REPORT ON THE FIRST INTERNATIONAL CONFERENCE ON COMPUTERS AND GAMES

Tsukuba, Japan
10-12 November, 1998

I. Frank and R. Grimbergen[1]

Tsukuba, Japan

The first international conference on computers and games (CG'98) took place at the Electrotechnical Laboratory, Tsukuba, Japan on November 11 and 12, 1998. A workshop was held on November 10. The conference was attended by more than 60 researchers from nine different countries, who all contributed to making it a highly rewarding event.

In the conference preliminaries, Hiroyuki Iida welcomed the participants, and Tony Marsland gave an opening address that stressed the impact of games research on the rest of computer science. The keynote address was then given by Jaap van den Herik, who discussed the role that *Predictions* play in the development of research, both in Artificial Intelligence in general, and in computer-games research in particular. He recalled (and agreed with) Turing's (1950) prediction that by the end of the century it will be no longer be controversial to speak of machines that think. He also looked at other predictions by prominent researchers such as Shannon and Simon. Moreover, he included many predictions of his own, among which the prediction that a Grandmaster-level Go (19×19) program would exist by the year 2030.

The main body of the conference then consisted of oral presentations of twenty-one papers. We review these presentations by categorising them into the following main groupings: Search (Section 1), Learning (Section 2), Mathematical Analysis (Section 3), Further Aspects of Go (Section 4) and Analysis of the Historical Development of Games (Section 5). We indicate the first introduction of a speaker's name by using bold text. In Section 6 we provide a summary of the conference side events and in Section 7 we present some conclusions and further pointers. Finally, in Section 8 we give the authors list (Conference and Workshop) together with the titles of their presentations.

1. SEARCH

The first broad category of papers are those dealing with look-ahead search. Since search constitutes one of the most important abilities of many strong game-playing programs, it is fitting that this section of our review is the largest, covering seven papers.

We start with a success story. This was provided by **Reijer Grimbergen**, who compiled a survey of computer programs that use variable-depth search to solve *tsume-shogi* problems. Shogi is a Japanese game, similar in style to western chess, but with the unique feature that when a piece is captured from the opponent it can be re-used. (A captured piece is called a "piece in hand" and at any move a player can *drop* a captured piece on a vacant square instead of moving a piece that is on the board.) Tsume-shogi problems are specialised endgame positions where one side has a mating sequence that consists exclusively of consecutive checks. Solving tsume-shogi problems is clearly simpler than perfect endgame analysis, but in shogi the possibility of dropping pieces typically makes the endgame a race, so that in practice the ability to find tsume-shogi sequences is very important. The average branching factor of tsume-shogi has been shown to be about 5, but the lengths of solution sequences can be very long (the longest on record is called *Microcosmos*, and has 1525 moves). Grimbergen's talk gave an overview of the development (which started in the 1960s) of programs for finding such sequences. This took us through the work by Noshita, Ito, Kawano, and Seo, who all produced programs that solve tsume-shogi problems of up to 15 plies better than human experts. These programs have also been tested on a set of almost 200 hard problems, some with solution sequences of over 100 plies. To solve such problems, most programs use best-first search based on the freedom of the defending King, and avoid redundant search by employing standard transposition tables. However, in shogi there are two

[1] Complex Games Lab, ETL, Umezono 1-1-4, Tsukuba, Ibaraki, Japan, 305. Email: {ianf, grimberg}@etl.go.jp

further important concepts: *domination* and *simulation*. A domination relationship occurs between two positions when the pieces on the board are exactly the same, but the pieces in hand differ. This is typically dealt with by adding extra information to the transposition table. Simulation occurs when the defender can drop different pieces between a checking piece and his King, but the mating sequence is not affected by the piece dropped. This is handled by the killer heuristic used in most chess programs. It was shown that the best program based on these techniques (Seo's) outperforms human players by far on the problem test sets, failing to solve just 6 out of the set of hard tsume-shogi problems. Seo's program can also solve the Microcosmos problem mentioned earlier.

A second example of a game-playing configuration with very long solution sequences was presented by **Andreas Junghanns** (paper with Jonathan Schaeffer) in a talk about Sokoban. This is another Japanese game (Sokoban is Japanese for "warehouse keeper"), in which a single agent (the warehouse keeper) is used by the player to push stones through the warehouse rooms and corridors into a particular goal configuration. There is a freely-distributed version of Sokoban that includes 90 puzzles, graded into degrees of difficulty. The framework used by Junghanns and Schaeffer to tackle this problem is IDA*. In previous work they have introduced enhancements to the basic algorithm, such as the automatic creation of deadlock databases and the identification of move sequences that recur frequently (macro operators). In his CG'98 presentation, Junghanns described a new technique based on the notion of *relevance*. This technique relies on the real-time computation of a domain-dependent pruning function that can be used to decide – based on the history of the search to date – what new moves are not relevant. Junghanns pointed out that the simple geometric distance from the previous moves is not sufficient to provide a good measure of relevance, and instead described an *influence* function that includes factors such as (1) the number of freedoms on the path that connects two squares, (2) the connectivity between two squares, and (3) a biasing so that squares near goals have more influence than squares that are distant from goals. This influence function was used to prune branches in the search. Such prunings were called "relevance cuts". These cuts enabled Junghanns' program to solve 5 problems that it could not handle previously, and also resulted in significant progress towards solving others. The best version of Junghanns' program can now solve a total of 52 of the 90 original Sokoban problems.

A different type of pruning in game-tree search was the *forward pruning* discussed by **Tony Marsland** (paper with Yngvi Björnsson). In forward pruning, some continuations are terminated before reaching the nominal search depth. This carries the risk of overlooking a good continuation, but also the advantage that the time saved in examining unpromising lines can be spent searching other lines more deeply. The authors investigated how to minimise the risk of pruning a good continuation by formalising the *multi-cut* $\alpha\beta$-pruning technique. The multi-cut pruning criterion is based on the *number* of promising moves at any given node ν. The criterion is met if, after examining m of the moves from ν, c of them say that ν is a cut node. Thus, the more moves that have the potential to produce a pruning, the more likely it is that ν will not be expanded. However, some method is required to determine which moves are promising. For this, Marsland proposed the following techniques: (1) number of legal moves, (2) history heuristic, (3) quiescence search, and (4) null-window search. The first two of these schemes have the advantage that they can be performed statically, whereas the latter two require additional shallow search (but could therefore be expected to be more accurate). Marsland described experiments that measured how good these four methods are at predicting cut-nodes. Quiescence search and null-window search were indeed found to be clearly more accurate than the first two heuristics, with null-window search being the most accurate overall. Marsland also presented preliminary experiments on chess that suggest that multi-cut pruning results in a 15 to 20 point rating increase.

Wim Pijls (paper with Arie de Bruin) also considered search algorithms and how they conduct pruning, but this time from the perspective of constructing a general theory. This theory was based on the concept of *solution trees*. Given a game tree T, a *MIN solution tree* can be defined as a modified subtree with all but one branch at every MAX node pruned (this is equivalent to a strategy for MAX, and is denoted by T^-). Conversely, a *MAX solution tree* can be defined as a subtree with all but one branch at every MIN node pruned (this tree is a MIN strategy, and is denoted T^+). Pijls pointed out that one of the main objectives of game tree algorithms is to eliminate as many nodes as possible from the game tree search, and that each algorithm therefore incorporates its own cut-off criterion. By using solution trees to define the notion of *alive* and *dead* nodes, he showed that it was possible to design a general cut-off criterion underlying a theory of game-tree search. A node is alive if it lies on the intersection path of a MAX solution tree T^+ and a MIN solution tree T^-, and $g(T^- < g(T^+)$. Here, $g(T^-)$ is the minimal play-off of the closed terminal nodes in the MIN solution tree T^- and $g(T^+)$ is the maximal

pay-off of the closed terminal nodes in the MAX solution tree T^+. A node is dead if it is not alive. A general cut-off criterion can then be derived which states that alive nodes cannot be discarded, and that expanding descendants of dead nodes is useless. Pijls showed that both alpha-beta pruning and SSS* obey this cut-off criterion.

One of the general problems of search algorithms is how to deal with the problem of encountering identical nodes in different parts of the search tree. In the depth-first algorithms used in most chess programs, positions are stored in a transposition table. However, in best-first search algorithms the search tree is converted into a graph by allowing only one node for identical positions. When encountering a position for a second time, a new node is not generated, but instead a pointer is created to the identical node that is already in the search graph. This approach has an important flaw: identical nodes are not the same as identical search states. In chess, for example, castling is only legal if neither the King or Rook have moved. This problem is known as the graph-history-interaction (GHI) problem. **Jos Uiterwijk** (paper with Dennis Breuker, Jaap van den Herik and Victor Allis) presented a solution to the GHI problem for depth-first search in which only a few relevant properties need to be stored at nodes. This algorithm classifies all nodes generated during a search into *base* nodes and *twin* nodes (hence the name Base-Twin Algorithm or BTA). For any given search state, if an identical state has not been encountered previously, a base node is created; otherwise a twin node is created, with a pointer to the base node that had an identical search state. The difference with standard DCG (Directed Cyclic Graph) approaches is that the use of a pointer means that nodes arising from identical search states are no longer merged. Thus, base nodes and their twin nodes can be evaluated differently, based on their respective histories. To solve the GHI problem in games such as chess, additional information must be stored at twin nodes, such as the depth of the node in the search path and a flag to mark the node as a possible draw. With this information it can be established whether the node is an actual draw or not. Uiterwijk showed that the BTA algorithm can solve the GHI problem for proof-number search and is only slightly less efficient than other, non-reliable, DCG algorithms. Even though the BTA algorithm has only been tested for pn-search, it should be usable with any best-first search algorithm.

Another general problem with search algorithms – how to model the opponent – was addressed by **Hiroyuki Iida** (paper with Xinbo Gao, Jos Uiterwijk, and Jaap van den Herik). The minimax algorithm, like many other search paradigms, carries an implicit assumption that both players use the same evaluation function and back-up procedure. In real games, however, players often attempt to infer their opponent's intentions and try to take advantage of possible mistakes. Iida discussed his previous work on opponent modelling (OM) search, in which MAX and MIN apply minimax search using the different evaluation functions EV_{max} and EV_{min}, with MAX knowing MIN's evaluation function. In the talk, he extended OM-search to include the possibility that MIN will also search to a shallower (known) depth. This resulted in the formalisation of (D, d)-OM search, where D denotes the search depth used by the MAX player, and d is the search depth for the MIN player. For nodes below depth d+1 in a tree, (D, d)-OM search simply backs up the values of EV_{max} in the normal minimax fashion. Above this depth, however, (D, d)-OM search can make use of its model of the opponent. Specifically, the values of EV_{min} are also backed up in the tree. Then, when backing up MAX's evaluations above depth d+1 the values selected at MIN nodes are not the minimum values, but the values along the branches chosen by MIN when backing up EV_{min}. Since the values selected in this way must be greater than or equal to the actual minimum values at each MIN node, (D, d)-OM will never give a result less than the plain depth D minimax value of EV_{max}. Iida also explained how to implement (D, d)-OM efficiently by using alpha-beta pruning on the search below depth d+1, and beta-pruning on the search above depth d+1. He gave test results demonstrating the effectiveness of this technique on random tree simulations and in the game of Othello.

At CG'98, there was just a single presentation on games with imperfect information. In this talk, **Ian Frank** (paper with David Basin) considered the *best defence* model of a game. In this model, it is assumed that the opponent (MIN) has perfect information and can choose a strategy with full knowledge of the strategy selected by MAX. These assumptions are often used by human experts in the analysis of real games (such as Bridge), because finding optimal strategies against best defence provides a lower bound on the pay-off that can be expected when the opponents are less informed. Frank's talk presented a proof that finding optimal strategies for the best defence model is NP-complete in the size of the game tree. Thus, arbitrary (imperfect information) *trees* are hard to analyse against best defence. This is in contrast to perfect information games, where arbitrary trees are easy to analyse (the minimax algorithm runs in time linear in the size of the tree), but arbitrary *games* are hard to analyse (at least when their definitions generate trees of exponential size). Frank has previously worked

on the heuristic of *payoff-reduction* for efficiently finding optimal strategies against best defence. In his CG'98 talk, he went on to introduce two new heuristics that (like pay-off reduction) work by improving the modelling of the dependencies in the search process. Test results were given for single-suit Bridge problems showing that a combination of heuristics solved 648 out of the 650 problems from the *Official Encyclopedia of Bridge*. For comparison, using Monte-Carlo sampling (the main other viable approach) solved just 430 problems. Furthermore. since the new heuristics even revealed nine errors in the test set itself, they actually performed better than the human experts who produced the model solutions.

2. LEARNING

A total of five presentations concentrated on the theme of learning in games. In the first of these, **Don Beal** (paper with Martin Smith) described how temporal-difference learning can be used to learn shogi piece values for material evaluation functions. In previous research, the authors have investigated how to do this for chess, automatically learning an ordering of Pawn, Knight, Bishop, Rook, Queen, with strengths similar to the assessments made by human players. Shogi, however, is a very tactical game, which varies in character distinctly in the opening, middle game, and endgame, so it is not obvious that single values will apply over the whole game. Nevertheless, values can be learned, and runs over 6000 games gave the following consistent ranking: Pawn, Lance, Knight, Silver, Gold, Bishop, Rook. To test the effectiveness of these learned values, a series of matches was played between identical search engines that used different piece values. Three alternative sets of values were compared: one taken from the program YSS (the winner of the 1997 Computer Shogi Championships), one from the widely available GNU SHOGI program, and a final set constructed by an examination of the shogi literature. That these rankings were all different is itself an illustration of the shogi community's lack of concensus on the correct values. However, in Beal's tests, the learnt values performed best, scoring 55%, 60% and 62% against the three other data sets. Beal emphasised that the learnt values were produced entirely with self-learning, and that no initial shogi knowledge was required. The results also showed that deep searches are not necessary to learn piece values. He closed by suggesting that the research frontier is now the learning of complete evaluation functions.

This frontier challenge was taken up by **Michael Buro**, whose paper discussed "the semi-automatic construction of evaluation functions". His proposal was a Generalised Linear Evaluation Model (GLEM), in which features are linearly correlated with winning in a game. In typical evaluation functions, there is a small number of hand-crafted features, and the weights are often also chosen manually. This can be a time-consuming process, and the small number of features may limit the evaluation quality. Thus, the GLEM is a generalisation that allows an evaluation function to give weights to *conjunctions* of atomic features, rather than just a linear combination of hand-chosen features. A basic atomic feature set is first defined by hand, and compound features are then built up automatically by a process of repeated specialisation. For any given feature, this specialisation process continues until the number of matching configurations in the training data set drops below some pre-set threshhold number. The GLEM has a number of attractive properties, such as allowing the approximation of non-linear effects by the combination of conjunctions. Yet, since the core is linear, all the weights can still be computed efficiently. As an example of this efficiency, Buro described how an evaluation function for the game of Othello was learned in just 30 hours on a Pentium II CPU, despite having 1.5 million weights. A "very similar" evaluation function was used in the LOGISTELLO program that beat the human Othello World Champion 6-0 in August 1997.

Two further presentations on learning considered the problem of move selection in the game of Go. The first of these, by **Nobusuke Sasaki** (paper with Yasuji Sawada and Jin Yoshimura) examined how a neural network could be used to learn solutions to next-move problems in *tsume-go*. This is a special type of Go problem in which the task is to kill a single group of the opponent's stones in a given setting. A simple three-layer network was trained using 2000 problems extracted from several tsume-go books. These problems were all given on a 9×9 board, so the network had 162 inputs (to indicate for each point whether it contained a black or a white stone) and 81 outputs (one for each point). The network was trained using back-propagation with 300 hidden nodes, and then tested on a number of new test sets, containing a total of about 1400 unseen problems. For these problem sets, the number of times the correct move was contained in the top three moves suggested by the network was between 55% and 65%. Since other researchers had previously used the same data sets to investigate human performance on tsume-go problems, Sasaki was able to estimate the network's performance as approximately

equivalent to that of human players at a one-dan level.

A related talk, by **Takuya Kojima** (paper with Atsushi Yoshikawa) tackled the same problem of move selection in tsume-go, but used it mainly as motivation for the introduction of a two-step model of pattern acquisition. In this model, the production of the actual patterns is separated from the acquisition of the conditions that dictate when to apply them. Kojima argued that this separation makes it easier to compare algorithms for each of the steps. He then described some algorithms (three for the production of patterns, and two for the acquisition of conditions), and examined their performance on tsume-go problems. On the same data set as mentioned above when reviewing Sasaki's talk, Kojima's best algorithm found the correct move in 31% of the problems, rising to 62% if the first three move suggestions were examined. Again, this is roughly equivalent to a human one-dan level performance. Kojima further investigated whether the number of possible moves in a problem was correlated with the algorithm's performance. However, even for a data set constructed of only the problems with more than ten possible moves, the performance rate was some 29%. Kojima therefore suggested that a good application of this pattern-matching technique would be in tandem with a search algorithm, as a method of pruning at nodes with large numbers of branches.

The notion of using multiple techniques in concert was discussed in detail in the final learning talk, given by **Daniel Borrajo** (paper with Ignacio Giráldez). Borrajo suggested that, since problem solving is a multi-faceted task, and there have been many attempts at problem solving in games, it may be profitable to combine different approaches together. In particular, he suggested that it may be possible to do this within the framework of a multi-agent system. The goal of such a system would be to have multiple agents that learn how to cooperate together to play a game. We heard the details of a system that implemented these ideas by coordinating four distinct types of agent: *a referee*, *workers*, *advisors*, and *trainers*. The worker agents consist of three learners (based on C4.5, back-propagation, and Bayesian learning), and two alpha-beta workers (one that carries out plain alpha-beta, and a second variant that uses one of the three learners to generate its leaf-node evaluations). This system is allowed to play games under the supervision of a single referee agent, with a single trainer agent directing the choice of game instances, and every worker agent generating suggestions for every move. A set of advisor agents then decides which of the suggested moves is actually the best, and the worker agents' behaviour is reinforced when they agree with the advisors (or if they disagree with the advisors and the result is a loss). By learning when each worker agent is most likely to be correct, the system should acquire the ability to select between them at the right times in real games. A test of this system on the game of checkers showed that the cooperative use of the five worker agents indeed outperformed any of the agents operating individually. For instance, the five agents together had a 4% winning advantage over plain alpha-beta search. In fact, the five agents working together still primarily opted for the move selected by one of the alpha-beta workers, but deviated sufficiently to produce a measurable advantage.

3. MATHEMATICAL ANALYSIS

One of the main themes of CG'98 was the relation between mathematics and games. There were four contributions in which this theme was especially present. In the first of these, **Aviezri Fraenkel** (paper with Dmitri Zusman) proposed a new heap game. In heap games, players take turns to remove tokens from a number of heaps, with the player who can take the last token winning. The most famous of these heap games is *Nim*. Fraenkel proposed an extension of Wythoff's game (a two-heap game where tokens can be removed from one or both heaps). In the extended new heap game, there are k finite heaps, and a move is either to remove a positive number of tokens from a maximum of k-1 heaps, or to remove the same number of tokens from all k heaps. For this game, Fraenkel stated and proved a theorem showing how the winning positions can be characterised by a simple mathematical formula. Using this theorem, a player can tell whether it is possible to win from any given position. Furthermore, the proof of the theorem shows how to compute such a winning move if it exists. Using these results, then, a strategy for the game can be computed. Fraenkel showed that this strategy can be found in polynomial time.

In a second contribution, **Aviezri Fraenkel** (paper with Ofer Rahat) turned to a more general class of combinatorial games, the infinite cyclic impartial games. In an impartial game, both players have the same set of moves when they are allowed to play next in any given position (the heap game Nim is impartial, but chess is not impartial because the players can only move their own pieces). In cyclic games there is also the possibility of returning to the same position (this includes pass moves, which

are cycles of length 0). The infinite cases of these games have only been briefly discussed in the literature. Infinite cyclic impartial games can be represented by infinite digraphs, and Fraenkel defined a subclass of infinite digraphs called *locally path-bounded digraphs*. A path is a sequence of vertices in a graph that are all distinct; a graph is path-bounded if all paths in the graph have a length smaller than a certain upper bound, and a cyclic digraph is locally path-bounded if for every vertex in the graph there is an upper bound on the length of all paths from the vertex. In games that can be described by locally path-bounded digraphs it is possible to get draws because of cycles. Fraenkel showed that for this subclass of infinite digraphs, an optimal strategy can be computed and that a win (if possible) can be realized in a finite number of moves. To do this, he first showed that the *Generalised Sprague-Grundy function* (the γ-function) exists on such digraphs with finite, though not necessarily bounded, values. To actually use this function to decide which moves to make, a counter function has to be defined that counts the length of the paths from every node. If the γ-function gives the same value for different vertices from the same node n, the counter function makes it possible to distinguish between them. (If $\gamma = \infty$, then the path from n is a cycle and the game is therefore a draw.) Fraenkel explained that his proof is only correct if the graph has leaves. However, a large part of combinatorial game theory is concerned with such digraphs.

An application of combinatorial game theory to the game of Go was presented by **Bill Spight**. He noted that when Go players evaluate a local position in Go, they are most interested in the territorial count and the value of a local play. Combinatorial game theory is applicable here because many Go positions are combinatorial games where the mean value of the game corresponds to the count and the temperature corresponds to the value of a play. The technique of thermography can be used to find these mean values and temperatures, but only for positions without cycles. However, positions with cycles do arise in Go, a common example being the case of *ko fights*. Berlekamp has found a method to break cycles in the case of a single ko, but this is not applicable when more than one ko is active in a position. Spight's contribution was to give a method to deal with multiple kos inside the combinatorial game framework. He did this by introducing *environment plays* into the thermograph of a combinatorial game. An environment play is a play outside the local game. To utilise this notion, such plays are first introduced to a local game by translating the combinatorial game tree into an AND/OR tree in which the environment plays are made explicit. Then, this AND/OR tree can be used to construct the thermograph of the combinatorial game. In this way, it is in theory possible to locally evaluate any Go position, and to explore relations between kos and threats.

Unfortunately, **Marcel Crâsmaru** was not able to attend the conference. However, his paper deals with the question of why the programming of Go can be expected to be difficult. Specifically, he proves that a restricted subclass of life-and-death problems in Go is NP-complete. In this subclass there is always a unique good move for one of the players and only two good moves for the other player. Since the restricted case is NP-complete, the general problem of deciding whether a group is alive or dead in Go is also NP-complete. It is already known that playing sums of (even relatively simple) combinatorial games is NP-hard. Go endgames are naturally thought of as such sums of combinatorial games, so the significance of Crâsmaru's proof is that even if an exceptionally good Go heuristic can be found to reduce a game of Go to a set of independent life-and-death problems, there will still be positions in which the complexity of the search is NP-complete.

4. FURTHER ASPECTS OF GO

This section covers four presentations that dealt with further aspects of the game of Go. In the first of these, **Morihiko Tajima** (paper with Noriaki Sanechika) introduced a new way of estimating the strength of groups of stones. This is important in Go because group strength plays a crucial part in the global evaluation of positions, and yet is in general too expensive to calculate exactly. The solution suggested by Tajima was to use the number of possible omission moves (*tenuki* in Japanese) to estimate group strength. Specifically, the PON (*Possible Omission Number*) was defined as the number of consecutive moves that a player requires to change the status of a group from dead to alive or from alive to dead. Exact calculation of the PON for each group on a Go board is expensive, so Tajima proposed a method that calculates an *estimate* of the PON using the number of liberties (*dame*) and the rub numbers (*kosuri*) of stones in a group. This PON estimate can be computed efficiently with just a small number of counting rules, for example as a simple linear function between the n-th dame (n = 1,2,3,4). We heard how weights can be used to capture the difference in importance of three types of points (edge points, non-edge points with one kosuri, and other points), and how experiments had been

carried out to find good values for these weights. Tajima argued that the simplicity of the resulting evaluation makes it very useful in positions where the groups on the board do not have excessive interactions, i.e., in the opening and middle game.

A contrasting approach to computer Go was suggested by **Julian Richardson** (paper with Steven Willmot, Alan Bundy, and John Levine). As an alternative to the established techniques of using pattern matching and complex evaluation functions, he instead described a goal-based *adversarial planning* approach that employs plan schemas to describe how goals can be split. Each planning operator has an abstraction level, and applying a planning action cannot increase the level of abstraction. Abstract goals are refined down to concrete moves as soon as possible, and then control is passed to the other player, to look for a refutation of the goals in the plan. Thus, the search is carried out largely in terms of knowledge, not in moves, and open positions can be handled. Richardson suggested that it is important for Go programs to recognise the advantages of each of these approaches and switch between them when they are most appropriate. He also noted that a useful side-effect of the high-level formalisation of goals is a declarative formalisation of Go knowledge. One use of this knowledge would be to generate automatic explanations of the program's moves. Finally, he described test results that showed that a system based on this adversarial planning technique was able to solve 74% of a test set of 85 problems extracted from volume I of *Graded Go Problems for Beginners*.

A different aspect of Go research was examined by **Atsushi Yoshikawa** (paper with Takuya Kojima and Yasuki Saito), who presented the latest results in their investigation of the cognitive aspects of Go. The authors use Go as a domain for investigating how humans acquire knowledge in semantically rich domains. In previous studies they have examined how humans learn high-level Go terms at the same time as learning Go patterns. An interesting illustration of this is the skill of *mental reconstruction* of games. Unlike in chess and shogi, good Go players have difficulty in reconstructing a game by just hearing the moves. However, supplementing the Go moves by high-level terms (such as "...player A cuts at (3,8)...") makes visualisation much easier. In the CG'98 presentation, Yoshikawa described three experiments on the terms that are used by Go players. The first two involved vocabulary tests using commonly occurring words collected from books and magazines, and the third involved *soudan-Go*, in which two players form a team that can discuss a game as they play together against another two-player team. Based on these experiments, Yoshikawa postulated an *iceberg* model to explain the difference in usage of Go terms between beginners, intermediate players, and experts. In this model, each Go term has a lexical meaning that is easy to describe and to find in dictionaries. This is the portion of the iceberg "above the water" that even beginners can identify with enough study. The part of the iceberg "below the water" is emergent reasoning: when players understand how a Go term relates to a given board position they update their knowledge of the term with this information. Yoshikawa argued that this structure explains why misunderstandings do not occur between experts when describing a Go game, even though they tend to use abstract words instead of actual Go terms.

A fitting way to end this section was provided by **Martin Müller**, who gave a presentation that discussed the future of research on computer Go. Superficially, the field appears very healthy, with over 100 active programs being developed (including about ten that are commercial), and a number of computer tournaments (two of which have the status of World Championships). However, Müller identified some problems. For example, he estimated that to produce a full-scale program takes five person years, and cited a number of obstacles to the faster development of programs. Two of the main obstacles are the lack of publicly-available source code, and the way that commercial interest hinders the distribution of knowledge so that only a few of the top programmers have an interest in publishing their methods. Another serious problem is that a typical program has between 50 and 100 modules. Müller pointed out that it is generally the *weakest* of these modules that determines the strength of the program; the best programs *usually* play good, master level moves, but the performance over a full game is much lower, and in most games the programs will make a number of large blunders. Thus, the current state of computer Go is that 100 programs are being developed in largely the same way, with all of the program developers trying to improve the level of their worst module. Müller made a number of suggestions for improving the efficiency of this research effort. First and foremost would be to increase the level of cooperation between researchers so that a complete implementation of a standard model could be achieved. A basis for this standard model already exists in the form of David Fotland's identification of 60 basic modules. Müller also pointed out that a better understanding of program performance is important. For example, one way to establish an upper limit on current program performance would be to carry out Dreihirn experiments in Go. (These experiments involve allowing a human player to play a game by selecting each move from among the suggested moves made by a

number of different computer programs.) [See p. 247 of this issue. – Ed.] Finally, to truly push the limits of current technology, a large-scale programming project would be necessary. This would be a major software-engineering project, starting from an existing program. Müller argued that this would provide an important milestone and reference point to better understand the strengths and weaknesses of the standard model.

5. ANALYSIS OF THE HISTORICAL DEVELOPMENT OF GAMES

The contribution presented by **Jin Yoshimura** (paper with Hiroyuki Iida, Kazuro Morita, and Jos Uiterwijk) is hard to place into any category, so we have given it one of its own. It addressed the possibility of using computers to reconstruct the historical development of a game. The game in question was shogi, and Yoshimura took us back to the version of the game played during the Heian period in Japan (from A.D. 800-1200). There is no definitive record of the rules or of the exact board size of Heian shogi (there seems to have been both an 8×8 and a 9×8 version). An ancient record called the Nichureki suggests that the rules of the game were similar to modern shogi modulo the absence of the Rook and Bishop and the inability to drop captured pieces. However, there is no clear rule for the end of a game of Heian shogi, so Yoshimura assumed a set of possible rules: (1) If a player has no legal moves, he loses, (2) The player who takes all of the opponent's pieces wins the game, (3) If both players have only a King, the game is drawn. These rules are clearly inconsistent, so Yoshimura split them into two consistent sets: one that he argued was similar to the rules of Heian shogi, and another that was closer to those of modern shogi. Yoshimura also gave the arguments of historians about the development of Heian shogi into modern shogi. These arguments considered the role of complexity (games should allow many strategies but not take too long to play), the aesthetics of board size (square boards are more appealing), and the desirability of being able to play the game to a definite conclusion (as few games as possible should end in draws). Yoshimura then tried to give a new insight into the evolution of modern shogi from Heian shogi by analysing the possible board sizes of Heian shogi. A close game of Heian shogi naturally ends in a King and Gold versus King endgame, so Yoshimura used a computer to examine the number of non-trivial draws in such endgames. This analysis demonstrated that on $N \times N$ boards of size up to 10×10 there are no non-trivial draws, whereas for any N greater than 10 non-trivial draws constitute the large majority of board positions. He put this forward as a reason that 9×9 would have been the largest board to survive into the next shogi variation, since 10×10 itself suffers when judged on the aesthetic criterion of preferring the symmetrical placement of pieces on either side of the King. Historians have already pointed out that for 9×9 Heian shogi there is a simple non-losing strategy, which explains why features such as the Rook and Bishop and the re-use of pieces would have been introduced soon after changing the board size to 9×9.

6. SIDE EVENTS: COMPLEX GAMES LAB WORKSHOP & SOCIAL EVENTS

CG'98 was well supported by side-events, which included a reception, games tournaments, and a party. A half-day workshop was also organised by the ETL Complex Games Lab on the afternoon before the conference. This workshop was essentially a scaled-down version of the Game Programming Workshops (GPW) coordinated in Japan from 1994 to 1997 by ETL's Hitoshi Matsubara. In fact, it was partially the success of these GPW events that helped make it possible to stage CG'98 in Japan. With CG'98 making a full GPW unnecessary in 1998, the half-day Complex Games Lab workshop avoided breaking the GPW series completely. As in the GPW events, the official languages for the workshop were English and Japanese. This year, however, all the papers were written in English, and half the speakers also spoke in English. Speakers came from institutions within Japan and also from Hong Kong and Israel and were rewarded with a large audience, since almost all the CG'98 participants also attended the workshop. We briefly summarise each of the workshop papers below, and direct those readers interested in the full papers to the electronic versions at http://www.etl.go.jp/~7236/Events/workshop98.

- The first talk was by **Hiroto Yoshii**, who gave a presentation on pattern recognition in Go. He proposed a new pattern recognition algorithm called METS (Move Evaluation Tree System), which builds pattern trees from large databases of expert moves. The algorithm has been used in the Go program MONKEY JUMP.

- **Shinichi Sei** (paper with Toshiaki Kawashima) explained aspects of the KATSUNARI Go program.

This program combines a large database of patterns with a local capture search to suggest candidate moves. As in Yoshii's program, the fundamental reliance on pattern databases for move evaluation is what sets it apart from most other Go programs.

- **Aviezri Fraenkel** gave a presentation in which he discussed how the representation of a game can affect the efficiency with which optimal strategies can be found. In particular, he gave as an example a two-heap game that represents a generalisation of Wythoff's game.

- **Robert West** discussed the scissors-papers-stone game. Although the game-theoretical optimal strategy for this game is to play randomly, West pointed out that humans are normally quite bad at generating random numbers. By treating the game as a very simple distributed system with neural-network players that look for sequential dependencies, he showed how emergent properties not inherent in the individual players can arise.

- **Toshinori Kasuga** (paper with Tsuyoshi Suzuki and Yoshiyuki Kotani) described experiments on evaluating the artistic value of a tsume-shogi problem. The weights of a linear evaluation function with fifteen hand-picked features were trained using least-squares on 80 tsume problems and tested on 30 unseen problems.

- **Takeshi Ito** gave a presentation about modelling how players learn about a game during the post-mortem analysis (called *kanso-sen* in Japanese). He suggested that learning is localised in three stages: the generation of candidate moves, the narrowing of the number of candidates, and selection of the next move.

- **Tsuyoshi Suzuki** (paper with Nobuo Inui and Yoshiyuki Kotani) presented a technique for increasing the speed of shogi programs by generating candidate moves incrementally. His preliminary results suggest that the technique may be of use for programs searching more than 10,000 positions a second.

- The final workshop presentation was by **Ayumu Nagai**. He gave a presentation about using proof-number search for solving mating problems in shogi (tsume-shogi). His algorithm in theory works better than Seo's tsume-shogi solver mentioned in Section 1, but there are still problems with efficient implementation.

Directly after the workshop – on the evening before the conference – a reception was held at the Electrotechnical Laboratory. Following this reception, an informal games tournament was staged, organised by Takenobu Takizawa. There was the possibility of playing Go and shogi, but the main event was Amazons, a novel game that combines features of both chess and Go. Of the 19 participants, three players remained undefeated at the end: N. Nagumo (3-0), Y. Kajihara (3-0) and H.J. van den Herik (1-0). Each of these winners received a small prize.

After the first day of the conference, there was a conference party with drinks, sushi and Chinese food. The party was opened by Jaap van den Herik, and although it officially ended at about eight o'clock, unofficially it was continued at the participants' main hotel until well after midnight.

7. CONCLUSIONS

CG'98 was a successful and highly worthwhile event. A second CG conference will take place in the year 2000 (exact date and location to be announced). In 1999, the series of Game Programming Workshops in Japan will be resumed.

The CG'98 proceedings will be published in the spring of 1999 by Springer-Verlag, as a volume in the Lecture Notes in Computer Science (LNCS) series. Around 10 papers will also be selected by the Program Chairs to appear in a special issue of the mathematical games section of the journal Theoretical Computer Science. These papers will be substantially extended from the conference versions.

8. REFERENCES

Conference papers

Beal, D.F. and Smith, M.C. *First Results From Using Temporal Difference Learning in Shogi*.

Björnsson, Y. and Marsland, T.A. *Multi-Cut Pruning in Alpha-Beta Search*.

Breuker, D.M., Herik, H.J. van den, Allis, L.V., and Uiterwijk, J.W.H.M. *A Solution to the GHI Problem for Best-First Search*.

Buro, M. *From Simple Features to Sophisticated Evaluation Functions*.

Crâsmaru, M. *On the Complexity of the Tsume-go*.

Fraenkel, A.S. and Rahat, O. *Infinite Cyclic Impartial Games*.

Fraenkel, A.S. and Zusman, D. *A New Heap Game*.

Frank, I. and Basin, D. *Optimal Play Against Best Defence: Complexity and Heuristics*.

Gao, X., Iida, H., Herik, H.J. van den, and Uiterwijk, J.W.H.M. *A Speculative Strategy*.

Giráldez, J.I. and Borrajo, D. *Distributed Decision Making in Checkers*.

Grimbergen, R. *A Survey of Tsume Shogi Programs Using Variable Depth Search*.

Junghanns A. and Schaeffer, J. *Relevance Cuts: Localizing the Search*.

Kojima, T. and Yoshikawa, A. *A Two-Step Model of Pattern Acquisition: Application to Tsume-go*.

Müller, M. *Computer Go: a Research Agenda*.

Pijls, W. and Bruin, A. de. *Game Tree Algorithms and Solution Trees*.

Sasaki, N., Sawada, Y., and Yoshimura, J. *A Neural Network Program of Tsume-go*.

Spight, B. *Extended Thermography for Multiple Kos in Go*.

Tajima, M. and Sanechika, N. *Estimating the Possible Omission Number for Groups in Go by the Number of n-th Dame*.

Willmott, S., Richardson, J., Bundy, A., and Levine, J. *An Adversarial Planning Approach to Go*.

Yoshikawa, A., Kojima, T, and Saito, Y. *Relation Between the Way of Using Terms and the Skill: from analysis of protocols of the game of Go*.

Yoshimura, J., Iida, H., Morita, K., and Uiterwijk J.W.H.M. *Retrograde Analysis of the KGK Endgame in Shogi; its implication for ancient Heian Shogi*.

Workshop papers

Fraenkel, A. *Succinct Games and Exotic Numeration Systems*.

Ito, T. *How players learn at KANSO-SEN*.

Kasuga, T., Suzuki, T. and Kotani, Y. *Evaluation of Tsume-shogi with the Method of Least Square*.

Nagai, A. *A new AND/OR Tree Searching Algorithm Using Proof Number and Disproof Number*.

Sei, S. and Kawashima, T. *Memory-Based Approach in Go-program Katsunari*.

Suzuki, T., Inui, N. and Kotani, Y. *Incremental Generation of Possible Moves in Shogi*.

West, R.L. *Zero Sum Games as Distributed Cognitive Systems*.

Yoshii, H. *Move Evaluation Tree System*.