Problem Solving and Planning

CompSci 765
Meeting 5

Pat Langley
Department of Computer Science
University of Auckland
Outline of the Lecture

• Novel activity and problem solving
• Problem solving and search
• States, operators, and problem spaces
• Defining a problem space
• Planning as problem solving
• Knowledge-rich planning
• Implications for social cognition
• Summary remarks
Routine and Novel Cognition

Most work on production systems has focused on complex but routine tasks, such as:

- Multi-column subtraction
- Reading syntactic sentences
- Teaching algebra procedures
- Flying tactical air missions

Each uses knowledge to support agents that engage in some goal-directed behavior in familiar settings.

But humans can also operate in unfamiliar contexts, where they exhibit novel behaviors.
Novel Activity and Problem Solving

When we encounter unfamiliar situations, we cannot rely on routine behaviors to achieve our goals.

- Of course, this is a continuum, with some settings being less familiar than others and requiring more novel responses.
- But we need mechanisms beyond those we have discussed to support goal-directed behavior in new situations.

We will refer to behavior in such settings, whether physical or mental, as *problem solving*.

Early AI research emphasized problem solving, and this ability is clearly a distinctive element of human intelligence.
An Example: The Tower of Hanoi

Consider a puzzle that is known as the Tower of Hanoi, which involves three pegs and N disks.

The standard task is to move all disks from an initial peg where they are located to another peg.
An Example: The Tower of Hanoi

The standard task is to move all disks from an initial peg where they are located to another peg; however:

• You can move only one disk at a time;
• You cannot move a disk if a smaller one is on top of it; and
• You cannot move a disk to a peg that has a smaller disk on it.

There exist routine procedures for the Tower of Hanoi, but they are not available at first encounter.

Yet most people can solve this puzzle with moderate effort.

Studies of this task have produced many insights about problem solving in humans and machines.
Varieties of Problem-Solving Tasks

Problem solving arises in many different forms; instances of this cognitive behavior occur when:

- Solving a novel puzzle
- Proving a unfamiliar theorem
- Generating an innovative plan
- Creating a complex schedule
- Writing an essay or paper
- Designing a new artifact

Each of these problem classes is complex enough that, without knowledge to make them routine, they require innovation.
Problem Solving and Dynamic Composition

One thing needed for novel behavior is the ability to compose cognitive structures dynamically.

• We have seen this idea in production systems, but combinations there were highly constrained.

• Thus, dynamic composition is a key element even in producing routine behavior.

• To solve unfamiliar tasks, we must combine similar elements but with less knowledge to guide the process.

However, this raises a deeper issue. Without knowledge about which decisions to make, what choices do we make?
Problem Solving and Search

Novel tasks are hard because there are many different ways to combine a sequence of steps.

• Each step along the way requires making a decision that, without knowledge, may not help solve the problem.
• Finding a sequence of actions, physical or mental, that produces a solution may require trying different paths.
• We can restate this as the need to carry out search on unfamiliar problems.

Search is a major idea that has played a central role in both AI and cognitive psychology.
The Search Metaphor

The notion of search in problem solving is based on an analogy with finding the way through a physical maze.

Like logical reasoning and remembering, search is a useful metaphor for understanding intelligent behavior.

Such metaphors are not right or wrong, but they suggest ways to approach mental phenomena.

The Tower of Hanoi shows that problem solving, and search, can occur in the physical world or in the mind.
The Notion of a Problem Space

Viewing problem solving as maze traversal leads naturally to the notion of a *problem space* that involves:

- **States** in the problem space (places in the maze), including:
  - The *initial* state (the maze entrance); and
  - The *goal* state (the maze exit or center);
- **Operators** that change states (steps in the maze); and
- **Solution path(s)** for the problem (way through the maze).

These elements of a problem space play central roles in most AI analyses of problem solving.
The Task of Problem Solving

Once we have adopted the search metaphor, we can define the generic task of problem solving as:

- *Given:* A current situation (physical or mental);
- *Given:* A specification for some desired situation;
- *Given:* A set of operators for changing situations, along with constraints on their application;
- *Find:* A situation that satisfies the desired specification and a sequence of operators that produces it.

This is *not* the only way to formulate problem solving, but it dominates AI and has led to many insights.

The statement's generality makes it relevant to many tasks.
Representing Problem States

If we adopt the problem-space paradigm, then problem solving requires that we represent states in the space.

In many cases, each stated is encoded as a set of a modular, logic-like facts or elements.

E.g., we might represent the initial Tower or Hanoi state as:

\[
\begin{align*}
\text{is\_disk}(\text{disk1}) & \quad \text{is\_peg}(\text{pega}) & \quad \text{on}(\text{disk1}, \text{pega}) \\
\text{is\_disk}(\text{disk2}) & \quad \text{is\_peg}(\text{pegb}) & \quad \text{on}(\text{disk2}, \text{pega}) \\
\text{is\_disk}(\text{disk3}) & \quad \text{is\_peg}(\text{pegc}) & \quad \text{on}(\text{disk3}, \text{pega}) \\
\text{smaller}(\text{disk1}, \text{disk2}) & \quad \text{smaller}(\text{disk1}, \text{disk3}) & \quad \text{smaller}(\text{disk2}, \text{disk3})
\end{align*}
\]

Other formalisms are certainly possible, but ones of this sort make close contact with other AI methods.
Representing Operators

The problem-space approach also requires that we represent operators that let us move through the space.

These often take a form very similar to production rules, with:

- conditions for legal application of the operator;
- elements / relations to add / remove upon application.

E.g., we might represent the Tower or Hanoi operator as:

\[
\text{move}(D, P, Q)
\]

\[
\text{if} \quad \text{is\_disk}(D) \text{ and } \text{is\_peg}(P) \text{ and } \text{on}(D, P) \text{ and } \text{is\_peg}(Q) \text{ and } \\
\text{not\_eq}(P, Q) \text{ and } \text{not}(\text{on}(\text{Any}, P) \text{ and } \text{smaller}(\text{Any}, D)) \text{ and } \\
\text{not}(\text{on}(\text{Other}, Q) \text{ and } \text{smaller}(\text{Other}, D)), \\
\text{then remove}(\text{on}(D, P)) \text{ and } \text{add}(\text{on}(D, Q)).
\]

Operation notation must map cleanly onto state representation.
Defining a Problem Space Implicitly

Upon entering a maze, we do not know its size or connectivity, and the same holds for other problem spaces.

• For physical problems, we can explore the environment in an effort to find a solution path.
• Given an initial state and operators, we can do the same with a mental problem space.
• One simply applies legal operators to the initial state, applies operators to the new states, and so on.

This lets us specify a large (and even an infinite) problem space *implicitly* with a small problem description.

Most AI / psychology work on problem solving uses this idea.
Tower of Hanoi – Problem Space

Starting from a single state, we can apply operators repeatedly to generate the entire problem space.
The Task of Planning

An important type of problem solving involves the generation of plans or courses of action:

• *Given:* A current situation (typically physical);
• *Given:* A set of goals to be achieved;
• *Given:* A set of actions for changing situations, along with their conditional effects;
• *Find:* A situation that satisfies the goals and a sequence of actions that produces it.

Planning differs from *design* and *constraint satisfaction* tasks in its focus on sequences of *action* that transform the world.
Varieties of Planning Techniques

AI researchers have developed a variety of planning methods that differ along dimensions like:

• Search direction (forward from states, backward from goals)
• Search regimen (e.g., depth first, best first, iterative sampling)
• Methods used to select states, goals, and operators
• Time of commitment (eager vs. delayed)

Most current planning systems carry out a form of forward search, although backward chaining was once common.

Planners usually adopt a knowledge-lean approach in which domain content resides only in operators.
Knowledge-Rich Planning

However, knowledge-rich planning techniques are also possible. Most research in this paradigm uses *hierarchical task networks* (HTNs) to encode knowledge about activities.

A hierarchical task net comprises a set of methods, each with:

- a task predicate and its arguments;
- a set of conditions under which that method applies;
- a set of subtasks that let one implement the method; and
- an (optional) set of effects that the method produces.

Such knowledge is similar in structure to a Prolog program, but it describes activities that occur over time.
An HTN for Obtaining Money

Consider a simple HTN with two methods for obtaining money:

\[
\text{obtain\_money}(X) \\
\quad \text{conditions: have\_job}(X, Y) \\
\quad \text{subtasks: perform\_job}(X, Y), \text{collect\_paycheck}(X, Y) \\
\quad \text{effects: have\_money}(X) \\
\]

\[
\text{obtain\_money}(X) \\
\quad \text{conditions: beneficiary\_of}(X, Y, Z) \\
\quad \text{subtasks: kill}(X, Y), \text{collect\_insurance}(X, Z) \\
\quad \text{effects: dead}(Y), \text{have\_money}(X) \\
\]

\[
\text{kill}(X, Y) \\
\quad \text{conditions: have\_gun}(X, W) \\
\quad \text{subtasks: shoot\_with}(X, Y, W) \\
\quad \text{effects: dead}(Y), \text{gsr\_on}(X) \\
\]

We can use this knowledge to generate plans for carrying out tasks like \text{obtain\_money}(abe).
Hierarchical Plan to Obtain Money

obtain_money(abe)

beneficiary_of(abe, bob, p1)

have_gun(abe, g1)

kill(abe, bob)

shoot_with(abe, bob, g1)

gsr_on(abe)

dead(bob)

collect_insurance(abe, p1)

have_money(abe)

C = Condition          S = Subtask          E = Effect
Planning and Execution

The purpose of planning is to achieve goals, which cannot occur without executing the planned actions.

Researchers have explored a number of strategies for combining planning with execution:

• Executing only after planning has found a full solution path;
  – Open-loop execution that assumes things will go as planned;
  – Closed-loop execution with monitoring and replanning as needed;

• Executing actions as soon as one has solved a subproblem;

• Interleaving lookahead search with single actions (game playing);

• Reactive execution that accesses stored generalized plans / HTNs.

Some form of execution over time is required by all interactive cognitive systems.
Implications for Social Cognition

Planning and plan execution play central roles in interactive cognitive systems:

- Both synthetic characters and robots must carry out extended activities to reach their goals;
- Mixed-initiative planning aids must contribute to plans that human teammates propose; and
- Dialogue systems must participate in conversations to achieve joint task-oriented objectives.

Interactive cognitive systems exist over time, which means they must generate, or at least execute, sequences of actions.
Summary Remarks

Problem solving and planning are crucial abilities for cognitive systems that involve:

• Representing problem spaces in terms of states and operators;
• Carrying out heuristic search through these problem spaces to achieve goals or desired states; and
• Using domain-independent methods when needed but drawing on knowledge (e.g., HTNs) when available.

An interactive cognitive system must combine planning with execution to actually achieve its objectives.

Applications include synthetic characters, mixed-initiative planning, human-robot interaction, and dialogue systems.
Reading Assignments

We will have three reading assignments for the next meeting:


These papers describe some applications of knowledge-based planning and execution that involve interactive systems.
End of Lecture