Approximate matching for Go board positions

Alonso GRAGERA

Department of Computer Science
Graduate School of Information Science and Technology
The University of Tokyo

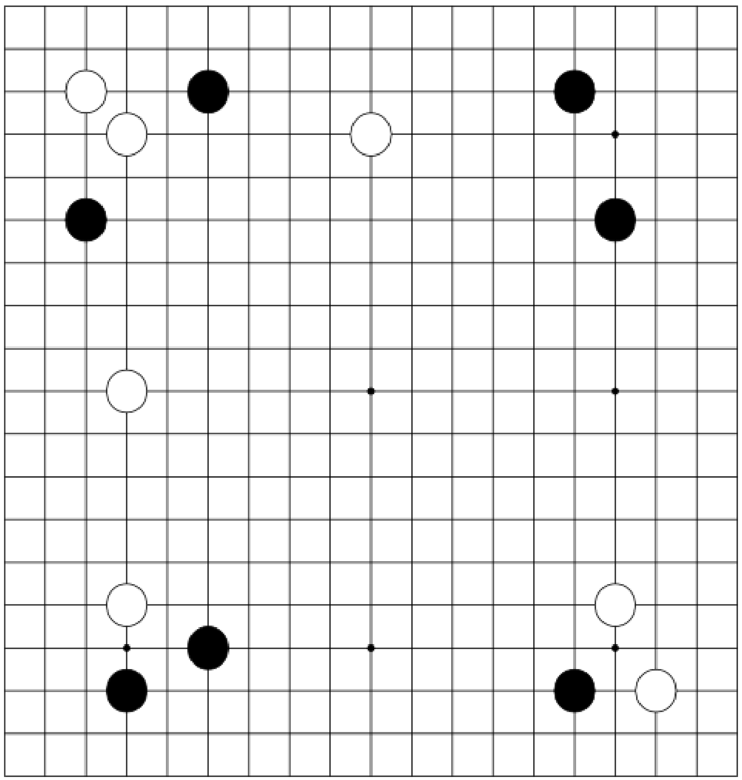
Today's content

- **Introduction to Go board positions**
- **Exact matching**
  - Zobrist hashing
- **Influence models**
- **Approximate matching** *(Our contribution)*
  - Similitude *a-posteriori*
  - Similitude *a-priori*
- **Applicability in subproblems**
  - Opening book construction
  - Message traffic reduction in massive parallelization
  - Professional game clustering
- **Experiments** *(Work In Progress)*
- **Summary**

Today's content

- **Introduction to Go board positions**
- **Exact matching**
  - Zobrist hashing
- **Influence models**
- **Approximate matching** *(Our contribution)*
  - Similitude *a-posteriori*
  - Similitude *a-priori*
- **Applicability in subproblems**
  - Opening book construction
  - Message traffic reduction in massive parallelization
  - Professional game clustering
- **Experiments** *(Work In Progress)*
- **Summary**

Introduction to Go board positions



The game of Go *(Baduk)*
Is a zero-sum, perfect-information, partisan, deterministic strategy board game involving two players, that originated in ancient China more than 2,500 years ago.

Board position
In Go a board position is given by the status *{empty, black, white}* of each of the *361* intersection that form the *19x19* go board.

東京大学 THE UNIVERSITY OF TOKYO

Today's content

- Introduction to Go board positions
- **Exact matching**
  - Zobrist hashing
- Influence models
- Approximate matching *(Our contribution)*
  - Similitude *a-posteriori*
  - Similitude *a-priori*
- Applicability in subproblems
  - Opening book construction
  - Message traffic reduction in massive parallelization
  - Professional game clustering
- Experiments *(Work In Progress)*
- Summary

Exact matching – Zobrist hashing

Zobrist hashing algorithm (Zobrist A., 1969)

1.- Generate a set of 2 (black and white) x 361 (intersections) random integers, creating a table with a random number for each intersection of each colour.

2.- XOR all together the values corresponding to the intersections occupied in the board position to obtain the key.

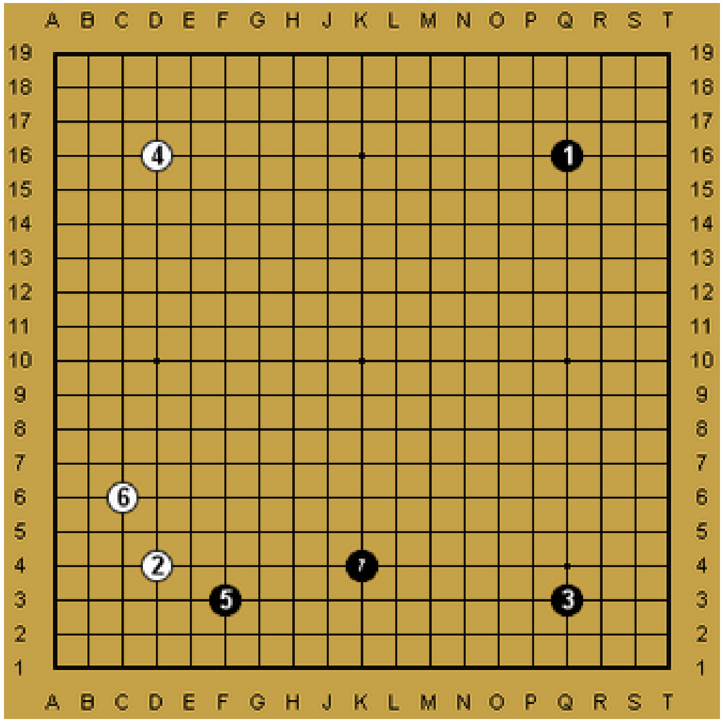
Properties:

- Each board position's key is uniformly random *(XOR property)*
- The collision probability can be easily controlled *(Key length)*
- Easy to recalculate *(XOR property)*

Exact matching – Zobrist hashing

Moves table

…
C6w = *677446201*
D4w = *368448338*
D16w = *1893209679*
F3b = *2250333161*
K4b = *820429074*
Q3b = *4113264331*
Q16b = *358864888*
…



Key

C6w *XOR* D4w *XOR* D16w *XOR* F3b *XOR* K4b *XOR* Q3b *XOR* Q16b = **465577196**

Example

Exact matching

Zobrist hashing is GREAT, but…

…sometimes the search space is just too big for exact matching

Problem

The equivalent of the *Shannon number* *(10^{120})*, estimation of the game tree size of chess, for Go is 10^{360} (Allis, 1994).

Regardless of how big we make our game database is still going to be a tiny fraction.

Move	Games
1st	102654
2nd	21124
3rd	10277
4th	9327
5th	6482
6th	3683
7th	1647
8th	632

Number of games of the most common sequence according to the [Fuseki Info for KGS Go Server]

東京大学 THE UNIVERSITY OF TOKYO

Today's content

- Introduction to Go board positions
- Exact matching
  - Zobrist hashing
- **Influence models**
- Definition of similitude *(Our contribution)*
  - *a-posteriori*
  - *a-priori*
- Applicability in subproblems
  - Opening book construction
  - Message traffic reduction in massive parallelization
  - Professional game clustering
- Experiments *(Work In Progress)*
- Summary

Influence models

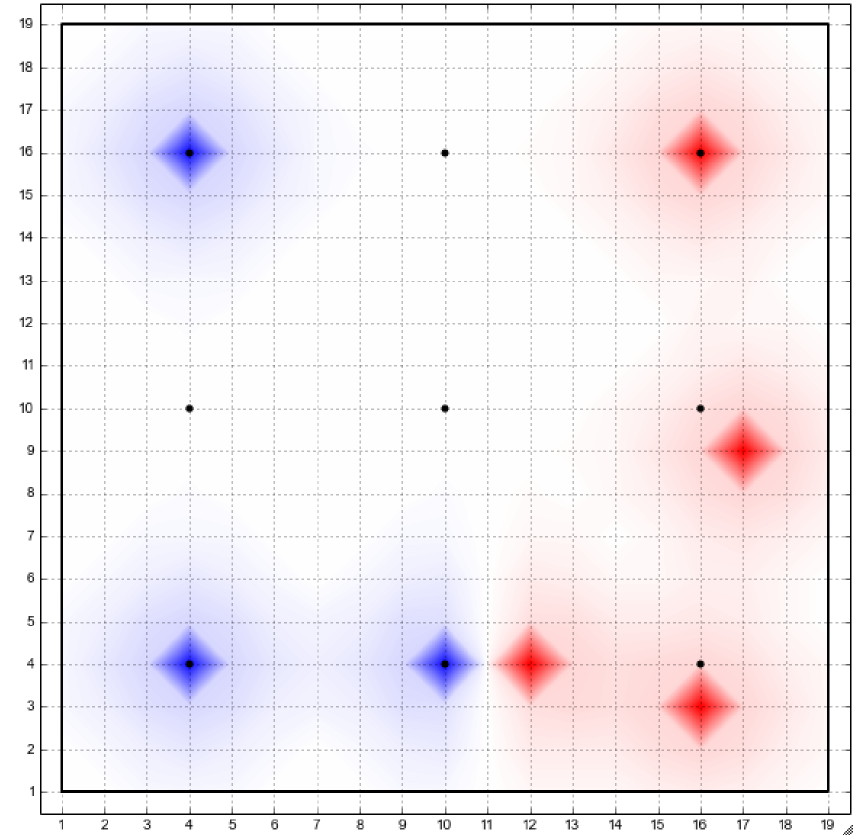
In Go influence models (or influence maps) are a representation of how much each stone affect to its surroundings.

Definition

$$f : \mathcal{P} \to \mathbb{Z}^{19\times 19}$$

Well-known models

- Zobrist's influence model
- Ryder's influence model
- Spight's influence model
- Bouzy's influence model



Today's content

- Introduction to Go board positions
- Exact matching
  - Zobrist hashing
- Influence models
- **Approximate matching** *(Our contribution)*
  - Similitude *a-posteriori*
  - Similitude *a-priori*
- Applicability in subproblems
  - Opening book construction
  - Message traffic reduction in massive parallelization
  - Professional game clustering
- Experiments *(Work In Progress)*
- Summary

Approximate matching

To cope with the problems of exact matching, while maintaining the condition of full board matching.

We introduce new approach based in the relaxation of the criterion of the matching, allowing to obtain close enough results.

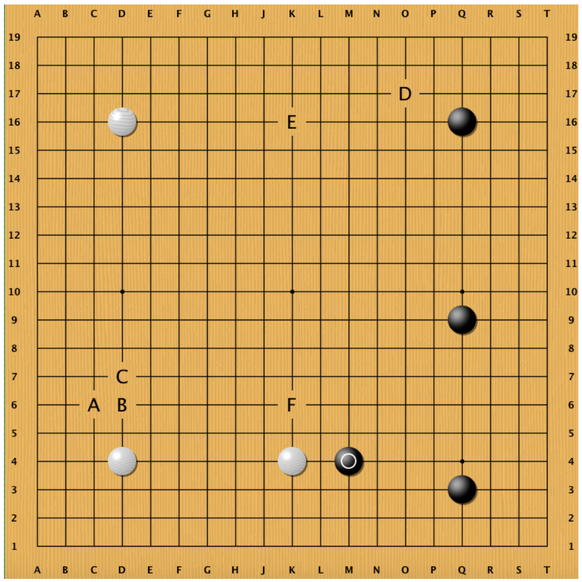
In order to be able to do so, we need to define one or more similarity measures.

Similitude *a-posteriori*

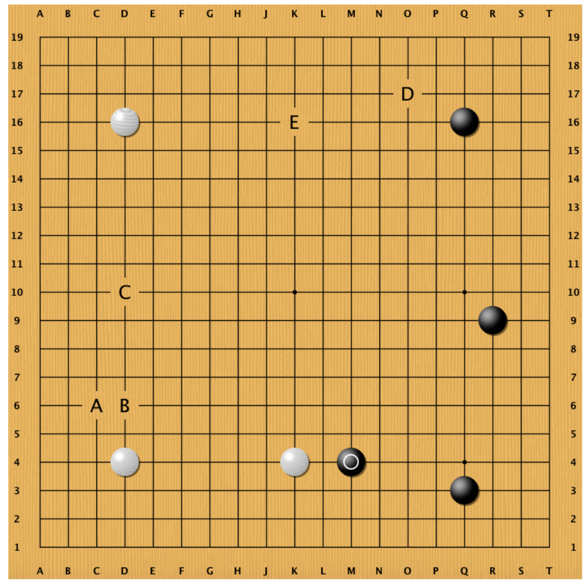
Definition (Similitude a-posteriori)**.** Let $next(x)$ be the set of follow up moves of the position $x \in \mathcal{P}$, we can define the similutude as

$$\hat{s}_{pos}(x, y) = 1 - \frac{2|next(x) \cap next(y)|}{|next(x)| + |next(y)|}$$

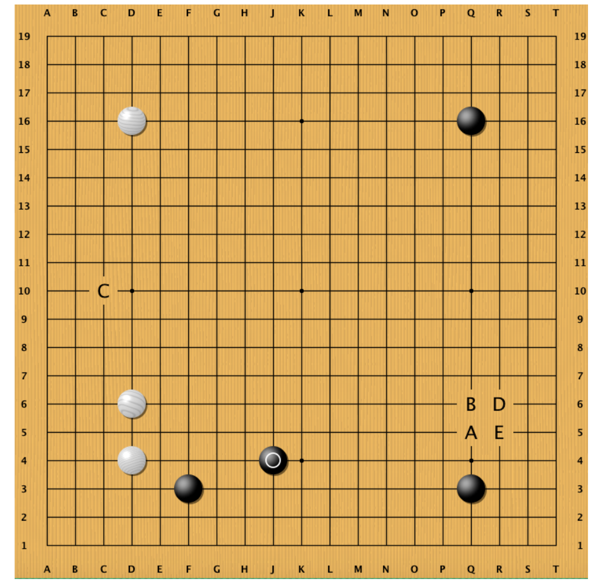
Similitude *a-posteriori*



(A)



(B)



(C)

$$\hat{s}_{pos}(A, B) = 0.72 \text{ and } \hat{s}_{pos}(A, C) = 0$$

Example

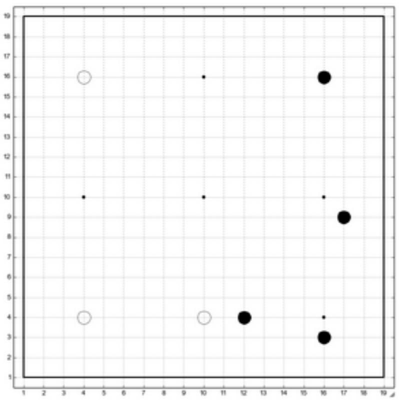
Similitude *a-priori*

Definition (Similitude a-priori). Let $\hat{S}(\mathcal{P}, \mathcal{P}, \mathcal{C})$ be a family of similitude measures between two board positions $x, y \in \mathcal{P}$, under a given influence model $f \in \mathcal{F}$, defined by

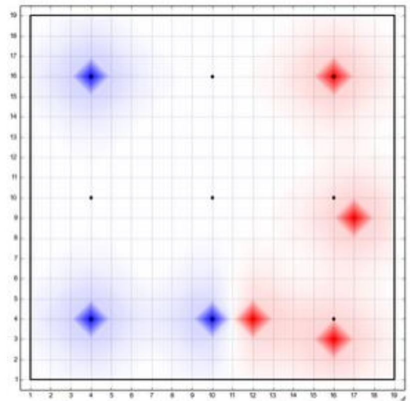
$$\hat{s}_f(x, y) = \begin{cases} 1 & if\ x = y \\ 1 - \dfrac{2}{\dfrac{1}{1 + e^{\alpha \sum \sum |f(x) - f(y)|}}} & otherwise \end{cases}$$

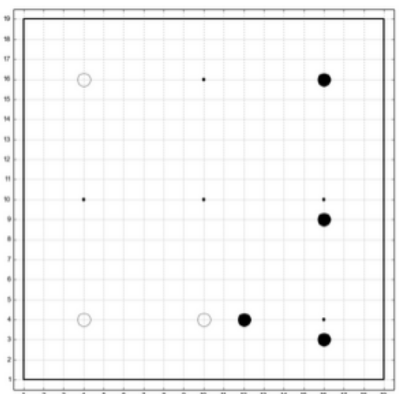
where α is a configurable model-dependent parameter.

Similitude *a-priori*

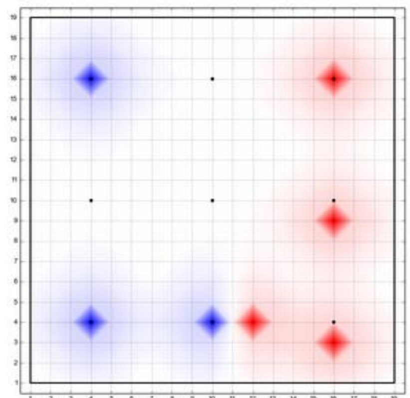


```
[  0,  0,  -2,  -2,  -2,  0,  0,  0,   0,   0, 0,  0,  0, 0,  2,  2,  2,  0, 0]
[  0, -2,  -4,  -6,  -4, -2,  0,  0,   0,   0, 0,  0,  0, 2,  4,  6,  4,  2, 0]
[ -2, -4,  -8, -10,  -8, -4, -2,  0,   0,   0, 0,  0,  2, 4,  8, 10,  8,  4, 2]
[ -2, -6, -10, -64, -10, -6, -2, -1,   0,   0, 0,  1,  2, 6, 10, 64, 10,  6, 2]
[ -2, -4,  -8, -10,  -8, -4, -2,  0,   0,   0, 0,  0,  2, 4,  8, 10,  8,  4, 2]
[  0, -2,  -4,  -6,  -4, -2,  0,  0,   0,   0, 0,  0,  0, 2,  4,  6,  4,  2, 0]
[  0,  0,  -2,  -2,  -2,  0,  0,  0,   0,   0, 0,  0,  0, 0,  2,  2,  3,  0, 0]
[  0,  0,   0,  -1,   0,  0,  0,  0,   0,   0, 0,  0,  0, 0,  0,  3,  2,  2, 0]
[  0,  0,   0,   0,   0,  0,  0,  0,   0,   0, 0,  0,  0, 0,  2,  4,  6,  4, 2]
[  0,  0,   0,   0,   0,  0,  0,  0,   0,   0, 0,  0,  0, 2,  4,  8, 10,  8, 4]
[  0,  0,   0,   0,   0,  0,  0,  0,   0,   0, 0,  0,  1, 2,  6, 10, 64, 10, 5]
[  0,  0,   0,  -1,   0,  0,  0,  0,   0,  -1, 0,  1,  0, 2,  4,  8, 10,  8, 4]
[  0,  0,  -2,  -2,  -2,  0,  0,  0,  -2,  -2, 0,  2,  2, 0,  2,  5,  6,  4, 2]
[  0, -2,  -4,  -6,  -4, -2,  0, -2,  -4,  -5, 0,  5,  4, 2,  2,  4,  4,  2, 0]
[ -2, -4,  -8, -10,  -8, -4, -3, -4,  -8,  -8, 0,  8,  8, 5,  5,  6,  5,  2, 0]
[ -2, -6, -10, -64, -10, -6, -4, -6, -10, -64, 0, 64, 10, 8,  9, 10,  8,  4, 2]
[ -2, -4,  -8, -10,  -8, -4, -3, -4,  -8,  -8, 0,  8,  9, 8, 10, 64, 10,  6, 2]
[  0, -2,  -4,  -6,  -4, -2,  0, -2,  -4,  -5, 0,  5,  5, 5,  8, 10,  8,  4, 2]
[  0,  0,  -2,  -2,  -2,  0,  0,  0,  -2,  -2, 0,  2,  2, 2,  4,  5,  4,  2, 0]
```





```
[  0,  0,  -2,  -2,  -2,  0,  0,  0,   0,   0, 0,  0,  0, 0,  2,  2,  2, 0, 0]
[  0, -2,  -4,  -6,  -4, -2,  0,  0,   0,   0, 0,  0,  0, 2,  4,  6,  4, 2, 0]
[ -2, -4,  -8, -10,  -8, -4, -2,  0,   0,   0, 0,  0,  2, 4,  8, 10,  8, 4, 2]
[ -2, -6, -10, -64, -10, -6, -2, -1,   0,   0, 0,  1,  2, 6, 10, 64, 10, 6, 2]
[ -2, -4,  -8, -10,  -8, -4, -2,  0,   0,   0, 0,  0,  2, 4,  8, 10,  8, 4, 2]
[  0, -2,  -4,  -6,  -4, -2,  0,  0,   0,   0, 0,  0,  0, 2,  4,  6,  4, 2, 0]
[  0,  0,  -2,  -2,  -2,  0,  0,  0,   0,   0, 0,  0,  0, 0,  2,  3,  2, 0, 0]
[  0,  0,   0,  -1,   0,  0,  0,  0,   0,   0, 0,  0,  0, 0,  2,  3,  2, 0, 0]
[  0,  0,   0,   0,   0,  0,  0,  0,   0,   0, 0,  0,  0, 2,  4,  6,  4, 2, 0]
[  0,  0,   0,   0,   0,  0,  0,  0,   0,   0, 0,  0,  2, 4,  8, 10,  8, 4, 2]
[  0,  0,   0,   0,   0,  0,  0,  0,   0,   0, 0,  1,  2, 6, 10, 64, 10, 6, 2]
[  0,  0,   0,  -1,   0,  0,  0,  0,   0,  -1, 0,  1,  2, 4,  8, 10,  8, 4, 2]
[  0,  0,  -2,  -2,  -2,  0,  0,  0,  -2,  -2, 0,  2,  2, 2,  4,  6,  4, 2, 0]
[  0, -2,  -4,  -6,  -4, -2,  0, -2,  -4,  -5, 0,  5,  4, 2,  3,  4,  3, 0, 0]
[ -2, -4,  -8, -10,  -8, -4, -3, -4,  -8,  -8, 0,  8,  8, 5,  5,  6,  4, 2, 0]
[ -2, -6, -10, -64, -10, -6, -4, -6, -10, -64, 0, 64, 10, 8,  9, 10,  8, 4, 2]
[ -2, -4,  -8, -10,  -8, -4, -3, -4,  -8,  -8, 0,  8,  9, 8, 10, 64, 10, 6, 2]
[  0, -2,  -4,  -6,  -4, -2,  0, -2,  -4,  -5, 0,  5,  5, 5,  8, 10,  8, 4, 2]
[  0,  0,  -2,  -2,  -2,  0,  0,  0,  -2,  -2, 0,  2,  2, 2,  4,  5,  4, 2, 0]
```



$\hat{s}_{zobrist}$ = 0.7503172108131795 (with α = 0.013)

$\hat{s}_{ryder}$ = 0.8825766897875749 (with α = 0.017)

Example

Today's content

- Introduction to Go board positions
- Exact matching
  - Zobrist hashing
- Influence models
- Approximate matching *(Our contribution)*
  - Similitude *a-posteriori*
  - Similitude *a-priori*
- **Applicability in subproblems**
  - Opening book construction
  - Message traffic reduction in massive parallelization
  - Professional game clustering
- Experiments *(Work In Progress)*
- Summary

Applicability – Opening book construction

Problem 1

The programs switch out of the opening book to normal search as soon as it reach an "out of book" position, in case of Computer Go full board opening books *(fuseki books)* this happens in the 10th move (Hersey A., Sylvester N. and Drake P., 2010).

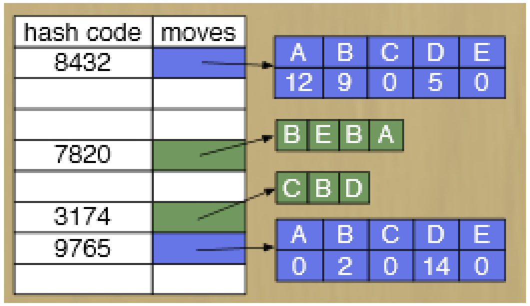
Problem 2

With the use of local opening books *(joseki books)*, the program stays more moves "in book" (up to the 26th move), but it doesn't improves the game strength (may even worsen the result), since the choice of the local pattern is highly context dependent (Mullins J. and Drake P. , 2010).

Wining percentage of Orego (raw) vs GNUgo	35.9%
Wining percentage of Orego (fuseki) vs GNUgo	38.0%
Wining percentage of Orego (fuseki + joseki) vs GNUgo	28.4%

Applicability – Opening book construction

With exact matching



Zobrist hash → Moves → Frequency

With approximate matching


```
{
    "011001100000101100" => {
        [ [ 0, 0, -2, -2, -2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 0, 0 ], [ 0, -2, -4, -6, -4, -2, 0, 0, 0, 0, 0, 0, 0
, 2, 4, 6, 4, 2, 0 ], [ -2, -4, -8, -10, -8, -4, -2, 0, 0, 0, 0, 0, 2, 4, 8, 10, 8, 4, 2 ], [ -2, -6, -10, -64, -10, -
6, -2, -1, 0, 0, 0, 1, 2, 6, 10, 64, 10, 6, 2 ], [ -2, -4, -8, -10, -8, -4, -2, 0, 0, 0, 0, 0, 2, 4, 8, 10, 8, 4, 2 ],
 [ 0, -2, -4, -6, -4, -2, 0, 0, 0, 0, 0, 0, 0, 2, 4, 6, 4, 2, 0 ], [ 0, 0, -2, -2, -2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2
, 3, 0, 0 ], [ 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 2, 2, 0 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 2, 4, 6, 4, 2 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 4, 8, 10, 8, 4 ], [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 2, 6, 10, 64, 10, 5 ], [ 0, 0, 0, -1, 0, 0, 0, 0, 0, -1, 0, 1, 0, 2, 4, 8, 10, 8, 4 ], [ 0, 0, -2, -2, -2, 0, 0, 0,
 -2, -2, 0, 2, 2, 0, 2, 5, 6, 4, 2 ], [ 0, -2, -4, -6, -4, -2, 0, -2, -4, -5, 0, 5, 4, 2, 2, 4, 4, 2, 0 ], [ -2, -4, -
8, -10, -8, -4, -3, -4, -8, -8, 0, 8, 8, 5, 5, 6, 5, 2, 0 ], [ -2, -6, -10, -64, -10, -6, -4, -6, -10, -64, 0, 64, 10,
 8, 9, 10, 8, 4, 2 ], [ -2, -4, -8, -10, -8, -4, -3, -4, -8, -8, 0, 8, 9, 8, 10, 64, 10, 6, 2 ], [ 0, -2, -4, -6, -4,
-2, 0, -2, -4, -5, 0, 5, 5, 5, 8, 10, 8, 4, 2 ], [ 0, 0, -2, -2, -2, 0, 0, 0, -2, -2, 0, 2, 2, 2, 4, 5, 4, 2, 0 ] ] =>
 {
            "O17" => [
                [0] 1,
                [1] 2
            ],
             "D6" => [
                [0] 0,
                [1] 1
            ]
        }
    }
}
```


Local-sensitive hash → Influence map → Moves → Frequency

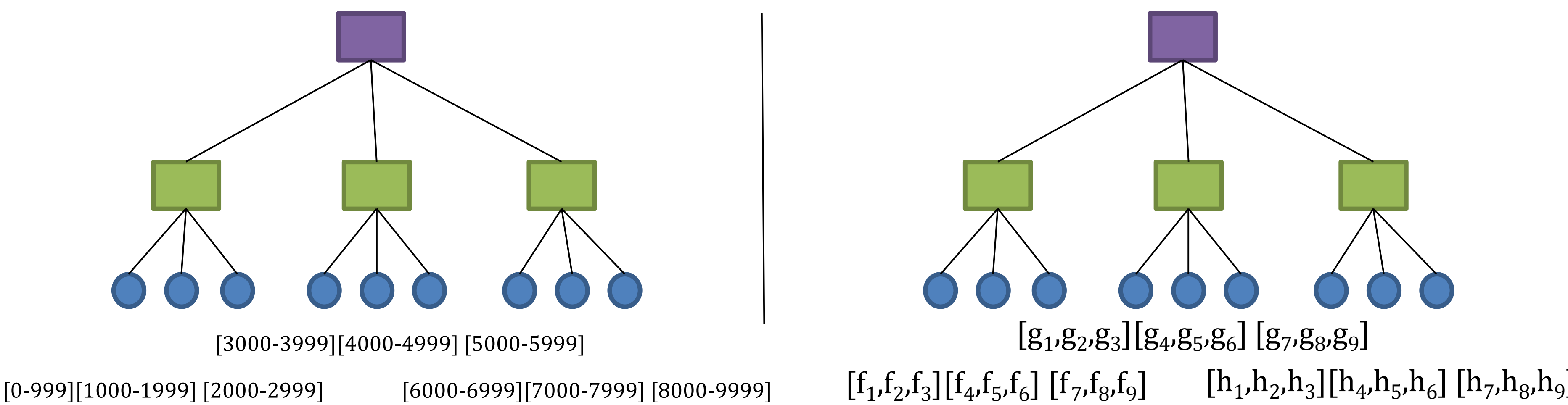
Example

Applicability – Message traffic reduction in massive parallelization

Problem

After achieving a huge break-through in communication overhead reduction by using Depth-First Upper Confidence Trees with Transposition-table Driven Scheduling (Yoshizoe K., Kishimoto A., Kaneko T., Yoshimoto H. and Ishikawa Y., 2011), the biggest remaining bottleneck in the communication is that the nodes are assigned uniformly distributed rather than matching their similitude to the network topology.

Uniformly distributed server switch **Distributed by similitude**

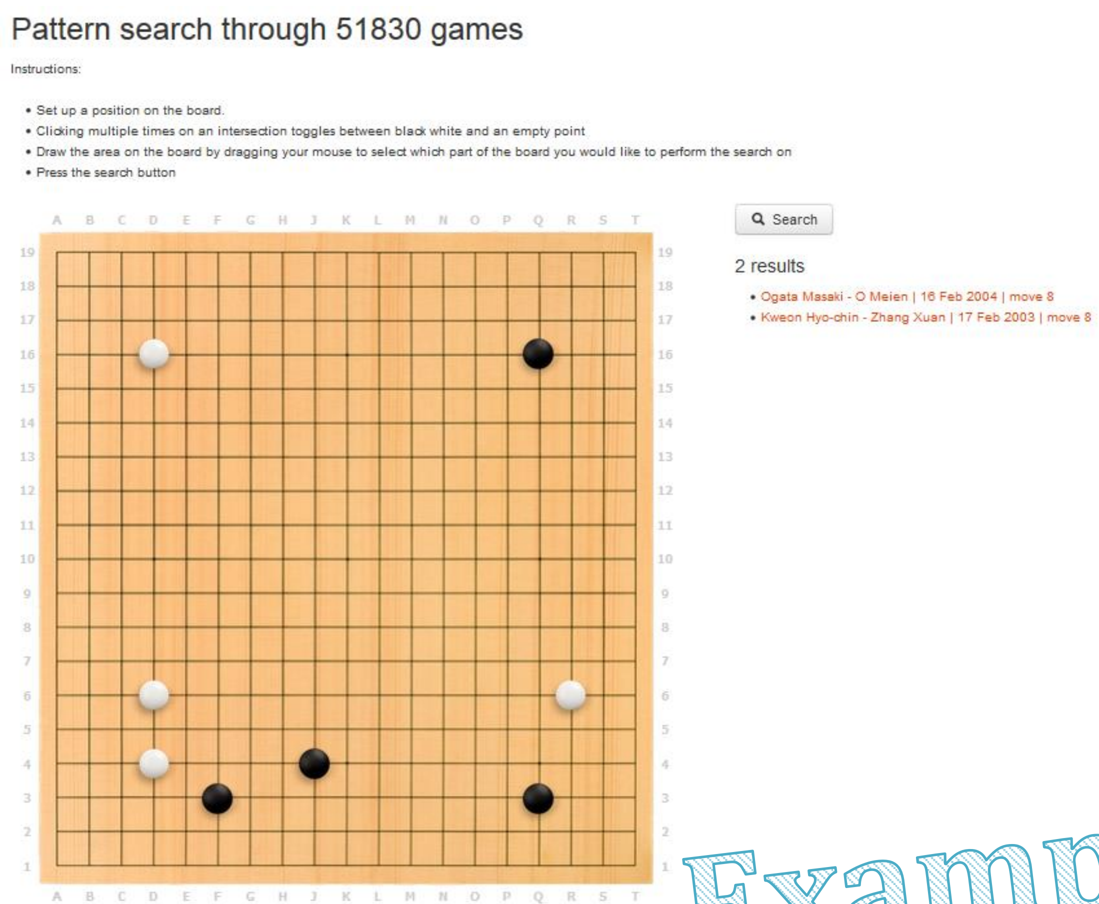


Applicability – Professional game clustering

Problem

Even in tools for aid the learning of human players, the exact matching is often a problem.

Sometimes there are lots of information hidden in nearby positions.



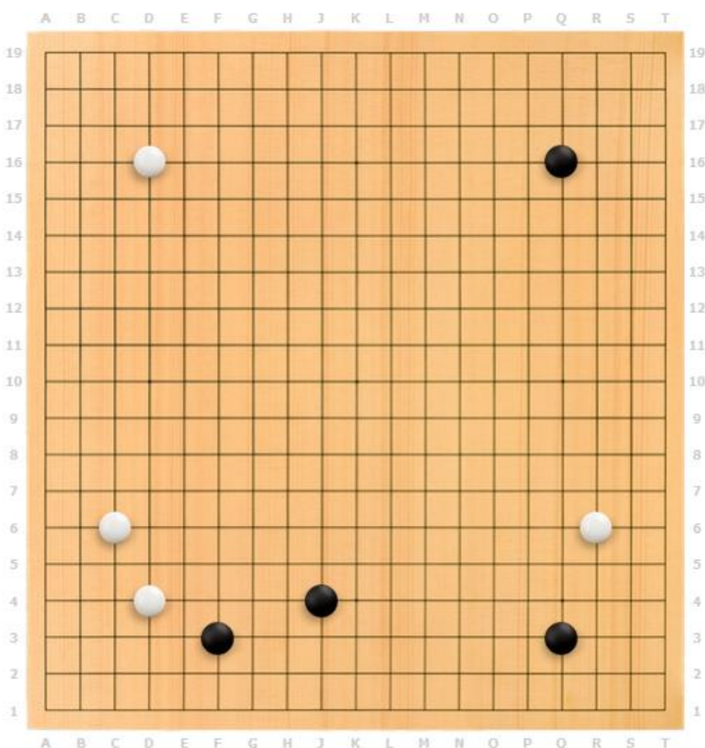
Example

Applicability – Professional game clustering

Pattern search through 51830 games

Instructions:

- Set up a position on the board.
- Clicking multiple times on an intersection toggles between black white and an empty point
- Draw the area on the board by dragging your mouse to select which part of the board you would like to perform the search on
- Press the search button



Search

93 results

- Ye Hongyuan - Hei Jiajia | 08 Nov 2012 | move 8
- Xie Yimin - Hei Jiajia | 30 Oct 2012 | move 8
- Ko Iso - Kim Jiseok | 04 Oct 2010 | move 8
- Okada Shinichiro - Kudo Norio | 17 Sep 2009 | move 8
- O Rissei - Kanazawa Hideo | 18 Jun 2009 | move 8
- Kim Dongyeop - Yun Junsang | 18 Oct 2008 | move 8
- Zhou Kui - Piao Wenyao | 29 Mar 2008 | move 8
- On Sojin - Lee Younggu | 11 Feb 2008 | move 8
- Michael Redmond - Sakai Hideyuki | 25 Jan 2007 | move 8
- Tan Xiao - Ma Xiaochun | 02 Dec 2006 | move 8
- On Sojin - Kim Kiyoung | 24 Jan 2006 | move 8
- Zhou Kui - Luo Xihe | 24 Dec 2005 | move 8
- Lee Sedol - Luo Xihe | 15 Nov 2005 | move 8
- Zhou Pingqiang - Chen Seuyen | 09 Aug 2005 | move 8
- Pae Yun-chin - Pak Chi-eun | 14 Jun 2005 | move 8
- Hong Minpyo - Cho Hunhyun | 16 Feb 2005 | move 8
- Zhang Xuan - Park Shiun | 19 Jan 2005 | move 8
- Choi Cheolhan - Lee Yeongkyu | 19 Jan 2005 | move 8
- Kim Eun Seon - Ye Gui | 20 Dec 2004 | move 8
- Ogawa Tomoko & Cho Chikun - Okada Yumiko & Takemiya Masaki | 04 Dec 2004 | move 8
- Kim Su-chang - Seo Neung-uk | 02 Dec 2004 | move 8
- Park Seungcheol - Qiu Jun | 02 Dec 2004 | move 8
- Zheng Yan - Zhang Xuan | 18 Nov 2004 | move 8
- Rui Naiwei - Kim Hyeoimin | 22 Oct 2004 | move 8
- Cho Hyeyeon - An Dalhun | 16 Aug 2004 | move 8
- Park Yeong Hun - Song Tae Kon | 11 Jul 2004 | move 8
- Zheng Ce - Zhang Xuebin | 24 Jun 2004 | move 8
- Rin Kaiho - Kato Masao | 13 May 2004 | move 8
- Kim Su Jang - Gwon Gap Ryong | 06 Apr 2004 | move 8

Example

Today's content

- Introduction to Go board positions
- Exact matching
  - Zobrist hashing
- Influence models
- Approximate matching *(Our contribution)*
  - Similitude *a-posteriori*
  - Similitude *a-priori*
- Applicability in subproblems
  - Opening book construction
  - Message traffic reduction in massive parallelization
  - Professional game clustering
- **Experiments** *(Work In Progress)*
- Summary

Experiments *(Work In Progress)*

Experiment A

Investigate the relation between the two kinds of similitudes. Check how well the similitude a-priori estimates the similitude a-posteriori, grouped by high similitude, medium similitude and low similitude.

Experiment B

Evaluate the Approximate Opening Book against the traditional Fuseki + Joseki opening books.

Experiment C

Measure if assigning the simulations by similitude to each server in a distributed parallelization can effectively reduce the communication overhead.

Today's content

- Introduction to Go board positions
- Exact matching
  - Zobrist hashing
- Influence models
- Approximate matching *(Our contribution)*
  - Similitude *a-posteriori*
  - Similitude *a-priori*
- Applicability in subproblems
  - Opening book construction
  - Message traffic reduction in massive parallelization
  - Professional game clustering
- Experiments *(Work In Progress)*
- **Summary**

Summary

Due to the vast size of the search space in Computer Go, exact full board matching has some limitations specially when trying to retrieve information.

There are two approaches to face this:

- Relax the size of the matching (Local matching)
- Relax the criteria of the matching (Approximate matching)

And this newly proposed approach seems interesting for several subproblems of Computer Go.


Thank you!

Q&A