

Applying data mining to the study of joseki

Michiel Helvensteijn (mhelvens@liacs.nl)

September 13, 2007

Abstract

Go is a strategic two player boardgame of Chinese origin. In terms of game theory, it is a deterministic perfect information game. But despite of these factors it is terribly complex because of its high branching factor. Many studies have been done with regard to joseki, localized exchanges of stones that are considered fair for both players. I will attempt to write a program that tries to find and catalogue as many joseki as it can, as well as the global circumstances under which they are likely to be played, by analyzing a large number of professional go games.

1 Introduction

I will write a program that analyzes all games in a go database and find the joseki that it contains by searching in the corners during the opening game. This will result in a trie (see Section 4). The algorithm adds a summary of the global board influence to the trie as extra information.

Section 2 will explain roughly how the game go works using an example. Section 3 will explain what joseki are, how they relate to this project and how I plan to find them. Section 4 will explain the algorithm used in the program in more detail using an example game from the database. Section 5 further explains several parts of the algorithm in greater detail. Section 6 will discuss the results of the algorithm and Section 7 will explain several decisions I have made. The last part of the experiment, where I try to find out the global circumstances under which a joseki is played, is explained in Section 8, together with the results of that part. Section 9 contains conclusions and discusses further research.

This bachelor project was supervised by Walter Kosters (Universiteit Leiden). My thanks go to Jan Ramon (Katholieke Universiteit Leuven) for the original idea, his support and his access to the go4go game database. I also thank Hendrik Blockeel (Katholieke Universiteit Leuven) for his help.

2 Go

2.1 The game

Go is played on a board with an 18×18 grid of squares, making for 19×19 intersections.¹ Both players take turns in placing a stone on one of these 361

¹The game is also sometimes played on a 9×9 or 13×13 board. But never professionally. And in this project, only 19×19 boards are taken into account.

intersections. One plays with black, one plays with white. The ultimate goal is to secure as much territory on the board as possible.² This is generally accomplished by first going after the corners, then the edges and then the center of the board, because each is more easily secured than the following.

Go is often called a game of balance. One has to play ones own stones far enough apart to encompass as much territory as possible, yet close enough together to stay strong and avoid successful enemy invasion. For more general information about the game, visit Sensei's Library [1]. See Diagram 1 for an example of a position from an opening game.

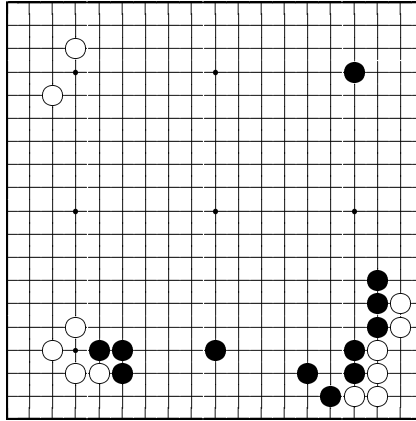


Diagram 1: A typical opening game position

2.2 The flow

In the opening game, the foundations of future territories are built. Both players try to establish themselves in the corners and edges of the board, to invade the opponents territory and to protect their own. To illustrate this process, let's see how the board position of Diagram 1 was formed.

Diagram 2 shows the first 11 moves of the game. As expected, the players first secure the four corners of the board. Exactly how to play in the opening game is largely a matter of personal taste. Black likes to open on hoshi, the star-point. White is more conservative and plays komoku, the (3,4) point³, for both corner stones.

Black is unwilling to wait for the action and attacks with 5. This launches a joseki, which ends when black plays 11.

White now has the bottom-left corner of the board, with good prospects on the left and upper sides. He has the next move. He could, for example,

²You can also gain points by capturing your opponents stones. However, in professional games, capturing is strictly a strategic maneuver and almost all points come from territory. In fact, many a professional game has ended without the capture of a single stone.

³When talking about a corner of a goban (go board), coordinates given refer to a point relative to that corner. Which coordinate refers to the x-axis and which to the y-axis should be clear from the context. In this case we are referring to white 2 and white 4 from Diagram 2.

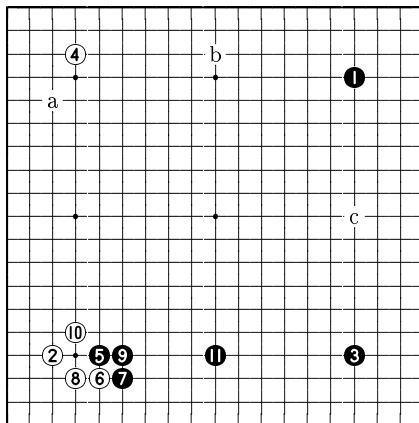


Diagram 2: The first corner joseki

play at 'a' next. This is called a shimari, or a corner enclosure. A shimari-move following white 4 like this is worth a lot at this stage of the game, since it virtually secures the corner for white and forms a good base from which to extend along the upper side, to 'b' for example.

Black now controls the bottom. Black 3, 7, 9 and 11 work together particularly well. Black's corner stones are placed on the hoshi-points. Any shimari move near them is not worth enough at this stage of the game. Black could next choose to play at 'b' or 'c' to expand his influence.

You could call this joseki a fair exchange, both locally and globally.

White now chooses to make the choice easier for black, by playing 12 in Diagram 3. Black has to respond to this move. This starts another well-known joseki that ends with black 23.

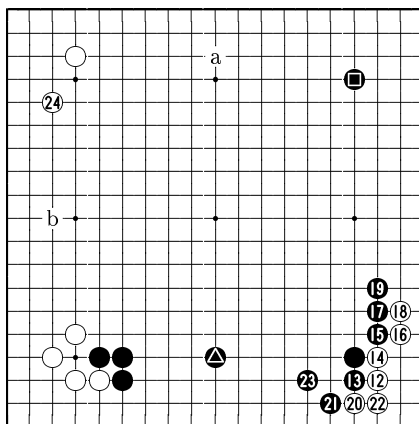




Diagram 3: Another corner joseki

While the joseki is well-known and often used, it was a mistake for white to start it now. The end result is this: Whites group in the bottom-right corner

is alive. But in exchange black has gained a thick wall that exerts a lot of influence towards the center of the board. In addition, the stone marked  works together even better with black 23 than it did with the original corner-stone. It will be pretty much impossible for white to invade there now. Invading on the right side will also be very difficult (though not impossible), because the  stone is now working together with black 19.

White still has to play 24.⁴ This leaves black with the initiative. He could choose to expand at ‘a’, invade at ‘b’ or even play at tengen (the center star-point) right now.

Locally, the black influence would not be worth more than the white territory in the corner. However, in this instance, the joseki was not a fair exchange. Black clearly has the advantage. You should only initiate this joseki if the resulting wall will not be this valuable to your opponent.

3 Joseki

You’ve seen two examples of joseki in Section 2.2. Now let’s take a look at what joseki really are and how they work. During this section, we will also come closer to which parts of the definition are important to the program. Davies [2] defines a joseki as follows:

There are patterns which arise in parts of the board again and again during the opening. They occur in especially large numbers in the corners, where the initial fighting of the game generally takes place. These corner formulas, which have been discovered by trial and error, and are still being added to and discarded, are called joseki.

Wikipedia [4], however, uses this definition:

A joseki is a sequence of moves in Go which results in a fair outcome for both black and white sides.

It goes on to explain that joseki most often occur in the opening game and in the corners, but that there are also joseki for the middle game. This definition is a bit imprecise, since the outcome for both players is only fair locally, as explained in Section 2.2.

The definitions are different (but not mutually exclusive). Other sources will give different definitions still. My own definition, the definition that will be used for this program, is the following:

A joseki is a localized sequence of play, in which both players play locally optimal moves. These sequences occur often in recorded games, especially in the corners of the board and the beginning of the game.

Notice how I have inserted two variations of the word “local” in the definition. It is an important property of a joseki that it is a local phenomenon. It takes place in a certain part of the board and moves that are played elsewhere (before, during or after the joseki) are not part of it. The players can sometimes break

⁴If white doesn’t, black will. A black move at 24 would be called kakari and is just as valuable as the shimari it prevents.

away from a joseki and get back to it later. This way, theoretically, multiple joseki can be in progress at the same time. The move that breaks away from a joseki is called a tenuki. The move that continues the joseki after a tenuki is sometimes called a follow-up play.

I do not think that a joseki results in a fair outcome for both players by definition. In fact, in Section 2.2, you have seen an example of a well-known joseki which does not result in a fair outcome. The most that can be said is that the outcome is locally fair. But even that is not always the case. Properly one should say that the outcome will be fair if the sequence is played under the right global conditions.

Of course, if a sequence did not result in a fair outcome under *some* global condition, it could never have been played often enough to be noticed and given the name joseki. This is important. Professional players do not blindly play joseki in their games. Not even under optimal global conditions. (At most, they use their knowledge of joseki as a heuristic.) They play the move that they think is best. And that is how joseki are found.

This is why I have not mentioned fairness in my definition. It is implied already. It is also irrelevant to my program. A computer can not calculate whether a sequence is really fair. Instead I choose to rely on human intuition, in that a sequence must be a joseki under my own definition if it is played often enough by professionals.

4 The algorithm

I use a database from the go4go website [3]. This database contains 13,325 go games played by professional go players. It is the job of the program to analyze the games from this database and eventually output the joseki (plural) that were found in the form of a tree in a `.sgf` file (see Section 5.3 for an explanation of `.sgf` files). The algorithm that it employs is depicted in Figures 1 and 2, which show phase 1 and 2 of the algorithm respectively. Each phase consists of two steps.

The first phase extracts all distinguishable sequences from the games in the database and stores them in a special tree. The second phase prunes that tree so that only the more interesting sequences remain, hoping it will result in a tree of joseki.

4.1 Phase 1

At the end of this phase, we hope to have a tree of all distinct move-sequences to be found in the games in which we can quickly find any sequence. Phase 1, step 1 is basically a nested loop that goes over all moves of all games in the database. Every move is compared to all currently stored sequences⁵ that belong to the current game. Its Manhattan distance⁶ to the stones of those sequences is used to determine whether it belongs to one of them. It is also

⁵A sequence is a potential joseki. I call it a sequence because it might eventually turn out to be nonsense and be pruned from the tree in phase 2. Only the sequences that survive that process are called joseki.

⁶The Manhattan distance between two points is the difference between their x-coordinates plus the difference between their y-coordinates. Also called the city block distance. The Manhattan distance between the points (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$

possible for one move to belong to more than one sequence. Two thresholds are defined. If a new stone is within distance b of an existing sequence, but not within distance a (which is always smaller than or equal to distance b), it will both belong to that sequence and start a new sequence of its own. Also, a move can simply be within proximity of more than one existing sequence. It has been decided that if a new move is within distance b of more than one sequence, it will not start a sequence of its own. It is not possible for so many joseki to exist so near each other and not have significant influence on each other.

Because the joseki we are looking for are played mostly in the corners and in the beginning of the game, the program will stop looking for stones after each corner contains at least 20 stones. This means that at the least, we will examine 80 stones. Anything more than that and we will have entered mid-game. The reason we don't just look at the first 80 stones instead is that sometimes a single corner can remain empty for a large portion of the game, which means we might miss some obvious joseki.

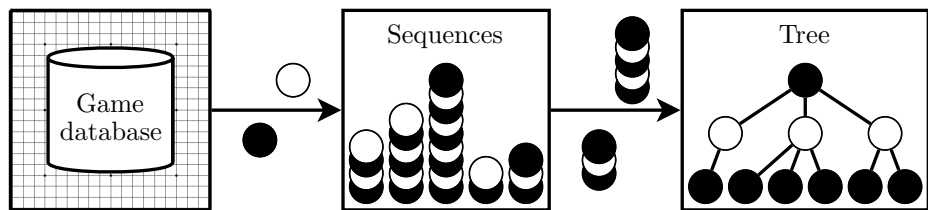


Figure 1: The algorithm, phase 1: creating the tree

Step 2 moves these sequences to the tree. The root of this tree represents the empty board. Its children are the first moves of the sequences, and so on. Each node in the tree represents a sequence prefix to that point and their children represent their continuation. However, the complete sequences are not really stored in their node. The tree is instead implemented as a trie. Each node contains a mapping of point \rightarrow node (where “point” is a point on the board or a tenuki) to find its children. This provides very fast lookup and insertion of sequences. The complexity of these operations is $O(n)$ where n is the length of the sequence. Each node also has a counter indicating how often a certain sequence has been played. An insertion increases the right counters in the tree and adds new nodes if necessary.

Step 2 can be performed after step 1 is done. However, this would be a waste of computer memory and effort, because it is much easier to perform both steps at the same time. After each go-game from the database, the found sequences are added to the tree immediately, which leaves the sequence-store empty for the next game.

4.2 Phase 2

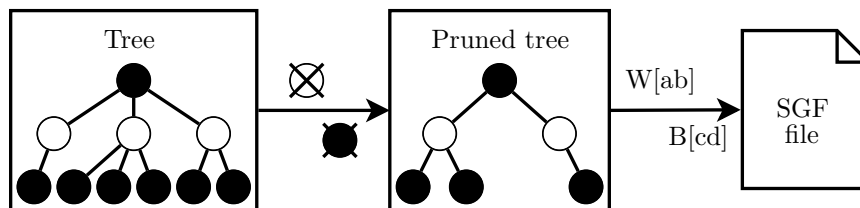


Figure 2: The algorithm, phase 2: pruning the tree

Phase 2 consists of pruning the tree that we built up in phase 1. This might be the most important part of the algorithm (well, almost) because it separates the joseki from the trash, so to speak.

Because of the nature of the trie, the counter of any node is equal to the sum of the counters of its children, and so higher than or equal to the counter of any child node. The basic pruning approach is to cut off any subtree that does not have a counter higher than or equal to a given threshold value. I have experimented with several threshold values and will show the results in Section 6. This value is about 1% of the amount of games in the database.

After step 1 of phase 2, we hope to have a tree full of joseki. Step 2 is to make this tree readable for humans. This is done in two ways. First of all, it is printed to a `.sgf` file (see Section 5.3), which can be opened by many free sgf-editors and viewers that allow you to see the tree and any board position stored in it. Also, a `.dot` file is created that can be converted to the diagrams from Section 6.

These two steps could also be performed simultaneously. But in this case it is simpler to create the files after the tree has been pruned. This does not have a significant performance-cost.

4.3 Example

To clarify the algorithm, why not see it in action? This section will follow the algorithm as it processes one of the games from the database.

The example game is a match between Xie He (6p) and Duan Rong (7p)⁷ from the first round of the 18th Chinese CCTV Cup. Xie He plays white and Duan Rong plays black. See Diagram 4.

Both players first occupy the four corner hoshi. As you can see next to the goban in Diagram 4, each of the first four moves starts a new sequence. For now we will call them sequence 1, 2, 3 and 4.

Black 5 starts the first joseki of the match. It ends with black 9. A move belongs to a sequence if it is within Manhattan distance x of it. For this example, $x = 5$. There is no “maybe”-distance in effect. (That is, it is equal to the “certain”-distance.) Black 5 is clearly only within proximity of white 2. And

⁷6p and 7p are the ranks of the players. Beginning amateurs have a rank ranging from 30 kyu to 1 kyu. Amateurs can then advance to 1 dan after official examination. They can rise as high as 7 dan. Professional players can reach the rank of 9 dan. However, a professional player of any rank is generally stronger than even 7 dan amateurs. Their rank is abbreviated ‘p’. 7p means “seventh dan professional”.

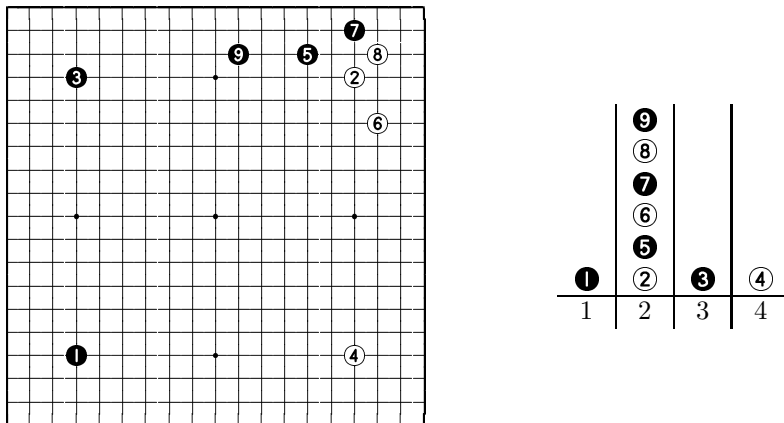


Diagram 4: Xie He vs. Duan Rong, part 1

so it is added to sequence 2. The same goes for white 6 to white 8. Black 9 also belongs to sequence 2, because it is within proximity of black 5, which was added to the sequence earlier.

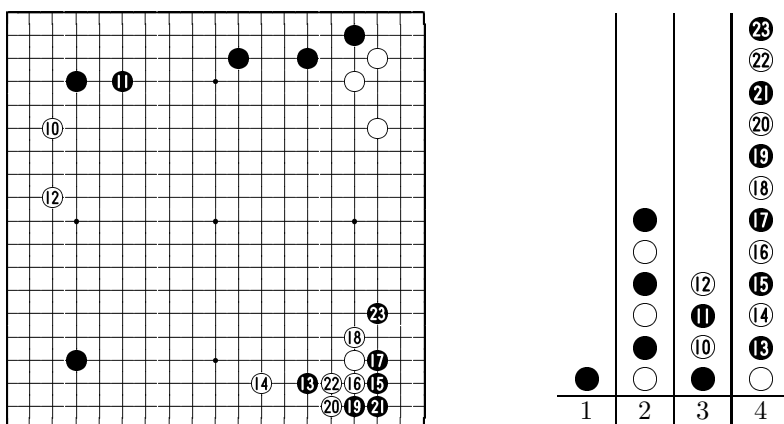


Diagram 5: Xie He vs. Duan Rong, part 2

White 10, black 11 and white 12 are added to sequence 3 (see Diagram 5). You will notice that black 11 is only just outside the reach of sequence 2. Black 13 starts a long joseki that ends with black 23. All of those moves are part of sequence 4. I cannot pretend I didn't choose this match because it has nice examples like this. But such long, isolated joseki are not as uncommon as you might imagine, as proven by my algorithm. That exact joseki occurs 566 times in the database.

White 24 and black 25 add another small joseki to the collection, in sequence 1 (Diagram ??). But something else has also happened here. Black 25 is within range of white 12, as well as black 1, so it is also part of sequence 3. As is white 26. After two non-joseki moves, black 29 does the same thing. It is part of both sequence 1 and 4. This is bound to happen as more and more stones are

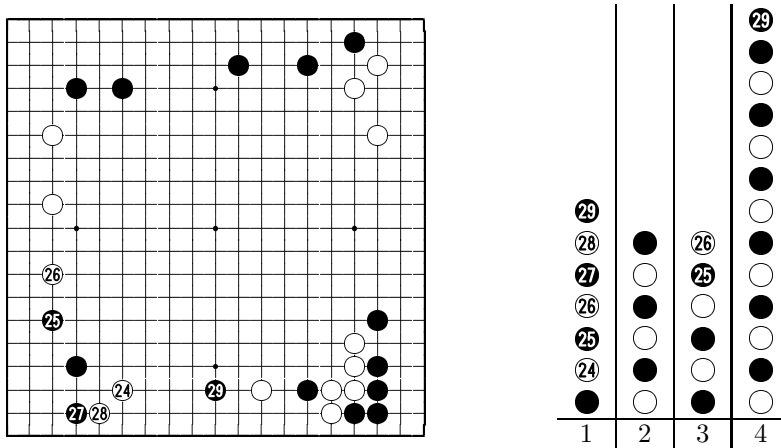


Diagram 6: Xie He vs. Duan Rong, part 3

	Joseki prefix	Transformation	Color-swap?
1	① ②④ ②⑤	H	No
2	② ⑤ ⑥ ⑦ ⑧ ⑨	V	Yes
3	③ ⑩ ⑪	I	No
4	④ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ ㉑ ㉒ ㉓	H × V	Yes

Table 1: Xie He vs. Duan Rong, the four joseki

played on the board. But the assumption here is that either joseki are played in isolation before the sequences start interfering with each other or that only one of the sequences really “owns” the new move. It is not unthinkable that a stone with a distance of 5 doesn’t really belong to the sequence. For example, if black 25 were part of any joseki, it would be sequence 1. These things will be recognized in phase 2, when follow-up stones that don’t belong to the joseki are pruned out of the tree.

Now to continue with the algorithm. We are done with this game, but the program will not know it at this point and will continue to build the sequences until at least 20 stones have been played in each corner.⁸

Prefixes of all four of the sequences played out so far will turn out to be important joseki in the final tree. Table 1 shows these joseki.

The first column gives the sequence reference number. The “Joseki prefix” column gives the suffix of the sequence that forms the joseki. In other words, the stones that would be pruned off in phase 2 are not shown here. The “Transformation” column shows the matrix manipulation that should be applied to each stone of the sequence, so it will yield the sequence’s normal form, which is then added to the tree. **H** means a reflection in the horizontal ($y = 10$) axis. **V** means a reflection in the vertical ($x = 10$) axis. **I** means the identity-matrix, or no transformation at all. \times is the matrix multiplication operator. So sequence 4 has to be reflected in the horizontal and vertical axes to get to its normal form.

⁸In fact, in this case, the whole game will be played out. Before the two left corners of the board contain at least 20 stones, white will win by resignation after the 112th move.

Sometimes a reflection in one of the diagonal axes is also needed. The “Color-swap?” column indicates if the colors of the stones need to be swapped from black to white and vice versa. This is, of course, the case for all sequences where white moves first. Because by convention, black has the first move. And we will adopt this convention to get our normal form.

This very procedure is performed for all games in the database, resulting in a big tree with a lot of junk inside. After phase 2, however, it will be a relatively small tree with assorted recognizable joseki.

4.4 Complexity

The beauty of this algorithm is that it is linear from start to finish. Because of several choices I have made, only one pass through the data is necessary to collect the data. And the several seconds it takes to run the program are only due to the size of the database, which contains more than 13,000 games of which we record 80 to 150 moves each.

5 Subprograms and techniques

The algorithm uses several techniques that I will further clarify in the following subsections. They will also explain some techniques I didn’t use, but which are nonetheless relevant. In those cases, I will also explain why I did not use them.

5.1 Symmetry

A stone on the board has a symmetry group with 8 elements which can be reached by reflecting the stone in one or two of the four board axes. Reflecting twice can also be seen as a rotation around the tengen. Another dimension of symmetry with regard to joseki is color, in that two spacially identical stones are still equivalent, even if one is black and the other is white. Diagram 7 depicts this symmetry, and also shows 16 equivalent moves. This symmetry can be extended to a sequence of moves. When two joseki are equivalent in this way, we want the program to recognize this. The Transformer class was written for this purpose. It takes a stream of board-coordinates and transforms each of them using reflections and color-swaps so that:

1. the first stone in the stream is placed in the area delimited by ‘o’s in Diagram 7 (including the ‘o’s themselves),
2. when there is still choice in the transformation (if any stone that has been evaluated already was placed on one of the four symmetrical axes) the next stone will be placed on the half of the board that contains that area (that would be the left half, the top-right half or both),
3. each stone is still in the same position relative to every other stone,
4. the first move of each sequence is played by black and
5. if the first move had to swap its color for this, all other moves have to swap color as well.

So each first move in Diagram 7 would transform to the black move marked \blacktriangle .

In reality, another dimension could be taken into consideration. Namely, mathematical translation of the joseki. Joseki that occur along the top edge of the board might be equally valid two places to the right or to the left. Joseki in the center of the board might even be shifted in both spacial dimensions and still be equivalent to the original. My program does not take this into account, however. See Section 7.2 for an explanation.

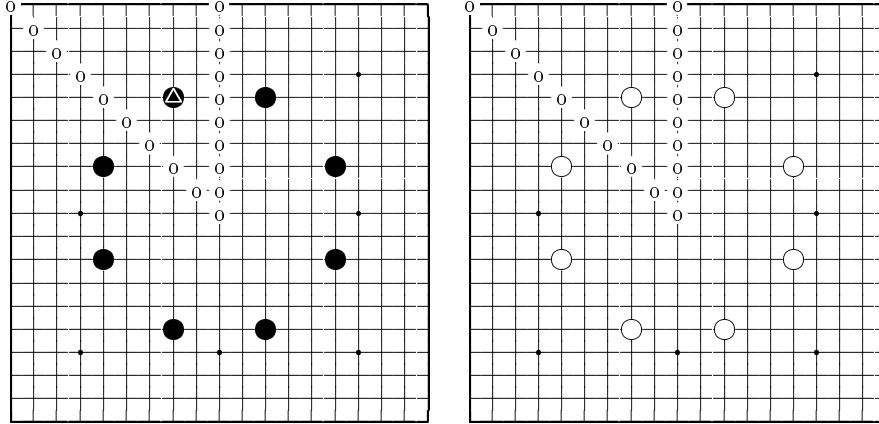


Diagram 7: 16 equivalent first moves

This transformation is applied to a sequence before it is added to the tree in phase 1 (Figure 1) of the algorithm.

5.2 Noise recognition

The definition of joseki could be stretched beyond corners and the opening-game. If you wanted to find all common sequences of play, you'd have to look at the entire board and the entire game. This means several things. First of all it means that an extra dimension of symmetry should be added, namely move translation. (See Section 7.2 for an explanation of why I have not implemented this.) Also, you'd have a much more difficult time finding equivalent sequences because of the many unrelated stones that may have been played in the area already. As an extreme example, see Diagram 8. In the left diagram, black has played a standard komoku followed by a shimari. In the right diagram, the stones are played in the same positions and in the same order. However, the two sequences are clearly not equivalent. The two white stones are in the way and the two black stones lose all their strength. But based only on their Manhattan distance they would be considered equivalent anyway.

To fix this, I came up with the concept of noise recognition. To implement it, you would have to keep a memory record of the entire game while you add stones to the sequences. Each time you want to add a stone, you have to first check the board to see if there are any unrelated stones in the way. This will be more expensive than a simple calculation of Manhattan distance (which does not take into account the intervening stones), but that's the price you pay.

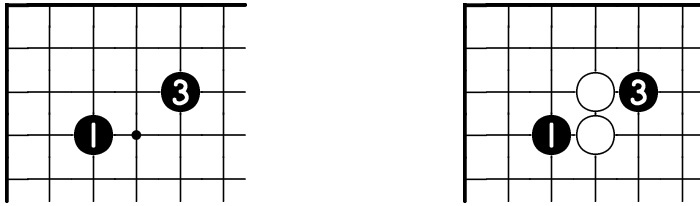


Diagram 8: Illustration of noise during a sequence (white 2 played elsewhere)

5.3 Smart Game Format

The Smart Game Format (sgf) [5] is a standardized and universally recognized file format for storing several types of game records, especially for go games (which is why the abbreviation is sometimes misinterpreted as Smart Go Format). In addition to game records, the format can also contain tsumego or joseki in a tree-structure.⁹

This is the file format that all the games in the database come in. These files are linear and do not contain any variations. I also use the same file format for the output of my program. The tree of joseki, which does use the format's ability to store variations and comments.

6 Results

The database consists of 13,325 games exactly.

The following settings resulted in the tree of Figure 3.

- Pruning score: 150
- Sequence binding distance: 5
- Sequence 'maybe' distance: 5

This tree contains 81 leafs, meaning 81 complete joseki. However, I found that the algorithm does not only find some joseki, but also a lot of what might more properly be called fuseki structures (opening game sequences). Which is not surprising, since the program looks primarily in the corners of the board and the opening game, which are exactly the time and place fuseki structures are formed. It may have to do with the length of the sequence, but the set of joseki and the set of fuseki seem to overlap when you only consider the opening game.

The tree of Figure 3 shows us some well-known joseki and fuseki (which is easier to see if you open the .sgf file with an sgf viewer, see Section 5.3). You may notice a variation of the joseki shown in Diagram 2. The one in the example prevents black from effectively extending up. It is more often played if white has some influence on the left side already. The variation in the tree, where white 10 is instead played on the (3,6) point, is used to gain more territory on

⁹Tsumego are life-and-death problems. A very important aspect of the game, but relatively unrelated to joseki. Nevertheless, both tsumego and joseki can have several variations, so it takes the same sort of tree-format to store them.

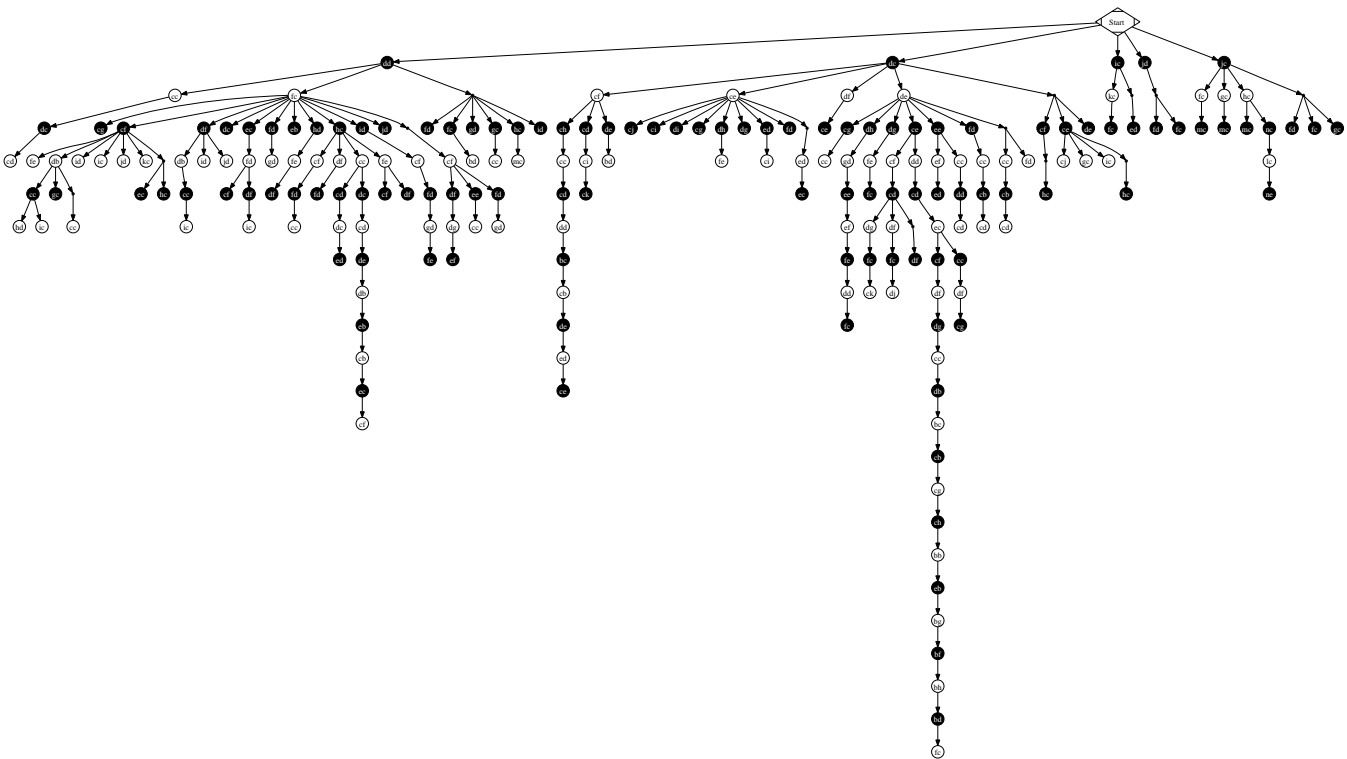


Figure 3: Joseki tree, distance=5, pruning-score=150

that side, and is apparently more common. Also, the two joseki shown in the Wikipedia article [4] are part of the tree.

Now lets tweak the variables a little. The following tree (Figure 4) has a distance of 6 instead of 5. The pruning score remains the same.

You will notice that the distance of 6 is rarely used. If it is, it is in the very short fuseki-like sequences, and the distance seems a little to great to be useful. Also, the longer distance has an adverse effect on several joseki we found already. Some of the longer ones have lost up to 50 appearances because of the interference of a stone 6 places away. Some others have dissapeared entirely because they no longer pass phase 2. This could be remedied by lowering the pruning score, but there seems to be no point. And we'd attract more useless sequences into the results. So the distance of 6 seems to be too high.

So why not try a distance of 4? It is true, there are some sequences that use the distance 5 that I could do without. But there are also one or two that I would really like to keep. On the other hand, all the sequences that do not use distance 5 would be strengthened as a result. It seems to be a border case. Take a look at Figure 5.

It does, in fact, look fine. But we are simply missing some common joseki that use distance 5. It does have some more junk, because distance 5 stones do not interfere with their score anymore. This could be fixed by using a higher pruning score.

So do we use distance 4 or 5? I don't think there is an answer. Maybe the whole distance mechanism is wrong, and improvements could be made in future work (see Section 9).

It is quite predictable how the pruning score works. Take any tree A made with this algorithm, any game database and pruning score a and any other tree B made with this algorithm, the same game database and pruning score b . Let $a < b$, then B is a subtree of A . The perfect pruning score is of course entirely dependent on the size of the database. But even if you take a percentage of the database size, it is still not easy to pick the right score. Perhaps a pruning function would be better than a pruning score. This is worth further study.

7 Assumptions and justifications

Unfortunately, my program does not yield the exact set of joseki. It finds some sequences that would not be considered joseki, and a lot of existing joseki are left out. My program will not be able to take the place of hundreds of years of human study. It is limited by the nature of the computer and the size of the game database. But it goes further. I have deliberately made some assumptions and shortcuts. And I will now discuss them.

7.1 Mistakes

Even if a sequence of moves occurs often, it is officially only called a joseki after enough study and discussion. This is mostly to make sure that none of the moves are mistakes, or that there are no obvious moves that are superior. However, since all games in the database were played by professional players, I'm assuming that if a certain sequence occurs often enough, it is not a mistake.

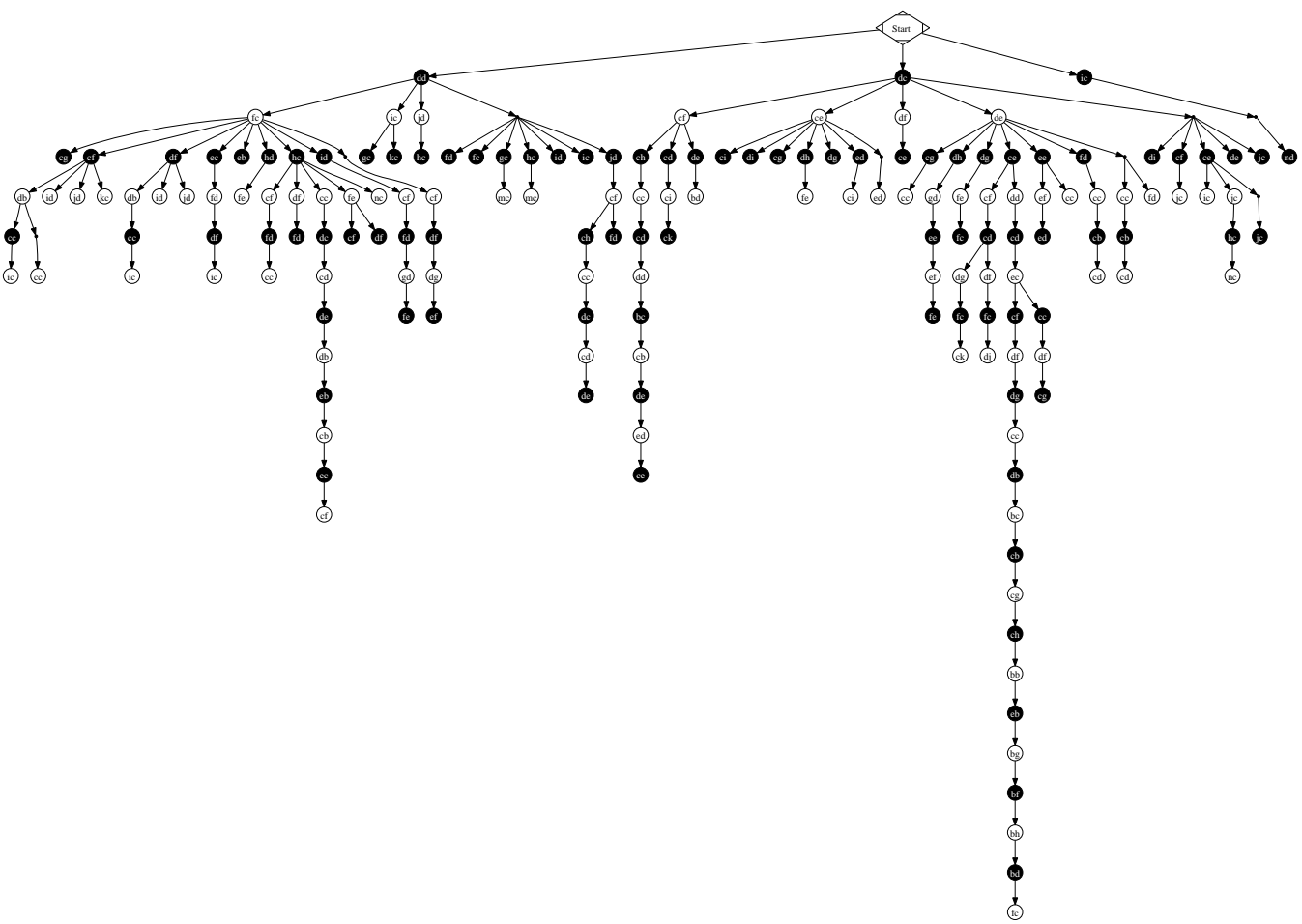


Figure 4: Joseki tree, distance=6, pruning-score=150

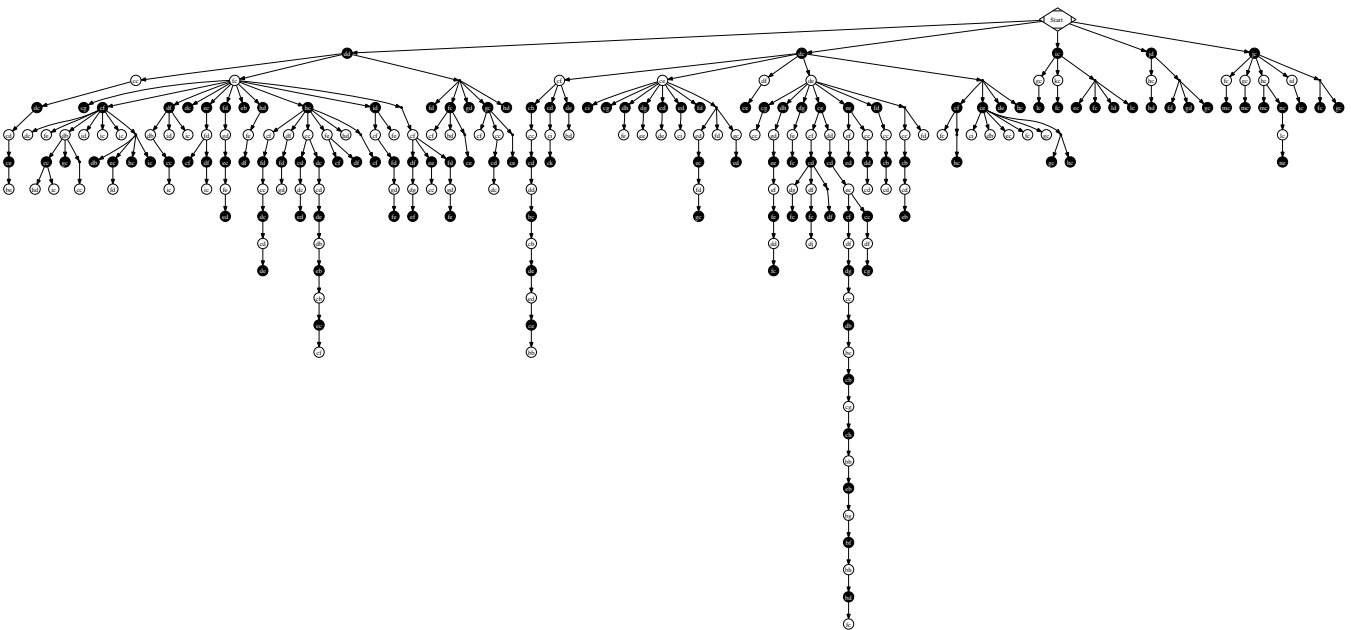


Figure 5: Joseki tree, distance=4, pruning-score=150

Even professionals make mistakes, of course. But these will be pruned out of the tree because the same mistake will not be committed often enough.

7.2 Sequence translation

Two joseki are seen as equivalent if they can be made equal through reflection and color-switching, as discussed in Section 5.1. In theory, joseki could sometimes also remain equivalent after a translation along the x or y axis. I have deliberately not included translation in the transformation mechanism for the following reasons:

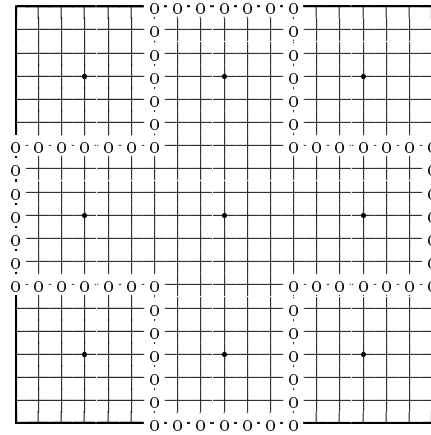


Diagram 9: Within the delimited area (the cross), joseki are free from at least one board edge.

For two joseki to be equal across the translation along one or both dimensions, they must have no stones outside the area roughly delimited in Diagram 9. If a joseki does have a stone in one of the corners, both edges of the corner would have much too great an influence on the sequence. Neither player could ignore an edge, or especially a corner, because the rules are different there. However, most joseki by far occur exclusively in the corners. For this reason, I decided it would be safe to ignore translation. This would be different if we were playing on a goban of infinite size (which wouldn't have any edges), but we are not.

Translation would also make the code much more complex, because it would sometimes require the algorithm to go back and change previously translated stones back, because suddenly the condition of the validity of the translation is no longer met. Without translation, this algorithm is linear. Color-switching is decided after the first stone. And reflection-axes are only added, never removed, when going through the moves of a sequence.

Not only that, but translation can simply be wrong sometimes. The borders in Diagram 9 are not definite. There are small edge-joseki that can be shifted beyond it towards a corner and still work the same (like the move called “monkey jump”). There are also bigger joseki in which the influence of the edge can not be ignored, even from the center of the board. Reflection and color-switching are what you might call lossless transitions. Translation is not. It would be a source of mistakes and a pain to calibrate correctly.

7.3 Joseki length

I assume that no joseki is shorter than 3 moves and no joseki is longer than 30 moves. Without a minimum length, and with the simplest pruning approach, each single first move would register as a joseki, and we are not interested in seeing those. I have also found through experimentation that the program will never find joseki longer than 30 moves, so by capping them off at that length in phase 1, we save valuable memory and time. Indeed, by setting this maximum length, the program run-time has dropped from 5 minutes to 5 seconds on the average computer. This is probably because the memory-usage of the first version was so large, the program had to address harddisk swap-space.

8 Global influence

After the first part of the experiment, I still want to know more about these joseki. We know now which joseki (or fuseki) are played, but we still don't know *when* they are played. As one could see in Section 2.2 and as explained in Section 3, this is important information.

There are several factors that could have an influence on the “fairness” or “validity” of a joseki.

1. Global influence around the board. Because some joseki are only profitable if you have nearby support stones, for example. The opposite is also possible. Some joseki steal away a corner that would otherwise have been territory for your opponent (see Diagram 3).
2. Ladder-breakers. Some joseki can start a ladder, a diagonal ongoing succession of stones where one player repeatedly puts his opponents group in atari¹⁰ and the opponent repeatedly escapes gaining only one liberty each time until said group is captured at the board edge or at a stone belonging to the chasing player (see Diagram 10). A ladder can be broken if the chased player has a stone in its path. This stone is called a ladder-breaker. Some joseki depend on the presence or absence of such a ladder-breaker.
3. Old stones nearby. Some joseki are not played in the beginning of the game. In such cases, old stones may be nearby to help or hinder a joseki. This is the case, for example, with the “monkey jump” move (see Diagram 11).
4. The overall score in points. If a player is behind, he might try riskier moves, resulting in specific joseki. If a player is ahead in points, he will probably try safer joseki: play a balanced game with which he does not gain any more points.

Ladder-breakers are very difficult to find, especially with an algorithm that works the way mine does. A ladder may or may not be formed because of a

¹⁰A group of stones that has only one liberty left is said to be in atari. A liberty is what you might call breathing room for a group: empty spaces directly touching it. The opponent would have to fill each liberty to capture the group. A group is said to be alive if it can get two liberties that cannot be filled (eyes). It is said to be dead if it can never create such liberties.

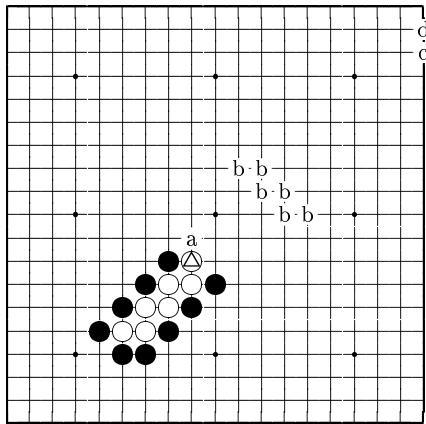


Diagram 10: A simple example of the ladder. White just played the triangle-marked stone to escape. Black's next move would be 'a'. The locations marked 'b' are possible locations for ladder-breakers. Without a ladder-breaker, if this would go on, the white stones would eventually be captured after white 'c' and black 'd'. Of course, no moderately experienced player would let it get go that far.

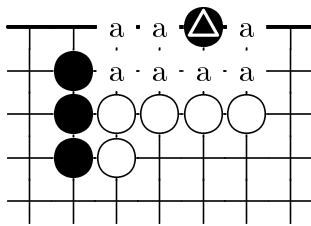


Diagram 11: The black stone marked with a triangle is called the monkey jump. It is a way to sneak as far as possible into enemy territory without being captured, and take away some points or even connect to another group. Had any of the spaces marked 'a' contained a white stone, the move would not have worked. This is, coincidentally, also a nice example of an edge-joseki, as opposed to one played in a corner.

certain joseki. The algorithm could only know about a ladder if it actually forms. And even then, it would have to go back to earlier moves of the sequence to pass on that information. Because the algorithm may not have known at that earlier time that a ladder could be formed, but the players certainly did, and kept it in mind while playing the joseki. But then, at which move in the sequence did the ladder become a factor? These are things that could be figured out, but not in this project.

For joseki like the monkey jump to be found, the algorithm would have to search from mid- to endgame. It would also have to translate the sequence along the edge to a specific normal form (see Section 7.2). My algorithm does not find this sort of joseki.

It takes a somewhat experienced player to know the score at any time in the game. It is a matter of recognizing territory, and computers still have a lot of trouble with this. It is true that the score may still be estimated somehow. And then it could be taken into consideration when recording a joseki. However, my algorithm does not.

The algorithm does check the global influence around a joseki. This is relatively easy and was a fun last thing to try. It doesn't require any existing code to be fundamentally altered and the output is only being extended, not changed. The algorithm as explained in Section 4 remains mostly unaltered. The most important changes to the code are:

- During the analysis of each game, the current state of the board is kept in memory. A black stone has the value -1 . A white stone has the value 1 . An empty space has the value 0 .
- At each move, the state of the board is altered accordingly.
- When adding a move to a sequence, this sequence first updates its bounding box. This box consists of:
 - The x -coordinate of the leftmost stone in the sequence (\mathbf{xMin}).
 - The x -coordinate of the rightmost stone in the sequence (\mathbf{xMax}).
 - The y -coordinate of the uppermost stone in the sequence (\mathbf{yMin}).
 - The y -coordinate of the lowermost stone in the sequence (\mathbf{yMax}).

Then the upwards, rightwards, downwards and leftwards influence is calculated and stored for that move, counted from the corresponding boundaries of the bounding box. The width of the “searching beam” is the width of the box plus one stone to each side (see Diagram 12).

- When a sequence is added to the tree, the transformer also transforms the directions of these influences, and also performs color-swaps on them. This way, the influence transforms right along with the sequence, to the upper-left corner.

Diagram 12 shows how the influence is calculated, using the example from Section 2.2. The stone marked \blacktriangle has just been played and the influence of the bottom-right sequence of the board has to be calculated for that move. From the left edge of its bounding-box, a search to the left is done for each row indicated by the stream of ‘o’s. Each time a stone is found, that row stops searching and its

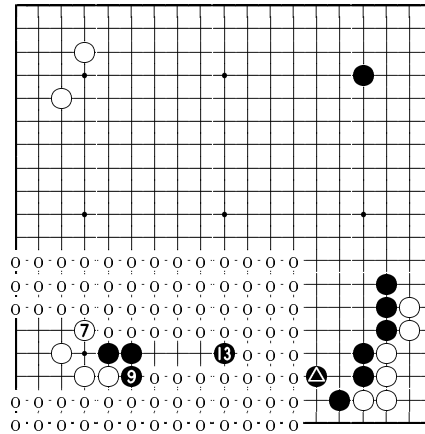


Diagram 12: Calculation of left-side influence

score (17 minus the distance from the bounding-box)¹¹ is added to the influence score (negative for black, positive for white). If that score lies between -5 and 5 , the influence is considered neutral. Below that, it is black. Above that, it is white. The influence score for this search turns out to be $-13 - 9 + 7 = -15$. This means black has the most influence in that direction, which is quite clearly the case. The influence scores up, right and down are -8 , 0 and 0 respectively.

When a sequence is added to the tree, its influence is transformed along with the sequence. And every node of the tree gets four groups of counters. One for each direction. Each group has one counter for the number of instances where black has influence, one for white influence and one for neutrality. Logically, the sum of these three counters equals the score of the node. After pruning the tree, a joseki (the last stone of the joseki, really) is said to be played under *color* influence in *direction*, if that *color* has influence in that *direction* in more than 30% of the cases and the other color does not. Another option is that any influence is needed for a joseki. This is the case if both colors have influence in *direction* in more than 30% of the cases. The last option is that no influence is needed in *direction*.

The value of these 12 counters is eventually recorded in the `.sgf` file as a comment, together with the “verdict”, which is simply a word indicating one of the four possibilities. A new kind of dot-file for a diagram is also created, which you can see in Figure 6. To save space, it only shows the influence to the “right” and “down”, because the “up” and “left” directions are almost always completely neutral. This is because we find mostly corner-joseki, and (after the transformation) the search quickly reaches the edge in those directions, returning a score of 0, as you can see in example from Diagram 12.

The tree diagram uses the abbreviations ‘B’ for black influence, ‘W’ for white influence, ‘A’ for any influence and an empty space for no influence. It

¹¹If a stone is closer to the sequence, it has more influence on it. If a stone has more influence on a sequence, it is more likely that a player will deviate from the joseki that would have been played without this influence. Even if a stone is on the other side of the board, though, it can have influence. The number 17 was chosen experimentally. It seemed the furthest distance from which a stone could still have *any* influence.

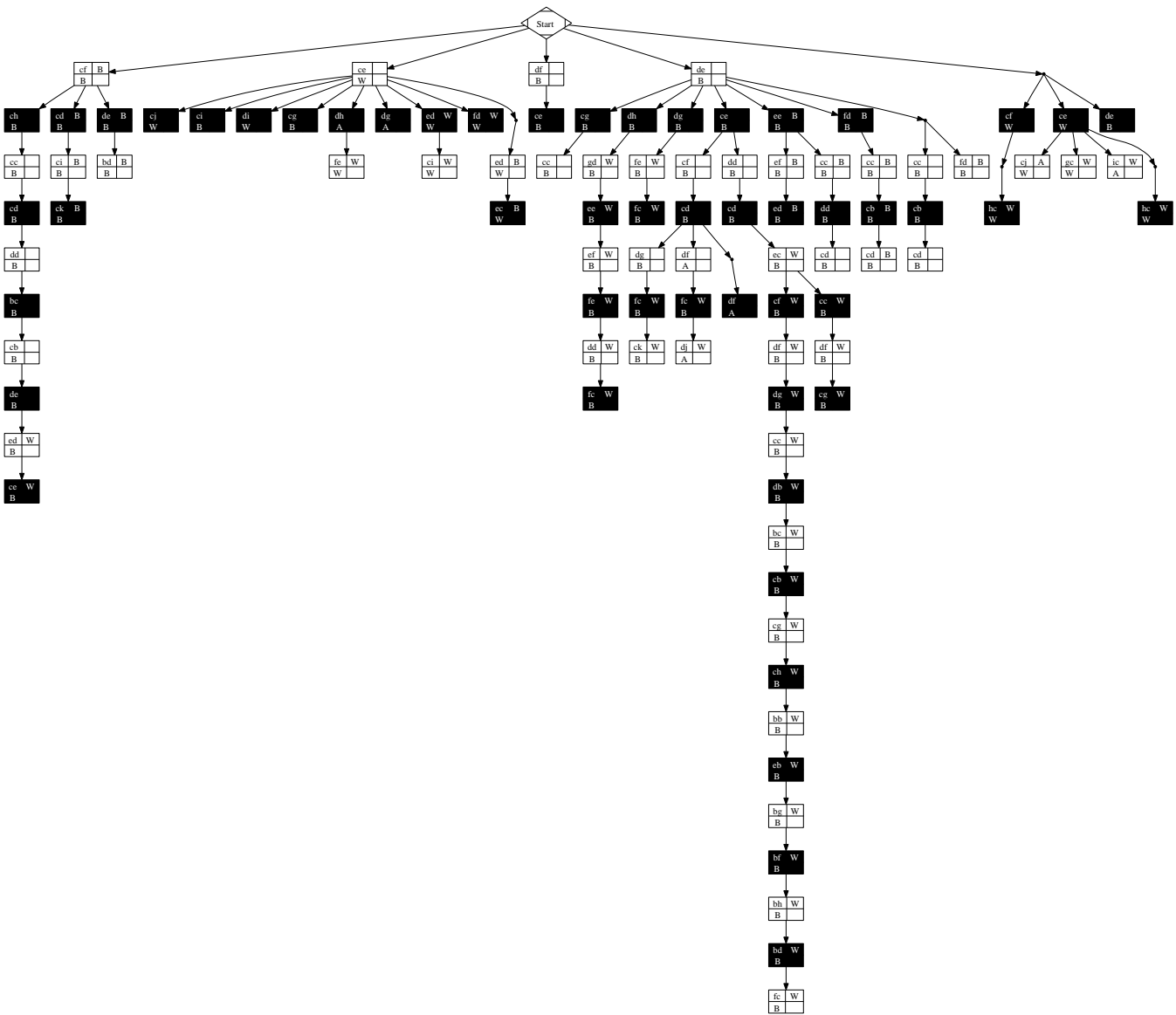


Figure 6: Joseki tree, distance=5, pruning-score=150, with influence information, starting after black moved at the (4,3) point (komoku)

also shows only a part of the tree, because otherwise it would become even more illegible. The part of the tree shown is the part after black moved at the (4,3) point (komoku). This is the most interesting point, because most joseki start with it, and so do most games.

Manual inspection of the tree indicates that for most joseki, influence is most definitely a factor, as expected. However, the line between “any influence” and “no influence” is somewhat fuzzier, except for the consistent and predictable “no influence” for the left and upper sides.

9 Conclusions and future work

The technique of finding joseki in a database as described in this thesis seems to have merit, though there is some room for improvement. It finds some very well known joseki and fuseki sequences and none of them seem out of place. The search for global influence also produced promising results.

For example, the algorithm finds the joseki shown in Diagram 13. This joseki is known as the nadare, which means avalanche in Japanese. This joseki, and some common variations on it are described in the Kosugi/Davies book [2]. In the tree of Figure 6 it is the joseki of 9 moves deep, closest to the really long one (which seems to be a variation on the nadare not described by Kosugi and Davies). By far most of the occurrences of this joseki have a consistent black influence from below throughout the sequence. To a lesser degree, white seems to have more influence to the right, which would play well with white’s new wall. The fact that verifiable joseki such as this one can be found like this is very encouraging.

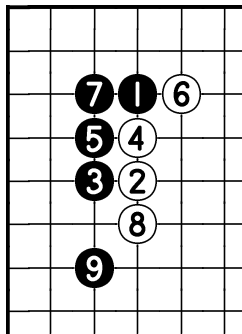


Diagram 13: The nadare. Most variations branch off after white 6.

However, the amount of joseki that were found might be seen as a bit disappointing. There are several reasons for this. First of all, I limited myself with this algorithm to the opening game and the corner moves. I also used some shortcuts that made the algorithm less precise, but simpler to understand (see Section 7). I often thought I didn’t have to be too precise in phase 1 (possibly missing some opportunities for finding joseki), because in phase 2 most mistakes would be removed anyway. In theory, if I had been more precise (though careful) in phase 1, the pruning-score in phase 2 could have been set a bit higher. Another important reason is the nature of the database. All of the players in

it are professionals, and they do not often follow joseki. And if they do, they follow different joseki from less experienced players. It could be interesting to try this algorithm on a database of amateur players.

In the algorithm, the decision whether a move belongs to a sequence or not is decided by Manhattan distance. This is a nice heuristic, but maybe the algorithm should be more smart. Especially if you want to search deeper than the opening game, since the further you go, the closer the stones on the board are played together. Of course, a data mining tool is more elegant if it is less smart (or convoluted), but go may be much too complex a game to be analyzed so easily.

And the tree is now pruned with a single pruning-score. A subtree is cut off if its score is less than the pruning-score. Because of this, some joseki are not finished properly in the results and some sequences go on too long, even if they don't deserve to. Perhaps some more intelligence could be used here too. A pruning function that takes into account how close a node score is to that of its parent. If it is close enough (even if it is below the threshold) it could be let through.

The factors described in Section 8 could also be taken into account while researching the global influence.

And perhaps there are better algorithms than this one altogether. Like frequent pattern mining around every stone that is played. This would not really store sequences, but logical rules.

Despite of these shortcomings, I hope that this thesis lays the groundwork for further exploration of joseki using data mining.

List of Diagrams

1	A typical opening game position	2
2	The first corner joseki	3
3	Another corner joseki	3
4	Xie He vs. Duan Rong, part 1	8
5	Xie He vs. Duan Rong, part 2	8
6	Xie He vs. Duan Rong, part 3	9
7	16 equivalent first moves	11
8	Illustration of noise during a sequence	12
9	Joseki free from board edge	17
10	Ladder	19
11	Monkey Jump	19
12	Calculation of left-side influence	21
13	Nadare	23

List of Figures

1	The algorithm, phase 1: creating the tree	6
2	The algorithm, phase 2: pruning the tree	7
3	Joseki tree, distance=5, pruning-score=150	13
4	Joseki tree, distance=6, pruning-score=150	15
5	Joseki tree, distance=4, pruning-score=150	16
6	Joseki tree, distance=5, pruning-score=150 + influence	22

List of Tables

1	Xie He vs. Duan Rong, the four joseki	9
---	---	---

References

[1] Sensei's Library, The collaborative Go website
<http://senseis.xmp.net>

[2] K. Kosugi and J. Davies, *Elementary Go Series, Volume 2, 38 Basic Josekis*, Kiseido Publishing Company, Eighth Printing: Februari 2007

[3] Go4Go.net, Go for it!
<http://www.go4go.net/v2>

[4] Joseki — Wikipedia, the free encyclopedia
<http://en.wikipedia.org/wiki/Joseki>

[5] Smart Game Format specifications
<http://www.red-bean.com/sgf>