 1C 1N 1N 3N 2D 3D  ?
1336 :   P  P  P  P  P 1C 1C 1C 1C 1N 1C 3N 3C 3N 3S 2D
1264 :   P  P  P  P 1D 1D 1C 1D 1D 1D 1D 3N 3N 2D  ?  ?
1255 :   P  P  P  P  P 1C 1C 1C 1D 1C 1D 2D 3D 2C 3N  ?
1246 :   P  P  P  P  P 1C 1C 1C 1D 1C 1C 1C 3N 3N  ?  ?
```

9.5 Conclusions

The runs show it is possible to find a reasonable bidding system within a few hours of time. A relatively high learning rate combined with high exploration resulted in the best performance. The test-set evaluations are quite close to the training evaluations, indicating that no over-fitting is achieved yet. An initial q_0 value set to a reasonable lower acceptable threshold seem to steer the learning into realistic areas. Good runs with decreasing learning rate was not possible to achieve due to the integer nature of this implementation.

The resulting bidding system, studied in detail is also interesting. The responder has learnt "understood", that an opener shows hearts or spades or a strong hand with the two diamond opening bid and thereby never passes and does not bid two hearts with heart support. The no-trump bidding is also interesting; two-bid responses seem to be sign off, two NT is never used and three in a suit seems to be highly conventional and never passed by the NT-opener. Three NT is often the response with medium strength hands and not an extreme shape and four in suit is a transfer bid. Maybe an interesting NT-bidding part of the system can be developed if the learning were concentrated on this?

Anyhow; the learnt system is not very useful until the defensive bidders are taken into consideration.

10 The Sum-game

After some discussion with David Sumpter we invented a very simplified "mini-bridge" game which we called the Sum-game. The total number of "hands" is just five. It is played by two players and the objective is to find out the sum of the numbers given to each player. Reinforcement given is 0 to both players if the sum is not correct and 1 if it is correct. We assume each player receives a small random number (in the interval 1..5) and the bids are limited to a minimum and the second player has to increase the bid made by the first player. If the first player gives a higher bid than the total sum the second player is without recourse. How should the second player enforce the first one into reconsidering his bidding system? One optimal bidding system is:

Opener have "state"	Bid "action" (1..5 available)
1	1
2	2
3	3
4	4
5	5

The state space for the responder is two-dimensional (the first two columns); a matrix ;

Responder have	Openers bid was	Bid "action" openers+1..openers+5 available
1	1	2
1	2	3
..		
2	1	3
2	2	4
..		
5	5	10

This bidding strategy is not implied by the Q-learning algorithm; opener selects just a random bid (depending on initial values of the Q-table) and the responder tries his best to keep the reinforcement high by adapting (following).

10.1 What is learnt?

A typical bidding system learnt with q-learning is

Opener have "state"	Bid "action" (1..5 available)
1	3
2	1
3	4
4	5
5	2

(or some other bad permutation) and the action selection table for the responder is

Responder have	Openers bid was	Bid "action" openers+1..openers+5 available
1	1	3 (correct, r=1)
1	2	6
1	3	4 (wrong, r=0, bid 2 not available)
..		
2	1	4
2	2	7
..		
5	5	9

and the average reinforcement is in the 0.8 area instead of the optimum value of 1. Other problems are due to coordination ambiguities:

Opener have	Bid
1	3 in 60% of cases and 2 in the other 40 %
2	2 in 27% and 4 in 73%
3	1..5 (every bid equally bad with expected reinforcement around 0.1..)

etc.

forcing the responder into probabilistic responses. These problems seem to converge into a (often non-optimal) single bid but the process takes unnecessarily long learning time. One possibility is for the opener to track the responder's action and try to force the opener to relearn.

"When I opened 3 with my 1 our reinforcement was sometimes bad, maybe I should try to relearn?" or "We never get any good reinforcement when I have a 1 and respond to a 3 bid, how should I force the bidding system into something else?"

On this particular evaluator function, Player two may keep on exploring (select random actions) and on average the opener will probably find his best moves, but this is not a generally usable method [San00].

Extensions to the Sum-game to make it approach Bridge will be to have more random states than opening bids, say number 1..10 but only bids 1..5 available and maybe introduce a third bid; rebid. Another extension might be to have some distributional selection/information by making the second player's average number slightly lower if the first player has got a high one, thus keeping the average sum standard deviation lower.

10.2 Some similar games

Assume we have two players, the first one "the Talker" and the second one "The Thinker". Each Talker consists of a small vector with the size of the problem (number of states) with numbers denoting what to say when situated in those states. The Thinker has a matrix with his own state on one axis and the Talker's output on the other and the values will be the thing the Thinker says when situated in a state and listening to the Talker.

We have a population with Talkers and Thinkers and they perform discussions with the reinforcement zero when they fail in their cooperative task and reinforcement one when they succeed.

I invented some new games to play around with these scenarios;

a) Listening

b) Thinkforyourselfing

c) Combining

10.2.1 Listening

Here the reinforcement was one if the thinker simply repeated the saying of the talker independent of what state it was situated in.

10.2.2 Thinkforyourselfing

Reinforcement was set to one when the Thinker said what state it was situated in.

10.2.3 Combining

The "difficult" task. The Thinker should say something which is a combination of what state it was put in and what the Talker said. The reinforcement signal is a permutation of the sum-matrix mentioned above to disable the possibility of just keep exploring.

10.3 Results

Although these games were not possible to solve with standard Q-learning we could solve them easily with a genetic algorithm(GA) approach. All three games were successfully learnt during the few tries but more, and more sophisticated, testing is to be carried out in later work. This indication leads us into trying to solve the bridge problem with a genetic approach and preliminary attempts have already been made with some positive results although on a small scale, only 3 bids. The GA implementation transforms the separate learners into a single population so in a way this is slightly unfair and maybe explains the advantageous results; the Q-learning would probably work if it was reformulated into a single agent problem but then this problem would be out of the scope for this thesis. The GA can probably be re-implemented, more carefully, in a fair way without reducing it's power. Distributed or parallel programming suits GA well and it is easy to dimension the population into actual memory size. It can also be interrupted any time, getting the currently best result when needed. One of the problems we found was premature convergence, when most individuals look the same, and we solved it by increasing the mutation rate but this should be dealed with in a better way, i.e. by keeping diversity and reduce the selective pressure, as indicated in the literature [Mic99].

The Q-learning was tried with an $\epsilon - greedy$ exploration rate of 0.02 for both players and a learning rate of 0.001. We tried to learn for 1,000,000 cycles

without finding the optimal solution. The genetic approach used a population of 1000 individuals formed into pairs, involved in 2000 discussions each and converged within 1000 generations. We used a roulette wheel selection with the number of successful discussions as the fitness and mutated at a rate of 0.001.

11 Acknowledgments and Epilogue

Thanks to my supervisors Patrik Eklund and Per-Åke Wedin. Thank you also to Lars-Erik Janlert, Bo Kågström, Göran Broström, David Sumpter, Stephen Hegner, Torun Israelsson, Thomas Hellström, Lollo Sandström, Lennart Edblom, Iradj Roozbeh, Andreas Hed, Staffan Andersson, Elisabeth Blackham, Martin Löfgren, My family and Bridge partners.

Some of this research was conducted using the resources of High Performance Computing Center North (HPC2N).

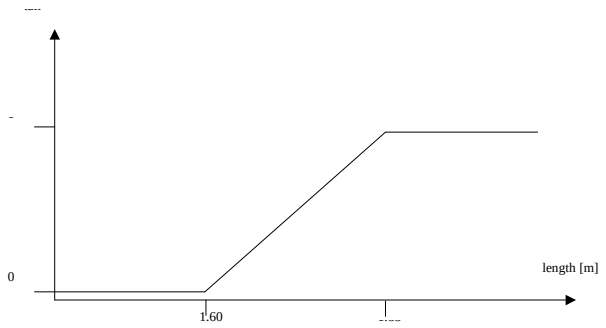
Since the middle of the eighties I have tried to get the opportunity to do some research. Because of the good times[23] and opportunities around the world, people are leaving university for well-paid jobs. I have gone the other way and have now got room in the Academic Community. Thank you all ex-researchers, spending days and nights with boring computerized card-registers (Thomas Hellström, 1999) for making space for me!

In the middle of the seventies I was introduced to computers and also learnt the card-game of Bridge, spending plenty of hours in these two fields. Since then I have spent a lot of time with other things like trying to create a family, playing music, making friends etc. but now it seems possible for me to go back to the Basics and combine the fun things I had time to do as a teenager and rediscover the happiness of life. A perfect thing to do would of course be to combine all those funny pleasures into a research project; and now it is done. I am proud to present this work, it has been a joy to do but painful to write and document, why is it always so? Of course the music and friends still take up a big place in my heart and I think they are more important even than Bridge and computers. A special red box in my heart belongs to Isa, not named after the Artificial Intelligence term IS-A or the Isa-bus on IBM PC Computers, she simply is-a daughter of mine, who enjoys going by bus (among other things).

[23] In the 90ies..

.

A Fuzzy Logic



The membership function for tall.

In normal logic a value of a statement is either true (1) or false (0). To apply to the Swedish Police Academy you have[24] to be at least 1.85 meters tall. Either you are tall enough or you have to go for something else. In Fuzzy Logic[JCTS97], (related to Multi-valued logic), a statement has a level of veracity. Suppose you have a function *tall(height)* where tall returns 1 for heights above 1.85 m and 0 for heights under 1.60 m and a increasing (not necessarily linear) value between 0 and 1 for heights in the range 1.60 to 1.85. This function *tall* is called a membership function denoting the (Fuzzy) belonging to the set of tall people for a particular argument. Lotfi A. Zadeh and many more have developed mathematics to calculate the value of statements with logical conjunctions like if tall(a) AND rich(a) THEN lucky(a). Sets of these kind of rules can also be combined into the most suitable answer. Observe that there is no probability assumptions in those descriptions; a person with height 1.79 is tall to a degree of 0.7; the probability is not 0.7 to be called tall by some randomly selected person although such interviews can be done to estimate parameters of the membership functions. The value 1.79 is called a crisp value and the value tall is a fuzzy value (membership function). A person with height 1.79 probably belongs highly to *averageheightpeople* but belongs to *shortpeople* also (to a lesser extent).

A.1 Fuzzy control

By giving a controller vague simple descriptions in the form of: "if the road turns left then decrease the steering-wheel angle" the main behavior can be specified by a human operator and further refined by the operator or the controller during operation. Membership functions defuzzify statements into crisp

[24] Not necessarily true today.

values and vice versa. Several if-statements can be combined by using for instance Mamdani[JCTS97] rules into one averaged action. However, beware of dangerous averaging, e.g.: one rule saying turn to the right of the tree and another one saying turn to the left of tree resulting in an average of "continue straight" (and collide). The way to learn is normally finding the membership functions and their parameters but it can also be generating or eliminating rules[PDH97]. Fuzzy logic rule sets can, under some conditions, be seen as an ANN with many layers. Generalized approximate reasoning-based intelligent control (GARIC)[BK92] is one such approach (with five layers) which uses reinforcement learning to learn and adjust membership functions. In the bridge area concepts can be found or invented like a balanced hand and bidding rules like "IF balanced(h) AND strong(h) THEN open(1NT)". The value for balanced is probably 1 for 4333 but less for 5422 and zero for shapes like 4711[25]. "Strong" can be a Gaussian function with the peak at the strength (see section 8.4) 16 high card points (h.c.p.) overtaken by a ramp-function "VeryStrong" at higher h.c.p.:s. By using such rules we are more flexible and soft in our decisions; a person can study 'policing' although the length is only 1.83 if other factors in the evaluative function is high like empathy or muscular strength. More information about Fuzzy Control can be found in [Kos97] and more advanced information in [PDH97].

[25] The Eau de Cologne distribution.

References

[AM94] Sten Alfredsson and Jan Malmström. *Trappan (in Swedish)*. Svenska Bridgeförlaget, Tolg, 360 40 Rottne, 1994.

[And99] Thomas Andrews. Double dummy bridge evaluations, 1999. http://www.best.com/ thomaso/bridge/valuations.html.

[Art94] W. B. Arthur. Complexity in economic theory: Inductive reasoning and bounded rationality. *The American Economic Review*, 84:406–411, 1994.

[Åst95] Karl J. Åström. *Adaptive Control, 2nd Ed.* Addison-Wesley, 1995.

[ÅW84] Karl-Johan Åström and Björn Wittenmark. *Computer Controlled Systems*. Prentice-Hall, 1984.

[Bel57] R. E. Bellman. *Dynamic Programming*. Princeton University Press. Cited via [Sut98], 1957.

[Ber76] Dimitri P. Bertsekas. *Dynamic Programming and Stochastic Control*. Academic Press, New York, 1976.

[BF98] H. Scott Bierman and Luis Fernandez. *Game Theory with Economic Applications*. Addison-Wesley, 1998.

[BGP93] Manny Rayner Björn Gambäck and Barney Pell. Pragmatic reasoning in bridge. Technical Report 299, University of Cambridge, Computer Laboratory, Cambridge, England, 1993.

[BK92] H. R. Berenji and P. Khedkar. Learning and tuning fuzzy logic controllers. *IEEE Transactions on neural networks*, 3:724–740, 1992.

[Bor98] Magnus Borga. *Learning Multidimensional Signal Processing*. PhD thesis, Linköping, 1998.

[Bro86] V. B. Brooks. *The Neural Basis of Motor Control*. Oxford University Press, 1986.

[BS99] Don F. Beal and Martin C. Smith. Temporal coherence and prediction decay in TD learning. *Proceedings from Sixteenth International Joint Conference on Artificial Intelligence*, pages 564–569, 1999.

[BT01] Ronen I. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Proceedings from IJCAI'01*, 2:953–958, 2001.

[BV00] Michael Bowling and Manuela Veloso. An analysis of stochastic game theory for multiagent reinforcement learning. 2000.

[BV01] Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In Bernhard Nebel, editor, *Proceedings from IJCAI'01*, pages 1021–1026, San Fransisco, 2001. Morgan Kaufmann.

[Car62] Gay Carley. A program to play contract bridge. Master's thesis, Dept. of Electrical Engineering, Massachussets Institute of Technology,Cambridge,Massachussets, 1962.

[Car02] Johan Carstensen. Kontroll och styrning av fyrbenta robotar med hjälp av invers kinematik (in swedish). Master's thesis, Umeå Universitet, 2002.

[CB96] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. *Advances in Neural Information Processing Systems (proceedings 1995)*, pages 1017–1023, 1996.

[CB98] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *Proceedings of AAAI-97 Workshop on Multiagent Learning*, 1998.

[Coh92] Larry Cohen. *To Bid or Not to Bid - The Law of Total Tricks*. Natco Press, 1992.

[Dag98] Eva Dagnegård, editor. *Reglermöte '98*, Box 118, 221 00 Lund, Sweden, 1998. Dept of Automatic Control, Lund Institute of Technology.

[Ers92] Christina Erskine. Omar sharif's bridge. *PC Review*, July:58–59, 1992.

[Fer99] Jacques Ferber. *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, England, 1999.

[FL00] Henry Francis and Paul Linxwiler. Bill gates finds duplicate bridge a "window" to fun. *2000 Summer NABC Daily Bulletin*, 72(10):1–2, August 20 2000.

[Fra98] Ian Frank. *Search and Planning Under Incomplete Information*. Springer-Verlag, London, 1998.

[Gin93] Matthew L. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46, 1993.

[Gin96a] M. L. Ginsberg. How computers will play bridge. *The Bridge World*, June 1996.

[Gin96b] Matt Ginsberg. An analysis of the law of total tricks. *The Bridge World*, November, 1996.

[Gin96c] Matthew L. Ginsberg. Partition search. *AAAI-1996*, 1996.

[Gin99] M. L. Ginsberg. GIB: Steps toward an expert-level bridge-playing program. *Proceedings from Sixteenth International Joint Conference on Artificial Intelligence*, pages 584–589, 1999.

[GS01] Frédérick Garcia and Florent Serre. From q(lambda) to average q-learning. *Proceedings from IJCAI'01*, 2:959–964, 2001.

[Hay99] Simon Haykin. *Neural Networks, 2nd Ed.* Prentice-Hall, 1999.

[Hil01] Robin Hillyard. Extending the law of total tricks. *http://www.bridge.hotmail.ru/file/TotalTricks.html*, 2001.

[Htt01] Http://Www.Robocup.Org. Robocup. 2001.

[JCTS97] J.-S. R. Jang and E. Mizutani C.-T. Sun. *Neuro-Fuzzy and Soft Computing.* Prentice-Hall, 1997.

[Joh99] Stefan J. Johansson. *Game Theory and Agents.* PhD thesis, University of Karlskrona/Ronneby, 1999.

[Koh82] T. Kohonen. Self-organized formation to topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[Kos97] Bart Kosko. *Fuzzy Engineering.* Prentice-Hall, 1997. Software included.

[Lan97] Tomas Landelius. *Reinforcement Learning and Distributed Local Model Synthesis.* PhD thesis, Linköping, Sweden, 1997.

[Lin83] E.T. Lindelof. *The Computer Designed Bidding System - COBRA.* Number ISBN 0-575-02987-0. Victor Gollancz, London, 1983.

[Lju81] Lennart Ljung. *Reglerteori, Moderna Analys- Och Syntesmetoder (in Swedish).* Studentlitteratur, Lund, Sweden, 1981.

[Mac91] J. MacLeod. Microbridge - a computer developed approach to bidding. *Heuristic Programming in AI - The First Computer Olympiad*, pages 81–87, 1991.

[Man94] B. Manley. Bridge software: Moving forwards or backwards? *The Bulletin, American Contract Bridge League*, October, 1994.

[ME99] Wolfgang Maass and Christopher M. Bishop (Editors). *Pulsed Neural Networks.* The MIT Press, Massachusetts, Cambridge, 1999.

[MF00] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics.* Springer-Verlag, Berlin Heidelberg New York, 2000.

[Mic99] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin Heidelberg New York, 1999.

[Min61] M. L. Minsky. Steps towards artificial intelligence. *Proc. Of the Institute of Radio Engineers*, 49:8–30, 1961.

[Nil78] Mats Nilsland. *Modern Standard*. Sveriges Bridgeförbund, Stockholm, 1978.

[NT74] K. S. Narendra and M. A. L. Thathachar. Learning automata - a survey. *IEEE Trans on Systems, Man and Cybernetics*, 4:323–334, 1974.

[NW03] Peter Nordin and Johanna Wilde. *Självlärande Robotar Och Artificiell Intelligens (in Swedish)*. Liber, Stockholm, Sweden, 2003.

[PDH97] Rainer Palm, Dimiter Driankov, and Hans Hellendoorn. *Model Based Fuzzy Control*. Springer-Verlag, 1997.

[Pro84] Kalle Prorok. Analys och simulering av kommunicerande system (in swedish). Master's thesis, Uppsala, 1984.

[Pro99] Kalle Prorok. Gubben i lådan (in swedish). *Svensk Bridge*, 31(no 4, october):26–27, 1999.

[San00] Thoumas Sandholm. Personal communication at autonomous agents, 2000. Barcelona.

[Sar94] Warren Sarle. Kangaroos and training neural networks, 1994. ftp://ftp.sas.com/pub/neural/kangaroos.

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*. The MIT press, London, 1998.

[SBW91] R. S. Sutton, A. G. Barto, and R. J. Williams. Reinforcement learning is direct adaptive optimal control. In *Proc. Of the American Control Conf.*, pages 2143–2146, 1991.

[SC85] Tuomas W. Sandholm and Robert H. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems Journal*, 37:144–166, 1985.

[Sch99] Jürgen Schmidhuber. Artificial curiosity based on discovering novel algorithmic predictability through coevolution. 1999.

[Sen00] Sandip Sen. Learning agents, 2000. Tutorial on the conference Autonomous Agents, Barcelona.

[Sha50] C. E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41(4):256–275, 1950.

[SN90] L. Sterling and Y. Nygate. Python: An expert squeezer. *Journal of Logic Programming*, 8:21–40, 1990.

[Sta77] A. Stanier. *Decision-Making with Imperfect Information.* PhD thesis, Essex University, 1977.

[Sto00] Peter Stone. *Layered Learning in Multiagent Systems.* The MIT Press, 2000.

[Str97] Bjarne Stroustrup. *The C++ Programming Language, 3rd Ed.* Addison-Wesley, 1997.

[Tes94] Gerald Tesauro. TD-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6:215–219, 1994.

[Tes00] Gerald Tesauro. Pricing in agent economies using neural networks and multi-agent q-learning. *In Proceedings of IJCAI Workshop on Learning About, From, and With Other Agents, August 1999*, 2000.

[Tho11] E. L. Thorndike. *Animal Intelligence.* Hafner, Darien, CT, 1911.

[TS02] Kagan Tumer and Peter Stone, editors. *Collaborative Learning Agents*, Menlo Park, California, USA, 2002. AAAI Press.

[Tur50] A. M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[Ver69] Jean-Rene Vernes. The law of total tricks. *The Bridge World*, 1969.

[Vol99] Hans Vollbrecht. Kd-q-learning with hierarchic generalization in action- and state-space. *Fourth European Workshop on Reinforcement Learning (EWRL-4) notes on the web-pages; http://iridia0.ulb.ac.be/ ewrl/EWRL4/EWRL4.program.html*, 1999.

[WD92] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[WF65] M. D. Waltz and K. S. Fu. A heuritic approach to reinforcement learning control systems. *IEEE Transactions on Automatic Control*, 10:390–398, 1965.

[Win77] Patrick Henry Winston. *Artificial Intelligence.* Addison-Wesley, 1977.

[Woy97] Peter Woyzescke. Design and implementation of a reinforcement learning controller. Master's thesis, Umeå University, 1997.

[WT00] D. H. Wolpert and K Tumer. An illustration of the COIN approach to design of multi-agent systems. In Peter Stone and Sandip Sen, editors, *Workshop on Learning Agents*, pages 19–26, Barcelona, Spain, 2000.

Index

Abbreviations, 9
ace-asking, 56
Acol, 18
Adaptive Control, 37
adversary domain, 41
agent, 26
Allowed systems, 62
Analyses of the play, 18
Applications with RL, 31
Artificial Neural Nets, 32

Background, 10
Bayesian Belief Networks, 80
Bellman equation, 29
best play, 18
bias effects, 73
bidding, 16
bidding examples, 93
bidding scheme, 16
bidding system, 18
bidsyst.txt, 82
binary integer conversion, 86
Blocks world, 13
board, 17
Boltzmann, 72
Borel, 24
Bridge
    rules, 14

Chess, 14
chromosome, 35
climbing game, 47
collaborative domain, 41
Collaborative filtering, 11
common shapes, 75
complex bidding system, 13
concepts, 80
Continuing tasks, 28
contract, 14
Control systems, 36
conventional bid, 17
conventions, 18
cooperative game, 39
Coordination, 55
Credit assignment, 12
crisp value, 107
Culbertson, 58

Deal statistics, 74
declarer, 14
defender, 14
discard, 16
Discounted return, 27
distributional strength, 58
double dummy solver, 51
doubleton, 58
dummy, 14
duplicate, 17
dynamic game, 39
Dynamic Programming, 30

Eligibility traces, 31
Episodic tasks, 28
epsilon-greedy, 72
Evolution program, 34
expected number of tricks, 60
Experiments, 86
Exploration, 71
Exploration/Exploitation dilemma, 12
exploratory action, 10

feedback, 11
Fictitious play, 46
follow suit, 16
Frank, Ian, 44
Fuzzy control, 107
Fuzzy Logic, 107

Game, 20
Game Theory, 38
games, 8
gamma, 79
Generalized approximate reasoning-based intelligent control, 108
generalized policy iteration (GPI), 30

Genetic Algorithms, 34
genetic operators, 35
GIB, 22
Ginsberg, Matthew, 43
goal, 24
Goren, Charles, 58
guide the learning, 90

hand patterns, 57
Hidden states, 31
high card points (h.c.p.), 57
Honor Trick, 58
honors, 14

imperfect information, 39
independent learner (IL), 46
International Match-Points (IMP), 22
intervention, 17
Iterated Prisoner's Dilemma, 42
Iterative solution, 30

Joint-Action Learning, 44
jumps in the performance, 90

Laplace ratio, 67
large-scale Q-learning, 81
law of effect, 11
Layered learning, 42
leading, 14
leads the learning, 90
Learning, 10
Learning by doing, 12
Learning rates, 70
lift, 67
loosing trick count (LTC), 58
lure of multiagent systems, 41

Machine Learning, 11
majors, 18
make a sacrifice, 17
manually enter Q-values, 72
Markov, 27
Markov property, 27
meaning of the bid, 16
membership function, 107
membership functions, 107
minimax algorithm, 51
minors, 18
Monte Carlo, 31
Monte-Carlo method, 31
multiple learning agents, 41

Nash equilibrium, 39
natural bid, 17, 18
needle's eye, 90
nontechnical comparison, optimization, 36
number of hands, 75

On/Off-policy, 73
opening, 16
opening bids, defining, 82
Operant conditioning, 11
optimal state-value function, 30
optimistic start, 72
over-fitting, 101

partially observable Markov decision processes (POMDP), 31
Partition Search, 53
Perceptual aliasing, 12
Periodic Team Synchronization (PTS), 42
Pitfalls of Multiagent Systems, 41
policy improvement, 30
possible hands, 57
preemptive bid, 17
Premature convergence, 35
price wars, 39
Prisoner's Dilemma, 39
Proportional-Integr.-Deriv.(PID), 36
Psychological aspects, 19

Q-learning, 31

Random number generator., 73
ranking of suits, 16
recursive least-square identifying, 50
Reducing amount of information, 13
Reese, Terence, 81
Reinforcement Learning, 10, 26
Reinforcement learning, 12
repeated games, 39

Representations of a hand, 57
responses, learnt, 97
reward, 27
Risk taking, 19
Roulette wheel selection, 10
ruffing, 16
rule set, 66
rule-based approach, 44

sacrifice, 17
safety, 20
scientific bidding, 20
Scoring, 20
search graph, 53
self-organizing map, 11
semantics of the bid, 16
single-dummy solver, 59
singleton, 58
SOM, 11
Standard American, 18
static game, 38
Stayman convention, 17
strengths, fuzzy, 70
strictly dominant, 38
suit holdings, 57
Sum-game, 102
Support Vector machine, 78
survival of the fittest, 34
Symbols, 9
system, 8

Talkers and Thinkers, 104
temperature, 72
Temporal difference (TD), 29
The Law of Total Tricks, 58
Trade-offs, 62
transposition tables, 53
trick, 14
Trick-taking abilities, 59

unopposed bidding, 17
usefulness, 101

Value function, 28
Victory points (VP), 22
void, 58

Why bridge, 13

Zero-sum games, 39