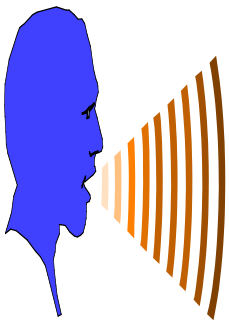
SubjuGator

EEL5840: Elements of Machine Intelligence

Announcements



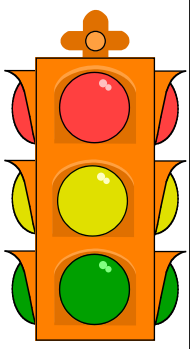
- Reading Assignment:
  - > Nilsson chapters 10
- Announcements:
  - > None
- Today's Handouts in WWW:
  - > Tentative 2nd Exam Date:
    - – 12/1/11 (Thursday)
  - > LISP Project due 11/29/11
  - > Outline Class 24
  - > www.mil.ufl.edu/eel5840
  - > Software and Notes

SubjuGator

EEL5840: Elements of Machine Intelligence

Today's Menu

- Sample A* Problem
- Sample A* Code
- Traveling Salesman - Brute Force
  {See "Traveling_Salesman.ppt"}

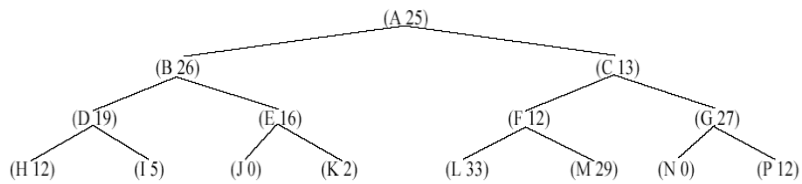


SubjuGator

EEL5840: Elements of Machine Intelligence

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



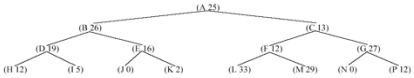
SubjuGator

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

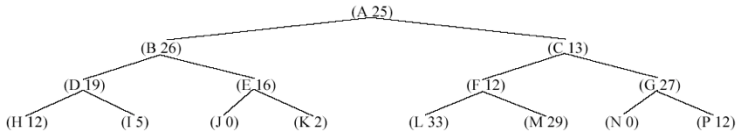
reassign parent pointers



Algorithm Details: You can use algorithm graphsearch for everything
Start: Open={A} Closed={} G={} M={} $f(n)=g(n)+h(n)$ where $g(n)=depth(n)$ & $h(n)$=heuristic fcn

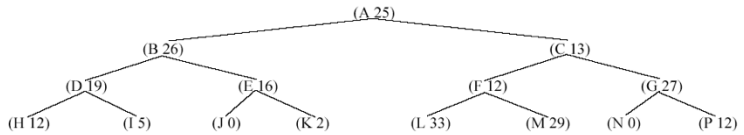
Breadth First: Use the function $f(n)$=car(open) and append M at the end of the open list.

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={C,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={F,G}, $f(n_5)=1$; $f(n_6)=1$
   n_5=F; n_6=G; Open={D,E,F,G}, Closed={A,B,C}, G={A,B,C,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} $f(n_9)=1$; $f(n_{10})=1$
   n_9=J; n_{10}=K; Open={F,G,H,I,J,K}, Closed={A,B,C,D,E}, G={A,B,C,D,E,F,G,H,I,J,K},
6. The algorithm expands F in order to obtain M={L,M}, $f(n_{11})=1$; $f(n_{12})=1$, n_{11}=L; n_{12}=M;
   Open={G,H,I,J,K,L,M}, Closed={A,B,C,D,E,F}, G={A,B,C,D,E,F,G,H,I,J,K,L,M}
7. The algorithm expands G in order to obtain M={N,P}, $f(n_{13})=1$; $f(n_{14})=1$, n_{13}=N; n_{14}=P;
   Open={H,I,J,K,L,M,N,P}, G={A,B,C,D,E,F,G,H,I,J,K,L,M,N,P}, Closed={A,B,C,D,E,F,G}
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}
   Open={I,J,K,L,M,N,P}, Closed={A,B,C,D,E,F,G,H}
9. The algorithm expands I in order to obtain M={}, G={A,B,C,D,E,F,G,H,I,J,K,L,M,N,P}
   Open={J,K,L,M,N,P}, Closed={A,B,C,D,E,F,G,H,I}
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} 9 nodes



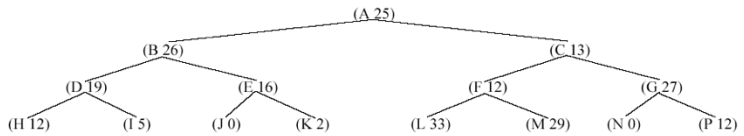
Depth-First: Use the function $f(n)$=depth(n) and append M at the front of the open list.

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,C}, 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}
   n_4=H; n_5=I; Open={H,I,E,C}, Closed={A,B,D}, G={A,B,C,D,E,H,I}, $f(n_5)=3$; $f(n_6)=3$
4. The algorithm expands H in order to obtain M={}
   Open={I,E,C}, Closed={A,B,D,H}, G={A,B,C,D,E,H,I}
5. The algorithm expands I in order to obtain M={}, G={A,B,C,D,E,H,I}
   Open={E,C}, Closed={A,B,D,H,I}
6. The algorithm expands E in order to obtain M={J,K}, $f(n_9)=3$; $f(n_{10})=3$
   n_6=J; n_7=K; Open={J,K,C}, Closed={A,B,D,H,I,E}, G={A,B,C,D,E,H,I,J,K},
7. The algorithm selects J. J is a goal node and the algorithm terminates.
   G={A,B,C,D,E,H,I,J,K}, DFS expands Closed={A,B,D,H,I,E,} 6 nodes



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 Γ) in order to obtain M={B,C}
   n_1=B; n_2=C; Open={B,C}, Closed={A}, G={A,B,C}, $f(n_1)$=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,C}, G={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,C,F}, G={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}, $f(n_7)$=19; $f(n_8)$=16
   n_7=D; n_8=E; Open={E,D,G,M,L}, Closed={A,C,F.B}, G={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; $f(n_9)$=0; $f(n_{10})$=2
   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. Open={K,D,G,M,L},
   G={A,B,C,F,G,L,M,D,E,J,K} Heuristic search expanded Closed={A,C,F,B,E} 5 nodes



A* Search: Uses $f(n)=g(n)+h(n)$ where $g(n)=depth(n)$ & $h(n)$=cost and sort the open list using f

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+26; $f(n_2)$=1+13
2. The algorithm expands C in order to obtain M={F,G}, $f(n_3)$=2+12; $f(n_4)$=2+27
   n_3=F; n_4=G; Open={F,B,G}, Closed={A,C}, G={A,B,C,F,G}
3. The algorithm expands F in order to obtain M={L,M}, G={A,B,C,F,G,L,M}
   n_5=L; n_6=M; Open={B,G,M,L}, Closed={A,C,F}, $f(n_5)$=3+33; $f(n_6)$=3+29
4. The algorithm expands B in order to obtain M={D,E}, G={A,B,C,F,G,L,M,D,E}
   n_7=D; n_8=E; Open={E,D,G,M,L}, Closed={A,C,F.B}, $f(n_7)$=2+19; $f(n_8)$=2+16
5. The algorithm expands E in order to obtain M={J,K}, G={A,B,C,F,G,L,M,D,E,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

SubjuGator

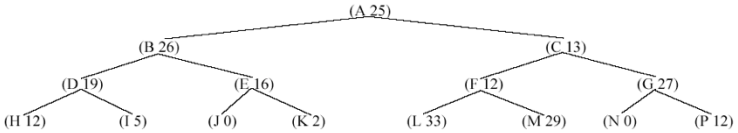
PROCEDURE GRAPH-SEARCH

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 "astarclass24.lsp")
; loading astarclass24.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))
```


EEL5840: Elements of Machine Intelligence

SubjuGator




```
(putprop 'a '(b c) 'sons )
(putprop 'b '(d e) 'sons )
(putprop 'c '(f g) 'sons )
(putprop 'd '(h i) 'sons )
(putprop 'e '(j k) 'sons )
(putprop 'f '(l m) 'sons )
(putprop 'g '(n p) 'sons )
```

```
(putprop 'a 25 'hval )
(putprop 'b 26 'hval )
(putprop 'c 13 'hval )
(putprop 'd 19 'hval )
(putprop 'e 16 'hval )
(putprop 'f 12 'hval )
(putprop 'g 27 'hval )
(putprop 'h 12 'hval )
(putprop 'i 5 'hval )
(putprop 'j 0 'hval )
(putprop 'k 2 'hval )
(putprop 'l 33 'hval )
(putprop 'm 29 'hval )
(putprop 'n 0 'hval )
(putprop 'p 12 'hval )
```

```
(putprop 'a 0 'gval )
(putprop 'b 1 'gval )
(putprop 'c 1 'gval )
(putprop 'd 2 'gval )
(putprop 'e 2 'gval )
(putprop 'f 2 'gval )
(putprop 'g 2 'gval )
(putprop 'h 3 'gval )
(putprop 'i 3 'gval )
(putprop 'j 3 'gval )
(putprop 'k 3 'gval )
(putprop 'l 3 'gval )
(putprop 'm 3 'gval )
(putprop 'n 3 'gval )
(putprop 'p 3 'gval )
```

SubjuGator

EEL5840: Elements of Machine Intelligence

The End!

© Dr. A. Antonio Arroyo