General Game Playing

Sam Schreiber

(schreib@cs.stanford.edu)

Computers playing games?

- Not hard to imagine.
  IBM's *Deep Blue* beat the World Chess Champion in 1997 *(that's 13 years ago!)* .
- Simple approach: go through every position in the game, work backwards to winning moves.
- Chess is too big to do that.
  Instead, get good strategies and hard-code them in.
- Is that all there is?

Games that IBM's *Deep Blue* Can Play

1. Chess

Games that a GGP Program Can Play

1. Chess
2. Chinese Checkers
3. Checkers
4. Connect Four
5. Connect Five

...

Games that a GGP Program Can Play

...
5. Quarto
6. Pentago
7. Othello
8. Blocker
9. Tic-Tac-Toe
10. Counterstrike (Simplified)
11. Lunar Lander (Simplified)
12. Breakthrough
13. Knight-through
14. Tic-Tac-Chess
15. TTTCC4
16. Block World
17. Lights Out
18. Cephalopod
19. Cylinder Checkers
21. Nine Men's Morris
22. Finding a Knight's Tour
23. Adversarial Knight's Tour
24. Numeric Tic-Tac-Toe
25. Flipping Pancakes
26. Solving an Eight Puzzle
27. Solving a Sudoku Puzzle
28. Smallest Unique Number
29. Qyshinsu
30. Zhadu
31. Nim

...

Games that a GGP Program Can Play

...

94. Bidding Tic-Tac-Toe
95. Nine-Board Tic-Tac-Toe
96. Solving a Maze
97. Rock-Paper-Scissors
98. The Iterated Prisoner's Dilemma
99. Chess and Othello in Parallel

(And many more!)

Games that a GGP Program Can Play

Any game that is:

- **Finite**
  - Unlike "chess on an infinite board"
  - Unlike Mario Kart
- **No Hidden Information**
  - Unlike Poker
- **Deterministic**
  - Unlike Snakes & Ladders

General Game Playing Basics

- Game rules are encoded in a logic language (GDL).
- Players are sent the rules of the game and have a few minutes to strategize. They have never seen the game before.
- Then, play begins. Players submit moves every minute or so, and receive updates about the state of the game.
- Play continues until the game ends, and each player receives a score between 0 and 100.
- Simple to set up. Can support many different games. Challenging to program an effective game player!

One Approach: Heuristics

- Come up with a way to quantify how "good" a game state is. Do this with heuristics: e.g. "more available moves for me is better" or "fewer moves for my opponent is better".
- Search as far into the game as time permits.

- Try to pick moves that will eventually take you to a state that your heuristics believe is good.

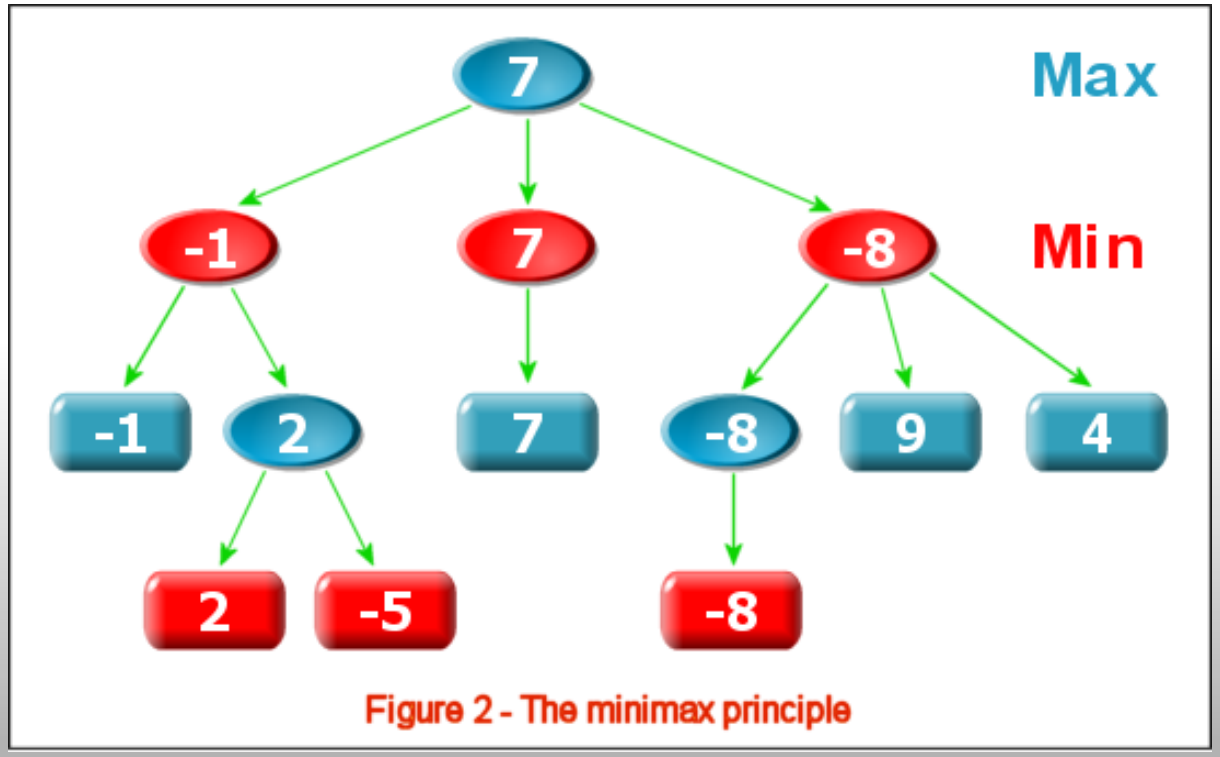
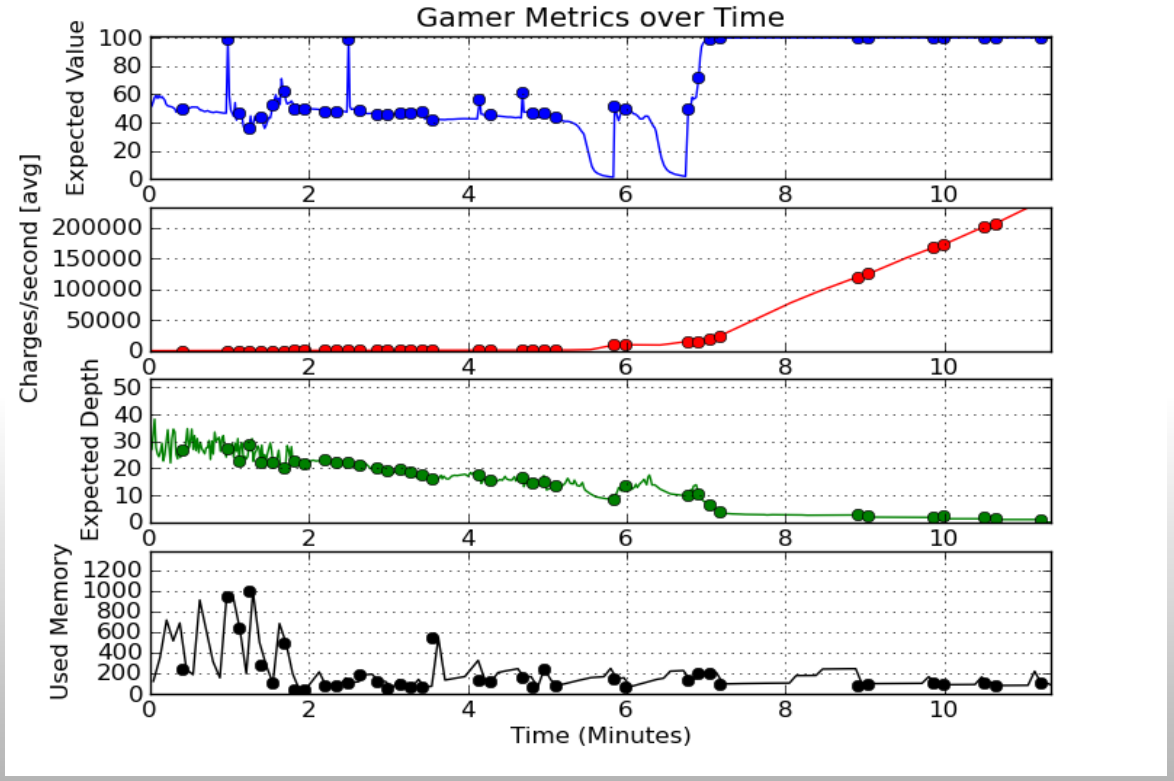


Figure 2 - The minimax principle

Another Way: Monte Carlo Simulation

- Simulate millions of games in which both players play randomly. For each of your possible moves, calculate the average score that you get in these games when you make that move. Pick the move with the highest average score!

- Fancier versions of this don't just play randomly.
- Surprisingly, this approach works really well on many games. (like Go!)



Out There in the Field Today

Annual General Game Playing Competitions at AAAI.

- Cluneplayer won in 2005 (a heuristic-based player)
- Fluxplayer won in 2006 (a heuristic-based player)
- Cadiaplayer won in 2007 (a simulation-based player)
- Cadiaplayer won in 2008 (a simulation-based player)
- Ary won in 2009 (a simulation-based player)

My player, *TurboTurtle* , is also simulation-based.

Unclear which approach will win out in the future!
Each has advantages, disadvantages.

Looks fun? Get involved!

There are many ways to get involved:

- Take CS 227b with Mike Genesereth!
- Check out *games.stanford.edu*
- Write a player and compete on the public round-robin TU Dresden GGP server.
- Talk with me for more details.

Questions?