The Monte-Carlo Revolution in Go

Rémi Coulom

Université Charles de Gaulle, INRIA, CNRS, Lille, France

January, 2009

JFFoS’2008: Japanese-French Frontiers of Science Symposium
### Game Complexity

<table>
<thead>
<tr>
<th>Game</th>
<th>Complexity*</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tic-tac-toe</td>
<td>$10^3$</td>
<td>Solved manually</td>
</tr>
<tr>
<td>Connect 4</td>
<td>$10^{14}$</td>
<td>Solved in 1988</td>
</tr>
<tr>
<td>Checkers</td>
<td>$10^{20}$</td>
<td>Solved in 2007</td>
</tr>
<tr>
<td>Chess</td>
<td>$10^{50}$</td>
<td>Programs $&gt;$ best humans</td>
</tr>
<tr>
<td>Go</td>
<td>$10^{171}$</td>
<td>Programs $\ll$ best humans</td>
</tr>
</tbody>
</table>

*Complexity: number of board configurations*
How can we deal with complexity?

Some formal methods

- Use symmetries
- Use transpositions
- Combinatorial game theory
### How can we deal with complexity?

**Some formal methods**
- Use symmetries
- Use transpositions
- Combinatorial game theory

**When formal methods fail**
- Approximate evaluation
- Reasoning with uncertainty
Dealing with Huge Trees

Classical approach = depth limit + pos. evaluation (E)

Monte-Carlo approach = random playouts

Full tree
Dealing with Huge Trees

Classical approach = depth limit + pos. evaluation (E)
(chess, shogi, . . . )
Dealing with Huge Trees

Classical approach = depth limit + pos. evaluation (E) (chess, shogi, ...)

Monte-Carlo approach = random playouts
A Random Playout
Principle of Monte-Carlo Evaluation

Root Position

Random Playouts

MC Evaluation

\[ \text{MC Evaluation} = \text{Random Playouts} \]
Basic Monte-Carlo Move Selection

Algorithm
- \( N \) playouts for every move
- Pick the best winning rate
- 5,000 playouts/s on 19x19
Basic Monte-Carlo Move Selection

Algorithm
- $N$ playouts for every move
- Pick the best winning rate
- 5,000 playouts/s on 19x19

Problems
- Evaluation may be wrong
- For instance, if all moves lose immediately, except one that wins immediately.

9/10  3/10  4/10
Monte-Carlo Tree Search

Principle

- More playouts to best moves
- Apply recursively
- Under some simple conditions: proven convergence to optimal move when \( \#\text{playouts} \rightarrow \infty \)
Incorporating Domain Knowledge with Patterns

Patterns
- Library of local shapes
- Automatically generated
- Used for playouts
- Cut branches in the tree

Examples (out of $\sim 30k$)
- Good
- Bad

○ to move
## History (1/2)

### Pioneers

- **1993**: Brügmann: first MC program, not taken seriously
- **2000**: The Paris School: Bouzy, Cazenave, Helmstetter
History (1/2)

Pioneers
- 1993: Brügmann: first MC program, not taken seriously
- 2000: The Paris School: Bouzy, Cazenave, Helmstetter

Victories against classical programs
- 2006: Crazy Stone (Coulom) wins $9 \times 9$ Computer Olympiad
- 2007: MoGo (Wang, Gelly, Munos, . . .) wins $19 \times 19$
Victories against professional players

- 2008-03: MoGo beats Catalin Taranu (5p) on 9×9
- 2008-08: MoGo beats Kim Myungwan (9p) at H9
- 2008-09: Crazy Stone beats Kaori Aoba (4p) at H8
- 2008-12: Crazy Stone beats Kaori Aoba (4p) at H7
Summary of Monte-Carlo Tree Search

- A major breakthrough for computer Go
- Works similar games (Hex, Amazons) and automated planning
Conclusion

Summary of Monte-Carlo Tree Search
- A major breakthrough for computer Go
- Works similar games (Hex, Amazons) and automated planning

Perspectives
- Path to top-level human Go?
- Adaptive playouts (far from the root)?
Conclusion

Summary of Monte-Carlo Tree Search
- A major breakthrough for computer Go
- Works similar games (Hex, Amazons) and automated planning

Perspectives
- Path to top-level human Go?
- Adaptive playouts (far from the root)?

More information: http://remi.coulom.free.fr/CrazyStone/
- Slides, papers, and game records
- Demo version of Crazy Stone (soon)