Problem Spaces & Search

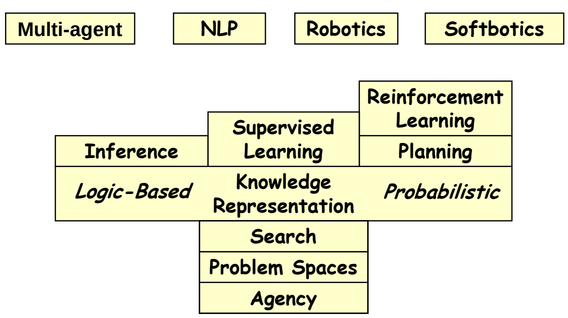
- CSE 573

Logistics

- Mailing list
- Reading
  - Ch 4.2, Ch 6
- Mini-Project I
  - Partners
- Game Playing?

© Daniel S. Weld

573 Topics



© Daniel S. Weld

Weak Methods

- "In the knowledge lies the power…"
- [Feigenbaum]
- But what if you have very little knowledge????

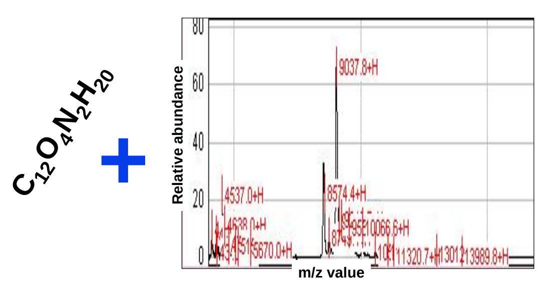
© Daniel S. Weld

Generate & Test

- As weak as it gets…
- Works on semi-decidable problems!

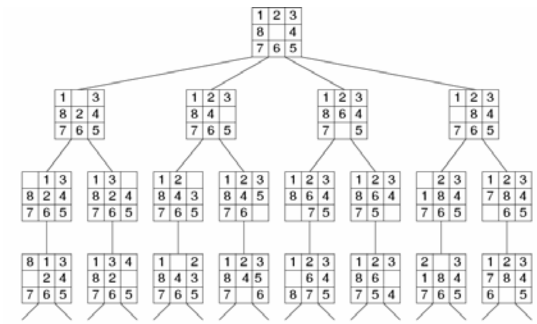
© Daniel S. Weld

Mass Spectrometry



© Daniel S. Weld

Example: Fragment of 8-Puzzle Problem Space



Search thru a Problem Space / State Space

- **Input:**
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state [test]
- **Output:**
  - Path: start ⇒ a state satisfying goal test
  - [May require shortest path]

Example: Route Planning

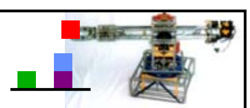
- **Input:**
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state (test)
- **Output:**

Example: N Queens

		Q	
Q			
			Q
	Q		

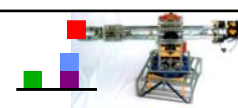
- **Input:**
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state (test)
- **Output**

Ex: Blocks World



- **Input:**
  - Set of states
    - Partially specified plans
  - Operators [and costs]
    - Plan modification operators
  - Start state
    - The null plan (no actions)
  - Goal state (test)
    - A plan which provably achieves
    - The desired world configuration

Multiple Problem Spaces



- **Real World**
  - States of the world (e.g. block configurations)
  - Actions (take one world-state to another)

Robot's Head

- **Problem Space 1**
- PS states =
  - models of world states
- Operators =
  - models of actions

- **Problem Space 2**
- PS states =
  - partially spec. plan
- Operators =
  - plan modificat'n ops

Classifying Search

- GUESSING ("Tree Search")
  - Guess how to extend a partial solution to a problem.
  - Generates a tree of (partial) solutions.
  - The leafs of the tree are either "failures" or represent complete solutions
- SIMPLIFYING ("Inference")
  - Infer new, stronger constraints by combining one or more constraints (without any "guessing")

    Example: X+2Y = 3
    X+Y =1
    therefore Y = 2
- WANDERING ("Markov chain")
  - Perform a (biased) random walk through the space of (partial or total) solutions

- *Guessing – State Space Search*
  1. BFS
  2. DFS
  3. Iterative Deepening
  4. Bidirectional
  5. Best-first search
  6. A*
  7. Game tree
  8. Davis-Putnam (logic)
  9. Cutset conditioning (probability)
- *Simplification – Constraint Propagation*
  1. Forward Checking
  2. Path Consistency (Waltz labeling, temporal algebra)
  3. Resolution
  4. "Bucket Algorithm"
- *Wandering – Randomized Search*
  1. Hillclimbing
  2. Simulated annealing
  3. Walksat
  4. Monte-Carlo Methods

Search Strategies v2

- Blind Search
  - Depth first search
  - Breadth first search
  - Iterative deepening search
  - Iterative broadening search
- Informed Search
- Constraint Satisfaction
- Adversary Search

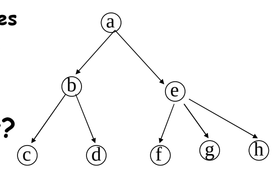
Depth First Search

- Maintain stack of nodes to visit
- Evaluation
  - Complete?

    Not for infinite spaces
  - Time Complexity?

    O(b^d)
  - Space Complexity?

    O(bd)



d=max depth of tree

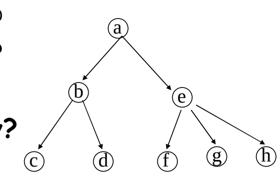
Breadth First Search

- Maintain queue of nodes to visit
- Evaluation
  - Complete?

    Yes (if branching factor is finite)
  - Time Complexity?

    $O(b^{d+1})$
  - Space Complexity?

    $O(b^{d+1})$

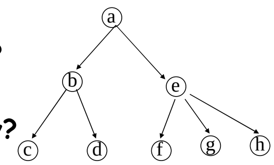


Memory a Limitation?

- Suppose:
  - 2 GHz CPU
  - 1 GB main memory
  - 100 instructions / expansion
  - 5 bytes / node
  - 200,000 expansions / sec
  - Memory filled in 100 sec … < 2 minutes

Iterative Deepening Search

- DFS with limit; incrementally grow limit
- Evaluation
  - Complete?
    Yes
  - Time Complexity?
    $O(b^d)$
  - Space Complexity?
    $O(d)$



Cost of Iterative Deepening

b	ratio ID to DFS
2	3
3	2
5	1.5
10	1.2
25	1.08
100	1.02

Speed

- Assuming 10M nodes/sec & sufficient memory

	BFS Nodes	BFS Time		Iter. Deep. Nodes	Iter. Deep. Time
8 Puzzle	10^5	.01 sec		10^5	.01 sec
2x2x2 Rubik's	10^6	.2 sec		10^6	.2 sec
15 Puzzle	10^{13}	6 days	1Mx	10^{17}	20k yrs
3x3x3 Rubik's	10^{19}	68k yrs	8x	10^{20}	574k yrs
24 Puzzle	10^{25}	12B yrs		10^{37}	10^{23} yrs

Why the difference?
Rubik has higher branch factor
15 puzzle has greater depth

\# of duplicates

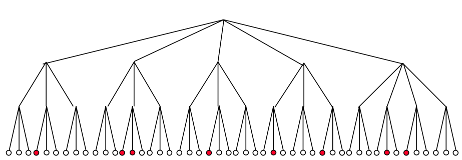
- Adapted from Richard Korf presentation

When to Use Iterative Deepening

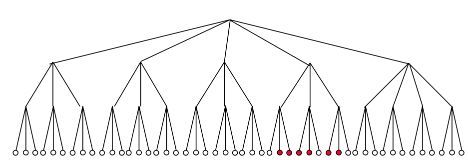
- N Queens?

		Q	
Q			
			Q
	Q		

Search Space with Uniform Structure

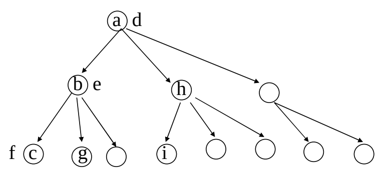


Search Space with Clustered Structure



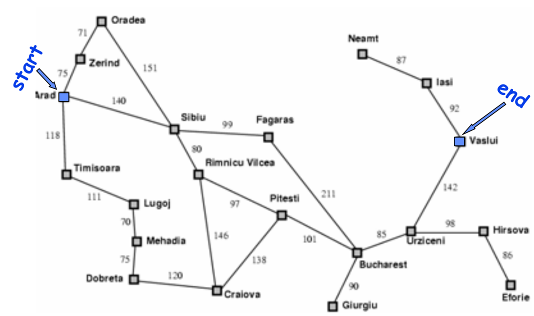
Iterative Broadening Search

- What if know solutions lay at depth N?
- No sense in doing iterative *deepening*
  - Increment sum of sibling indicies



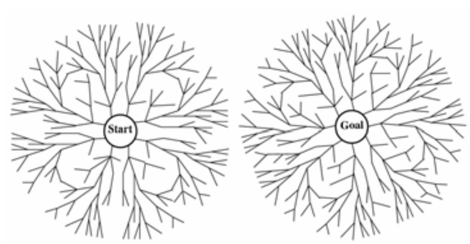
© Daniel S. Weld

Forwards *vs.* Backwards



© Daniel S. Weld

vs. Bidirectional



© Daniel S. Weld

Problem

- All these methods are slow (blind)
- Solution
  → add guidance ("heurstic estimate")
  → "informed search"

… coming next …

© Daniel S. Weld

Recap: Search thru a Problem Space / State Space

- Input:
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state [test]
- Output:
  - Path: start ⇒ a state satisfying goal test
  - [May require shortest path]

© Daniel S. Weld

Cryptarithmetic

```
  SEND
+ MORE
------
 MONEY
```

- Input:
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state (test)
- Output:

© Daniel S. Weld

Concept Learning

```
Labeled Training Examples
     <p1,blond,32,mc,ok>
     <p2,red,47,visa,ok>
     <p3,blond,23,cash,ter>
     <p4,…
Output: f: <pn…> → {ok, ter}
```

- Input:
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state (test)
- Output:

© Daniel S. Weld

Symbolic Integration

- E.g. $\int x^2 e^x \, dx =$
- $e^x(x^2-2x+2) + C$

Operators:

Integration by parts

Integration by substitution

…

© Daniel S. Weld

Search Strategies v2

- Blind Search
- Informed Search
  - Best-first search
  - A* search
  - Iterative deepening A* search
  - Local search
- Constraint Satisfaction
- Adversary Search

© Daniel S. Weld

Best-first Search

- Generalization of breadth first search
- Priority queue of nodes to be explored
- Cost function f(n) applied to each node

```
Add initial state to priority queue
While queue not empty
    Node = head(queue)
        If goal?(node) then return node
    Add children of node to queue
```

© Daniel S. Weld

Old Friends

- Breadth first = best first
  - With f(n) = depth(n)
- Dijkstra's Algorithm = best first
  - With f(n) = g(n), i.e. the sum of edge costs from start to n
  - Space bound (stores all generated nodes)

© Daniel S. Weld

A* Search

- Hart, Nilsson & Rafael 1968
  - Best first search with f(n) = g(n) + h(n)
  - Where g(n) = sum of costs from start to n
  - h(n) = estimate of lowest cost path n → goal

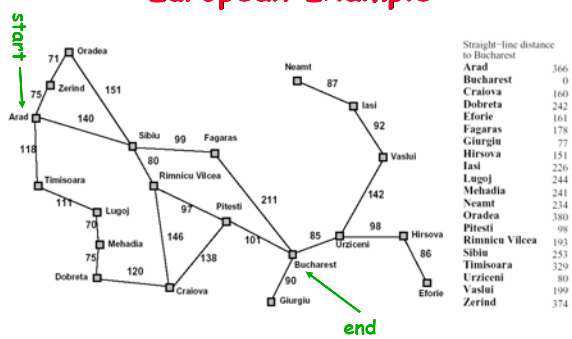
    h(goal) = 0
  - If h(n) is admissible (and monotonic) then A* is optimal

Underestimates cost of any solution which can reached from node

f values increase from node to descendants (triangle inequality)

© Daniel S. Weld

European Example

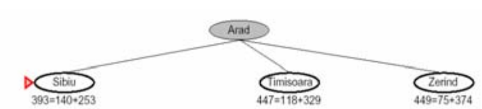


© Daniel S. Weld

A* Example

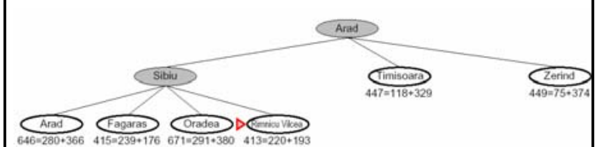
© Daniel S. Weld

A* Example



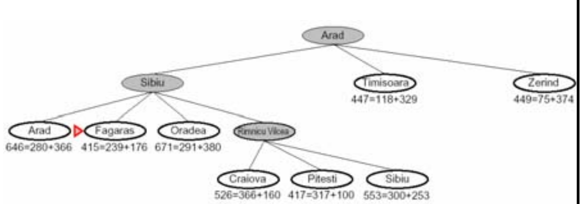
© Daniel S. Weld

A* Example



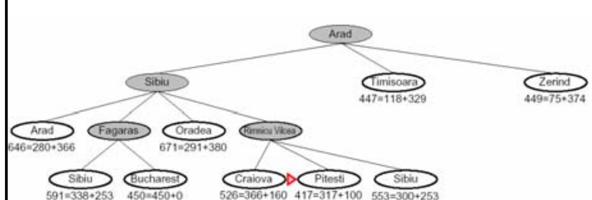
© Daniel S. Weld

A* Example



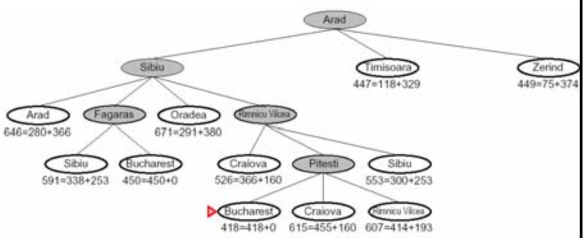
© Daniel S. Weld

A* Example



© Daniel S. Weld

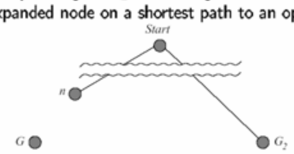
A* Example



© Daniel S. Weld

Optimality of A*

Suppose some suboptimal goal G_2 has been generated and is in the queue. Let n be an unexpanded node on a shortest path to an optimal goal G_1.



$$\begin{aligned} f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\ &> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\ &\geq f(n) && \text{since } h \text{ is admissible} \end{aligned}$$

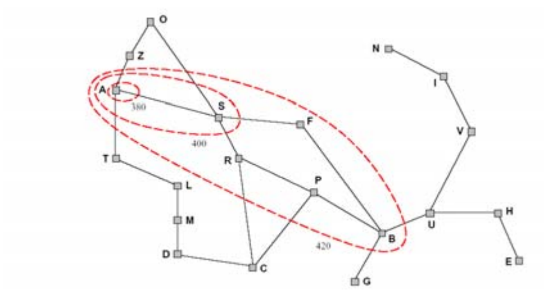
Since $f(G_2) > f(n)$, A* will never select G_2 for expansion

© Daniel S. Weld

Optimality Continued

Lemma: A* expands nodes in order of increasing f value*

Gradually adds "f-contours" of nodes (cf. breadth-first adds layers)
Contour i has all nodes with $f = f_i$, where $f_i < f_{i+1}$



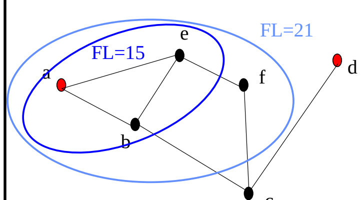
A* Summary

- Pros
- Cons

© Daniel S. Weld

Iterative-Deepening A*

- Like iterative-deepening depth-first, but...
- Depth bound modified to be an f-limit
  - Start with limit = h(start)
  - Prune any node if f(node) > f-limit
  - Next f-limit = min-cost of any node pruned



© Daniel S. Weld

IDA* Analysis

- Complete & Optimal (ala A*)
- Space usage $\propto$ depth of solution
- Each iteration is DFS - no priority queue!
- # nodes expanded relative to A*
  - Depends on # unique values of heuristic function
  - In 8 puzzle: few values $\Rightarrow$ close to # A* expands
  - In traveling salesman: each f value is unique
    $\Rightarrow$ 1+2+…+n = $O(n^2)$ where n=nodes A* expands
    if n is too big for main memory, n^2 is too long to wait!
- Generates duplicate nodes in cyclic graphs

© Daniel S. Weld