A Search Optimization Method for Rule Learning in Board Games

Hui Wang[1], Yanni Tang[1], Jiamou Liu[2], and Wu Chen[1,3(✉)]

[1] College of Computer and Information Science, Southwest University, Chongqing 400715, China
[2] The University of Auckland, Auckland, New Zealand
jiamou.liu@auckland.ac.nz
[3] Institute of Logic and Intelligence, Southwest University, Chongqing 400715, China
chenwu@swu.edu.cn

Abstract. A general game playing (GGP) system aims to play previously unknown board games with changeable rules without human intervention. Taking changeable game rules into consideration, a game description language presents formal descriptions of a game. Based on this description, legal moves can be automatically generated so that each player in GGP system only needs to solve the problems of *searching* and *learning* for playing well. Traditional search methods demand the player to compute all legal moves, which can be very time consuming. In GGP, the coordinate of cells in the game board is very important in board game rules. Thus we address the relationship among cell coordinates. Borrowing an idea from rule learning to prune the board game search tree, we propose a new search optimization method to reduce running time when searching a large search space. We further prove that this method can effectively improve the searching efficiency through a comparative experiment with Gomoku game in GGP system.

Keywords: General game playing · Game description language
Rule learning · Searching optimization

1 Introduction

One of the most prominent successes of artificial intelligence is Deep Blue's defeat of world chess champion Garry Kasparov. However, a fundamental limitation exists with Deep Blue as the game rules are built into the system that prevents the system from achieving a desirable level of machine intelligence. As it's not practical to construct specific strategic engines for all games, GGP emerged as a promising direction which aims for an intelligent system that automatically learns games rules and derives game strategies without human intervention [1]. A GGP sets itself apart from the traditional game program in that it implements

X. Geng and B.-H. Kang (Eds.): PRICAI 2018, LNAI 11013, pp. 174–181, 2018.
https://doi.org/10.1007/978-3-319-97310-4_20

an interpreter program [2], which takes as input formal descriptions of a game – as specified in a game description language (GDL) [3] – and generate legal moves under a specific state with an interpreter program.

Since a GGP platform allows us to play all kinds of board games, how to play these games well in GGP becomes an important issue. Numerous studies have been done to assist players on a specific game, such as AlphaGo [4]. But in general games, there are still many problems to solve. Since in GGP, games are described as rules, and the result of rule learning is also a set of rules. Rules learning from game records can be regarded as decision strategies and searching strategies. Therefore, this paper combines the knowledge of GGP with rule learning to try to find out a searching optimization method. To make it simple, we choose Gomoku game as our example. We introduce an experiment of playing Gomoku game with two players, in which players have the same decision strategies but one applied the rule obtained from training, and the other did not use it. The results show that the player who uses the search rules can more quickly find the best move in large scale games with an even win rate.

Our contributions can be summarized as follows:

1. This paper presents a searching optimization method that applies rule learning to GGP field. The idea is evidenced by game Gomoku. An algorithm of generating searching rules is illustrated based on *Inductive Logic Programming*.
2. Knowledge obtained from rule learning is formalized as game playing rules. It's important to study the relation between game playing rules and game rules, the method of this paper provides a basement to study this relationship further.

The paper is organized as follows. Section 2 illustrates related work. Section 3 recalls the basic concepts of GGP and rule learning. Section 4 introduces the rule learning process of searching rules of Gomoku game based on *Inductive Logic Programming* in GGP. Section 5 presents a comparative experiment with Gomoku in GGP. Section 6 concludes the paper and discusses on future work.

2 Related Work

Learning about general concepts or rules has been a crucial research problem in artificial intelligence. *Inductive Logic Programming* is an important paradigm to cope with the issue of rule learning. Arindam Mitra et al. addressed a question-answering challenge by combining statistical methods with inductive rule learning and reasoning [5]. Furthermore, David Garciá et al. proposed an interpretability improvement for fuzzy rule bases obtained by the iterative rule learning approach [6].

More recently, Ondřej Kuželka et al. introduced a setting for learning possibilistic logic theories from defaults of the form "if alpha then typically beta" [7]. They aim at studying the problem of reasoning with default rules from a machine learning perspective.

Specifically on GGP, there exist also numerous works that made great progress. Günther et al. proposed a search algorithm that exploits this information in single-player games [8]. Moreover, Dave and Zhang proposed a lifted backward search in GGP system [9].

Different from the methods we mentioned above, we refer to the way proposed by Krajnanský et al. [10] based on our requirements to generate simple rules to minimize search space.

3 Preliminaries

3.1 General Game Playing and Game Description Language

General game players are systems who are able to accept descriptions of arbitrary games at runtime and derive corresponding strategies to play those games effectively without human intervention. In other words, they do not know the rules until the games start [11,12]. In GGP structure, the game manager is at the center of the ecosystem. There are databases for game descriptions, match histories, and temporary states histories during game playing. The game manager interacts with game players through TCP/IP protocol.

Every finite game can be modeled as a state transition system. A game description is a finite collection of rules and a GDL is a specific language to describe this collection [13]. A basic game description includes the specific attributes and relations. The reader is referred to [3] for numerous examples and tutorials.

3.2 Rule Learning

Rule learning is a process of generating a set of rules to evaluate unknown instances from training data set [14]. A formalized rule form is $\oplus \leftarrow f_1 \wedge f_2 \wedge ... \wedge f_i ... \wedge f_L$, where the right hand side $f_1 \wedge f_2 \wedge ... \wedge f_i ... \wedge f_L$ is called *rule body* and represents this rule's preconditions, while the left part $\oplus$ is called *rule head* which represents the result of this rule. The rule body is a conjunction that consists of a sequence of literals f_i. The symbol $\wedge$ means conjunction. Every literal f_i is a boolean expression to examine the attributions/properties of examples. Suppose we have two rules: $\oplus \leftarrow f_1 \wedge f_2$; and $\oplus \leftarrow f_3 \wedge f_4$. The first rule means that an example is positive if it is covered by properties $f_1 \wedge f_2$. The second rule means that an example is positive if it is covered by properties $f_3 \wedge f_4$. This could also be combined to write as a single disjunctive rule $\oplus \leftarrow (f_1 \wedge f_2) \vee (f_3 \wedge f_4)$.

4 Learning Search Rules

Since a GGP system can record game history, one can easily obtain a training set from the game records. Many studies exist, e.g. [15], that provide efficient instance selection algorithms to reconstruct training sets. In order to distinguish the samples in the training set, these samples should be divided into two parts,

we let E^+ and E^- to represent these two parts respectively. In order to define a framework for our GGP solution, a general game state transition model for games can be simply regarded as sets of state-action pairs (s, a) [16]. Formally,

$$E^+ = \{(s,a)|s \in S^*, a \in A^+\} \quad \text{and} \quad E^- = \{(s,a)|s \in S^*, a \in A^-\} \quad (1)$$

where S^* represents the set of history states, A^+ (A^-) represents the set of actions which are (not) most possible to be applied in s, A^+ (A^-) can make positive (negative) effect on winning, $A^+ \subseteq A$ and $A^- \subseteq A$. $A^+ \cap A^- = \varnothing$ and $A^+ \cup A^- = A$. The state s' is the outcome after applying a to s, where $s \subseteq S^*$ and $s' \subseteq S^*$. E^+ is called *good* examples set, E^- is called *bad* examples set.

Our purpose is to get a formal representation R of rules by rule learning. The representation R should allow us to reduce the search space. The rule is generated from covering all samples of the same domain D. In order to ensure that R can cover all good examples and try its best to exclude bad examples, in any state s, we should include all possible action a and avoid any impossible actions. In order to find R, here are some problems we need to solve: **(a)** What kind of representation can describe R; **(b)** What "covering" a state-action pair (s, a) means; **(c)** How R can be learned.

In fact, Michal et al. presented their answer in [10]. However, in GGP, things are different. In our work, we refer to GDL since GDL provides us the game rules collection. As for board games, such as Tic-Tac-Toe, Connect Four, Hex, the coordinates of cells are the most important part in their GDL description: States of board games are described in different cells' states (coordinates, marked or not, by whom. e.g. (cell 1 1 x)); actions are also described using coordinates (e.g. mark(1 1)). So the coordinates of different cells are the key to represent the basic rule form. Thus, we propose a model to process this in Fig. 1:

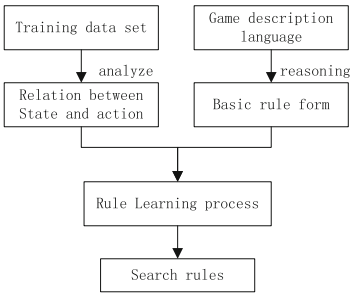

Fig. 1. Model of generating search rules with game description

In a Gomoku game, our example, we can easily find that the game goal description is to construct a line with 5 pieces. The player firstly needs to construct a line with 2 pieces, then three then four and lastly five pieces to reach the termination of the game. In the other hand, the player must stop the opponent from constructing a line with pieces. Thus, either we'd like to attack or defense, we both firstly need to search within the places next to our or opponents pieces.

Through the analysis and reasoning above, we propose an algorithm for learning searching rules in board games as follows:

Algorithm 1. Algorithm of generating searching rules based on board coordinate

Input:
The set of *good* (*bad*) examples, E^+ (E^-); Search space ω; Whole board space Ω;
Output:
The set of searching rules R;
1: $r_1 : \omega = \Omega \leftarrow s = \varnothing$; $\triangleright s = \varnothing$ means game starting state.
2: **for** each (s,a) in E^+ **do**
3: $s = \{(m_1, n_1), (m_2, n_2), \ldots (m_k, n_k), \ldots, (m_j, n_j), 1 \leq k \leq j\}$;
4: $a = move(m, n)$;
5: $r_2 : \omega = \Omega \cap \bigcup_{k=1}^{j} \{move(m_k \pm \sigma, n_k) \cup move(m_k, n_k \pm \sigma) \cup move(m_k \pm \sigma, n_k \pm \sigma)\} \leftarrow \sigma = min\{N | \exists move(m_k, n_k) s.t. |m_k - m| \leq N) \wedge (|n_k - n| \leq N)\}$;
6: **end for**
7: **for** each (s,a) in E^- **do**
8: $s = \{(m_1, n_1), (m_2, n_2), \ldots (m_k, n_k), \ldots, (m_j, n_j), 1 \leq k \leq j\}$;
9: $a = move(m, n)$;
10: $M = \{a\}$;
11: $r_3 : \{\omega = \omega \leftarrow M \subsetneq s\}$;
12: $r_4 : \{(\omega = \omega - M \leftarrow M \subseteq s) \wedge (\omega = \omega + M \leftarrow$ after searching under this state $s)\}$;
13: **end for**
14: $R = \{r_1, r_2, r_3, r_4\}$;
15: **return** R;

In Algorithm 1, we can see, when game board is empty, we use r_1, i.e., searching the whole board. Then in *good* examples using *Inductive Logic Programming* to find the relationship between the coordinate of action and the existing coordinates of game state. This relationship is described as r_2. Based on r_2, check if r_2 also covers *bad* examples, if not, i.e., $M \subsetneq s$, then search space keep the same ($\omega = \omega$), written as r_3. Otherwise, exclude this *bad* example ($\omega = \omega - M$). The computational complexity is $\bigcirc(n^2)$, where n is the number of training examples.

5 Experiment Results

According to Algorithm 1, we learn searching rules based on the training set in Gomoku. We get the result that $\sigma = 1$, the search space is $\omega = \Omega \cap \bigcup_{k=1}^{j} \{move(m_k \pm 1, n_k) \cup move(m_k, n_k \pm 1) \cup move(m_k \pm 1, n_k \pm 1)\}$, which means we should search the places just next to the pieces existing in the board. We call these places (legal moves) as *activists* positions. The result holds the same information as what we analyzed and reasoned above.

In our implementation, for instance, we know X player has occupied position $(4, 4)$, that Player O has occupied position $(3, 3)$, and now it's X player's turn.

According to the rules learned before, the next move is most probably to mark one of the positions in the set of {(2,2), (2,3), (2,4), (3,2) (3,4), (3,5), (4,2), (4,3), (4,5), (5,3), (5,4), (5,5)}. Note that in the experiment we should return a legal move instead if there is no expected move in the activists set.

We create a player based on this searching method with the same playing strategies as opponent's (always get a draw), which is called ActivistsFirstPlayer. Obviously, the player's name means searching the activists list first to get expected results. The opponent use traditional ways like searching in all legal moves list. There are three games with a size of 8×8, 16×16 and 32×32 board respectively. Obviously, from Fig. 2, two players have different search spaces, the search space of ActivistsFirstPlayer increases from 0, and then goes down, while the opponent's decreases linearly by the board size. More importantly, in every step, the search space of ActivistsFirstPlayer stays below the opponent's.

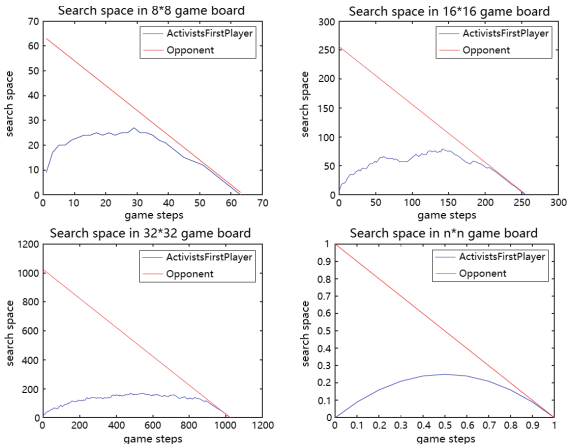

Fig. 2. Search space of different size game boards

Since ActivistsFirstPlayer only needs to search through a smaller space to find the next move, the time spending of ActivistsFirstPlayer on calculate is evidently less than the opponent's. However, ActivistsFirstPlayer will cost some time to get such a set of moves. From the experiment results of time spending in Fig. 3, it is worthy to do this job, especially to search a great scale area. In addition, we can decrease this cost to a smaller level by several ways comparing with traversing the whole area. On the 8×8 board, the cost of time of both players has little difference. In fact, ActivistsFirstPlayer pays little time to search but spends extra time on constructing the activists list, because the list is always dynamically changing. In the 16×16 board, opponent spends several times time than ActivistsFirstPlayer for each move. And in the 32×32 board, the time gap between two payers' time spending is getting bigger and bigger. The results prove that this method is applicable for the games and which can also reduce search space through rule learning, rather than a simple statistic machine learning.

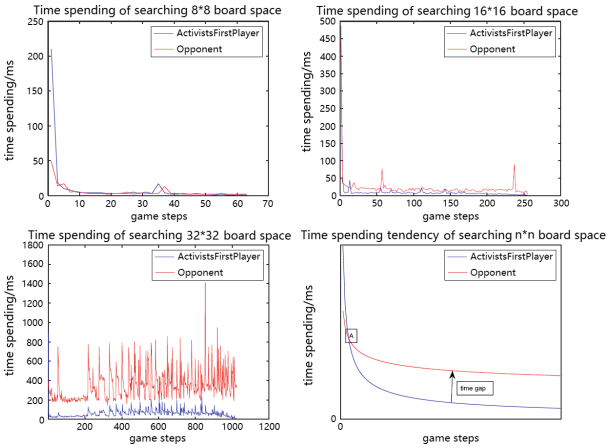

Fig. 3. Time spending on searching different size game boards

Based on these three real experiments outcomes, we can simulate the time spending while searching $n \times n$ board space. In Fig. 3, at the beginning, tradition search method spends less time than the ActivistsFirstPlayer. The reason is that ActivistsFirstPlayer needs to spend extra time on constructing the activists list. But this extra time spending will be compensated. In the dot A marked in the diagraph, the time spending of both players are the same. Later on, the time spending of ActivistsFirstPlayer is getting less than the tradition method. It is obviously that when the board size becomes bigger and bigger, the dot A will get closer to y-axis. The time gap will become bigger. Therefore, the experiments results have demonstrated this search optimization method is more efficient while searching a large scale field at a high accurate rate level.

6 Conclusion and Future Work

This paper first introduce rule learning into GGP to learn trustworthy rules to determine what should or shouldn't be searched for board games playing. We found the *goal* function written in GDL for the game is more possible to tell players which state can be closer to game termination. We analyzed properties from *goal* function to define the rule form as the input of rule learning process first, and then applied *Inductive Logic Programming* method to learn out searching rules. Last, we applied these rules to the game searching process, and introduced an experiment to prove the accuracy and effectiveness of this new method.

In the future, our purpose is to establish an independent module to generate search rules, and apply these rules to most of board games automatically and effectively. So that this module can be integrated to enrich GGP system with high efficiency. Our method in this paper is practical, but there is no doubt that further work should be done to improve it.

Acknowledgement. This work was supported by the Key Project of Chongqing Humanities and Social Science Key Research Base: Research on Coalition Welfare Distribution Mechanism and Social Cohesion Based on Cooperative Game Theory and the Ratification Number is 18SKB047.

References

1. Genesereth, M., Love, N., Pell, B.: General game playing: overview of the AAAI competition. AI Mag. **26**(2), 62–72 (2005)
2. Świechowski, M., Mańdziuk, J.: Fast interpreter for logical reasoning in general game playing. J. Log. Comput. **26**(5), 1697–1727 (2014)
3. Love, N., Hinrichs, T., Haley, D., et al.: General game playing: game description language specification (2008)
4. Silver, D., Huang, A., Maddison, C.J., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
5. Mitra, A., Baral, C.: Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (2016)
6. Garciá, D., Gaḿez, J.C., Gonzaĺez, A., Peŕeza, R.: An interpretability improvement for fuzzy rule bases obtained by the iterative rule learning approach. Int. J. Approximate Reasoning **67**, 37–58 (2015)
7. Günther, M., Schiffel, S., Thielscher, M.: Factoring general games. In: Proceedings of the International Joint Conference on Artificial Intelligence-09 workshop on general game playing, pp. 27–34 (2009)
8. Kuželka, O., Davis, J., Schockaert, S.: Learning possibilistic logic theories from default rules. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (2016)
9. de Jonge, D., Zhang, D.: Lifted backward search for general game playing. In: Kang, B.H., Bai, Q. (eds.) AI 2016. LNCS (LNAI), vol. 9992, pp. 3–16. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50127-7_1
10. Krajnanský, M., Hoffmann, J., et al.: Learning pruning rules for heuristic search planning. In: European Conference on Artificial Intelligence, pp. 483–488 (2014)
11. Genesereth, M., Thielscher, M.: General Game Playing. Morgan & Claypool Publishers, Williston (2014)
12. Kaiser, D.M.: The design and implementation of a successful general game playing agent. In: Wilson, D., Sutcliffe, G. (eds.) International Florida Artificial Intelligence Research Society Conference 2007, pp. 110–115. AAAI Press, California (2007)
13. Thielscher, M.: A general game description language for incomplete information games. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 994–999. AAAI Press, Atlanta (2010)
14. Fürnkranz, J., Gamberger, D., Lavrac, N.: Foundations of Rule Learning. Cognitive Technologies. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-540-75197-7
15. Liu, C., Wang, W., Wang, M., Lv, F., Konan, M.: An efficient instance selection algorithm to reconstruct training set for support vector machine. In: Knowledge-Based Systems, pp. 58–73 (2017)
16. Zhang, D., Thielscher, M.: Representing and reasoning about game strategies. J. Philos. Log. **44**(2), 203–236 (2015)