PREDICTING THE NEXT MOVE: PATTERN GENERATION IN THE GAME OF GO

BY- ADITI SINGH, 09BCE003

GUIDED BY- PROF. KUMAR K

ABSTRACT

- a data-driven technique
- models based on PAST EXPERIENCES of other players
- ASSUMPTION → predict what a player will do in a given situation

  →from data of former players in similar situations

- WHY PATTERNS?
  - Rules in machine understandable form
  - Ease of generation
  - Use to limit valid moves in search trees

- WHY NOT GNU Go PATTERNS?
  - Hand made
  - Prone to errors
  - Not all covered

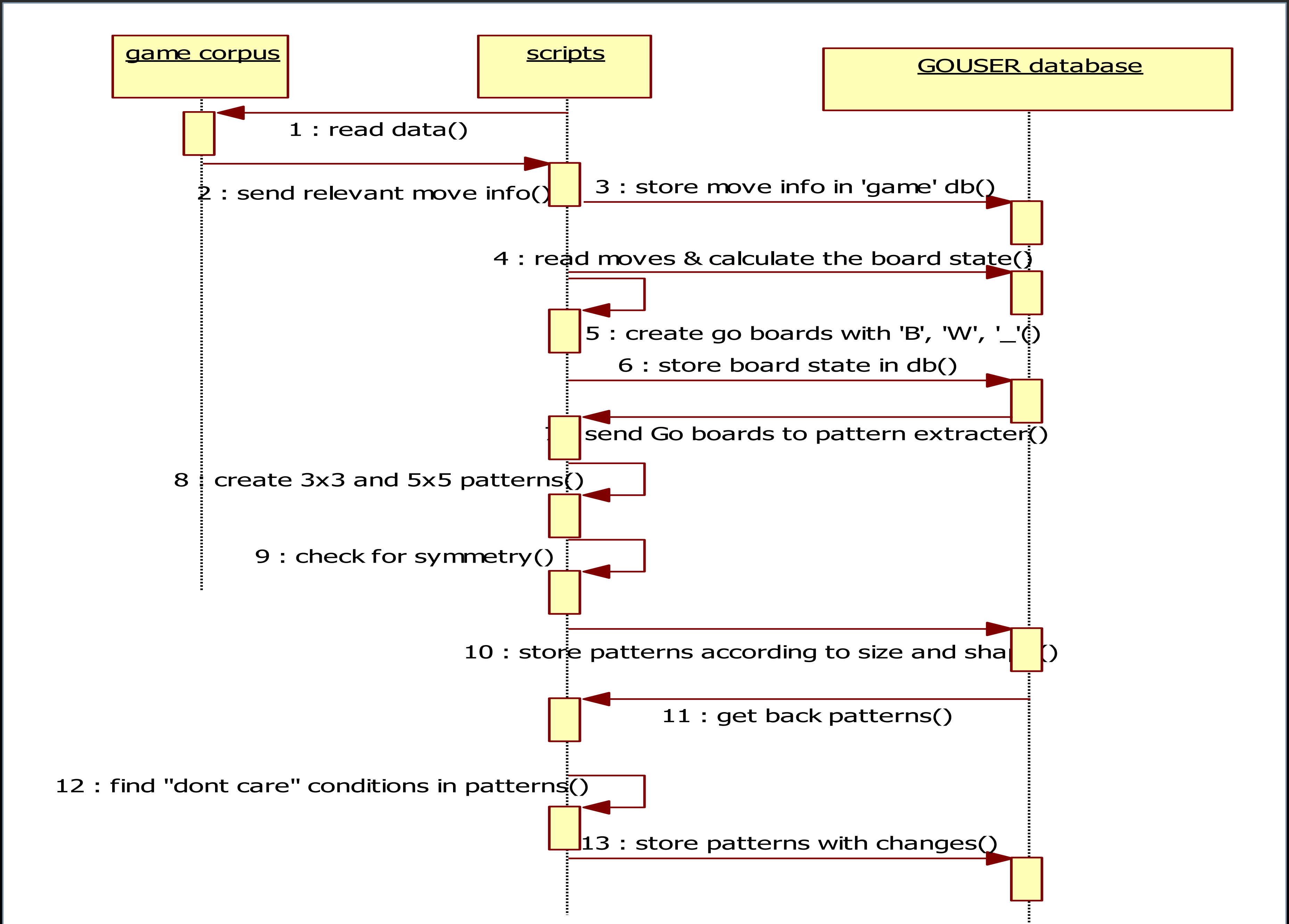
DESCRIPTION

- Automatic Pattern generation from larger game corpuses
- JOSEKI →moves in the corner for which black and white are balanced
- FUSEKI → opening go board theory
- These two classes are studied to gain local advantage.
- Since, learning JOSEKI patterns , are shown not to increase the overall strength of player, thus patterns not generated.
- Thus general patterns are extracted .

PREREQUISITES (STEPS):

1. Collect professional game records
2. Read all files via scripts
3. Store the move sequence plus the players' data in a database
4. Retrieve only move sequences from DB
5. Create a sample game board with 'B','W','_' for the moves.
6. Only spatial positioning is required for pattern generation.
7. ROTATION , REFLECTION and COLOR FLIP module created to remove canonical forms.

SEQUENCE DIAGRAM


game corpus
scripts
GOUSER database
1 : read data()
2 : send relevant move info()
3 : store move info in 'game' db()
4 : read moves & calculate the board state()
5 : create go boards with 'B', 'W', '_'()
6 : store board state in db()
7 : send Go boards to pattern extracter()
8 : create 3x3 and 5x5 patterns()
9 : check for symmetry()
10 : store patterns according to size and sha ()
11 : get back patterns()
12 : find "dont care" conditions in patterns()
13 : store patterns with changes()

AIM

- **Find patterns from SGF files $\rightarrow$ *electronic versions of professional games***
- **Find frequency of original patterns $\rightarrow$ *remove canonical forms***
- **Different sizes and two primary shapes $\rightarrow$ *square, diamond***
- **Find its K bit data $\rightarrow$ *number of bits of info per diamond pattern.***

OBJECTIVE

- To make this process simpler and more efficient
- patterns are generated, not by hand but by writing simple scripts.
- Determine what to do when board state matches previous game records
- Players will react similarly when put in a similar situation
- Go game tree in Monte Carlo simulation → search for random moves
- Limit this by starting out with a high frequency pattern
- Smaller exploration → higher exploitation

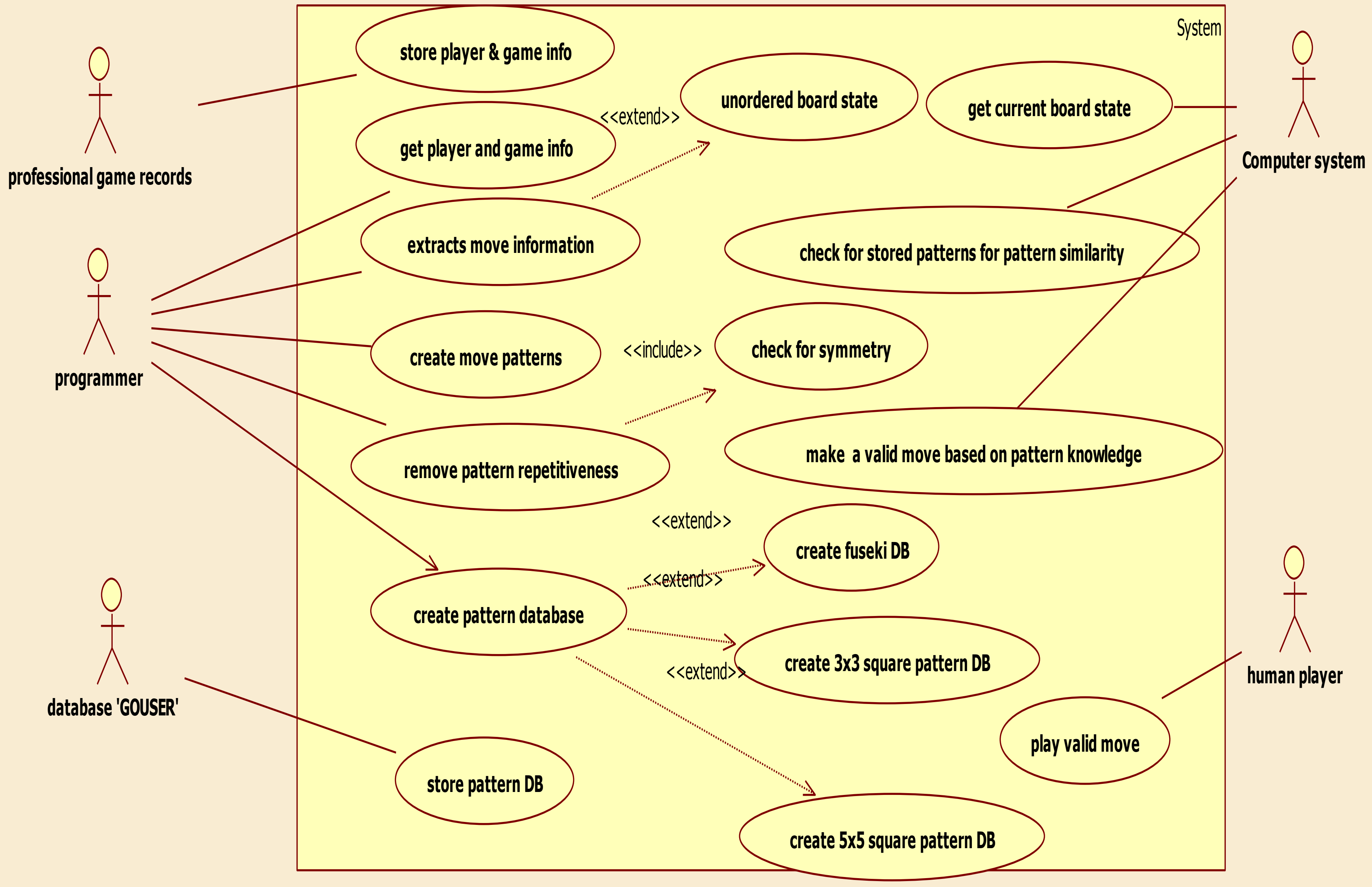
PROBLEM DESCRIPTION

- **An *n*-gram is a continuous sequence of *n* items from a given sequence of text or speech.**
- **DRAWBACK:**
- **GO patterns don’t follow CHRONOLOGICAL ORDER.**
- **So move sequences with order not required**
- **Although there are many patterns dictionaries available, the difference in their shapes and size limit the applicability.**
- **Repetition by symmetry removed → 1 pattern == 12 canonical forms**
- **4 are obtained due to the rotation operation,**
- **2 by reflection along x and y axis**
- **6 are due to the flipping operation.**
- **The shapes are in the form of a 3x3, 5x5 and by a 5x5 in the form of a diamond.**
- **In the diamond they are sorted according to the information bits in it.**
- **Plus they are sorted according to their frequency of its usage in the other game records.**
- **Pattern templates are also created when one of the positions is subject to the “don’t care” condition**

OTHER METHODS

1. GOAL SEARCH
2. HEURISTIC EVALUATION METHOD
3. MONTE-CARLO → UCT (UPPER CONFIDENCE TREE)
4. NEURAL NETWORKS
5. CELLULAR AUTOMATA

DESIGN AND ANALYSIS

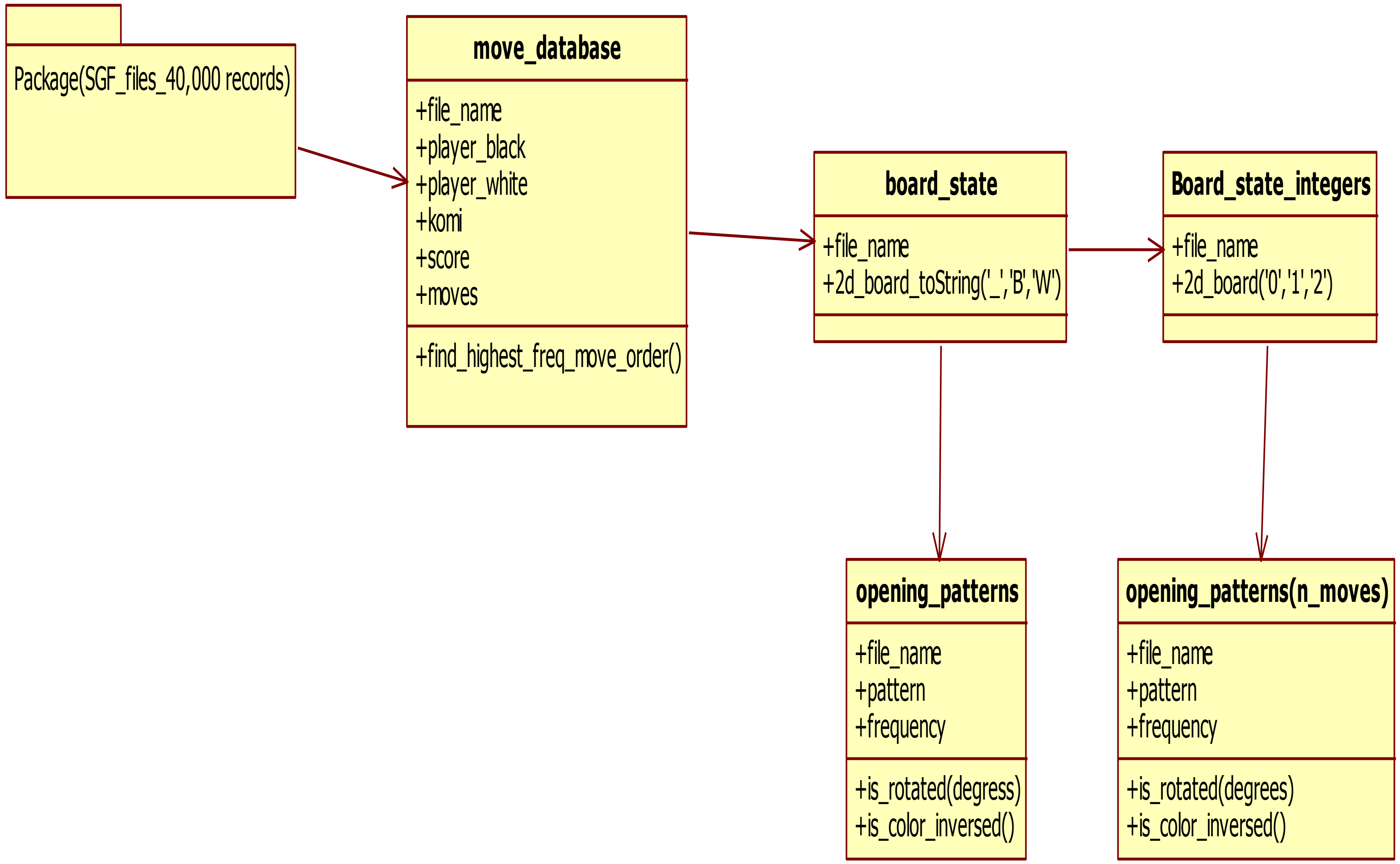


USE CASE ACTORS

- PROFESSIONAL GAME RECORDS →these are digitized game corpuses which contain 36000+ professional games in the SGF format
- DATABASE ‘GOUSER’→ this is the database collection which has tables ‘board_state’, ‘fuseki’, ‘pattern_db’, ‘move’, ‘game’.
- PROGRAMMER→ does the data extraction , scripting for the pattern generation and stores them into the database
- HUMAN PLAYER→ uses his game knowledge to make the move after the system has made his move on the go game board.
- COMPUTER SYSTEM → system knowledge database is in the form of a pattern database which contains the different types and shapes of the patterns (both square and diamond) in sizes of 3x3 and 5x5.

  →It uses the patterns to match the board state currently under play and based on the pattern makes the best possible move.

CLASS DIAGRAM



IMPLEMENTATION

- Collect records of professional go players.
- Use the records in SGF format to see the results in GOGUI software, for analysis.
- Use JAVA and MySQL to read the SGF files.
- Use players' info as the primary key for table and store the move text in a column
- Use the database records and find patterns in the data,
- which can be used for predicting the type of attack. Remove its canonical forms by checking for symmetry by rotation, reflection and colour flips.
- Pattern frequency generation program is used to get the "FUSEKI" and "JOSEKI" patterns.
- To select the next best move by the computer, use the pattern analysed result to filter the best local move available
- find the don't care patterns and remove its canonical forms

FINDING AND ADDING THE COMMON FUSEKI PATTERNS IN DB.

- **SAMPLE OUTPUT:**
- **Database: gouser**
- **Table : fuseki**
-
- **mysql> select * from fuseki where frequency >100;**

Pattern	**Frequency**
;B[pd];W[dd];B[qp];W[dq];B[do]; W[co];B[dp];W[cp];B[eq];W[cn]	**104**

FINDING THE 3X3 PATTERNS

```
.....original..........................

_ _ _

B _ W

B B B

.................inverse........................

_ _ _

W _ B

W W W

...............original_90....................

B B _

B _ _

B W _

...............original_90_inverse.......

W W _

W _ _

W B _
```

```
...............original_180..................

B B B

W _ B

_ _ _

...............original_180_inv...........

W W W

B _ W

_ _ _

...............original_270................

_ W B

_ _ B

_ B B

...............original_270_inv..........

_ B W

_ _ W

_ W W
```

```
...............original_x......................

B B B

B _ W

_ _ _

...............original_x_inv...............

W W W

W _ B

_ _ _

...............original_y....................

_ _ _

W _ B

B B B

...............original_y_inv.............

_ _ _

B _ W

W W W
```

FINDING THE DIAMOND SHAPED PATTERNS- K-NEAREST PATTERN

[7142]...

```
    _
  _ _ _
_ _ B B _
  _ B W
    W
```

Frequency=8

K bits of info=5

[9794]...

```
    B
  _ B B
B B B W W
  W B B
    W
```

Frequency=6

K bits of info=12

[5198]...

```
    W
  W B W
B W B B B
  B B _
    _
```

Frequency=37

K bits of info=11

REDUCING THE PATTERNS USING DON’T CARE CONDITIONS

```
.....[10729]......     .....[10731]......     .....[10733]......

_ W _                  _ W _                  _ W _
_ W _                  ? W _                  _ W _
B W ?                  B W W                  B ? W

.....[10730]......     .....[10732]......     .....[10734]......
_ ? _                  _ W _                  ? W _
_ W _                  _ W _                  _ W _
B W W                  ? W W                  B _ B
```

REFERENCES

[1] Documentation about SGF format. http://www.red-bean.com/sgf/

[2] American Go Association. http://www.usgo.org.

[3] GNU go documentation, http://www.gnu.org/software/gnugo/gnugo_toc.html

[4] Brent Harrison, David L. Roberts, *Using Sequential Observations to Model and Predict Player Behaviour*.

[5] Sylvain Gelly, Yizao Wang , R´emi Munos , Olivier Teytaud, *Modification of UCT with Patterns in Monte-Carlo Go*, 2006

[6] LIU Zhi-qing, DOU Qing, Automatic pattern acquisition from game records in go, THE JOURNAL OF CHINA UNIVERSITIES OF POSTS AND TELECOMMUNICATIONS, Volume 13, Issue 4, December 2006

[7] T. Cazenave. *Generation of patterns with external conditions for the game of GO.*

[8] Sylvain Gelly a,1, David Silver, Monte-Carlo tree search and rapid action value estimation in computer Go

[9] Woo-Jun PARK, *Representation and Comparison of Fuseki Patterns of Go-games*

[10] Emil H.J., *Learning Patterns in the Game of Go* , MSc Thesis

[11] Martin M¨uller. *Computer go. Artificial Intelligence*, 134(1):145–179, 2002.

[12] Brett Alexander Harrison, *Move Prediction in the Game of Go*, thesis for Harvard College

Cambridge, Massachusetts,April 1, 2010

[13] Wei Xin, Sun Yinglong, Yang Hui, Wang Jiao(Corresponding Author), *The Research of Pattern Symmetry Problem in Learning in Computer Go*, IEEE 2012.

[14] Nakamura T., .Acquisition of move sequence patterns from game record database using n-gram statistics., In Game Programming Workshop 97, 1997.

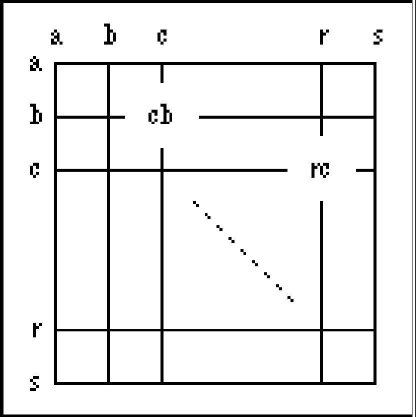
Framework:

4.1.1. Data Integration: First of all the data are collected and integrated from all the different sources. The data was collected from sources such as http://www.go4go.net/go, http://www.u-go.net/links/gamerecords/ etc. These files are in the SGF file format [1]. Some files are even sorted player and tournament wise.

SGF FILE SPECIFICATION:

In Go the Stone becomes Point and the Move and Point type are the same: two lowercase letters.

Coordinate system for points and moves



The first letter designates the column (left to right), the second the row (top to bottom). The upper left part of the board is used for smaller boards, e.g. letters "a"-"m" for 13*13.

A pass move is shown as '[]' or alternatively as '[tt]' (only for boards <= 19x19), i.e. applications should be able to deal with both representations.

4.1.2. Data Selection: in this step select only those data which is useful for data mining. Of all the various data fields that we obtain, only the required data fields is to be selected.

OUTPUT:

```
**mysql> desc game;
+-----------+-----------+------+-----+---------+-------+
| Field     | Type      | Null | Key | Default | Extra |
+-----------+-----------+------+-----+---------+-------+
| file_name | char(100) | NO   | PRI |         |       |
| file_info | text      | YES  |     | NULL    |       |
+-----------+-----------+------+-----+---------+-------+
**mysql> desc move;
+--------------+-----------+------+-----+---------+-------+
| Field        | Type      | Null | Key | Default | Extra |
+--------------+-----------+------+-----+---------+-------+
| file_name    | char(20)  | NO   | PRI | NULL    |       |
| player_white | char(20)  | YES  |     | NULL    |       |
| player_black | char(20)  | YES  |     | NULL    |       |
| komi         | char(10)  | YES  |     | NULL    |       |
| result       | char(100) | YES  |     | NULL    |       |
| move_made    | text      | YES  |     | NULL    |       |
+--------------+-----------+------+-----+---------+-------+
**mysql> desc board_state;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| file_name | char(20) | NO   | PRI |         |       |
| state     | blob     | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
```

4.1.3. Data Cleaning: The data we have collected are not clean and may contain errors, missing values, noisy or inconsistent data. So we need to apply different techniques to get rid of such anomalies.

OUTPUT:

```
mysql> select * from move where file_name="1.sgf";
+-----------+-------------+-------------+---------+----------+-----------------------------------------------
| file_name | player_white | player_black | komi    | result    | move_made|
+-----------+-------------+-------------+---------+----------+-----------------------------------------------
|    1.sgf                    |    WR[8p]                              |    BR[8p]                              |    KM[5.5]    |    RE[B+2.5]    |
;B[qd];W[dc];B[dp];W[pq];B[ce];W[oc];B[ld];W[of];B[oe];W[ne];B[pe];W[md];B[nf];W[le];B[mf];W[k
d];B[og];W[qn];B[lc];W[me];B[kc];W[jd];B[jc];W[id];B[mb];W[nb];B[mc];W[nc];B[hb]
;W[gc];B[hc];W[hd];B[gd];W[de];B[df];W[ee];B[cd];W[gb];B[ic];W[ge];B[jq];W[cc];B
[ci];W[cq];B[cp];W[dq];B[eq];W[er];B[fq];W[fr];B[gq];W[bp];B[bo];W[bq];B[cn];W[e
i];B[fd];W[fe];B[ql];W[on];B[pj];W[mq];B[qq];W[pr];B[mp];W[lp];B[nq];W[lq];B[pp]
;W[op];B[po];W[pn];B[ro];W[rn];B[qr];W[oo];B[qo];W[nr];B[rm];W[sn];B[rq];W[ek];B
[ck];W[gr];B[hp];W[rk];B[qk];W[rj];B[qi];W[rl];B[lf];W[kf];B[kg];W[jf];B[ri];W[q
b];B[rc];W[hr];B[jo];W[ch];B[dh];W[di];B[bh];W[cj];B[bi];W[eh];B[cg];W[ji];B[gl]
;W[em];B[gj];W[eo];B[ep];W[li];B[fn];W[lg];B[ih];W[gh];B[ii];W[ng];B[pf];W[mg];B
[of];W[jg];B[jj];W[kj];B[jk];W[iq];B[oq];W[np];B[kn];W[ip];B[kq];W[io];B[in];W[r
b];B[ps];W[or];B[qm];W[sm];B[nh];W[ni];B[mh];W[lh];B[oh];W[kl];B[jm];W[ho];B[dj]
;W[dk];B[bj];W[cl];B[bl];W[hm];B[hn];W[gm];B[gn];W[fm];B[kk];W[lk];B[jl];W[cm];B
[bm];W[ln];B[lo];W[lm];B[ga];W[fc];B[fb];W[ec];B[gi];W[fk];B[ko];W[mo];B[ml];W[l
l];B[nk];W[nj];B[bc];W[bb];B[bd];W[gk];B[jh];W[do];B[ki];W[lj];B[eg];W[hj];B[hi]
;W[hl];B[dm];W[dl];B[fh];W[fi];B[fg];W[dn];B[hf];W[ab];B[en];W[co];B[bn];W[fo];B
[go];W[fl];B[sb];W[qc];B[re];W[ma];B[la];W[na];B[ja];W[lr];B[jr];W[ol];B[ok];W[n
l];B[mk];W[mm];B[od];W[nd];B[eb];W[db];B[fa];W[gf];B[gg];W[hg];B[ig];W[if];B[kh]
;W[os];B[qs];W[so];B[sp];W[ac];B[ad];W[he];B[ij];W[si];B[sh];W[sj];B[sl];W[sk];B
[dd];W[ed];B[oi];W[hk];B[pc];W[pb];B[ap];W[aq];B[ao];W[hf];B[hh];W[pm];B[pd];W[l
s];B[is];W[hs];B[pl];W[oj];B[ra];W[sl];B[nm];W[om];B[qj];W[mj];B[pk];W[rh];B[sg]
;W[sd];B[sc];W[rf];B[sf];W[se];B[rd];W[se];B[sd];W[qh];B[pi];W[qf];B[qe];W[qg];B
[pg];W[qa];B[km];W[da];B[sa];W[ej];B[cj];W[hq];B[gp];W[ef];B[im]) |
```

4.1.4. Data Transformation: The data even after cleaning are not ready for mining as we need to transform them into forms appropriate for mining. The techniques used to accomplish this are smoothing, aggregation, normalization etc.

```
printing the int array
0 0 1 1 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0
2 1 1 2 1 1 1 2 0 2 0 0 1 0 2 0 0 0 0
2 2 2 2 2 1 2 1 1 2 2 1 0 0 1 1 0 0 0
1 2 2 1 1 2 2 0 1 1 2 0 0 2 2 0 1 0 0
2 2 1 0 1 2 0 2 0 0 0 0 0 2 1 1 0 0 0
2 2 1 1 1 1 1 2 2 0 2 2 2 1 1 0 0 0 0
2 1 1 2 2 2 2 0 0 0 2 1 1 2 0 0 0 0 0
2 2 1 2 0 1 0 2 0 0 0 0 0 1 0 0 0 0 0
1 1 0 1 1 0 2 0 0 1 2 0 0 0 0 0 0 0 0
0 1 1 1 2 1 0 1 1 2 2 0 0 0 0 0 0 0 0
0 2 2 1 2 2 0 2 1 0 0 0 0 0 0 1 0 0 0
0 2 1 2 1 2 2 0 0 0 0 1 0 0 1 2 1 0 0
0 0 2 2 1 1 1 2 2 2 2 2 0 0 2 1 2 1 0
0 0 0 0 2 0 1 1 1 1 1 2 2 2 2 2 1 1 0
0 2 2 2 1 1 0 0 0 0 0 1 0 1 0 1 2 0 0
0 1 2 1 2 2 0 0 0 0 0 1 0 1 0 1 2 0 0
0 1 1 1 1 2 0 0 0 1 0 0 0 0 0 2 2 0 0
0 0 0 1 2 0 0 0 0 1 0 2 0 2 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

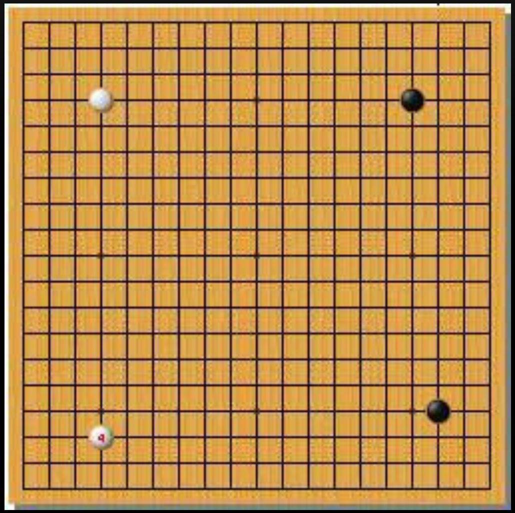
4.1.5. Data Mining: Now we are ready to apply data mining techniques on the data to discover the interesting patterns. Techniques like clustering and association analysis are among the many different techniques used for data mining.

OUTPUT:

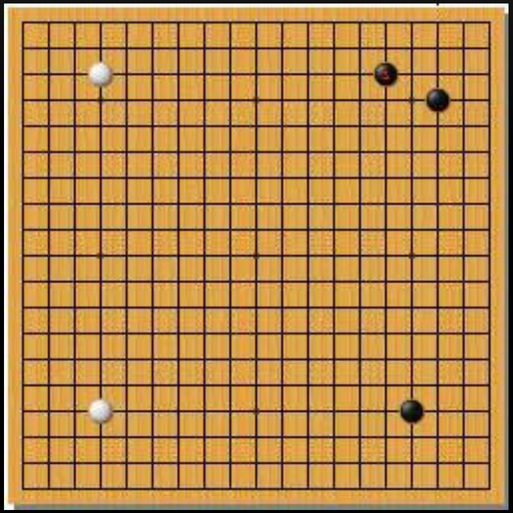
```
mysql> select file_name , move_made from move where move_made like"_B[qd]_W[ce]_B[pq]_W[od]%";
| file_name | move_made |
+-------------+----------------------------------------------------------------
| 6388.sgf   |

 ;B[qd];W[ce];B[pq];W[od];B[oc];W[nc];B[pc];W[md];B[qf];W[gc];B[cp]
;W[eq];B[hp];W[go];B[dn];W[cq];B[bq];W[dp];B[co];W[ho];B[jp];W[ip];B[iq];W[io];B
[hq];W[lp];B[jo];W[jn];B[ko];W[po];B[np];W[no];B[op];W[mo];B[kn];W[km];B[jm];W[i
n];B[lm];W[kl];B[ll];W[oo];B[qp];W[ln];B[kq];W[lq];B[fp];W[lk];B[nl];W[kk];B[pl]
;W[ol];B[om];W[nm];B[ok];W[ml];B[fq];W[br];B[dr];W[er];B[cr];W[dq];B[bs];W[ar];B
[aq];W[eo];B[lo];W[pp];B[mn];W[qq];B[qr];W[rq];B[or];W[rr];B[mr];W[qs];B[nn];W[e
n];B[dl];W[dj];B[jd];W[ge];B[lc];W[ld];B[kd];W[lb];B[kb];W[mc];B[kg];W[kc];B[jc]
;W[ng];B[ig];W[ie];B[je];W[if];B[mh];W[pe];B[pf];W[jf];B[kf];W[le];B[ke];W[qe];B
[re];W[of];B[ne];W[oe];B[rg];W[mg];B[lh];W[jg];B[jh];W[ih];B[ji];W[hh];B[il];W[i
k];B[gl];W[el];B[ek];W[fl];B[hk];W[ij];B[fk];W[dk];B[dm];W[hm];B[fn];W[fo];B[em]
;W[hl];B[cc];W[ec];B[cf];W[cd];B[ei];W[di];B[eh];W[dh];B[dc];W[df];B[eb];W[fb];B
[ed];W[fc];B[bd];W[be];B[bb];W[mp];B[lr];W[qm];B[nb];W[mb];B[ob];W[jb];B[bk];W[k
a];B[ql];W[rl];B[rk];W[ad];B[ac];W[bc];B[rn];W[rm];B[bd];W[rj];B[qk];W[bc];B[qn]
;W[pn];B[bd];W[ni];B[mj];W[bc];B[pm];W[ro];B[bd];W[mi];B[lj];W[bc];B[sl];W[qo];B
[bd];W[kj];B[li];W[bc];B[cg];W[dg];B[bd];W[mk];B[nh];W[bc];B[og];W[ab];B[sm];W[b
j];B[ck];W[ej];B[fj];W[fi];B[gi];W[fh];B[gh];W[fg];B[hj];W[bl];B[cl];W[ak];B[gg]
;W[gf];B[cj];W[bi];B[ci];W[ch];B[hg];W[bm];B[gn];W[jl];B[bn];W[pr];B[ic];W[nq];B
[nr];W[oq];B[jj];W[jk];B[ii];W[hs];B[is];W[hb];B[al];W[am];B[mm];W[bh];B[hd];W[g
d];B[an];W[aj];B[pd];W[lf];B[gs];W[gr];B[hr];W[fr];B[cm];W[al];B[ib];W[ia];B[so]
;W[gm];B[fm];W[ds];B[as];W[sp];B[lg];W[nf];B[ki];W[im];B[ma];W[nk];B[nj];W[os];B
[ns];W[kp];B[sn];W[ps]) |
```

4.1.6. Pattern Evaluation and Knowledge Presentation: This step involves visualization, transformation, removing redundant patterns etc. from the patterns we generated.



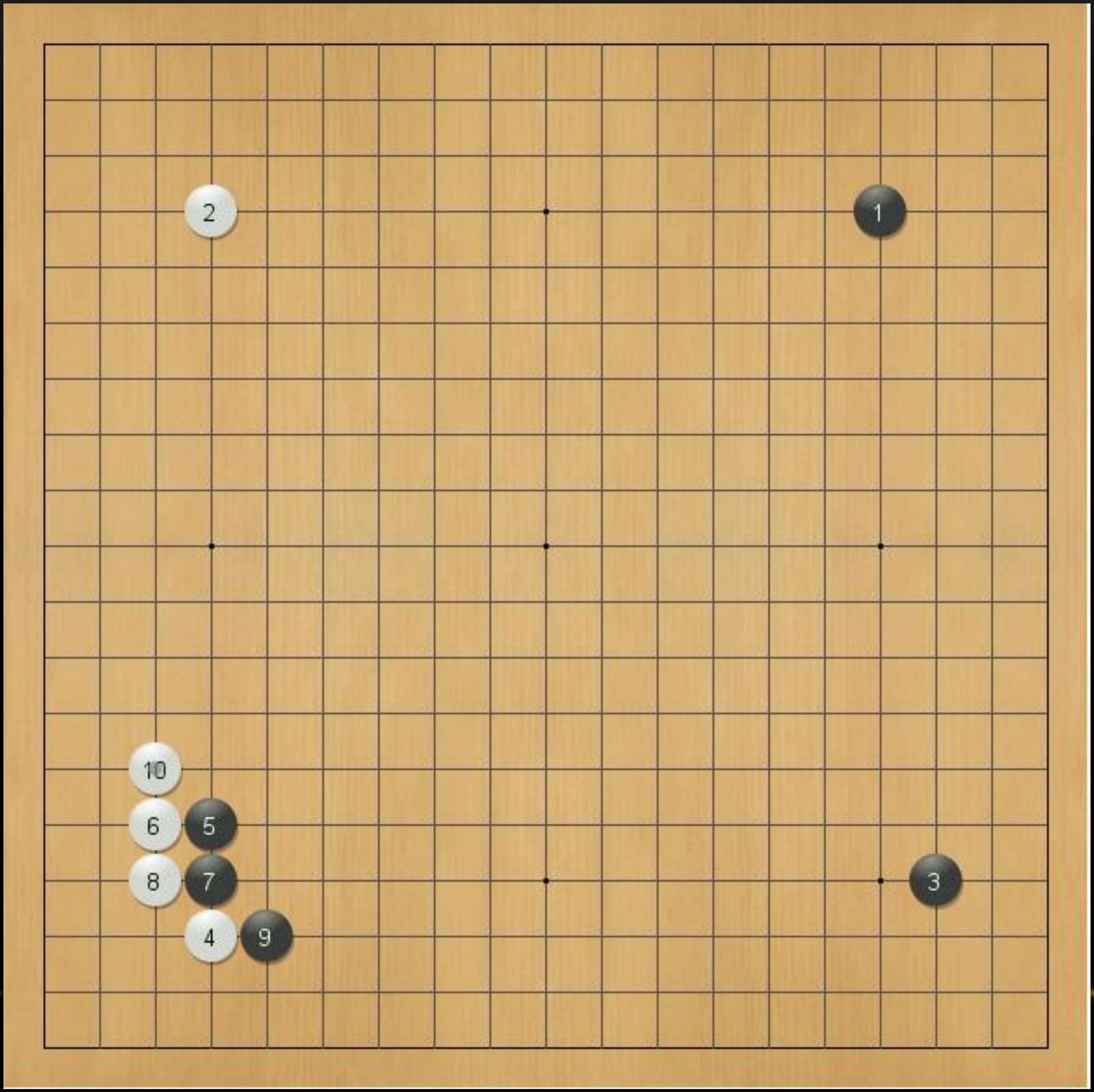
FUSEKI pattern:
parallel fuseki



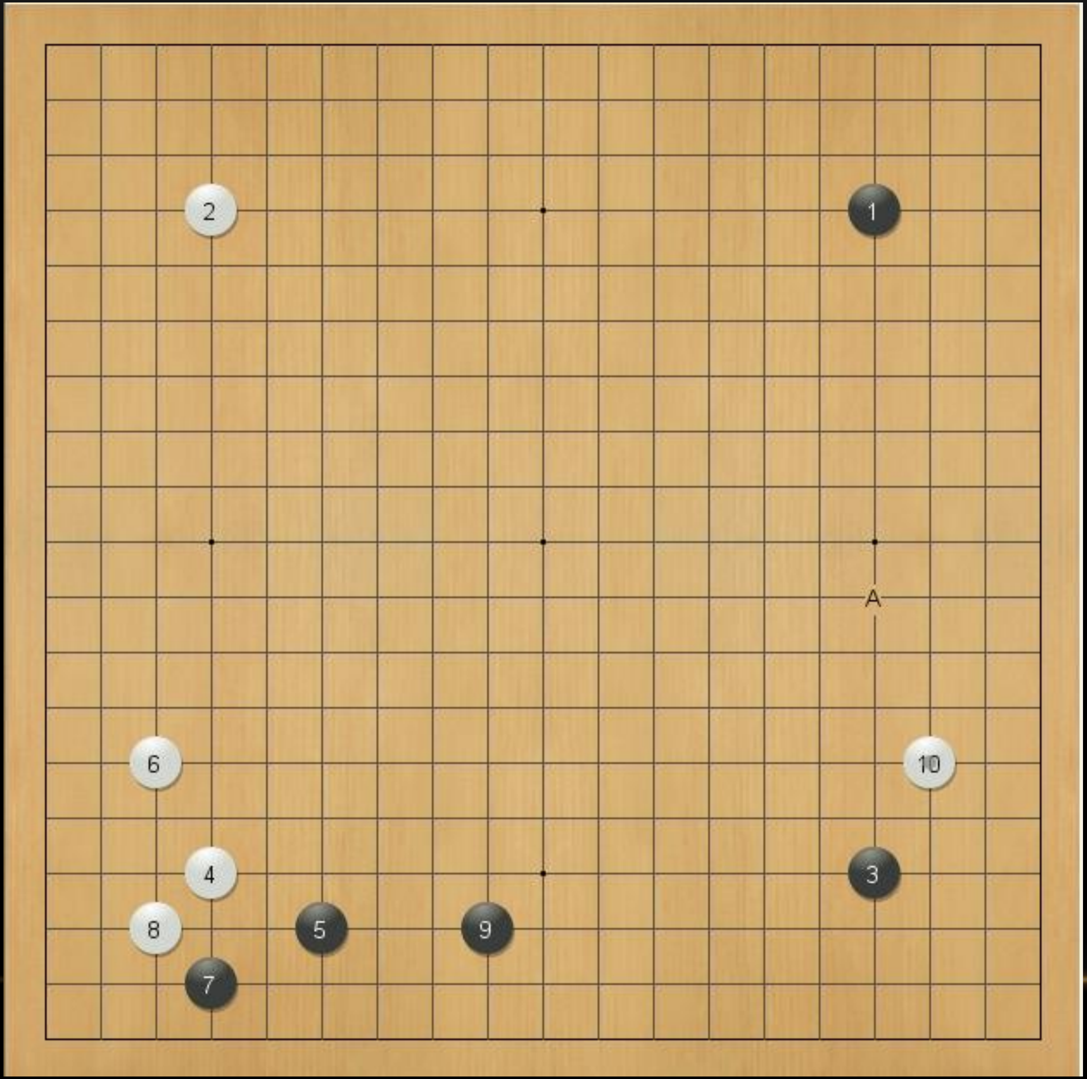
FUSEKI pattern:
Mark II Kobayashi formation

```
mysql> select frequency, pattern from fuseki where frequency>50;
+-----------+-------------------------------------------------------------+
| frequency | pattern                                                     |
+-----------+-------------------------------------------------------------+
|        82 | ;B[pd];W[dd];B[pp];W[dp];B[fq];W[cn];B[dr];W[cq];B[iq];W[qn]
|        63 | ;B[pd];W[dd];B[pp];W[dp];B[fq];W[hq];B[cq];W[dq];B[cp];W[do]
|        53 | ;B[pd];W[dd];B[pp];W[dp];B[pj];W[nc];B[lc];W[qc];B[qd];W[pc] |
|        78 | ;B[pd];W[dd];B[pp];W[dq];B[do];W[co];B[cn];W[cp];B[dn];W[fq] |
|        77 | ;B[pd];W[dd];B[pq];W[cp];B[ep];W[eq];B[fq];W[dq];B[fp];W[cn] |
|       104 | ;B[pd];W[dd];B[qp];W[dq];B[do];W[co];B[dp];W[cp];B[eq];W[cn]
|        54 | ;B[pd];W[dd];B[qp];W[dq];B[do];W[co];B[dp];W[cp];B[eq];W[cq] |
|        72 | ;B[pd];W[dp];B[pp];W[dc];B[de];W[ce];B[cf];W[cd];B[df];W[fc] |
|        55 | ;B[pd];W[dp];B[pp];W[dc];B[de];W[ce];B[cf];W[cd];B[fq];W[cn] |
|        64 | ;B[pd];W[dp];B[pp];W[dc];B[de];W[ce];B[dd];W[cd];B[ec];W[cf] |
|        71 | ;B[pd];W[dp];B[pp];W[dd];B[fq];W[cn];B[dr];W[cq];B[iq];W[qn] |
|        68 | ;B[pd];W[dp];B[pp];W[dd];B[fq];W[hq];B[cq];W[dq];B[cp];W[do] |
|        51 | ;B[pd];W[dp];B[pp];W[dd];B[pj];W[nc];B[lc];W[qc];B[qd];W[pc] |
|        54 | ;B[pd];W[dp];B[pq];W[dd];B[qk];W[nc];B[pf];W[pb];B[qc];W[kc] |
|        59 | ;B[qd];W[dc];B[pq];W[dq];B[oc];W[po];B[qo];W[qn];B[qp];W[pm] |
+-----------+-------------------------------------------------------------+
```

;B[pd];W[dd];B[qp];W[dq];B[do];W[co];B[dp];W[cp];B[eq];W[cn]



```
mysql> select file_name, move_made from move where move_made like "_B[pd]_W[dd]_
B[pp]_W[dp]_B[fq]_W[cn]_B[dr]_W[cq]_B[iq]_W[qn]%";
```



4.1.7. <u>Decisions / Use of Discovered Knowledge</u>: This step helps user to make use of the knowledge acquired to take better decisions.

Fuseki pattern

1. **Characteristics**

- **Less systematic : Since each move is typically isolated and unforced, patterns for play on the whole board have seen much less systematic study than for Joseki, which are often contact moves which require specific and immediate responses. Hence a game of Go may easily explore an unfamiliar path.**
- **Recognised names : Only a proportion of fusekis have recognised or specific names. These include the two-star fuseki (nirensei fuseki), three-star fuseki (sanrensei fuseki), Chinese fuseki, Kobayashi fuseki, andShusaku fuseki. These are names for the influential formations which Black makes on one side of the board.**

THE END