Artificial Intelligence 15-381
Optimal Searching: A*-Family

Jaime Carbonell jgc@cs.cmu.edu
5-September-2000

Today's Agenda

- Macro-operators
- Admissible heuristics
- Full A^* Search
- B^* Search
- Means-Ends Analysis (if time permits)

Admissible Heuristics

Cost Function Definitions

Let $g(S_{curr})$ =
actual cost to reach S_{curr} from S_0

Let $h(S_{curr})$ =
estimated cost from S_{curr} to S_G

Let $f(S_{curr}) = g(S_{curr}) + h(S_{curr})$

Optimality Definition

A solution is **optimal** *if it conforms to the minimal-cost path between* $\mathbf{S_0}$ *and* $\mathbf{S_G}$*. If operators cost is uniform, then the optimal solution = shortest path.*

Admissibility Definition

A heuristic is **admissible** *with respect to a search method if it guarantees finding the* **optimal solution first***, even when its value is only an estimate.*

Admissible Heuristics for BestFS

- "Always expand the node with $\min(g(S_{curr}))$ first"
  If Solution found, expand any S_i in s-Queue
  where $g(S_i) < g(S_G)$
- "Find solution any which way. Then BestFS(S_i) for all intermediate S_i in solution as follows:

  1) if $g(S(^1{}_{curr}) >= g(S_G)$ in previous, quit,
  2) else if $g(S(^1{}_G < g(S_G)$, Sol := Sol^1,
     & redo (1).

A^*: Better Admissible Heuristics

Observations on admissible heuristics

- Admissible heuristics based only on look-back (e.g. on g(S)) can lead to massive inefficiency!
- Can we do better?
- Can we look forward (e.g. at $h(S_{curr})$) too?
- Yes, we can!

The A^* Criterion

If $h(S_{curr})$ always equals or underestimates the true remaining cost, then $f(S_{curr})$ is admissible with respect to Best-First Search.

A^* Search

A* Search = BestFS with admissible **f = g + h** under the admissibility constraints above.

A^* Optimality Proof

Goal and Path Proofs

- Let S_G be optimal goal state, and s-path(S_0,S_G) be the optimal solution.
- Consider an A^* search tree rooted at S_0 with S^1_G on fringe.
- Must prove $f(S_{G2}) >= f(S_G)$, ***and*** g(path(S_0,S_G)) is minimal (optimal).
- Text proves optimality by contradiction.

Simpler Optimality Proof for A^*

Assume s-Queue sorted by f.

Pick a suboptimal S_{G_2}: $g(S_{G_2}) > g(S_G)$

Since $h(S_{G_2}) = h(S_G) = 0$,

$f(S_{G_2}) > f(S_G)$

If s-Queue is sorted by f,

$f(S_G)$ is selected before $f_{S_{G_2}}$

B* Search

Ideas

- Admissible heuristics for mono- and bi-polar search
- "Eliminates" horizon problem in game-trees [more later]

Definitions

Let Best(S) = Always optimistic eval fn.

Let Worst(S) = Always pessimistic eval fn.

Hence: Worst(S) < True-eval(S) < Best(S)

Basic B* Search

B*(S) is defined as:

If there is an S_i in SUCC(S_{curr})
**   s.t. Forall other S_j in SUCC(S_{curr})**
**   W(S_i) > B(S_j)**
**   Then select S_i**
**   Else Prove-Best(SUCC(S_{curr})**
**   or Disprove-Rest(SUCC(S_{curr})**

Difficulties in B*

- Guaranteeing eternal pessimism in W(S)
  (eternal optimism is somewhat easier)
- Switching among Prove-Best and Disprove-Rest
- Usually W(S) << True-eval(S) << B(S)
  (not possible to achieve W(S_i) > B(S_j))

Means-Ends Analysis

Generalized Operator Examples

```
OPERATOR: move-robot(r,x,y)
     PRE: AT(r,x) & EMPTY(y) & CONNECTED(x,y)
     ADD: AT(r,y)
     DEL: AT(r,x)

OPERATOR:pick-up(r,z)
     PRE: AT(r,x) & AT(z,y) & NEXT-TO(x,y) &
          NOT(holding(r,w))
     ADD: HOLDING(r,z)
     DEL: AT(z,y)
```

Basic MEA

- Match(Goal, ADD(op))
- If all PRE(op) are met, then apply OP => GOAL
- Else Subgoals = unmet PRE(op)
- MEA recursively on subgoals
- Check for goal-loop conditions
- Fail if un-meetable subgoal

Planning with MEA

- Full MEA (multiple ops, multiple bindings)
- Plan = maximally variableized solution
- Use learnable "control rules" as heuristics
    - "If op_i & op_j applicable,
      choose op with fewest unmet PRE’s"