1. Introduction – Artificial Intelligence and Games

Because playing games is one of the intellectual behaviors of human beings, games have been used as a representative testbed to conduct research in Artificial Intelligence (AI). Games are ideal domains for AI researchers to investigate research ideas, because of the structural simplicity and the combinatorial complexity of the domain, and for the challenge of outperforming the best humans. Additionally, the techniques developed in one game are not only easily transferred to another game but also are often adapted to a wide range of applications such as path-finding in robotics and DNA sequence alignment in bioinformatics.

Chess and Go are popular games that have fairly long histories. These games are played by two players and belong to the so-called two-player zero-sum games with perfect information. Zero-sum indicates that one player wins if the other player loses. Perfect information indicates that both players always know the entire state of the game while they are playing the game. I will restrict my attention to this kind of games in this abstract.

Practically all game-playing systems such as chess-playing programs employ look-ahead search, which represents a procedure for investigating game states of multiple moves ahead, to improve their move decisions. Over the last 50 years, improving the efficiency of search algorithms has been one of the main subjects of game research, because efficient search algorithms improve the strength of programs.

2. AND/OR Tree Search and Games

One of the crucial procedures in games is to find a winning way from a given game state. To prove a win for a player, one has to prove that at least one of the moves of the player leads to a win (i.e., OR procedure), as well as that all the moves played by the opponent lead to losses for the opponent (i.e., AND procedure). Therefore, this process can be seen as AND/OR tree search by mapping a game state and a move to a search node and a branch of an AND/OR tree, respectively. Many high-performance game solvers employ AND/OR tree search algorithms to find winning ways. Two properties are required for such solvers:

1. Efficiency: It is necessary to return solutions as quickly as possible.
2. Correctness: When a solver returns solutions, these solutions must always be reliable.

Researchers have successfully exploited efficient search algorithms. In particular, Nagai’s depth-first proof number (df-pn) search algorithm [5] using the notion of proof and disproof numbers [1] has been shown to be very effective in many domains. Formally, the proof (or disproof) number of node \( n \) is defined as the minimum number of unexpanded nodes that must be expanded to prove that \( n \) is a win (or loss) for the first player. Proof and disproof numbers are good heuristic estimates of the difficulty of finding
a solution in a partially expanded AND/OR tree, i.e. the smaller the (dis)proof number of
n is, the easier it looks to prove that n is a win (loss) for the first player.
One issue that must be addressed is that many AND/OR tree search algorithms have a
problem about correctness. One of the most valuable enhancements in a solver is the use
of the transposition table, a large cache that keeps results of previous search efforts.
Since many search spaces are graphs, not just trees, there are often many different paths
that lead to an identical node. Typically the solver saves in/retrieves from the
transposition table a search result by ignoring paths that the solver takes. Thus the
transposition table can reduce duplicate effort by a large margin. However, many games
involve repetitions and their outcomes are determined based on what is called the
repetition rule. For example, three-fold repetition is a draw in chess, and the situational
super-ko rule of Go declares that a move leading back to the previous game state is illegal.
AND/OR tree search algorithms enhanced with the transposition table suffer from the
Graph History Interaction (GHI) Problem [2], which causes the algorithms to occasionally
return incorrect results. In practice, programmers have either ignored the GHI problem,
since they did not want to degrade the performance of their solvers, or reduced the
number of recognized transpositions in order to guarantee correctness.

3. Correct and Efficient AND/OR Tree Search in the Presence of Repetitions

In this talk, I will present techniques behind df-pn(r), which is the major contribution of
my research [3]. Df-pn(r) is a high-performance AND/OR tree search algorithm that
guarantees correctness in the presence of repetitions. More specifically, my contributions
are summarized as follows:
1. I have developed a novel solution to the GHI problem. Compared with the previous
solutions, my GHI solution is so general and practical that it is applicable to many
games and algorithms while incurring only a very small overhead. I have incorporated
the GHI solution into df-pn(r).
2. I found empirical evidence that repetitions cause not only the GHI problem, but also a
performance degradation issue for df-pn. Df-pn(r) cures this problem.
3. I have successfully developed high-performance solvers based on df-pn(r) for several
games, such as tsume-shogi (check-mating problems in shogi, Japanese chess-like
game), tsume-Go (life and death problems in Go), and checkers. In particular, df-pn(r)
contributed to not only developing the currently best tsume-Go solver [4], but also
solving the game of checkers that has roughly $5 \times 10^{20}$ possible game states [6].

References
problems in Go. In Proceedings of the 20th National Conference on Artificial Intelligence
(AAAI’05), pages 1374-1379, 2005.
thesis, Department of Information Science, University of Tokyo, 2002.