Research, impact, and playing games

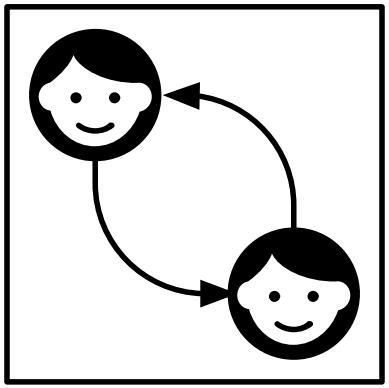
Chris J. Maddison

PRISM, Department of Computer Science, UofT, 2021

Today

How to build computer programs that play **two-player games**.

Game-playing

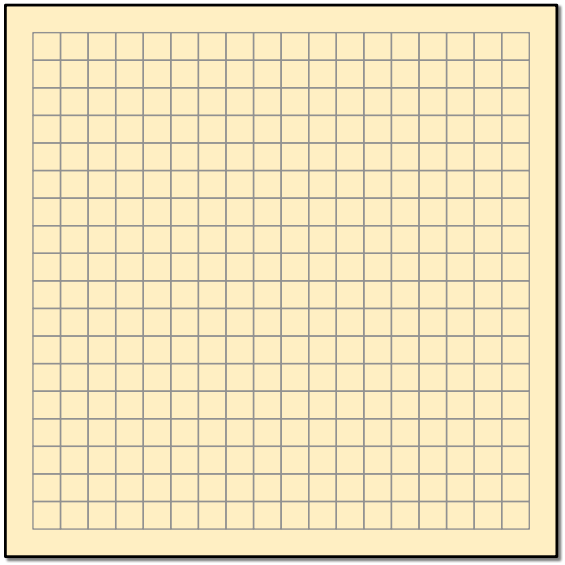


Today

- Game-playing has always been at the core of CS.
  - Simple well-defined rules, but mastery requires a high degree of intelligence.
- Landmark achievement: in 1997 DeepBlue defeated Garry Kasparov in chess.
- The East Asian board game Go was the last classical game for which we could not build strong bots.
  - In this talk, I hope to give you a brief history of computer Go game playing.
  - I will end on the story of DeepMind's AlphaGo bot, which defeated the world champion Lee Sedol in 2016.

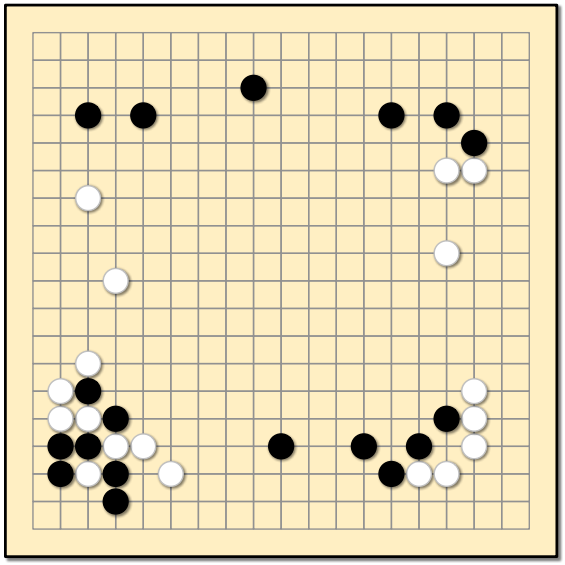
The game of Go

- Initial position is an empty 19×19 grid.



The game of Go

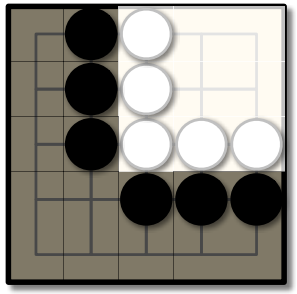
- 2 players alternate placing stones on empty intersections. Black stone plays first.



The game of Go

- **(Territory)** The goal is to capture territory.
- The player with the maximum number of occupied or surrounded intersections wins.

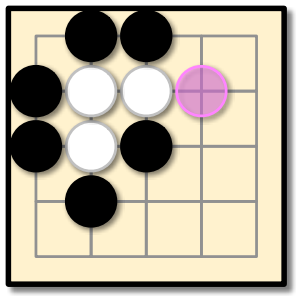
Territory



The game of Go: Play

- **(Capture)** You can remove any connected group of your opponent's stones by surrounding them.
- **(Ko)** Players cannot recreate a former board position.

Capture



Outline of the lecture

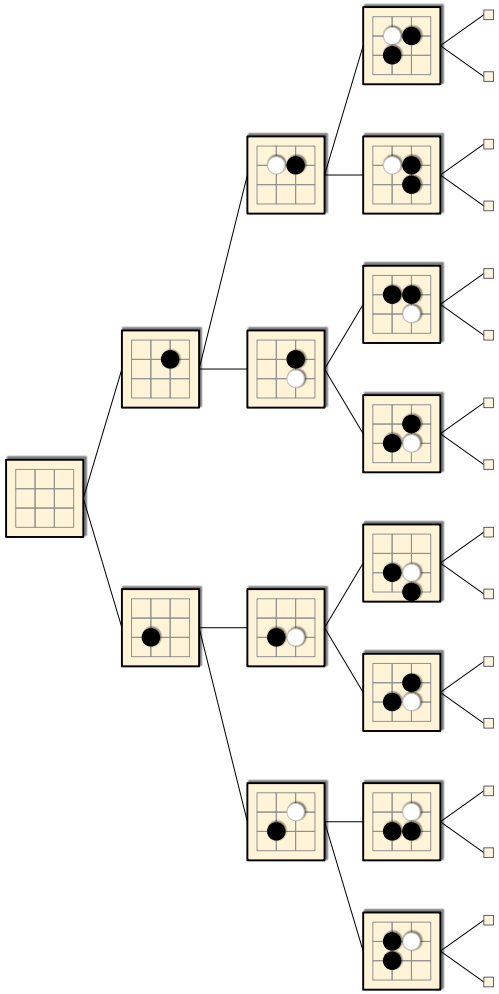
To build a strong computer Go player, we will answer:

- What does it mean to play optimally?
- Can we compute (approximately) optimal play?

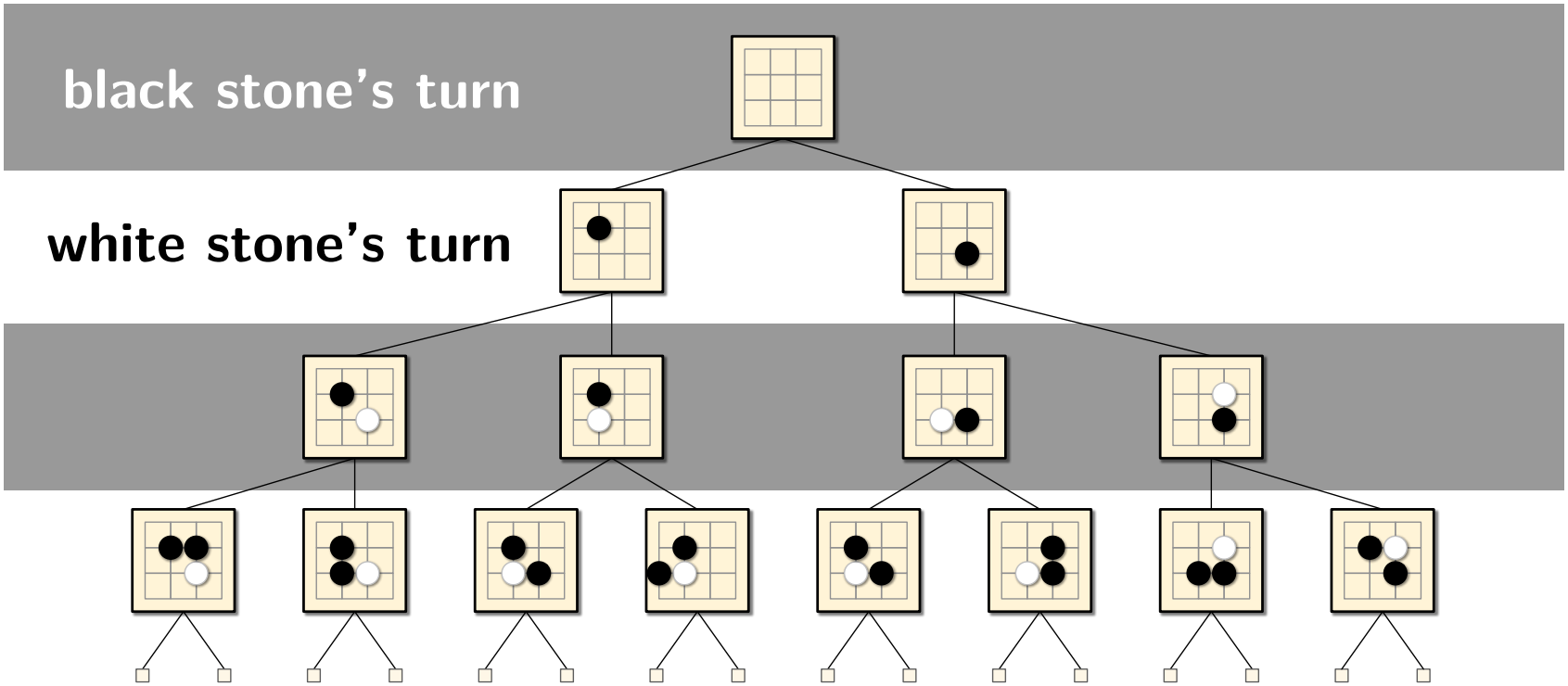
Optimal play

Game trees

- Put all possible games into a tree.
  - Each node s contains a legal position.
  - Child nodes enumerate all possible actions taken by the current player.
  - Leaves are terminal states.
- The Go tree is finite (Ko rule).

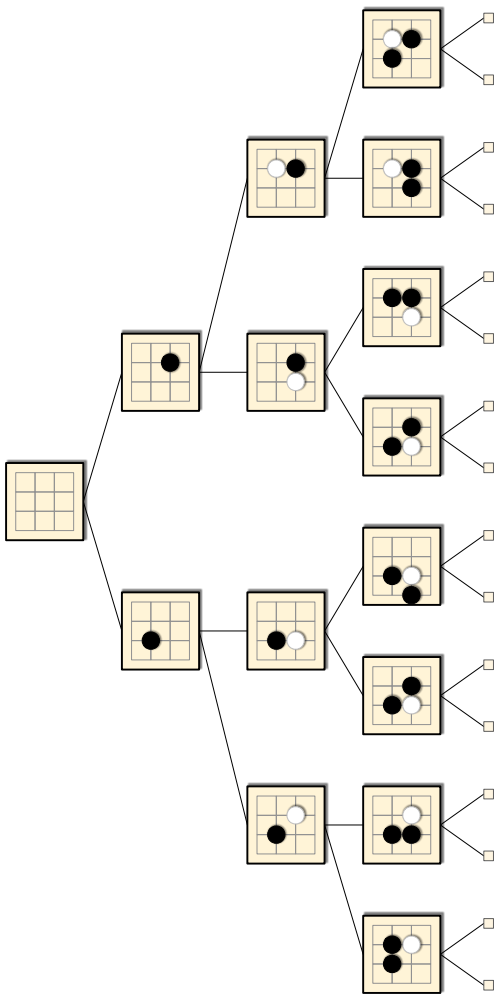


Game trees



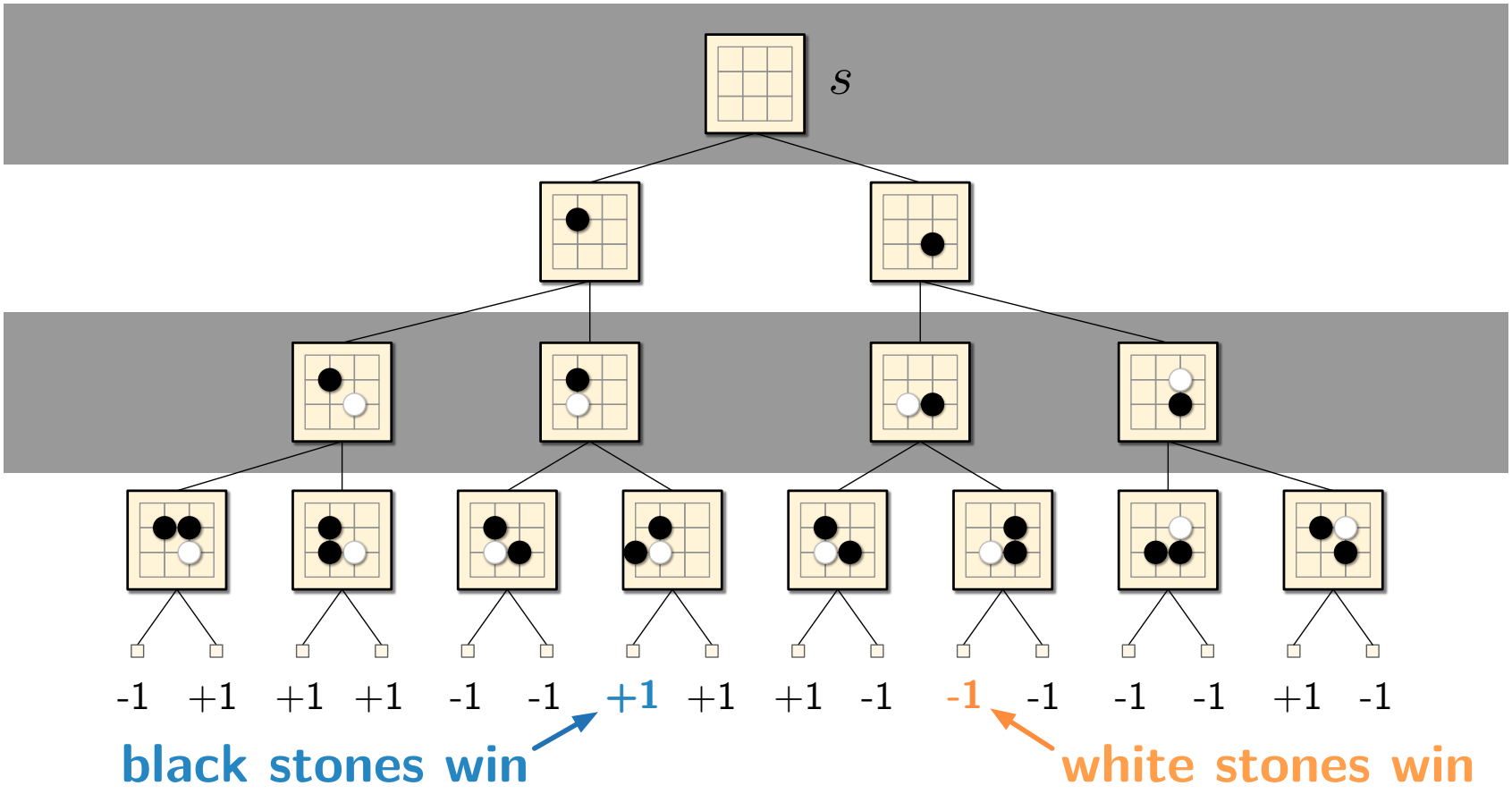
Evaluating positions

- We will **assign a value $v^*(s)$ to each node s, which tells us who is winning**.
  - Use three values $\{+1, -1, 0\}$.
  - $+1$ if a position is guaranteed to be a win for the black stone player, -1 if it is a guaranteed loss for black, and 0 for a guaranteed draw.
- **In optimal play, every state is either completely winning, losing, or drawing.**



Evaluating leaf positions

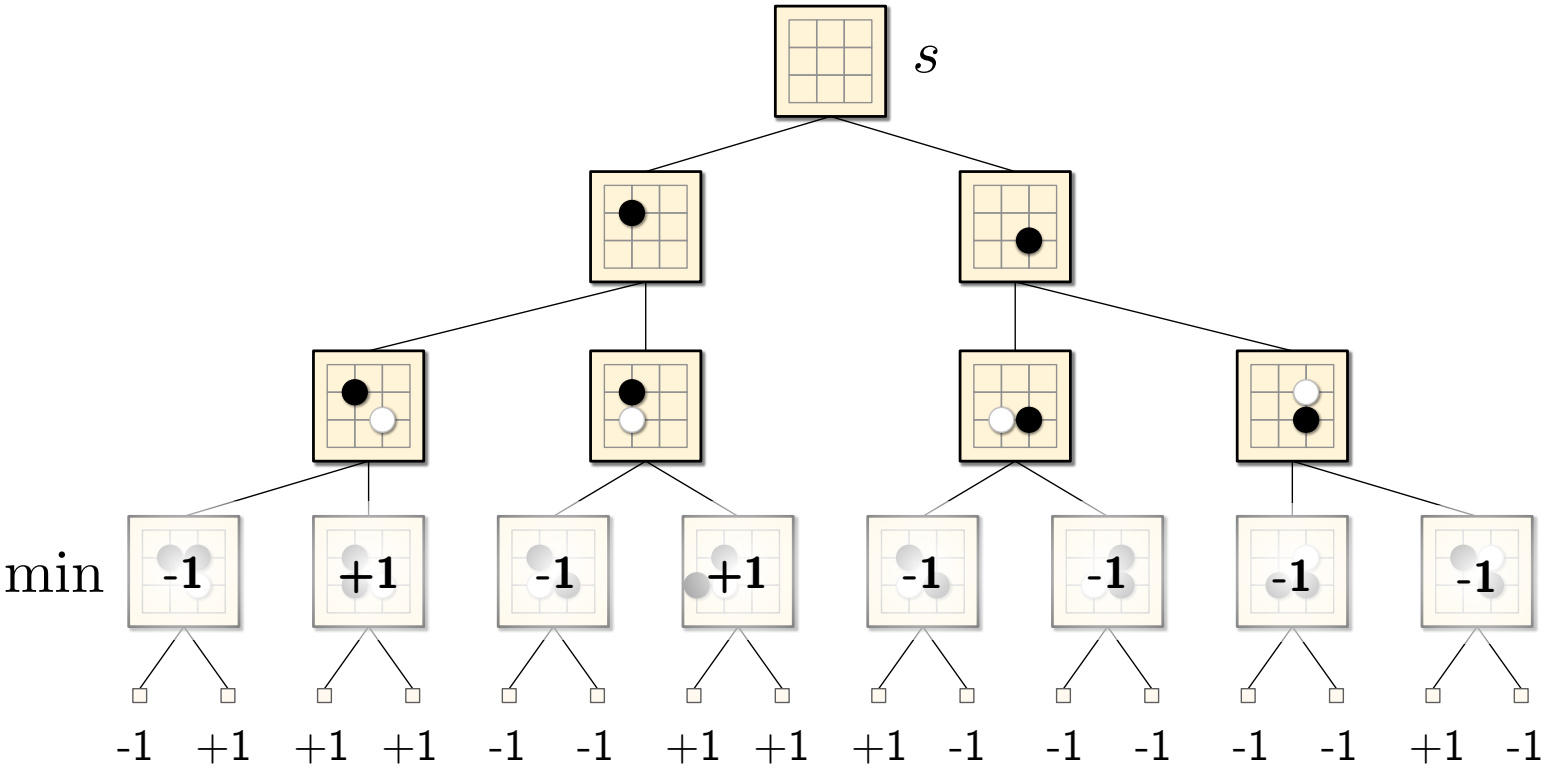
At a leaf, the winner is determined.



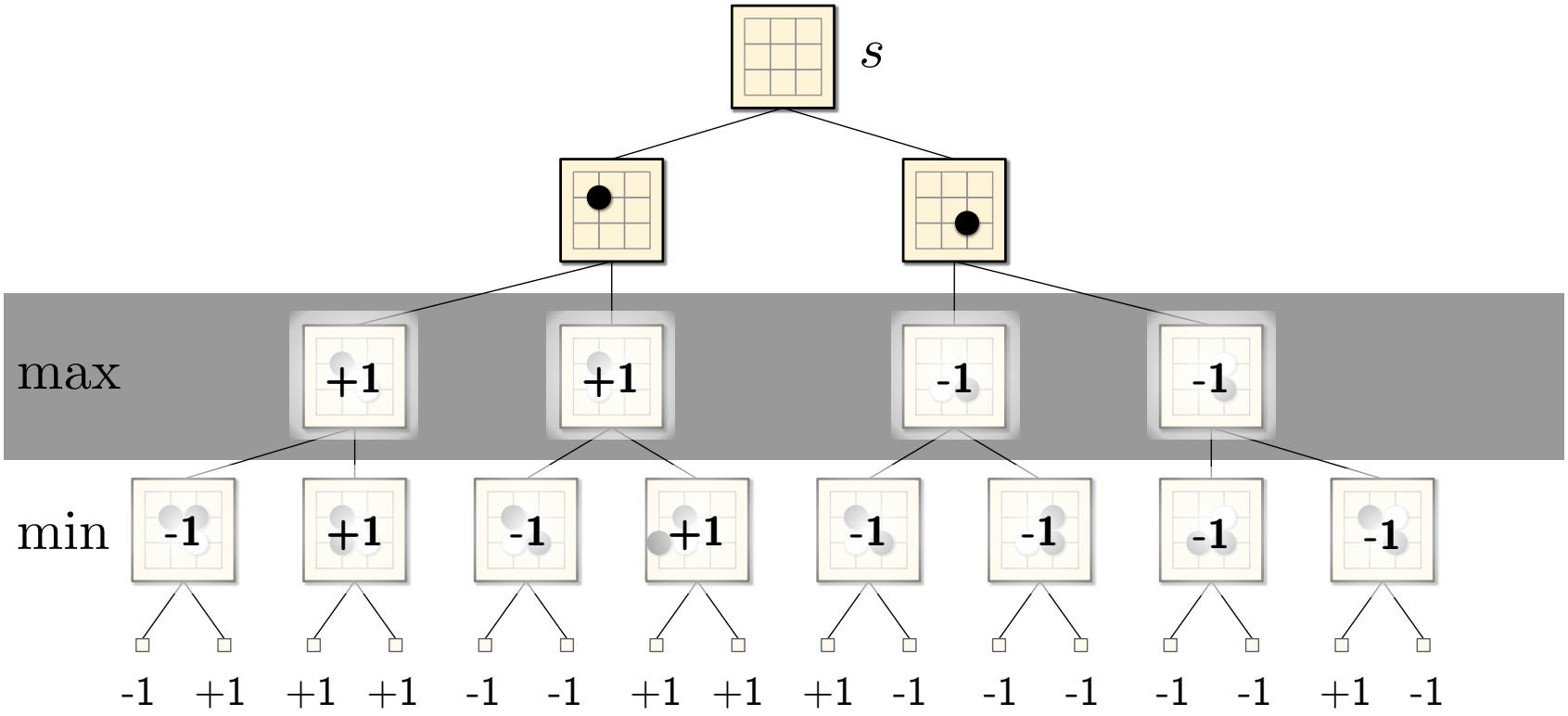
Evaluating internal positions

- The value of internal nodes depends on the strategies of the two players.
- Define $v^*(s)$ to be **the best achievable value for black in the worst case**.
  - i.e., the highest value that the black stone player can achieve against an optimal white stone player.
- Can compute this recursively up the tree.
  - Black stone player is trying to maximize, white stone player is trying to minimize the value.

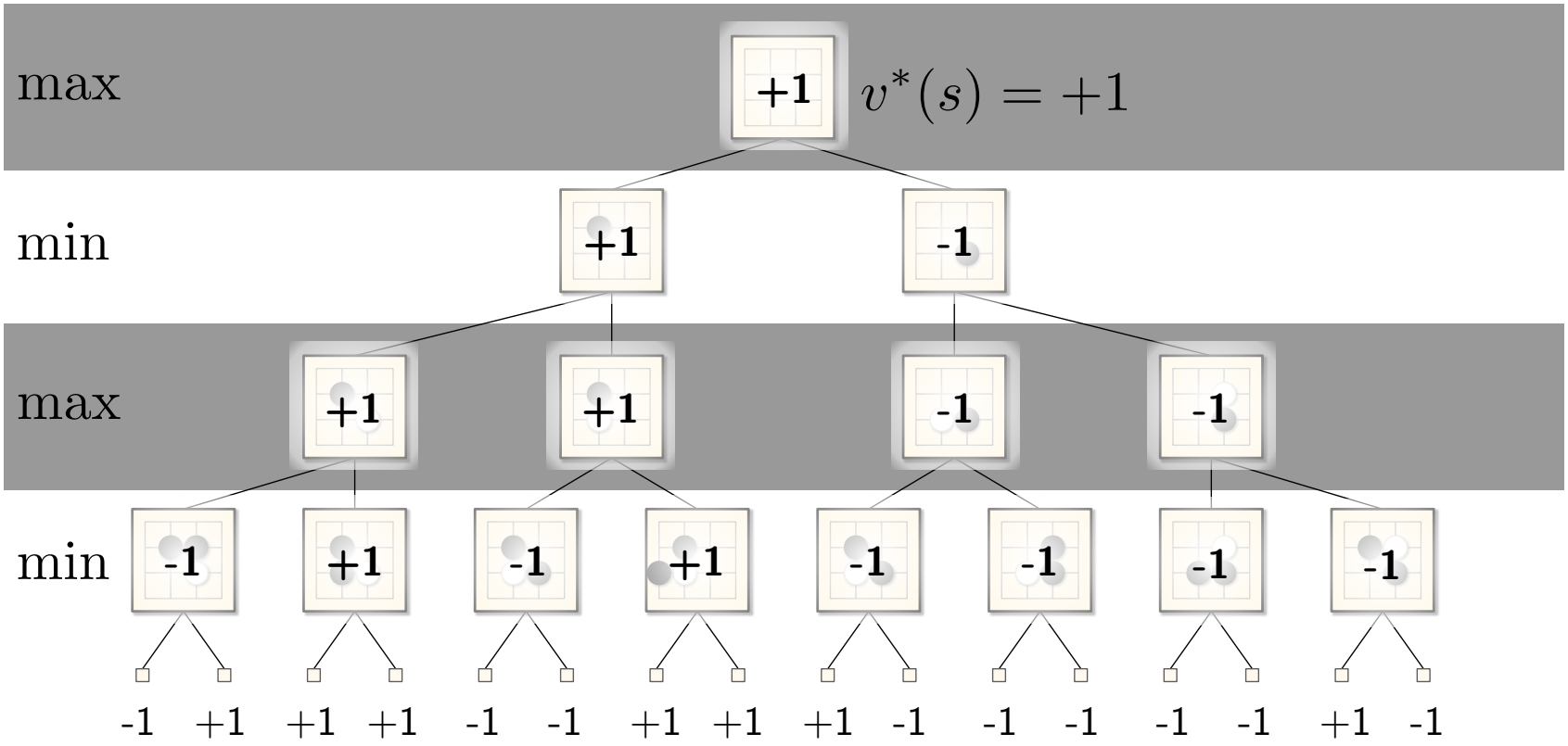
Evaluating positions under optimal play



Evaluating positions under optimal play

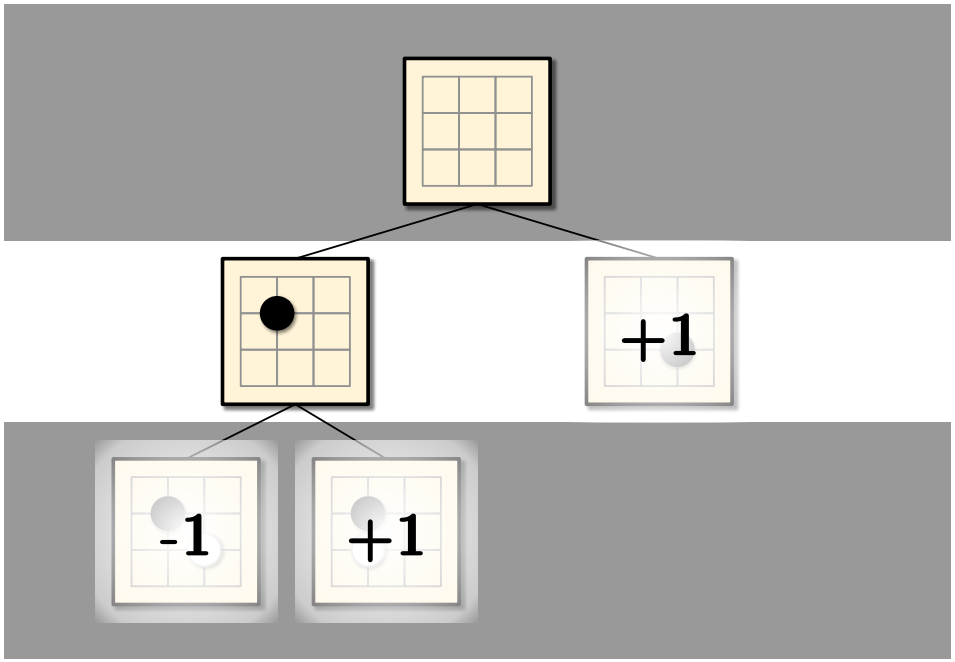


Evaluating positions under optimal play



Quiz!

$v^*(s) = ?$



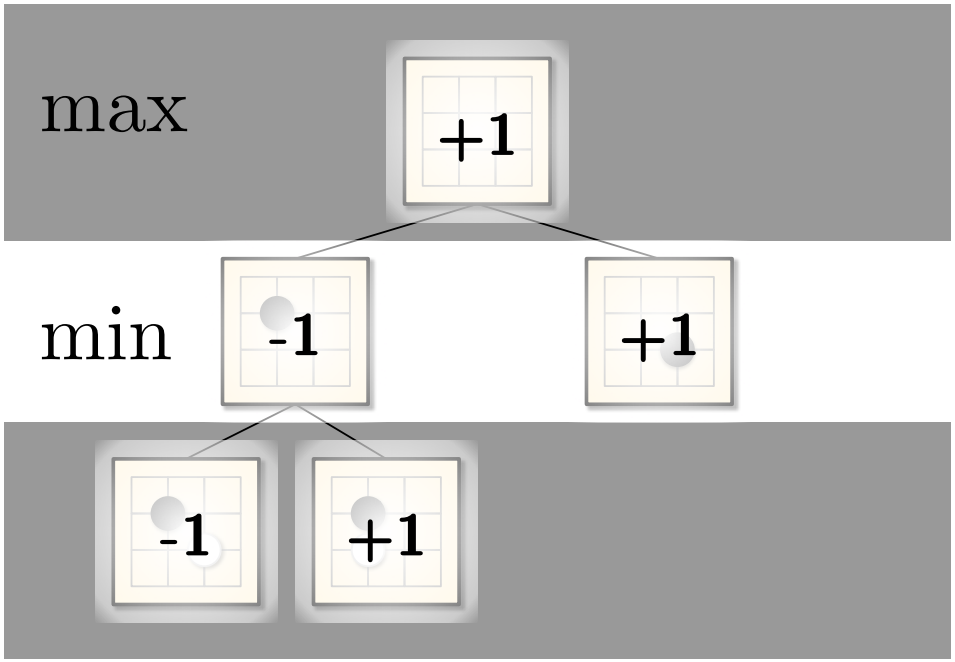
What is the optimal value $v^*(s)$ of the root?

1. -1?
2. $+1$?

Recall: black plays first and is trying to maximize, whereas white is trying to minimize.

Quiz!

$$v^*(s) = +1$$



What is the optimal value $v^*(s)$ of the root?

1. -1?
2. $+1$?

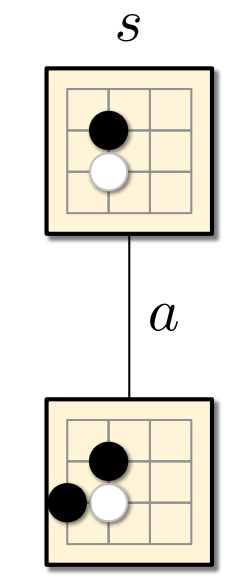
Recall: black plays first and is trying to maximize, whereas white is trying to minimize.

In a perfect world

- All you need is the optimal value function to play optimally in the worst case.
- Pick the move that maximizes your value in the next state:

$$a^* = \arg\max_a \{v^*(\mathrm{child}(s, a))\}$$

- Claude Shannon (1950) pointed out that you can find a^* by recursing over the whole game tree, **minimax search**.
  - Go has $\sim 10^{170}$ legal positions in the tree...

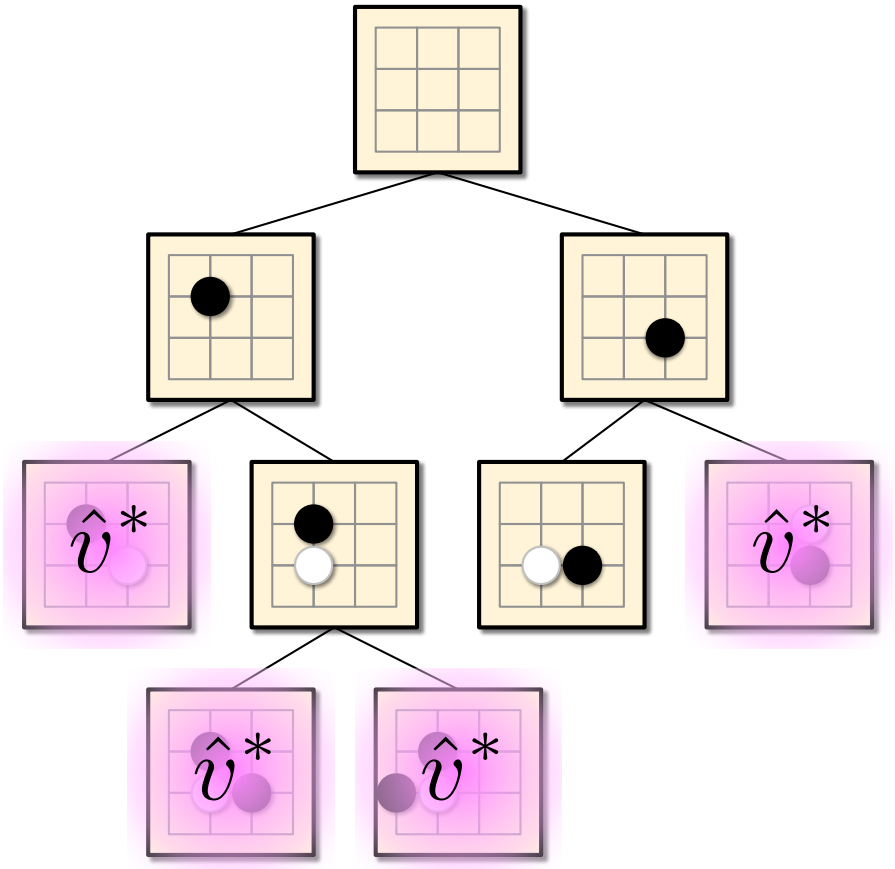


$\mathrm{child}(s, a)$

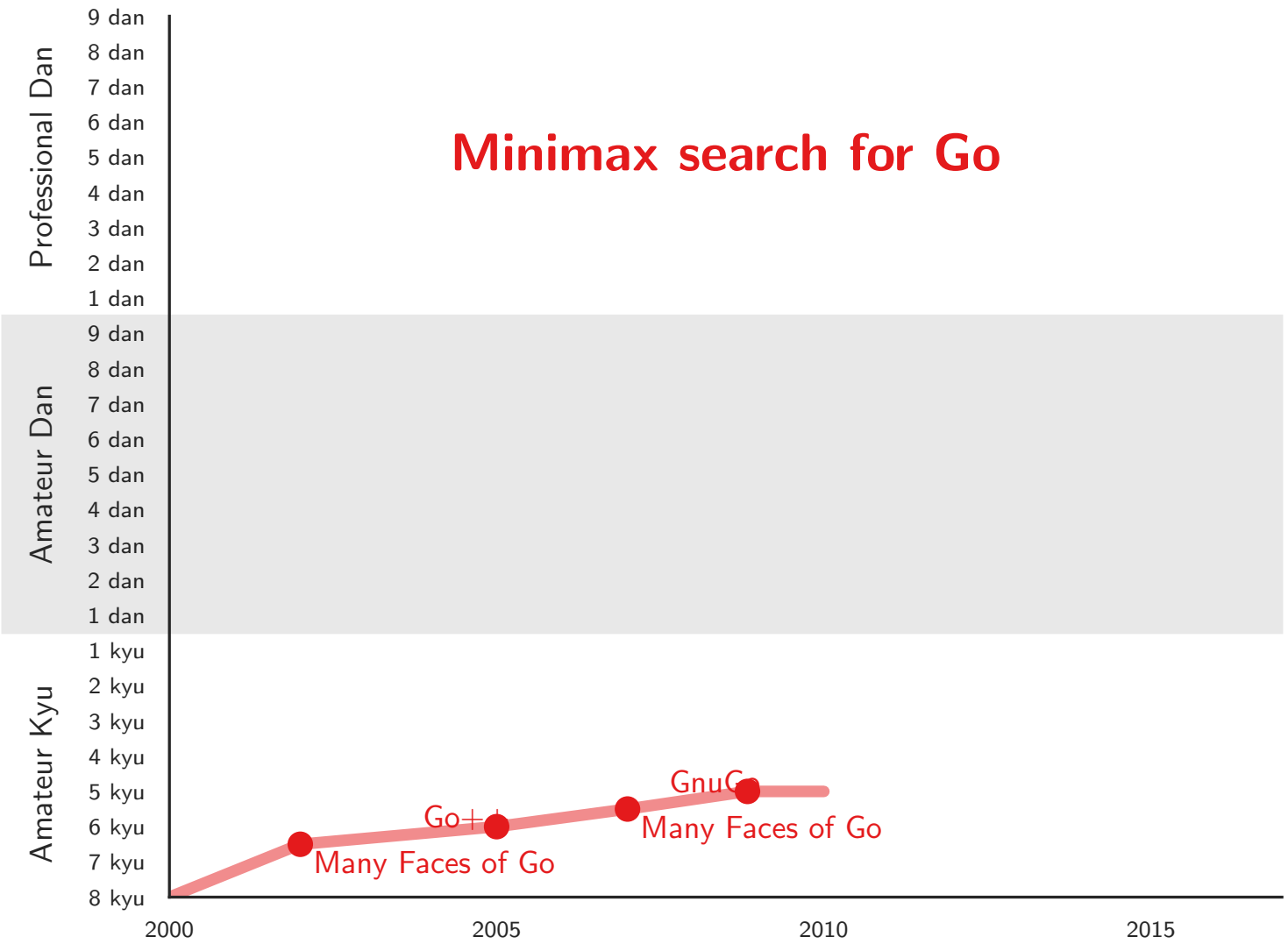
Approximating optimal play

Depth-limited Minimax

- In practice, recurse to a small depth and back off to a **heuristic evaluation** $\hat{v}^*$.
  - Designed by experts.
  - For Go (Müller, 2002).



Progress in Computer Go



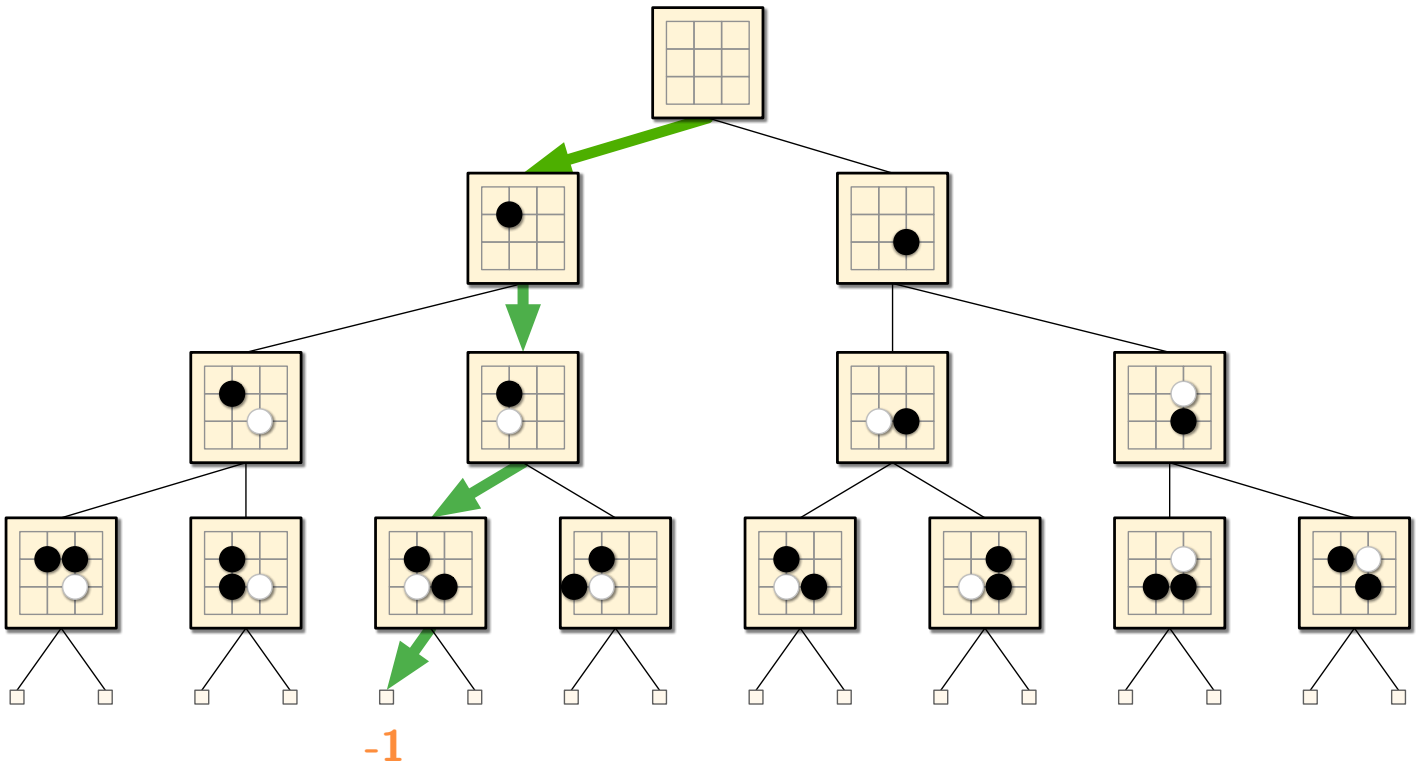
adapted from Sylvain Gelly & David Silver, Test of Time Award ICML 2017

Problems with minimax

- Designing heuristic evaluations is hard, especially so for Go.
  - Somewhat obvious, otherwise search would not be needed!
- Depth-limited minimax is brittle and very sensitive to misevaluations.
- Can we design cheap evaluators and still behave optimally?

Random value functions

Cheap to simulate games from players that chose moves randomly:



Average outcomes of random games approximate very weak play.

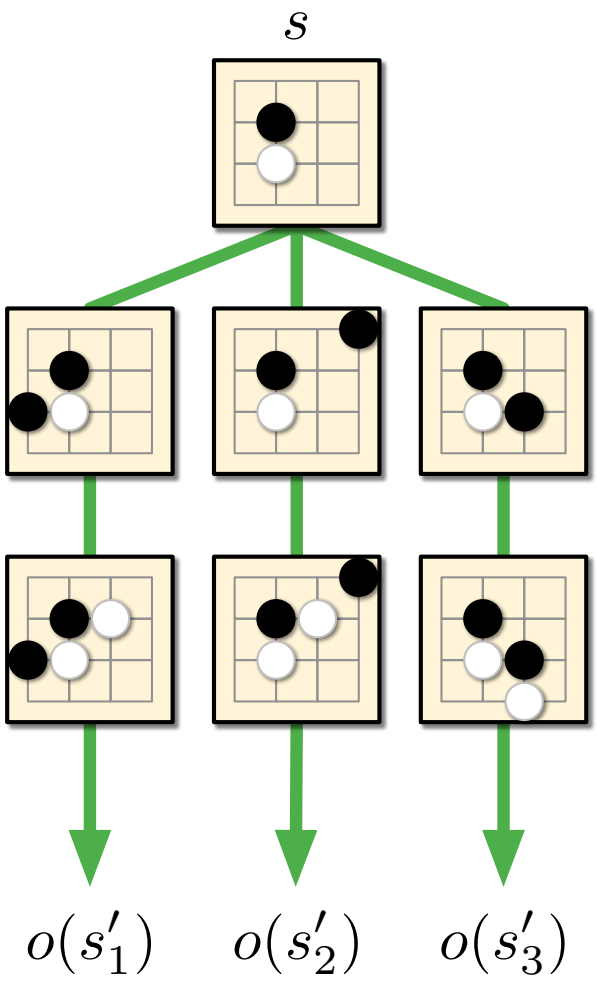
Random value functions

- Use an average of random evaluations v_n?

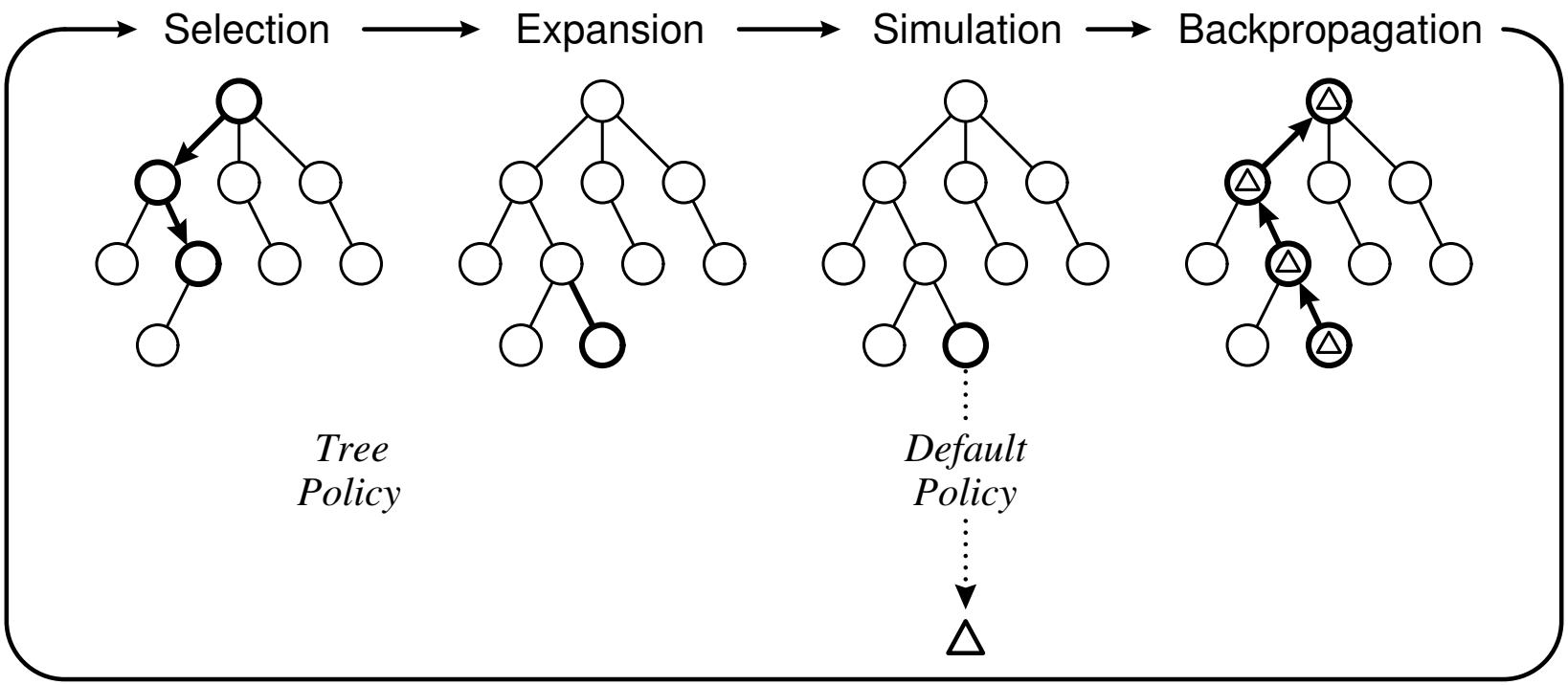
$$v_n(s) = \frac{1}{n} \sum_{i=1}^{n} o(s'_i)$$

  $o(s) = \pm 1$ if black wins / loses.

- Minimax with v_n won't work, because it is a pure exploitation strategy that assumes perfect leaf evaluations.
- How can we use v_n?



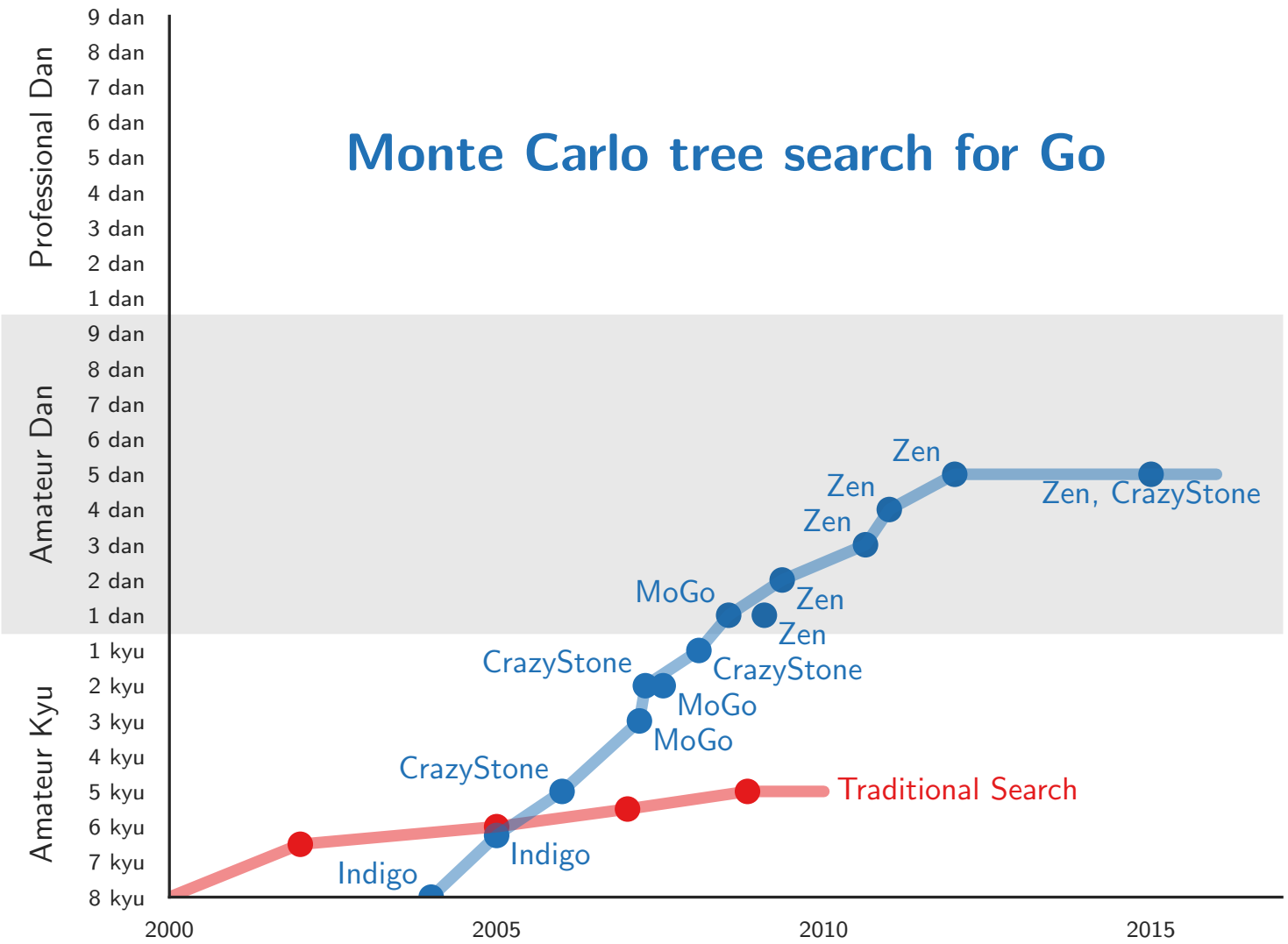
Monte Carlo tree search



(Browne et al., 2012)

- **Key idea**: pair v_n simulations with a selection strategy the biases the tree search towards high value regions.
- This is known as Monte Carlo tree search (MCTS; Kocsis and Szepesvári, 2006; Coulom, 2006).

Progress in Computer Go



adapted from Sylvain Gelly & David Silver, Test of Time Award ICML 2017

Scaling with compute and time

- The **strength of MCTS bots scales with the amount of compute and time at play-time**.
  - It is not limited by our ability to design evaluations.
- But play-time is limited. Can we improve computer Go players using compute when we are not playing? **Machine learning!**
  - You can try to think harder during a test vs. studying more beforehand.

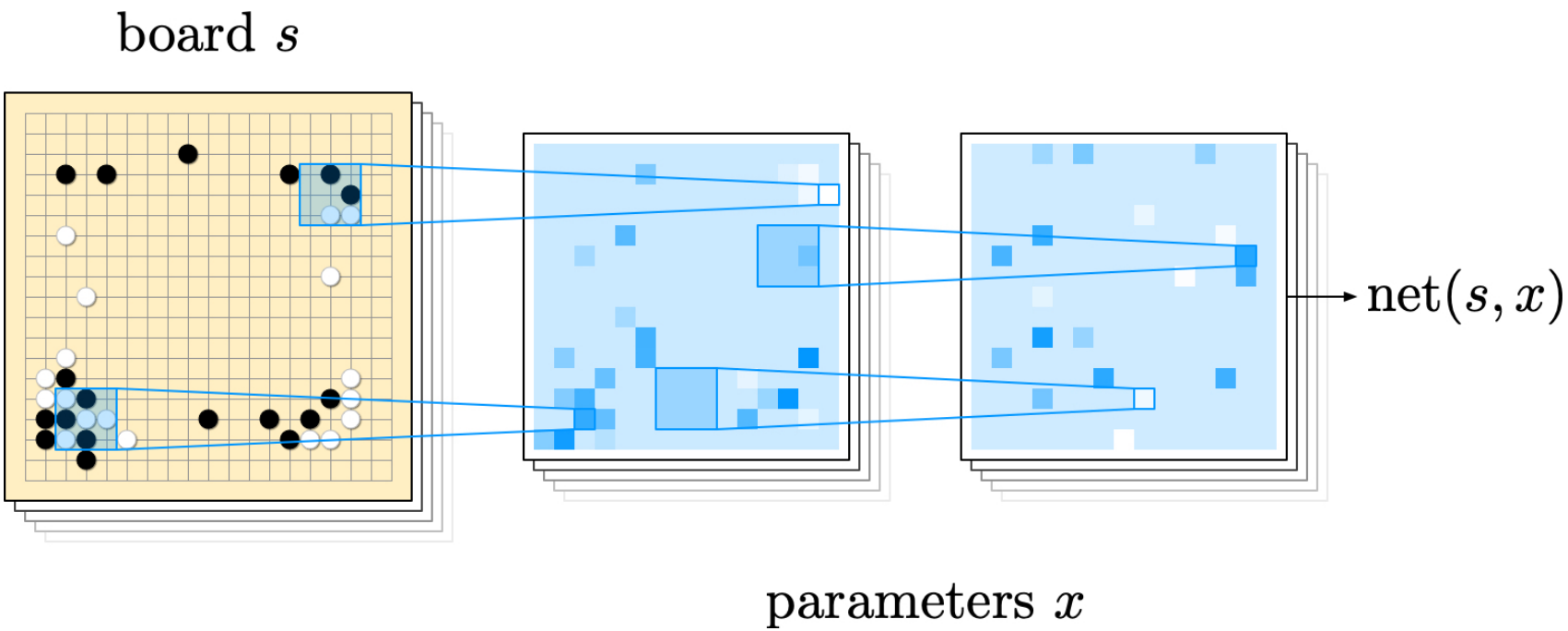
Learning to play Go

This is where I come in

- 2014 Google DeepMind internship on neural nets for Go.
  - Working with Aja Huang, David Silver, Ilya Sutskever, I was responsible for designing and training the neural networks.
  - Others came before (e.g., Sutskever and Nair, 2008).
- Ilya Sutskever's (Chief Scientist, OpenAI) argument in 2014: expert players can identify a good set of moves in 500 ms.
  - This is only enough time for the visual cortex to process the board—not enough for complex reasoning.
  - At the time we had neural networks that were nearly as good as humans in image recognition, thus we thought we would be able to train a net to play Go well.
- **Key goal: can we train a net to understand Go?**

Neural nets for Go

Neural networks can approximate very complex functions.

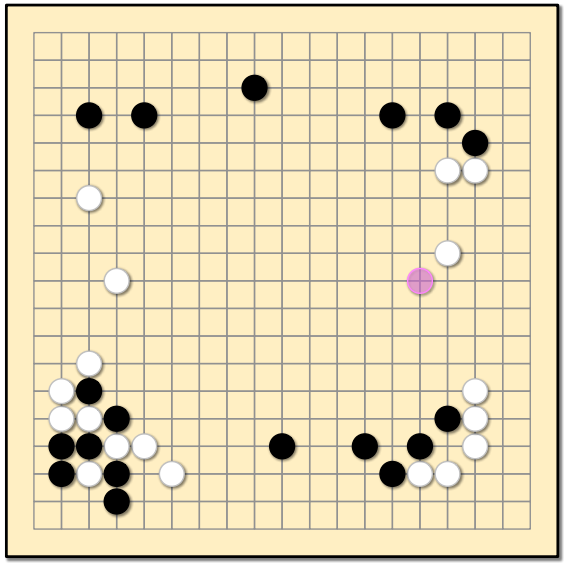


Idea: map board position s (input) to a next move or an evaluation (output) using simple convolutional networks.

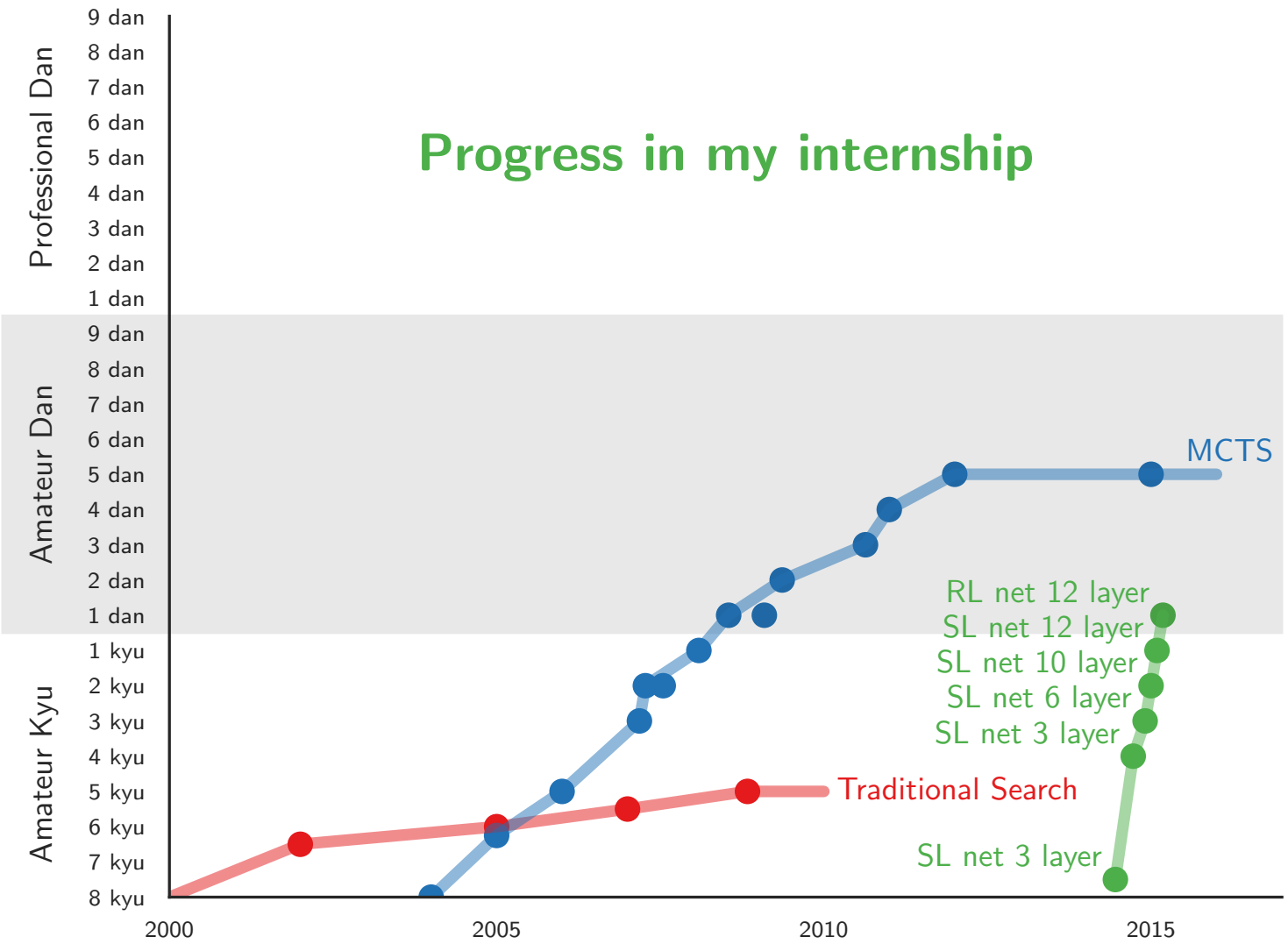
Neural nets for Go

- I spent about 4 months failing to produce results.
- I gave up on my clever ideas, and tried the simplest idea we had: **training a neural network to approximate an expert's next move**.
  - Input: board position s
  - Output: next move a
  - Data: games played by expert players on a Go server
- Networks that I trained made incredible progress.

An expert move (pink)



Progress in Computer Go

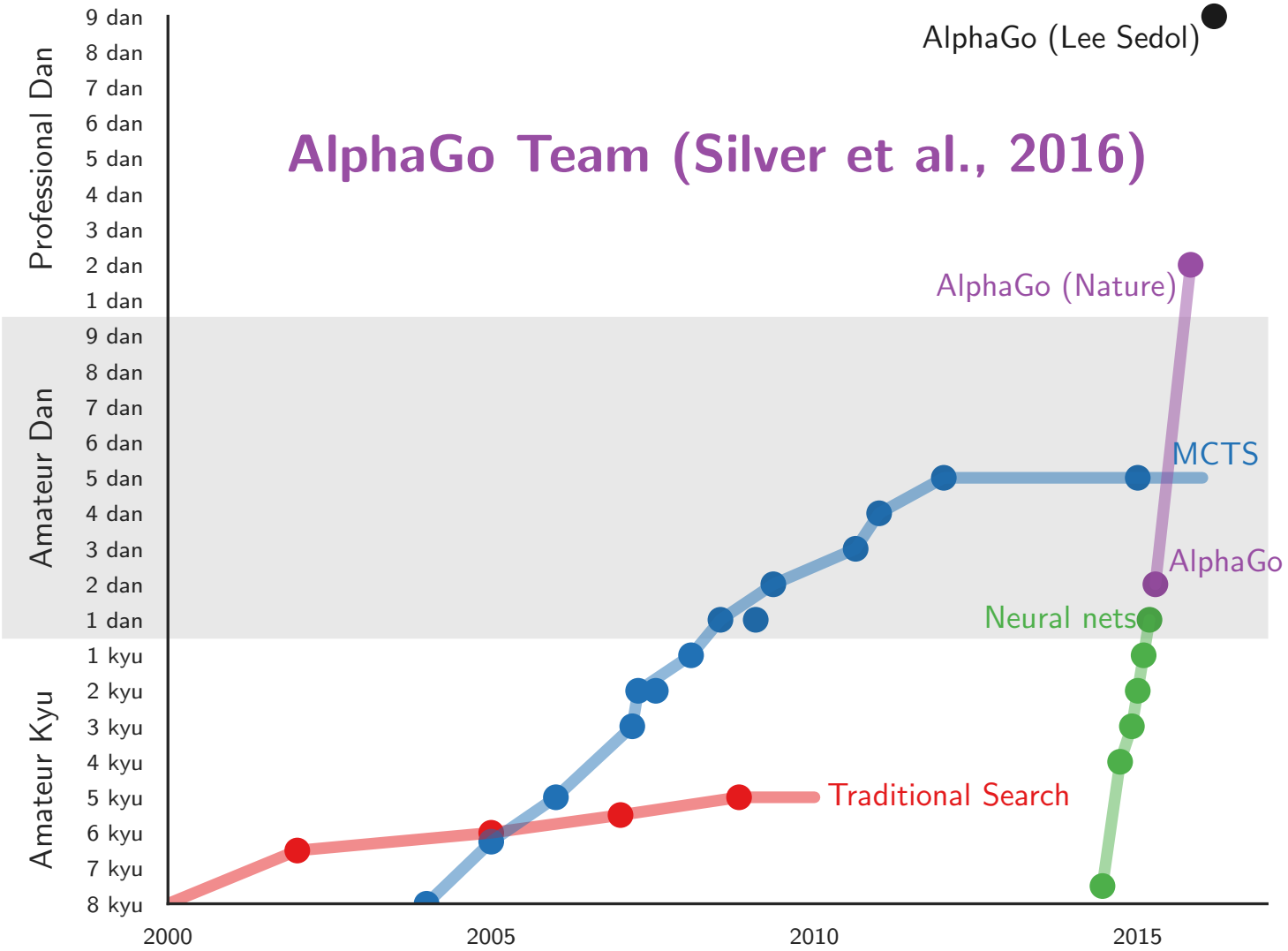


adapted from Sylvain Gelly & David Silver, Test of Time Award ICML 2017

AlphaGo (Silver et al., 2016)

- These results prompted the formation of big team inside DeepMind to combine MCTS and neural networks.
- The final version of AlphaGo combined two different kinds of neural networks and Monte Carlo tree search:
  - Move-prediction neural networks.
  - Value-prediction neural networks.
  - Random playouts.

Progress in Computer Go



adapted from Sylvain Gelly & David Silver, Test of Time Award ICML 2017

Impact

Go is not just a game

- Go originated in China more than 2,500 years ago. Reached Korea in the 5th century, Japan in the 7th.
- In the Tang Dynasty, it was one of the four arts of the Chinese scholar together with calligraphy, painting, and music.
- The aesthetics of Go (harmony, balance, style) are as essential to top-level play as basic tactics.

2016 Match—AlphaGo vs. Lee Sedol

- Best of 5 matches over the course of a week.
- Most people expected AlphaGo to lose 0-5.
- AlphaGo won 4-1.

Human moments

Lee Sedol is a titan in the Go world, and achieving his level of play requires a life of extreme dedication.

It was humbling and strange to be a part of the AlphaGo team that played against him.

Thoughts on research

- Research is a very different skill to coursework.
  - Coursework is about learning foundational ideas and solving well-defined problems.
  - Research is about developing poorly-explored ideas and sometimes defining new problems.
- Research is hard.
  - Most ideas don't work. I cope with this by letting myself enjoy the process of discovery and exploration, without worrying about the end result.
  - Cutting edge knowledge is not compressed into textbooks.
- If you're interested in research, it's important to start early and with people that think like you do. It is important to develop your taste in research problems, and this comes from the people you surround yourself with.

More practically

- How can I get started with research?
  - NSERC USRA and UTEA summer research.
  - CSC494 / CSC495 project courses.
- OK, but how do I get a professor to answer my emails? Here is what I suggest.
  - Look for a professor's publications in which they are the first or last author (this is important).
  - Read a couple of those papers, and implement a toy version of one of them.
  - Think about the shortcomings of the work, and propose a way to address those.
  - Implement your fix, and then email the prof. with a *short* email explaining your idea.
- I can't guarantee this will work, but you will stand out.

Thanks!

I played a key role at the start of AlphaGo, but the success is owed to a large and extremely talented team of scientists and engineers.

- David Silver, Aja Huang, **C**, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel & Demis Hassabis.

C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

M. Müller. Computer go. *Artificial Intelligence*, 134(1-2):145–179, 2002.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484 – 489, 2016.

I. Sutskever and V. Nair. Mimicking go experts with convolutional neural networks. In *International Conference on Artificial Neural Networks*, pages 101–110. Springer, 2008.