Finish Search

Computer Science cpsc322, Lecture 10

(Textbook Chpt 3.6)

January, 25, 2010

Lecture Overview

- Optimal Efficiency
- Pruning Cycles and Repeated states Examples
- Dynamic Programming
- 8-puzzle Applet
- Search Recap

Optimal efficiency of A^*

- In fact, we can prove something even stronger about A^*: in a sense (given the particular heuristic that is available) no search algorithm could do better!

- Optimal Efficiency: Among all optimal algorithms that start from the same start node and use the same heuristic h, A^* expands the minimal number of paths.

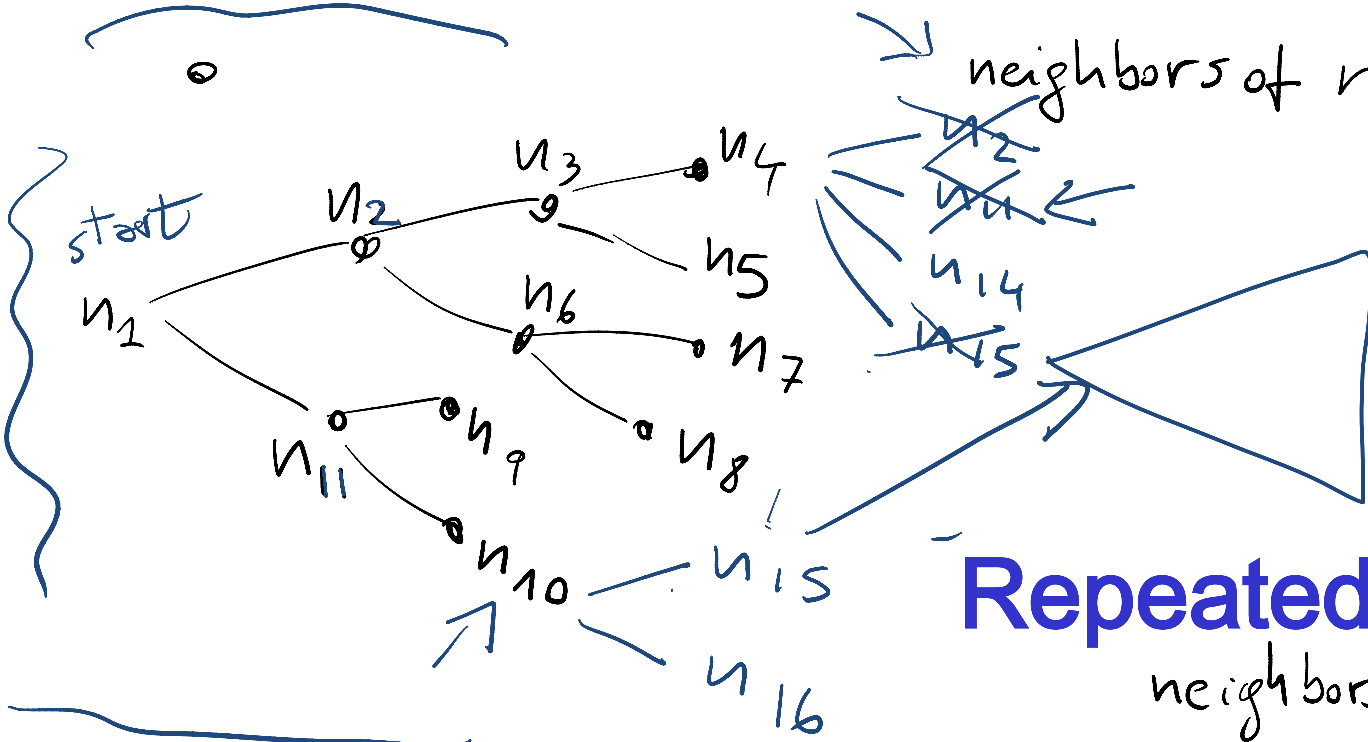
Lecture Overview

- Optimal Efficiency Example
- **Pruning Cycles and Repeated states Examples**
- Dynamic Programming
- 8-puzzle Applet
- Search Recap

Example

Pruning Cycles

neighbors of $n_4 = \{n_2, n_{11}, n_{14}, n_{15}\}$



Repeated States

neighbors of $n_{10} =$
$\{n_{15}, n_{16}\}$

Lecture Overview

- Optimal Efficiency Example
- Pruning Cycles and Repeated states Examples
- Dynamic Programming
- 8-puzzle Applet
- Search Recap

Dynamic Programming

Idea: for statically stored graphs, build a table of *dist(n)* the actual distance of the shortest path from node *n* to a goal.

This is the perfect……. heuristics

This can be built backwards from the goal:

$$dist(n) = \begin{cases} 0 & \text{if} \quad is\_goal(n), \\ \min_{\langle n,m\rangle \in A} \left(cost(n,m) + \boxed{dist(m)} \right) & otherwise \end{cases}$$

all the neighbors m

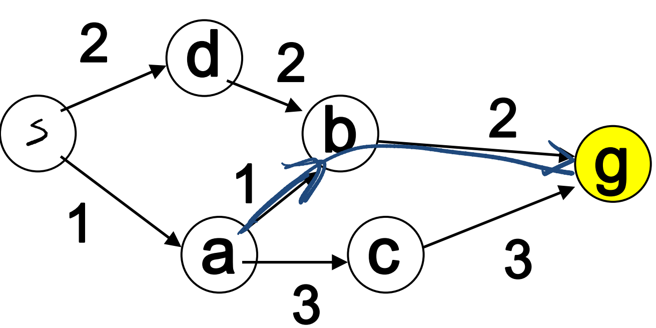
dist(n)

g 0

dist(b) = min[(2 + 0)] = 2

dist(c) = min[(3 + 0)] = 3

dist(a) = min[(3+3), (1+2)] = 3

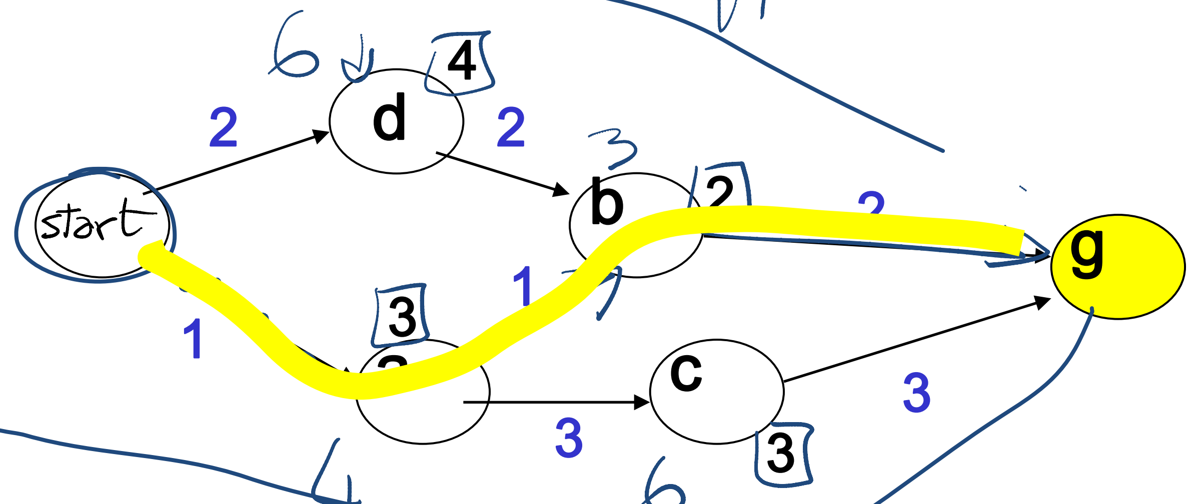


Dynamic Programming

This can be used locally to determine what to do.

From each node n go to its neighbor which minimizes

$$\text{cost}(n, m) + \mathit{dist}(m)$$



But there are at least two main problems:

- You need enough space to store the graph.
- The *dist* function needs to be recomputed for each goal

Lecture Overview

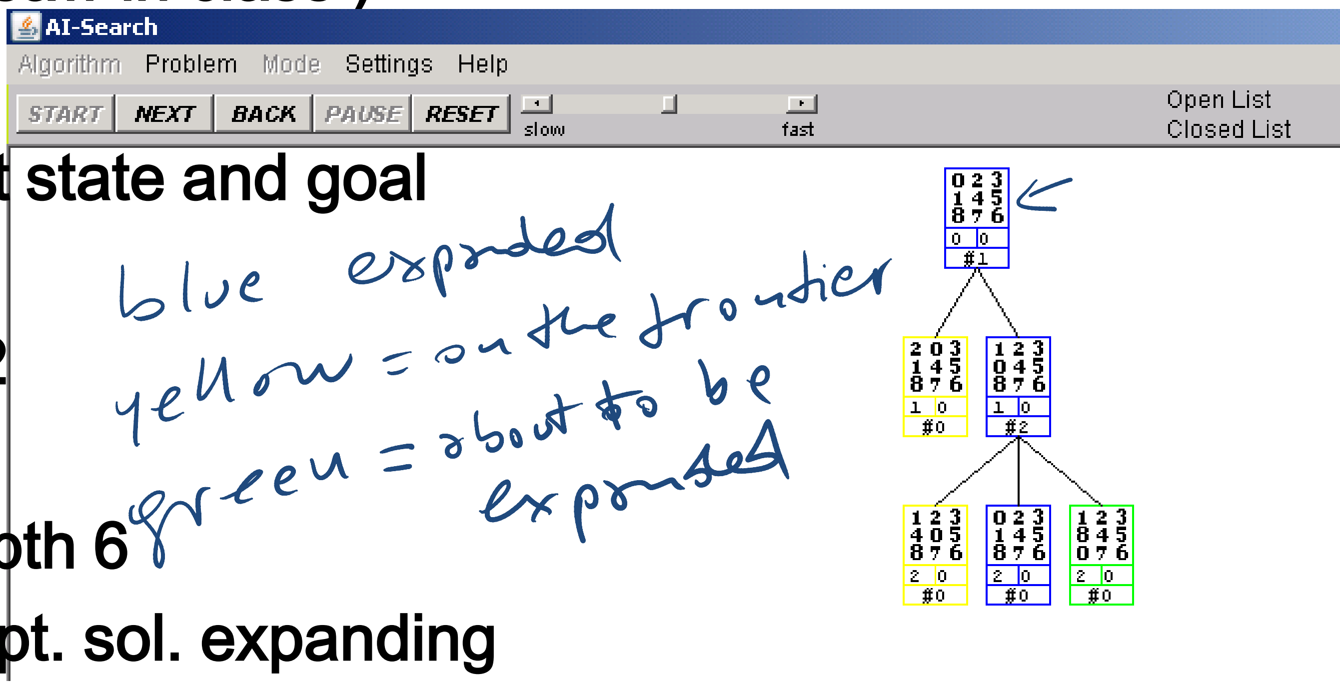
- Optimal Efficiency Example
- Pruning Cycles and Repeated states Examples
- Dynamic Programming
- 8-puzzle Applet
- Search Recap

DFS, BFS, A* Animation Example

- The AI-Search animation system

http://www.cs.rmit.edu.au/AI-Search/Product/

- *To examine Search strategies when they are applied to the 8puzzle*
- Compare only DFS, BFS and A* (with only the two heuristics we saw in class)



- With default start state and goal
- DFS will find

Solution at depth 32

- BFS will find

Optimal solution depth 6

- A* will also find opt. sol. expanding much less nodes

nPuzzles are not always solvable

Half of the starting positions for the *n*-puzzle are impossible to resolve (for more info on 8puzzle) http://www.isle.org/~sbay/ics171/project/unsolvable

- So experiment with the AI-Search animation system with the default configurations.
- If you want to try new ones keep in mind that you may pick unsolvable problems

Lecture Overview

- Optimal Efficiency Example
- Pruning Cycles and Repeated states Examples
- Dynamic Programming
- 8-puzzle Applet
- Search Recap

Recap Search

	Selection	Complete	Optimal	Time	Space
DFS	LIFO	N	N	$O(b^m)$	$O(mb)$
BFS	FIFO	Y	Y	$O(b^m)$	$O(b^m)$
IDS(C)	LIFO	Y	Y	$\boxed{O(b^m)}$	$O(mb)$
LCFS	min cost	Y	Y	$O(b^m)$	$O(b^m)$
BFS	min h	N	N	$O(b^m)$	$O(b^m)$
A*	min f=c+h	Y	Y	$O(b^m)$	$O(b^m)$
B&B	LIFO + pruning	N	Y	$O(b^m)$	$O(mb)$
IDA*	LIFO	Y	Y	$O(b^m)$	$O(mb)$
MBA*	min f	N	Y	$O(b^m)$	$O(b^m)$

Recap Search (some qualifications)

	Complete	Optimal	Time	Space
DFS	N	N	$O(b^m)$	$O(mb)$
BFS	Y	Y	$O(b^m)$	$O(b^m)$
IDS(C)	Y	Y	$O(b^m)$	$O(mb)$
LCFS	Y	Y ? $c \geq 0$	$O(b^m)$	$O(b^m)$
BFS	N	N	$O(b^m)$	$O(b^m)$
A*	Y	Y ? h admissible	$O(b^m)$	$O(b^m)$
B&B	N	Y ?	$O(b^m)$	$O(mb)$
IDA*	Y	Y	$O(b^m)$	$O(mb)$
MBA*	N	Y	$O(b^m)$	$O(b^m)$

Search in Practice

	Complete	Optimal	Time	Space
DFS	N	N	$O(b^m)$	$O(mb)$
BFS	Y	Y	$O(b^m)$	$O(b^m)$
IDS(C)	Y	Y	$O(b^m)$	$O(mb)$
LCFS	Y	Y	$O(b^m)$	$O(b^m)$
BFS	N	N	$O(b^m)$	$O(b^m)$
A*	Y	Y	$O(b^m)$	$O(b^m)$
B&B	N	Y	$O(b^m)$	$O(mb)$
IDA*	Y	Y	$O(b^m)$	$O(mb)$
MBA*	N	Y	$O(b^m)$	$O(b^m)$
BDS	Y	Y	$O(b^{m/2})$	$O(b^{m/2})$

Search in Practice (cont')

Informed?

F

IDS

T

Many paths to solution, no ∞ paths?

T

B&B

T

IDA*

Large branching factor?

F

MBA*

(Adversarial) Search: Chess

Deep Blue's Results in the second tournament:

- second tournament: won 3 games, lost 2, tied 1

- 30 CPUs + 480 chess processors
- Searched 126.000.000 nodes per sec
- Generated 30 billion positions per move reaching depth 14 routinely

- **Iterative Deepening** with evaluation function (similar to a **heuristic**) based on 8000 features (e.g., sum of worth of pieces: pawn 1, rook 5, queen 10)

Modules we'll cover in this course: R&Rsys

Environment

Problem		Deterministic	Stochastic
Static	Constraint Satisfaction	*Vars + Constraints* Arc Consistency Search	
Static	Query	*Logics* Search	*Belief Nets* Var. Elimination
Sequential	Planning	*STRIPS* actions preconds effects Search	*Decision Nets* Var. Elimination *Markov Processes* Value Iteration

Representation

Reasoning Technique

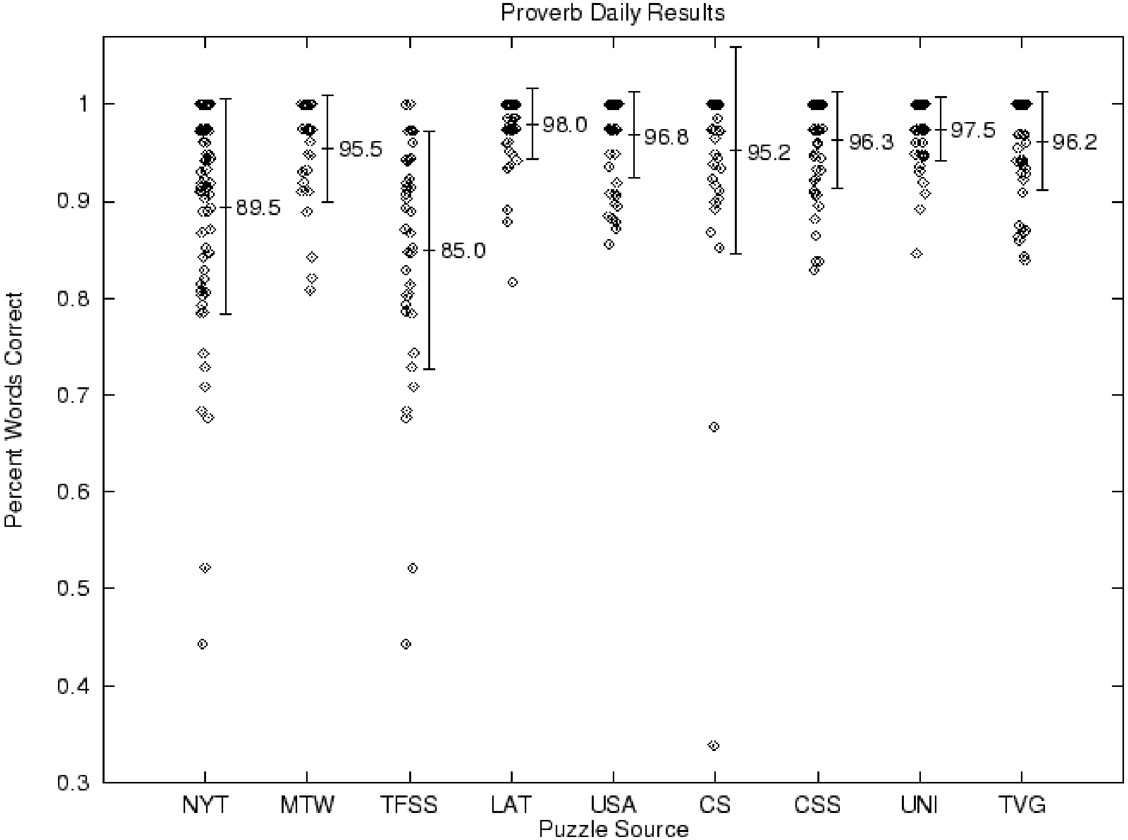
CSPs: Crossword Puzzles

Daily Puzzles

370 puzzles from 7 sources.

Summary statistics:

- 95.3% words correct (miss three or four words per puzzle)
- 98.1% letters correct
- 46.2% puzzles completely correct



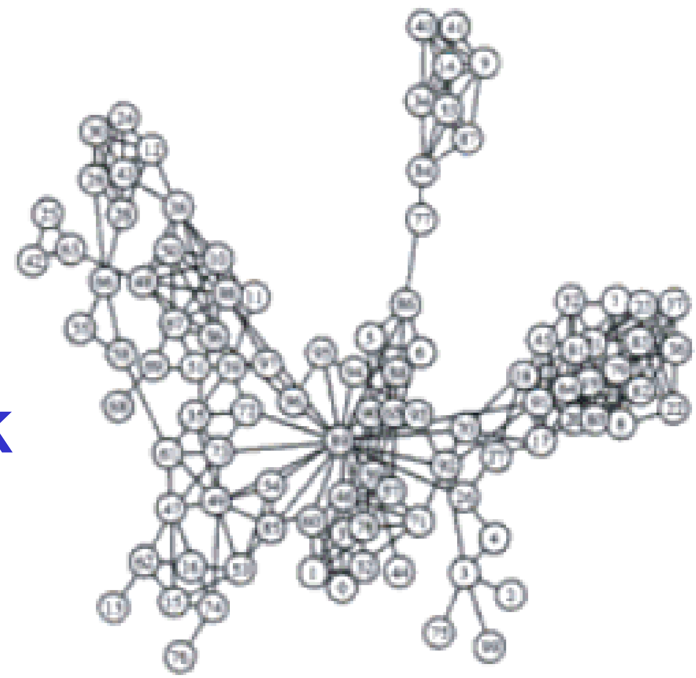
Source: *Michael Littman*

CSPs: Radio link frequency assignment

Assigning frequencies to a set of radio links defined between pairs of sites in order to avoid interferences.

Constraints on frequency depend on position of the links and on physical environment .

Source: *INRIA*

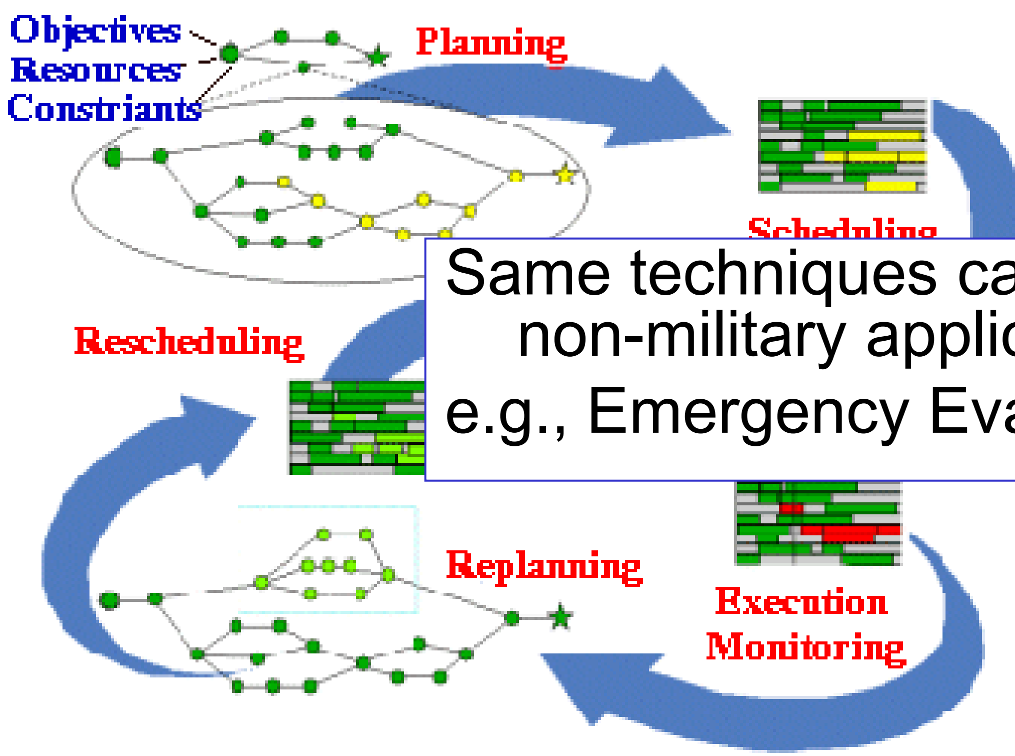


Sample Constraint network

Planning & Scheduling: Logistics

Dynamic Analysis and Replanning Tool (Cross & Walker)

- logistics planning and scheduling for military transport
- used for the first time in the 1991 Gulf War by the US
- problems had 50,000 entities (e.g., vehicles); different starting points and destinations



Same techniques can be use for non-military applications:
e.g., Emergency Evacuation

Source: *DARPA*

Standard Search vs. Specific R&R systems

Constraint Satisfaction (Problems):

- State
- Successor function
- Goal test
- Solution
- Heuristic function

Planning :

- State
- Successor function
- Goal test
- Solution
- Heuristic function

Inference

- State
- Successor function
- Goal test
- Solution
- Heuristic function

Next class

Start **Constraint Satisfaction Problems** (CSPs)

Textbook 4.1-4.3