Incomplete Search

Evaluation Functions

Complete search is usually impractical.

Alternative is to limit search depth in some way and apply a heuristic *evaluation function* to fringe states (whether terminal or non-terminal).

Chess examples:
- Piece count
- Board control
Example - Mobility

Mobility is a measure of the number of things a player can do.

Basis - number of actions in a state or number of states reachable from that state.

Horizon - current state or $n$ moves away.

Example - Focus

Focus is a measure of the narrowness of the search space. It is the inverse of mobility.

Sometimes it is good to focus to cut down on search space.

Often better to restrict opponents’ moves while keeping one’s own options open.
Some General Evaluation Functions

Conservative
value = 0 for all nonterminal states

Mobility and Focus
Maximize own
Minimize opponent’s

Novelty (especially with reversibility)
New states better than old states or vice versa
Similarity of states (compare state descriptors)

Goal proximity

Weighted Linear Evaluation Functions

Definition
\[ f(s) = w_1 \times f_1(s) + \ldots + w_n \times f_n(s) \]

Examples:
Piece count in chess
Board control in chess
Combination of piece count and board control
Mobility
Goal proximity
Novelty
Minimax

function maxscore (role, state)
{if (terminalp(state)) {return goal(role,state)};
 var value = [];
 for (var action in legals(role,state))
   {value[action] = minscore(role, action, state)};
 return max(value)}

function minscore (role, action, state)
{var value = [];
 for (move in findmoves(role,action,state))
   {value[move] = maxscore(role,next(move,state))};
 return min(value)}

Minimax’

function maxscore (role, state, level)
{if terminalp(state) return payoff(state);
 if level>levels then return evalfun(state);
 var value = [];
 for (action in findactions(role,state))
   {value[action] = minscore(role,action,state,level)};
 return max(value)}

function minscore (role, action, state, level)
{var value = [];
 for (move in findmoves(role,action,state))
   {ns = next(move,state);
    value[move] = maxscore(role,ns,level+1)};
 return min(value)}
Problems With Depth-Limited Search

Horizon Problem
white gains a rook but loses queen or loses game
example - sequence of captures in chess

Local Maxima

Variable Depth Search

Idea - use expansion function in place of fixed depth

Examples:
Quiescence search (attacks horizon problem)
Evaluation function values
Minimax’’

\[
\text{function maxscore (role, state, level)} \\
\quad \{\text{if (terminalp(state)) \{return goal(role,state)}\}; \\
\quad \text{if (\!expfun(state,level)) then \{return evalfun(state)}\}; \\
\quad \text{var value = new Array();} \\
\quad \text{for (action in legals(role,state))} \\
\quad \quad \{\text{value[action] = minscore(role,action,state,level)}\}; \\
\quad \text{return max(value)}\}
\]
Monte Carlo Method / Depth Charge

Basic Idea
(1) Explore game graph to some level storing generated states
(2) Beyond this, explore to end of game making random choices for moves of all players, not storing states (to limit space growth)
(3) Assign expected utilities to states by summing utilities and dividing by number of trials

Features
Fast because no search
Small space because nothing stored
Problems With Monte Carlo Methods

Optimistic
opponent *might* not respect probabilities
CadiaPlayer’s answer - use action utilities

unfocussed
CadiaPlayer’s answer - UCT (see paper)

No higher level reasoning
does not utilize game structure in any way

Evaluation Functions in GGP

General
Methods that are applicable to all games
Just discussed

Statistical
Estimate payoffs by random samples
Just discussed

Guaranteed
Find features that vary directly with final payoff
Sometimes doable in time proportional to description
More on this in weeks to come