Computer Go: from the Beginnings to AlphaGo

Martin Müller, University of Alberta

2017

Outline of the Talk

- Game of Go
- Short history - Computer Go from the beginnings to AlphaGo
- The science behind AlphaGo
- The legacy of AlphaGo

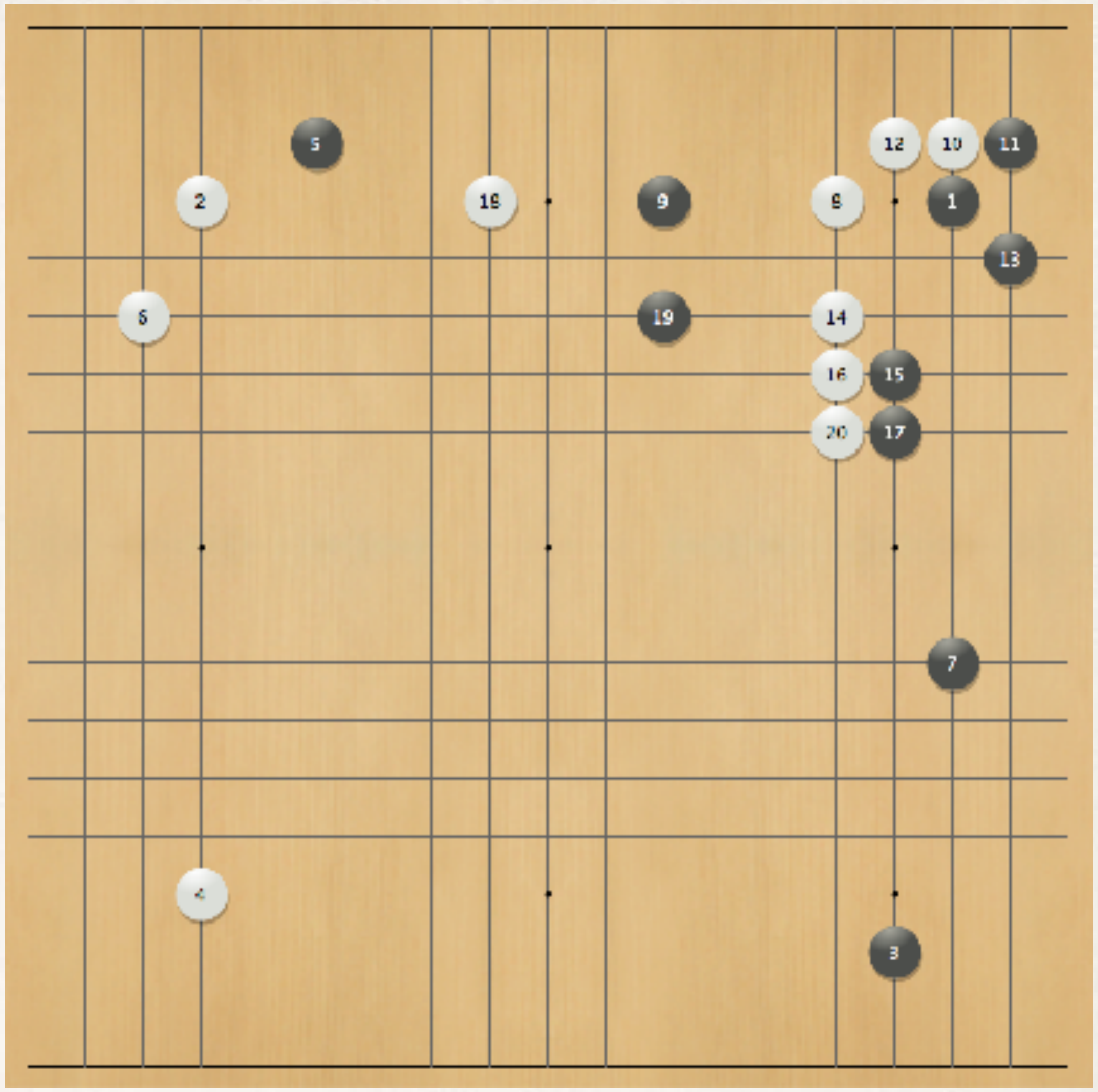
The Game of Go

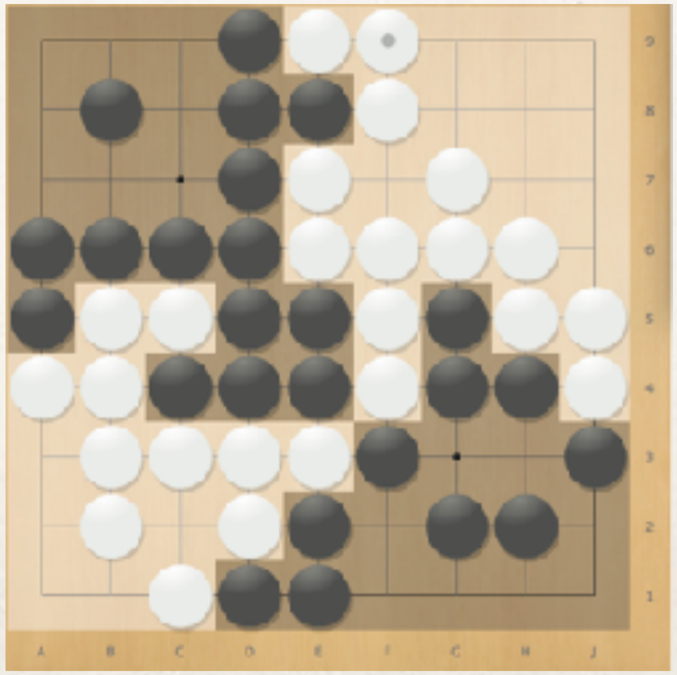
Go

- Classic two-player board game
- Invented in China thousands of years ago
- Simple rules, complex strategy
- Played by millions
- Hundreds of top experts - professional players
- Until 2016, computers weaker than humans

Go Rules

- Start with empty board
- Place stone of your own color
- Goal: surround empty points or opponent - capture
- Win: control more than half the board
- Komi: first player advantage





Final score, 9x9 board

Measuring Go Strength

- People in Europe and America use the traditional Japanese ranking system
- Kyu (student) and Dan (master) levels
  - Separate Dan ranks for professional players
- Kyu grades go down from 30 (absolute beginner) to 1 (best)
- Dan grades go up from 1 (weakest) to about 6
- There is also a numerical (Elo) system, e.g. 2500 = 5 Dan

Short History of Computer Go

Computer Go History - Beginnings

- 1960's: initial ideas, designs on paper
- 1970's: first serious program - Reitman & Wilcox
  - Interviews with strong human players
  - Try to build a model of human decision-making
  - Level: "advanced beginner", 15-20 kyu
  - One game costs thousands of dollars in computer time

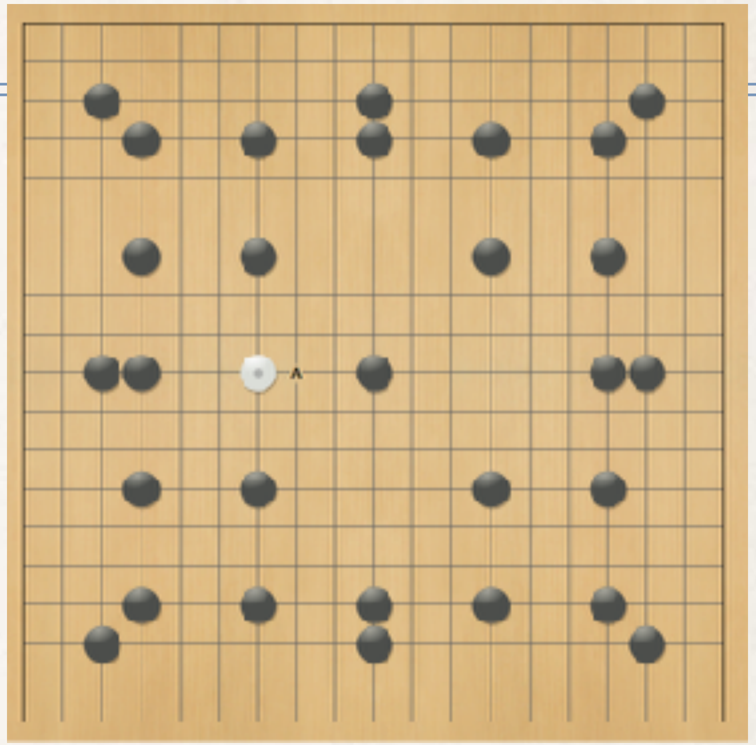
1980-89 The Arrival of PC

- From 1980: PC (personal computers) arrive
- Many people get cheap access to computers
- Many start writing Go programs
- First competitions, Computer Olympiad, Ing Cup
- Level 10-15 kyu

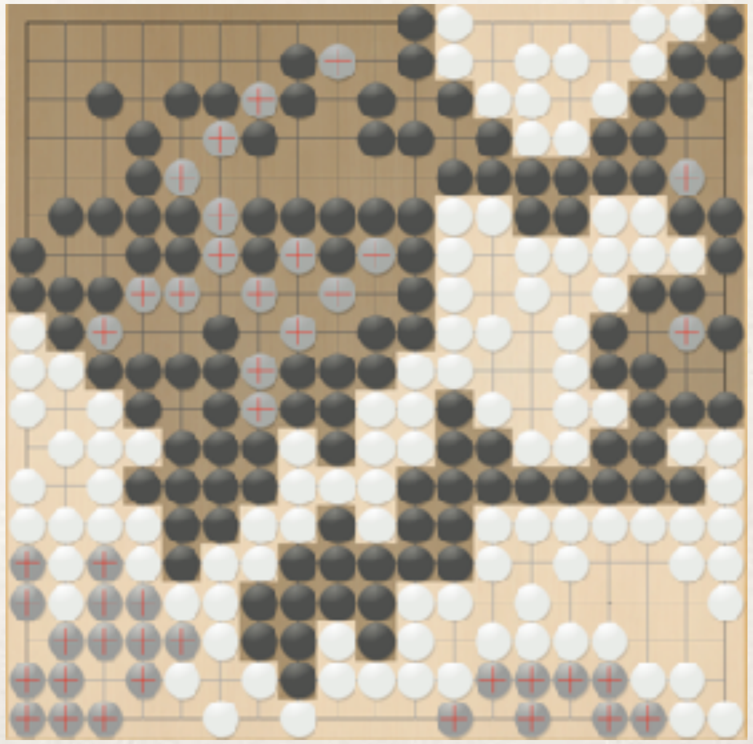
1990-2005: Slow Progress

- Slow progress, commercial successes
- 1990 Ing Cup in Beijing
- 1993 Ing Cup in Chengdu
- Top programs Handtalk (Prof. Chen Zhixing), Goliath (Mark Boon), Go++ (Michael Reiss), Many Faces of Go (David Fotland)
- GNU Go - open source program, almost equal to top commercial programs
- Level - maybe 5 Kyu, but some "blind spots"

1998 - 29 Stone Handicap Game

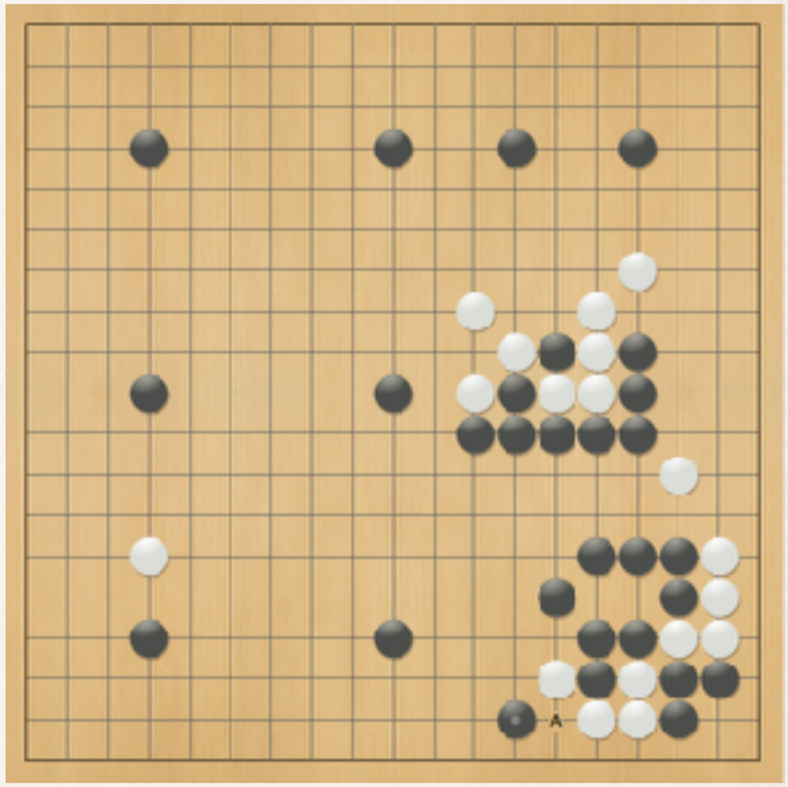


- Played at US Go Congress
- Black: Many Faces of Go, world champion and one of the top Go programs at the time
- White: Martin Müller, 5 Dan amateur
- Result: White won by 6 points



2006-08 Monte Carlo Revolution

- Remi Coulom, Crazy Stone program: Monte Carlo Tree Search (MCTS)
- Levente Kocsis and Csaba Szepesvari: UCT algorithm
- Sylvain Gelly, Olivier Teytaud et al: MoGo program
- Level: about 1 Dan



Search - Game Tree Search

- All possible move sequences
- Combined in a tree structure
- Root is the current game position
- Leaf node is end of game
- Search used to find good move sequences
- Minimax principle

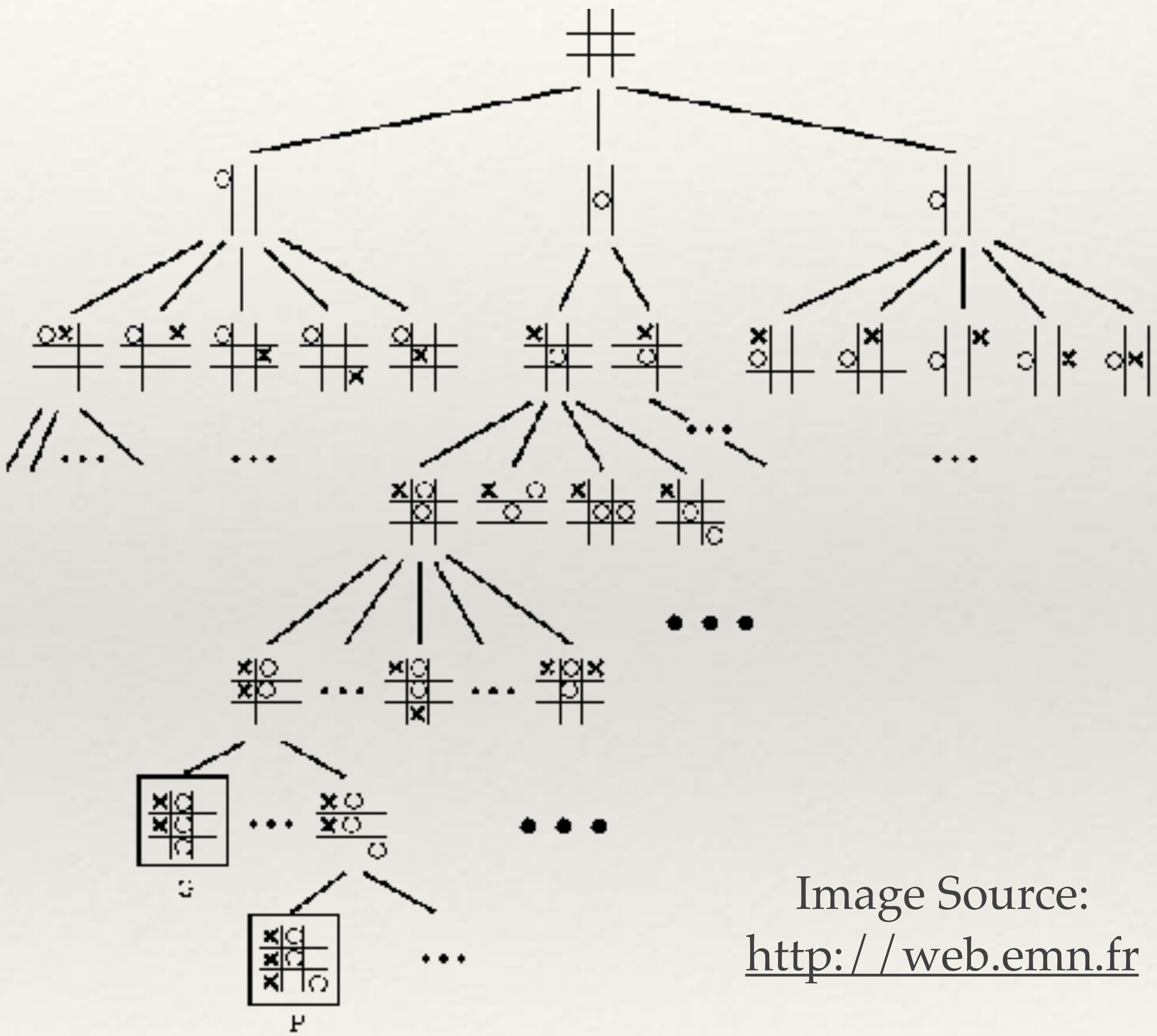
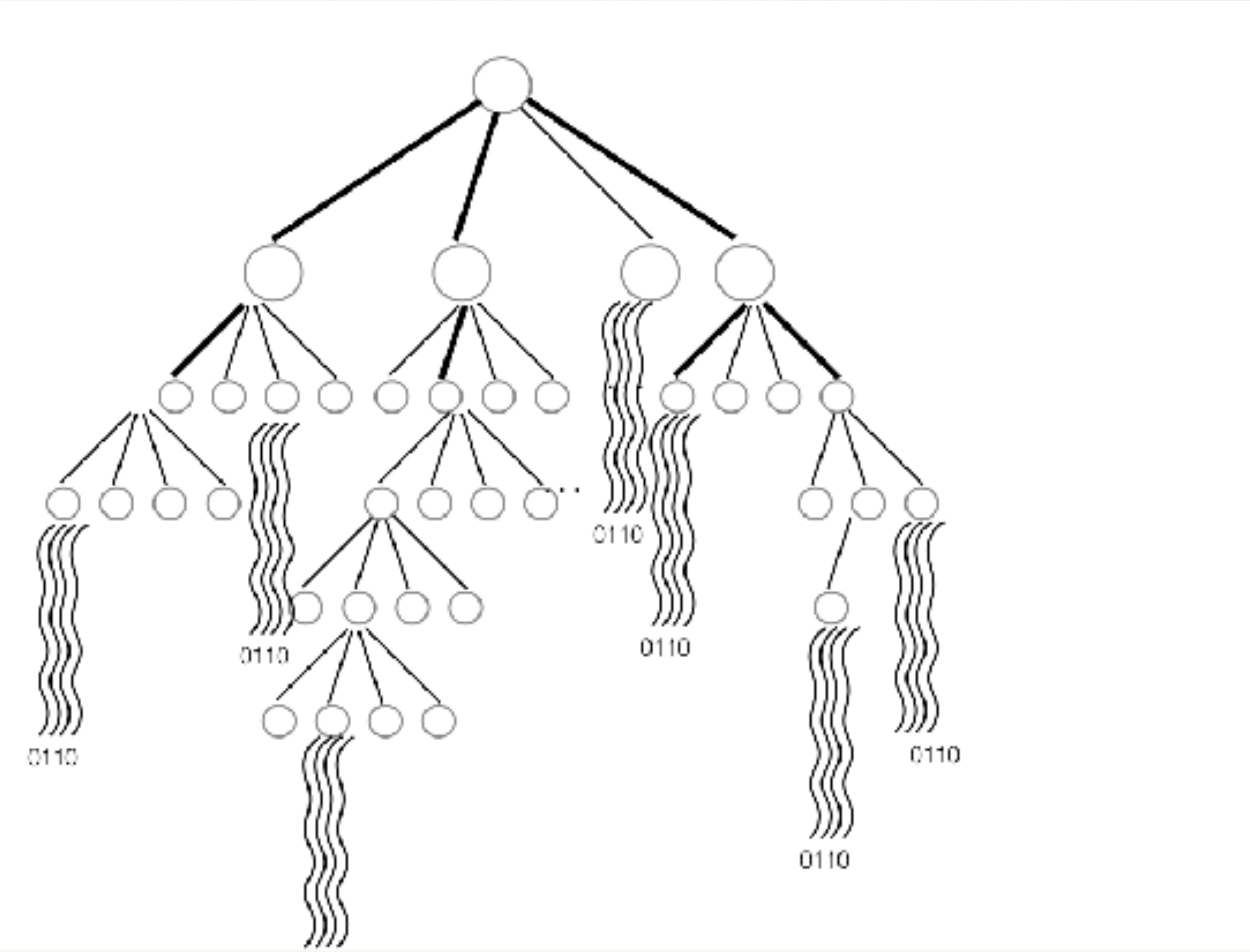


Image Source: http://web.emn.fr

Search - Monte Carlo Tree Search

- Invented about 10 years ago (Coulom - Crazystone, UCT)
- Grow tree using win/loss statistics of simulations
- First successful use of simulations for classical two-player games
- Scaled up to massively parallel
  - MoGo; Fuego on several thousand cores



Simulation

- For complex problems, there are far too many possible future states
- Example:
  predict the path of a storm
- Sometimes, there is no good evaluation
- We can sample long-term consequences by simulating many future trajectories

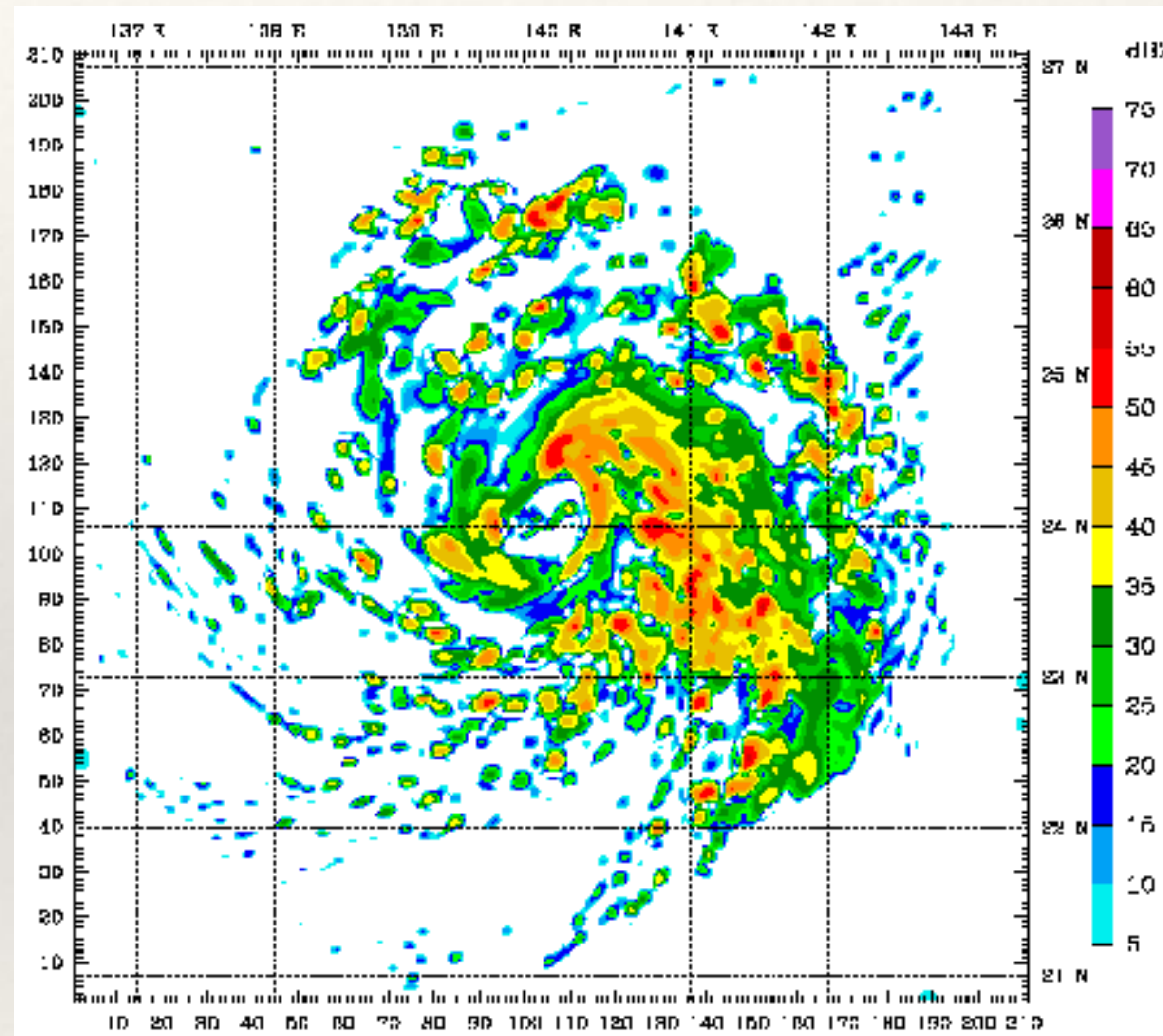
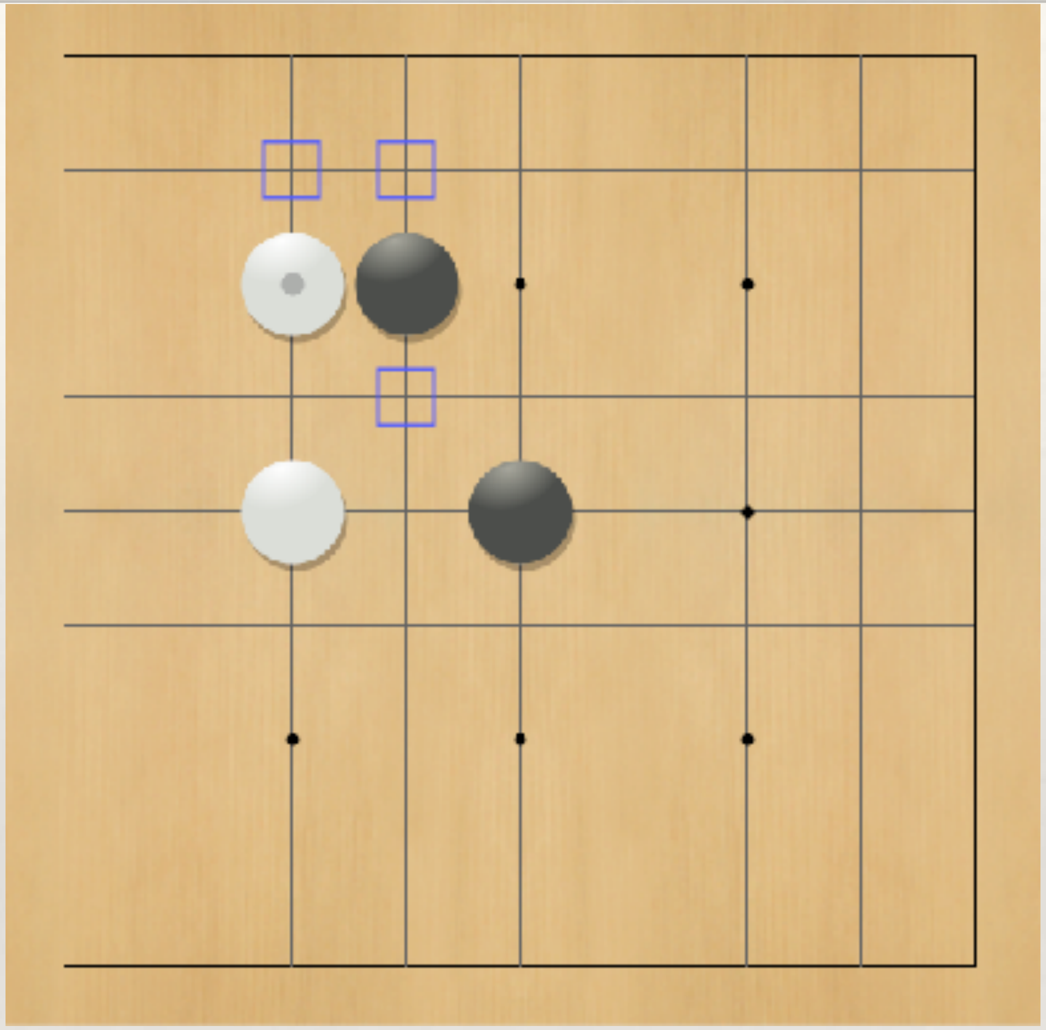


Image Source:
https://upload.wikimedia.org

Simulation in Computer Go

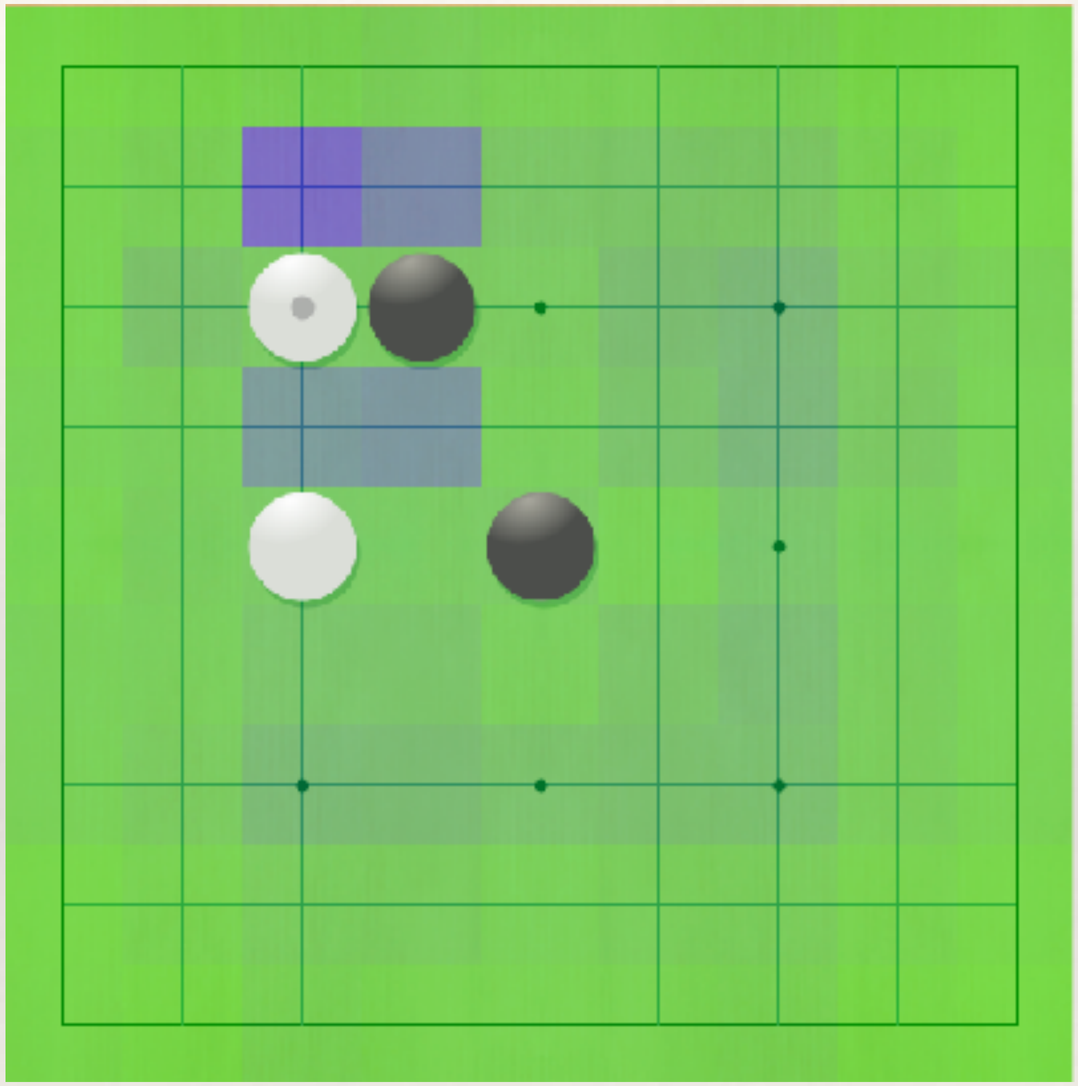
- Play until end of game
- Find who wins at end (easy)
- Moves in simulation: random + simple rules
- Early rules hand-made



Example:
Simple rule-based policy

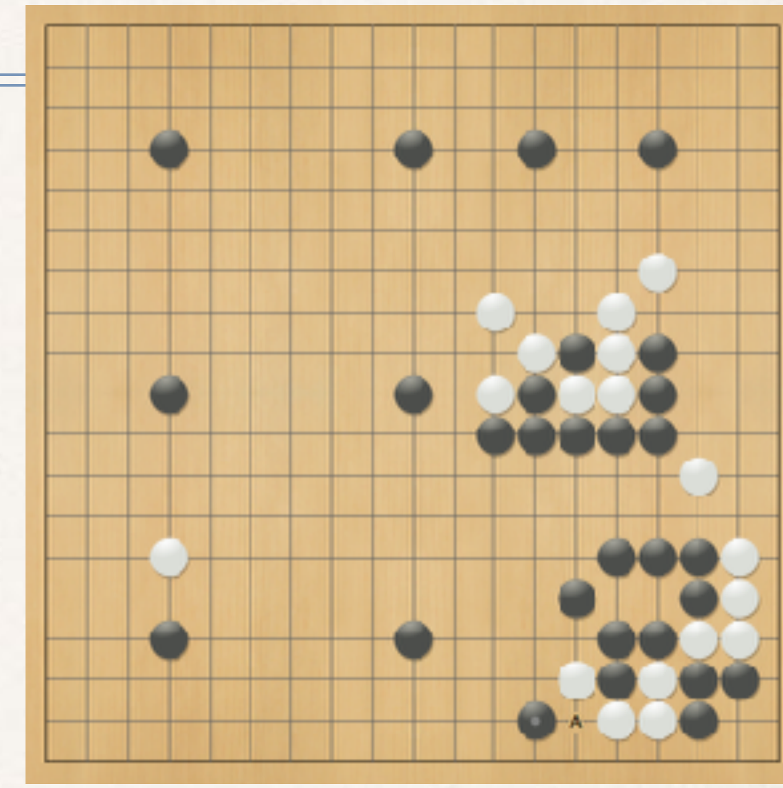
Simulation in Computer Go (2)

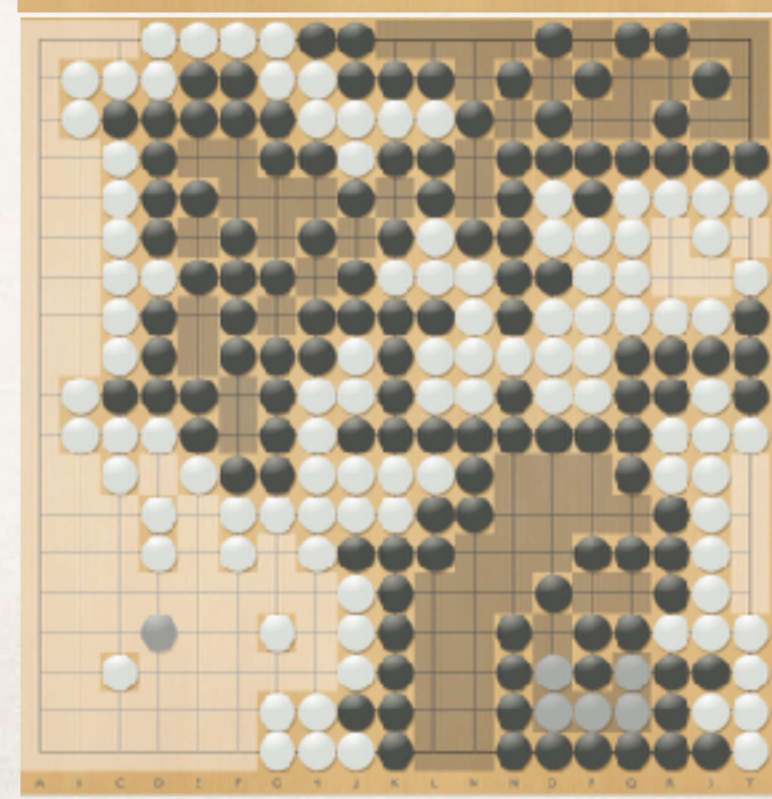
- Later improvement:
- Machine-learned policy based on simple features
- Probability for each move
- AlphaGo: machine-trained simple network
- Fast: goal is about 1,000,000 moves / second / CPU



2008 First win on 9 Stones

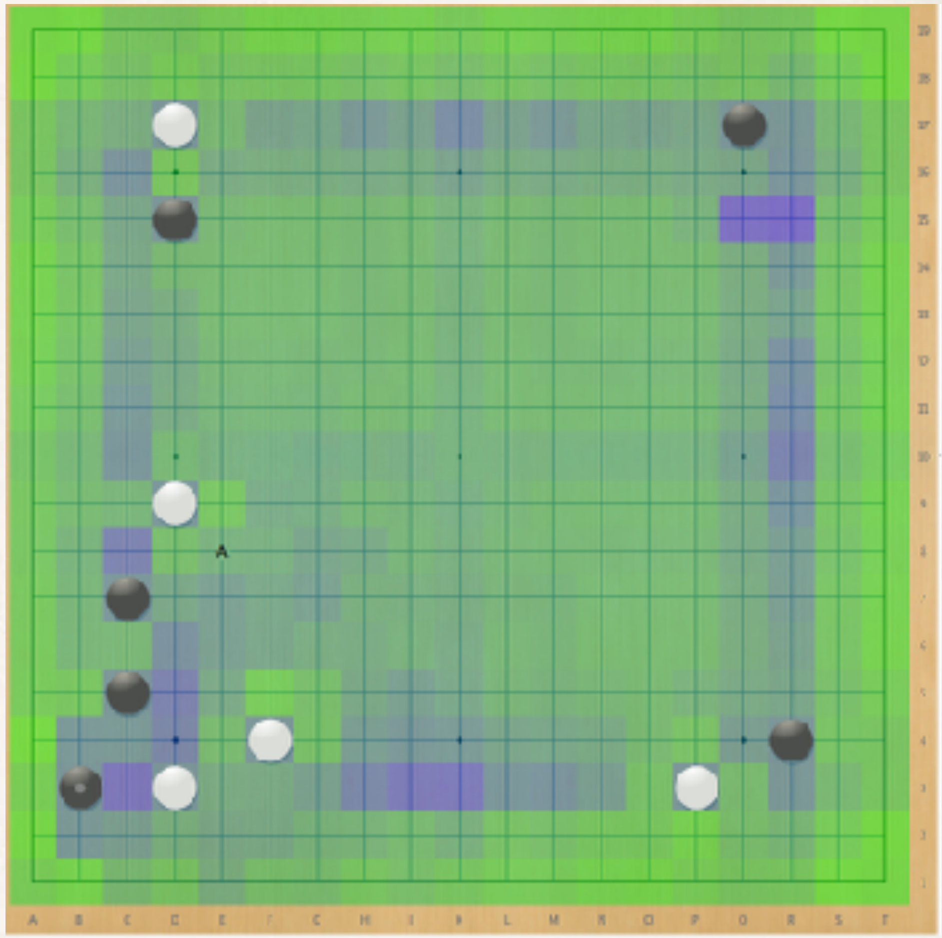
- MoGo program
- Used supercomputer with 3200 CPUs
- Won with 9 stones handicap vs Myungwan Kim, 8 Dan professional





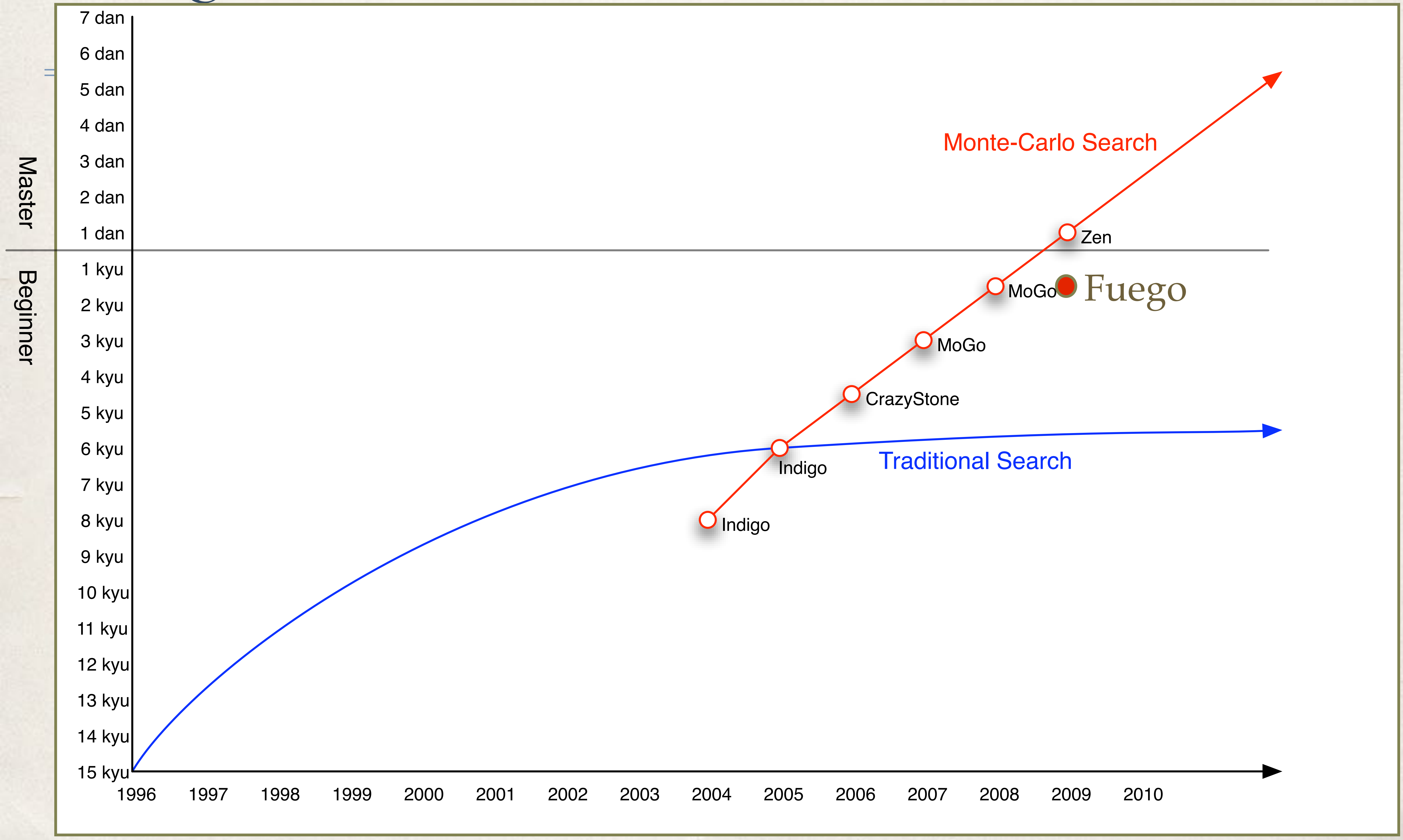
2008-15: Rapid Improvement

- Improve Monte Carlo Tree Search
- Better simulation policies
  (trial and error)
- Add Go knowledge in tree
  - Simple features, learn weights
    by machine learning
- Level: about 5-6 Dan
  3-4 stones handicap from top human players



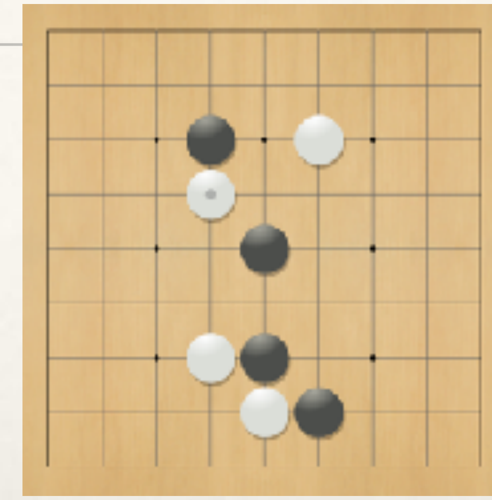
Knowledge based
on simple features
in Fuego

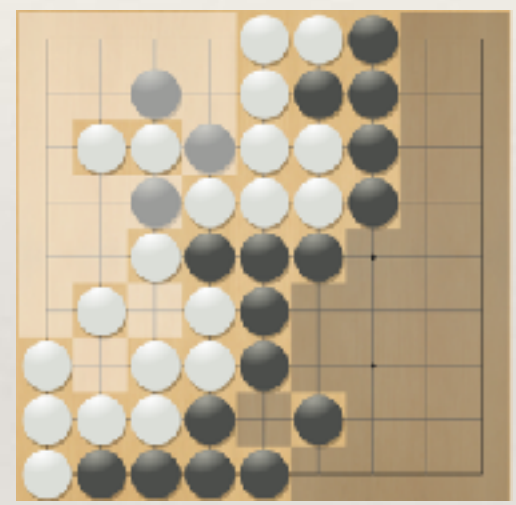
Progress In 19x19 Go, 1996-2010



2009 - First 9x9 Win vs Top Pro

- Fuego open source program
  - Mostly developed at University of Alberta
- First win against top human professional on 9x9 board
- MCTS, deep searches
- 80 core parallel machine

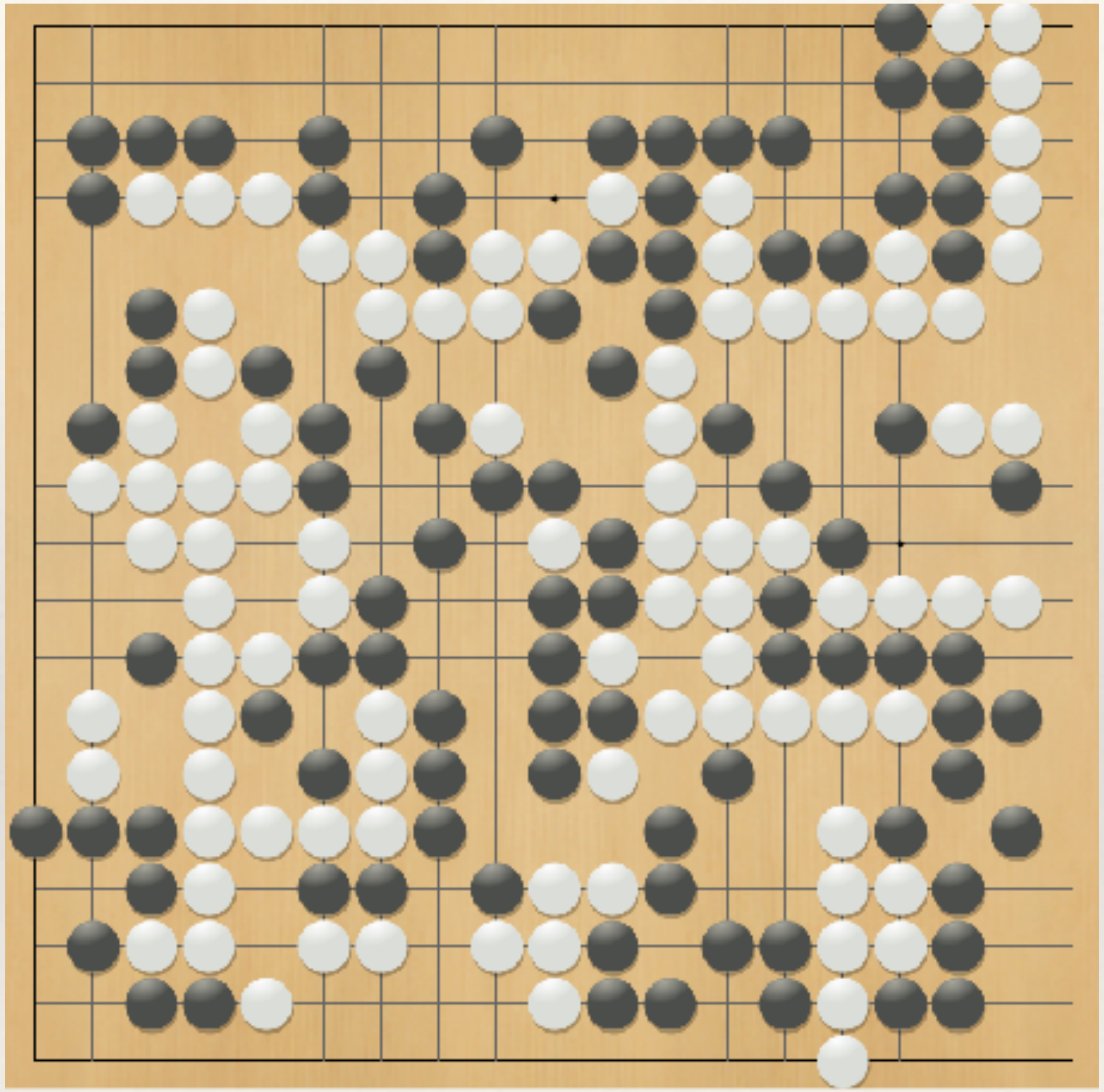




White: Fuego
Black: Chou Chun-Hsun 9 Dan
White wins by 2.5 points

Computer Go Before AlphaGo

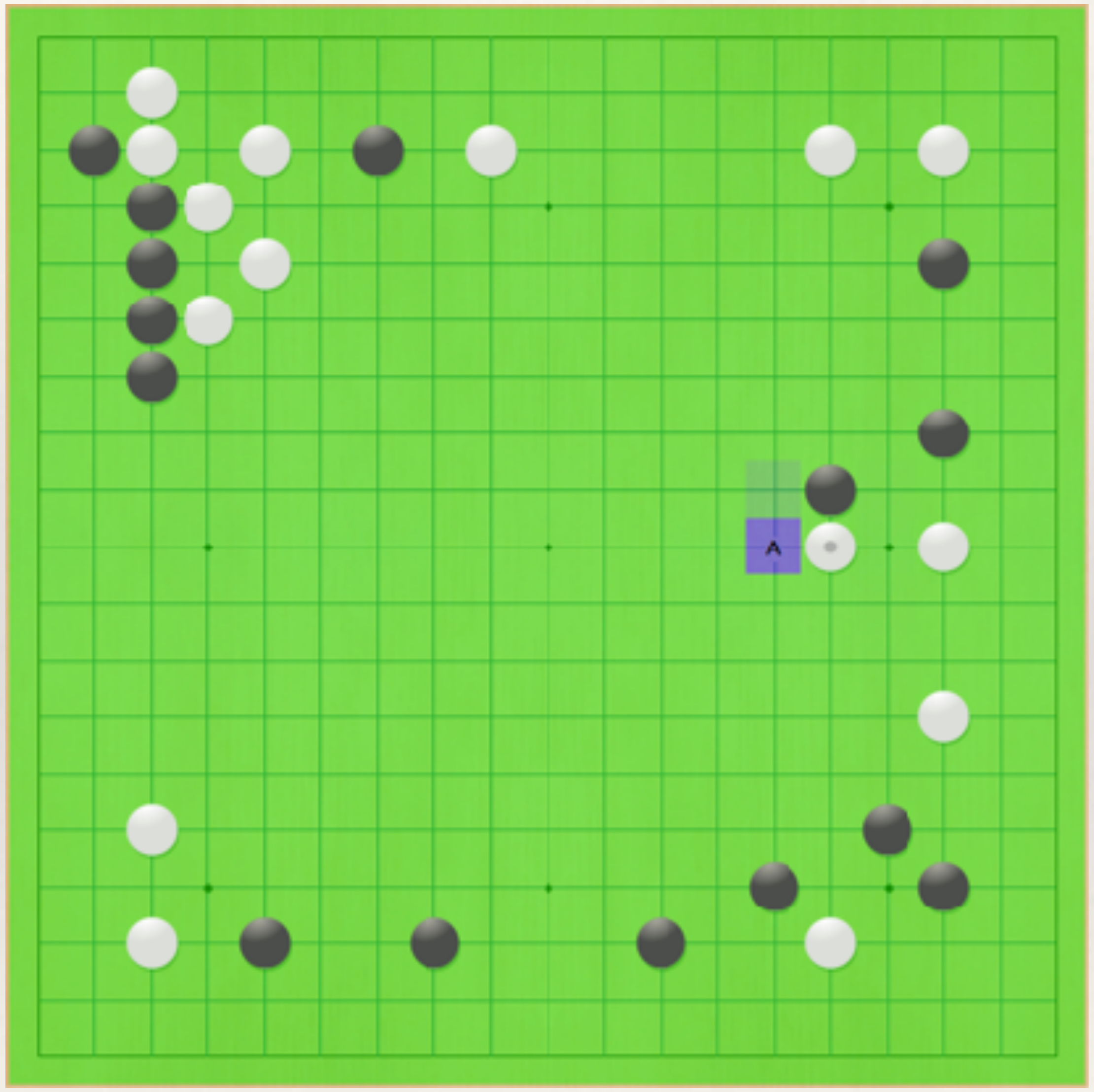
- Summary of state of the art before AlphaGo:
- Search - quite strong
- Simulations - OK, but hard to improve
- **Knowledge**
  - Good for move selection
  - **Considered hopeless for position evaluation**



Who is better here?

2015 - Deep Neural Nets Arrive

- Two papers within a few weeks
  - First by Clark and Storkey, University of Edinburgh
  - Second paper by group at DeepMind, stronger results
- Deep convolutional neural nets (DCNN) used for move prediction in Go
- Much better prediction than old feature-based systems



AlphaGo

- Program by DeepMind
- Based in London, UK **and Edmonton (from 2017)**
- Bought by Google
- Expertise in Reinforcement Learning and search
- 2014-16: worked on Go program for about 2 years, mostly in secret
- One paper on move prediction (previous slide)

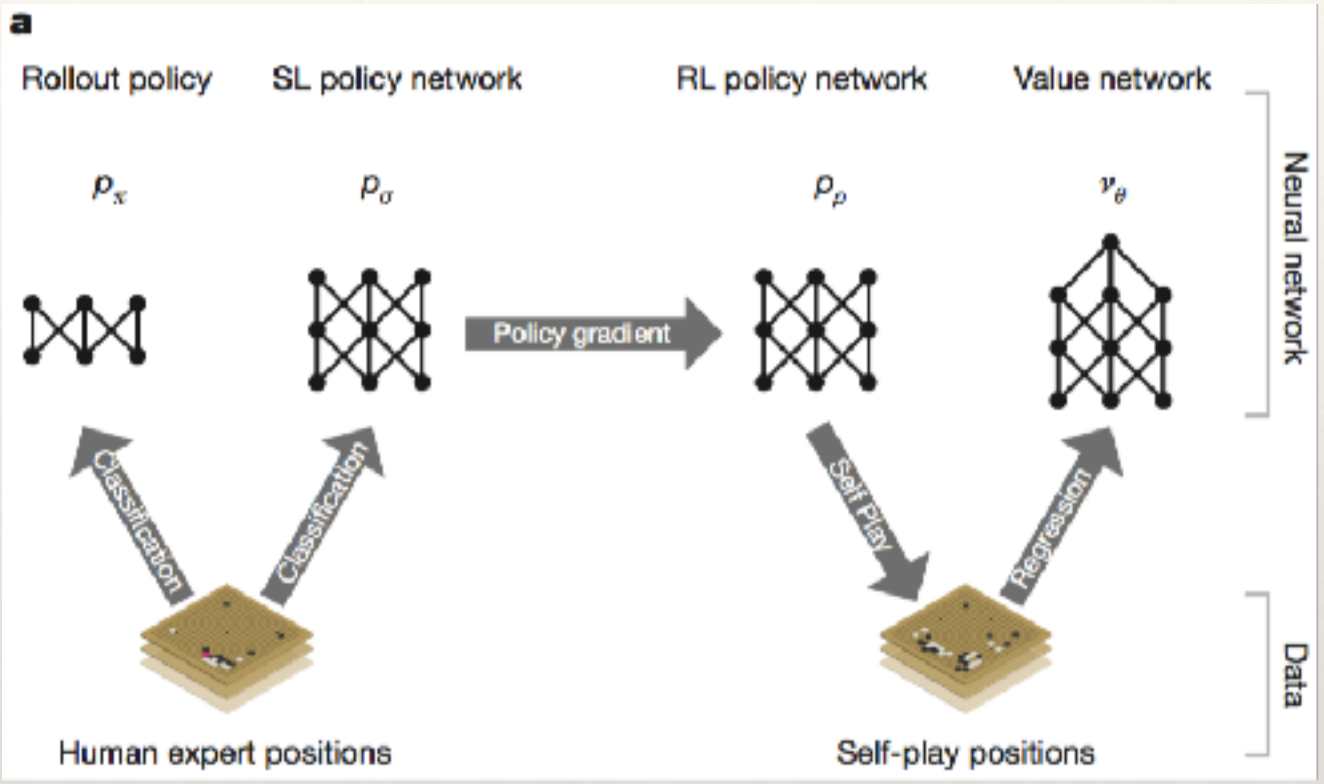
AlphaGo Matches

- Fall 2015 - beat European champion Fan Hui by 5:0 (kept secret)
- January 2016 paper in Nature, announced win vs Fan Hui
- March 2016 match vs Lee Sedol Wins 4:1
- January 2017, wins fast games 60:0 against many top players
- May 2017 match vs Ke Jie Wins 3:0 then retires

The Science Behind AlphaGo

The Science Behind AlphaGo

- AlphaGo builds on decades of research in:
  - Building high performance game playing programs
  - Reinforcement Learning
  - (Deep) neural networks

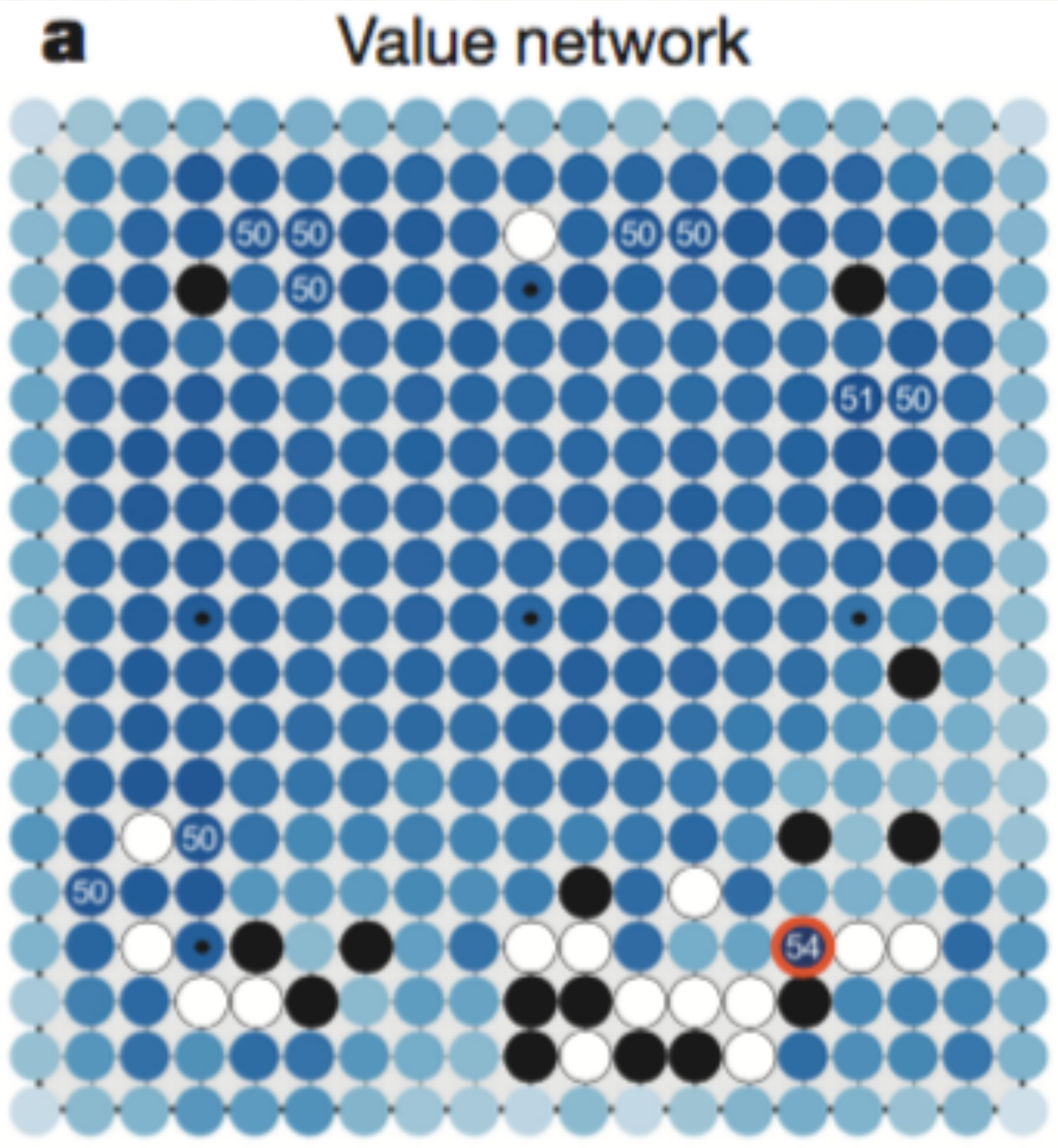


Main Components of AlphaGo

- AlphaGo shares the same main components with many other modern heuristic search programs:
  - Search - MCTS (normal)
  - Knowledge created by machine learning (**new types of knowledge**)
  - Simulations (normal)

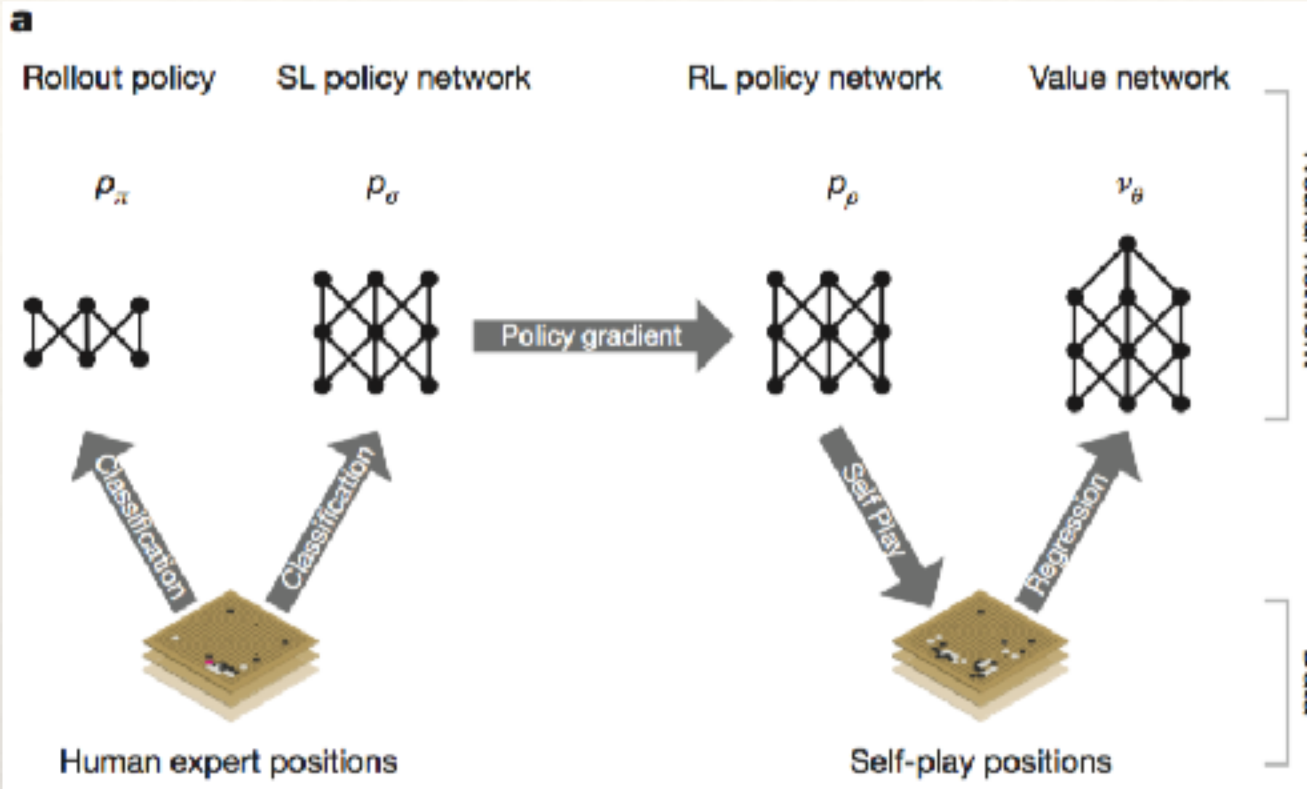
Knowledge - Policy and Evaluation

- Two types of knowledge
- Encoded in deep convolutional neural networks
- *Policy network*
  selects good moves for the search (as in move prediction)
- *Value network*:
  evaluation function,
  measures probability of winning



Deep Neural Networks in AlphaGo

- Three different deep neural networks
- Supervised Learning (SL) policy network as in 2015 paper
    - Learn from master games: improved in details, more data
- New: **Reinforcement Learning (RL)** from self-play for **policy network**
- New: **value network** trained from labeled data from self-play games



RL Policy Network

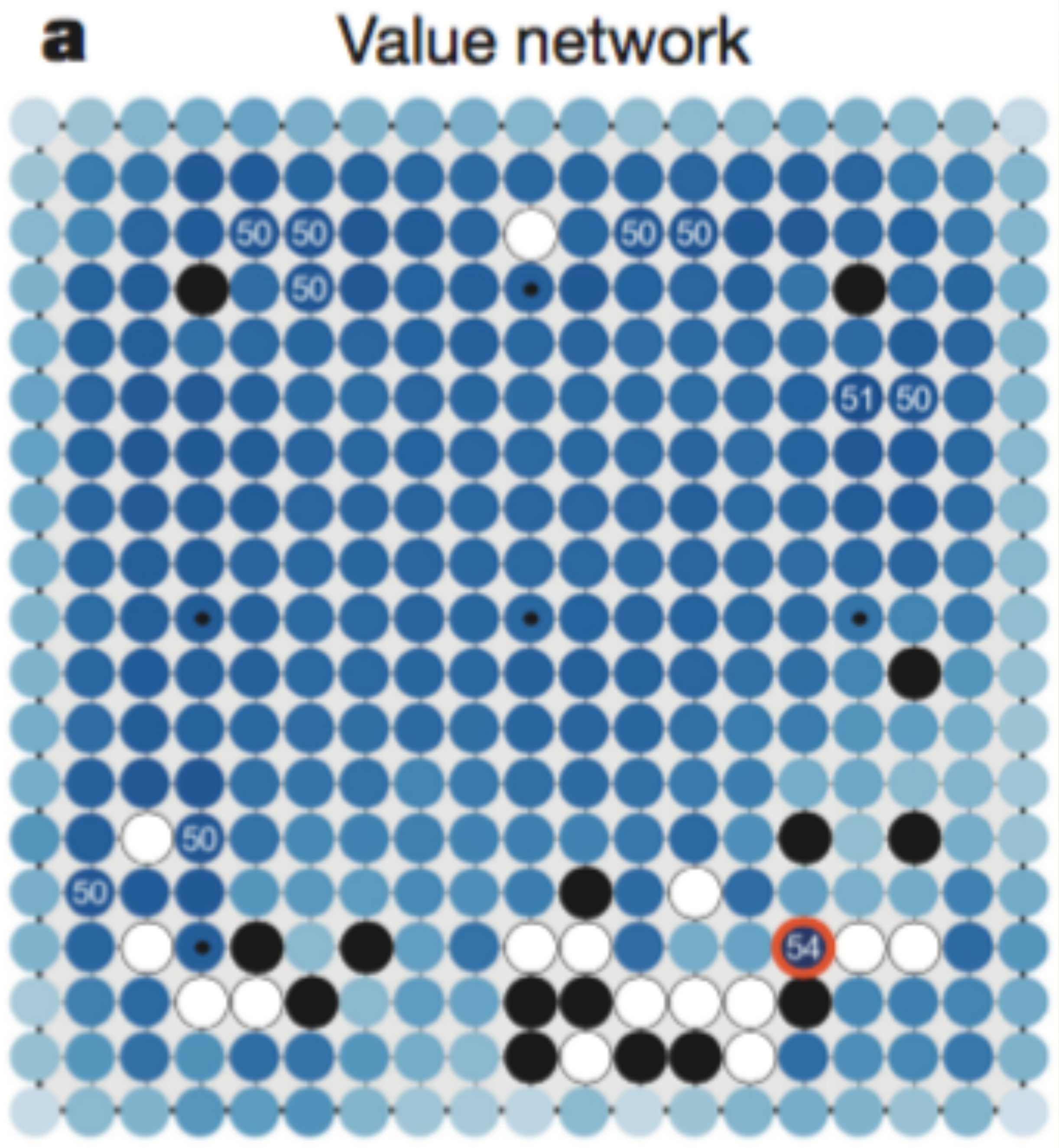
- Deep neural network, same architecture as SL network
- Given a Go position
- Computes probability of each move being best
- Initialized with SL policy weights
- Trained by Reinforcement Learning from millions of self-play games
- Adjust weights in network from win/loss result at end of game only

Data for Training Value Network

- Policy network can be used as a strong and relatively fast player
- Randomize moves according to their learned probability
- After training, played 30 million self-play games
- Pick a single position from each game randomly
- Label it with the win/loss result of the game
- Result: data set of 30 million Go positions, each labeled as win or loss
- Next step: train the value network on those positions

Value Network

- Another deep neural network
- Given a Go position
- Computes probability of winning
- Static evaluation function
- Trained from the 30 million labeled game positions
- Trained to minimize the prediction error on the (win/loss) labels



Putting it All Together

- A huge engineering effort
- Many other technical contributions
- Massive amounts of self-play training for the neural networks
- Massive amounts of testing / tuning
- Large parallel hardware in earlier matches
- “Single TPU machine” in 2017

What's New in AlphaGo 2017?

- Few details known as of now
- More publications promised
- Main change: better games data for training the value net
- Old system: 30 million games played by RL policy net
- New system: unknown number of games played by the full AlphaGo system

- Consequences:
  - Much better quality of games
  - Much better quality of final result labels
  - From strong amateur (RL network) to full AlphaGo strength
- Most likely,
  many other improvements
  in all parts of the system

The Legacy of AlphaGo

Legacy of AlphaGo

- Research contributions, the path leading to AlphaGo
- Impact on communities
  - Go players
  - Computer Go researchers
  - Computing science
  - General public

Review: Contributions to AlphaGo

- Deepmind developed AlphaGo, with many great breakthrough ideas
- AlphaGo is *also* based on decades of research in heuristic search and machine learning
- Much of that research was done at University of Alberta
- Next slide: references from AlphaGo paper in Nature
    - Over 40% of references have a University of Alberta (co-)author

U. Alberta Research and Training

- Citation list from AlphaGo paper in Nature
- Papers with Alberta faculty or trainees in yellow

1. Allis, L. V. *Searching for Solutions in Games and Artificial Intelligence*. PhD thesis, Univ. Limburg, Maastricht, The Netherlands (1994).
2. van den Herik, H., Uiterwijk, J. W. & van Rijswijck, J. Games solved: now and in the future. *Artif. Intell.* **134,** 277–311 (2002).
3. Schaeffer, J. The games computers (and people) play. *Advances in Computers* **52,** 189–266 (2000).
4. Campbell, M., Hoane, A. & Hsu, F. Deep Blue. *Artif. Intell.* **134,** 57–83 (2002).
5. Schaeffer, J. *et al.* A world championship caliber checkers program. *Artif. Intell.* **53,** 273–289 (1992).
6. Buro, M. From simple features to sophisticated evaluation functions. In *1st International Conference on Computers and Games*, 126–145 (1999).
7. Müller, M. Computer Go. *Artif. Intell.* **134,** 145–179 (2002).
8. Tesauro, G. & Galperin, G. On-line policy improvement using Monte-Carlo search. In *Advances in Neural Information Processing*, 1068–1074 (1996).
9. Sheppard, B. World-championship-caliber Scrabble. *Artif. Intell.* **134,** 241–275 (2002).
10. Bouzy, B. & Helmstetter, B. Monte-Carlo Go developments. In *10th Internationa Conference on Advances in Computer Games*, 159–174 (2003).
11. Coulom, R. Efficient selectivity and backup operators in Monte-Carlo tree search. In *5th International Conference on Computers and Games*, 72–83 (2006).
12. Kocsis, L. & Szepesvári, C. Bandit based Monte-Carlo planning. In *15th European Conference on Machine Learning*, 282–293 (2006).
13. Coulom, R. Computing Elo ratings of move patterns in the game of Go. *ICGA J.* **30,** 198–208 (2007).
14. Baudiš, P. & Gailly, J.-L. Pachi: State of the art open source Go program. In *Advances in Computer Games*, 24–38 (Springer, 2012).
15. Müller, M., Enzenberger, M., Arneson, B. & Segal, R. Fuego – an open-source framework for board games and Go engine based on Monte-Carlo tree search. *IEEE Trans. Comput. Intell. AI in Games* **2,** 259–270 (2010).
16. Gelly, S. & Silver, D. Combining online and offline learning in UCT. In *17th International Conference on Machine Learning*, 273–280 (2007).
17. Krizhevsky, A., Sutskever, I. & Hinton, G. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1097–1105 (2012).
18. Lawrence, S., Giles, C. L., Tsoi, A. C. & Back, A. D. Face recognition: a convolutional neural-network approach. *IEEE Trans. Neural Netw.* **8,** 98–113 (1997).
19. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature* **518,** 529–533 (2015).
20. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521,** 436–444 (2015
21. Stern, D., Herbrich, R. & Graepel, T. Bayesian pattern ranking for move prediction in the game of Go. In *International Conference of Machine Learning* 873–880 (2006).
22. Sutskever, I. & Nair, V. Mimicking Go experts with convolutional neural networks. In *International Conference on Artificial Neural Networks*, 101–110 (2008).
23. Maddison, C. J., Huang, A., Sutskever, I. & Silver, D. Move evaluation in Go us deep convolutional neural networks. *3rd International Conference on Learnin Representations* (2015).
24. Clark, C. & Storkey, A. J. Training deep convolutional neural networks to play go. In *32nd International Conference on Machine Learning*, 1766–1774 (2015).
25. Williams, R. J. Simple statistical gradient-following algorithms for connectior reinforcement learning. *Mach. Learn.* **8,** 229–256 (1992).
26. Sutton, R., McAllester, D., Singh, S. & Mansour, Y. Policy gradient methods fc reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 1057–1063 (2000).
27. Sutton, R. & Barto, A. *Reinforcement Learning: an Introduction* (MIT Press, 199
28. Schraudolph, N. N., Dayan, P. & Sejnowski, T. J. Temporal difference learning of position evaluation in the game of Go. *Adv. Neural Inf. Process. Syst.* **6,** 817–824 (1994).
29. Enzenberger, M. Evaluation in Go by a neural network using soft segmentati In *10th Advances in Computer Games Conference*, 97–108 (2003). 267.
30. Silver, D., Sutton, R. & Müller, M. Temporal-difference search in computer Gc *Mach. Learn.* **87,** 183–219 (2012).
31. Levinovitz, A. The mystery of Go, the ancient game that computers still can't win. *Wired Magazine* (2014).
32. Mechner, D. All Systems Go. *The Sciences* **38,** 32–37 (1998).
33. Mandziuk, J. Computational intelligence in mind games. In *Challenges for Computational Intelligence*, 407–442 (2007).
34. Berliner, H. A chronology of computer chess and its literature. *Artif. Intell.* **10** 201–214 (1978).
35. Browne, C. *et al.* A survey of Monte-Carlo tree search methods. *IEEE Trans. Comput. Intell. AI in Games* **4,** 1–43 (2012).
36. Gelly, S. *et al.* The grand challenge of computer Go: Monte Carlo tree search and extensions. *Commun. ACM* **55,** 106–113 (2012).
37. Coulom, R. Whole-history rating: A Bayesian rating system for players of time-varying strength. In *International Conference on Computers and Games*, 113–124 (2008).
38. KGS. Rating system math. http://www.gokgs.com/help/rmath.html.
39. Littman, M. L. Markov games as a framework for multi-agent reinforcem learning. In *11th International Conference on Machine Learning*, 157–163 (1994).
40. Knuth, D. E. & Moore, R. W. An analysis of alpha-beta pruning. *Artif. Intel* 293–326 (1975).
41. Sutton, R. Learning to predict by the method of temporal differences. *Mach. Learn.* **3,** 9–44 (1988).
42. Baxter, J., Tridgell, A. & Weaver, L. Learning to play chess using temporal differences. *Mach. Learn.* **40,** 243–263 (2000).
43. Veness, J., Silver, D., Blair, A. & Uther, W. Bootstrapping from game tree s In *Advances in Neural Information Processing Systems* (2009).
44. Samuel, A. L. Some studies in machine learning using the game of checkers II - recent progress. *IBM J. Res. Develop.* **11,** 601–617 (1967).
45. Schaeffer, J., Hlynka, M. & Jussila, V. Temporal difference learning applied to high-performance game-playing program. In *17th International Joint Conference on Artificial Intelligence*, 529–534 (2001).
46. Tesauro, G. TD-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.* **6,** 215–219 (1994).
47. Dahl, F. Honte, a Go-playing program using neural nets. In *Machines that lear to play games*, 205–223 (Nova Science, 1999).
48. Rosin, C. D. Multi-armed bandits with episode context. *Ann. Math. Artif. Intell.* **61,** 203–230 (2011).
49. Lanctot, M., Winands, M. H. M., Pepels, T. & Sturtevant, N. R. Monte Carlo tree search with heuristic evaluations using implicit minimax backups. In *IEEE Conference on Computational Intelligence and Games*, 1–8 (2014).
50. Gelly, S., Wang, Y., Munos, R. & Teytaud, O. Modification of UCT with patterns Monte-Carlo Go. Tech. Rep. 6062, INRIA (2006).
51. Silver, D. & Tesauro, G. Monte-Carlo simulation balancing. In *26th Internation Conference on Machine Learning*, 119 (2009).
52. Huang, S.-C., Coulom, R. & Lin, S.-S. Monte-Carlo simulation balancing in practice. In *7th International Conference on Computers and Games*, 81–92 (Springer-Verlag, 2011).
53. Baier, H. & Drake, P. D. The power of forgetting: improving the last-good-repl policy in Monte Carlo Go. *IEEE Trans. Comput. Intell. AI in Games* **2,** 303–309 (2010).
54. Huang, S. & Müller, M. Investigating the limits of Monte-Carlo tree search methods in computer Go. In *8th International Conference on Computers and Games*, 39–48 (2013).
55. Segal, R. B. On the scalability of parallel UCT. *Computers and Games* **6515,** 36–47 (2011).
56. Enzenberger, M. & Müller, M. A lock-free multithreaded Monte-Carlo tree search algorithm. In *12th Advances in Computer Games Conference*, 14–20 (2009).
57. Huang, S.-C., Coulom, R. & Lin, S.-S. Time management for Monte-Carlo tree search applied to the game of Go. In *International Conference on Technologies and Applications of Artificial Intelligence*, 462–466 (2010).
58. Gelly, S. & Silver, D. Monte-Carlo tree search and rapid action value estimatio in computer Go. *Artif. Intell.* **175,** 1856–1875 (2011).

Impact on Game of Go

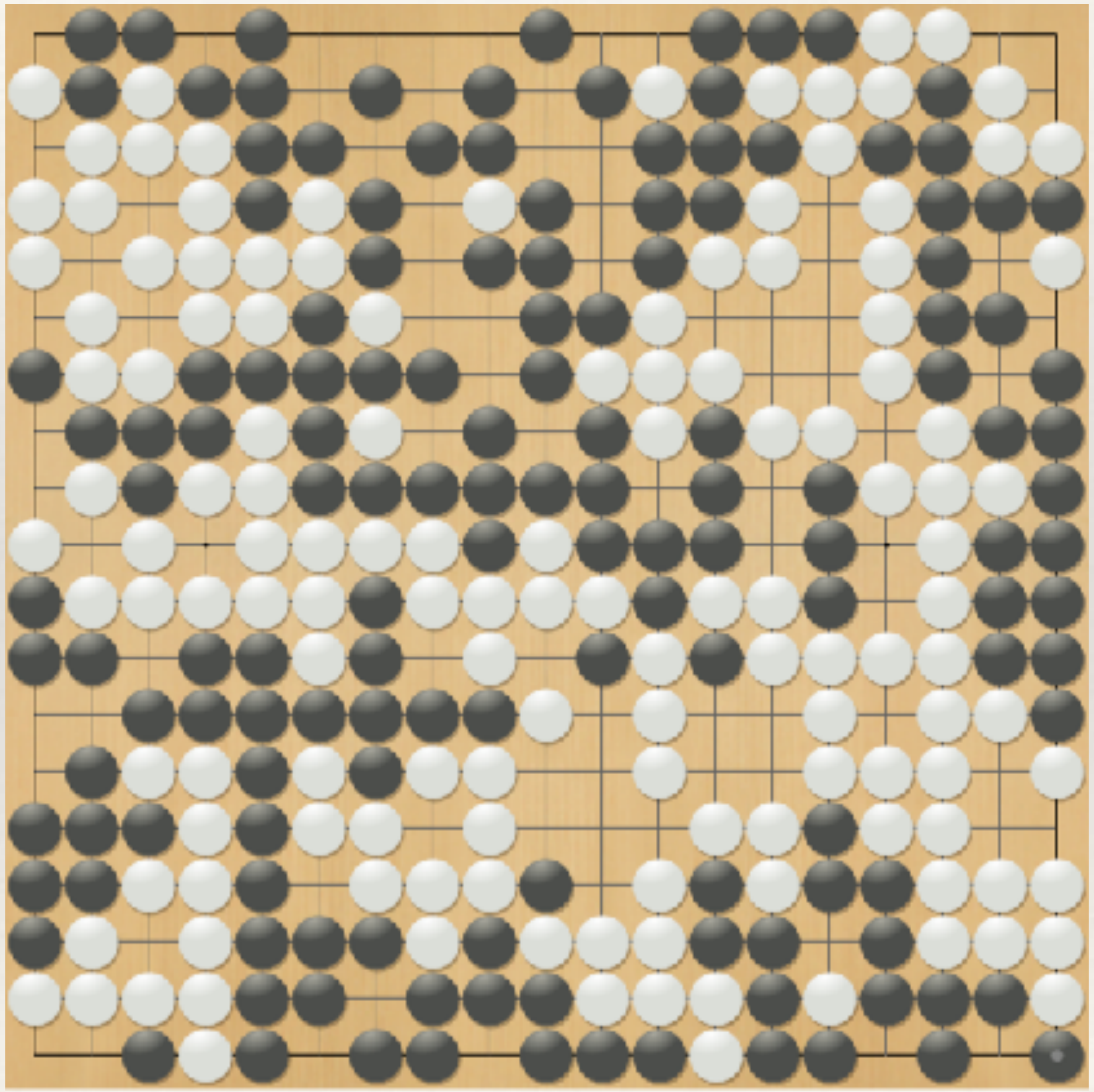
- AlphaGo received honorary 9 Dan diploma from both Chinese and Korean Go associations
- Strong impact on professional players
- Many new ideas, for example Ke Jie has experimented a lot with AlphaGo style openings
- Goal: Go programs as teaching tools
- Potential problem: cheating in tournaments?

What's Next in Computer Go?

- Currently, developing a top Go program is *Big Science*
  - Needs a large team of engineers
  - Example: Tencent's FineArt
- What can a small-scale university project contribute?
- One idea: work on *solving* parts of the game

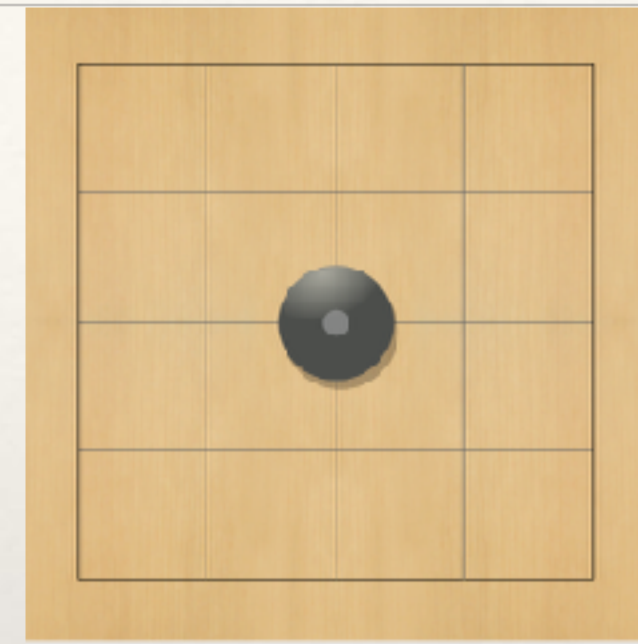
Is the Game of Go Solved Now?

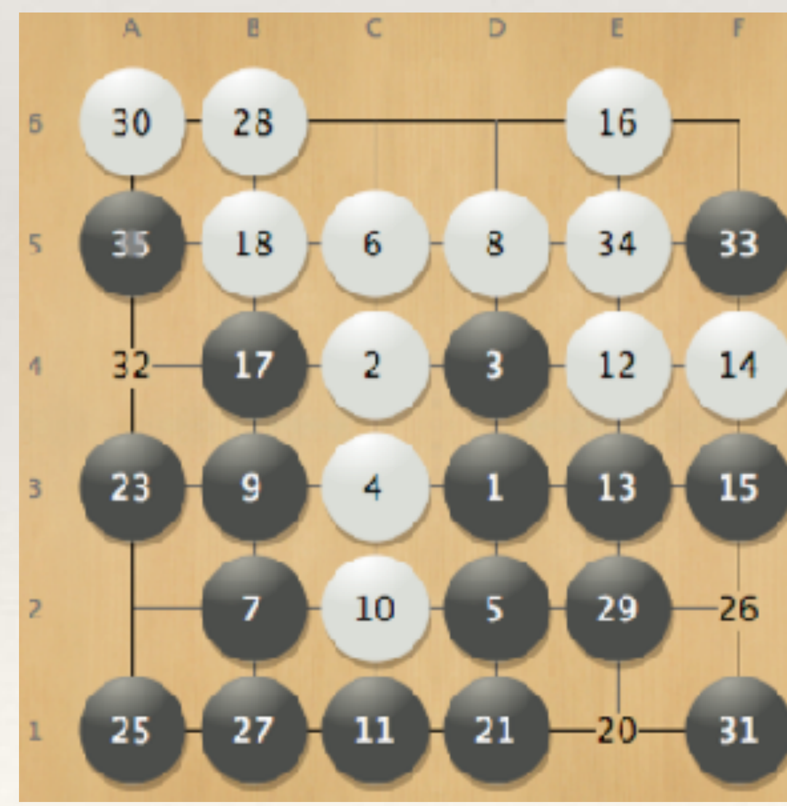
- **No!**
- AlphaGo is incredibly strong but…
  - … it is all based on heuristics
- AlphaGo still makes mistakes
- Example: 50 self-play games
  - Which color should win?
  - 38 wins for White
  - 12 wins for Black
  - One of these results must be wrong



Solving Go on Small Boards

- Solving means *proving* the best result *against any possible* opponent play
- Much harder to scale up than heuristic play
- 5x5, 5x6 Go are the largest solved board sizes (v.d.Werf 2003, 2009)
- Much work to be done: 6x6, 7x7, …

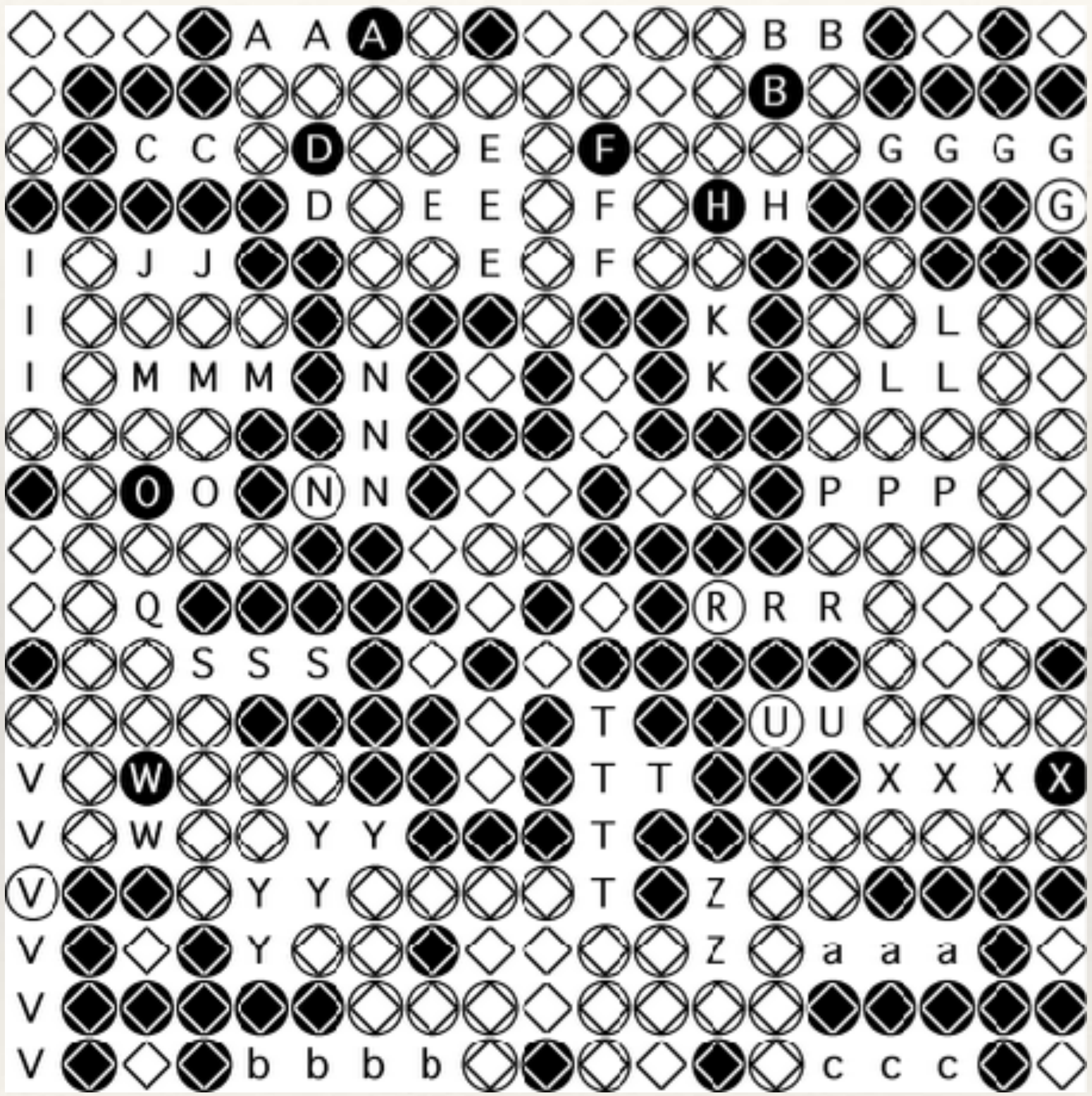


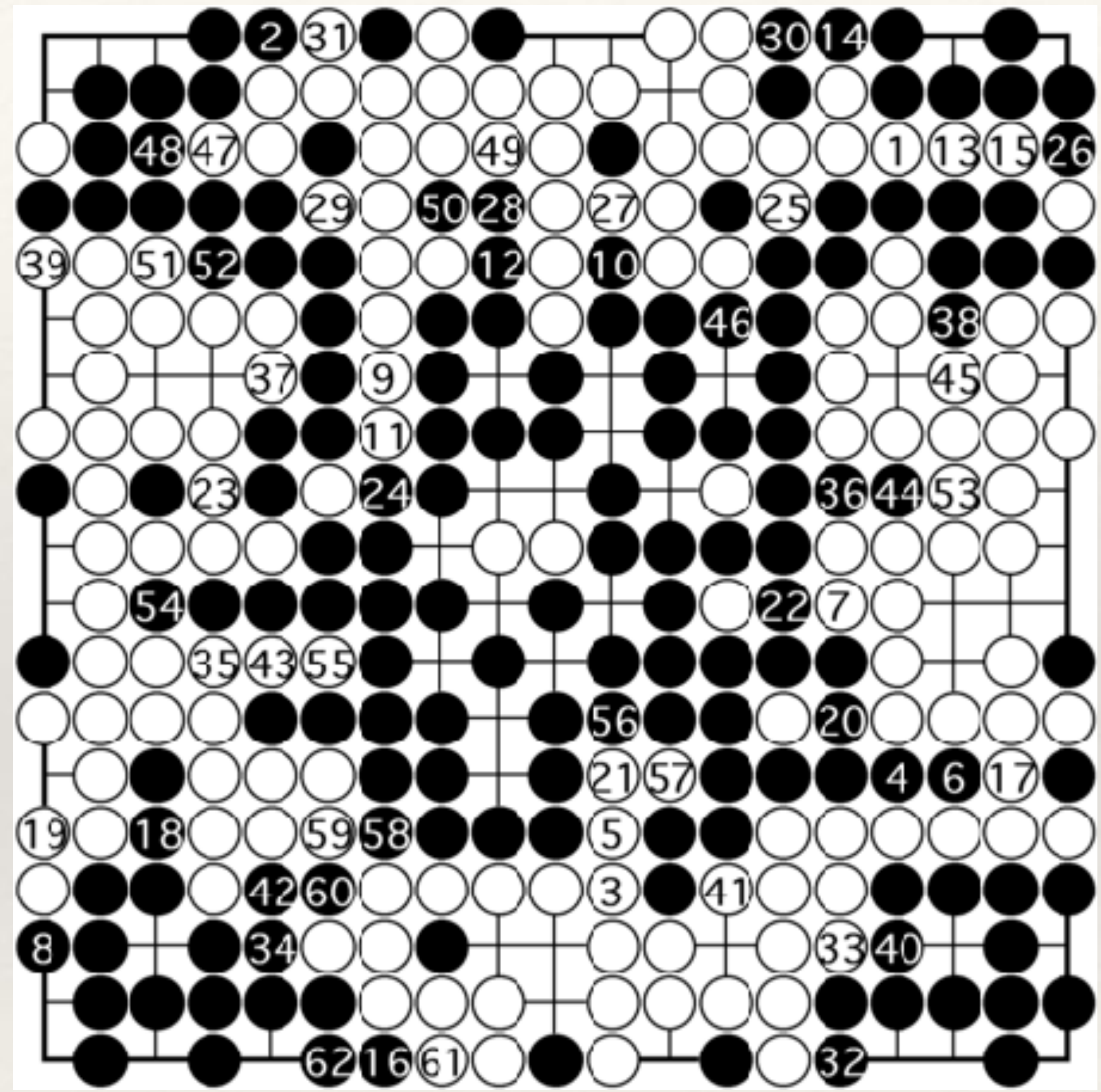


Solving Go Endgames

- How about solving 19x19 Go?
- Completely impossible, much too hard
- Solving endgames is more promising
- Can play *some* full-board 19x19 puzzles perfectly
  - Algorithms based on *combinatorial game theory* (Berlekamp+Wolfe 1994, Müller 1995)

Solving Go Endgame Puzzles





(Theory Berlekamp+Wolfe 1994,
computer program Müller 1995)

Impact on Computing Science, AI

- The promise of AlphaGo: methods are general, little game-specific engineering
- Shown that we have algorithms to acquire strong knowledge from very complex domains
- Challenge: what about real life applications?
  - Rules are not clear and change, hard to simulate
  - Even more actions
  - Less precise goals and evaluation

Impact on General Public

- Massive publicity about AlphaGo's success
- Illustration of the power of AI methods
- Feelings of both opportunities and fear
  - We can solve many complex problems with AI
  - Will AI destroy many good human jobs?
    Or replace boring jobs with better ones?

Summary and Outlook

- DeepMind's AlphaGo program is an incredible research breakthrough
- Landmark achievement for Computing Science
- Reviewed the main techniques that made this progress possible
- One big question: will the techniques apply to other problems?