Center for LifeLong Learning & Design
University of Colorado at Boulder

Wisdom is not the product of schooling
but the lifelong attempt to acquire it.
- Albert Einstein

Problem Solving — Game Playing: Dealing with Adversaries in Exploring Alternatives

Gerhard Fischer

AI Course, Fall 1996, Lecture September 23rd

Why Study Games in AI?

- problems are formalized

- real world knowledge (common sense knowledge) is not too important

- rules are fixed

- adversary modeling is of general importance (e.g., in economic situations, in military operations, ...)
  - opponent introduces uncertainty
  - programs must deal with the contingency problem

- complexity of games??
  - number of nodes in a search tree (e.g., 10^{40} legal positions in chess)
  - specification (missing information, ill-defined problems, semantically rich problems)

Game Playing - Overview

- games as search problems
- perfect decisions in two person games
- imperfect decisions
- alpha-beta pruning
- games with a chance
- state-of-the-art game programs

Examples

- Nim
- Tic-Tac-Toe
- Checkers
- Othello
- Chess (programs play at Grandmaster level)
- Go
- Backgammon (program has beaten the world champion, but ...)
- Blackjack

Formal Definition of Games as Search Problems

- initial state: board position + whose move
- a set of operators defining the legal moves of a player
- terminal test determining when the game is over
- utility function giving a numeric value for the outcome of a game (chess: +1, 0, and -1; backgammon: +192 to -192

Search Procedures

- **MINI-MAX** ---> static evaluation; conclusions about what to do at the deeper nodes of the search tree percolate up to determine what should happen at higher nodes

- **ALPHA-BETA**
  - there is no need to explore disastrous moves any further
  - can be augmented by a number of heuristic pruning procedures (danger: optimal moves may not be selected)

- **general trade-off:**
  - look-ahead operations
  - pattern-directed play

Minimax Algorithm

- generate the whole game tree, all the way down to the terminal states
- apply the utility function to each terminal state to get its value
- use the utility function of the terminal states to determine the utility of the nodes one level higher up in the search tree
- continue backing up the values from the leaf node toward the root, one layer at a time
- top of the tree: MAX chooses the move that lades to the highest value = minimax decision (maximizes the utility under the assumption that the opponent will play perfectly to minimize it)
- minimax search is depth-first

Heuristic Evaluation Functions

- allow us to approximate the true utility of a state without doing a complete search

- **changes:**
  - utility function is replaced by an heuristic evaluation function EVAL
  - terminal test is replaced by a cutoff test CUTOFF-TEST

- example for **Tic-Tac-Toe** (and Number Scrabble): static value associated with each field:
  - center: 4
  - corners: 3
  - middle field of a row: 2

- **chess:**
  - material value: pawn=1 — knight or bishop=3 — rook=5 — queen=9
  - other features: good pawn structure, king safety, mobility, ...

Alpha-Beta Pruning

- **basic idea:** it is possible to compute the correct minimax decision without looking at every node in the search tree ---> **pruning** (allows us to ignore portions of the search tree that make no difference to the final choice)

- **general principle:**
  - consider a node n somewhere in the tree, such that a player has a chance to move to this node
  - if player has a better chance m either at the parent node of n (or at any choice point further up) then n will never be reached in actual play

- **effectiveness:**
  - depends on the ordering in which the successors are examined
  - try to examine first the successors that are likely to be best

Game Playing: Case Study Othello — Questions to Think about

- how would you write a game playing program for Othello?
- what kind of evaluation function would you use or would you not use?
- what is the most difficult aspect of playing the game well?
- if you are an experienced Othello player, articulate some of your Othello knowledge

Rules

- each player takes 32 discs and chooses one color (64 discs are available to play)
- move: "outflanking" your opponent ---> then flipping the outflanked discs to your color
- definition of "outflank": ..... *establishing a domain vocabulary*
- black moves first ---> then take turns if legal moves are available
- if a move is available ---> one **must** take it
- outflanking occurs in all directions: horizontally, vertically, diagonally
- **all** discs outflanked in any one move must be flipped (even if it is to the player's disadvantage)
- end of game: when it is no longer possible for either player to move (either because all squares are filled or no legal move is available)

Incremental Development of Game Playing Programs

- let humans play against each other
  - the program serves as a representational media
  - the program checks for legal moves
- humans against program
  - legal moves by the program
  - good moves by the program
- humans against program — the program being in the role of a coach
- program against program
- learning component (program improve its play by playing games)

Humans against Program — Incremental Additions to the "Smartness" of the Program

- play randomly (but legal; may involve a non-trivial amount of knowledge / computation)

- have a static value associated with each square on the board

- have a dynamically value associated with each square on the board

- have an evaluation function taking other factors into account (e.g., number of pieces)

- search / look-ahead / exploring alternatives (using the evaluation function):
    - look one move ahead
    - look several moves ahead using minimax, alpha-beta,

Strategy

- goal is clear -- but how can we achieve the goal?
- corners are special: they can **never** be outflanked ---> question: how do we get one of our pieces into the corner (backward reasoning)
- squares next to corners are not good
- border squares are desirable (they can only be outflanked in only two directions)
- squares next to border squares are not desirable
- get **control** of the game: have many possible moves to choose from
- try to have as pieces of your color at any time in the game as possible

Rules themselves may be changed

- original set-up can vary:

  b w
  w b
  or
  b b
  w w

- turn
  - **one** direction
  - **all** directions

- let the player decide

- an extended version of the program could handle all strategies

- in chess: many variations

Other Issues

- Othello as a computer game -- claim: brute-force search based on a good evaluation function can yield excellent play

    - number of legal moves is small (in most situations)

    - humans have difficulties to "visualize" the long range consequences of a move

- knowledge elicitation / acquisition techniques: two humans play the game against each other and think-aloud

- thin spread of domain knowledge: claim: any amount of programming knowledge (e.g., in Lisp, C, ...) will not allow you to write a program which plays Othello well