Cognitive Modeling

Lecture 4: Models of Problem Solving

Sharon Goldwater

School of Informatics
University of Edinburgh
sgwater@inf.ed.ac.uk

January 21, 2010

Reading: Cooper (2002: Ch. 4).

Problem-solving

Many daily and long-term tasks involve problem-solving.

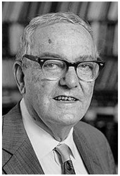
- buying airline tickets given particular time and money constraints;
- finding and following directions to a new location;
- figuring out why your computer isn't behaving as you expect;
- devising a winning strategy in a board game.

How do we solve these tasks?

History

Historical approaches to studying problem solving:

- Early work focused on *reproductive* problem-solving, associationist explanations (stimulus-response).
- 1940s Gestalt psychologists studied *productive* problems, believed problem-solving reduced to identifying appropriate problem structure.
- Today we'll look at work by Herb Simon on *well-defined, knowledge-lean* problems.



(photo: Wikipedia)

Types of problems

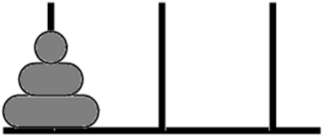
Problems can be categorized on two dimensions:

	knowledge-lean	knowledge-rich
well-defined	chess Towers of Hanoi	fixing computer problem diagnosing a patient
ill-defined	??	win an election design a better car

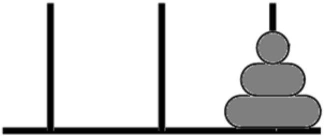
Example: Towers of Hanoi

- Starting point: all disks stacked on leftmost peg in order of size (largest on bottom); two other pegs empty.
- Legal moves: any move which transfers a single disk from one page to another without placing it on top of a smaller disk.
- Goal: transfer all disks to the rightmost peg.

Using three disks:



initial state



goal state

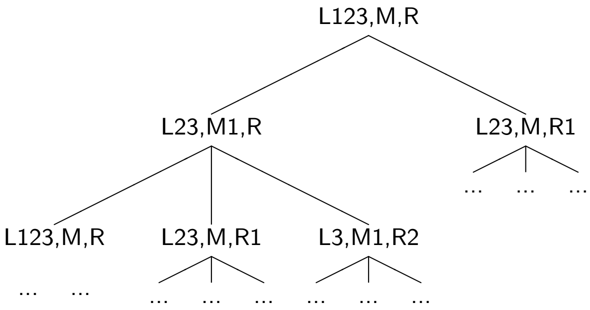
Well-defined, knowledge-lean problems

Can be characterized by a *state space*.

- Chess: configuration of pieces on the board
- Towers of Hanoi: configuration of disks and pegs, e.g.
  - L123,M,R: All three disks located on leftmost peg (initial state)
  - L3,M,R12: Largest disk on left peg, smaller two on right peg

Possible solution methods

Computers often solve similar problems by exploring the search space in a *systematic* way.



- depth-first search, breadth-first search

Might humans do this as well?

Cognitive plausibility

DFS and BFS strategies don't match human behavior.

- humans show greater difficulty at some points during solving than others – not necessarily those with more choices.
- for complex tasks (e.g., chess), both methods can require very large memory capacity.
- human *learn* better strategies with experience: novices may not find the best solution, but experts may outperform computers.

Case study: Towers of Hanoi

The problem can be decomposed into a series of sub-problems:

1. move the largest disk to the right peg;
2. move intermediate-sized disk to the right peg;
3. move the smallest disk to the right peg.

Solve sub-problems in order:

- move largest disk to the right peg: achieve a state where this can be solved in one move (i.e., no other disks on it, no disks on right peg).

Now move two-disk tower from left to middle peg: *easier version of initial problem; the same principles used to solve it.*

Simon (1975)

Simon (1975) analyzed possible solution strategies and identified four classes of strategy:

- problem decomposition strategy (see above);
- two simpler strategies that move disks (rather than towers), moves triggered by perceptual features of the changing state;
- a strategy of rote learning.

Strategies have different properties in terms of generalization to larger numbers of disks and processing requirements.

Anzai and Simon (1979)

Analyzed *verbal protocols* of one subject in four attempts at five-disk task:

1. Initially *little sign of strategy,* moving disks using simple constraints – avoid backtracking, avoid moving same disk twice in a row.
2. By third attempt had a sophisticated *recursive strategy,* with sub-goals of moving disks of various sizes to various pegs.
3. Final attempt, strategy evolved further, with sub-goals involving moving *pyramids* of disks.

Developed adaptive production system to simulate acquisition and evolution of strategies.

Selection Without Search

At each stage in the solution process:

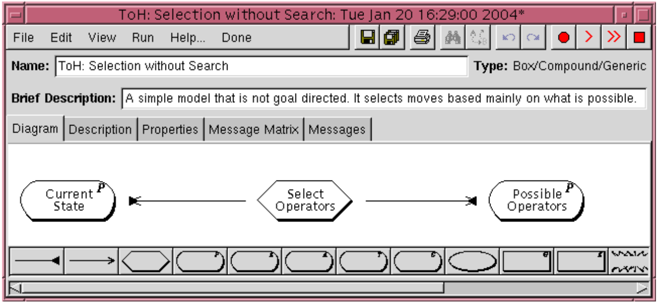
- enumerate the possible moves;
- evaluate those moves with respect to *local information*;
- select the move with the highest evaluation;
- apply the selected move;
- if the goal state has not been achieved, repeat the process.

This approach can be applied to any well-specified problem (Newell and Simon 1972).

Selection Without Search

We require:

- one buffer to hold the current state;
- one buffer to hold the representation of operators;
- and one process to manipulate buffer contents.



Representing the Current State

Each disk may be represented as a term:

```
disk(Size,Peg,Position)
```

With this representation, the initial state of the five disk problem might be represented as:

```
disk(30,left,5)
disk(40,left,4)
disk(50,left,3)
disk(60,left,2)
disk(70,left,1)
```

Operator Proposal

In the operator proposal phase, we propose moving the top-most disk on any peg to any other peg:

IF not `operator(AnyMove,AnyState)` is in **Possible Operators**
  `top_disk_on_peg(Size,Peg1)`
  `other_peg(Peg1,Peg2)`
THEN add `operator(move(Size,Peg1,Peg2),possible)` to
    Possible Operators

Some possible operators may violate task constraints.

Operator Evaluation

In operator evaluation phase, assign numerical evaluations to all possible operators:

IF `operator(Move,possible)` is in **Possible Operators**
`evaluate_operator(Move,Value)`
THEN delete `operator(Move,possible)` from **Possible Operators**
add `operator(Move,value(Value))` to **Possible Operators**

Possible operators that violate task constraints receive low evaluations.

Other operators receive high evaluations.

Evaluation Function

Further aspects of the subject's first attempt:

- she avoids backtracking and moving the same disk twice.
- she never moves the small disk back to the peg it was on two moves previously.

How would we incorporate these into the evaluation function?

Operator Selection

The selection rule should fire at most once on any cycle:

IF not `operator(AnyMove,selected)` is in **Possible Operators**
  `operator(Move,value(X))` is in **Possible Operators**
  not `operator(OtherMove,value(Y))` is in **Possible Operators**
    `Y` is greater than `X`
THEN add `operator(Move,selected)` to **Possible Operators**

Once an operator has been selected, others can be deleted:

IF exists `operator(Move,selected)` is in **Possible Operators**
  `operator(AnyMove,value(V))` is in **Possible Operators**
THEN delete `operator(AnyMove,value(V))` from
    Possible Operators

Operator Application

Applying an operator involves changing the current state:

IF `operator(move(Size,FromPeg,ToPeg),selected)`
    is in **Possible Operators**
  `disk(Size,FromPeg,FromPosition)` is in **Current State**
  `get_target_position(ToPeg,ToPosition)`
THEN delete `disk(Size,FromPeg,FromPosition)` from
    Current State
  add `disk(Size,ToPeg,ToPosition)` to **Current State**
  clear **Possible Operators**

Properties

Properties of selection without search:

- selection of the first move is random;
- if the model selects the wrong first move, it can go off into an unproductive region of the problem space;
- the model will find a solution eventually, but it can be very inefficient.

Nevertheless subjects seem to use this strategy first (Anzai and Simon 1979).

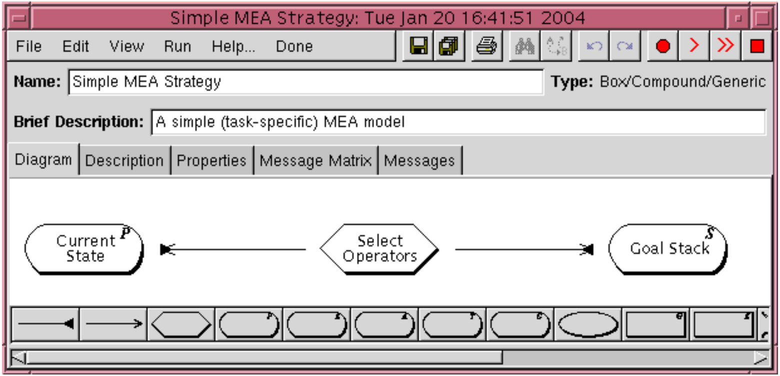
Goal Directed Selection

Strategy: set intermediate goals and move disks to achieve them.

- subgoal: move the largest blocking disk to the middle peg.
- maintain a *stack* for further subgoals, in case initial subgoal is not directly achievable.
- when completed, top subgoal is popped from the stack.

A Revised Diagram

Possible Operators becomes a stack buffer, now called **Goal Stack**.



Setting Primary Goals

IF the goal stack is empty
there is a difference between the current and goal states
THEN find the biggest difference between the current and goal states
(the largest disk out of place)
set a goal to eliminate that difference
(move the largest disk to its goal location)

Setting Subgoals and Making Moves

Setting Subgoals:

IF	the current goal is not directly achievable
THEN	set a goal to achieve current goal's preconditions

Moving Disks and Popping Subgoals:

IF	the current goal is directly achievable
THEN	move the disk
	pop the goal off the goal stack

Properties

Properties of goal-directed selection:

- selection of moves is no longer random;
- selection is guided by the goal of moving the largest disk that is in an incorrect position;
- if the goal is not directly achievable, it is recursively broken down into subgoals;
- efficient strategy that avoids unproductive regions of the search space.

Goal-directed selection seems to be used by experienced players (evidence for learning; Anzai and Simon 1979).

Generalized Means-Ends Analysis

The problem solving strategy in the previous model is known as *means-ends analysis.*

In general, MEA involves locating the largest difference between current and goal state, and selecting an operator to eliminate this difference. To apply to a specific problem, must

- identify appropriate distance measure for differences;
- identify operators that can eliminate differences.

Most people seem to have access to this general strategy.

Switching to MEA

Why didn't the subject use MEA from the outset?

- She may have assumed a simpler solution strategy (selection without search) was sufficient.
- She may have lacked the knowledge of the problem space needed to perform MEA (operators and differences that they can be used to eliminate).

Hypothesis: during her first attempt the subject acquired an understanding of how to decompose the problem into subgoals.

Evidence: the explicit mention of subgoals became more common as she gained experience with the task.

Learning to Solve

At least three types of learning were seen in the subject:

- switching from Selection without Search to Goal-directed strategy;
- changing the goal type from moving disks to moving pyramids;
- chunking subtasks (treating movement of top 3 disks as a single move).

How could these be modeled?

Summary

- Tower of Hanoi case study shows how people's problem-solving strategies change over time. Strategies include:
  - *selection without search:* enumerate all solutions, select the best one;
  - *goal-directed selection:* decompose the problem into subgoals and solve those;
  - *generalized means-ends analysis:* find the largest difference between the current state and the goal state and select an operator to eliminate it.
- Anzai and Simon's (1979) model captures individual stages, but less satisfactory explanation of transitions between stages.

References

Anzai, Y. and H. A. Simon. 1979. The theory of learning by doing. *Psychological Review* 86:124–140.

Cooper, Richard P. 2002. *Modelling High-Level Cognitive Processes*. Lawrence Erlbaum Associates, Mahwah, NJ.

Newell, Alan and Herbert A. Simon. 1972. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, NJ.

Simon, H. A. 1975. The functional equivalence of problem solving skills. *Cognitive Psychology* 7:268–288.