CSC 380/480 – Foundations of Artificial Intelligence
Winter 2007
Assignment 1 – Solution to Selected Problems

1. News Filtering Agent:

Summary:

There is not a single correct answer to this problem. Specific answers depend partly on the assumptions we make about the agent.

a. **PEAS description**:

Assumed that agent is not responsible for any sort of troubleshooting or searching for alternate sources in the event there is a disruption in connection or news content. Assumed news site could be searched by following links as well as a form based search. Assumed news story content/links live until a known expiration time.

PEAS – Performance, Environment, Actuators, Sensors

Performance Measure:
- Maximize the number of matching unread stories returned
- Maximize number of matches per story returned
- Minimize impact to site, and other users of site
- Minimize time from when a new story is posted to when it gets returned

Environment:
- Internet connection
- Browsable shared internet news site updated as news happens
- Terminal to display results
- Static user profile of key words
- Database of news stories found and viewed by user, stories removed from database that are past the expiration date

Actuators:
- Ability to follow HTML links (Crawling; Issuing HTTP commands)
- Ability to issue queries in search page
- Ability to display found stories (HTML generation)
- Ability to add new stories to the database

Sensors
- Parse HTML pages for keywords
- Recognize pages returned as being either viewed or not viewed by user

b. Environment:

Partially observable – Agent is not able to observe when news stories are going to be posted. The agent's only option is to poll at intervals for changes.

Primarily Deterministic – Since the actions within a particular search are determined by the agents actions, although the appearance of news stories is stochastic since it alters the environment in ways not able to be determined through actions by the agent.

Sequential – At each episode the agent searches the environment and finds stories matching the user profile. Not episodic since the choices made in earlier steps, the number of downloaded pages and past user history may all affect the future actions taken by the agent.

Dynamic – Environment is dynamic for the agent since new stories can be posted while the agent is deliberating

Discrete – From the agent's perspective the environment is discrete since there are a limited number of types of situations/pages the agent would have to recognize and react to.

Single agent environment – Our agent is operating individually in the environment, the changing of the news site is treated as part of the environment rather than as another agent.

c. Agent Architecture:

A reflex agent with internal state will work; goal-based agent is better, since the agent can guide its actions based on how each action gets it closer to a goal node. In the case of a reflex agent, the agent blindly searches through the site and each page accessed is compared to the profile. A goal-based agent would use some additional criteria (e.g., heuristics) to decide if access to a page may be more likely to lead to an "interesting" page. For example, the agent may decide not to follow the branch in the site leading to the "Investor Relations" section of the site, since that is not likely to lead to news stories. Finally, if we want the ability to rank among the selected pages and filter out the ones that are less relevant, then a utility-based agent would be appropriate.

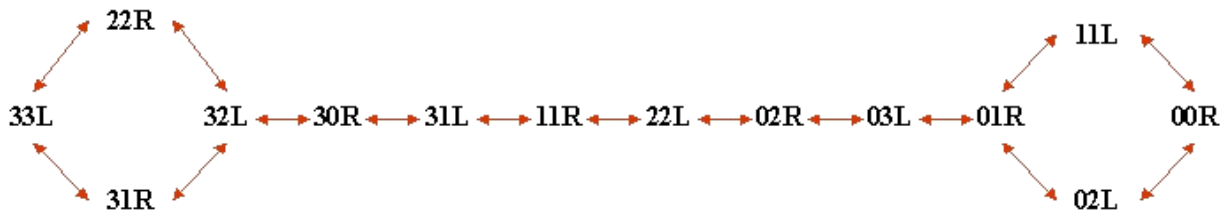
d. **Changes if the profile is obtained dynamically by observing past user behavior**:

The most important factor to consider here is that in this case, each of the actions taken by the user and the agent, may result in changing the profile. One implication of this is that the agent no longer has a static goal. Percepts would now have to include various user browsing actions (e.g., time of access to a page, frequency of visit, scrolling actions, etc.). Actions would have to include the ability to measure different metrics related to user actions, e.g., time spent on a page. The part of the environment relating to user profile will cause this to become a dynamic environment. Furthermore, the problem may become an exploration problem, if the agent needs to modify the criteria by which it determines that a user has found a page interesting during execution. The dynamic profile requirement may change the agent design to allow for adding a learning component to the agent which would focus on refining expiration date of stories, i.e. at what point is the user no longer interested in a story because it is "old" news regardless of how long it may remain on the news site. Also add learning component to see what combinations of words are most viewed to refine profile of user's interest and weighting of terms when ranking stories.

2. Missionaries and Cannibals Problem: Three Missionaries and three cannibals are on the left bank of a river. There is a boat on their side of the river that can be used to carry one or two people. The goal is to use this boat to cross the river in such a way that cannibals never outnumber missionaries on either bank of the river.

Solution:

We can represent each state in the problem as a triple **MCS**, where **M** is the number of missionaries on the left bank, **C** is the number of cannibals on the left bank, and **S** is the side the side the boat is on (L or R). We will also disallow moves that bring us back right to the state we start from. These are essentially useless moves. For example, from the state **33L** we can go to the state **32R** (i.e., a cannibal goes to the right bank alone). But, in this case, the only possible next move is for the cannibal to come right back. Given this representation of the problem, the state-space graph is as follows:

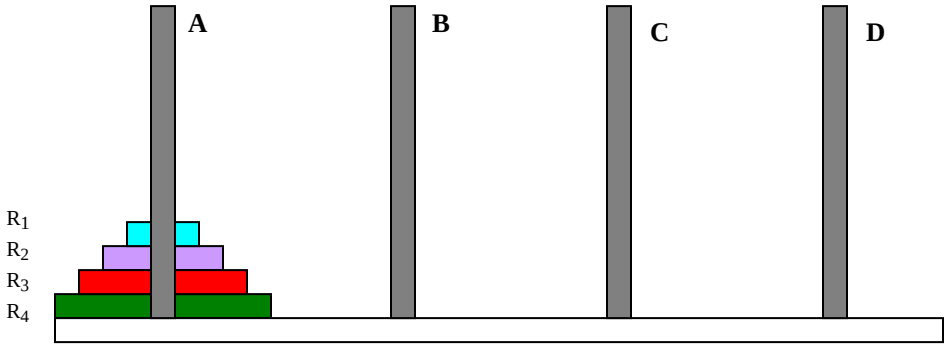


There are two possibilities from the start node (and M and a C can go, or two C's can go). Also, there are 2 possibilities from the second-to-last node (everyone is across but one cannibal, so either a missionary or a cannibal can go back for him). All other nodes have a branching factor of 1.

Despite its simplicity, this problem has some interesting features. First, this is a completely symmetric search space, so forward and backward searches can be done in an identical way to solve the problem. However, the problem "seems" much simpler working in the forward direction (starting from the intial state and working towards the goal state). Why is this? No one knows!

There is also a great deal of room for meta-level reasoning. Suppose that we had given names to each of the missionaries and cannibals. The apparent branching factor is now much higher. For example, Jimmy the cannibal going across is now not the same move as, say, Billy the cannibal going across. A person working on this problem will quickly realize that the names of the individuals is completely irrelevant to solving the problem, and will collapse the search space to the original one (meta-level reasoning). A computer, on the other hand, faced with the problem in which the individuals are named, may fail to recognize that the search space can be simplified.

3. Towers-of-Hanoi problem:



(a) **A state can be represented** by a quadruplet (*a*,*b*,*c*,*d*), where each element represents a peg (*a* for A, *b* for B, …) and is the list of ring indices (integers) on that peg from top to bottom. **The initial state** is ((R_1,R_2,…,R*n*),nil,nil,nil), where 'nil' denotes the empty list. The goal state is (nil,nil,nil, (R_1,R_2,…,R*n*)). Given any state (*a*,*b*,*c*,*d*), its **successors** are obtained as follows: For any pair (*x*, *y*) of distinct pegs such that $x \neq$ nil and, either $y =$ nil, or the first element of *x* is smaller than the first element of *y*, then remove the first element of *x* and insert it at the front of *y*.

(b) Each ring can be on any one of the 4 pegs, so there are 4^n states in total (i.e., 256 when n=4).

(c) There are at most 6 successors, each corresponding to an unordered pair of distinct pegs. Indeed, for any unordered pair (*x*, *y*) of distinct pegs, the motion of a ring can happen only in one direction, from *x* to *y*, or from *y* to *x*, and there are exactly 6 unordered pairs. Any state where no two pegs hold 0 ring has 6 successors.

4. *The Water-Jug problem*:

Solution:

We represent the state-space for this problem as

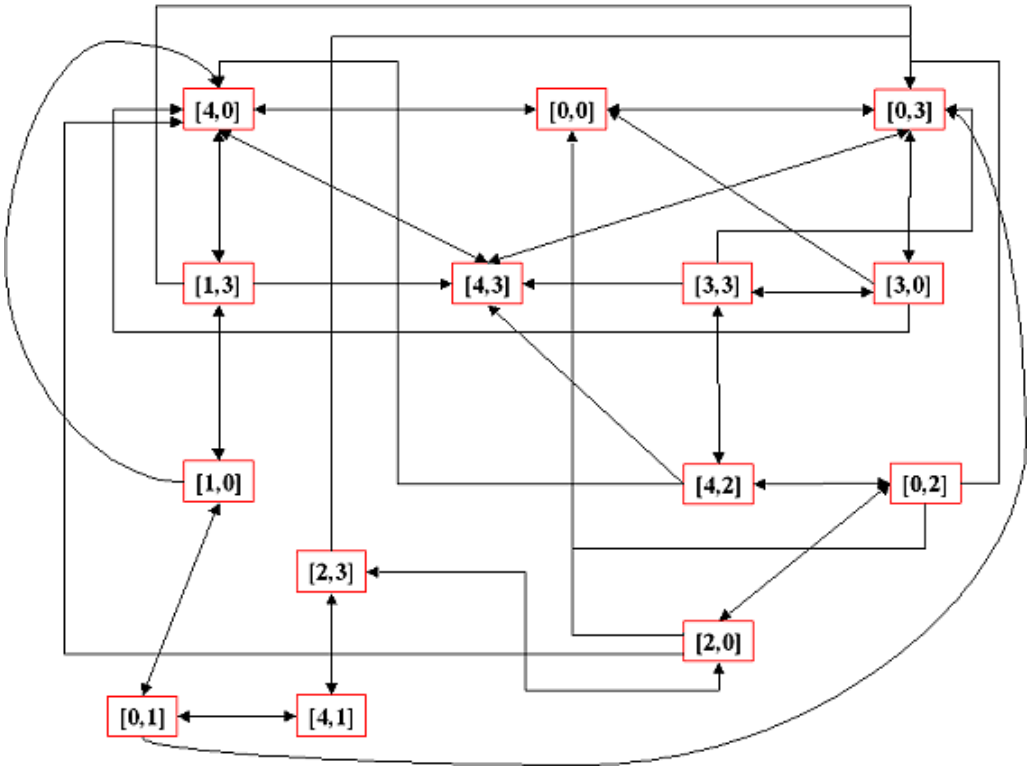
$$\{[x, y] \mid x = 0,1,2,3,4 \quad and \quad y = 0,1,2,3\}$$

Where ***x*** represents the gallons of water in the 4-gallon jug, and ***y*** represents the water in the 3-gallon jug.

The operators for the problem can, for example, be specified as the following list of contingency rules:

1. Fill 4-gallon jug: $([x, y] \mid x < 4) \rightarrow [4, y]$
2. Fill 3-gallon jug: $([x, y] \mid y < 3) \rightarrow [x, 3]$
3. Empty 4-gallon jug: $([x, y] \mid x > 0) \rightarrow [0, y]$
4. Empty 3-gallon jug: $([x, y] \mid y > 0) \rightarrow [x, 0]$
5. From 3-gallon jug to 4 gallon jug, until full: $([x, y] \mid x+y \geq 4 \text{ and } y > 0) \rightarrow [4, y-(4-x)]$
6. From 4-gallon jug to 3 gallon jug, until full: $([x, y] \mid x+y \geq 3 \text{ and } x > 0) \rightarrow [x-(3-y)], 3]$
7. All of 3-gallon jug into 4-gallon jug: $([x, y] \mid x+y \leq 4 \text{ and } y > 0) \rightarrow [x+y, 0]$
8. All of 4-gallon jug into 3-gallon jug: $([x, y] \mid x+y \leq 3 \text{ and } x > 0) \rightarrow [0, x+y]$

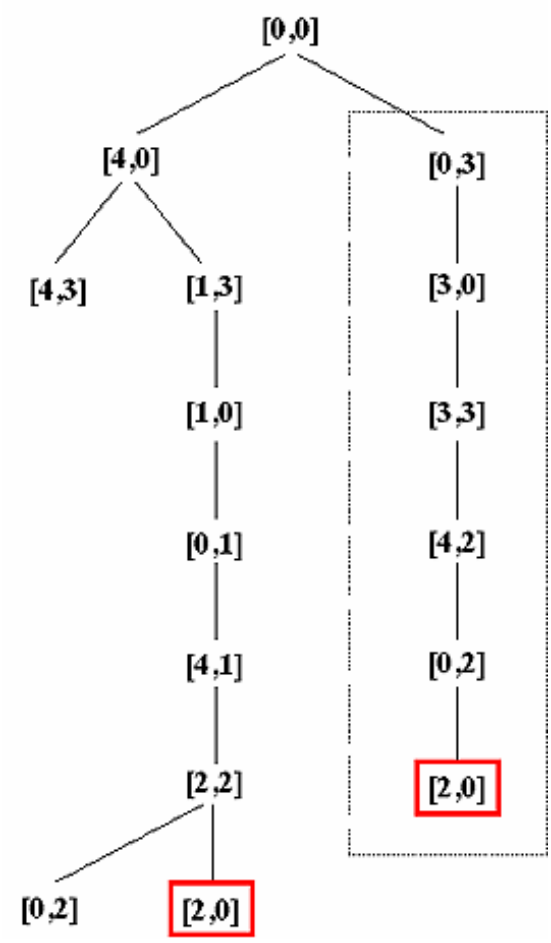
A ***portion*** of the state-space graph is shown In Figure below. The labels on the edges (i.e., operators applied) are omitted since they are clear from context. The part of the graph that is not shown include states [1,1], [1,2], [2,1], [2,2], [3,1], and [3,2]. These states only have "out" edges going to states shown below, but no "in" edges coming from any of the other states (so they can never be reached, unless they are start states).



Given the above rules, we can now solve the problem with the initial state [0, 0] and the goal state [2, 0]. For example, one possible solution is given below:

[0, 0]	➔ apply rule 2 ➔	[0, 3]
	➔ apply rule 7 ➔	[3, 0]
	➔ apply rule 2 ➔	[3, 3]
	➔ apply rule 5 ➔	[4, 2]
	➔ apply rule 3 ➔	[0, 2]
	➔ apply rule 7 ➔	[2, 0]

The portion of the complete search tree (ignoring repeated nodes) leading to two goal nodes is shown in Figure below. The right subtree rooted by the initial node gives the optimal path to a goal node. It is not easy to come up with a good (or in fact admissible) heuristic for this problem. For example, you might note that the simple heuristic of taking the sum of distances of each of the two components from the goal node, is not admissible and in general fails to find the optimal path. Since there are a small number of possible states and small branching factor, a good strategy might be to do breadth-first search.



Search tree for the specified instance of the Water Jug problem. There are two possible solutions (paths leading to a goal node. The path highlighted represent the optimal solution to the problem (assuming that the cost associated with any of the moves is the same for all of the operation (i.e., we have unit costs).