Computing Science (CMPUT) 455

Search, Knowledge, and Simulations

James Wright

Department of Computing Science
University of Alberta
james.wright@ualberta.ca

Fall 2021

Topics for Today - Lecture 1

- Introduction - What is Cmput 455 about?
- Goals of course - What will I learn?
- Readings, Activities, Resources
- Assessment - quizzes, assignments, exams
- Introduction to the game of Go and Computer Go
- Demo of Python 3 programs `Go0` and `Go1`

Coursework

- Do Lecture 1 **activities** on course webpage
- Read Krakovsky, Reinforcement Renaissance
- Do Quiz 0 and Quiz 1 on eClass (they open after class, 3:20pm Sep 2)

Part I

Intro - Problem Solving for Humans and Computers

What is Cmput 455 about?

Broad Goals of this Course:

- The main technologies in modern heuristic search
- From basics all the way to AlphaGo, Alpha Zero, and beyond
- Gain a full understanding of the foundations
- Study the biggest successes
- See how they came about
- See working code using games such as Go, TicTacToe
- Learn how to apply techniques in own projects

Organization - Main Points

- This course has only lectures. No labs
- Activities - do at your own pace (before the deadline)
- Coursework - readings, assignments, quizzes, exams
- Main course site `https://jrwright.info/cmput455/`
  - All content - slides, assignments, course information
- EClass course site `https://eclass.srv.ualberta.ca/course/view.php?id=72409`
  - Write quizzes, exams, submit assignments, read/write forum, announcements, access readings

Teaching Team, Office Hours, Forum

- Instructors: James Wright (james.wright)
- TAs: Abbas Tork (masoumza), Amir Sattarifard (sattarif)
- We will monitor the eClass forum and answer questions
    - Asking questions on the forum is strongly preferred to emailing me
- Also watch the announcements on eClass
- We will have office hours
    - James: After lecture
    - Amir: Wednesdays 1-2pm
    - Abbas: Fridays 1-2pm
    - Also see Teaching Team webpage

What Will I Learn - 455 Goal Statements

1. To understand
   modern computer problem-solving methods
   - which use a combination of search, machine-learned knowledge, and simulations
2. To achieve a working knowledge of
   how to model decision-making tasks
   - in both humans and machines
3. To study randomized search methods
   such as Monte Carlo Tree Search
   - and practice how to improve such programs by machine learning

Topics of Cmput 455

Five topics, 4 - 6 lectures each

1. Introduction - problem solving for humans and computers
2. Search and Knowledge
3. Simulations and Monte Carlo Tree Search
4. Machine Learning for Heuristic Search
5. Reinforcement Learning, AlphaGo and Beyond

Background/Prerequisites

- Very weakly defined prerequisites
  Any 300-level CS course
- It is a 4th year course
- I assume you have broad general CS knowledge
- I do not assume specific knowledge beyond basics
- Quiz 0 has many questions about your background

Dealing with Gaps in Background

- All of you will have different gaps
- We provide some optional reading material to cover gaps
  - Examples: Python bootcamp, basic algorithms
- You can refer to those case-by-case, as needed

Is Cmput 455 Right for You?

- Goal for now:
    - Give you a good estimate of how much work this course is for you...
    - ...before the course drop deadline
    - General approach:
        - Lower math content
        - Focus on important concepts (precise but not too formal)
        - Fair bit of experimenting and programming in Python 3
- Know lots already? Optional materials allow you to dig deeper.
    - You can always ask me for more materials

Course Resources

- Directly from main course page:
    - Course outline, policies, slides, readings, activities, sample code, assignments
- Other resources linked from main page:
    - Python programming
    - Algorithms review and sample codes (from Cmput 204)
    - Useful software, e.g. Go programs and tools
    - Study guides (published before exams)
    - Weblinks, blog posts, videos, assorted textbooks,...

Python Programming

- We use Python 3 code throughout
- Course code - see website
- Python programming - some references listed, use as needed
- I expect you can read all sample code given
- I expect you can modify code and write new functions and tests
    - Used in assignments and activities
    - Tested in quizzes and exams

Coursework and Assessment

- Readings and other activities
- Quizzes
- Coding assignments
- Midterm and final exam on eClass

See outline for percentages of each part

Readings And Activities

- Read article or do activity
- Readings and activities prepare and expand topics from class
- Some also prepare for assignments
- Organized by lecture, on readings and activities webpage

Quizzes

- 20% of total marks, 1-2% per week
- One quiz per week, some are double length
- Quizzes review classes, plus some reading/activities
- Marked automatically in eClass
- Selected questions will be reviewed in class afterwards

Quiz 0 and 1

- Quiz 0 and 1 published now (1% each)
- Will open on eClass today after class (3:20pm)
- Quiz 0 is “participation only”
  - You get marks just for doing it
- Quiz 1 is regular, marked for correctness
  - Topic: game of Go
  - Review of today’s lecture

Coding Assignments

- Relatively small (worth 5% each)
- About 3 weeks for each
- Teams of up to three students
    - Read details as part of Activity 1a
- All assignments use the game of gomoku (see later)
- Start from a Go program provided as Python 3 code
    - Some of the Activities prepare for assignments

Coding Assignments (2)

- Typical tasks: Add functionality, test
- We provide tools for your own testing
    - See activities: install Python 3, tools, first Go programs
- Marking done by TA
- Automated scripts to test your code
- Scripts send text commands to your program, check the answer computed

Coding Assignments - Team Submissions

- One submission per team from *designated submitter*
- Details on webpage: `https://jrwright.info/cmput455/assignments/assignments.html`
- **Follow format requirements exactly**
  - Formatting mistakes are a leading cause of frustration and wasted time for both you and us
- We will post detailed instructions for how you test your submission

Coding Assignments - Testing, Feedback and Submission

- For each assignment we will provide sample test data
- You must do testing as part of your assignment
- The day after the submission deadline, TA will run automated tests on a standard lab machine
- You will get feedback, e.g. if files are missing, or if your program does not run
- Submission deadlines are absolutely firm

Coding Assignments - Late Submission

- You can do a late submission, for any reason
- Deadline is 2 days after the regular deadline
- Late submissions are marked with a 20% deduction
- Example: if your normal submission did not work, the TA will tell you the problems found by the script. Fix them and do a late submission.
- **Important:** the only way to react to, and fix, submission problems is for problems with the **regular** submission. There is **no second round** of feedback on late submissions.

Assignment 1

- Assignment 1 published on website
- Start from our sample code, the `Go0` program
- You will modify it in the assignment
- Preview in second lecture

Midterm and Final Exam

- Will be conducted over eClass
- Will follow format of the Quizzes
- Study guide will be published before each exam

Honesty and Plagiarism

- Don’t cheat. We will check
- Be aware of collaboration rules
- Link to rules: on policies page

Summary

- Discussed content, format, rules and expectations for this course
- Everything is on web for your later reference
- **Do use the eClass discussion forum**
- **Do use the instructor and TA office hours**

Introduction to Go and Computer Go

Topics:

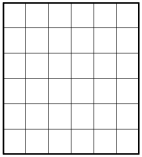
- Game of Go
- Rules of Go
- Scoring
- Strength of Go players and rating system
- Quick introduction to computer Go
- Random Go player `Go0`
- `Go1`: fix `Go0` to make it finish a game

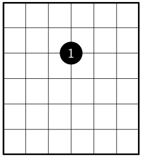
Game of Go

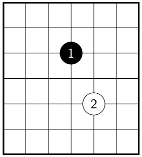
Image source: `https://upload.wikimedia.org/wikipedia/commons/2/2a/FloorGoban.JPG`

- Classic two player board game
- Most popular in East Asia
- Invented thousands of years ago in China
- Simple rules, complex strategy
- Played by millions
- Hundreds of top human experts - professional players

Game of Go Rules - Basics

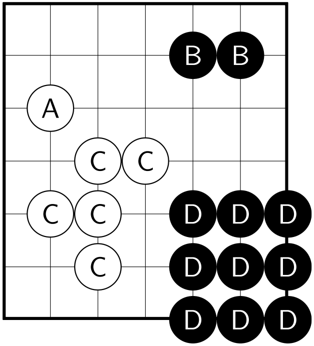






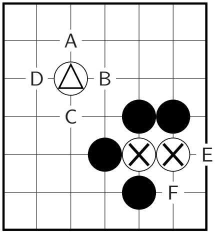
- Start with an empty grid
- Usual size is 19×19
- We will often use 7×7 in this course
- Two players Black and White
- Black goes first
- Move: place a stone of your color on an intersection
  - An intersection is also called a *point*
- Example: empty board, first move by Black, second move by White

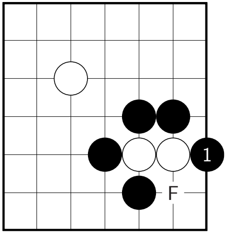
Game of Go Rules - Blocks



- Connected stones of the same color are called *blocks*
- A is a single stone block
- Two stones B are connected by a line. They are one block
- C is a single block of 5 white stones
- D is a block of 9 black stones
- A and C are *not* in the same block
    - No connection diagonally

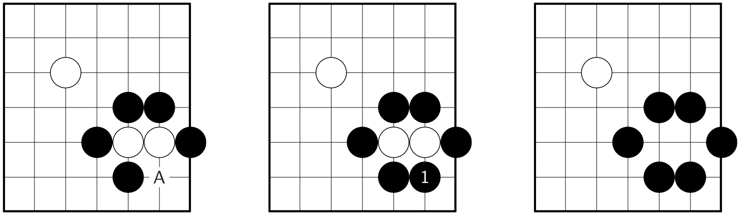
Game of Go Rules - Liberties





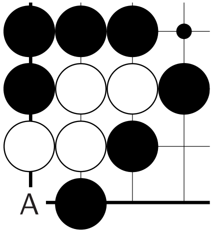
- Empty points adjacent to a block are called *liberties*
- The single marked white stone has four Liberties A, B, C, D
- The block of two marked white stones has two liberties, E and F
- After Black plays on 1, the white stones have only one liberty at F left
- A block that loses its last liberty is *captured* (see next slide)

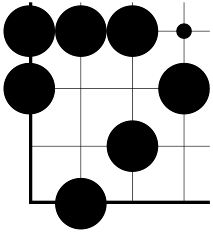
Game of Go Rules - Capture



- The block of two white stones has only one liberty at A
- Black can play there
- Effect: the two stones are *captured*
- Removed from the board

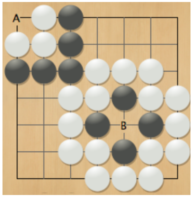
Illegal Move - Suicide

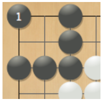




- Example with White to play
- White at A would be *suicide*
- White would take its own last liberty
- Suicide is forbidden in most versions of Go rules
    - In this course: we *never* allow suicide
- Capturing always takes precedence over suicide - see next slide

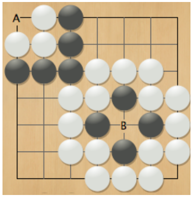
Capture vs Suicide: Example 1

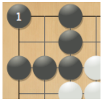




- Top left:
  move A for Black looks like suicide

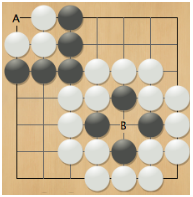
Capture vs Suicide: Example 1



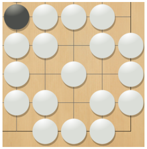


- Top left:
  move A for Black looks like suicide
- However, move A **also**
  takes the last liberty
  of the three white stones
- Move A is a capture as well
- Capture takes precedence over suicide
- Move A is legal for Black

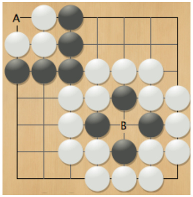
Capture vs Suicide: Example 2

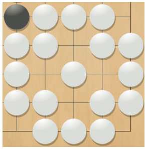


- Is B a legal move?



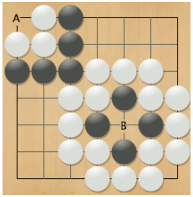
Capture vs Suicide: Example 2

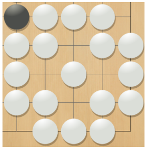




- Is B a legal move?
- It looks like suicide for White at first sight
- However, it also captures four single black stones
- Capture takes precedence
- Yes, move B is legal

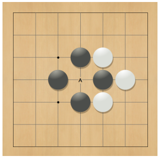
Capture vs Suicide: Example 2 Continued

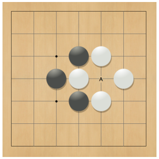


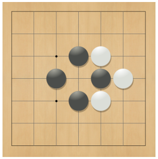


- After the capture, the new white stone does have liberties
- This holds in general - after any legal move, all blocks have at least one liberty
- What if you find a block without liberties in your game?
    - You made an illegal move
    - Or you forgot to remove some captured stones (more likely)
    - Of course, correct Go programs should never get into such a state
- For **Black to play B**, would be illegal - suicide

Repetition Rules - Basic Ko

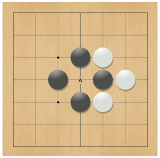


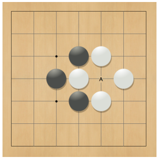




- From top to middle picture: White can capture one black stone by playing A
- From middle to bottom picture: Now if Black captures back one white stone...
- The position would repeat, infinite loop
- This is called a (basic) ko.
- Go rules forbid such repetition

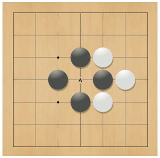
Resolving a Ko Situation

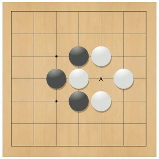




- Ko rule: after White captured, Black cannot re-capture **right away**
- Q: How to resolve the situation?

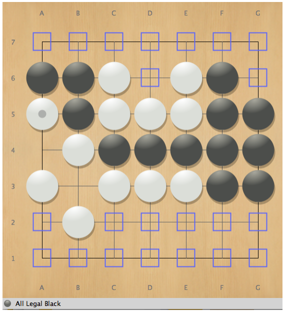
Resolving a Ko Situation





- Ko rule: after White captured, Black cannot re-capture **right away**
- Q: How to resolve the situation?
- Black must play somewhere else
- Now White has a chance to connect
- If White also plays elsewhere, then Black can capture
- There are more complex ways to create illegal loops (may discuss later)
    - Basic Ko is by far the most common

Game of Go Rules - Legal Moves

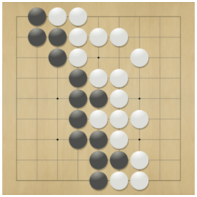


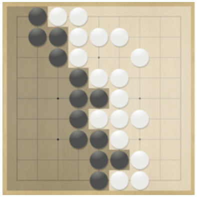
- Legal move:
  play on any empty intersection, except points forbidden by:
  - repetition (ko rule)
  - suicide
- Example:
  legal moves for Black, after White captured a ko
  - A4 forbidden by repetition (ko rule)
  - B3 forbidden by suicide

Legal Moves - Pass Move

- *Pass move* is always allowed
  - Board does not change
  - It is now the other player's turn to play
- Usually, there are some moves better than *Pass*
- Competent players only pass at end of game

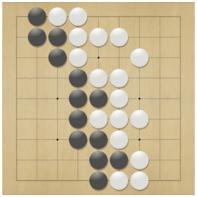
End of Game and Scoring

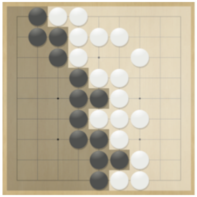




- Game ends after two successive passes
    - Some rule versions require three passes
- Next, count the score for each player - stones plus territory
- Add the *komi* (adjustment for going second)
- The winner is the player with higher score
- Draws are possible if the komi is integer

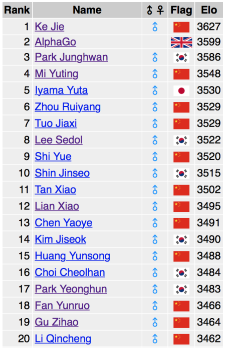
Scoring Example





- Assume komi = 7.5
- Black score = 37
    - 13 Black stones +
    - 24 empty points surrounded by Black
- White score = 51.5
    - 17 White stones +
    - 27 empty points surrounded by White +
    - 7.5 komi
- White wins by 51.5 - 37 = 14.5 points

Playing Strength and Rating System



Rank	Name	♂ ♀	Flag	Elo
1	Ke Jie	♂	🇨🇳	3627
2	AlphaGo		🇬🇧	3599
3	Park Junghwan	♂	🇰🇷	3586
4	Mi Yuting	♂	🇨🇳	3548
5	Iyama Yuta	♂	🇯🇵	3530
6	Zhou Ruiyang	♂	🇨🇳	3529
7	Tuo Jiaxi	♂	🇨🇳	3529
8	Lee Sedol	♂	🇰🇷	3522
9	Shi Yue	♂	🇨🇳	3520
10	Shin Jinseo	♂	🇰🇷	3515
11	Tan Xiao	♂	🇨🇳	3502
12	Lian Xiao	♂	🇨🇳	3495
13	Chen Yaoye	♂	🇨🇳	3491
14	Kim Jiseok	♂	🇰🇷	3490
15	Huang Yunsong	♂	🇨🇳	3488
16	Choi Cheolhan	♂	🇰🇷	3484
17	Park Yeonghun	♂	🇰🇷	3483
18	Fan Yunruo	♂	🇨🇳	3466
19	Gu Zihao	♂	🇨🇳	3464
20	Li Qincheng	♂	🇨🇳	3462

Top 20 Go players, January 2017.

Source:

`https://www.goratings.org`

- Rating system with amateur student (kyu) and master (dan) grades
- Separate rating system for professional players
- Numerical rating systems, similar to Elo in chess
- No single wordwide system, each organization has their own

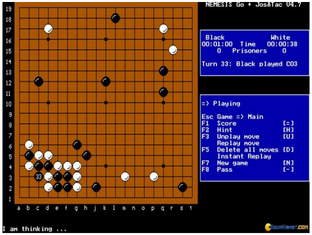
How to Learn to Play Go

- Becoming a serious Go player is not required for this course
- However, you should understand the basic concepts well
- Many Go-related resources on our course resource page
    - Internet Go servers, video lessons, addresses of clubs, computer opponents

Quick Introduction to Computer Go

- Computer Go, from beginnings to AlphaGo
- Examples: `Go0` and `Go1`, random Go players written in Python 3
- How to program a computer to play Go?
- Studied for over 50 years
- Considered the hardest of the classical games

Computer Go - Beginnings



```
 Black              White
00:01:00  Time   00:00:38
    0   Prisoners    0

Turn 33: Black played C03
```

```
I am thinking ...
```

Nemesis, an early commercial Go program.

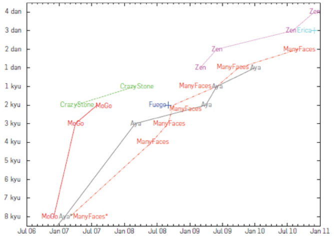
Source:

`http://blogs.discovermagazine.com/science-sushi/2016/03/10/go-ai-alphago-nemesis`

Early programs:

- Hand-written rules and patterns to generate moves
- Try to implement human Go knowledge
- Specialized goal-oriented search to capture stones
- Level: advanced beginner
- Slow progress

Computer Go - Monte Carlo Tree Search

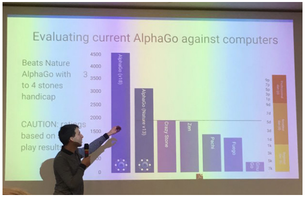


Monte Carlo Tree Search Revolution.

Image source: `acm.org`

- Monte Carlo Tree Search (MCTS)
- Developed about 12 years ago
- Breakthrough in playing strength
- Small boards ($7 \times 7, 9 \times 9$): level of top human professionals
- 19×19: Close to top amateur after 6-7 years of research
- Clearly weaker than professionals
- MCTS was first applied to Go
- Today, used for many other decision-making problems

Computer Go - AlphaGo



Evaluating current AlphaGo against computers
Beats Nature AlphaGo with 3 to 4 stones handicap
CAUTION: ratings based on self-play results
AlphaGo (v18)
AlphaGo (Nature v13)
Crazy Stone
Zen
Pachi
Fuego

Picture of David Silver's talk at UCL,

unknown photographer

- 2015 - 2017:
  AlphaGo quickly surpasses human professionals
- Project by Deepmind in London
- Led by two UofA alumni, David Silver and Aja Huang
- MCTS, deep convolutional neural networks, deep reinforcement learning
- Far exceeds human abilities
- Matches and sample games:
  `www.alphago-games.com`

Computer Games - Beyond AlphaGo

- AlphaZero: learn from rules and selfplay only, no other human knowledge
- MuZero: learn rules as well, from sample games
- Poker, Atari, Starcraft, etc. - beyond classic board games

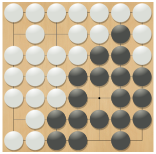
Go Program Demo

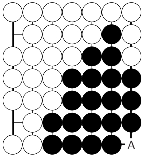
- We use our own simple Go programs in this class
- Written in Python 3
- Communicate via `GTP` - a text-based interface
  - Can run it directly from console
  - Often easier to use a graphical user interface
  - See Activity 2D (“Install Gogui”): `https://jrwright.info/cmput455/html/activities.html`

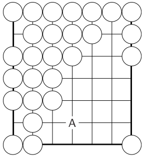
Go0: Random Player on 7×7 Board

- `Go0` is our first example
- Algorithm:
    - Create list L of all legal moves on board
    - If L is empty, then play pass
    - Else select one move m from L uniformly at random
    - Play m
- Python 3 program: `Go0.py`
- Our demo uses a 7×7 board

Problem With Go0 Player





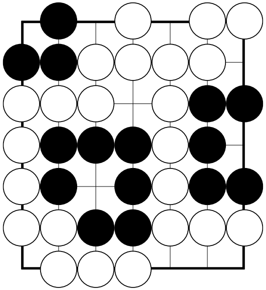


- `Go0` fills the board, but then ...
- It never seems to stop with two passes
- It cannot keep any stones safe
- It fills its own liberties and territories
- Eventually, even strong-looking stones get captured
- Game never ends...

How to Fix the `Go0` Player?

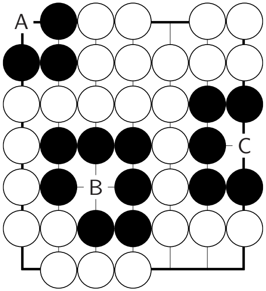
- Plan: disable some of the most obvious stupid moves
- Make sure the game ends in reasonable time
- Make sure safe stones don’t get captured
- Surrounding territory is a big part of Go
- Filling one’s own territory afterwards is usually bad
- Simplest case: “one point eyes”

Eyes



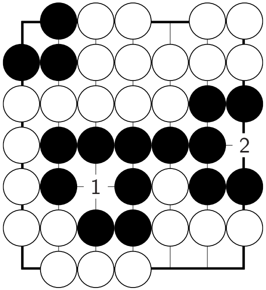
- An eye is a point that is surrounded by one color
- An eye makes stones safer
- Opponent cannot play in eyes surrounded by black stones
  - Suicide, illegal to play there for white

One Eye is Not Enough



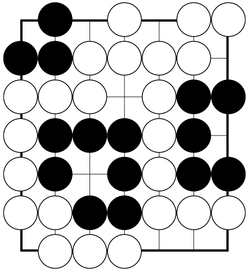
- One eye is not enough
- Moves inside eyes A, B, C become legal if they are a capture
    - Examples: move A takes the last liberty of the three surrounding black stones
    - One eye helps, but not enough for safety

Stones with Two Eyes are Safe



- Here, Black has one block surrounding two eyes 1 and 2
- White cannot attack
  - Both 1 and 2 are suicide for white
- Black is safe as long as Black leaves the eyes alone
- Black should NEVER play 1 or 2
  - Can always pass,
    if no good moves left

How to Recognize a Simple Eye?



Simple eyes for Black:

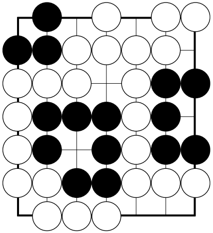
1. top left corner
2. right edge of board
3. center

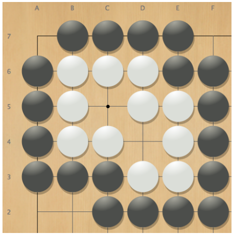
Definition of simple eye:

1. Single empty point *p*
2. All neighbor points *nb*(*p*) occupied by stones of the *same color*
3. All these stones are *connected* in a single block

- Question:
  by the definition above, which points are simple eyes for White?
- There are other, more complex kinds of eyes (later)

Detecting Simple Eyes Locally





- Can detect most simple eyes locally
  - Only look at neighbors and diagonals
  - Corner, edge:
    need all diagonal points to connect
    (1 in corner, 2 on edge)
  - Center: need at least 3 of 4 diagonal points to connect
- Can connect along some longer path
  - Pretty rare, ignored in `Go1`
  - Example: A7 is an eye
    Stones A6 and B7 connected over a long path

Simple Eyes - Summary

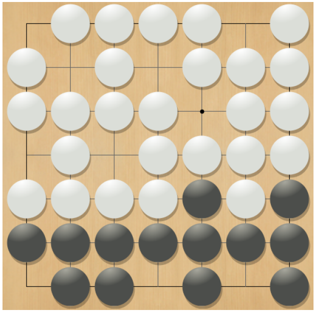
- Random player keeps playing senselessly...
  ...unless we stop it from filling its eyes
- A *simple eye* is an empty point,
  surrounded by a connected block of stones
- A local connection check finds almost all simple eyes
- Very fast to check in program, only look at a maximum of 8 neighbors and diagonals
- Having two (or more) eyes
  makes a block safe from capture

From Go0 to Go1

- `Go1` algorithm avoids filling simple eyes
- Implementation in `board_util.py` function `generate_random_move`

```
moves = board.get_empty_points()
np.random.shuffle(moves)
for move in moves:
    legal = not board.is_eye(move, color) \
            and board.is_legal(move, color)
    if legal:
        return move
return PASS
```

`Go1` in Practice



- `Go1` program ends game with two passes in the position at left
- `Go0` would continue senselessly, fill eyes, capture etc.
- `Go1` is still mostly random
- It stops when all moves fill simple eyes
- First usable version of our program
- Basis for all future programs which add search, simulations, knowledge