Problem Solving

- AI programs have a clean separation of
  - ✓ computational components of data,
  - ✓ operations and
  - ✓ control
- Search forms the core of many intelligent processes.
- It is useful to structure AI programs in a way that facilitates describing the search process.

Production System:

- It is a formalism for structuring AI programs which facilitates search process. It consists of
  - ✓ Initial or start state of the problem
  - ✓ Final or goal state of the problem

- It consists of one or more databases
    - ✓ consisting of appropriate information for the particular task.
    - ✓ information in these databases may be structured using knowledge representation schemes.
- Set of production rules,
    - ✓ each consisting of a left side that determines the applicability of the rule and
    - ✓ a right side that describes the action to be performed if the rule is applied.
    - ✓ These rules operate on the databases.
    - ✓ Application of rules change the database.
- A control strategy is one that specifies the order in which the rules will be applied when several rules match at once.

- One of the examples of Production Systems is an Expert System.
- In addition to its usefulness as a way to describe search, the production model has other advantages as a formalism in AI.
    - ✓ It is a good way to model the strong state driven nature of intelligent action. As new inputs enter the database, the behavior of the system changes.
    - ✓ New rules can easily be added to account for new situations without disturbing the rest of the system, which is quite important in real-time environment.

Water Jug Problem

Problem statement:
Given two jugs, a 4-gallon and 3-gallon having no measuring markers on them. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into 4-gallon jug.

Solution: State space for this problem can be described as the set of ordered pairs of integers (X, Y) such that X represents the number of gallons of water in 4-gallon jug and Y for 3-gallon jug.

1. Start state is (0,0)
2. Goal state is (2, N) for any value of N.

Following are the production rules for this problem.

Production Rules:

R1:	$(X, Y \mid X < 4)$	$\rightarrow$	$(4, Y)$ {Fill 4-gallon jug}
R2:	$(X, Y \mid Y < 3)$	$\rightarrow$	$(X, 3)$ {Fill 3-gallon jug}
R3:	$(X, Y \mid X > 0)$	$\rightarrow$	$(0, Y)$ {Empty 4-gallon jug}
R4:	$(X, Y \mid Y > 0)$	$\rightarrow$	$(X, 0)$ {Empty 3-gallon jug}
R5:	$(X, Y \mid X+Y >= 4 \Lambda Y > 0)$	$\rightarrow$	$(4, Y – (4 – X))$ {Pour water from 3- gallon jug into 4-gallon jug until 4-gallon jug is full}

R6: $(X, Y \mid X+Y >= 3 \wedge X > 0) \rightarrow (X - (3 - Y), 3)$

{Pour water from 4-gallon jug into 3-gallon jug until 3-gallon jug is full}

R7: $(X, Y \mid X+Y <= 4 \wedge Y > 0) \rightarrow (X+Y, 0)$

{Pour all water from 3-gallon jug into 4-gallon jug }

R8: $(X, Y \mid X+Y <= 3 \wedge X > 0) \rightarrow (0, X+Y)$

{Pour all water from 4-gallon jug into 3-gallon jug }

Superficial Rules: {May not be used in this problem}

R9: $(X, Y \mid X > 0) \rightarrow (X - D, Y)$

{Pour some water D out from 4-gallon jug}

R10: $(X, Y \mid Y > 0) \rightarrow (X, Y - D)$

{Pour some water D out from 3- gallon jug}

- It is to be noted that there may be more than one solutions.

Trace of steps involved in solving the water jug problem
First solution

Number of Steps	*Rules applied*	*4-g jug*	*3-g jug*
1	**Initial State**	0	0
2	R2 {Fill 3-g jug}	0	3
3	R7{Pour all water from 3 to 4-g jug }	3	0
4	R2 {Fill 3-g jug}	3	3
5	R5 {Pour from 3 to 4-g jug until it is full}	4	2
6	R3 {Empty 4-gallon jug}	0	2
7	R7 {Pour all water from 3 to 4-g jug}	2	0

Goal State

Second solution

Number of steps	*Rules applied*	*4-g jug*	*3-g jug*
1	**Initial State**	0	0
2	R1 {Fill 4-gallon jug}	4	0
3	R6 {Pour from 4 to 3-g jug until it is full }	1	3
4	R4 {Empty 3-gallon jug}	1	0
5	R8 {Pour all water from 4 to 3-gallon jug}	0	1
6	R1 {Fill 4-gallon jug}	4	1
7	R6 {Pour from 4 to 3-g jug until it is full}	2	3
8	R4 {Empty 3-gallon jug}	2	0

Goal State

Points to be noted

- There are initial and final description of the problem.
- There are more than one ways of solving the problem.
- One would exercise a choice between various solution paths based on some criteria of goodness or on some heuristic function.
- There are set of rules that describe the actions called production rules.
- Left side of the rules is current state and right side describes new state that results from applying the rule.

We can summarize that in order to provide a formal description of a problem, it is necessary to do the following things:

1. Define a state space that contains all the possible configurations of the relevant objects.
2. Specify one or more states within that space that describe possible situations from which the problem solving process may start. These states are called **initial states.**
3. Specify one or more states that would be acceptable as solutions to the problem called **goal states.**
4. Specify a set of rules that describe the actions. Order of application of the rules is called control strategy.
5. Control strategy should cause motion towards a solution.

Control Strategies

- How to decide which rule to apply next during the process of searching for a solution to a problem.
- Requirement for a good Control Strategy

✓ **It should cause motion**

In water jug problem, if we apply a simple control strategy of starting each time from the top of rule list and choose the first applicable one, then we will never move towards solution.

✓ **It should explore the solution space in a systematic manner**

If we choose another control strategy, let us say, choose a rule randomly from the applicable rules then definitely it causes motion and eventually will lead to a solution. But one may arrive to same state several times. This is because control strategy is not systematic.

Systematic Control Strategies (Blind searches)

Breadth First Search

Let us discuss these strategies using water jug problem. These may be applied to any search problem.

- Construct a tree with the initial state as its root.
- Generate all the offspring of the root by applying each of the applicable rules to the initial state.

$$\begin{tikzcd} & (0, 0) \arrow[dl] \arrow[dr] & \\ (4, 0) & & (0, 3) \end{tikzcd}$$

- Now for each leaf node, generate all its successors by applying all the rules that are appropriate

$$\begin{tikzcd}
 & & & (0, 0) \arrow[dll] \arrow[drr] & & & \\
 & (4, 0) \arrow[dl] \arrow[d] \arrow[dr] & & & & (0, 3) \arrow[dl] \arrow[d] \arrow[dr] & \\
(4, 3) & (0, 0) & (1, 3) & & (4, 3) & (0, 0) & (3, 0)
\end{tikzcd}$$

- Continue this process until some rule produces a goal state.

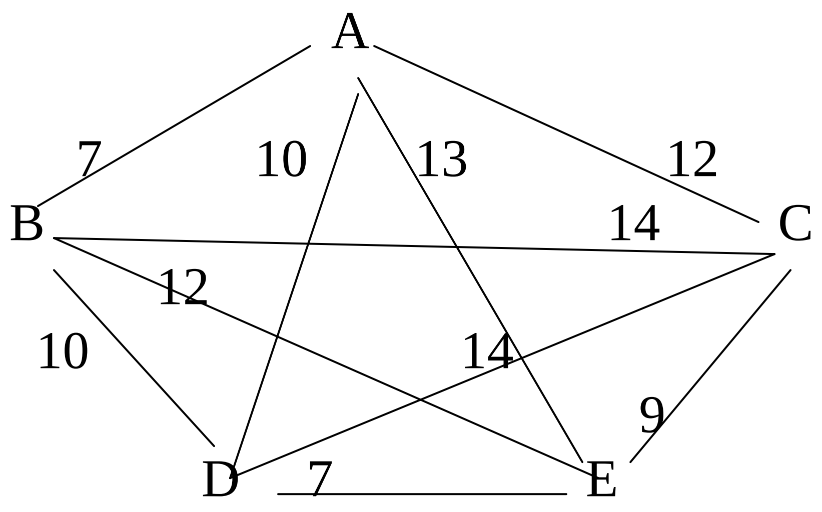
Depth First Search

- Here we could pursue a single branch of the tree until it yields a solution or until some pre-specified depth has reached and then go back to immediate previous node and explore other branches in DF fashion.
    - ✓ Blind searches are exhaustive in nature
    - ✓ These are uninformed searches
    - ✓ If the problem is simple then any control strategy that causes motion and is systematic will lead to an answer. But in order to solve some real world problems, we must also demand a control strategy that is efficient.
- Let us see the tree formation for water jug problem using DFS

$$\begin{tikzcd}[column sep=small]
 & & & & & (0, 0) \arrow[dl] & & \textbf{Start State} \arrow[ll] \\
 & & & & (4, 0) \arrow[dll] \arrow[d] \arrow[drr] & & & \\
 & & (4, 3) \arrow[dl] \arrow[dr] & & (0, 0) & & (1, 3) & \\
 & (0, 3) \arrow[dl] \arrow[d] \arrow[dr] & & (4, 0) & & & & \\
\times\ (4, 3) & \times\ (0, 0) & (3, 0) \arrow[dl] \arrow[d] \arrow[dr] & & & & & \\
 & (3, 3) \arrow[dl] \arrow[d] \arrow[dr] & (0, 0) & (0, 3) & & & & \\
\times\ (3, 0) & \times\ (0, 3) & (4, 2) \arrow[dl] \arrow[d] \arrow[dr] & & & & & \\
 & (0, 2) \arrow[dl] \arrow[d] \arrow[dr] & (3, 3) & (4, 0) & & & & \\
\times\ (4, 2) & \times\ (0, 0) & (2, 0) & & \textbf{Goal state} \arrow[ll] & & &
\end{tikzcd}$$

Traveling Salesman Problem (with 5 cities):

A salesman is supposed to visit each of 5 cities shown below. There is a road between each pair of cities and the distance is given next to the roads. Start city is A. The problem is to find the shortest route so that the salesman visits each of the cities only once and returns to back to A.



- A simple, motion causing and systematic control structure could, in principle solve this problem.
- Explore the search tree of all possible paths and return the shortest path.
- This will require 4! paths to be examined.
- If number of cities grow, say 25 cities, then the time required to wait a salesman to get the information about the shortest path is of 0(24!) which is not a practical situation.
- This phenomenon is called **combinatorial explosion**.
- We can improve the above strategy as follows.
  - ✓ Begin generating complete paths, keeping track of the shortest path found so far.
  - ✓ Give up exploring any path as soon as its partial length become greater than the shortest path found so far.
  - ✓ This algorithm is efficient than the first one, still requires exponential time $\propto$ some number raised to N.

Heuristic Search

- Heuristics are criteria for deciding which among several alternatives be the most effective in order to achieve some goal.
- Heuristic is a technique that improves the efficiency of a search process possibly by sacrificing claims of systematic and completeness. It no longer guarantees to find the best answer but almost always finds a very good answer.
- Using good heuristics, we can hope to get good solution to hard problems (such as travelling salesman) in less than exponential time.
- There are **general-purpose** heuristics that are useful in a wide variety of problem domains.
- We can also construct **special purpose** heuristics, which are domain specific.

General Purpose Heuristics

- A general-purpose heuristics for combinatorial problem is **nearest neighbor algorithms** which works by selecting the locally superior alternative.
- For such algorithms, it is often possible to prove an upper bound on the error which provide reassurance that one is not paying too high a price in accuracy for speed.
- In many AI problems, it is often hard to measure precisely the goodness of a particular solution.
- For real world problems, it is often useful to introduce heuristics based on relatively unstructured knowledge. It is impossible to define this knowledge in such a way that mathematical analysis can be performed.
- In AI approaches, behavior of algorithms are analyzed by running them on computer as contrast to analyzing algorithm mathematically.

- There are at least many reasons for the adhoc approaches in AI.
    - ✓ It is a lot more fun to see a program do something intelligent than to prove it.
    - ✓ AI problem domains are usually complex, so generally not possible to produce analytical proof that a procedure will work.
    - ✓ It is even not possible to describe the range of problems well enough to make statistical analysis of program behavior meaningful.
- But still it is important to keep performance question in mind while designing algorithm.
- One of the most important analysis of the search process is straightforward i.e., “Number of nodes in a complete search tree of depth D and branching factor F is F*D ”.
- This simple analysis motivates to
    - ✓ look for improvements on the exhaustive search.
    - ✓ find an upper bound on the search time which can be compared with exhaustive search procedures.

Summary

- Heuristics are useful in AI and real world problems
- General purpose heuristic - nearest neighbor
- Special purpose heuristic - domain specific
- In real world problems
    - Knowledge is not well structured so mathematical analysis is not accurately performed.
    - It is often difficult to measure the goodness of a particular solution
    - Behavior of algorithm is analyzed by running it on computer.
    - But performance question is important while designing algorithm.
    - Simplest way to measure performance of search process is to find number of nodes in complete search tree of depth D and branching factor F.

Problem Characteristics

Heuristic search is a very general method applicable to a large class of problem. In order to choose the most appropriate method (or combination of methods) for a particular problem it is necessary to analyze the problem along several key dimensions:

1. Is the problem decomposable into a set of independent smaller sub problems?

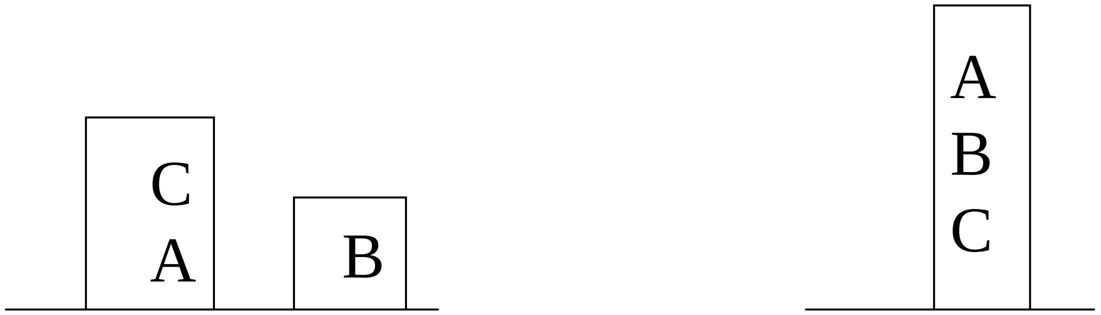
Example: Suppose we want to solve the problem of computing the integral of the following expression

$$\int (x^2 + 3x + \sin^2 x * \cos^2 x)\, dx$$

$$\begin{tikzcd}
 & \int (x^2 + 3x + \sin^2 x * \cos^2 x)\, dx \arrow[dl, no head] \arrow[d, no head] \arrow[drr, no head] & & & \\
\int x^2\, dx \arrow[d, no head] & \int 3x\, dx \arrow[d, no head] & & \int (\sin^2 x * \cos^2 x)\, dx \arrow[d, no head] & \\
x^3 / 3 & 3x^2 / 2 & & \int (1-\cos^2 x)*\cos^2 x\, dx \arrow[dl, no head] \arrow[dr, no head] & \\
 & & \int \cos^2 x\, dx \arrow[d, dotted, no head] & & \int \cos^4 x\, dx \arrow[d, dotted, no head] \\
 & & {} & & {}
\end{tikzcd}$$

- We can solve this problem by breaking it down into three smaller sub problems, each of which we can then be solved using a small collection of specific rules.
- Decomposable problems can be solved by the **divide-and-conquer** technique.
- Use of decomposing problems:
  - Each sub-problem is simpler to solve
  - Each sub-problem can be handed over to a different processor. Thus can be solved in parallel processing environment.
- There are non decomposable problems. For example, **Block world** problem is non decomposable.

Initial State (State0) Goal State



Start: ON(C, A) ….

Goal: ON(B, C) $\wedge$ ON(A, B)

2. Can solution steps be ignored or at least undone if they prove to be unwise?

- In real life, there are three types of problems: Ignorable, Recoverable and Irrecoverable.
- Let us explain each of these through examples.

Example1: **(Ignorable)**: **In theorem proving** - (solution steps can be ignored)

- Suppose we have proved some lemma in order to prove a theorem and eventually realized that lemma is no help at all, then ignore it and prove another lemma.
- Can be solved by using simple **control strategy**.

Example2: (**Recoverable**): **8 puzzle -** (solution steps can be undone)

- 8 puzzle: Objective is to rearrange a given initial configuration of eight numbered tiles on 3 X 3 board (one place is empty) into a given final configuration (goal state).
- Rearrangement is done by sliding one of the tiles into empty square.
- Solved by backtracking so control strategy must be implemented using a push down stack.

Example3: (**Irrecoverable**): **Chess** (solution steps cannot be undone)

- A stupid move cannot be undone
- Can be solved by planning process

3. Is the knowledge Base consistent?

Example: **Inconsistent knowledge**:

Target problem: A man is standing 150 ft from a target. He plans to hit the target by shooting a gun that fires bullets with velocity of 1500 ft/sec. How high above the target should he aim?

Solution:

- Velocity of bullet is 1500 ft./sec i.e., bullet takes 0.1 sec to reach the target.
- Assume bullet travels in a straight line.
- Due to gravity, the bullet falls at a distance $(1/2)gt^2 = (1/2)(32)(0.1)^2 = 0.16$ft.
- So if man aims up 0.16 feet high from the target, then bullet will hit the target.

- Now there is a contradiction to the fact that bullet travel in a straight line because the bullet in actual will travel in an arc. Therefore there is inconsistency in the knowledge used.

4. What is the Role of knowledge?

- In Chess game, knowledge is important to constrain the search for a solution otherwise just the rule for determining legal moves and some simple control mechanism that implements an appropriate search procedure is required.
- Newspapers scanning to decide some facts, a lot of knowledge is required even to be able to recognize a solution.

5. Is a good solution **Absolute** or **Relative ?**

- In water jug problem there are two ways to solve a problem. If we follow one path successfully to the solution, there is no reason to go back and see if some other path might also lead to a solution. Here a solution is absolute.
- In travelling salesman problem, our goal is to find the shortest route. Unless all routes are known, the shortest is difficult to know. This is a best-path problem whereas water jug is any-path problem.
- Any path problem can often be solved in reasonable amount of time using heuristics that suggest good paths to explore.
- Best path problems are in general computationally harder than any-path.