Monte-Carlo Tree Search (MCTS) for Computer Go

Bruno Bouzy
bruno.bouzy@parisdescartes.fr
Université Paris Descartes

AOA class

Outline

- The game of Go: a 9x9 game
- The « old » approach (*-2002)
- The Monte-Carlo approach (2002-2005)
- The MCTS approach (2006-today)
- Conclusion

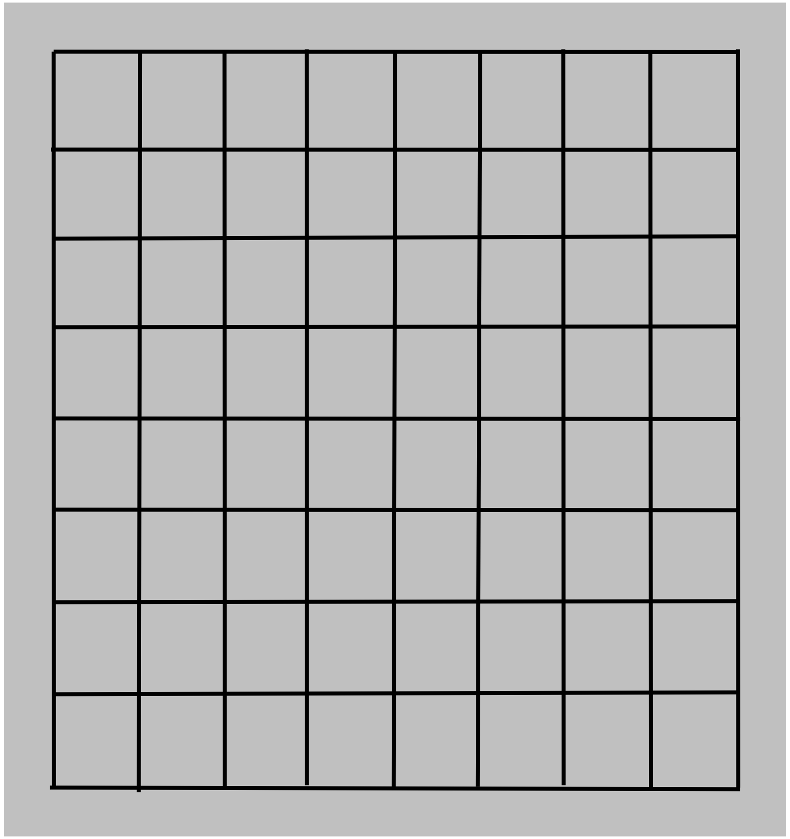
The game of Go

The game of Go

- 4000 years
- Originated from China
- Developed by Japan (20th century)
- Best players in Korea, Japan, China
- 19x19: official board size
- 9x9: beginners' board size

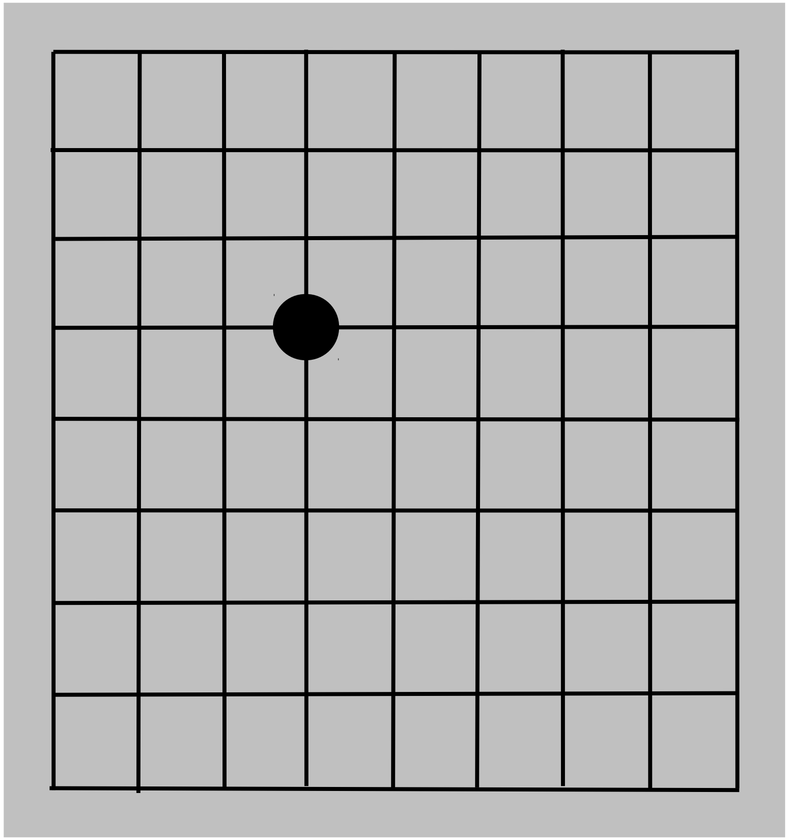
A 9x9 game

- The board has 81 « intersections ». Initially,it is empty.



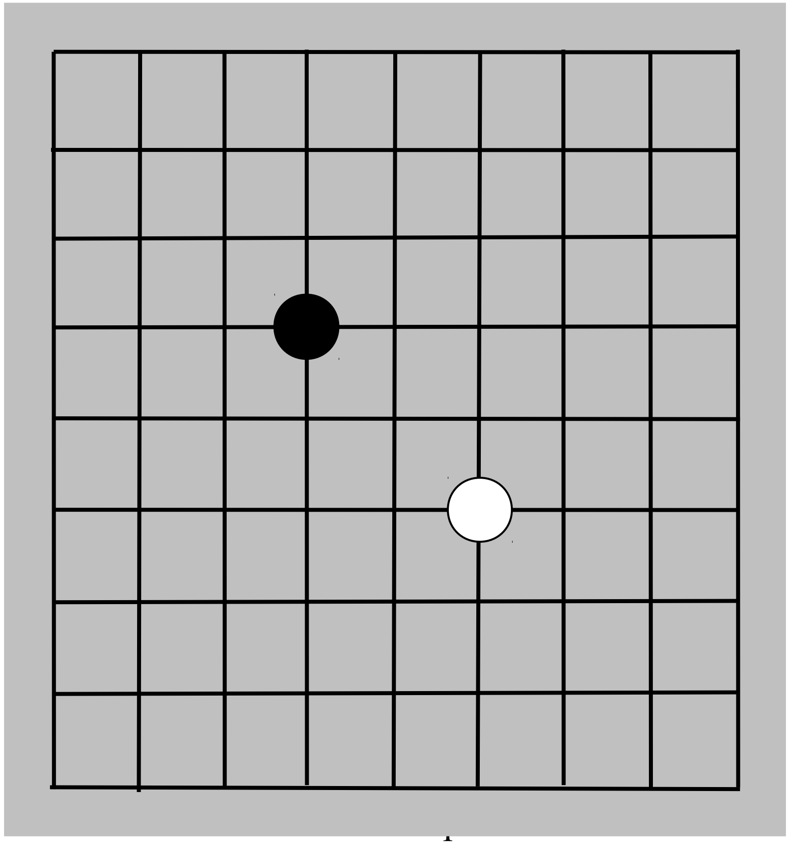
A 9x9 game

- Black moves first. A « stone » is played on an intersection.



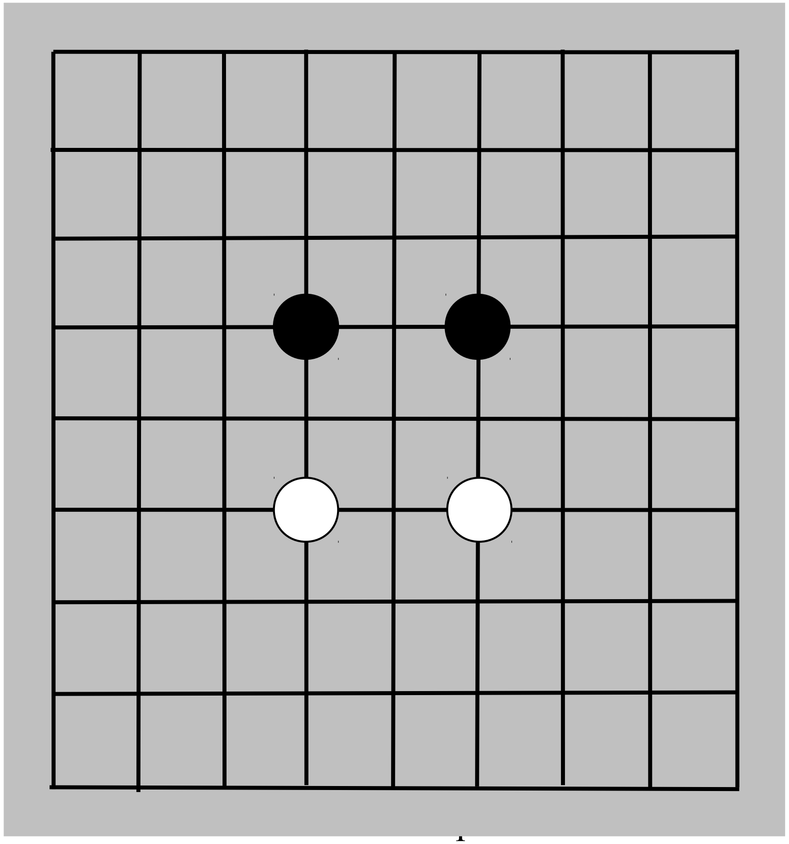
A 9x9 game

- White moves second.



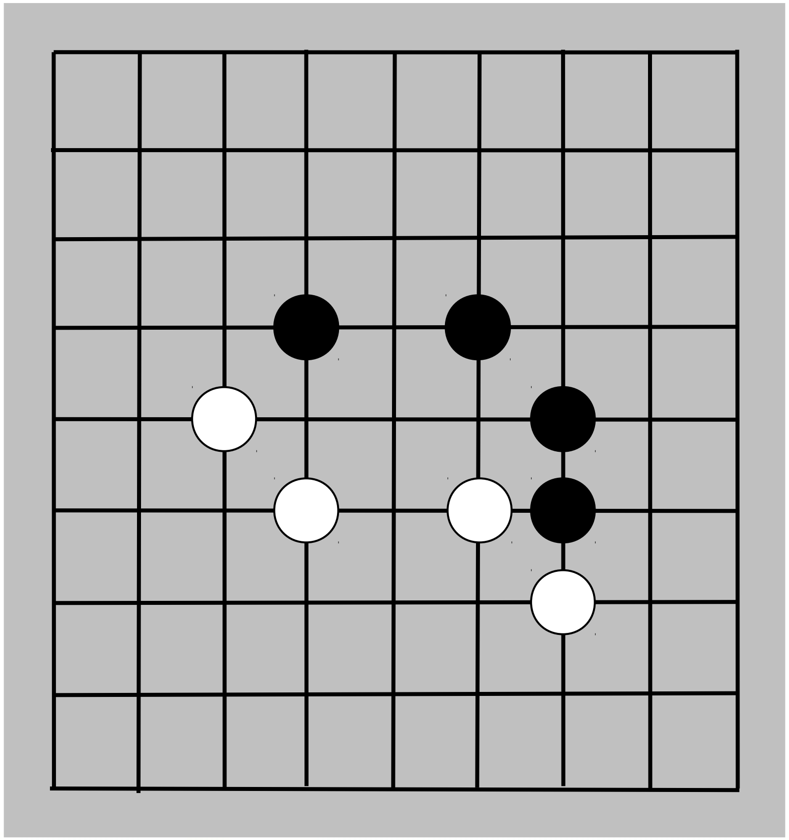
A 9x9 game

- Moves alternate between Black and White.



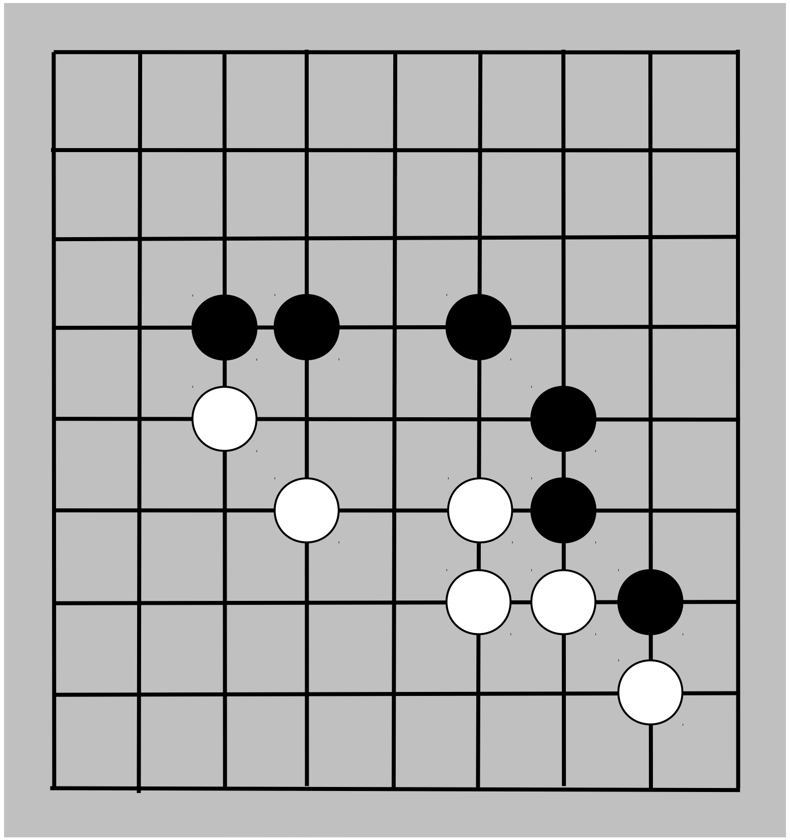
A 9x9 game

- Two adjacent stones of the same color builds a « string » with « liberties ».
- 4-adjacency



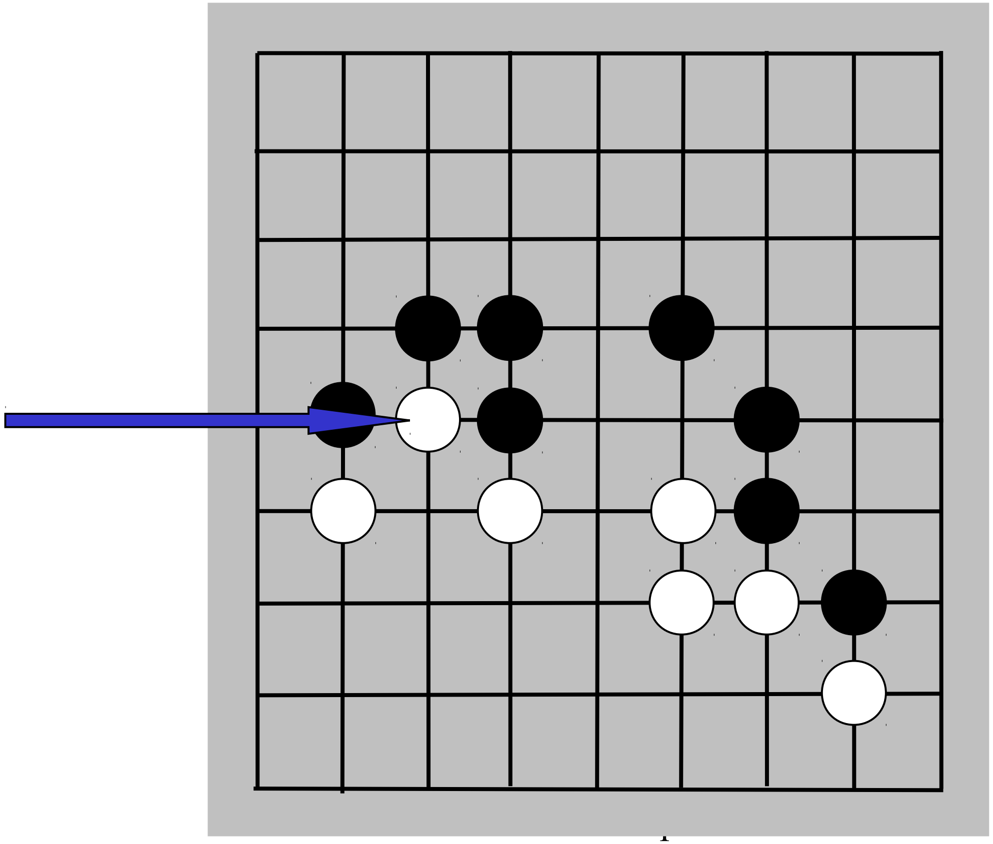
A 9x9 game

- Strings are created.



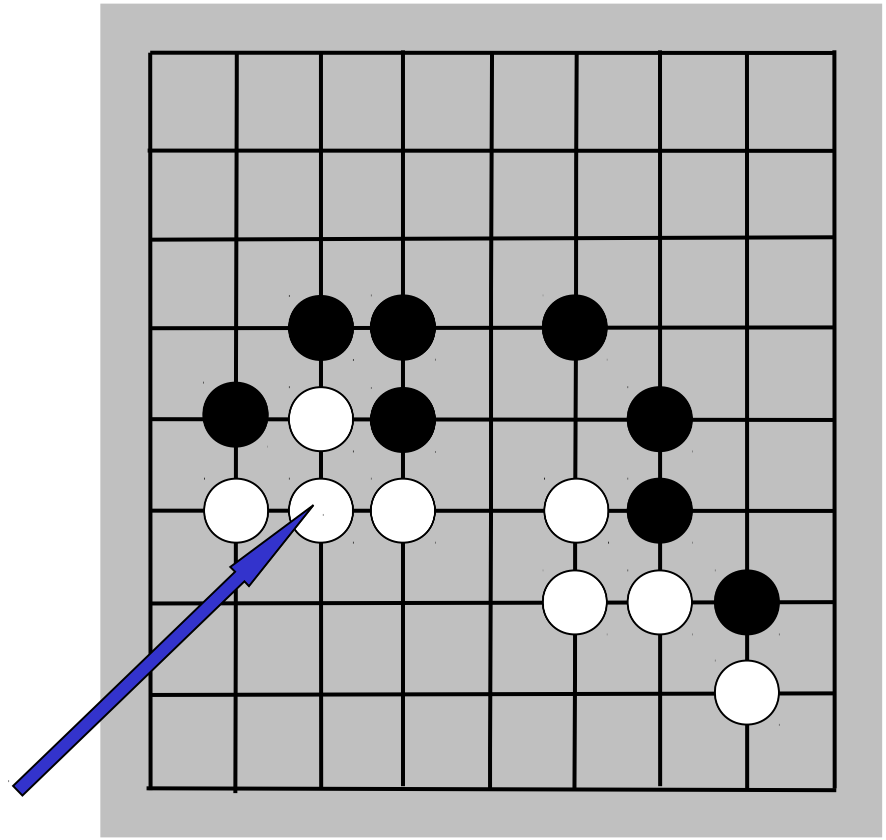
A 9x9 game

- A white stone is in « atari » (one liberty).



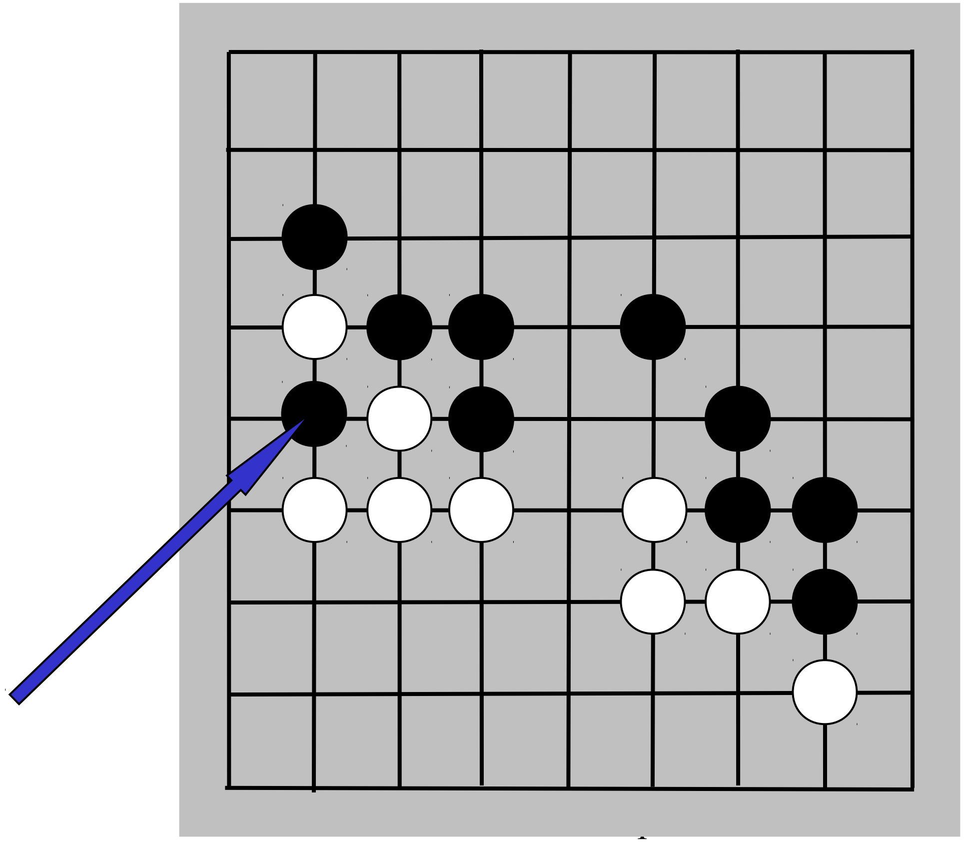
A 9x9 game

- The white string has five liberties.



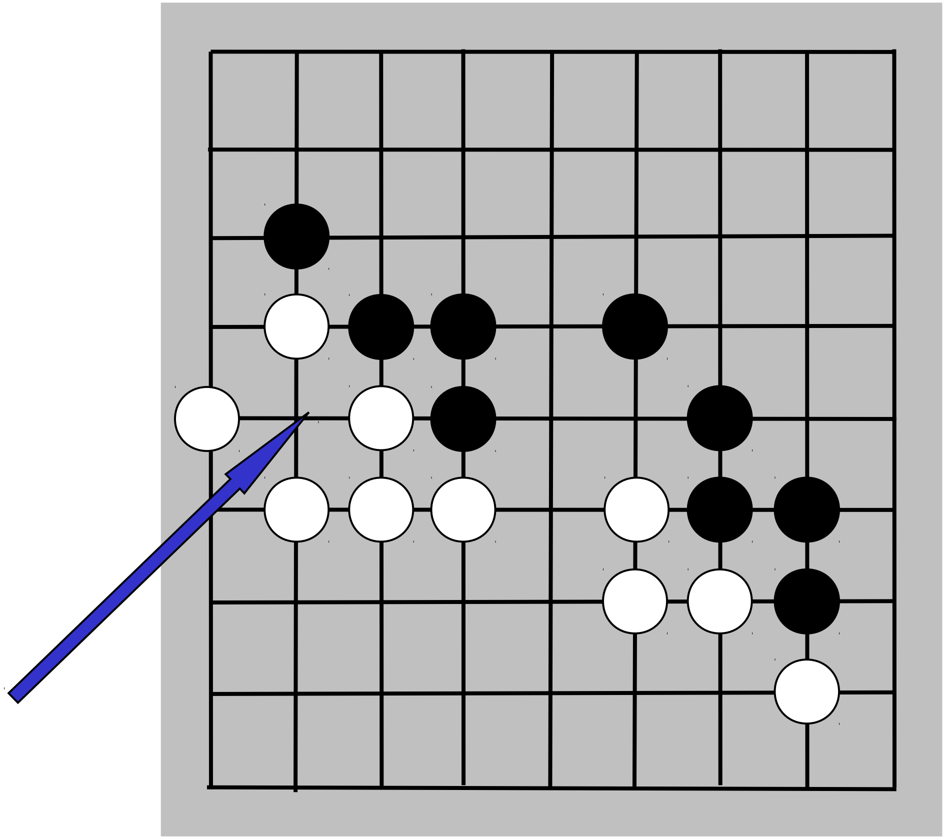
A 9x9 game

- The black stone is « atari ».



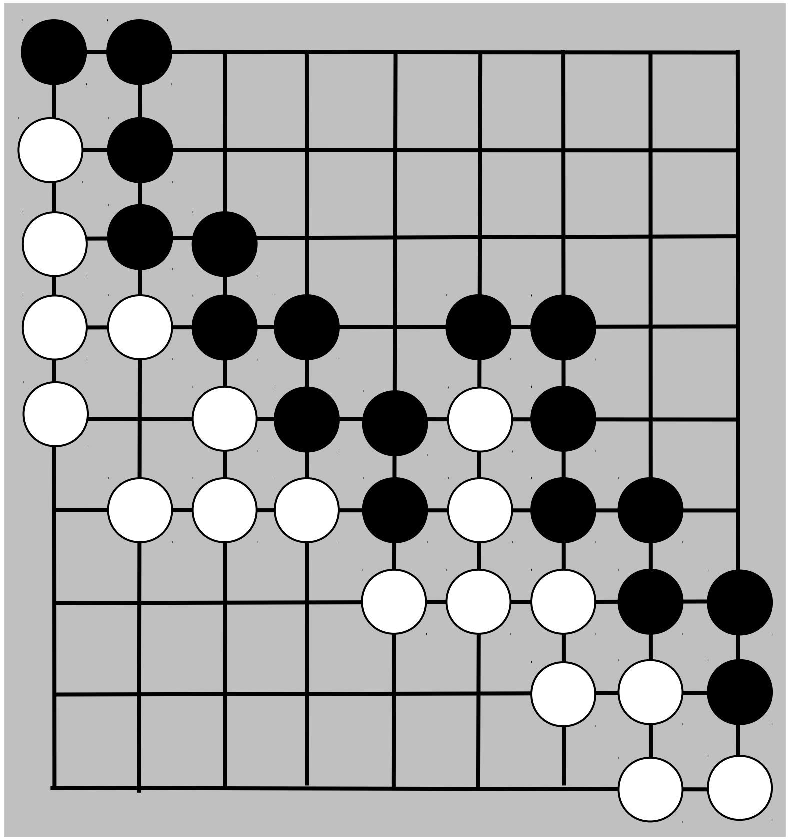
A 9x9 game

- White « captures » the black stone.



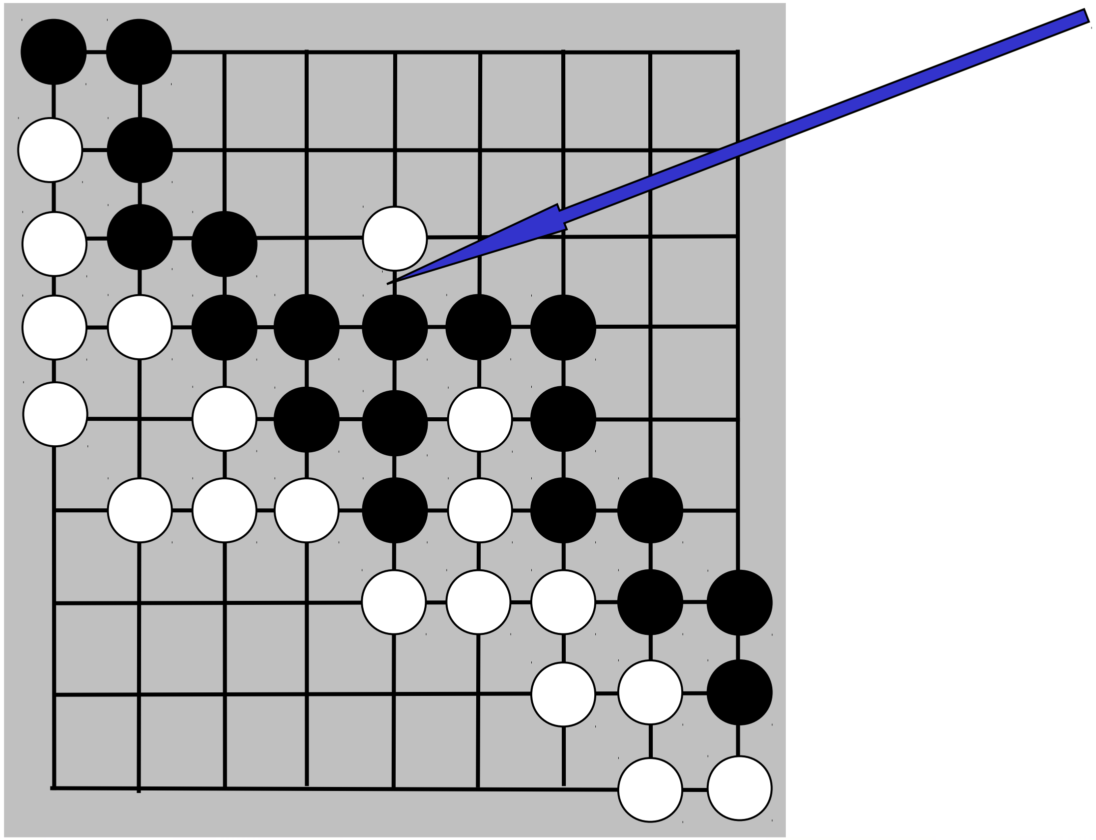
A 9x9 game

- For advised players, the game is over.
  - Hu?
  - Why?



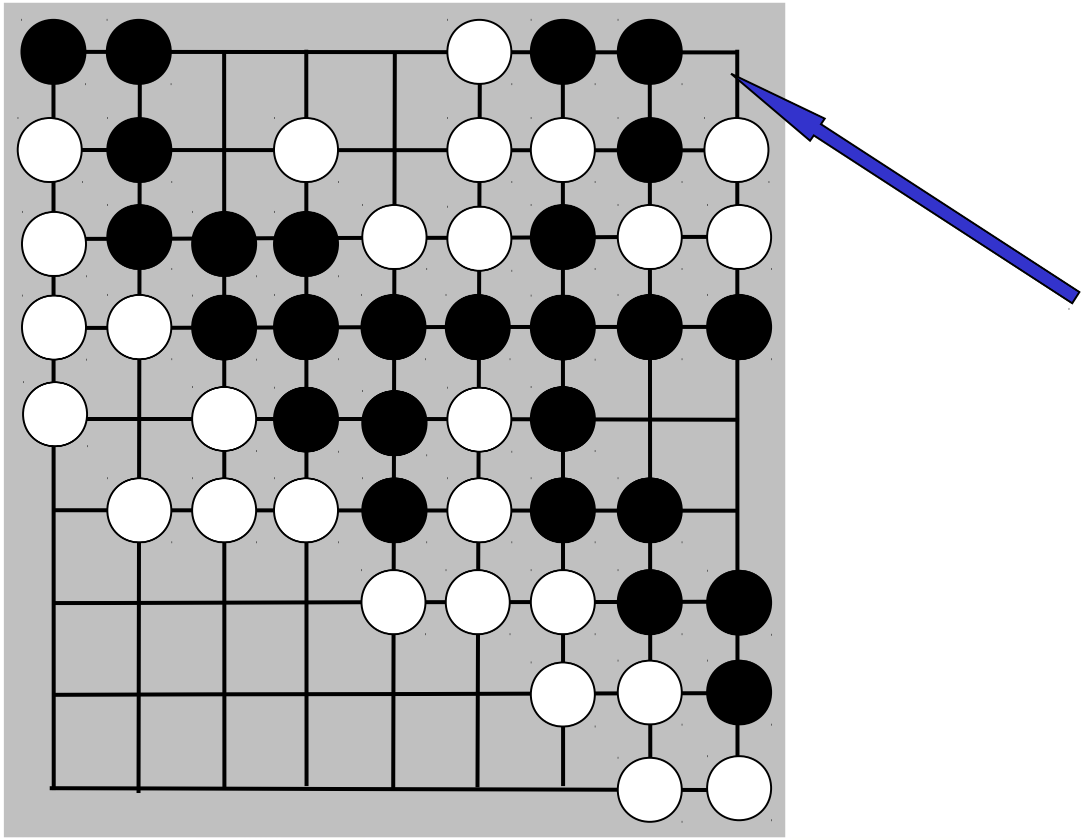
A 9x9 game

- What happens if White contests black « territory »?



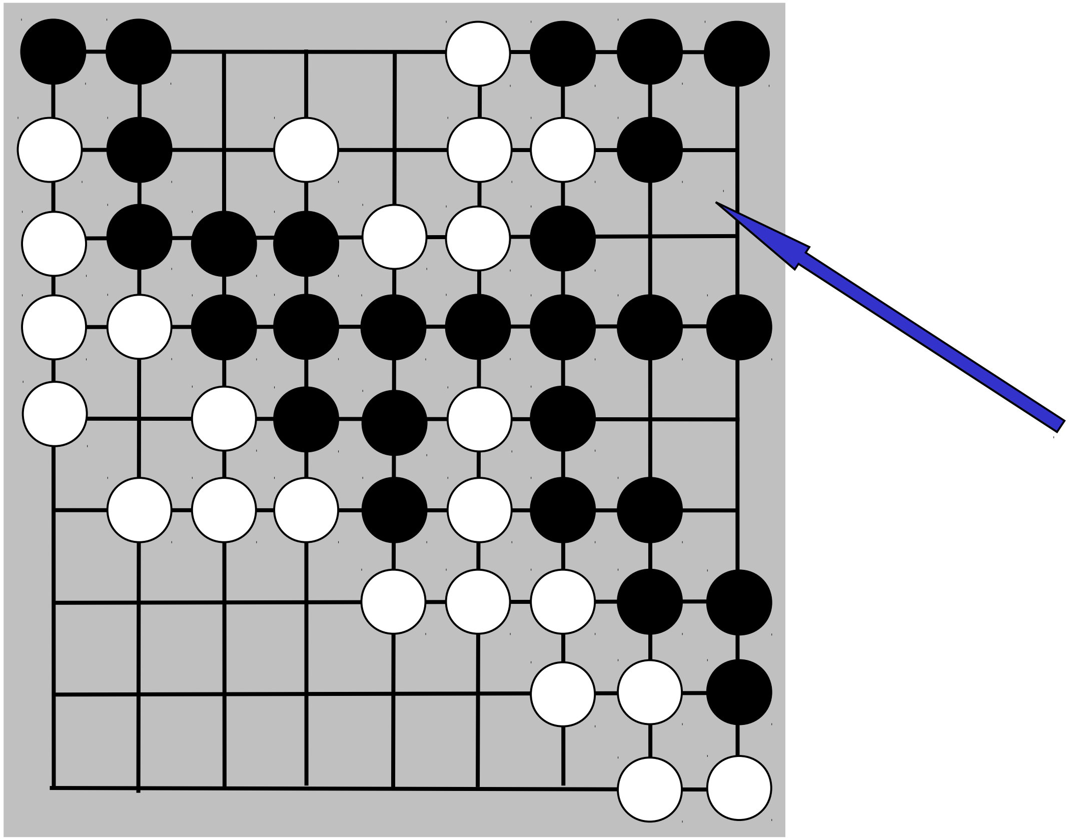
A 9x9 game

- White has invaded. Two strings are atari!



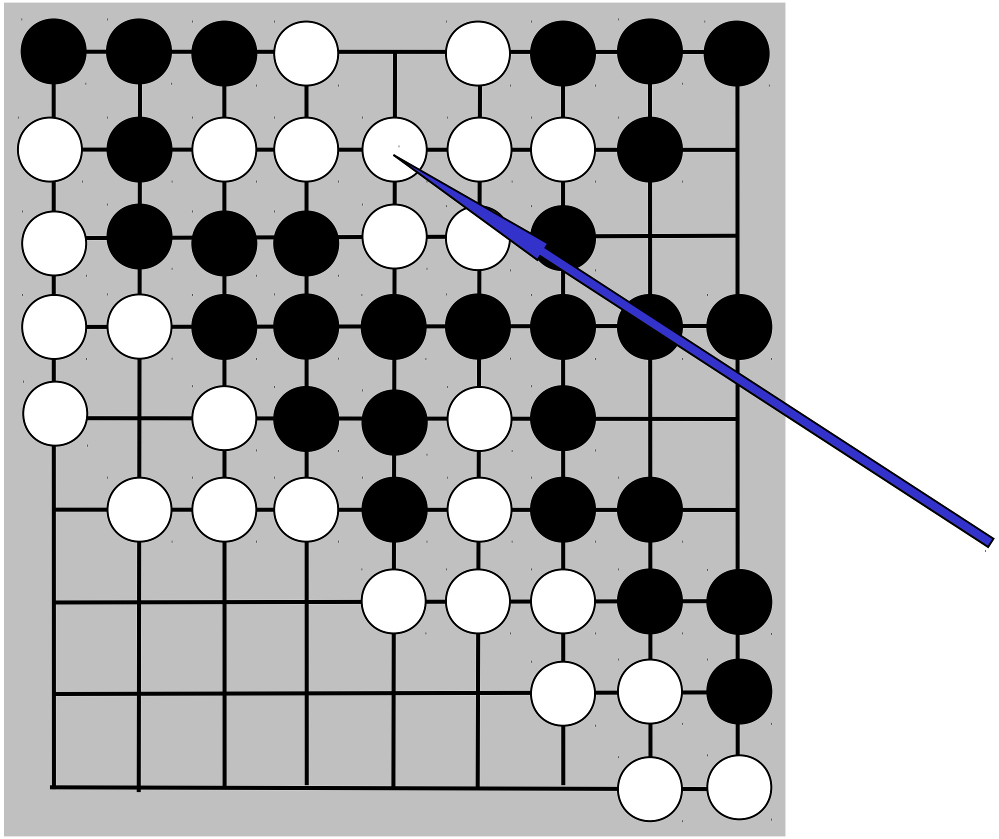
A 9x9 game

- Black captures !



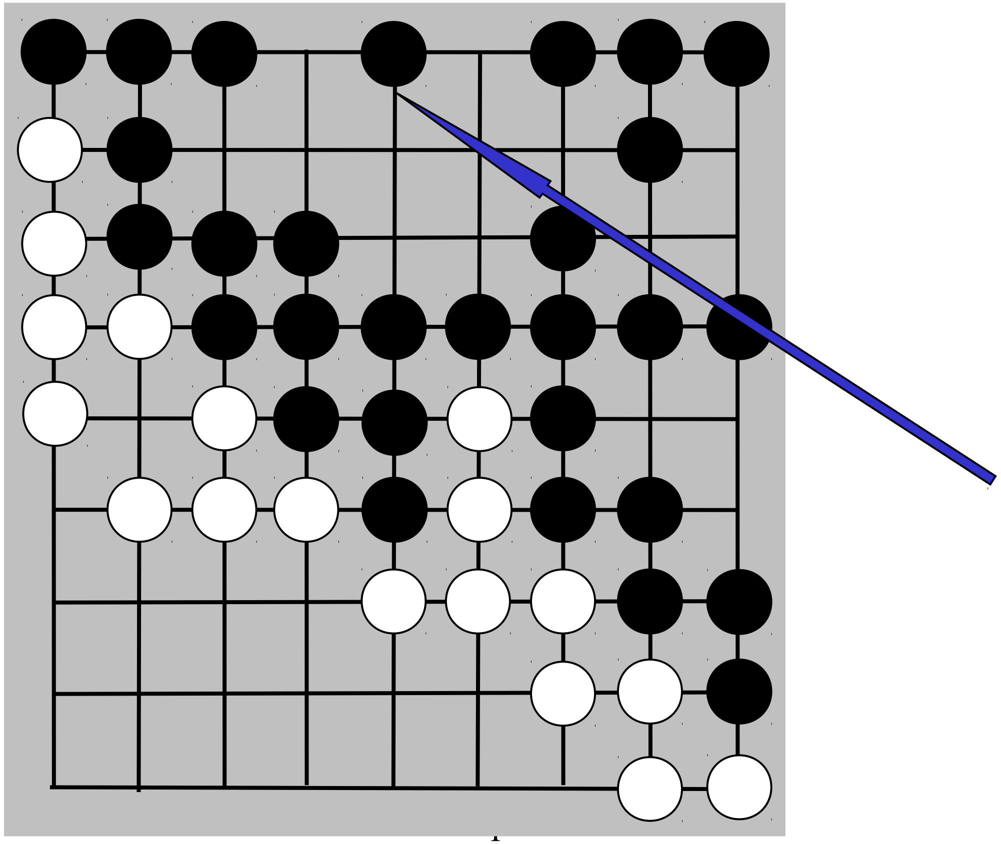
A 9x9 game

- White insists but its string is atari...



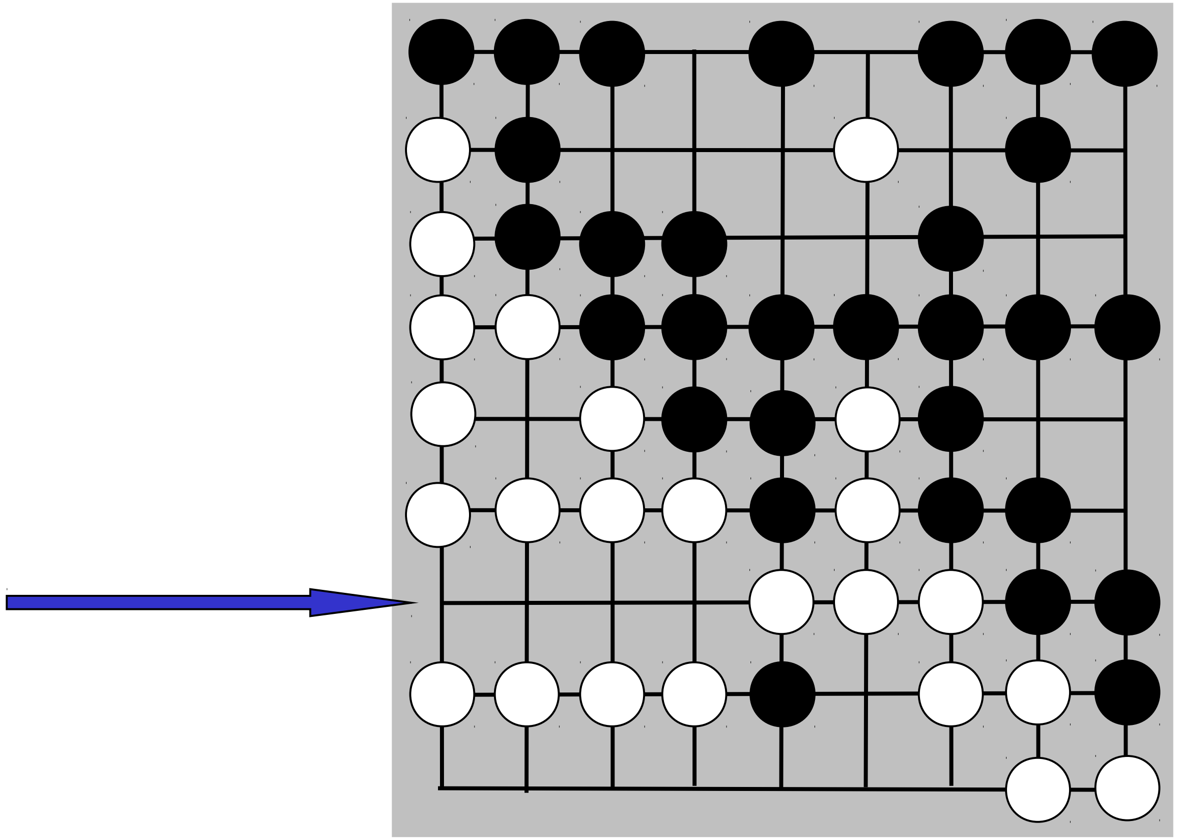
A 9x9 game

- Black has proved is « territory ».



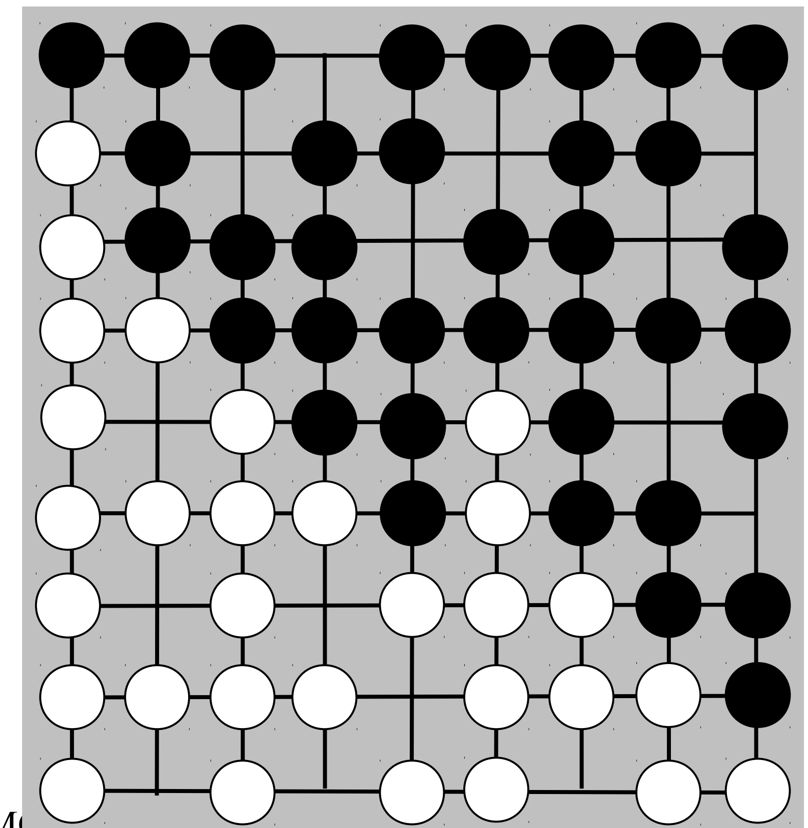
A 9x9 game

- Black may contest white territory too.



A 9x9 terminal position

- The game is over for computers.
  - Hu?
  - Who won ?

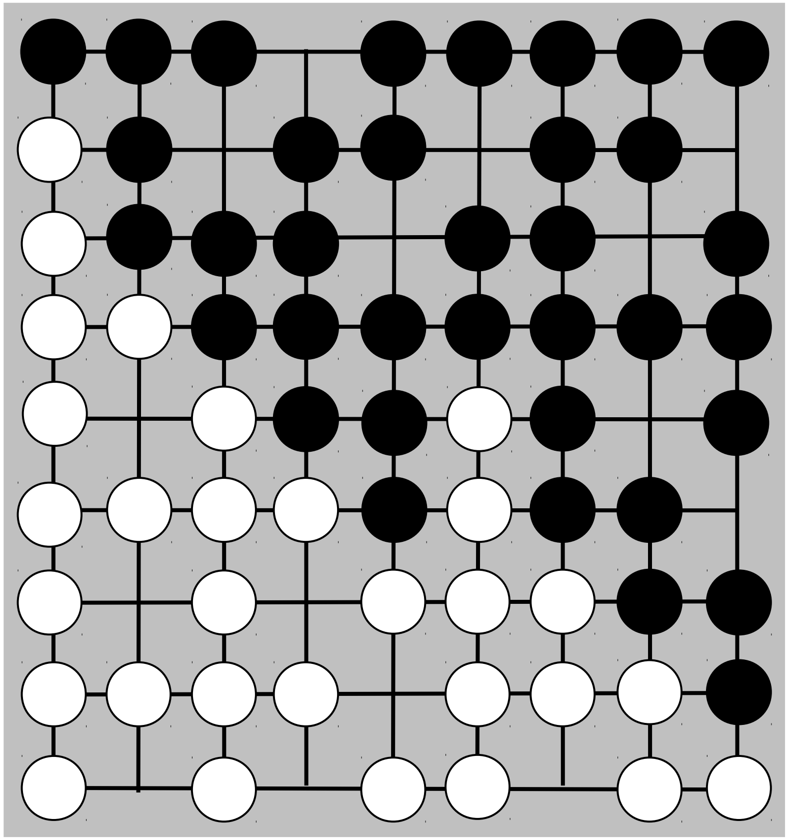


A 9x9 game

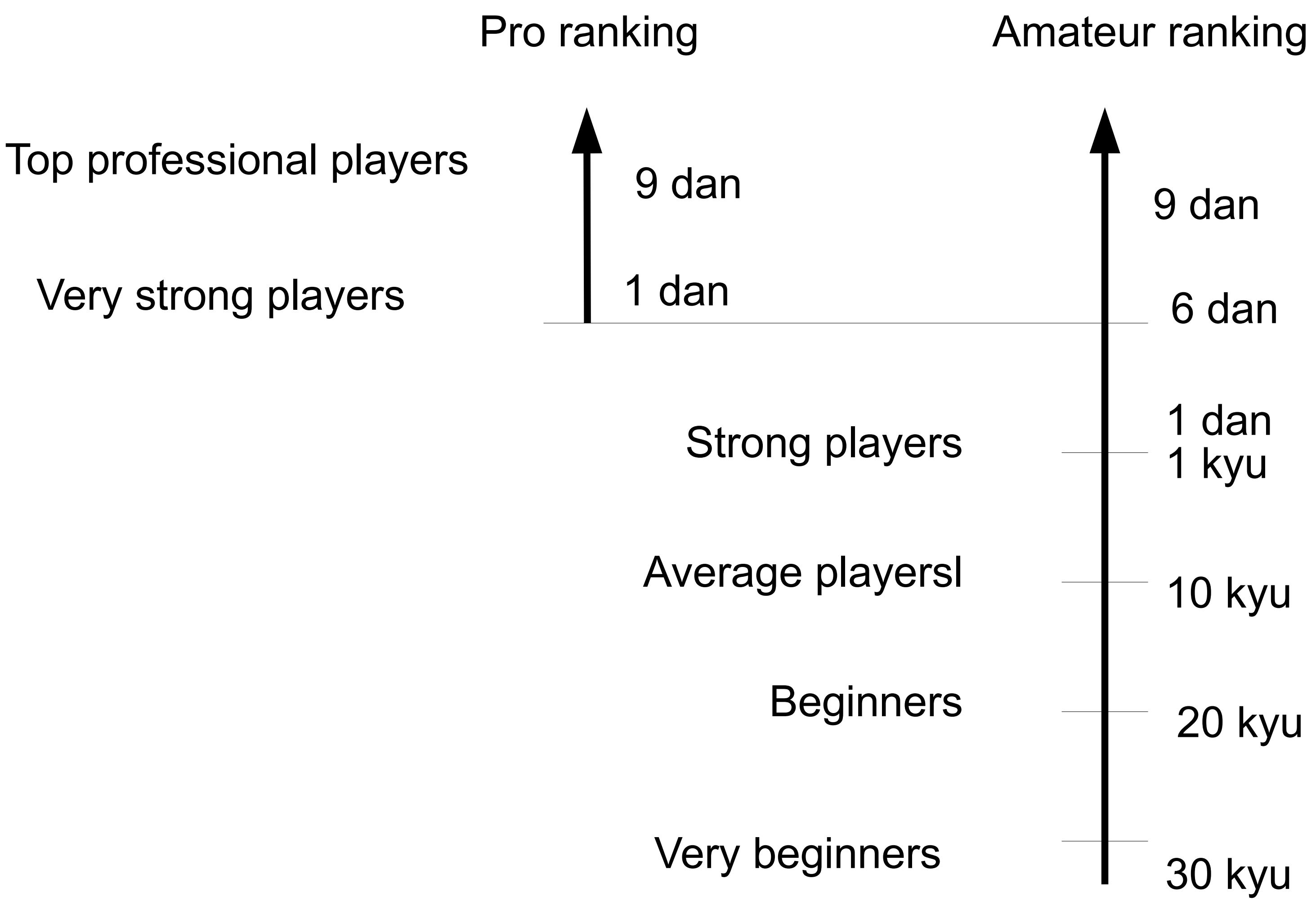
- The game ends when both players pass.
- One black (resp. white) point for each black (resp. white) stone and each black (resp. white) « eye » on the board.
- One black (resp. white) eye = an empty intersection surrounded by black (resp. white) stones.

A 9x9 game

- Scoring:
  - Black = 44
  - White = 37
  - Komi = 7.5
  - Score = -0.5
- White wins!



Go ranking: « kyu » and « dan »



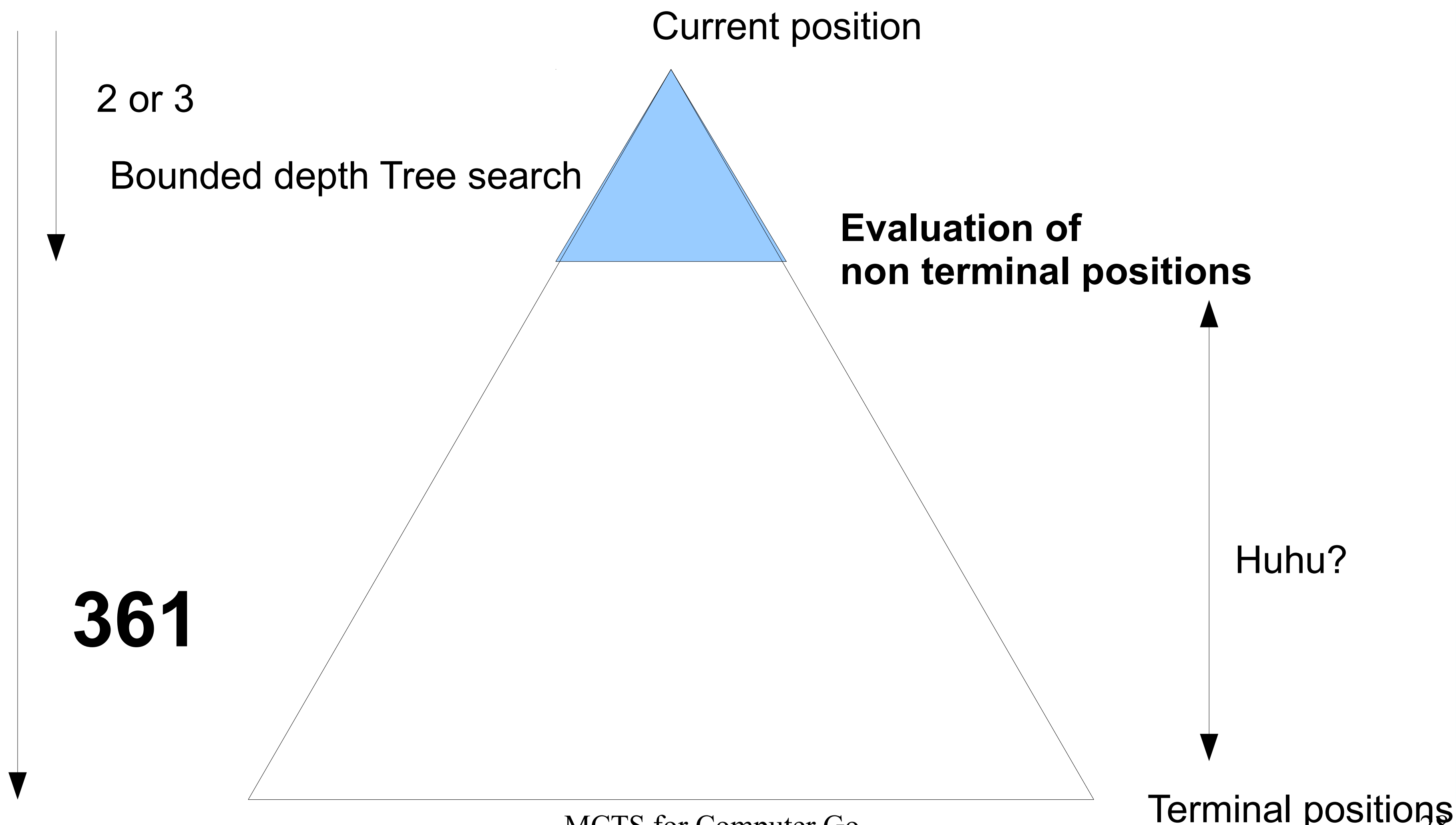
Computer Go (old history)

- First go program (Lefkovitz 1960)
- Zobrist hashing (Zobrist 1969)
- Interim2 (Wilcox 1979)
- Life and death model (Benson 1988)
- Patterns: Goliath (Boon 1990)
- Mathematical Go (Berlekamp 1991)
- Handtalk (Chen 1995)

The old approach

- Evaluation of non terminal positions
  - Knowledge-based
  - Breaking-down of a position into sub-positions
- Fixed-depth global tree search
  - Depth = 0 : action with the best value
  - Depth = 1: action leading to the position with the best evaluation
  - Depth > 1: alfa-beta or minmax

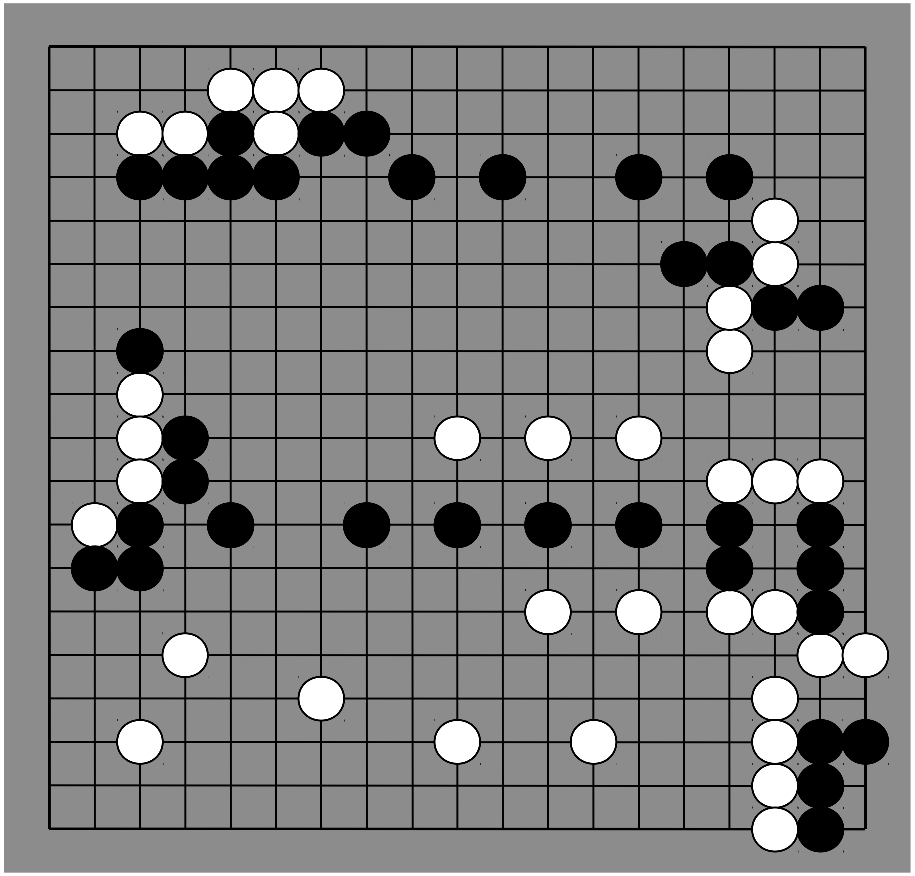
The old approach



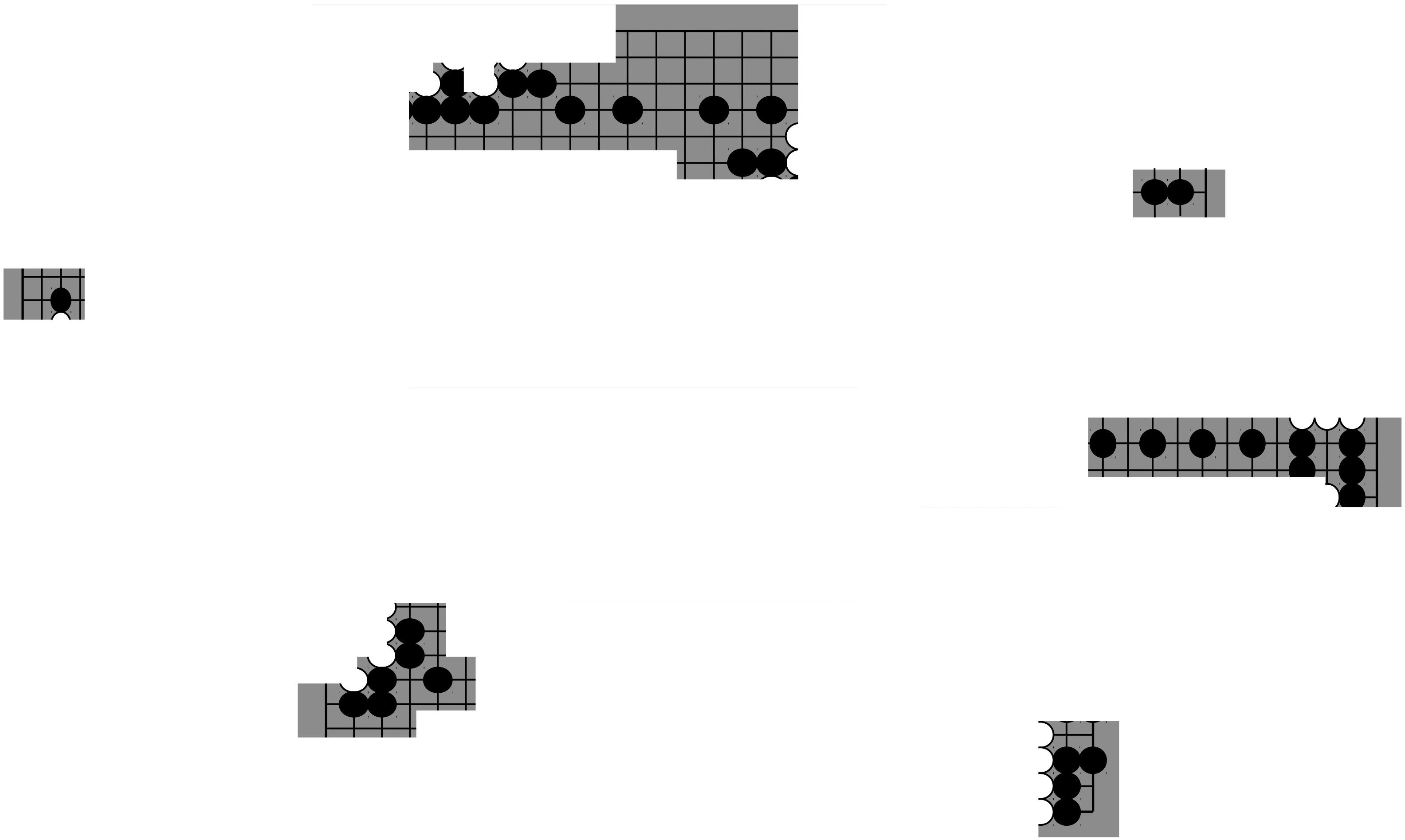
Position evaluation

- Break-down
  - Whole game (win/loss or score)
  - Goal-oriented sub-game
    - String capture
    - Connections, dividers, eyes, life and death
- Local searches
  - Alpha-beta and enhancements
  - Proof-number search

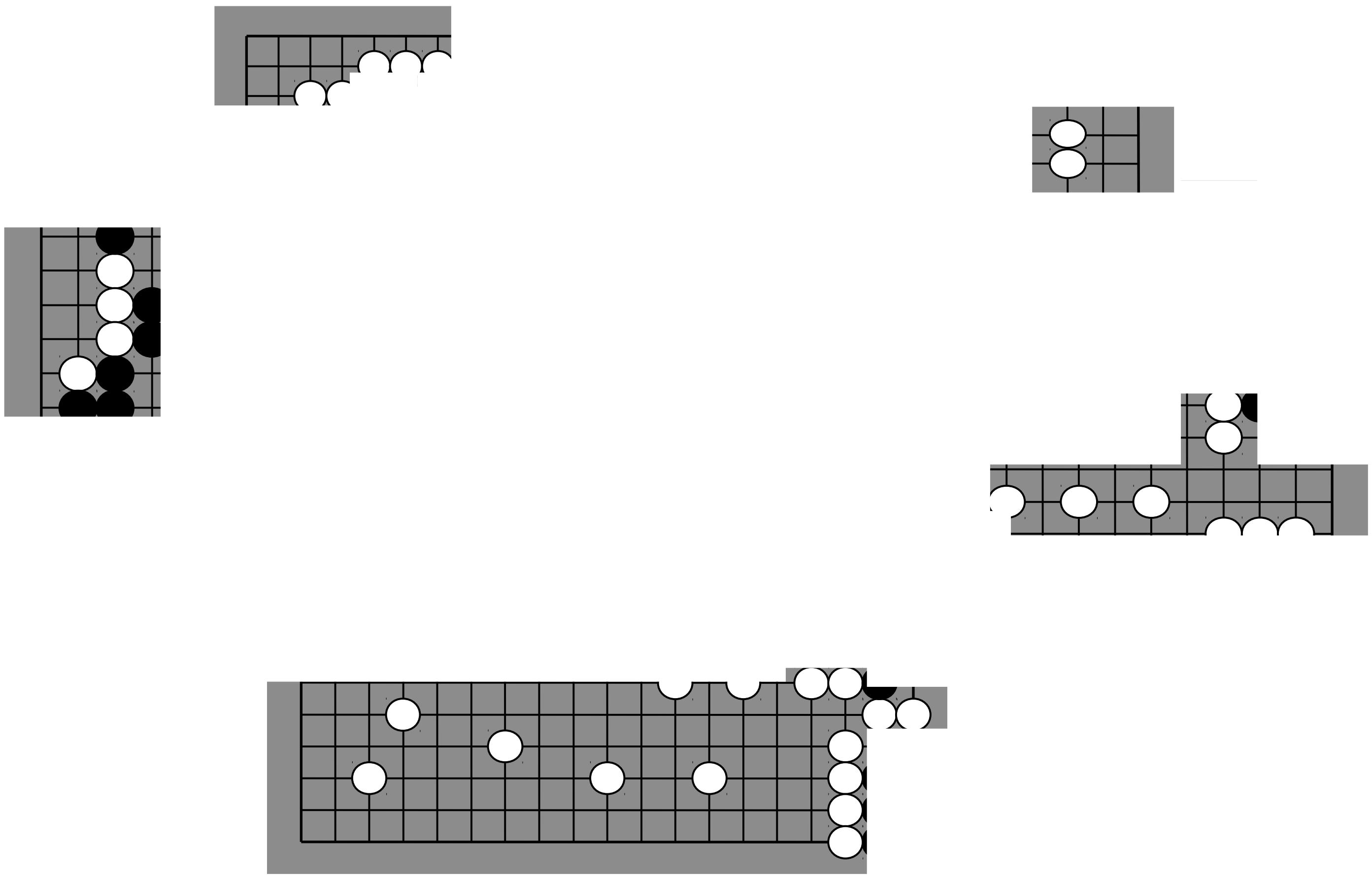
A 19x19 middle-game position



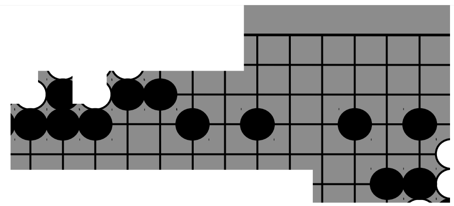
A possible black break-down



A possible white break-down

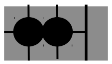


Possible local evaluations (1)



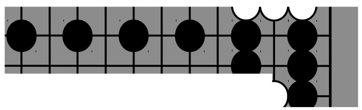
Alive and territory

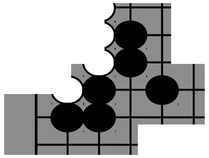
unstable



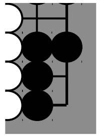
Not important

alive





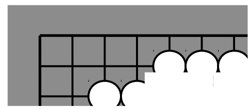
dead



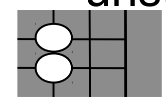
alive

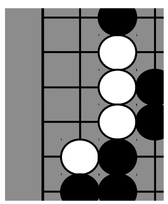
Possible local evaluations (2)

alive

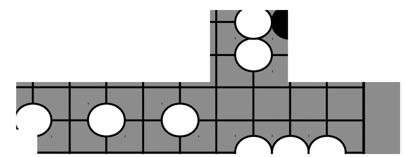


unstable



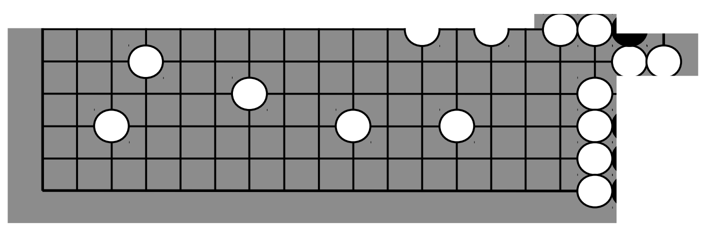


unstable



alive + big territory

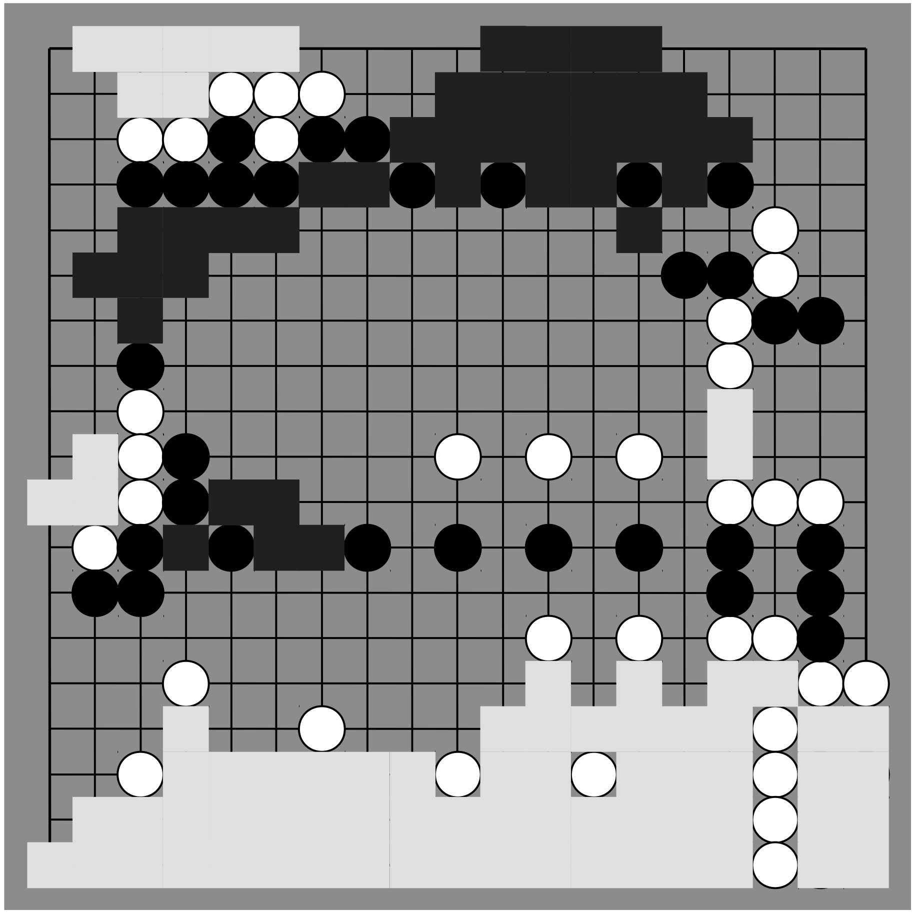
unstable



Position evaluation

- Local results
  - Obtained with local tree search
  - Result if white plays first (resp. black)
  - Combinatorial game theory (Conway)
  - Switches {a|b}, >, <, *, 0
- Global recomposition
  - move generation and evaluation
  - position evaluation

Position evaluation



Drawbacks (1/2)

- The break-down is not unique
- Performing a (wrong) local tree search on a (possibly irrelevant) local position
- Misevaluating the size of the local position
- Different kinds of local information
  - Symbolic (group: dead alive unstable)
  - Numerical (territory size, reduction, increase)

Drawbacks (2/2)

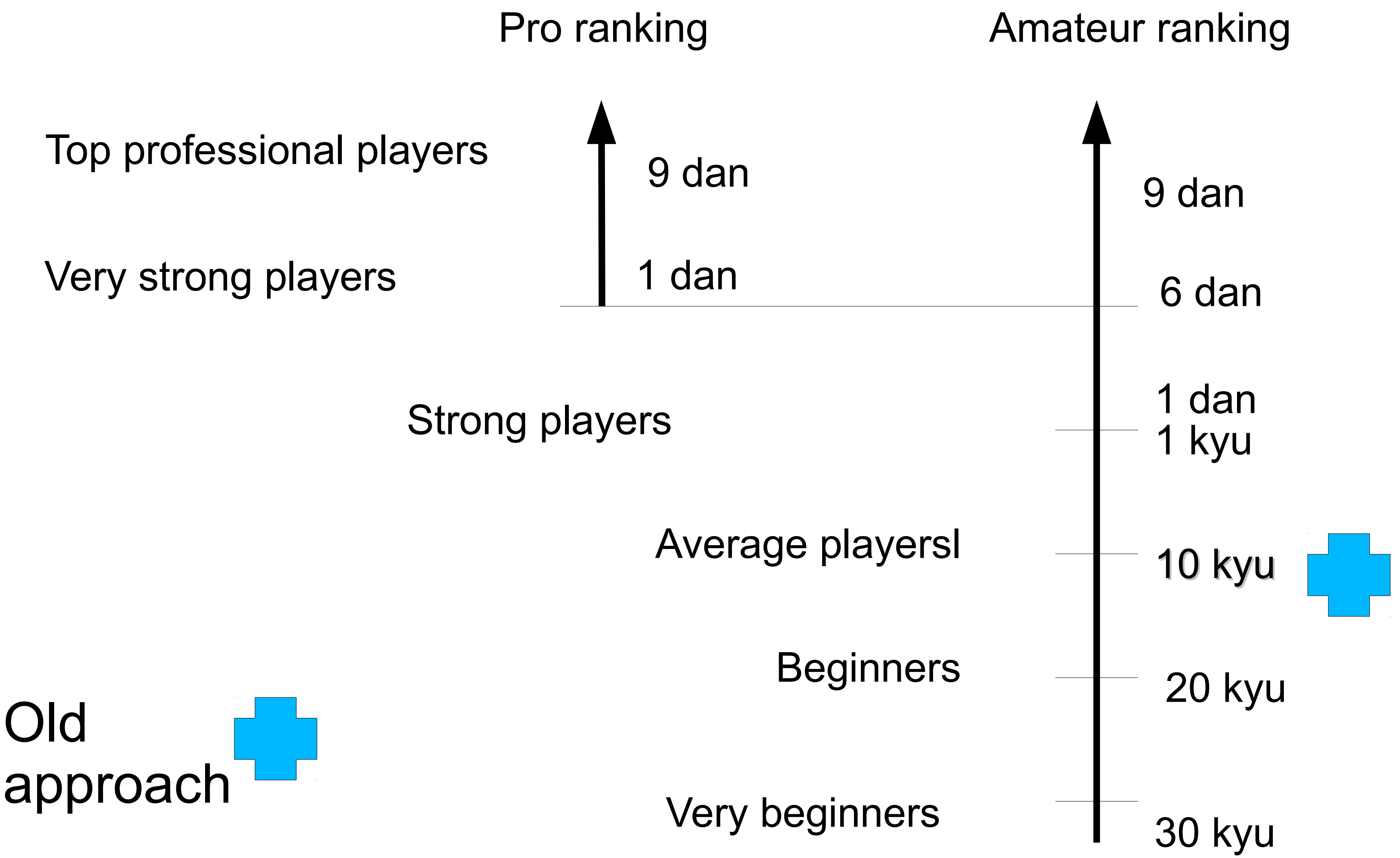
- Local positions interact
- Complicated
- Domain-dependent knowledge
- Need of human expertise
- Difficult to program and maintain
- Holes of knowledge
- Erratic behaviour

Upsides

- Feasible on 1990's computers
- Execution is fast

- Some specific local tree searches are accurate and fast

The old approach



End of part one!

- Next: the Monte-Carlo approach...

The Monte-Carlo (MC) approach

- Games containing chance
  - Backgammon (Tesauro 1989)

- Games with hidden information
  - Bridge (Ginsberg 2001)
  - Poker (Billings & al. 2002)
  - Scrabble (Sheppard 2002)

The Monte-Carlo approach

- Games with complete information
  - A general model (Abramson 1990)

- Simulated annealing Go
  - (Brügmann 1993)
  - 2 sequences of moves
  - « all moves as first » heuristic
  - Gobble on 9x9

The Monte-Carlo approach

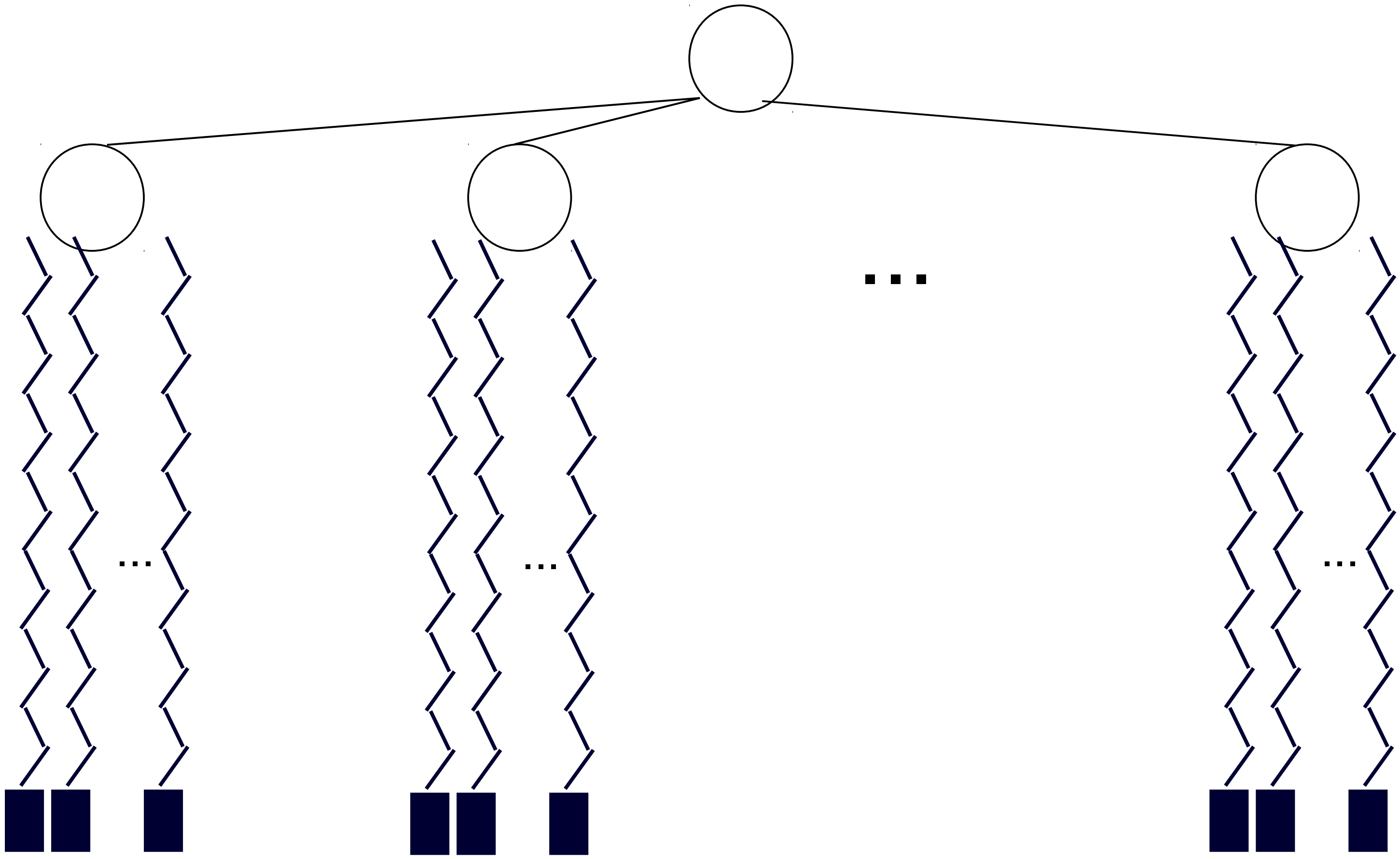
- Position evaluation:

```
Launch N random games
Evaluation = mean value of outcomes
```

- Depth-one MC algorithm:

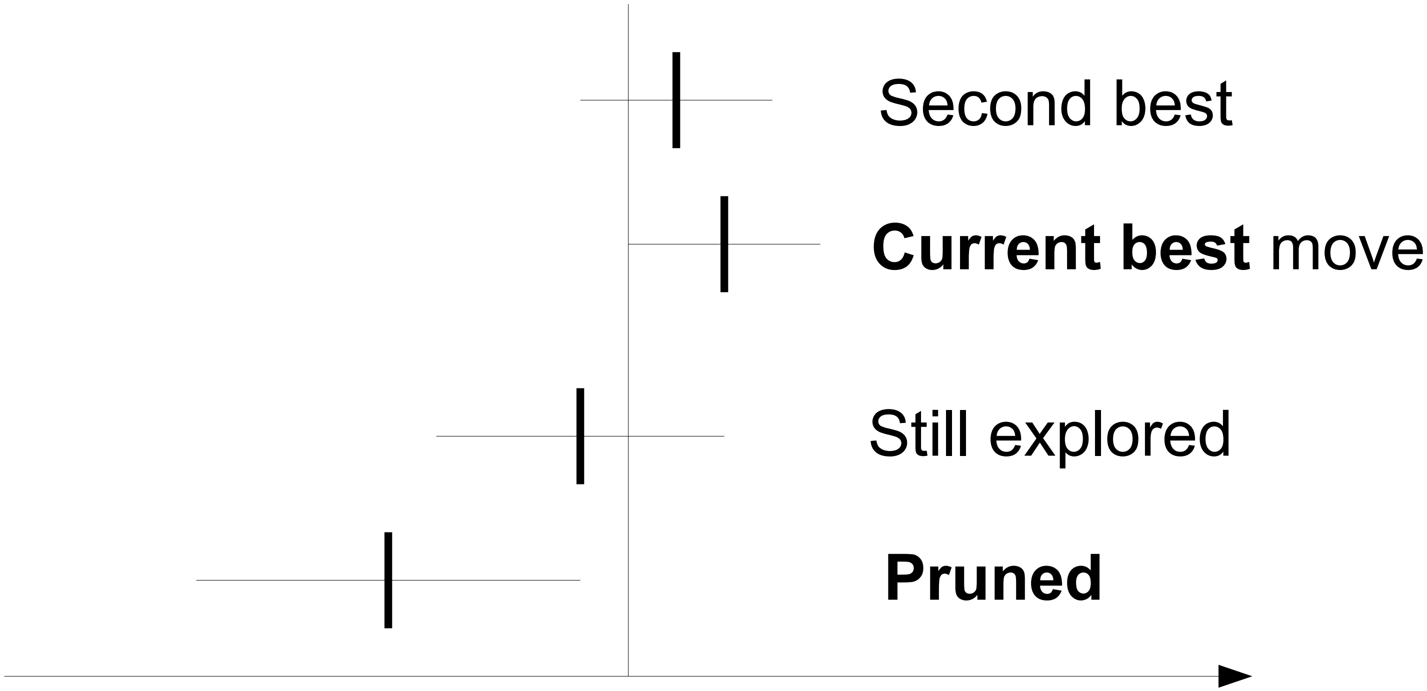
```
For each move m {
     Play m on the ref position
     Launch N random games
     Move value (m) = mean value
}
```

Depth-one Monte-Carlo



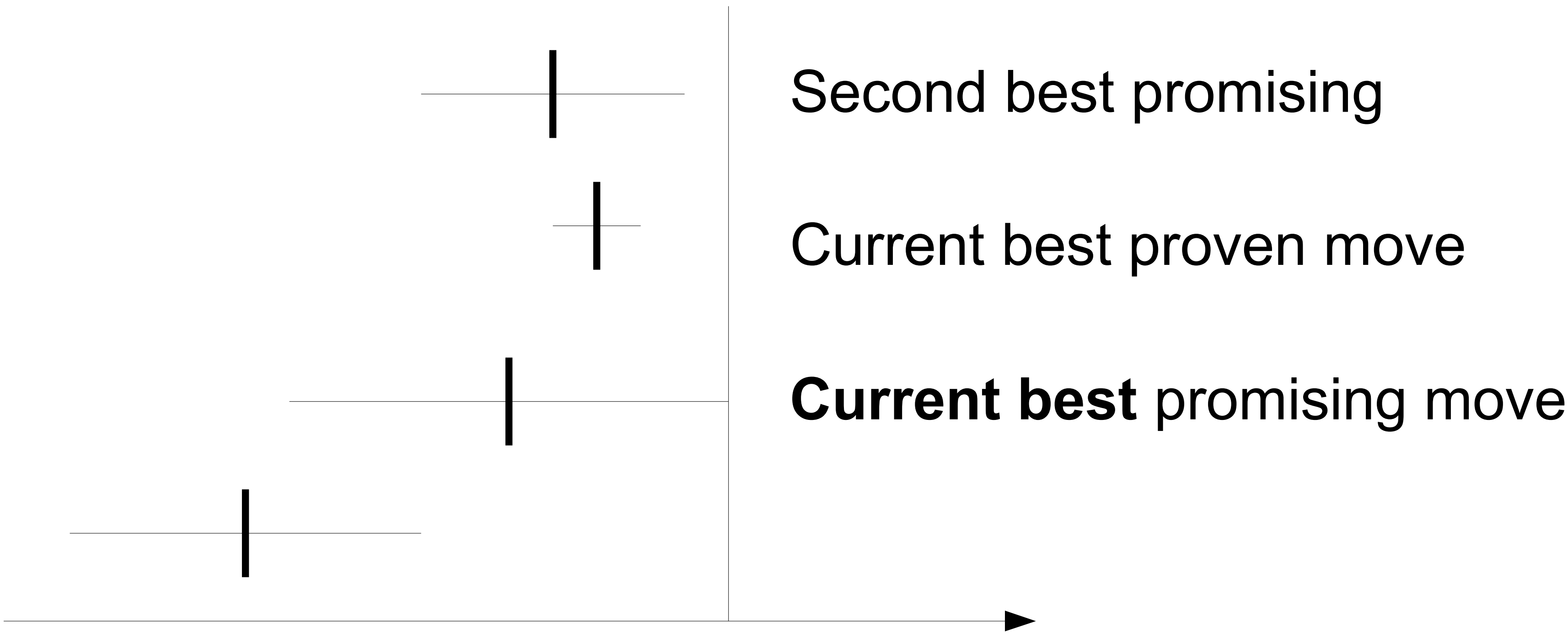
Progressive pruning

- (Billings 2002, Sheppard 2002, Bouzy & Helmstetter 2003)

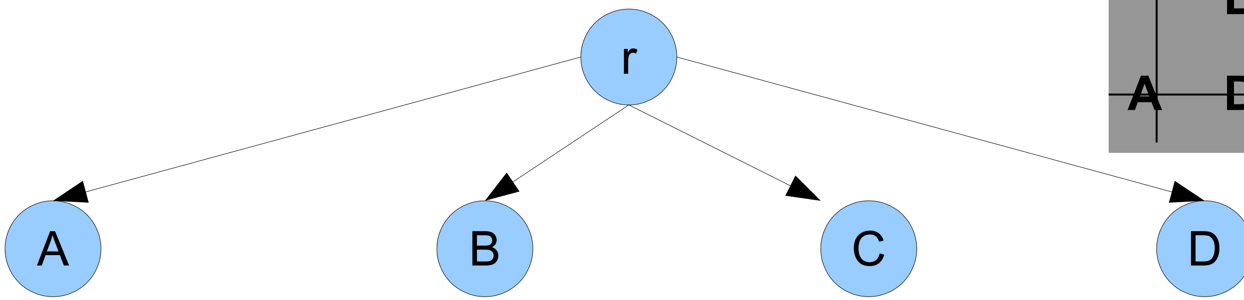


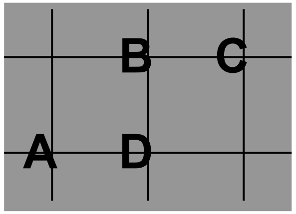
Upper bound

- Optimism in face of uncertainty
  - Intestim (Kaelbling 1993),
  - UCB multi-armed bandit (Auer & al 2002)

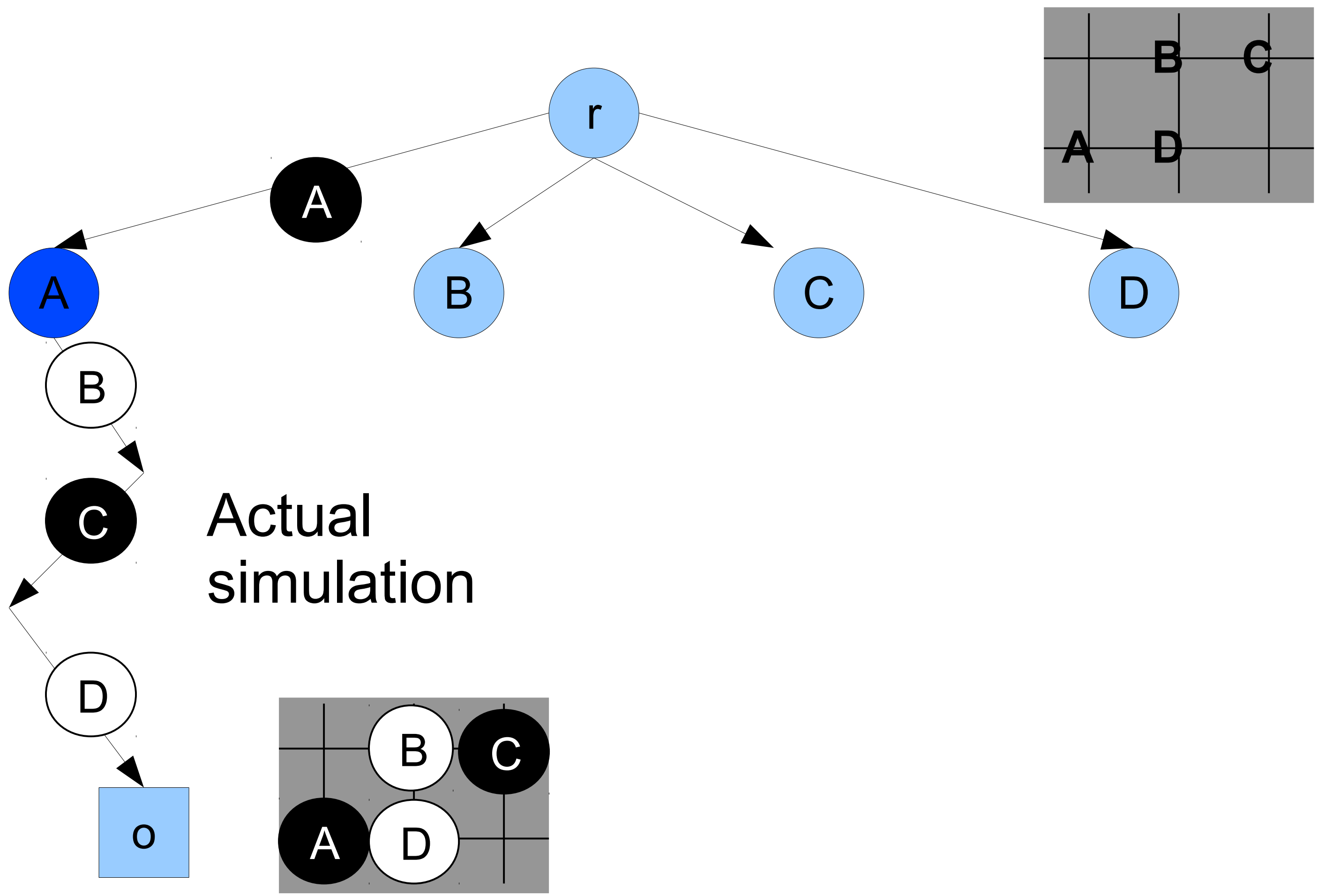


All-moves-as-first heuristic (1/3)

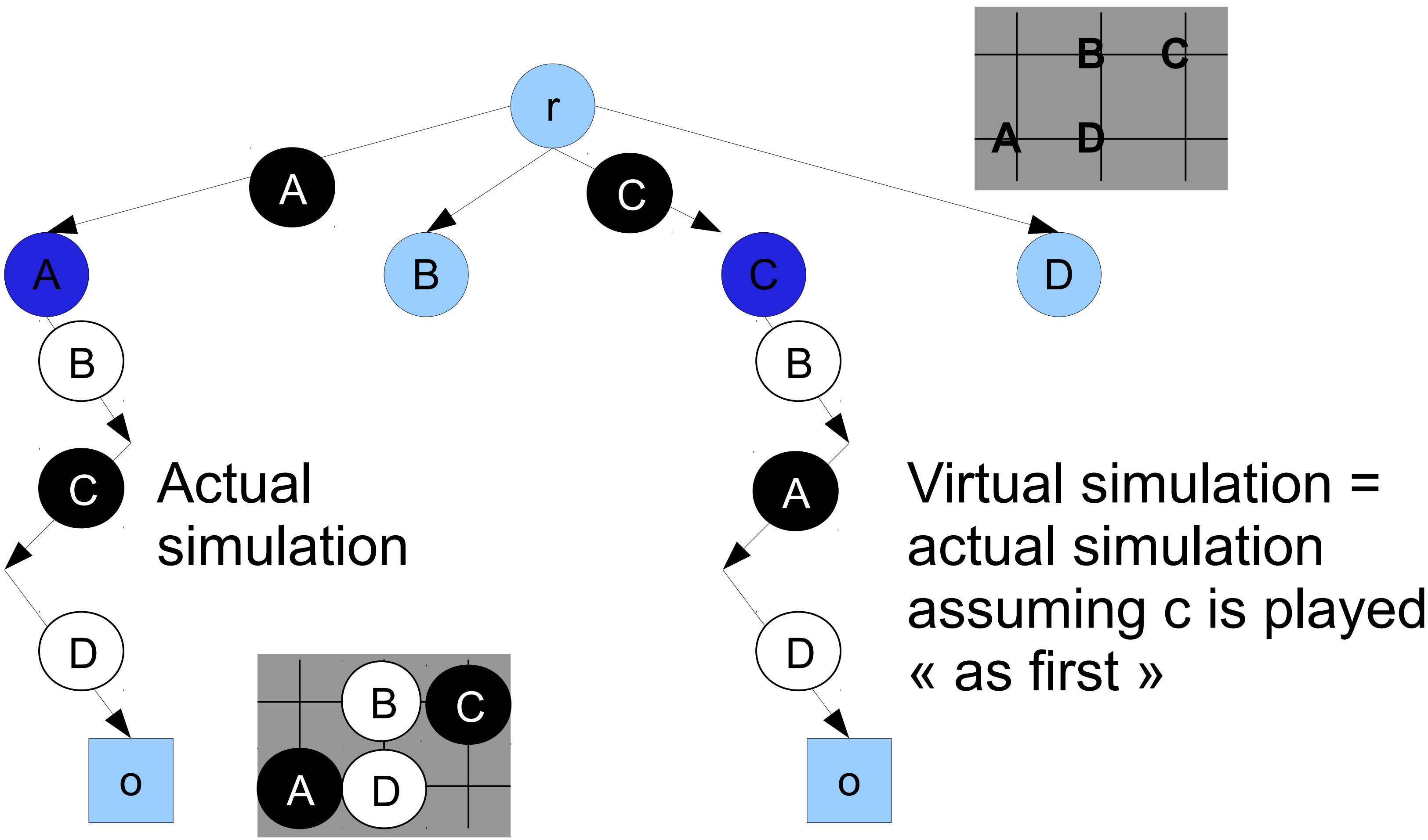




All-moves-as-first heuristic (2/3)



All-moves-as-first heuristic (3/3)

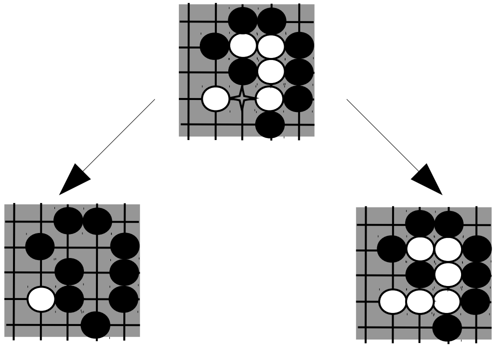


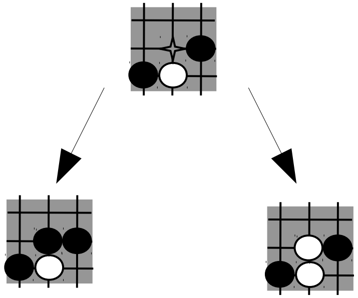
The Monte-Carlo approach

- Upsides
  - Robust evaluation
  - Global search
  - Move quality increases with computing power
- Way of playing
  - Good strategical sense but weak tactically
- Easy to program
  - Follow the rules of the game
  - No break-down problem

Monte-Carlo and knowledge

- Pseudo-random simulations using Go knowledge (Bouzy 2003)
  - Moves played with a probability depending on specific domain-dependent knowledge
- 2 basic concepts
  - string capture 3x3 shapes





Monte-Carlo and knowledge

- Results are impressive
  - MC(random) << MC(pseudo random)
  - Size 9x9 13x13 19x19
  - % wins 68 93 98
- Other works on simulations
  - Patterns in MoGo, proximity rule (Wang & al 2006)
  - Simulation balancing (Silver & Tesauro 2009)

Monte-Carlo and knowledge

- Pseudo-random player
  - 3x3 pattern urgency table with 3^8 patterns
  - Few dizains of relevant patterns only
  - Patterns gathered by
    - Human expertise
    - Reinforcement Learning (Bouzy & Chaslot 2006)
- Warning
  - p1 better than p2 does not mean MC(p1) better than MC(p2)

Monte-Carlo Tree Search (MCTS)

- How to integrate MC and TS ?
- UCT = UCB for Trees
  - (Kocsis & Szepesvari 2006)
  - Superposition of UCB (Auer & al 2002)
- MCTS
  - Selection, expansion, updating (Chaslot & al) (Coulom 2006)
  - Simulation (Bouzy 2003) (Wang & Gelly 2006)

MCTS (1/2)

```
while (hasTime) {
     playOutTreeBasedGame()
     expandTree()
     outcome = playOutRandomGame()
     updateNodes(outcome)
}
```

then choose the node with...

... the best mean value

... the **highest visit** number

MCTS (2/2)

```
PlayOutTreeBasedGame() {
     node = getNode(position)
     while (node) {
          move=selectMove(node)
          play(move)
          node = getNode(position)
     }
}
```

UCT move selection

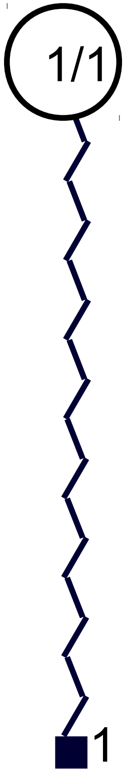
- Move selection rule to browse the tree:

  move=argmax (s*mean + C*sqrt(log(t)/n))

- Mean value for exploitation
  - s (=+-1): color to move
- UCT bias for exploration
  - C: constant term set up by experiments
  - t: number of visits of the parent node
  - n: number of visits of the current node

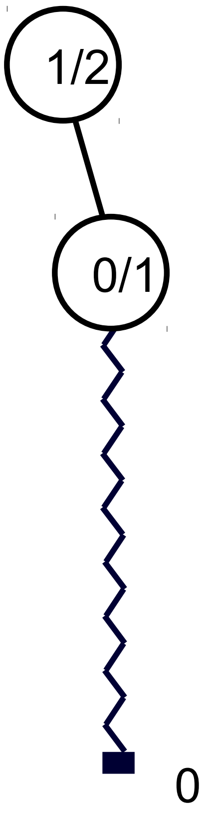
Example

- 1 iteration



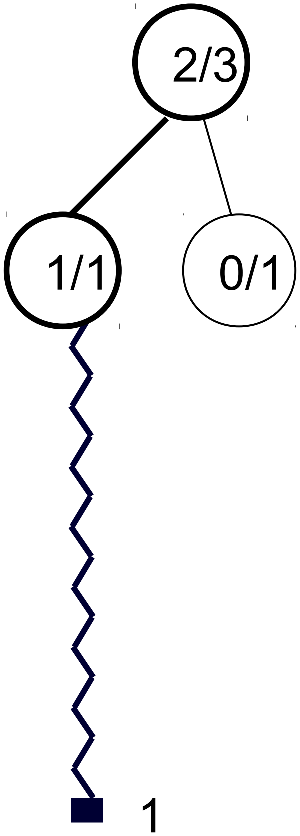
Example

- 2 iterations



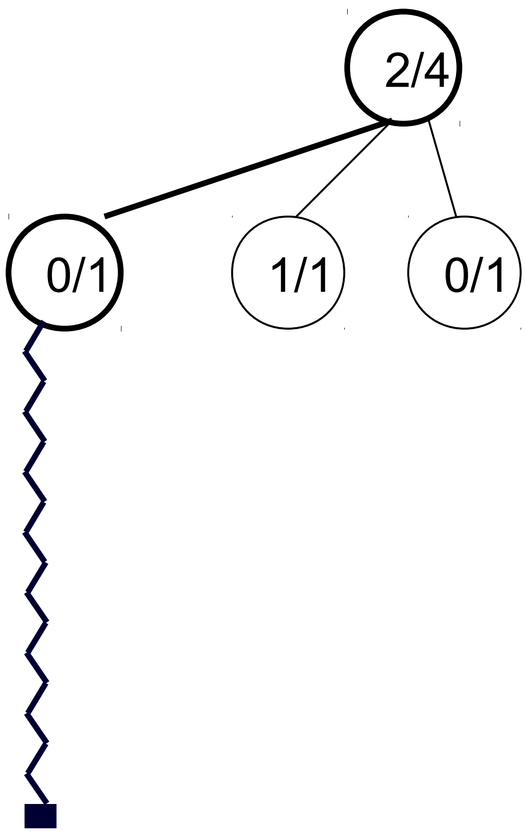
Example

- 3 iterations



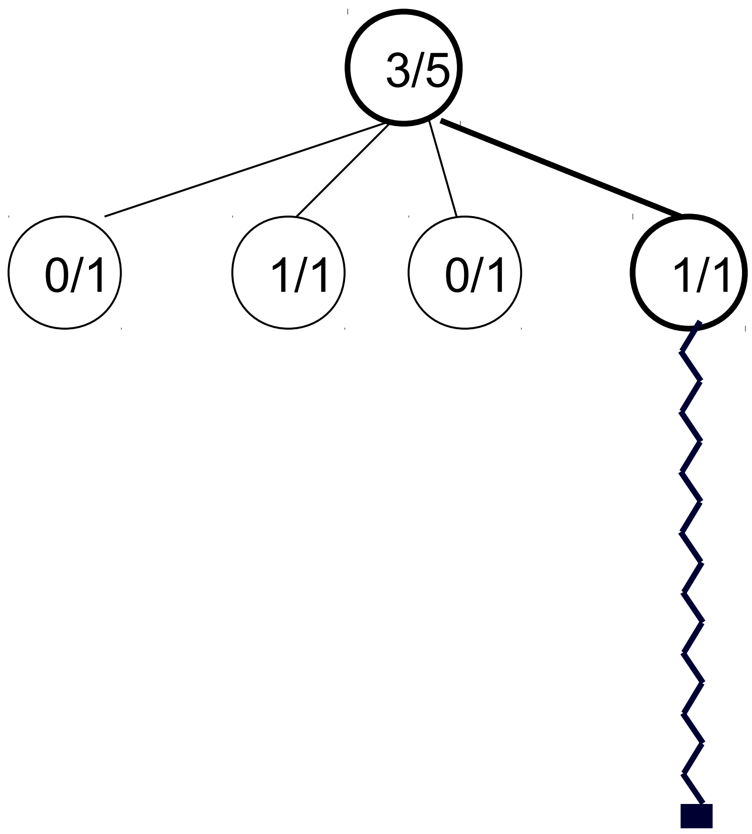
Example

- 4 iterations



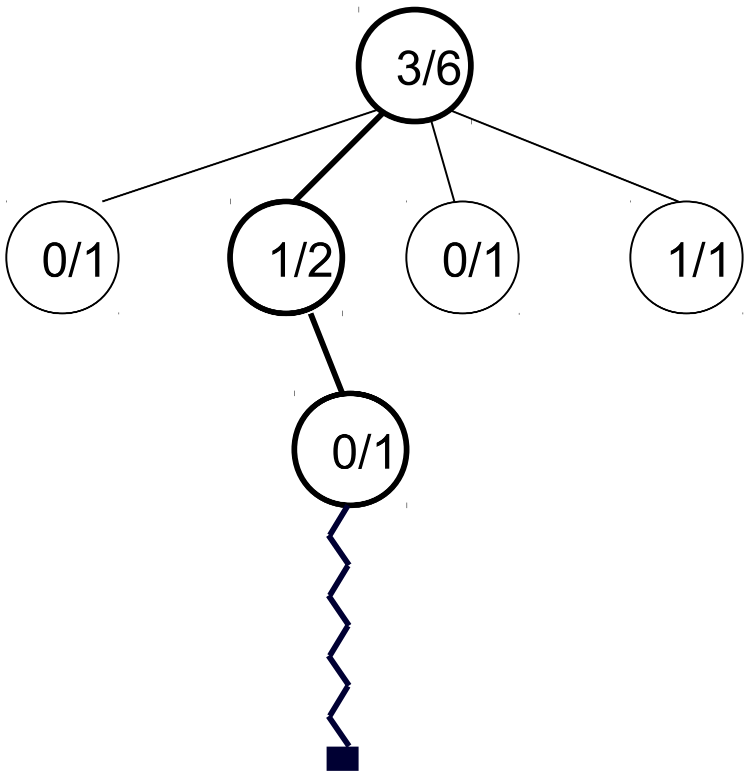
Example

- 5 iterations



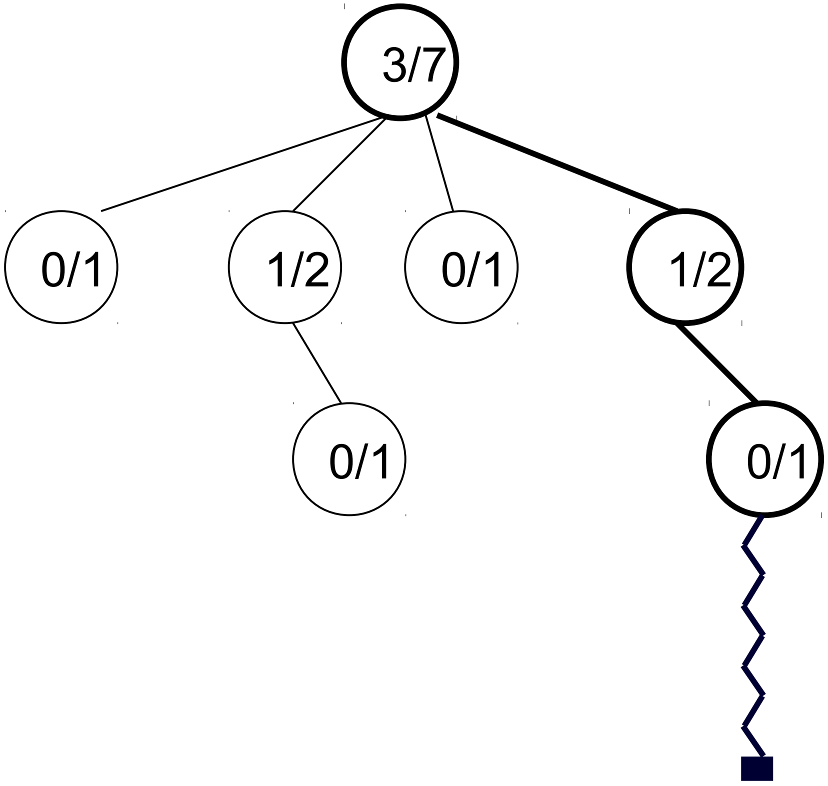
Example

- 6 iterations



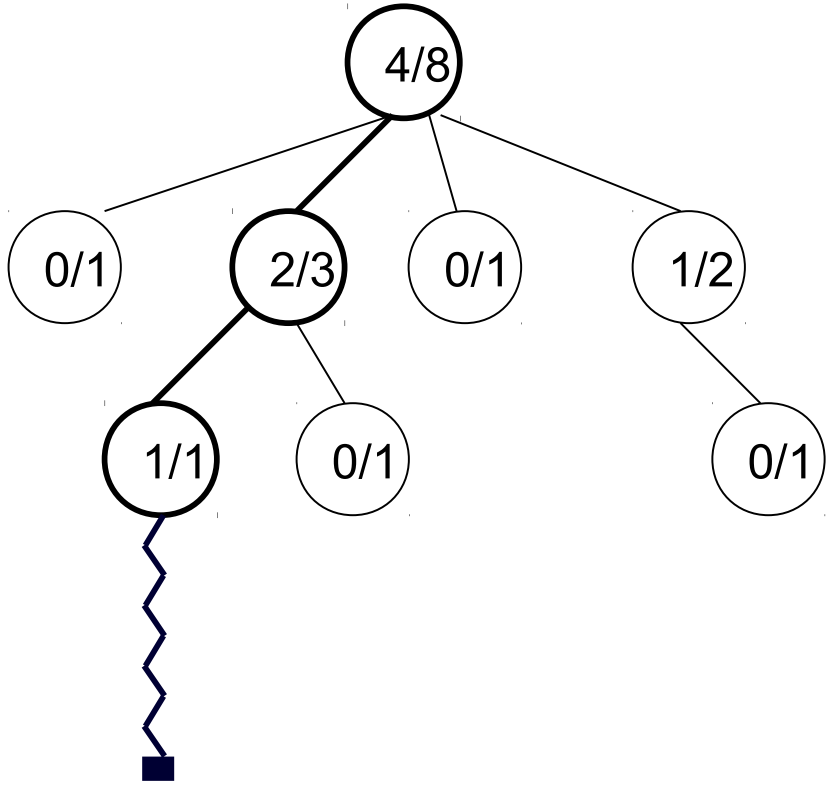
Example

- 7 iterations



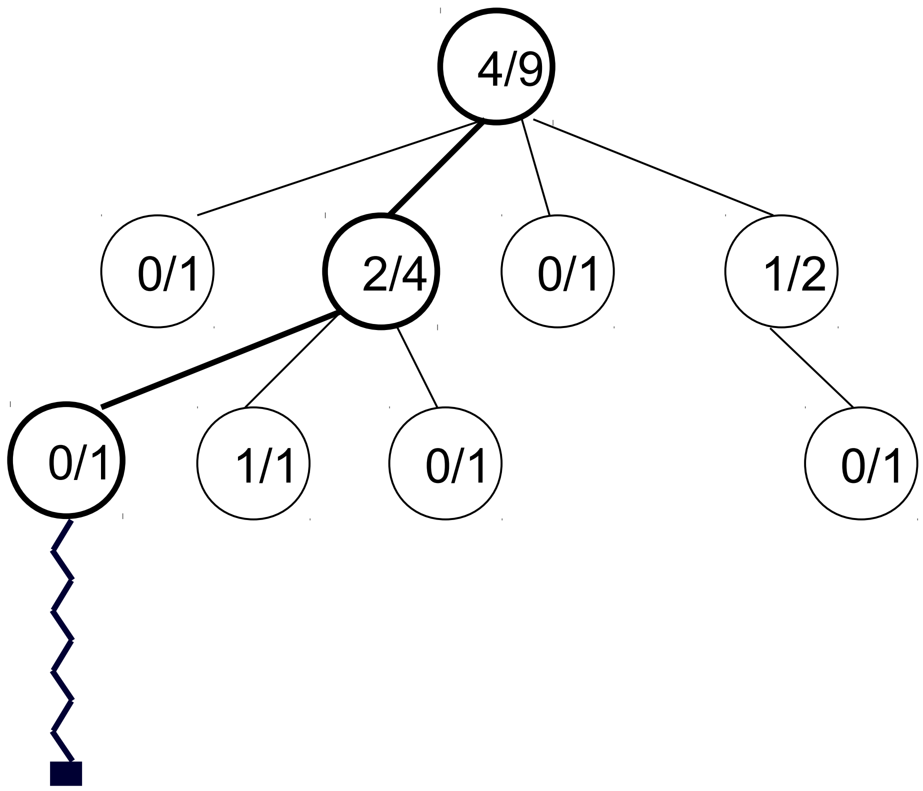
Example

- 8 iterations



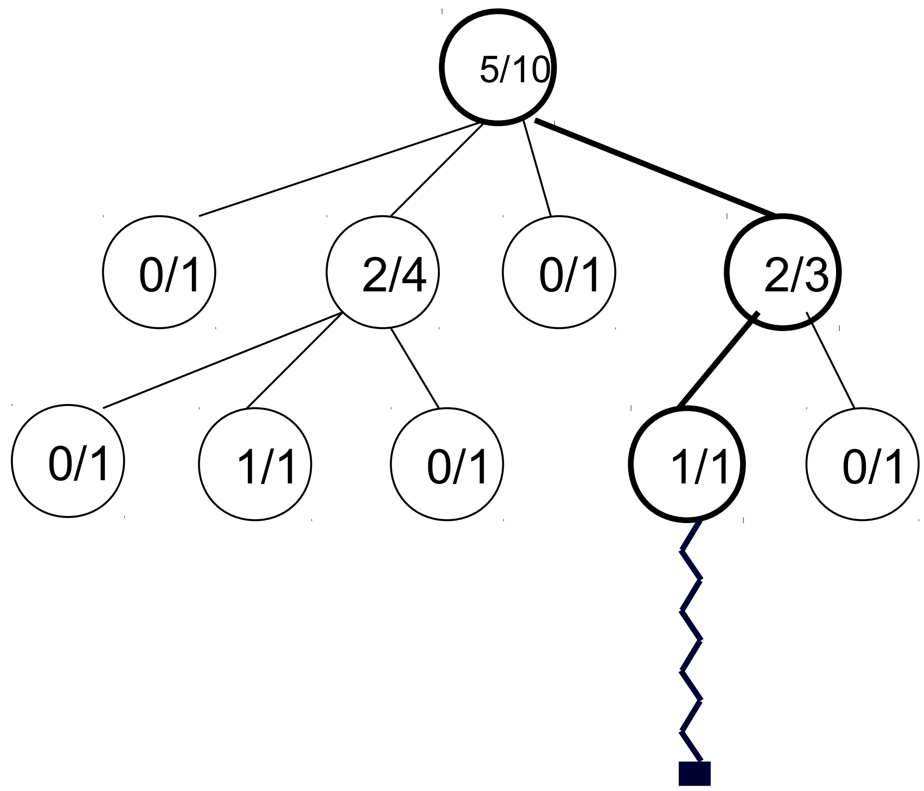
Example

- 9 iterations



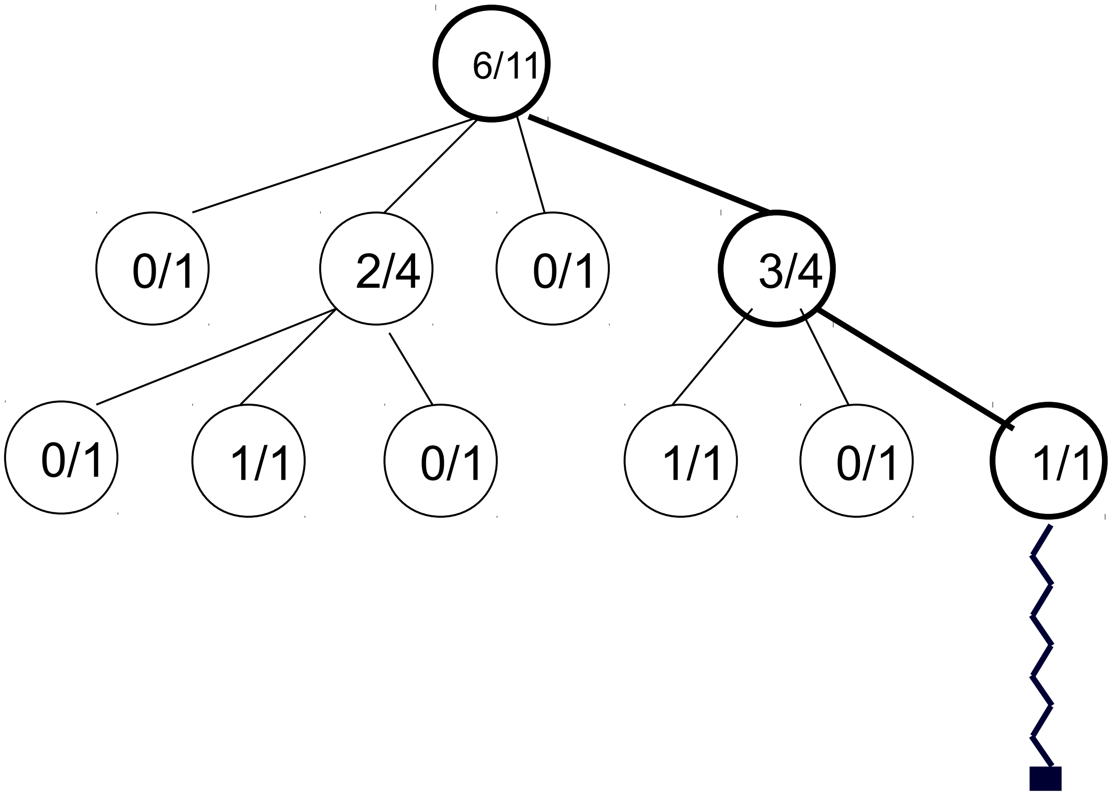
Example

- 10 iterations



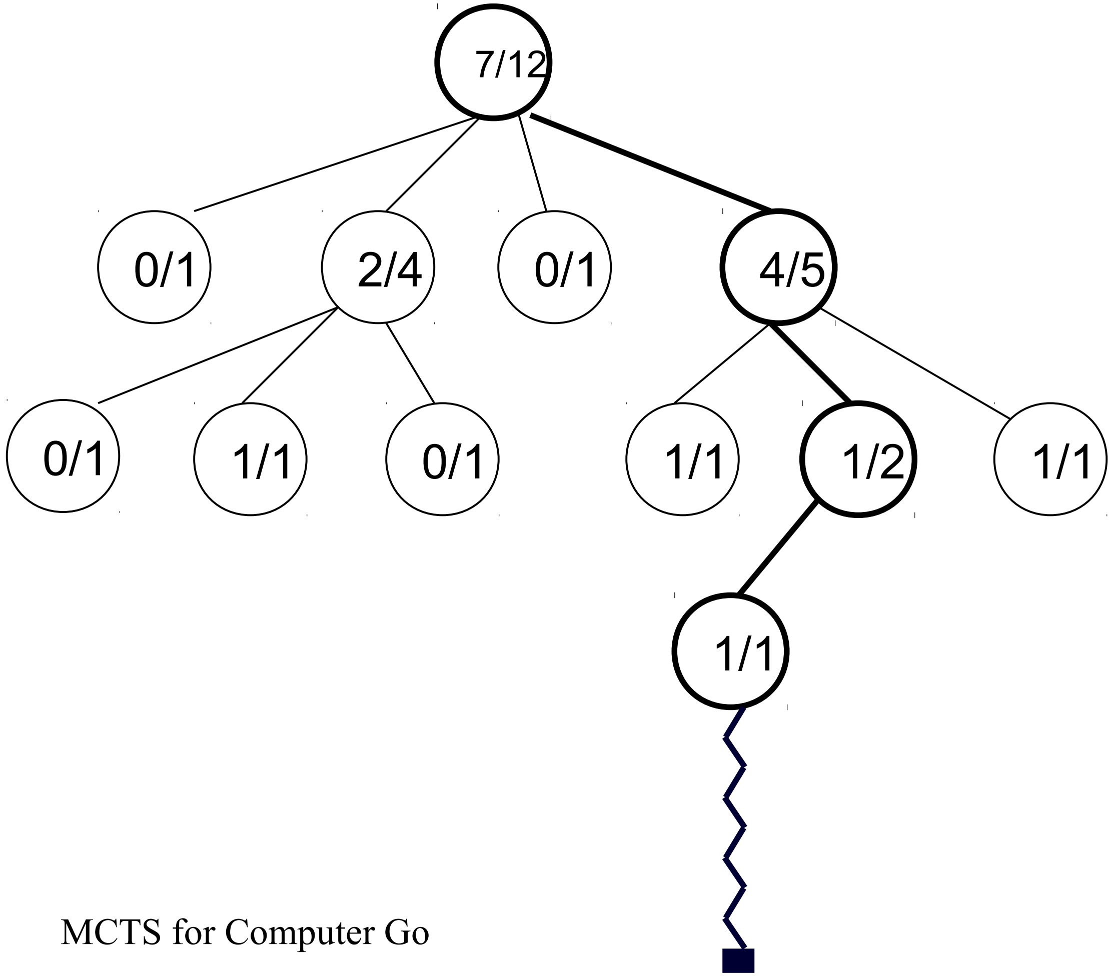
Example

- 11 iterations



Example

- 12 iterations



Example

- Clarity
  - C = 0
- Notice
  - with C != 0 a node cannot stay unvisited
  - min or max rule according to the node depth
  - not visited children have an infinite mean
- Practice
  - Mean initialized optimistically

MCTS enhancements

- The raw version can be enhanced
  - Tuning UCT C value
  - Outcome = score or win loss info (+1/-1)
  - Doubling the simulation number
  - RAVE
  - Using Go knowledge
    - In the tree or in the simulations
  - Speed-up
    - Optimizing, pondering, parallelizing

Assessing an enhancement

- Self-play
  - The new version vs the reference version
  - % wins with few hundred games
  - 9x9 (or 19x19 boards)
- Against differently designed programs
  - GTP (Go Text Protocol)
  - CGOS (Computer Go Operating System)
- Competitions

Move selection formula tuning

- Using UCB
  - Best value for C ?
  - 60-40%
- Using « UCB-tuned » (Auer & al 2002)
  - C replaced by min(1/4,variance)
  - 55-45%

Exploration vs exploitation

- General idea: explore at the beginning and exploit in the end of thinking time
- Diminishing C linearly in the remaining time
  - (Vermorel & al 2005)
  - 55-45%
- At the end:
  - Argmax over the mean value or over the number of visits ?
  - 55-45%

Kind of outcome

- 2 kinds of outcomes
  - Score (S) or win loss information (WLI) ?
  - Probability of winning or expected score ?
  - Combining both (S+WLI) (score +45 if win)
- Results
  - WLI vs S 65-35%
  - S+WLI vs S 65-35%

Doubling the number of simulations

- N = 100,000
- Results
  - 2N vs N   60-40%
  - 4N vs 2N   58-42%

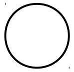
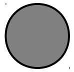
Tree management

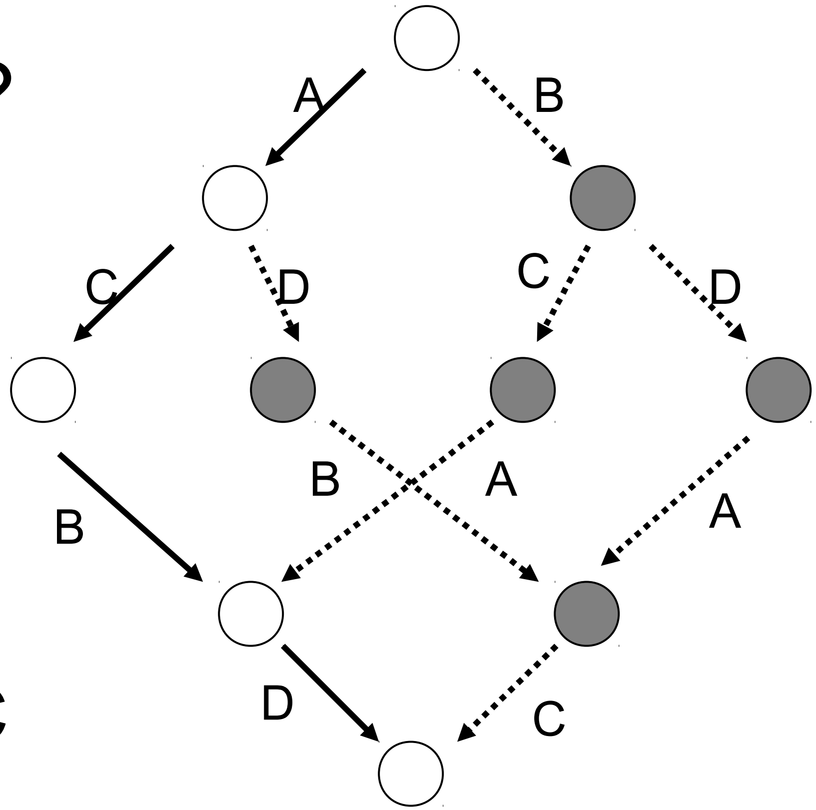
- Transposition tables
  - Tree -> Directed Acyclic Graph (DAG)
  - Different sequences of moves may lead to the same position
  - Interest for MC Go: merge the results
  - Result: 60-40%
- Keeping the tree from one move to the next
  - Result: 65-35%

RAVE (1/3)

- Rapid Action Value Estimation
  - Mogo 2007
  - Use the AMAF heuristic (Brugmann 1993)
  - There are « many » virtual sequences that are transposed from the actually played sequence
- Result:
  - 70-30%

RAVE (2/3)

- AMAF heuristic
- Which nodes to update?
- Actual
  - Sequence ACBD
  - Nodes 
- Virtual
  - BCAD, ADBC, BDAC
  - Nodes 



RAVE (3/3)

- 3 variables
  - Usual mean value M_u
  - AMAF mean value M_{amaf}
  - $M = \beta\ M_{amaf} + (1-\beta)\ M_u$
  - $\beta = \mathrm{sqrt}(k/(k+3N))$
  - K set up experimentally
- M varies from M_{amaf} to M_u

Knowledge in the simulations

- High urgency for...
  - capture/escape 55-45%
  - 3x3 patterns 60-40%
  - Proximity rule 60-40%
- Mercy rule
  - Interrupt the game when the difference of captured stones is greater than a threshold (Hillis 2006)
  - 51-49%

Knowledge in the tree

- Virtual wins for good looking moves
- Automatic acquisition of patterns of pro games (Coulom 2007) (Bouzy & Chaslot 2005)
- Matching has a high cost
- Progressive widening (Chaslot & al 2008)
- Interesting under strong time constraints
- Result: 60-40%

Speeding up the simulations

- Fully random simulations (2007)
  - 50,000 game/second (Lew 2006)
  - 20,000 (commonly eared)
  - 10,000 (my program)
- Pseudo-random
  - 5,000 (my program in 2007)
- Rough optimization is worthwhile

Pondering

- Think on the opponent time
  - 55-45%
  - Possible doubling of thinking time
  - The move of the opponent may not be the planned move on which you think
  - Side effect: play quickly to think on the opponent time

Summing up the enhancements

- MCTS with all enhancements vs raw MCTS
  - Exploration and exploitation: 60-40%
  - Win/loss outcome: 65-35%
  - Rough optimization of simulations 60-40%
  - Transposition table 60-40%
  - RAVE 70-30%
  - Knowledge in the simulations 70-30%
  - Knowledge in the tree 60-40%
  - Pondering 55-45%
  - Parallelization 70-30%
- Result: 99-1%

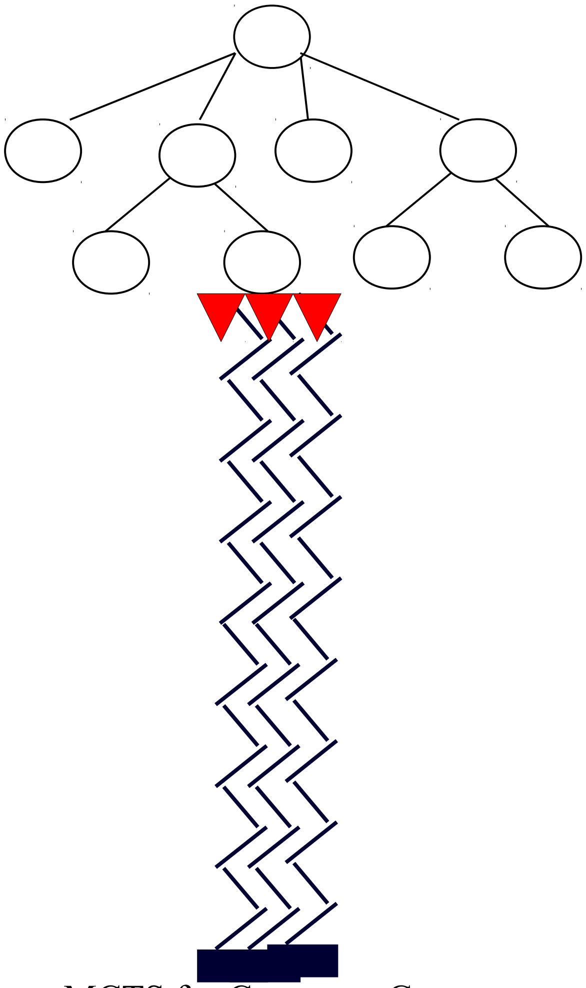
Parallelization

- Computer Chess: Deep Blue
- Multi-core computer
  - Symmetric MultiProcessor (SMP)
  - one thread per processor
  - shared memory, low latency
  - mutual exclusion (mutex) mechanism
- Cluster of computers
  - Message Passing Information (MPI)

Parallelization

```
while (hasTime) {
      playOutTreeBasedGame()
      expandTree()
      outcome = playOutRandomGame()
      updateNodes(outcome)
}
```

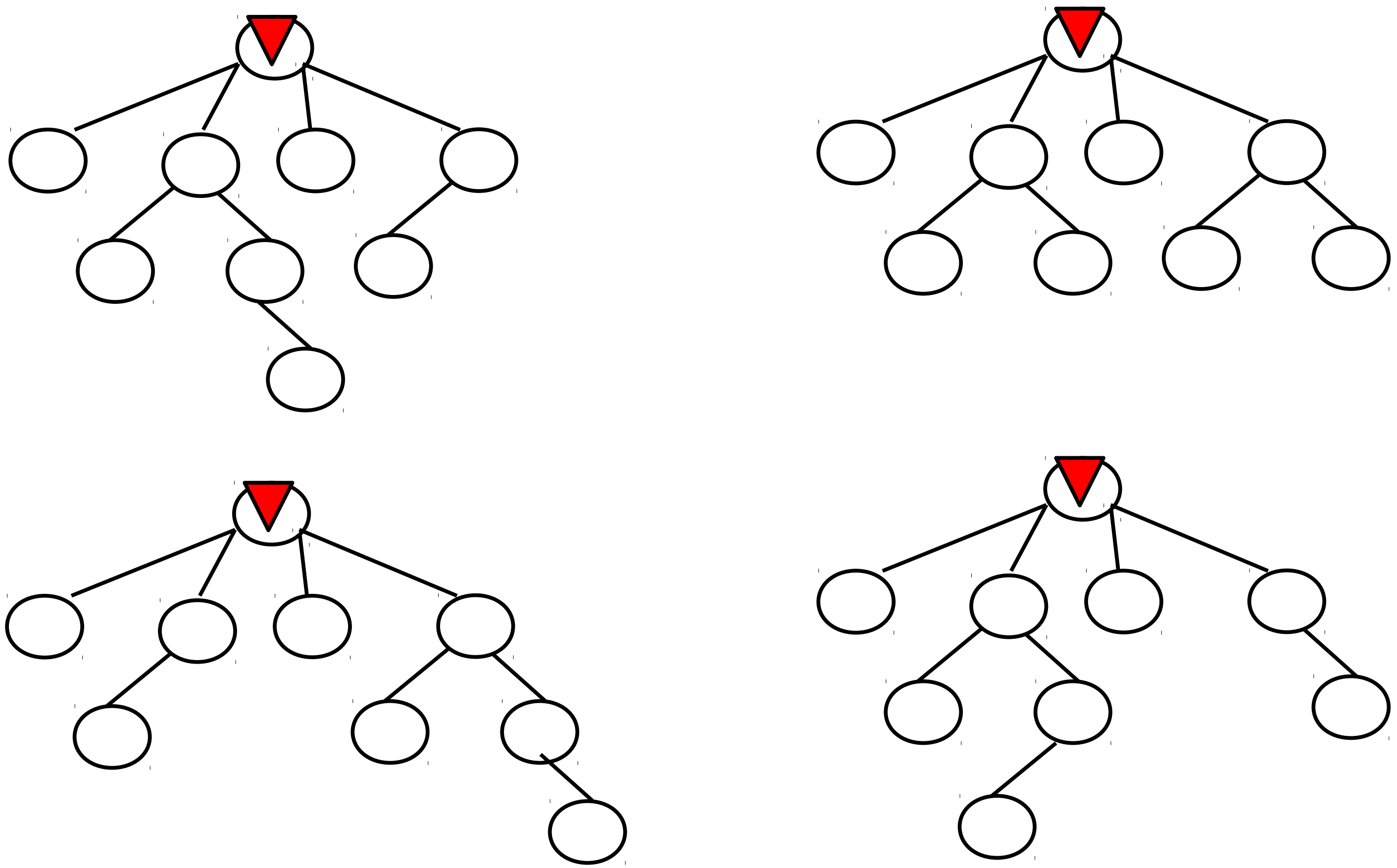
Leaf parallelization



Leaf parallelization

- (Cazenave Jouandeau 2007)
- Easy to program
- Drawbacks
  - Wait for the longest simulation
  - When part of the simulation outcomes is a loss, performing the remaining may not be a relevant strategy.

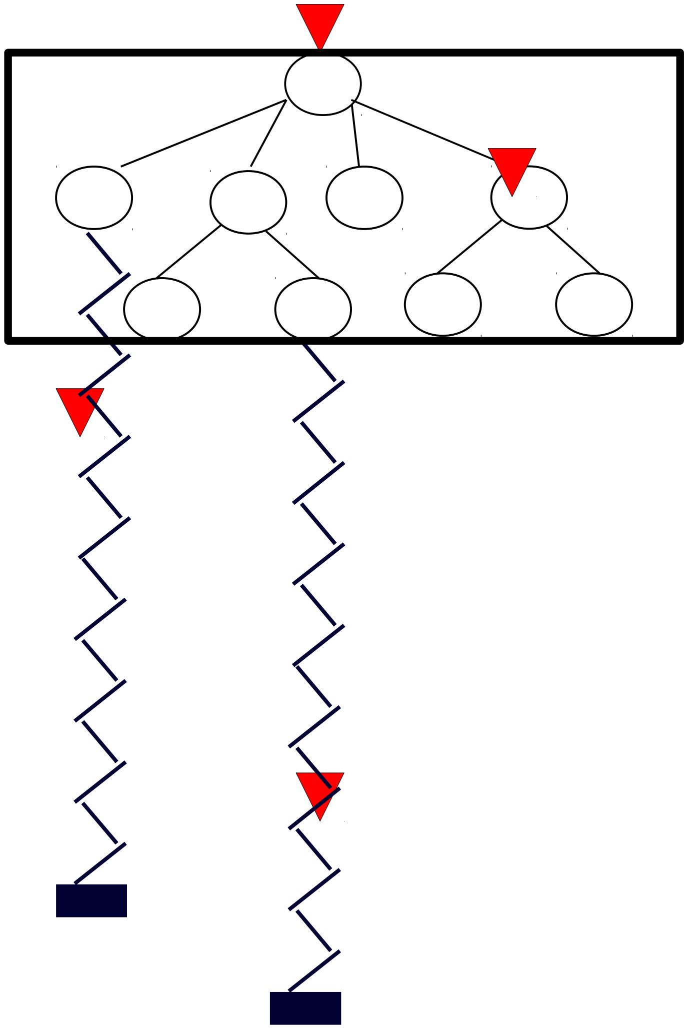
Root parallelization



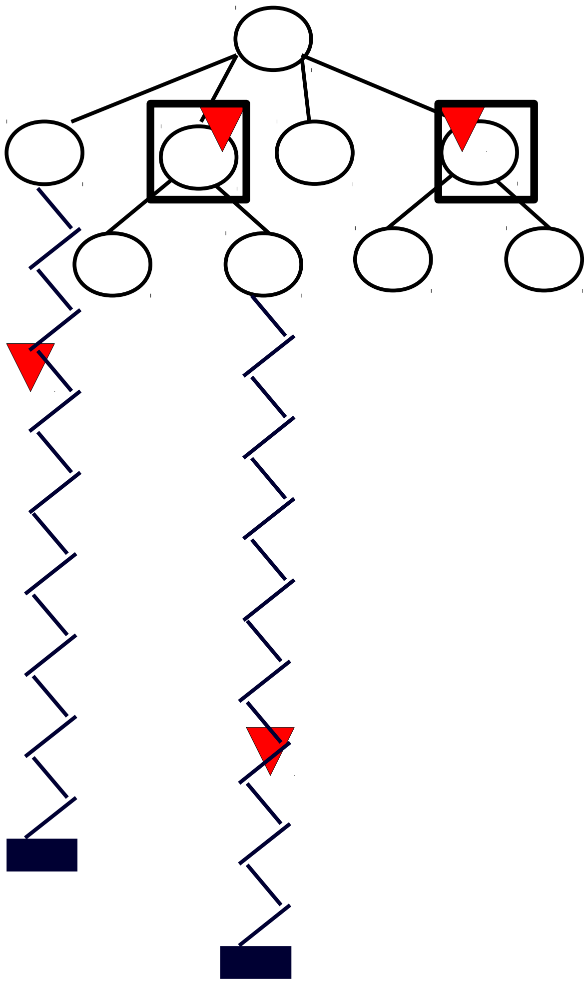
Root parallelization

- (Cazenave Jouandeau 2007)
- Easy to program
- No communication
- At completion, merge the trees
- 4 MCTS for 1sec > 1 MCTS for 4 sec
- Good way for low time settings and a small number of threads

Tree parallelization – global mutex



Tree parallelization – local mutex



Tree parallelization

- One shared tree, several threads
- Mutex
  - Global: the whole tree has a mutex
  - Local: each node has a mutex
- « Virtual loss »
  - Given to a node browsed by a thread
  - Removed at update stage
  - Preventing threads from similar simulations

Computer-computer results

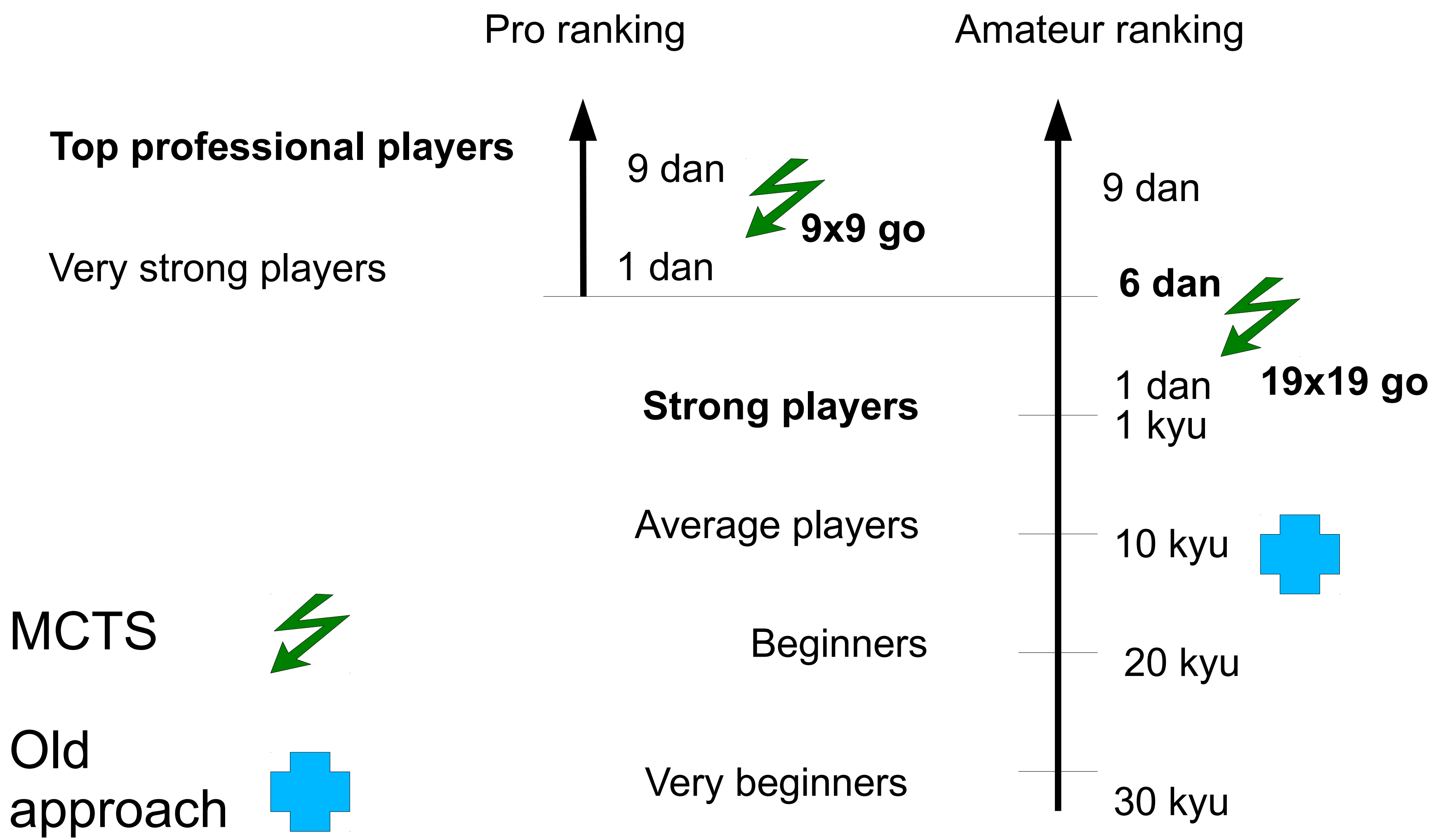
- Computer Olympiads

	19x19	9x9
2010	Erica, Zen, MFGo	MyGoFriend
2009	Zen, Fuego, Mogo	Fuego
2008	MFGo, Mogo, Leela	MFGo
2007	Mogo, CrazyStone, GNU Go	Steenvreter
2006	GNU Go, Go Intellect, Indigo	CrazyStone
2005	Handtalk, Go Intellect, Aya	Go Intellect
2004	Go Intellect, MFGo, Indigo	Go Intellect

Human-computer results

- 9x9
  - 2009: Mogo won a pro with black
  - 2009: Fuego won a pro with white
- 19x19:
  - 2008: Mogo won a pro with 9 stones
    Crazy Stone won a pro with 8 stones
    Crazy Stone won a pro with 7 stones
  - 2009: Mogo won a pro with 6 stones

MCTS and the old approach



Computer Go (MC history)

- Monte-Carlo Go (Brugmann 1993)
- MCGo devel. (Bouzy & Helmstetter 2003)
- MC+knowledge (Bouzy 2003)
- UCT (Kocsis & Szepesvari 2006)
- Crazy Stone (Coulom 2006)
- Mogo (Wang & Gelly 2006)

Conclusion

- Monte-Carlo brought a Big improvement in Computer Go over the last decade!
  - No old approach based program anymore!
  - All go programs are MCTS based!
  - Professional level on 9x9!
  - Dan level on 19x19!
- Unbelievable 10 years ago!

Some references

- PhD, MCTS and Go (Chaslot 2010)
- PhD, Reinf. Learning and Go (Silver 2010)
- PhD, R. Learning: applic. to Go (Gelly 2007)
- UCT (Kocsis & Szepesvari 2006)
- 1st MCTS go program (Coulom 2006)

Web links

- http://www.grappa.univ-lille3.fr/icga/
- http://cgos.boardspace.net/
- http://www.gokgs.com/
- http://www.lri.fr/~gelly/MoGo.htm
- http://remi.coulom.free.fr/CrazyStone/
- http://fuego.sourceforge.net/
- …

Thank you for your attention!