Recent Progress in Computer Go

Martin Müller
University of Alberta
Edmonton, Canada

40 Years of Computer Go

- 1960's: initial ideas
- 1970's: first serious program - Reitman & Wilcox
- 1980's: first PC programs, competitions
- 1990's: slow progress, commercial successes
- 2000's: GNU Go - strong open source program
- now: Monte-Carlo and UCT revolution,
  strong 9x9 programs

“Classical” Go Programs

- Goliath (Mark Boon)
- Go Intellect (Ken Chen)
- Handtalk (Chen Zhixing)
- Go++ (Michael Reiss)
- KCC (North Korean team)
- Many Faces of Go (David Fotland)
- GNU Go (international team)

Monte-Carlo Simulation and UCT for Go

- 1993 Bernd Brügmann - simulations for Go
- 200x Bouzy and students revive simulations
- 2006 Kocsis and Szepesvari - UCT algorithm
  - Sylvain Gelly, Yizao Wang - MoGo
  - Remi Coulom - Crazy Stone
  - Don Dailey - CGOS server, new programs

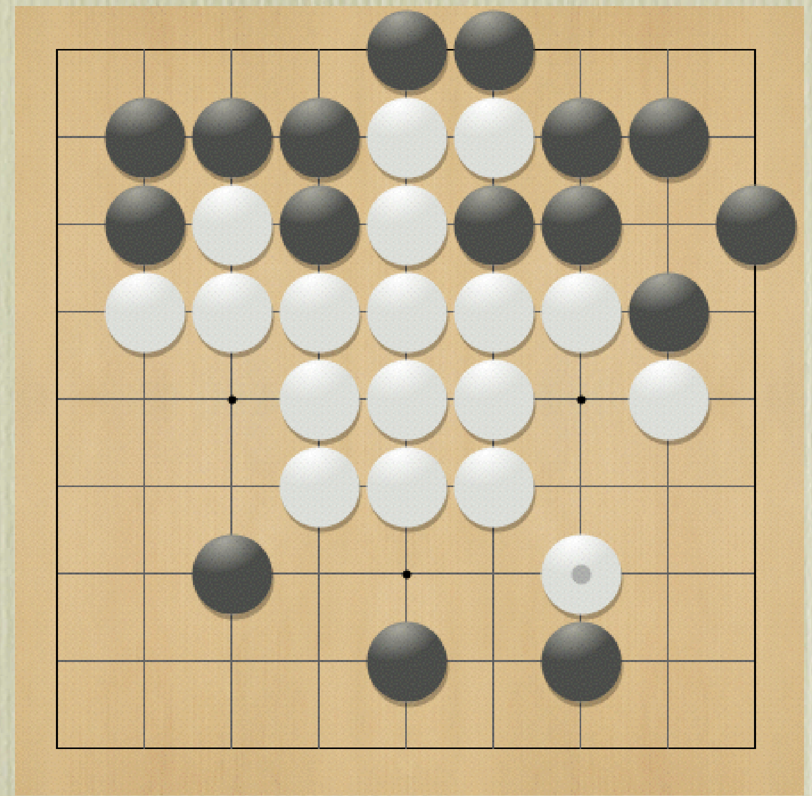
Classic vs New Go Programs

Classic	New
Knowledge intensive	Search intensive
Problem: heuristic position evaluation	No (!) heuristic evaluation
Local goal search	Global search + simulations

How Strong?

- almost perfect on 7x7
- amateur Dan level on 9x9
- 5 kyu on 19x19? Similar to top classic program

Dec. 2006 - My Wakeup Call

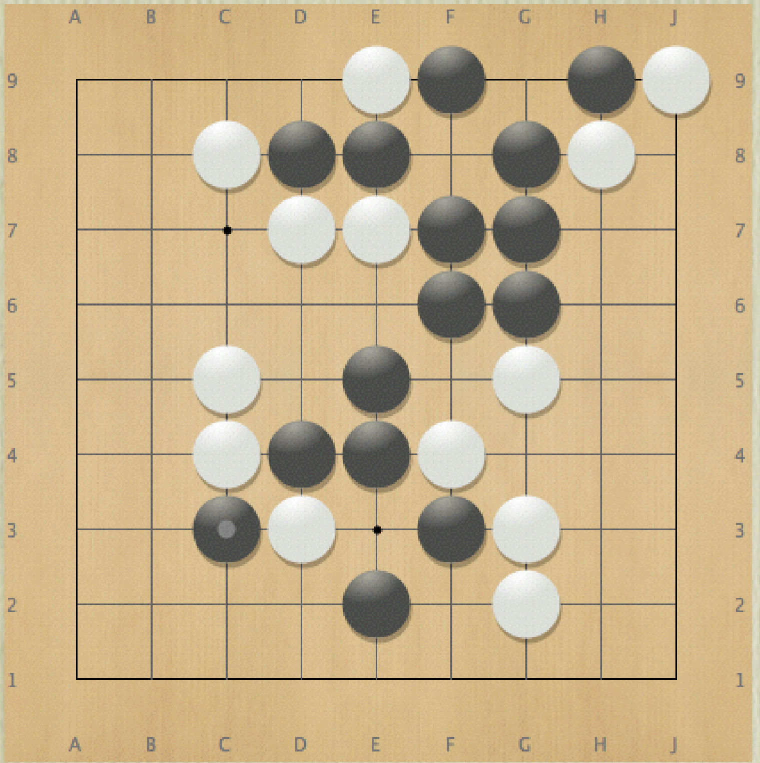


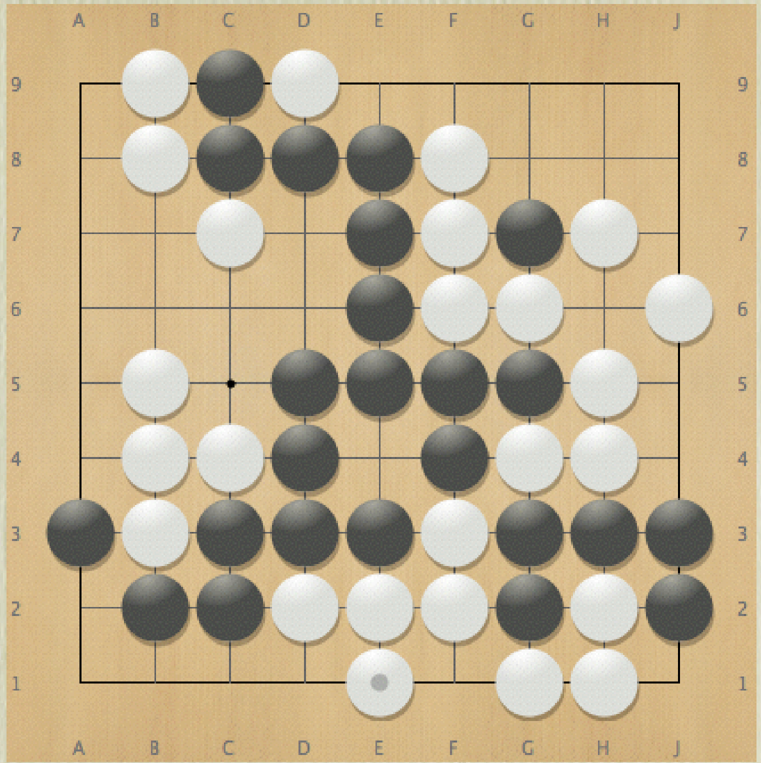
- Martin Müller vs Valkyria by Magnus Persson
  Komi 7.5

Games vs Guo Juan 5 Dan

- Aug. 2006 Match CrazyStone vs Guo Juan
  - 7x7 Board, 9 komi
  - CrazyStone white: always wins or jigo
  - Guo white: often wins
- June 2007 Match MoGo vs Guo Juan
  - 9x9 Board, MoGo black, 0.5 komi
  - 9 wins : 5 losses for MoGo

Examples



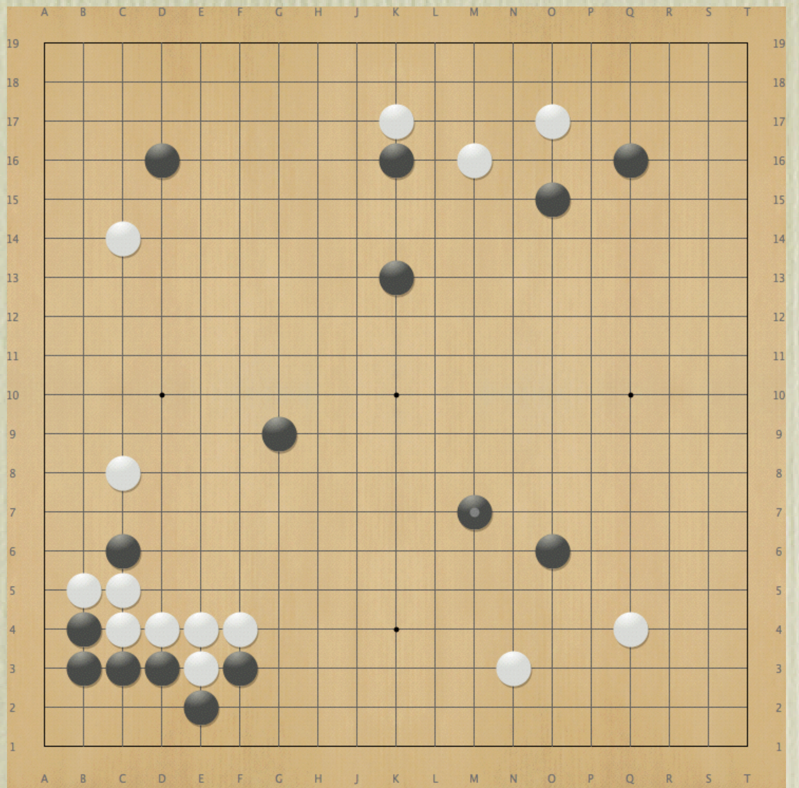


guojuan-MoGoBot.sgf, guojuan-MoGoBot-9.sgf

Playing Style

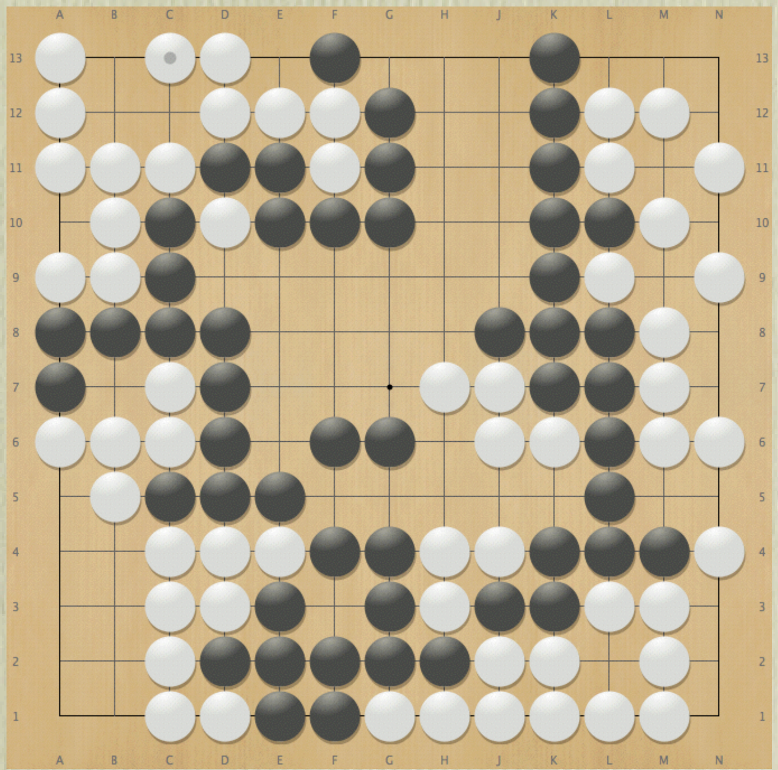
- Monte-Carlo based programs play many strange moves...
- ...but they are very good at winning!
    - only care about winning, not the score
    - play safe when ahead
    - try invasions when behind

“Cosmic” Style Opening



Ruky-MoGoBot-2.sgf moves 16-31

Example: Random Play in Decided Games



GNU-StoneCrazy.sgf moves 122-132

How Does it Work?

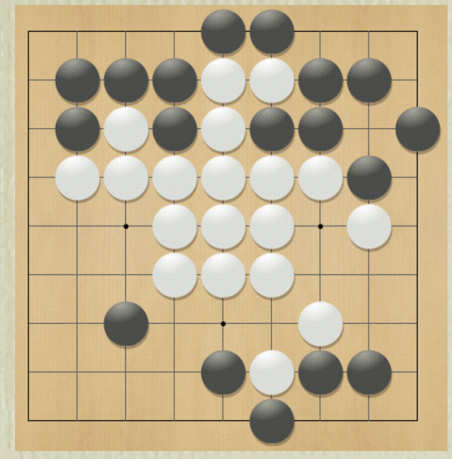
- Monte-Carlo Simulations
  - Basic Idea
  - Refinements
- **UCT** method
  (**U**pper **C**onfidence bounds applied to **T**rees)
  - Building a Game Tree
  - Evaluation

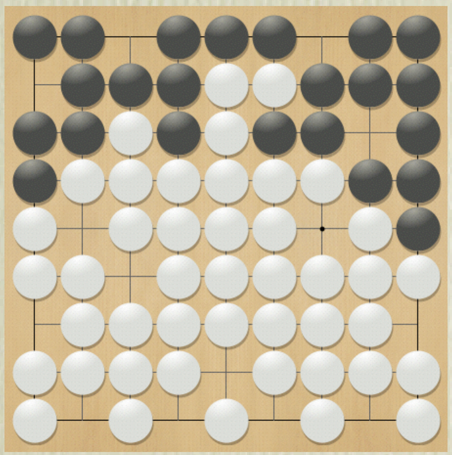
Simulations

- Monte-Carlo simulation
  - Popular in physics
  - Study behavior of complex system
    by running many random simulations
  - Go: play random game from current position

Simulation - Example

- Random legal move
- Do not fill one point eyes
- Game over after both pass
- Evaluate by Chinese rules
  1 for win
  0 for loss





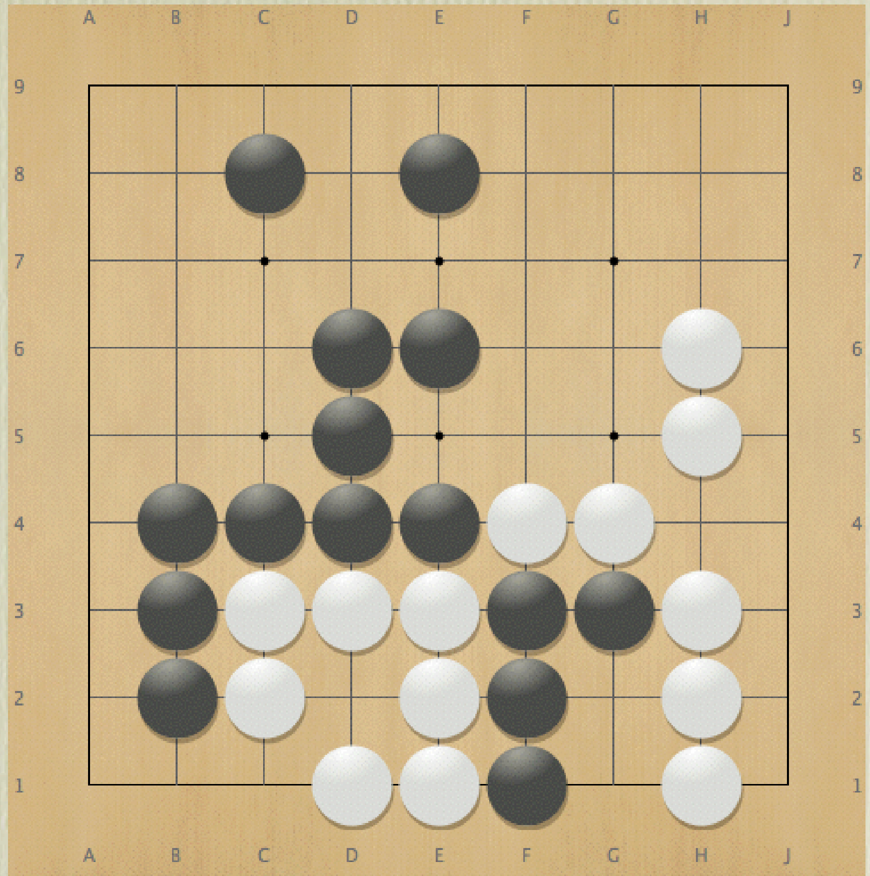
valkyria-ExBoss-randomgame.sgf

Simulation-Based Player

- Play many random games
- Win/loss statistics for each possible move
- Play move with highest win percentage
- Fast
  - Over 1 Million moves/sec.
  - Typical 100.000 simulations per move
- Weakness: loves to play threats

Example - Bad Threat

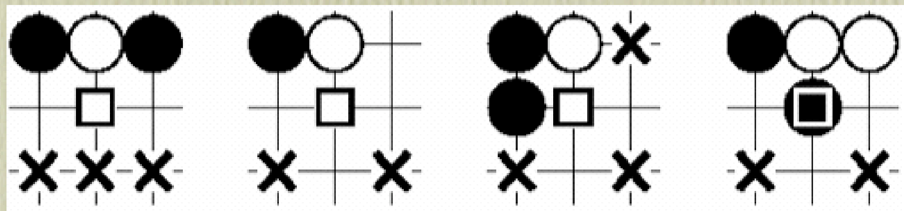
- C1 is a bad threat, if White captures on B1
- Black cannot save F1 stones
- In pure random simulations, C1 works very often!



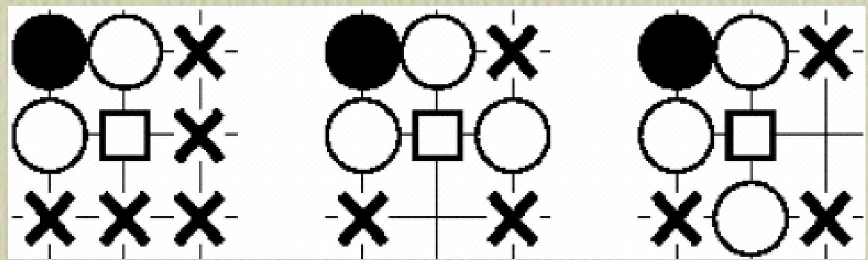
Refinement of Simulations

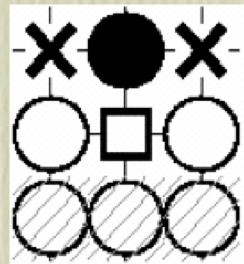
- Add Go knowledge
  - Capture/escape from capture
  - Avoid self-atari
  - Simple cutting/blocking patterns
  - Play near last move(s)
- Must be extremely fast to compute

The MoGo Patterns

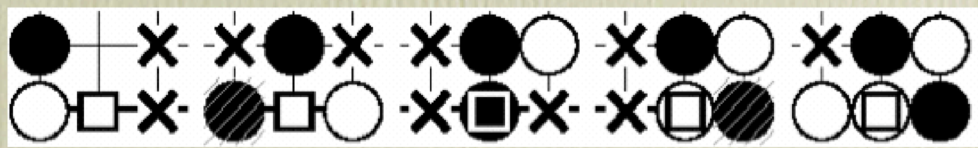


Hane/Extend



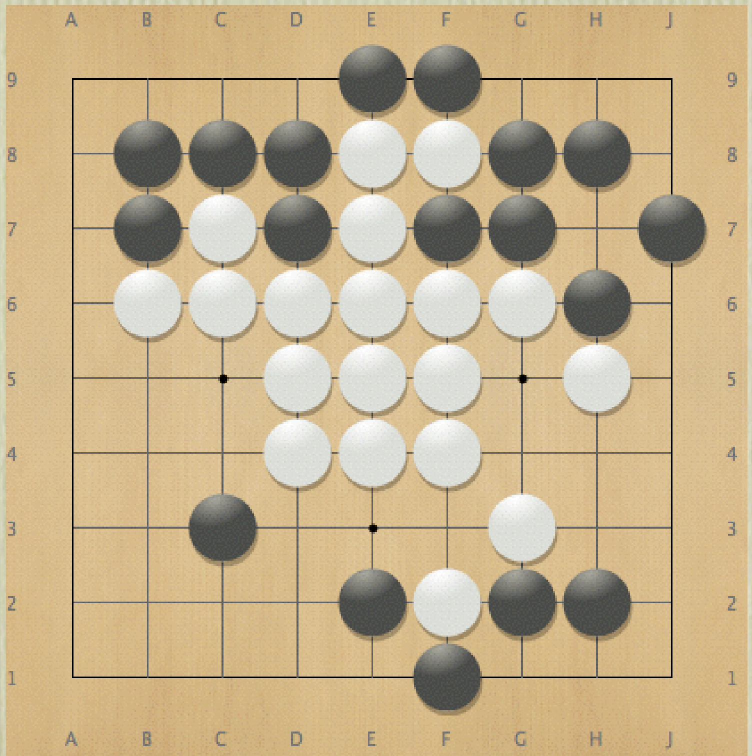


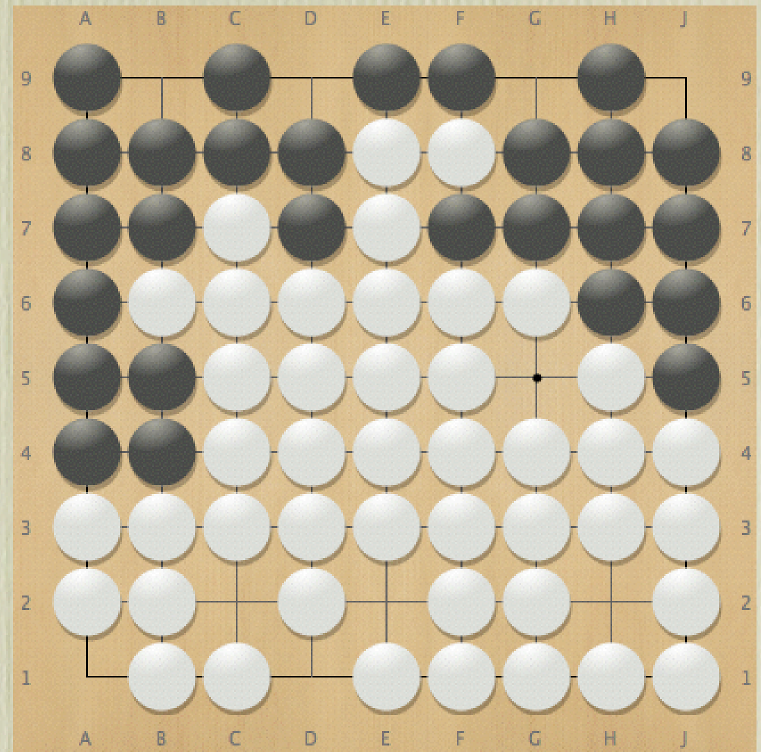
Cut/Connect



Edge of board

Example of Biased Simulation





valkyria-ExBoss-biased-random-game.sgf

Adding Game Tree Search

- Pure simulation is limited
- Weak in tactics
- Classical game-playing uses *game tree search*
  - minimax, alpha-beta
  - new selective search method - UCT

UCT Idea

- Follow “best moves” down the tree
- At leaf, start a simulation
- Add first new move to tree

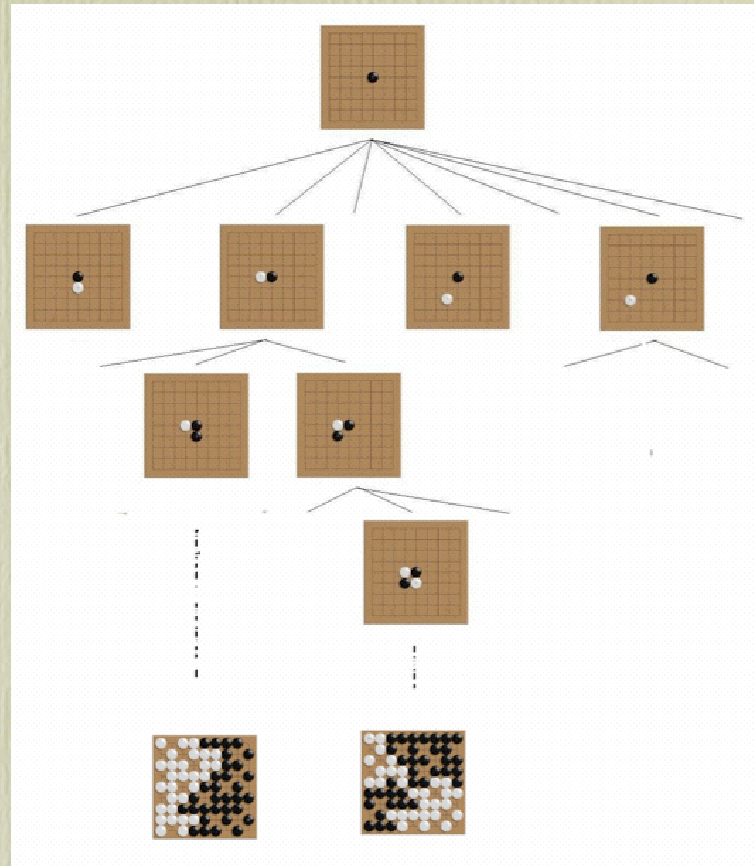


Image by Sylvain Gelly

What is the “Best” Move

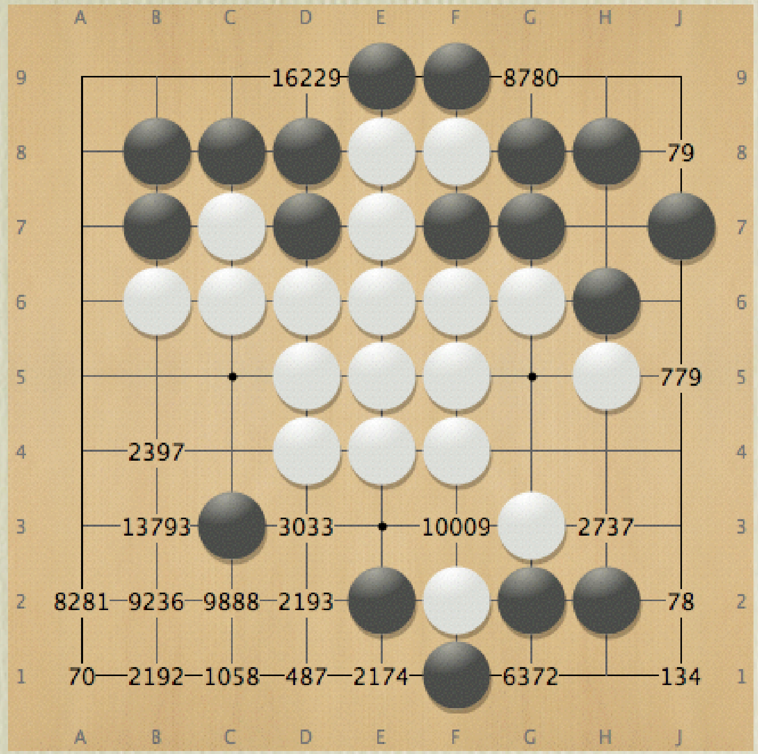
- Where can we gain most valuable information?
  - Move that looks good so far
  - Move that has not been analyzed much yet
- UCT is a compromise
  - Select move where *success rate + uncertainty* is highest.

UCT Evaluation

- Classical Minimax:
  - Value = value of position after best move
- UCT:
  - Value = weighted average of moves
  - Weight = number of simulations for that move

Example

- Very selective search
- Concentrates on few promising moves
- approaches minimax value if optimal move(s) get most simulations



Refinements to Tree Search

- RAVE (Gelly & Silver 2007)
- Add Go knowledge
  - Patterns (Coulom 2007)
  - Reinforcement learning (Gelly & Silver 2007)

RAVE - Rapid Action Value Estimation

- UCT needs many samples of all moves - slow
- Idea: moves later in simulation also important
  - All moves as first (Brügmann 1993)
  - Win statistics for each move in all games
- Use at beginning
- Phase out gradually

Using Go Knowledge

- Use Go knowledge to initialize value of moves
  - Also phase out gradually
  - Use RLGO evaluation function in MoGo (Gelly & Silver 2007)
  - Can be combined with RAVE
- Learn feature values for pruning and progressive widening of tree (Coulom 2007)

Why Does it Work so Well?

- No theoretical explanation
- Excellent empirical results
- Simulations: good move in random Go is often a good move in Go
- UCT: good moves in random Go are interesting moves to try in search

Future - Scaling Up

- Scales well with increasing computer power
  - No limit in sight - Don Dailey's experiment
- Challenge: parallel search
  - Shared memory
  - Computer clusters
  - Bottleneck: update tree, select best line

Summary

- Revolution through Monte-Carlo simulations and UCT
- Strong 9x9 programs
- When will we see strong 19x19?