CS322 Fall 1999
Module 5 (Search Issues)
Assignment 5

Solution

The aim of this assignment is to learn more about search, both advanced search techniques as well as how to represent an abstract problem as a search problem.

Question 1

The graph *mod*5*gr*1999, available from the web site in both CILog format as well as for the graph-drawing applet, is meant to be part of the road network for a city. For this graph, the aim is find a path from node *mi* to the location *cp* that can only be reached by round-about methods.

(a) Which of the following methods will find a path from *mi* to *cp* without loop detection or multiple-path pruning: depth-first search, A* search, breadth-first search, best-first search.

(b) For A* search, how much saving (in the number of nodes expanded) is obtained by using loop detection and using Multiple-path pruning? (Give the number of nodes selected from the frontier with and without each of the two pruning methods).

(c) Is a backward search more efficient than a forward search for breadth-first search or A*? Explain why.

(d) How could a bi-directional search help? Explain. What forward and backward searches would be useful?

(e) Give the distance table created by dynamic programming to find a path from *mi* to *cp*.

Solution

(a) Which of the following methods will find a path from *mi* to *cp* without loop detection or multiple-path pruning: depth-first search, A* search, breadth-first search, best-first search.

A* and breadth-first search both do. Depth-first search and best-first search don't find a path.

(b) For A* search, how much saving (in the number of nodes expanded) is obtained by using loop detection and using Multiple-path pruning? (Give the number of nodes selected from the frontier with and without each of the two pruning methods).

Number of nodes selected to get an answer:

- No pruning: 37
- Loop detection: 14
- Multiple Path Pruning: 10

(c) Is a backward search more efficient than a forward search for breadth-first search or A*? Explain why.

Yes, backward search is more efficient, and there are fewer choices at each stage. For example, A* search has the following number of nodes selected for the backward search.

- No pruning: 12
- Loop detection: 7
- Multiple Path Pruning: 7

Backwards, breadth-first search finds a solution in 13 steps, whereas forward search takes 95 steps (node expansions).

(d) How could a bi-directional search help? Explain. What forward and backward searches would be useful?

The problem with this graph is that the forward search is so much worse than the backward search. A very limited forward search expands more nodes than we need to find the backward path. So a bi-directional search wouldn't help.

One could imagine a limited A* search from mi, that quickly determines that the path length is at least 24.9 (when hi is selected). Therefore the distance from mi to bg must be at least 24.9-6.4=18.5. We could use this updated heuristic information to not expand bg, but this doesn't really save us, particularly with multiple path pruning.

You can get full marks for this part by writing anything sensible.

(e) Give the distance table created by dynamic programming to find a path from *mi* to *cp*.

$dist(cp) = 0$
$dist(fv) = 8.5$
$dist(tw) = 21.7$
$dist(gb) = 31.3$
$dist(mv) = 34.8$
$dist(bg) = 36.0$
$dist(mi) = 53.1$
$dist(hi) = 62.6$
$dist(fg) = 63.8$
$dist(uv) = 71.4$
$dist(rs) = 92.7$
$dist(ab) = 101.5$
$dist(re) = 113.6$

Question 2

Publish-on demand for textbooks and online courses is becoming more commonplace. We want to be able to deliver custom versions of cs322 for use at other places who may only want to use a subset of the modules, perhaps in different orders. Here we consider the problem of delivering a course to suit the goals of an instructor as a search problem.

Suppose *mod*(*Mod*, *Prereqs*, *Teaches*) is true if module *Mod* covers the elements of the list *Teaches* and requires that the students have already covered the elements of the list *Prereqs*.

```
mod(m1,[],[intro]).
mod(m2,[intro],[semantics,symbols]).
mod(m3,[symbols,semantics],[proofs]).
mod(m4,[intro],[search]).
mod(m5,[search,symbols],[advanced_search]).
mod(m6,[search,semantics],[csp]).
mod(m7,[symbols,semantics],[kr]).
```

```
mod(m8,[proofs],[metainterpreters]).
mod(m9,[semantics],[actions]).
mod(m10,[actions,metainterpreters,search],[planning]).
mod(m11,[search,metainterpreters],[dtlearning]).
mod(m12,[csp],[nnlearning]).
```

Suppose we decide to represent the problem of designing custom courses as a search problem where

- the nodes are lists of topics that have been covered and
- the arcs are labelled with modules. Suppose L is a node, and M is a module such that $mod(M, P, T)$, where P is a subset of L (every element of P is in L), and T is not a subset of L, then $L \cup T$ is a neighbour of L, with the arc labelled with M.

The start node is labelled with the empty list []. A goal node is a node that includes all of the topics the instructor wants to cover.

For example, suppose an instructor wants to cover *nnlearning*, and *proofs*, then any node that contains both *nnlearning* and *proofs* is a goal node. Then a solution is the path that starts with [], then has arc labelled with $m1$ to node [*intro*], then has arc $m4$ to node[1] [*intro*, *search*], then has arc $m2$ to node [*intro*, *search*, *semantics*, *symbols*], then has arc $m6$ to node [*csp*, *intro*, *search*, *semantics*, *symbols*], then has arc $m12$ to [*csp*, *intro*, *nnlearning*, *search*, *semantics*, *symbols*], then has arc $m3$ to node [*csp*, *intro*, *nnlearning*, *proofs*, *search*, *semantics*, *symbols*], which is a goal node.

(a) Draw the search graph to depth four. This should include all paths from the start node that contain three or fewer arcs.

(b) Is loop checking useful? Explain.

(c) Is multiple path pruning useful? Explain.

(d) Is backward search better than forward search for this problem? Explain.

(e) Suppose we want to use A^* search. Give a non-trivial heuristic function that is an underestimate of the actual distance from a node to a goal.

Solution

(a) Draw the search graph to depth four. This should include all paths from the start node that contain three or fewer arcs.

The graph is shown in Fugure 1.

(b) Is loop checking useful? Explain.

No. There are no cycles in this graph. This is because you can't do a module if you have already covered all of the topics. And you never forget topics.

(c) Is multiple path pruning useful? Explain.

Yes. There are many multiple paths. In particular, reorderings of modules produce the save nodes.

(d) Is backward search better than forward search for this problem? Explain.

No, not with the nodes as defined here. This is because there are too many goal nodes. For example, if the goal were to cover *nnlearning* and *proofs*, every set of topics that includes these two would be a goal node (there are $2^11 = 2056$ of these). This would mean that you have too many start nodes in the forward search.

[1] Note that here we are representing sets of nodes as lists sorted alphabetically.

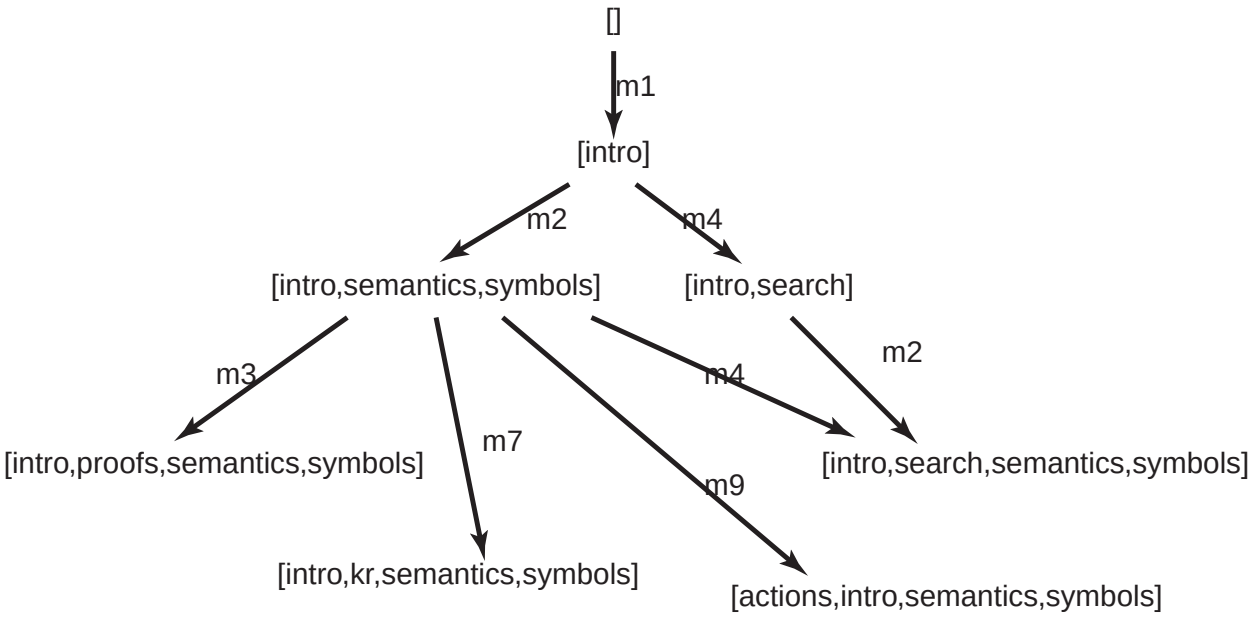

Figure 1: Solution to question two, part (a).

(e) Suppose we want to use A^* search. Give a non-trivial heuristic function that is an underestimate of the actual distance from a node to a goal.

We assume here that we are trying to minimise the number of topics taught.

The simplest is $h(n)$ is the number of topics in g that are not in n. This isn't quite right, as one module (module $m2$) covers two topics. So we should divide this number by two or somehow make sure we don't count both semantics and symbols. This would then be an underestimate of the number of topics that needs to be taught.