EEL5840: Elements of Machine Intelligence

Announcements

- Reading Assignment:
  > Nilsson chapters 10
- Announcements:
  > None
- Today’s Handouts in WWW:
  > Tentative 2nd Exam Date:
    > 12/03/15 (Thursday)
  > LISP Project due 12/01/15
  > Outline Class 22
  > www.mil.ufl.edu/eel5840
  > Software and Notes

Today’s Menu

- Sample A* Problem
- A* Code
- Traveling Salesman - Brute Force
  {See “Traveling_Salesman.ppt”}
Heuristic Searches

The following figure shows a search tree with the state indicated by the tuple inside parentheses. A letter indicates the state name and the integer indicates the estimated cost for finding a solution from that state (a cost of 0 indicates a goal state). Using the Graph-Search algorithm discussed in class, give the solution tree or steps using depth-first search. How many nodes did depth-first expand? Repeat using breadth-first search. How many nodes did breadth-first expand? Repeat using heuristic search. How many nodes did heuristic search expand? Repeat using A* search. How many nodes did A* expand? Can any of these algorithms ever find N as a solution? Explain.

```
PROCEDURE GRAPH-SEARCH
1. Create a search graph, G, consisting solely of the start node, s. Put s on a list called OPEN.
2. Create a list called CLOSED that is initially empty.
3. LOOP: if OPEN is empty, exit with failure.
4. Select the first node on OPEN, remove it from OPEN, and put it on CLOSED. Call this node n.
5. If n is a goal node, exit successfully with the solution obtained by tracing a path along the pointers from n to s in G. (see step 7.)
6. Expand node n, generating the set, M, of its successors and install them as successors of n in G.
7. Establish a pointer to n from those members of M that were not already in G (i.e., not already on either OPEN or CLOSED). Add these members of M to OPEN. For each member of M that was already on OPEN or CLOSED, decide whether or not to redirect its pointer to n. For each member of M already on CLOSED, decide for each of its descendants in G whether or not to redirect its pointer.
8. Reorder the list OPEN, either according to some arbitrary scheme or according to heuristic merit.
9. Go LOOP
```
Algorithm Details: You can use algorithm graphsearch for everything
Start: Open={A} Closed={} M={} f(n)=g(n)+h(n) where g(n)=depth(n) & h(n)=heuristic fcn

Breadth First: Use the function f(n)=depth(n) and append M at the end of the open list.
1. The algorithm selects A and expands A (applies Γ) in order to obtain M={B,C}
   n_1=B; n_2=C; Open={B,C}, Closed={A}, G={A,B,C}, f(n_1)=1; f(n_2)=1
2. The algorithm expands B in order to obtain M={D,E}
   n_3=D; n_4=E; Open={D,E}, Closed={A,B}, G={A,B,C,D,E}, f(n_3)=1; f(n_4)=1
3. The algorithm expands C in order to obtain M={FG}, f(n_5)=1; f(n_6)=1
   n_5=F; n_6=G; Open={D,E,F,G}, Closed={A,B,C}, G={A,B,C,D,E,F,G}
4. The algorithm expands D in order to obtain M={H,I}, f(n_7)=1; f(n_8)=1
   n_7=H; n_8=I; Open={E,F,G,H,I}, Closed={A,B,C,D}, G={A,B,C,D,E,F,G,H,I},
5. The algorithm expands E in order to obtain M={J,K,L,M}, f(n_9)=1; f(n_10)=1
6. The algorithm expands F in order to obtain M={LM}, f(n_11)=1; f(n_12)=1
7. The algorithm expands G in order to obtain M={NP}, f(n_13)=1; f(n_14)=1
8. The algorithm expands H in order to obtain M={}, G={A,B,C,D,E,F,G,H,I,J,K,L,M,N,P}
10. The algorithm selects J. J is a goal node and the algorithm terminates.
BFS expands Closed={A,B,C,D,E,F,G,H,I,J,K,L,M,N,P} 9 nodes

Depth-First: Use the function f(n)=depth(n) and append M at the front of the open list.
1. The algorithm selects A and expands A (applies Γ) in order to obtain M={B,C}
   n_1=B; n_2=C; Open={B,C}, Closed={A}, G={A,B,C}, f(n_1)=1; f(n_2)=1
2. The algorithm expands B in order to obtain M={D,E}
   n_3=D; n_4=E; Open={D,E}, Closed={A,B}, G={A,B,C,D,E}, f(n_3)=2; f(n_4)=2
3. The algorithm expands D in order to obtain M={H,I,J,K,L,M}, f(n_5)=2; f(n_6)=2
4. The algorithm expands E in order to obtain M={}
5. The algorithm expands F in order to obtain M={}
6. The algorithm selects J. J is a goal node and the algorithm terminates.
Heuristic-Search: Use the function \( f(n) = h(n) \) and sort the open list using \( f \) values.

1. The algorithm selects A and expands A (applies \( \Gamma \)) in order to obtain \( M=\{B,C\} \)
   \( n_1=B; n_2=C; \) Open=\( \{B,C\} \), Closed=\( \{A\} \), \( G=\{A,B,C\} \), \( f(n_1)=26; f(n_2)=13 \)

2. The algorithm expands C in order to obtain \( M=\{F,G\} \)
   \( n_3=F; n_4=G; \) Open=\( \{F,B,G\} \), Closed=\( \{A,B,C,F,G\} \), \( f(n_3)=12; f(n_4)=27 \)

3. The algorithm expands F in order to obtain \( M=\{L,M\} \)
   \( n_5=L; n_6=M \), Open=\( \{B,G,M,L\} \), Closed=\( \{A,B,C,F,G,L,M\} \), \( f(n_5)=33; f(n_6)=29 \)

4. The algorithm expands B in order to obtain \( M=\{D,E\} \)
   \( n_7=D; n_8=E \), Open=\( \{E,D,G,M,L\} \), Closed=\( \{A,B,C,F,G,L,M,D,E\} \)

5. The algorithm expands E in order to obtain \( M=\{J,K\} \)
   \( n_9=J; n_{10}=K \), Open=\( \{J,K,D,G,M,L\} \), Closed=\( \{A,C,F,B,E\} \), \( G=\{A,B,C,F,G,L,M,D,E,J,K\} \)

6. The algorithm selects J. J is a goal node and the algorithm terminates.

\( A^* \) search expands Closed=\( \{A,C,F,B,E\} \) 5 nodes.

\[ f(n)=g(n)+h(n) \]

1. The algorithm selects A and expands A (applies \( \Gamma \)) in order to obtain \( M=\{B,C\} \)
   \( n_1=B; n_2=C \), Open=\( \{B,C\} \), Closed=\( \{A\} \), \( G=\{A,B,C\} \), \( f(n_1)=1+26; f(n_2)=1+13 \)

2. The algorithm expands C in order to obtain \( M=\{F,G\} \)
   \( n_3=F; n_4=G \), Open=\( \{F,B,G\} \), Closed=\( \{A,B,C,F,G\} \), \( f(n_3)=2+12; f(n_4)=2+27 \)

3. The algorithm expands F in order to obtain \( M=\{L,M\} \)
   \( n_5=L; n_6=M \), Open=\( \{B,G,M,L\} \), Closed=\( \{A,B,C,F,G,L,M\} \), \( f(n_5)=3+33; f(n_6)=3+29 \)

4. The algorithm expands B in order to obtain \( M=\{D,E\} \)
   \( n_7=D; n_8=E \), Open=\( \{E,D,G,M,L\} \), Closed=\( \{A,B,C,F,G,L,M,D,E\} \), \( f(n_7)=2+19; f(n_8)=2+16 \)

5. The algorithm expands E in order to obtain \( M=\{J,K\} \)
   \( n_9=J; n_{10}=K \), Open=\( \{J,K,D,G,M,L\} \), Closed=\( \{A,C,F,B,E\} \), \( f(n_9)=3+0; f(n_{10})=3+2 \)

6. The algorithm selects J. J is a goal node and the algorithm terminates.

\( A^* \) search expands Closed=\( \{A,C,F,B,E\} \) 5 nodes. \( G=\{A,B,C,F,G,L,M,D,E,J,K\} \)

\( N \) will not be found by any of the algorithms because path \( \{A,B,E,J\} \) is considered before \( \{A,C,G,N\} \) due to the fact that \( h(E)=16 \) and \( h(G)=27 \) and \( h(E)<h(G) \).
EEL5840: Elements of Machine Intelligence

PROCEDURE GRAPH-SEARCH

1. Create a search graph, $G$, consisting solely of the start node, $s$. Put $s$ on a list called OPEN.
2. Create a list called CLOSED that is initially empty.
3. LOOP: if OPEN is empty, exit with failure.
4. Select the first node on OPEN, remove it from OPEN and put it on CLOSED. Call this node $n$.
5. If $n$ is a goal node, exit successfully with the solution obtained by tracing a path along the pointers from $n$ to $s$ in $G$. (see step 7.)
6. Expand node $n$, generating the set, $M$, of its successors and install them as successors of $n$ in $G$.
7. Establish a pointer to $n$ from those members of $M$ that were not already in $G$ (i.e., not already on either OPEN or CLOSED). Add these members of $M$ to OPEN. For each member of $M$ that was already on OPEN or CLOSED, decide whether or not to redirect its pointer to $n$. For each member of $M$ already on CLOSED, decide for each of its descendants in $G$ whether or not to redirect its pointer.
8. Reorder the list OPEN, either according to some arbitrary scheme or according to heuristic merit.
9. Go LOOP.

(defun astar (st) (prog ((OPEN (list st)) M n CLOSED (G (list st)) (e 0))
/* 1-2 */  (print (list 'OPEN '= OPEN 'CLOSED '= CLOSED))
LOOP /* 3-5 */  (cond
( (null OPEN) (return nil) )
( (equal 0 (get (setf n (car OPEN)) 'hval)) (return (list
"Expanded" e "nodes" "Solution:" (path (car OPEN)) 'g G 'closed CLOSED )))
/* 6 */     (setf M (exp n) e (+ 1 e) G (append G M) CLOSED (cons n CLOSED))
/* 7 */     (dolist (sex M) (putprop sex n 'father))
/* 8 */     (setf OPEN (sort (append M (cdr OPEN))))
/* 9 */      (go LOOP) ))
(defun path (end) (prog ((pth (list end)) (i end) j)
LOOP
(setf j (get i 'father))
(if j (setf pth (cons j pth)) (return (list 'path pth)))
(setf i j)
(go LOOP) ))
(defun exp (node) (get node 'sons) )
(defun fval (sex) (+ (get sex 'gval) (get sex 'hval))
> (load "astarclass25.lsp")
; loading astarclass25.lsp
T
> (load "astar.lsp")
; loading astar.lsp
T
> (astar 'a)
("Expanded" 5 "nodes" "Solution:" (PATH (A B E J))
G (A B C F G L M D E J K) CLOSED (E B F C A))

The End!