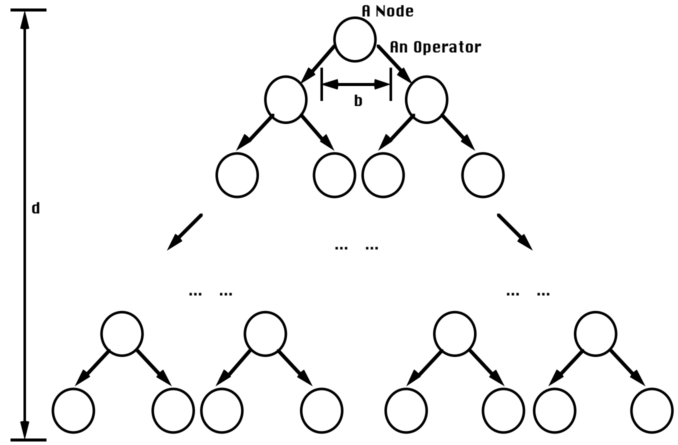
Problem Solving Paradigms

6.871 Knowledge Based Systems
Howard Shrobe, Randy Davis
Lecture #2:
Technical Background and Our Perspective

- This introduction discusses the nature of problem solving tasks, the notion of search in a problem space and the exponential growth of search related methods.
- We'll present a perspective of using large stores of knowledge to avoid using search oriented methods.
- These include both large numbers of facts and standard methods for solving specific types of tasks.
  - Knowing what
  - Knowing how
- Problem Solving Paradigms provide the key unifying concept. They dictate the choice of knowledge representation and they organize the inferential processes.

The Fundamental Problem Confronting An AI System: Search in A Problem Space

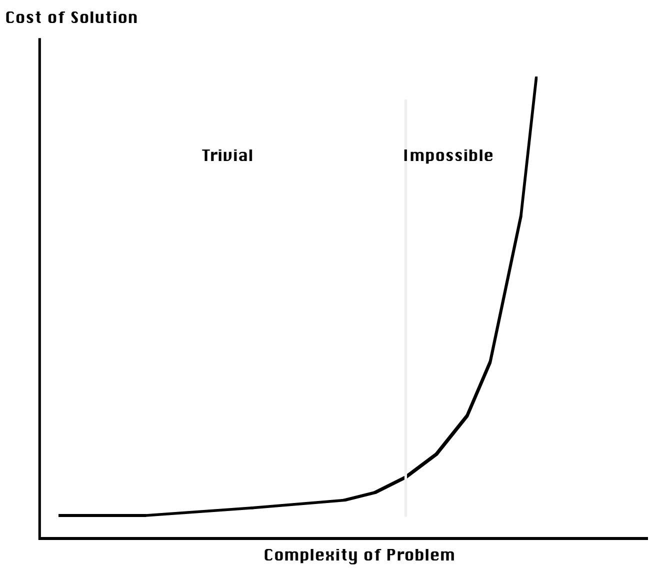


d = Depth of Tree, number of operators to find a solution.

b = Branching factor, number of relevant operators at each node.

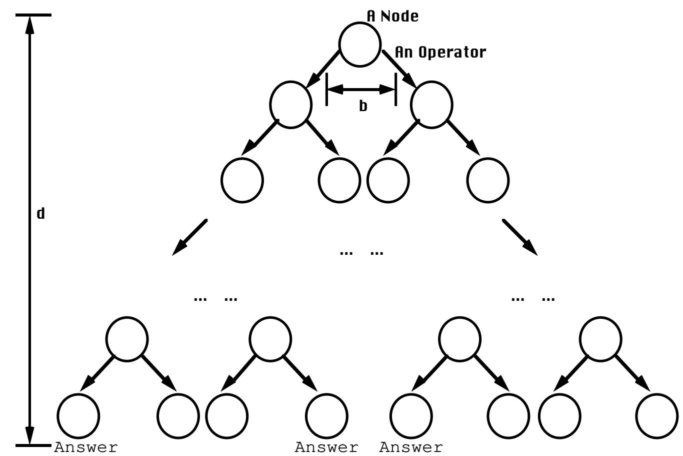
Size of Space = b^d

Search Spaces Grow Exponentially



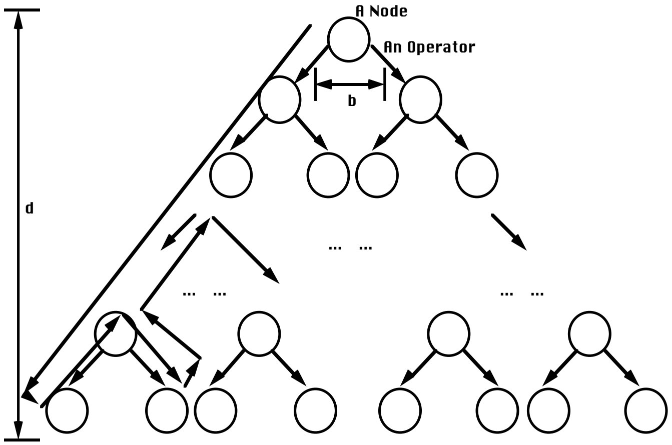
- The marginal cost of a slight improvement in solution quality is nearly infinite.
- Given the cost of optimality, people resort to finding a good enough solution.

The Shape of The Space

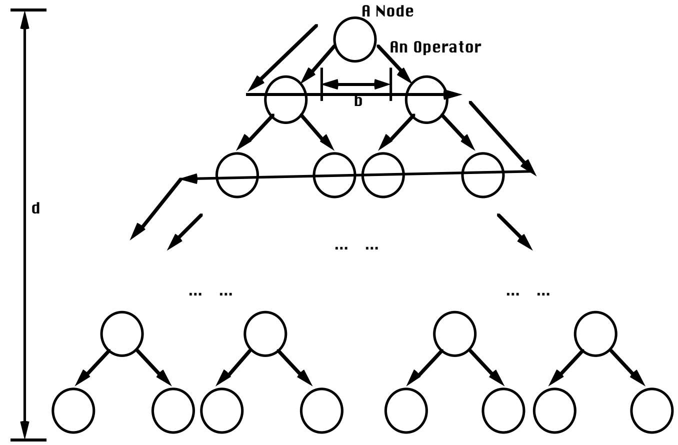


- How Densely Distributed are the Answers?
- How "close" together are the Answers?
- How do answer quality and "distance" relate?

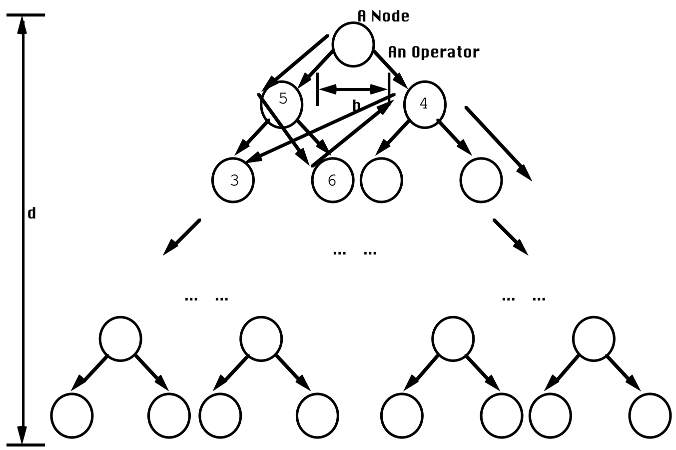
Depth First Search



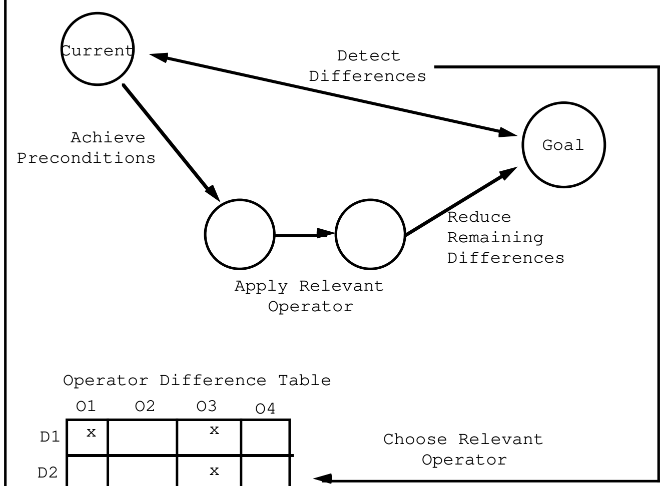
Breadth First Search



Best First Search



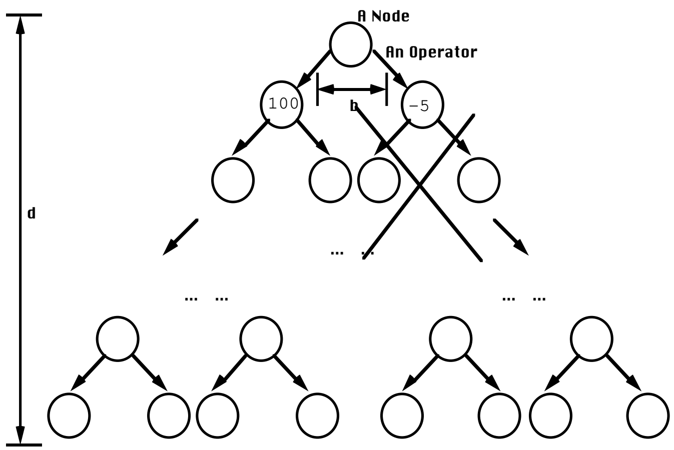
Means Ends Analysis



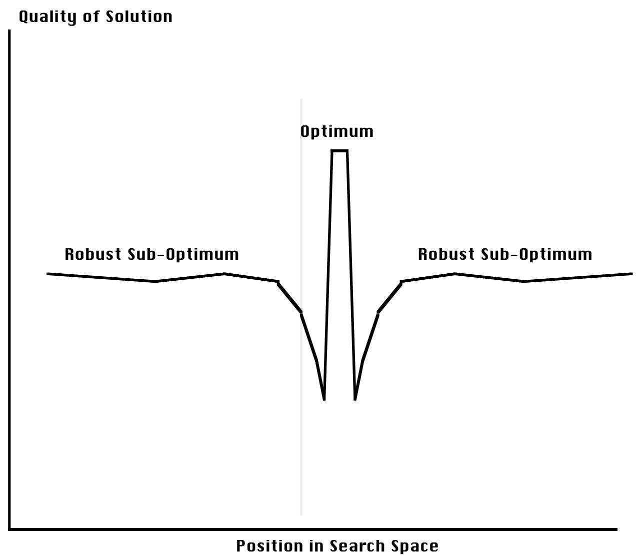
Operator Difference Table

	O1	O2	O3	O4
D1	x		x	
D2			x	
D3		x		x
D4				x

Pruning



Optimal Solutions Are Not Always Robust



- In Real world situations, the optimum is often disrupted:
  - People get sick
  - Machines break
- The expected benefit of a sub-optimal solution may be much higher than that of the optimum.

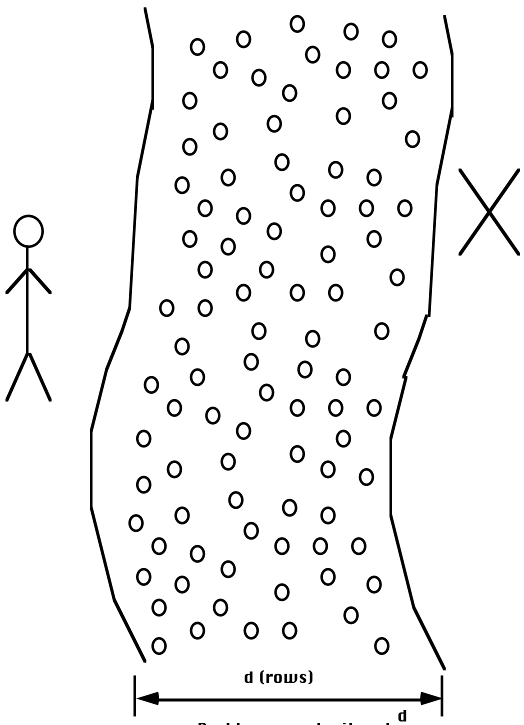
The Lessons of Classic AI Systems:

- The attempt to solve interesting problems is hampered by the intractability of search in large problem spaces.
- Uninformed search can only find answers to trivial problems.
  - The goal is not to build a powerful search engine, but rather to avoid search as much as possible.
  - The Use of Rules *per se* has no bearing on this.
  - To the contrary, all rule systems are search engines.
- Viewed most generally, people use two techniques to cope with this complexity:
  - Recognition of the form of the problem.
  - Abstraction.
- Both of these are a means for finding a plausible set of planning islands.
- Planning islands change a large exponential problem into several smaller ones.
- People represent difficult problems in symbolic form, because this allows the appropriate abstractions to be captured.

The Lessons of Three Pioneer AI Systems:

- Macsyma showed that:
  - Recognition of the form of the problem leads to knowledge of the form of the problem.
  - The use of appropriate representation is crucial. It is cheaper to use the right representation for each part of the problem and to translate between them.
  - The burden of translation is reduced by using a single *iter-lingua* for information interchange.
  - It is important to develop a deep understanding of the domain and to capture powerful, domain specific techniques.
- Strips and Abstrips showed that:
  - Abstraction is key: Solving the problem in an abstracted space produces a skeleton plan cheaply. The cost of refining the skeleton plan is trivial compared to that of blind search.
  - Macro-operators (formed from the results of previous problem solving) can recognize the form of larger scale problems and cheaply suggest the form of their solution.
- Dendral showed that:
  - The specific knowledge of experts in a domain can apply constraints to the search space which prune it sufficiently to reduce search to tractable levels.
  - The importance of multiple sources of knowledge.

Planning Islands The Power of Recognition

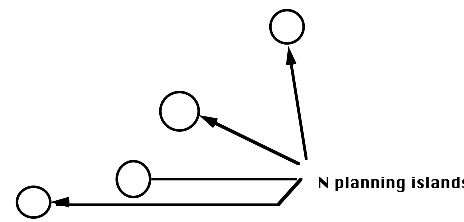


Problem complexity = b^d

A Little Knowledge Goes A Long Way

N subproblems:
- Each d/N deep.
- Each of size $b^{d/n}$

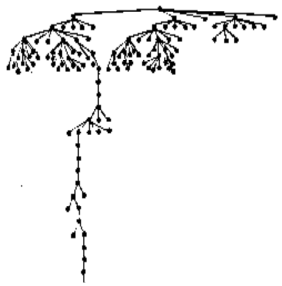
Total size: $N * b^{d/n}$



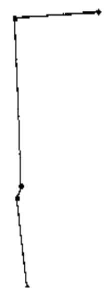
E.g. b = 2, d = 10, n = 5

- Without Islands: 1024
- With Islands: 5 * 4 = 20
- You can guess wrong 50 times and still be ahead of the game!

Abstrips and The Power of Abstraction

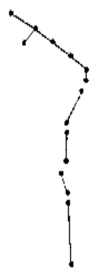


Search Space for the Sample Problem

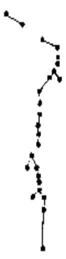


Abstrips Search Tree in the Space of Criticality 6

Abstrips Search Tree in the Space of Criticality 5



Abstrips Search Tree in the Space of Criticality 2



Abstrips Search Tree in the Problem Space.

A Recipe for Success:

- Study the problem domain seriously, looking for powerful general purpose techniques.
- Study the way the experts characterize problems and how they characterize the styles of solutions.
- Pay attention to both the procedural and the declarative knowledge.
- Study the technical vocabulary of the experts, trying to understand the abstractions represented by this.

- Use or create a representation which captures these abstractions.
- Build a problem solving mechanism which reflects the problem solving behavior of the experts.
- Capitalize on the speed and memory advantages of the computer.

Our Perspective: Representations and Problem Solving Paradigms

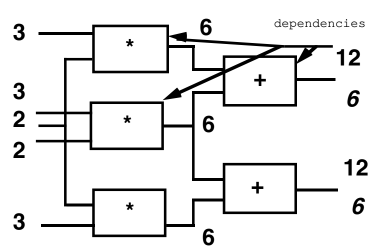
- There appear to be a modest number of problem solving paradigms and their representational frameworks which provide the groundwork for all expert problem solving (by modest I mean a number more like 1000, than either 1 or 1,000,000).
- The role of a representation is to provide a perspective, a "pair of glasses" with a distinct tint that tells us how to look at the world.
- The role of a representation is to capture directly the important abstractions of the problem domain.
- Although a representation may suggest certain data-structures, it is not synonymous with them.
- The role of a problem solving paradigm is to organize the representational, inferential and computational processes, to tell us when and how to draw and state conclusions.
- Although a problem solving paradigm may suggest certain inference mechanisms, it is not synonymous with them.

What is A Problem Solving Paradigm?

- A representational framework for factual knowledge.
- A representational framework for inferential methods (and procedural knowledge).
- A control structure dictating when to employ inferential methods and with what purpose.
- Given a problem solving paradigm, you make a problem solver by:
  - Providing the relevant factual information at the right level of abstraction.
  - Providing the relevant inference methods.
- This has the following advantages:
  - Encoding of Knowledge is simpler.
  - Expressive quality and maintainability is improved.
  - The need for "programming hacks" is reduced.
  - Knowledge acquisition is made simpler.
  - Special purpose languages for the paradigm can be created and reused.
  - The search for the right level of abstraction is emphasized.

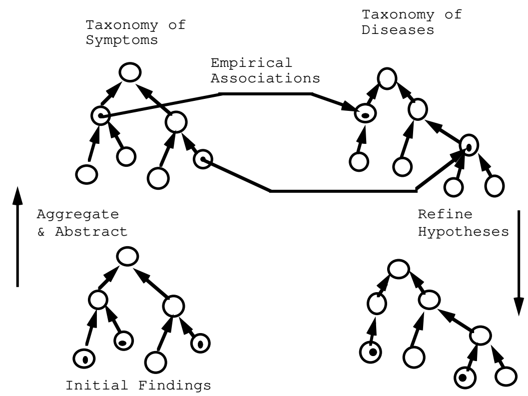
An Example Paradigm: Model Based Troubleshooting

- Representation:
  - Hierarchical decomposition of artifact into components.
  - Wiring diagram of connections.
  - Behavioral (invertible) models of components.
- Paradigm:
  - Apply test inputs to inputs of model.
  - Simulate by running behavioral models
  - Maintain dependencies between values and device models.
  - Identify discrepencies between predictions and observations.
  - Trace dependencies to possibly failing components.
  - Reduce to minimum set explaining all (and only) observed discrepencies.



An Example Paradigm: Mycin Three Phased Diagnosis.

- Three distinct sub problem solvers:
  - Abstract and Aggregate Symptoms and Experimental findings.
  - Abductive Leap from Symptom to Hypotheses.
  - Refinement of the Hypotheses.
- Three Representations:
  - Taxonomic classification of findings.
  - Empirical Associations between Symptoms and Causes.
  - Taxonomic classification of disease entities.



Relationship to AI Programming Languages

A programming language can provide several distinct forms of support for a problem solving paradigm:

- *Strong Support:* The primitives of the language bear a one to one relationship to the primitives of the paradigm's representational framework.
- *Embedding Support:* The primitives of the language can easily be assembled into the primitives of the paradigm's representational framework. The facitilites of the programming language remain visible.
- *Implementational Support:* The programming language is a universal computing language (of course) allowing the implementation of an interpreter or compiler for a language strongly supporting the intended paradigm.

Ideally we would like to provide strong support for paradigms.

Ideally we want to support heterogeneity.