Contextual Patterns and Pattern Collocations in the Game of GO

Zhiqing Liu, Wenfeng Li

Abstract— **Knowledge modeling, acquisition, and proper use are crucial for successful computer playing of the game of GO. This paper presents a systematic approach for knowledge representations of GO-playing in which the meaning of a move is defined as a contextual pattern with respect to local contexts of surroundings of the move. This representation allows large amounts of contextual patterns, along with their usage statistics, to be acquired efficiently from game records. Contextual patterns that appear jointly in local contexts may form a pattern collocation, with its significance measured by hypothesis testing. Our experiments show that GO-playing knowledge acquired in the form of contextual patterns and pattern collocations is effective in move generation at various stages of GO-playing.**

Index Terms— **Contextual pattern, Pattern collocation, Pattern recognition, Knowledge representation, Move generation**

I. INTRODUCTION

Computer game playing has been a part of the core of artificial intelligence (AI) since it became a field of study; and the game of GO is one of its grand challenges [12]. Computer GO has been studied for about four decades and many important progresses have been made. Some of the most influential works include [1], [7], [6]. However, all related works are still elusive with respect to the ultimate goal of defeating top-level human players. The reasons are well recognized: Current techniques for successful computer play of other games such as Chess do not apply directly to GO. A key of them is a static and accurate evaluation of game boards integrated with efficient search algorithms of game trees. However, this search-based technique is almost useless on the full board in GO because of the following two complications:

1) The game has a significantly larger decision complexity and state-space complexity than other games due to its much larger branching factors [10], and
2) Static and accurate evaluation of the game board is not tractable.

Consequently, researchers typically construct GO-playing programs using knowledge-based approaches with heuristics and pattern matching. Knowledge used in GO-playing programs can be of various types, and the most widely used one is patterns, which are well established sequences of moves that can be accepted by both players. Patterns are commonly used in GO games between human players, and it is believed in the GO community that effective recognition of patterns and their competent use are a key component of effective GO playing [8]. To this end, dictionaries of patterns, designed primarily for human players, have been compiled for various stages of the game, and pattern databases are used in almost all competent GO-playing programs.

The purpose of the paper is to acquire certain knowledge of patterns. The rest of the paper is organized as follows: Section II presents a brief introduction to the game of GO. Section III defines formally the concept of contextual pattern in GO, and discusses properties of contextual patterns acquired from a large collection of professional game records. Section IV defines formally the concept of pattern collocation, presents an algorithm for efficient acquisition of pattern collocations, and discusses properties of pattern collocations acquired from the same collection of game records. Section V demonstrates quality and sample uses of contextual patterns and pattern collocations with two applications in move generation. Section VI compares the use of Contextual Patterns and Mogo Patterns in the simulation of Monte-Carlo Tree Search. This paper is concluded in Section VII with a summary of results and discussions of future works.

II. THE GAME OF GO

Some essential knowledge of the game of GO is mandatory for subsequent discussions; and this section presents a brief introduction to GO, with some discussions on its key differences from other board games such as Chess, which make it unique and difficult. See [8] for more comprehensive treatments of the game. The game of GO is a two-person, zero-sum, perfect-information board game with deceivingly simple rules. In a game of GO, its two players alternately place stones of their colors, which are black and white, on an empty crossing on a 19-by-19 board, subject to the rule of "no suicide" and the rule of "no repetition" [8].

GO is a territorial game; and each players aims to secure more territories with her stones. All living chains of stones eventually secure certain territories, which may or may not include the stones in the chains themselves, depending on whether the Chinese rules or the Japanese rules are used. Differences between the two main sets of rules are generally minimal; both assuming no-trivial knowledge of the game when territories are counted in scoring, because potentially dead chains of stones are just recognized as such without being actually captured and removed from the board.

Additionally, chessmen in Chess have their inherent capability and relative values. Typically the queen is worth nine times more than a pawn while the king is invaluable [2]. The existence of inherent relative values of chessmen makes static

{Zhiqing Liu}, P.O.Box 146, Beijing University of Posts and Telecommunications, Beijing, China, 100876, E-mail: zhiqing.liu@gmail.com

{Wenfeng Li}, Beijing University of Posts and Telecommunications, E-mail:lwfeng115@gmail.com

978-1-4244-8738-7/11/$26.00

```
for each move m in a game record loop
  Play m on board;
  if CP_m not in database then
    Insert CP_m into database;
    CP_m.frequency := 0;
  end if;
  CP_m.frequency ++;
end loop;
```

Fig. 1. Algorithm is pseudo code for acquisition of contextual patterns

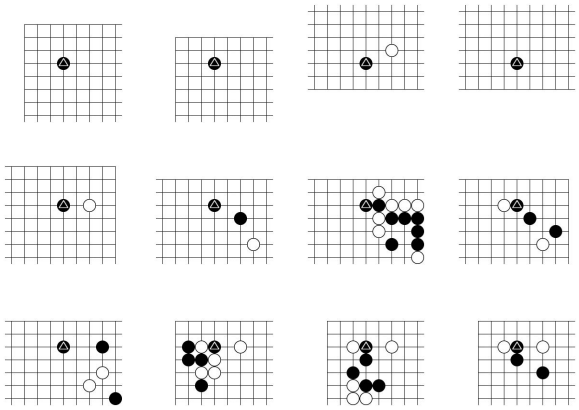

Fig. 2. Twelve 9-by-9 contextual patterns acquired from a collection of 16067 game records of the professional

evaluation of Chess board relatively easy. However, all GO stones of the same color on the board appear to be the same, but they have completely different values, depending on contexts of surroundings of the stones. In summary, all these differences, along with its enormous decision complexity and state-space complexity [10], make GO a unique and challenging game of our time.

III. Contextual Patterns

As stated above, the value of a move in the game of GO is not determined by itself, but largely by its contexts of surroundings. As such, a move in GO shall be studied along with its contexts of surroundings, and its value determined accordingly.

A. Definition

Given a move m played on the board in a GO game, a contextual pattern of m, denoted as CP_m, is defined as a collection of information of m, which consists of the stone of m, its surrounding stones, and other surrounding board configurations. We shall refer to the stone of m as the anchor stone of CP_m, and the surrounding stones as the contextual stones. In this definition, the contexts of surroundings are spatial instead of temporal, as found in natural language processing. We believe that the use of spatial definition of contexts of surroundings is appropriate because the game of GO is played on a game board that is of two dimensions, while natural languages are generated and processed in a strictly linear order.

B. Acquisition

Given the above definition, contextual patterns can be automatically acquired from a game record along with their frequency of occurrence using the algorithm shown in Figure 1, in which each move in the game record denotes the anchor stone of a contextual pattern, which is extracted along the play of the game and recorded in a database. Obviously, the complexity of the pattern acquisition algorithm is $O(N)$ and linear, where N denotes the number of moves in the game record, making the algorithm efficient to be able to acquire contextual patterns from a large collection of game records.

We have run the acquisition algorithm on a collection of 16067 game records of the professional, acquired 2817263 unique contextual patterns defined on 9-by-9 square regions, and sorted the contextual patterns based on their frequency of occurrence. Figure 2 shows twelve of the 9-by-9 contextual patterns acquired. Among them, the first six are the most frequent ones; while the frequency orderings of the rest six contextual patterns are ranked from the 1001st to the 1006th. Similarly, we have also acquired 2982 contextual patterns defined on 3-by-3 square regions from the same game record collection, and sorted them based on their frequency of occurrence.

C. Properties

Contextual patterns have a number of important properties and some of them are worth to be noted. First of all, the contextual patterns acquired from professional game records are in general of a high quality.

Secondly, frequency of occurrence statistics of contextual patterns from a large collection of game records ensures that the contexual patterns are consistent. Given a contextual pattern, its frequency of occurrence indicates, to a certain extent, its relative usefulness.

Thirdly, contextual patterns are typically of an enormous amount, especially when defined on large contexts of surroundings. For the example of the collection of 16067 professional game records, each game record contributes, on average, approximately 175 new 9-by-9 contextual patterns. Since each game record has approximately 212 moves on average, there are less than 40 moves in each game record whose 9-by-9 contextual patterns are known in the pattern database. In comparison, the amount of 3-by-3 contextual patterns is much smaller due to their limited possibilities, and each game record contributes less than one new pattern on average.

The property of large quality is just one aspect of a more general problem of "data sparseness". Figure 3 shows the frequency distribution of the 9-by-9 contextual patterns, which exhibits a Zipf's law like regularity [13]. That is, if the contextual patterns are sorted decreasingly by their frequency of occurrence, the frequency of a contextual pattern, denoted as f, is a power law function of its rank, denoted as r, in the form of $f \sim 1/r^e$ with the exponent e close to unity. Regardless of the size of a game record collection, most of the contextual patterns occurring in the game records

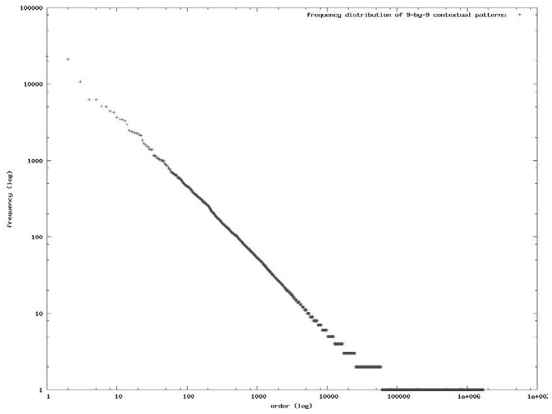

Fig. 3. Zipf's law like distribution of frequency of the 9-by-9 contextual patterns acquired from the collection of professional game records

```
for each move m in a game record loop
  Play m on board;
  Q := CP_m;
  for each preceding move l of m loop
    if l is a contextual move of Q then
      P := CP_l; break;
    end if;
  end loop;
  if no such P is found then
    P := empty contextual pattern;
  end if;
  if (P,Q) not in database then
    Insert (P,Q) into database;
    (P,Q).frequency := 0;
  end if;
  (P,Q).frequency ++;
end loop;
```

Fig. 4. Algorithm in pseudo code for acquisition of pairs of contextual patterns

have very low frequency and a small set of high-frequency patterns constitutes a large majority of occurrence in the game collection.

IV. Pattern Collocations

While contextual patterns provide a first order knowledge of GO-playing with respect to moves in the game, they are not enough. Observations and experiences indicate that contextual patterns are rarely used individually; instead a contextual pattern often appears jointly with certain others in a fixed order. As such, many important questions exist with respect to how contextual patterns are related with each other. This section aims to answer two critical ones of them, namely:

1) Which two contextual patterns appear jointly in local contexts, and
2) Whether such a joint appearance is of any statistical significance that is beyond just a pure coincidence.

A. Definition

Collocation is a term of linguistics, in which it means "two or more consecutive words, that has characteristics of a syntactic and semantic unit, and whose exact and unambiguous meaning or connotation cannot be derived directly from the meaning or connotation of its components"[9]. Following this, we shall use the term "pattern collocation", or just "collocation" in short, to refer to two consecutive contextual patterns that have characteristics of a logical unit in the game of GO.

More formally, we define a pattern collocation as a pair of contextual patterns that appear jointly in local contexts, and that their joint appearance is significant and not just a pure coincidence with a high degree of statistical confidence. Specifically, given two contextual patterns P and Q, we define that they appear jointly to form a pair, denoted as (P, Q), in local contexts if the anchor move of pattern P is temporally the closest contextual move of pattern Q. Additionally, we say that (P, Q) is a pattern collocation, denoted as $[P, Q]$ if the null hypothesis that P and Q are independent can be rejected with a high-degree of statistical confidence. For a collocation $[P, Q]$, we shall refer to P as the pre-pattern of the collocation and Q as the post-pattern.

B. Acquisition

According to the above definition, a two-step procedure is needed to acquire pattern collocations from game records. The first step to acquire pairs of contextual patterns and the second is to measure statistical confidence of each of the contextual pattern pairs to reject the null hypothesis. This subsection discusses both of the two steps.

Figure 4 shows our algorithm for acquisition of pairs of contextual patterns. Let N denote the number of moves in a game record, the worst-case complexity of this algorithm is $O(N^2)$.

Pairs of contextual patterns with high frequency of occurrence do not automatically qualify as pattern collocations because the frequency value of a contextual pattern pair can be accidentally high or low in a collection of game records. Given two contextual patterns P and Q with high-frequency, there is a chance that that the frequency of occurrence of (P, Q) is also high. What we really want to know is whether two contextual patterns appear jointly in local contexts more frequently than chance. It is can be done by hypothesis testing. The second step of our acquisition procedure is thus to formulate a null hypothesis H that there is no association between the contextual patterns beyond chance occurrences, compute the probability p that the event would occur if were true, and then reject the null hypothesis H if p is too low and retain the null hypothesis otherwise. The threshold for rejecting the null hypothesis used in our procedure is 0.005, meaning that the statistical confidence that the null hypothesis can be rejected is at least 99.5%. The hypothesis testing method that we use is the likelihood ratios test, which is asymptotically χ^2 distributed, is more appropriate for sparse data as in our case.

As such, the second step of the acquisition procedure is to use the likelihood ratios test to measure the statistical confidence of pairs of contextual patterns acquired in the first

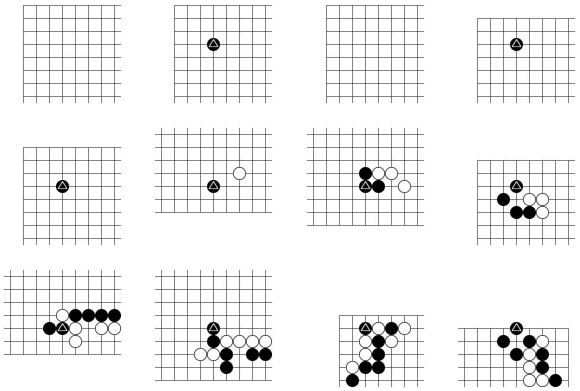

Fig. 5. Six pattern collocations of 9-by-9 contextual patterns from a collection of 16067 professional game records

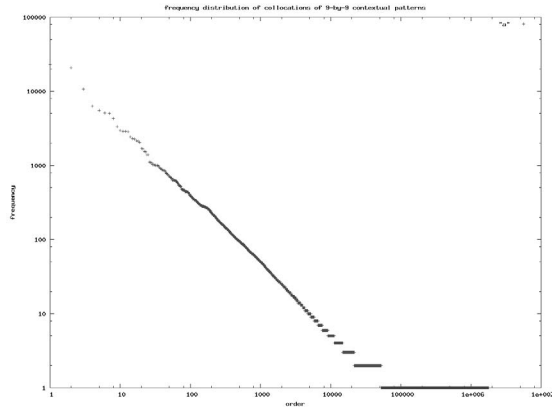

Fig. 6. Zipf's law like distribution of frequency of collocations 9-by-9 contextual patterns in the collection of professional game records

step. More specifically, given a pair (P, Q), the likelihood ratios test consists of the following two hypotheses:

1) $H_1 : Pr(Q|P) = Pr(Q|\neg P)$, i.e., pattern P and pattern Q are independent.
2) $H_2 : Pr(Q|P) \gg Pr(Q|\neg P)$, i.e., pattern P depends on pattern Q, indicating a pattern collocation.

Let $L(H_1)$ and $L(H_2)$ denote the likelihood of H_1 and H_2 respectively. Assuming a binomial distribution of the form

$$B_p(k, N) = \left(\begin{array}{c} N \\ k \end{array} \right) p^k (1-p)^{(N-k)}$$

$L(H_1)$ and $L(H_2)$ can be calculated respectively as follows:

$$B_p(C_{(P,Q)}, C_P) B_p(C_Q - C_{(P,Q)}, N - C_P)$$

and

$$B_{p_1}(C_{(P,Q)}, C_P) B_{p_2}(C_Q - C_{(P,Q)}, N - C_P)$$

where N is the total number of occurrence of contextual patterns, C_P, C_Q, and $C_{(P,Q)}$ are the numbers of occurrence of P, Q, and $P, Q)$ respectively, and $p = C_Q/N$, $p_1 = C_{(P,Q)}/C_P$, $p_2 = (C_Q - C_{(P,Q)})/(N - C_P)$. The likelihood ratios value of $-2\log(L(H_1)/L(H_2))$ is asymptotically χ^2 distributed.

We have run the collocation acquisition on the same collection of 16067 game records. For all of 2886347 pairs of 9-by-9 contextual patterns, only 2497 do not pass the probability threshold of the hypothesis. Figure 5 shows six 9-by-9 pattern collocations, of which the first three have the highest probability to reject the null probability, and the rest three have a sorted probability ranked in the 1001st to 1003rd respectively.

C. Properties

A number of properties are worth to be noted with respect to the pattern collocations. First of all, they show important relationships between contextual patterns, and the relationships generally denote interesting move exchanges. Given a pattern collocation $[P, Q]$, it denotes the anchor move of Q of one player as an immidiate response to the anchor move of P of her opponent, if the anchor moves are of different colors. Otherwise, the collocation denotes two consecutive moves of one player in local contexts when her opponent chooses to play elsewhere.

Secondly, pairs of contextual patterns with high frequency of occurrence do not necessarily lead to higher probability to reject the null hypothesis. Some pattern pairs that occur as frequently as 30 times do not qualify as pattern collocations; while many others that appear only once do qualify.

Thirdly, pattern collocations also have the "data sparseness" problem as contextual patterns. Of the 16067 game records processed, each contributes approximately 180 new pattern collocations out of about 212 encountered. Figure 6 shows the frequency distribution of the collocations of 9-by-9 contextual patterns, which also exhibits a Zipf's law like regularity.

V. Move Generation

Both contextual patterns and pattern collocations represent essential GO-playing knowledge that can be readily applied to GO-playing programs to improve move generation.

A. Contextual Patterns in Tsumego

Tsumego is a Japanese term adopted by the GO community referring to life-and-death problems in local settings. Search algorithms such as the alpha-beta search are commonly used to solve tsumego problems.

Effective search algorithms to tsumego problems typically include the following components:

1) Test of terminal conditions of search such that the search can return,
2) Use of transposition tables to avoid repetitive efforts on same board configurations, and
3) Use of heuristics to order move candidates during search.

While the test of terminal conditions is domain-specific, and the use of transposition tables is standard, the heuristics for move ordering are the key to the effectiveness of search algorithms.

```
Given a tsumego problem, find its
  solution of as a move sequence S;
for each move m in S loop
  Play m on board;
  if m is an attacking move then
    Insert CP_m into attack database;
  end if;
  if m is a defending move then
    Insert CP_m into defend database;
  end if;
end loop;
```

Fig. 7. Algorithm in pseudo code for acquisition of tusmego contextual patterns

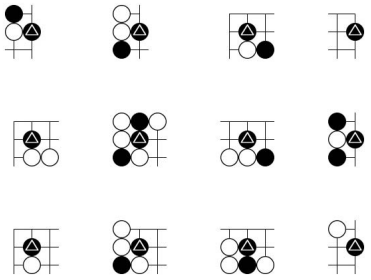

Fig. 8. Twelve most frequently used 3-by-3 attacking contextual patterns

Heuristics for move ordering heuristics are usually specified manually and are thus ad hoc in tsumego search. Manual and ad hoc specifications of heuristics often lead the following problems: dubious quality, limited quantity, and lack of consistency especially when the amount of heuristics reaches to a certain threshold. These problems often affect the quality of move ordering, which in turn makes searching in tsumego ineffective.

To address this issue, we have used contextual patterns as move ordering heuristics for tsumego search. Our acquisition of tsumego contextual patterns is conducted on tsumego exercise problems following a slightly modified algorithm in 1 of Section III that is shown in Figure 7.

This algorithm is envoked after a tsumego problem is solved, typically through a search, and tsumego contextual patterns are then extracted. We have run this algorithm along with a rudimentary version of the minimax search without move ordering heuristics on 200 tsumego problems from [11] for training. As results, two databases of contextual pattern are constructed, one for attacking in tsumego and the other for defending. Figure 8 and Figure 9 show, respectively, twelve most frequent 3-by-3 contextaul patterns in both the attacking and the defending database.

B. Pattern Collocations in Joseki

Joseki is a Japanese term adopted by the GO community referring to agreed-upon sequences of play. A joseki is a "settled pattern" consisting of a sequence of moves that results in a relatively fair outcome normally accepted by both players.

While an actual play of a joseki pattern is just a sequence of moves, many alternatives exist for each of the moves.

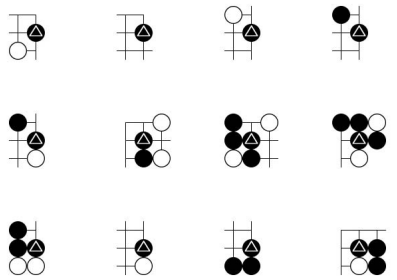

Fig. 9. Twelve most frequently used 3-by-3 defending contextual patterns

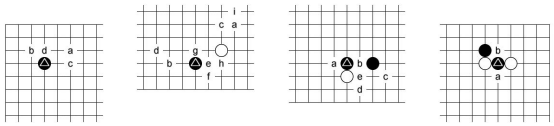

Fig. 10. Portions of joseki patterns shown in 9-by-9 pre-patterns of four sets of pattern collocations

Consequently, an appropriate data structure representation of moves in a joseki pattern is not just a linear list but a directed graph, because of the presence of different paths of moves from one board configuration to another. In such a directed graph representation, vertices are board configurations and edges are moves from one board configuration leading toward another directly. Because the development of GO playing is a changing process from a historical point of view, a directed graph representation of joseki patterns must be dynamic as well. Modifications occur not only on structures of the representation but also on its properties. While theoretically possible, representing all joseki patterns directly as directed graphs would require a prohibitively large amount of space and significant amounts of time for structure and value modifications, making this approach infeasible in practice.

Given a contextual pattern P, consider the set of all pattern collocations whose pre-pattern is P, which is denoted as $[P, *]$, where $*$ is a wildcard matching any contextual patterns. $[P, *]$ represents a portion of joseki pattern knowledge, i.e., a vertex in the directed graphs and all edges leading from the vertex. Figure 10 shows four such vertices, in which post-pattern moves of the opponent are marked in lower cases alphabetically following the order of their probability to reject the null hypothesis. It is clear that they all show joseki pattern moves in local contexts, and the order of the joseki pattern moves are sorted based on their relevance with respect to the contexts.

VI. EXPERIMENTS

Contextual Patterns and Pattern Collocations can be widely used in Computer Go. Not only in traditional Go programs, they can also play an important role in Go program with Monte-Carlo Tree Search(MCTS)[15]. It can be used for simulation, prunning and sorting playable points to make

TABLE I
EXPERIMENTS BETWEEN FOPS AND MOGO PATTERNS

frequency ordered patterns(FOPs)	Black:FOP, White:Mogo Patterns	Black:Mogo Patterns, White: FOP
Top-300	31:68	59:41
Top-400	55:45	42:58
Top-500	43:57	51:49
Top-800	30:70	64:36
Top-1000	36:64	61:39
Top-2000	31:69	67:33

MCTS more efficient.

A. Using Contextual Patterns for Simulation

In simulation, contextual patterns with frequency ordered can be used directly to generate moves. We design experiments to make comparison between frequency ordered patterns and mogo patterns [14]. We only use the frequency above 1.0, and we let the program plays 100 games with each other for each color. The results are shown in Table I , and we can get that 400 is a threshold of frequency ordered patterns.

From the result we can find that there may be an appropriate threshold to make a stronger program which required more test. Besides, as the limitation of the game records, there must be some obviously inaccurate frequency value of patterns, and this need the experienced human player to adjust it. And we believe that this may be a main reason that the more patterns cause weaker program. When these are done, the program may be reach a higher level.

B. Some Other Usages

Contextual Patterns can also be used for prunning the search tree, orderring moves to improve the efficiency of search. And the frequency can also be used to initialize the search tree to make the search more efficient.

VII. CONCLUSIONS

In summary, this paper has presented the concepts of contextual pattern and pattern collocation as a systematic approach of knowledge representations of GO playing, described the algorithms for efficient acquisition of contextual patterns and pattern collocations from game records and other sources of GO knowledge, and demonstrated the effectiveness of contextual patterns and pattern collocations in move generation. Contextual patterns and pattern collocations have the following key advantages in comparison with other manual and ad hoc representations of GO-playing knowledge:

1) Quailty knowledge. The knowledge represented in contextual patterns and pattern collocations is acquired automatically from game records of the professional, solutions of GO-playing exercises, and so on.
2) Efficient acquisition. Our complexity analysis of the acquisition algorithms shows that the acquisition of contextual patterns and pattern collocations is efficient and feasible in practice.
3) Effectiveness. We believe that it is effective to model moves based on their contexts of surroundings when GO-playing knowledge is constructed. The effectiveness of contextual patterns and pattern collocations is evident from the move generation applications presented.

Nevertheless, certain issues exist and should be properly addressed before contextual patterns and pattern collocations can be used more effectively in GO playing. The most severe one is the "data sparseness" problem, as discussed in Section III and IV. This is also the key reason that small contextual patterns are in general more effective than larger ones in tsumego, since known solutions to tsumego problems are rather limited and an exact match of large contextual patterns is rare. Abstraction mechanisms are needed to analyze and classify large amounts of individual contextual patterns into a higher-level representation of contextual knowledge of GO playing. This is a direction for future research.

VIII. ACKNOWLEDGEMENTS

The author is indebt to Professor Jin-tong Lin, the president of Beijing University of Posts and Telecommunications for helping establish the Computer GO Research Institute, which makes our research works possible. The author thanks Mr. Qing Dou for helping conduct many experiments reported in this paper, and thanks Mr. Yu, Ping and Mr. Yu, Bin, both professional GO players for insightful discussions of the game and for the collection of professional game records used in this paper.

REFERENCES

[1] D. B. Benson, "Life in the game of GO", *Information Sciences* **10**, 17–29, Reprinted in Computer Games, D. N. L. Levy (Ed), Vol. II, Springer-Verlag, New York, N. Y. (1976), 203–213.
[2] D. Biddle, "On the relative values of the chessmen" (1883).
[3] B. Bouzy, "Go patterns generated by retrograde analysis", *Computer Olympiads* (2001).
[4] B. Bouzy and G. Chaslot, "Bayesian generation and integration of k-nearest-neighbor patterns for 19x19 Go", *IEEE Symposium on Computational Intelligence in Games* (2005).
[5] B. Bouzy and G. Chaslot, "Monte-Carlo Go reinforcement learning experiments", *IEEE Symposium on Computational Intelligence in Games* (2006).
[6] T. Cazenave, "Generation of patterns with external conditions for the game of Go", *Advances in Computer Games Conference*, Padeborn (1999).
[7] K. Chen and Z. Chen, "Static analysis of life and death in the game of Go", *Information Sciences* **121** (1999).
[8] C. Cho, Go: A Complete Introduction to the Game, Kiseido Publishing Co (1997).
[9] Y. Choueka, "Looking for needles in a haystack or locating interesting collocational expressions in large textual databases", *Proceedings of the RIAO* (1988), 609–623.
[10] H. J. van den Herik, J. W. H. M. Uiterwijk, and J. van Rijswijck, "Games solved: now and in the future", *Artificial Intelligence* **134** (2002), 277–311.
[11] C. H. Lee, Lee Chang Ho's Selected Tsumego Problems, vol 1–6, Press of Beijing University of Physical Education, (2001) (in Chinese).
[12] J. Schaeffer and H. J. van den Herik, "Games, Computers, and Artificial Intelligence", *Artificial Intelligence* **134** (2002), 1–7.
[13] G. K. Zipf, Human Behavior and the Principle of Least Effort. Cambridge: Addison-Wesley Press (1949).
[14] Sylvain Gelly, Yizao Wang, Reḿi Munos, Olivier Teytaud, Modification of UCT with Patterns in Monte-Carlo Go (2006)
[15] Guillaume Chaslot, Sander Bakkes, Istvan Szita and Pieter Spronck. Monte-Carlo Tree Search: A New Framework for Game AI