

Iowa State University Department of Computer Science
Artificial Intelligence Research Laboratory



Goal-Based Agents Problem Solving through Problem Reduction

Vasant Honavar
Artificial Intelligence Research Laboratory
Department of Computer Science
Bioinformatics and Computational Biology Program
Center for Computational Intelligence, Learning, & Discovery
Iowa State University
honavar@cs.iastate.edu
www.cs.iastate.edu/~honavar/
www.cild.iastate.edu/
www.bcb.iastate.edu/
www.igert.iastate.edu

Vasant Honavar, 2006.

Iowa State University Department of Computer Science
Artificial Intelligence Research Laboratory

Problem Reduction Representation

- Divide and conquer
- Reduce solution recursively to problems to solutions to sub-problems
- Problem is solved when all sub-problems are solved

Vasant Honavar, 2006.

Iowa State University Department of Computer Science
Artificial Intelligence Research Laboratory

Example

- Problem
 - solving an integral
- Sub-problems
 - easier integrals to solve
- Operators
 - rules of integral calculus and algebra
- Primitive problems
 - problems whose solutions can be looked up or computed by executing a known procedure

Vasant Honavar, 2006.

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

Example

- Problem
 - solving an integral
- Sub-problems
 - easier integrals to solve
- Operators
 - rules of integral calculus and algebra
- Primitive problems
 - problems whose solutions can be looked up or computed by executing a known procedure

Vasant Honavar, 2006.

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

Problem reduction representation (PRR)

- A PRR problem is specified by a 3-tuple (G, O, P)
 - G is a problem to be solved
 - O is a set of operators for decomposing problems into sub-problems through AND or OR decompositions
 - P is a set of primitive problems
- Solution
 - An **AND decomposition is solved when each of the sub-problems is solved**
 - An **OR decomposition is solved when at least one of the sub-problems is solved**
 - A problem is unsolvable if it is neither a primitive problem nor can it be further decomposed
- PRR is a generalization of the state space representation (why?)

Vasant Honavar, 2006.

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

Problem reduction representation

- Solving a problem in PRR reduces to searching an AND-OR graph
- Nodes correspond to problems
- Connectors correspond to arcs
- Connectors correspond to AND or OR decompositions
- Connectors of arity k are called k -connectors

Vasant Honavar, 2006.

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

Problem **Example**

3 solutions

Vasant Honavar, 2006.

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

Solution to an SRR problem

- A sub-graph s_q of an AND-OR graph is said to be solution to a problem q if
 - s_q is rooted at q
 - Each non-leaf node y in s_q has **exactly one connector** out of it that belongs to s_q
 - Each leaf node in s_q is a primitive problem (i.e. a member of P)
- A problem q is said to be solvable if
 - a sub-graph s_q of an AND-OR graph is a solution to q
- Solving a problem G using a PRR (G, O, P) entails finding a sub-graph S_G of the corresponding AND-OR graph that is a solution of G

Vasant Honavar, 2006.

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

Question – How can we solve an SRR problem?

- Basic idea:
 - Generalize state-space search
- How?
 - partial paths \rightarrow sub graphs of the SRR AND-OR graph
 - Expanding a node must comply with the semantics of AND and OR connectors
 - Termination test must comply with the definition of a solution

Vasant Honavar, 2006.

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

Example – BFS

□ ← Primitive problem
 ○ ← Unsolvable problem

List

(G)
 (A) (B & C)
 (B & C)
 (D & F) (E & F) (D & K) (E & K)

Exercise: Solve the same problem using DFS

Vasant Honavar, 2006.

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

Optimal (minimum cost) solution of AND-OR graphs

Example:

Cost of an unsolvable primitive problem – infinity
 Cost of connectors and primitive problems are assumed to be strictly positive and bounded

$$Cost(S_A) = Cost(k_1) + Cost(S_B) + Cost(S_C)$$

$$Cost(S_C) = \text{Min}\{(Cost(k_2) + Cost(S_D)), (Cost(k_3) + Cost(S_E))\}$$

Vasant Honavar, 2006.

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

Optimal solution of an SRR problem

Example:

Cheapest solution:

$S_{Amin} = 5 + 1 + 3 = 9$

Vasant Honavar, 2006.

Iowa State University Department of Computer Science
Artificial Intelligence Research Laboratory

Branch and Bound Search for Optimal Solution

Example:

List

- (A)
- (B & C) (D)
- (E & C) (D) (F & C)
- (D) (E & G & H) (F & C)
- (E & G & H) (I & J) (F & C)

$Cost(A) = Cost(E \& G \& H) = 5 + 1 + 3 + 0 + 0 = 9$

Vasant Honavar, 2006.

Iowa State University Department of Computer Science
Artificial Intelligence Research Laboratory

Using Heuristics

$f(C \& D) = Cost(A) + 5 + h(C) + h(D)$

$f(E) = Cost(A) + 2 + h(E)$

Admissible heuristic function

$h(n) \leq h^*(n) = Cost(n)$ ← Cost of the cheapest solution of n

Vasant Honavar, 2006.

Iowa State University Department of Computer Science
Artificial Intelligence Research Laboratory

AO* - Searching AND-OR graphs

Example:

$h(C) = h(D) = h(I) = h(J) = 1$

List

- (A)
- (I & J) (C & D)
- (K & M & N) (C & D) (L & M & N)

Vasant Honavar, 2006.

Properties of AO*

- AO* is a generalization of A* for AND-OR graphs
- AO*, like A*, is admissible if the heuristic function is admissible and the usual assumptions (finite branching factor etc) hold
- AO*, like A* is also optimal among the class of heuristic search algorithms that use an additive cost / evaluation function

Proofs left as exercises
