Outline

- Stochastic Search (blind general game playing)
- Heuristics Generation (informed general game playing)
Monte Carlo Tree Search (1)
Monte Carlo Tree Search (2)

Value of move = Average score returned by simulation

\[ n = 60 \]
\[ v = 40 \]

\[ n = 22 \]
\[ v = 20 \]

\[ n = 18 \]
\[ v = 20 \]

\[ n = 20 \]
\[ v = 80 \]

\( n = \# \text{ of sample runs} \)
\( v = \text{average score} \)
Confidence Bounds

- Play one random game for each move
- For next simulation choose move

\[
\arg\max_i \left( v_i + C \sqrt{\frac{\log n}{n_i}} \right)
\]

Confidence bound

- \( n_1 = 4 \)  
  \( v_1 = 20 \)

- \( n_2 = 24 \)  
  \( v_2 = 65 \)

- \( n_3 = 32 \)  
  \( v_3 = 80 \)
Assessment

Monte Carlo Tree Search works particularly well for games that

- converge to the goal
- reward greedy behaviour
- have a large branching factor
- do not admit a good heuristics

Also, MCT Search is the most successful method for Computer Go to date!
Example: Game Without a Good Heuristics

2pttcc4 – a random combination of Chess, Tic-Tac-Toe, Connect4
Example: Game Where Simple MCT Search Fails

```
  knightbreakthrough
```
Informed Search: Exploiting Symmetries

Symmetries can be logically derived from the rules of a game.

A symmetry relation over the elements of a domain is an equivalence relation such that

- two symmetric states are either both terminal or non-terminal
- if they are terminal, they have the same goal value
- if they are non-terminal, the legal moves in each of them are symmetric and yield symmetric states

Example: Individual pebbles in Othello or Go
Reflectional Symmetry

Connect-3
Rotational Symmetry

Capture Go
Informed Search: Factoring

**hodgepodge** = Chess + Othello

- Branching factor: $a$
- Branching factor: $b$

Branching factor as given to players: $a \times b$
Fringe of tree at depth $n$ as given: $(a \times b)^n$
Fringe of tree at depth $n$ factored: $a^n + b^n$
Double Noughts And Crosses

Branching factor: 81, 64, 49, 36, 25, 16, 9, 4, 1
Branching factor (factored): 9, 8, 7, 6, 5, 4, 3, 2, 1 (times 2)
Game Factoring and its Use

1. Compute factors
   - Behavioural factoring
   - Goal factoring

2. Play factors

3. Reassemble solution
   - Append plans
   - Interleave plans
   - Parallelise plans with simultaneous actions
Factoring

A set $\mathcal{F}$ of features and moves is a *behavioural factor* if and only if there are no connections between the features and moves in $\mathcal{F}$ and those outside of $\mathcal{F}$.

Goal factoring (the simple case): goal is a conjunction
- Partition conjuncts over behavioural factors
- Create new goals for each factor

Goal factoring (the complex case): goal is a disjunction of conjunctions
- Split each conjunct as above
- Check for lossless joins, i.e. when recombined, we get the same results

Good:

$$(p_1 \land q_1) \lor (p_1 \land q_2)$$

Bad:

$$(p_1 \land q_1) \lor (p_2 \land q_2)$$
Blind Search

- Blind search: only assign scores to nodes based on the evaluation of the complete subtrees at those nodes

- Problem: can relatively rarely see all the way to the bottom of the tree for a single node, even less so for every successor node

- Solution: improve efficiency of inference

- Solution: assign intermediate scores to nodes based on an evaluation function

- Metagaming means to reason about properties of games
Informed Search: Using Evaluation Functions

- Typically designed by programmers/humans
- A great deal of thought and empirical testing goes into choosing one or more good functions
- E.g.
  - piece count, piece values in chess
  - holding corners in Othello
- But this requires knowledge of the game's structure, semantics, play order, etc.
The General Case

- No knowledge of features
- No insight into game structure
- No intuition about what is a good feature for this particular game

- Some general ideas work in many cases – but sometimes they don't ...
- E.g. mobility heuristics, novelty heuristics, goal distance
Mobility

- More moves means better state
  Optionally: limiting opponent moves is better too

- The good:
  In many games, being cornered or forced into making a move is quite bad
  - In Chess, when you are in check, you can do relatively few things compared to not being in check
  - In Othello, having few moves means you have little control of the board

- The bad: Mobility is counterproductive for Checkers
Worldcup 2006: Cluneplayer vs. Fluxplayer

Playclock:

Roles:
- Red
  - CLUNEPLAYER
- Black
  - FLUXPLAYER

Last Moves (step 2):
- Red
  - noop
- Black
  - move(bp,c,c6,d,c5)

Piece Count
- BLACK: 12
- RED: 12
Inverse Mobility

- Having fewer things to do is better
  Optionally: giving opponent things to do is better

- This works in some games, like Nothello, where you might in fact want to lose pieces

- How to decide between mobility and inverse mobility heuristics?
Novelty

- Changing the game state is better

- The good:
  - Changing things as much as possible can help avoid getting stuck
  - When it is unclear what to do, maybe the best thing is to throw in some directed randomness

- The bad:
  - Changing the game state can happen if you throw away your own pieces
  - Unclear if novelty per se actually goes anywhere useful for anybody
Identifying Structures: Relations

A **successor relation** is a binary relation that is antisymmetric, functional, and injective.

Example:

\[
\text{succ}(1, 2) \land \text{succ}(2, 3) \land \text{succ}(3, 4) \land \ldots
\]
\[
\text{next}(a, b) \land \text{next}(b, c) \land \text{next}(c, d) \land \ldots
\]

An **order relation** is a binary relation that is antisymmetric and transitive.

Example:

\[
\text{lessthan}(A, B) \leq \text{succ}(A, B)
\]
\[
\text{lessthan}(A, C) \leq \text{succ}(A, B) \land \text{lessthan}(B, C)
\]
 Boards and Pieces

An \((m\text{-dimensional})\) board is an \(n\)-ary state feature \((n \geq m+1)\) with

- \(m\) arguments whose domains are successor relations
- 1 output argument

Example:
\[
\text{cell}(a,1,\text{whiterook}) \land \text{cell}(b,1,\text{whiteknight}) \land \ldots
\]

A marker is an element of the domain of a board's output argument. A piece is a marker which is in at most one board cell at a time.

Example: Pebbles in Othello, White King in Chess
Simple Goal Distance

- The better an intermediate state satisfies the goal specification, the better it is.
- **Fuzzy Logic** to evaluate the "degree of truth" of a goal formula.
- Value $0.5 < p < 1.0$ for true literals (and $1-p$ for false literals).
Example: Noughts And Crosses

goal(xplayer,100) <= true(cell(M,1,x)) ∧
true(cell(M,2,x)) ∧
true(cell(M,3,x)) ∨
true(cell(1,N,x)) ∧
true(cell(2,N,x)) ∧
true(cell(3,N,x)) ∨
true(cell(1,1,x)) ∧
true(cell(2,2,x)) ∧
true(cell(3,3,x)) ∨
true(cell(1,3,x)) ∧
true(cell(2,2,x)) ∧
true(cell(3,1,x))
Evaluation of Intermediate States

\[
\text{fuzzy}_\text{eval}(\text{goal (xplayer, 100)}) \text{ after } \text{does (xplayer, mark (2, 2))} > \text{fuzzy}_\text{eval}(\text{goal (xplayer, 100)}) \text{ after } \text{does (xplayer, mark (1, 1))} > \text{fuzzy}_\text{eval}(\text{goal (xplayer, 100)}) \text{ after } \text{does (xplayer, mark (1, 2))}
\]
Advanced Goal Distance

The closer the current value of a functional state feature to the target value, the “less false” is the corresponding goal literal

- Remember how successor relations and order relations can be identified
- These relations define metrics $\Delta$ on the values of a functional feature
- Truth degree of $\text{true}(f(a))$ given that $\text{true}(f(b))$:

$$
(1-p) - (1-p) \times \frac{\Delta(b, a)}{|\text{dom}(f)|}
$$

where $(1-p)$ is the base value $>0$ assigned to false literals, as before
Example: The Goal in Racetrack

\[
\text{goal(white,100) \leq \text{true(lane(white,e))}} \]
\[
\text{init(lane(white,a))}
\]
Evaluation of Intermediate States

\[ \Delta(b,e) = 3 < \Delta(a,e) = 4, \text{ hence:} \]

\[
\text{fuzzy\_eval(goal(white,100)) after does(white,move(a,1,a,2)) < fuzzy\_eval(goal(white,100)) after does(white,move(a,1,b,1))}
\]
Another Example

\[ \text{init(cell(green,j,13))} \land \ldots \]
\[ \text{goal(green,100) } \leqslant \text{true(cell(green,e,5))} \land \ldots \]
Chinese Checkers (cont'd)

\[ \Delta((j,5), (e,5)) = 5 \ < \ \Delta((f,10), (e,5)) = 6 \]
Fuzzy goal evaluation works particularly well for games that

- have independent (conjunctive) sub-goals
  15-Puzzle
- converge to the goal
  Chinese Checkers
- have quantitative features
  Othello
- have partial goals
  Peg Jumping, Chinese Checkers with >2 players
Background Reading

Logic
- Russell & Norvig AIMA (3rd ed): Chapter 8 – Sections 8.1 and 8.2

Logic Programming
- Russell & Norvig AIMA (3rd ed): Chapter 9 – Sections 9.1, 9.2, and 9.4

Planning
- Russell & Norvig AIMA (3rd ed): Chapter 10 – Sections 10.1 and 10.2
- Chapter 11 – Section 11.3
  (2nd edition: 11.1, 11.2, 12.3)

General Game Playing
- games.stanford.edu/competition/misc/aaai.pdf
- www.ru.is/faculty/hif/papers/cadiaplayer_aaai08.pdf
- cgi.cse.unsw.edu.au/~mit/Papers/AAAI07a.pdf
General Game Playing – an AI Grand Challenge

Multiple AI methods come together
Logic and Reasoning, Planning and Search, Learning

For more information see

- general-game-playing.de
- games.stanford.edu

Or ask mit@cse.unsw.edu.au