Pruning Nodes in the Alpha-Beta Method Using Inductive Logic Programming

Nobuhiro Inuzuka, Hayato Fujimoto, Tomofumi Nakano and Hidenori Itoh
Department of Intelligence and Computer Science,
Nagoya Institute of Technology,
Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan.
{inuzuka,psycho,tnakano,itoh}@ics.nitech.ac.jp

Abstract

This paper reports a preliminary research of application of inductive logic programming for pruning nodes in the alpha-beta game-tree search method, which finds an appropriate move by looking game-trees in a limited depth ahead. The alpha-beta method reduces the number of nodes by keeping and updating lower and upper bounds of static evaluation. Pruning effect depends on an accidental order of nodes visited, because we can expect large pruning after we have large update. This paper proposes a method to learn rules to sort nodes to yield effective pruning, by using inductive logic programming framework. The method induces a logic program of a binary relation among nodes, and sorts nodes based on the relation. We inspected the method with the game othello.

1 INTRODUCTION

Inductive logic programming (ILP) is an inductive learning method in the framework of logic programming, which involves the first-order logic formalism. ILP tries to find a logic program, called a hypothesis, that explains given examples with respect to also given background knowledge. The examples and the background knowledge are described in logic programs. ILP has been evidenced to work for many classification problems in complex structural domains, such as molecular biology and natural language processing[De Raedt, 1995, Muggleton, 1992]. The field of game also supplies problems for ILP research. Many ILP methods are inspected by applying to the chess domain (for example, [Fürnkranz, 1996, Quinlan, 1990, Morales, 1992, Morales, 1997, Bain & Srinivasan, 1995]). Chess is recognized as a suitable domain for ILP because data in the domain, such as game configuration, have complex structure and are expressed better in the first-order logic formalism.

Our previous work [Nakano et al., 1998] challenged to induce game heuristics to evaluate game configurations in the domain of shogi, or Japanese chess. In the work we reduced the heuristics induction problem into a simple classification problem, which classifies instances of a relation better-choice. better-choice(b, m_1, m_2) is true when a move m_1 is better than a move m_2 to be taken for a game configuration b. We proposed a method to constitute a total order relation from a relation defined as the better-choice relation, which is not even partial order, induced by an ILP algorithm from instances collected from experience of solving shogi problems. The method, then, constitutes static evaluation for game configuration, the static evaluation which can be used in search algorithms.

In this paper we apply the method using better-choice relation to increase pruning effect in the alpha-beta method, which is a lookahead method for game-trees. The alpha-beta method reduces the number of nodes by keeping and updating lower and upper bounds of static evaluation. Pruning effect depends on accidental order of visited nodes. We try to induce a rule to sort visited nodes to yield effective pruning, by using the method of better-choice relation and ILP.

The following section revisits the alpha-beta method and detects conditions that nodes are pruned during execution of the method. Then, we specify our ILP problem. Section 3 gives a procedure to apply induced rules, logic programs of the better-choice relation, to execution of the alpha-beta method. Then, in Section 5, we report experimental results of the proposed method in othello. Section 6 discusses merits of the method and remained issues.

*Partially supported by the Hori Information Science Promotion Foundation.

†Currently working for Sumisho Computer Systems Corp., psycho@mbc.sphere.ne.jp.

No license: PDF produced by PStill (c) F. Siegert - http://www.this.net/~frank/pstill.html

Table 1: The Mini-Max Method.

Choose a move from a given board b of the first player's turn:

1. Construct a game-tree of depth d, i.e., a tree which has a root b, has depth d and consists of
   (a) as a node, a board configuration, and
   (b) as an arrow from one board b_1 to another b_2, a move m which brings b_1 to b_2.
2. For each leaf, calculate its static evaluation.
3. Give each board of the first player's turn the maximum evaluation of its children nodes as its evaluation.
4. Give each board of the second player's turn the minimum evaluation of its children nodes as its evaluation.
5. Choose a move or a branch from the board b that leads to the board with the maximum evaluation, as an appropriate move.

Table 2: The Alpha-Beta Method.

alpha-beta(b, α, β)

1 If b is at depth bound, return the static evaluation of b; else continue.

2 Let $b_1, \cdots, b_n$ be the successors of b, set $k := 1$, and if b is a MAX node, go to step 3 ; else go to step 3′.

3 Set $\alpha := \max(\alpha, \text{alpha-beta}(b_k, \alpha, \beta))$.

4 If $\alpha \geq \beta$ return β ; else continue.

5 If $k = n$ return α ; else set $k := k+1$ and go to step 3.

3′ Set $\beta := \min(\beta, \text{alpha-beta}(b_k, \alpha, \beta))$.

4′ If $\alpha \geq \beta$ return α ; else continue.

5′ If $k = n$ return β ; else set $k := k+1$ and go to step 3′.

2 THE ALPHA-BETA METHOD AND AN ORDER INDUCTION PROBLEM

We assume a strategic, complete information, deterministic game played by two persons, such as chess and othello. Two players, called the first and the second players, take moves by turns. An player wins when a move of its turn brings a status of game to satisfy a given condition. We call the status a goal.

The alpha-beta method is an algorithm to find an appropriate move. The appropriate move is the best in a sense of looking a restricted depth of game-trees ahead and of a given heuristic function. The algorithm visits only necessary nodes or prunes unnecessary nodes by using property of game-trees. In the rest of this section we revisit the mini-max method and the alpha-beta method.

We assume a static evaluator for board configuration. The evaluator estimates the quality of board from the side of the first player by a number, that is, the first player hopes large numbers and the second player hopes small numbers.

The mini-max method chooses an appropriate move for a board b of the first player's turn using a lookahead procedure. The method is shown in Table 1. It takes lookahead for all boards that can be reached in a given depth from a current board.

The alpha-beta method, shown in Table 2, calculates static evaluation and finds an appropriate move as the mini-max method does, but prunes unnecessary boards. It uses two variables α and β to keep lower and upper bounds of evaluation. By examining successor nodes of a node b,

- the lower bound α is updated, if b is a MAX node or of the first player's turn, and
- the upper bound β is updated, if b is a MIN node or of the second player's turn.

There are two cases when the method prunes nodes:

Case 1 At a board of the first player's turn, if a successor node gives a larger evaluation than the upper bound β, other successors need not to be visited.

Case 2 At a board of the second player's turn, if a successor node gives a smaller evaluation than the lower bound α, other successors need not to be visited.

Because of these pruning conditions, the order of nodes is important, the nodes which are successor nodes of the node b_k which the procedure is called with. That is, the larger update for bounds is made, the larger the possibility to have pruning has. To control the order for effective pruning has been a topic of the alpha-beta method[Nilsson, 1998]. For example, an easy way to control the order is to sort successor nodes by the static evaluation.

We apply a relational learning approach, or an ILP approach, to sort successor nodes, that is, we try to induce rules to sort nodes in order to have effective pruning. We expect the following advantages of this approach:

- ILP is enough powerful to deal with complex structural data in domains of games. A board configuration is expressed using a list whose elements are terms expressing pieces (i.e., their po-

No license: PDF produced by PStill (c) F. Siegert - http://www.this.net/~frank/pstill.html

sitions and types). Moves made by players are also expressed by first-order terms. These complex data are dealt with first-order expressions.

- We may have several rules to sort nodes, depending on board configurations. First-order formulae can describe rules that evaluate moves depending on a parameter of board expressed by complex first-order terms.
- Induced rules are easily readable and modified, because of the readability of first-order formulae.

3 AN ILP METHOD WITH THE better-choice RELATION AND ORDER INDUCTION

We have proposed a method of order induction based on a binary relation better-choice in the previous works [Inuzuka at al., 1997a, Inuzuka et al., 1998, Nakano et al., 1998] on learning game heuristics.

In the previous work we need to induce rules that decide the most preferable move among many possible moves for each game situation. For this purpose, rules have to evaluate moves and sort them based on the evaluation. It is difficult to induce this kind of rules using relational learning framework. Instead of inducing such rules, we defined and induced a binary relation better-choice, which is a pair-wise relation among moves with a parameter of board configuration or game situation.

For a board configuration b and two possible moves m_1 and m_2, better-choice(b, m_1, m_2) is true if and only if m_1 is better than m_2 at the board configuration b, in order to reach a game solution. Instances of this relation are collected from an experience of game execution and used for induction. Induced rules (or logic programs) of better-choice are used to calculate the evaluation of moves for future games. In [Inuzuka at al., 1997a] and [Nakano et al., 1998] we applied this method to solve the eight puzzles and shogi (i.e., Japanese chess) mating problems.

For the purpose of this paper we apply induction of the better-choice relation to induce rules of sorting successor nodes of the alpha-beta method in order to have effective pruning.

We intuitively define the better-choice relation as follows. For a board configuration b and two possible moves m_1 and m_2, better-choice(b, m_1, m_2) is true if and only if m_1 is more plausible to have large evaluation than m_2 at the board configuration b, or if m_1 should be visited earlier (later) to have large pruning effect than m_2 when b is a MAX node (MIN node, respectively). Note that the better-choice relation is defined depending on a given static evaluation.

Obeying this definition, we can collect examples of the relation from search execution with game-trees as follows. Here, we assume that we use a heuristic function h for static evaluation and apply a depth d for the depth of lookahead. For each node b of a game-tree, we assign its successor nodes $b_1, \cdots, b_n$ evaluation $v(b_1), \cdots, v(b_n)$ by the mini-max method, where $b_1, \cdots, b_n$ are brought by possible moves $m_1, \cdots, m_n$ from b. Then, an instance,

$$\text{better-choice}(b, m_i, m_j),\ i, j \in \{1, \cdots, n\},\ i \neq j \quad (1)$$

is a positive example of the relation if $v(b_i) \geq v(b_j)$, and is negative otherwise. For each node with n possible moves, $n(n-1)/2$ positive examples and the same number of negative ones are generated.

All examples for nodes and of game-trees are merged to induce a logic program that defines the better-choice relation.

4 APPLYING better-choice TO THE ALPHA-BETA METHOD

Although the better-choice relation is clearly defined as a total order relation among possible moves with a board parameter, we can not expect that a logic program induced from collected examples by an ILP algorithm gives neither a total order relation nor even a partial order relation. In [Inuzuka at al., 1997a, Nakano et al., 1998] we proposed a method to calculate points for preference of moves based on an induced relation. In the method we defined a plausibility flow of preference of moves to be visited earlier, and points were calculated as a stationary status of the flow. We used the points as an evaluation of board configuration. We, however, do not need evaluation or points for the purpose of this paper. We need only order of moves. We used an easier method to calculate order among moves.

Let b be a board configuration and M be a set of all possible moves on b. Then, we define the following value $w_b(m)$, called a *winning point*, of a move $m \in M$ against other moves.

$$w_b(m) = \left| \left\{ \begin{matrix} m' \in \\ M - \{m\} \end{matrix} \middle| \begin{matrix} \text{better} - \text{choice}(b, m, m') \\ = \text{true} \end{matrix} \right\} \right|,$$

where $|A|$ denotes the number of elements in A. We take the order of the winning point for the order of moves, or order of the successor nodes caused by the moves. This counts the moves that are not better than a move. For the case that two moves take the same winning points, we modify this and define a total order $\succ_b$ as follows:

$m_1 \succ_b m_2$ when $w_b(m_1) > w_b(m_2)$, or when $w_b(m_1) = w_b(m_2)$ and $w_b(m_1) \sqsupset w_b(m_2)$,

No license: PDF produced by PStill (c) F. Siegert - http://www.this.net/~frank/pstill.html

where $\sqsupset$ is a given total order. We chose the lexicographic order for $\sqsupset$. The order $\succ_b$, of course, always give a total order among possible moves. When an induced program is a definition of a total order, the increasing order by $\succ_b$ is exactly the same as the total order. The procedure using winning points is not only the way to make a total order, but we used this for its simplicity.

We replace Step 2 of Table 2 by the following lines, in order to build induced rules in the alpha-beta method.

---

2 Let $b_1, \cdots, b_n$ be the successors of b, the successors which are sorted in the increasing (decreasing) order by $\succ_b$, when b is a MAX node (MIN node, respectively). Set $k := 1$, and if b is a MAX node, go to step 3 ; else go to step $3'$.

We can guarantee that this replacement does not change results of search, except cases that two boards have the same evaluation. For the exceptional cases, the proposed method may changes results, because the sorting method gives an order even among two nodes with the same evaluation.

We can restrict the replacement of lines in Table 2 for only initial several levels of calls. The alpha-beta method is called recursively up to a given depth for every evaluation of nodes. The calculation of order takes certain amount of time cost and so it is an idea to apply the procedure to only the first a few levels of recursion. Note that in this case we apply the sorting procedure every call of the alpha-beta procedure but only the first a few levels of recursion.

5 EXPERIMENTS AND RESULTS

We conducted experiments to play *othello* using the alpha-beta method with the proposed sorting method. An outline of the experiment procedure is described as follows.

1. Let two computer players play othello game, called a training game, once. Both player use the mini-max method with a given heuristic function for static evaluation to decide their moves. History of the play is recorded.
2. Generate positive and negative examples of better-choice relation for all nodes visited during the play, using Rule (1).
3. Induce a logic program of better-choice relation from collected examples and a given set of background knowledge.
4. Again, let two computer players play another game of othello, the players who use alpha-beta

Table 3: A Part of Background Knowledge.

predicates	explanation
zero($+N$)	N is zero.
lt($+N_1, +N_2$)	$N_1 < N_2$.
color-of-this-turn ($+B, -C$)	C is color of a piece to be placed in board B in this turn.
color-of-another-turn ($+B, -C$)	C is color of a piece to be placed in board B in this turn.
number-of-pieces ($+C, +B, -N$)	N is the number of pieces of color C on the board B.
remain-places ($+B, -N$)	N is the number of places not occupied by any piece, on the board B.
corner($+P$)	P is at a corner
on-edge($+P$)	P is on an edge
many-opponent's-pieces ($+B, +P, +C$)	There are many (more than a threshold) opponent's pieces (wrt. color C) around P on B.
inner-than($+P_1, +P_2$)	P_1 is an inner place than P_2 (P_1 is nearer to the center of the board than P_2.)

Table 4: Clauses in an Induced Logic Program.

Clause 1

better-choice(X_0, X_1, X_2) : −
color-of-this-turn(X_0, X_3),
color-of-another-turn(X_0, X_4),
on-edge(X_1), number-of-pieces(X_3, X_0, X_6),
number-of-pieces(X_4, X_0, X_7), lt(X_6, X_7),
inner-than(X_2, X_1).

Explanation: When there are more pieces of opponent's color than ours on the board, a place on an edge is preferred to an inner place.

Clause 2

better-choice(X_0, X_1, X_2) : −
color-of-this-turn(X_0, X_3), remain-places(X_0, X_5),
on-edge(X_1), not-on-edge(X_2),
number-of-pieces(X_3, X_0, X_6), lt(X_6, X_5).

Explanation: When the number of places not occupied by any pieces is larger that the number of pieces of our color on the board, a place not on an edge is preferred to a place on an edge.

No license: PDF produced by PStill (c) F. Siegert - http://www.this.net/~frank/pstill.html

method with the sorting method using the better-choice relation. Observe the number of nodes visited during the play. For comparison, the number of nodes visited by normal alpha-beta method is also observed.

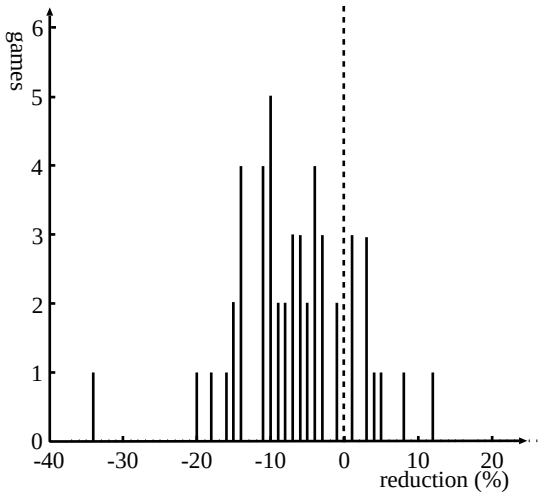
For our experiment, we set the depth of lookahead five. We prepared two heuristic functions, that is, two different experiments were conducted. For adaptation of induced logic programs (Step 4 above), we applied the sorting procedure only to the first level of recursion of the alpha-beta procedure. We let two computer players play 50 whole games of othello from randomly generated initial board configurations where three black pieces and two white pieces are placed at random.

A part of background knowledge that we used for induction is shown in Table 3. Predicates are all simple predicates, truth of which is calculated easily. In the table, symbols N, B, C and P denote predicate's arguments with types integers, board configurations, colors (white or black), and moves (or places to be put a piece), respectively.

We used an ILP algorithm FOIL-I[Inuzuka et al., 1996, Inuzuka et al., 1997b], which is a derivative of one of the successful earlier top-down ILP system FOIL[Quinlan, 1990, Quinlan, 1993]. FOIL-I can use intentional definition of relations as background knowledge, or any logic programs with type and mode information.

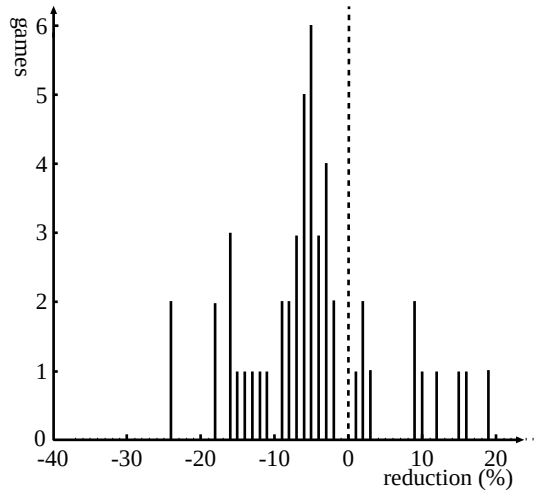
Table 4 shows two example clauses in an induced logic program of better-choice relation. The program compares two moves (or places to be put) with a board parameter.

Figure 1 summarizes results of experiments. Two graphs correspond to two given heuristic functions. A graph plots the number of games that had each reduction percentage, the reduction percentage which is a rate of the number of nodes visited during the game with proposed method against the normal alpha-beta method. The total number of games is 50 for each graph. -10% on the X-axis means 10% reduction for visited nodes, and $+10\%$ means 10% increase. We observe that approximately 80% of games had reduction of visited nodes by the method. The average reduction percentages were $-4.0 \pm 5.5\%$ and $-4.0 \pm 9.4\%$ for the two heuristic functions.



Average $-4.0 \pm 5.5\%$

(a) Experiments with Heuristic Function 1.



Average $-4.0 \pm 9.4\%$

(b) Experiments with Heuristic Function 2.

Figure 1: Results of the Experiments.
Graphs show histograms of reduction percentages of visited nodes by the proposed method.

6 CONCLUSIONS

We proposed and inspected a method to use an ILP technique to learn rules to sort nodes expanded in the alpha-beta method in order to have effective pruning. In order to reduce this order induction method into a classification problem, we used the better-choice rela-

No license: PDF produced by PStill (c) F. Siegert - http://www.this.net/~frank/pstill.html

tion, which was introduced for game heuristics induction in the previous paper. The experiments showed that the proposed method works to have expected rules.

Although by the advantages of ILP framework, we can use any predicates described in logic programs, we should use inexpensive predicates. We also take care for induction of recursive logic programs. If we allow an ILP system to induce recursive logic program, it may induce as a definition of better-choice relation a look ahead program. This program is not our aim for sorting procedure. We should restrict predicates and logic programs into shallow predicates, i.e., predicates only depending on board information without lookahead.

In this paper we inspected only a basic functionality of the idea, and the following two issues should be taken into consideration for enhancement of the method:

1. In the experiment we applied induced rules only to the first depth of lookahead. We should inspect the effect when we apply induced rules further depth, and also the trade off between effect of depth and computation overhead.
2. We dealt with all of examples collected from training games as a uniform sample set, and induced rules from the examples. However, they include examples for different stages of games, i.e., the beginning, middle, and the final stages of games. We may need different criteria, or rules to sort game configurations, depending on the stages. We should consider this factor to induce rules.

To see the effect of the proposing method we also need other experiments to compare other method for sorting. These remain for future work. Application to other games and study on effect of background knowledge also remains for future work.

References

[Bain & Srinivasan, 1995] Bain, M. & Srinivasan, A. (1995) Inductive Logic Programming With Large-Scale Unstructured Data. In *Machine Intelligence 14* (pp.235–274) Clarendon Press Oxford (OUP).

[De Raedt, 1995] De Raedt, L. (ed.) (1995) Advances in inductive logic programming. IOS Press/Ohmusha.

[Fürnkranz, 1996] Fürnkranz, J. (1996) Machine Learning in Computer Chess: The Next Generation. *International Computer Chess Association Journal 19*, 147–161.

[Inuzuka et al., 1996] Inuzuka, N., Kamo, M., Ishii, N., Seki, H. & Itoh, H. (1996) Top-down induction of logic programs from incomplete samples. *Proceedings of the Sixth International Inductive Logic Programming Workshop (ILP'96)* (pp.265–282) There are other choices to make a Lecture Notes in Artificial Intelligence 1314, Springer-Verlag.

[Inuzuka at al., 1997a] Inuzuka, N., Seki, H. & Itoh, H. (1997) An intelligent search method using Inductive Logic Programming. *Proceedings of IJCAI'97 Workshop Frontiers of Inductive Logic Programming* (pp.115-120) (url=`http://www.cs.kuleuven.ac.be/~lucdr/filp.html`).

[Inuzuka et al., 1997b] Inuzuka, N., Seki, H. & Itoh, H. (1997) Efficient induction of executable logic programs from examples. *Proceedings of the Third Asian Computing Science Conference* (pp.212-224) Lecture Notes in Computer Science 1345, Springer-Verlag.

[Inuzuka et al., 1998] Inuzuka, N., Nakano, T. & Itoh, H. (1998) Acquiring heuristic values in search trees. *Proceedings of 11th International Conference on Applications of Prolog (INAP98)*, pp.157-162.

[Nakano et al., 1998] Nakano, T., Inuzuka, N., Seki, H. & Itoh, H. (1998) Inducing shogi heuristics using inductive logic programming. *Proceeding of the Eighth Inductive Logic Programming Conference* (pp.155-164) Lecture Notes in Computer Science 1446, Springer-Verlag.

[Morales, 1992] Morales, E. (1992) Learning Chess Patterns. In S. Muggleton (ed.) *Inductive Logic Programming*, Academic Press.

[Morales, 1997] Morales, E. (1997) PAL: A pattern-based first-order inductive system. *Machine Learning 26* 227–252.

[Muggleton, 1992] Muggleton, S. (ed.) (1992) Inductive logic programming. Academic Press.

[Quinlan, 1990] Quinlan, J. R. (1990) Learning logical definitions from relations. *Machine Learning 5* 239–266.

[Quinlan, 1993] Quinlan, J. R. & Cameron-Jones, R. M. (1993) FOIL: A midterm report. *Proceedings of the Sixth European Conference on Machine Learning* (pp.3–20) Lecture Notes in Artificial Intelligence 667, Springer-Verlag.

[Nilsson, 1998] Nilsson, N. J. (1998) Artificial Intelligence: a new synthesis. Morgan Kaufmann Publishers.

No license: PDF produced by PStill (c) F. Siegert - http://www.this.net/~frank/pstill.html