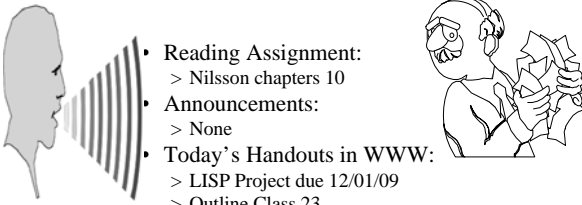


EEL5840: Elements of Machine Intelligence

SubjuGator

Announcements



- Reading Assignment:
 - > Nilsson chapters 10
- Announcements:
 - > None
- Today's Handouts in WWW:
 - > LISP Project due 12/01/09
 - > Outline Class 23
 - > www.mil.ufl.edu/eel5840
 - > Software and Notes

University of Florida
EEL 5840 - Class #23 - Fall 2009
© Dr. A. Aravamudan

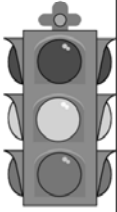
1

EEL5840: Elements of Machine Intelligence

SubjuGator

Today's Menu

- Sample A* Problem
- Sample A* Code
- IDA & IDA*
- Traveling Salesman - Brute Force {Questions?}



University of Florida
EEL 5840 - Class #23 - Fall 2009
© Dr. A. Aravamudan

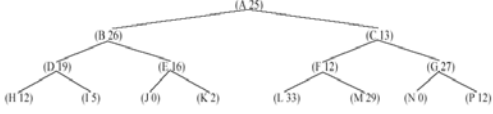
2

EEL5840: Elements of Machine Intelligence

SubjuGator

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



University of Florida
EEL 5840 - Class #23 - Fall 2009
© Dr. A. Aravamudan

3

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 *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

University of Florida
EEL 5840 - Class #23 - Fall 2009
© Dr. A. Aravamudan

4

Algorithm Details: You can use algorithm graphsearch for everything
 Start: Open={A} Closed={} G={ } M={ } $f(n)=g(n)+h(n)$ where $g(n)=depth(n)$ & $h(n)=heuristic\ fc$
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 $M=\{F,G\}; f(n_5)=1; f(n_6)=1$
 $n_5=F; n_6=G; Open=\{D,E,F,G\}; Closed=\{A,B,C\}; G=\{A,B,C,D,E,F,G\}$
 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 $M=\{ \}; G=\{A,B,C,D,E,F,G,H,I\}$
 $Open=\{J,K,L,M,N,P\}; Closed=\{A,B,C,D,E,F,G,H,I\}$
 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 $M=\{H,I\}$
 $n_5=H; n_6=I; Open=\{H,I,E,C\}; Closed=\{A,B,D\}; G=\{A,B,C,D,E,H,I\}; f(n_5)=3; f(n_6)=3$
 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 $M=\{J,K\}; f(n_9)=3; f(n_{10})=3$
 $n_9=J; n_{10}=K; Open=\{J,K,C\}; Closed=\{A,B,D,H,I,E\}; G=\{A,B,C,D,E,H,I,J,K\}$
 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 $M=\{F,G\}$
 $n_3=F; n_4=G; Open=\{B,C,F,G\}; Closed=\{A,C\}; G=\{A,B,C,F,G\}; f(n_3)=12; f(n_4)=27$
 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 $M=\{F,G\}; f(n_3)=2+12; f(n_4)=2+27$
 $n_3=F; n_4=G; Open=\{B,C,F,G\}; Closed=\{A,C\}; G=\{A,B,C,F,G\}$
 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

(putprop 'a '(b c) 'sons)	(putprop 'a 25 'hval)	(putprop 'n 0 'gval)
(putprop 'b '(d e) 'sons)	(putprop 'b 26 'hval)	(putprop 'b 1 'gval)
(putprop 'c '(f g) 'sons)	(putprop 'c 13 'hval)	(putprop 'c 1 'gval)
(putprop 'd '(h i) 'sons)	(putprop 'd 19 'hval)	(putprop 'd 2 'gval)
(putprop 'e '(j k) 'sons)	(putprop 'e 16 'hval)	(putprop 'e 2 'gval)
(putprop 'f '(l m) 'sons)	(putprop 'f 12 'hval)	(putprop 'f 2 'gval)
(putprop 'g '(n p) 'sons)	(putprop 'g 27 'hval)	(putprop 'g 2 'gval)
	(putprop 'h 12 'hval)	(putprop 'h 3 'gval)
	(putprop 'i 12 'hval)	(putprop 'i 3 'gval)
	(putprop 'j 0 'hval)	(putprop 'j 3 'gval)
	(putprop 'k 2 'hval)	(putprop 'k 3 'gval)
	(putprop 'l 33 'hval)	(putprop 'l 3 'gval)
	(putprop 'm 29 'hval)	(putprop 'm 3 'gval)
	(putprop 'n 0 'hval)	(putprop 'n 3 'gval)
	(putprop 'o 12 'hval)	(putprop 'o 3 'gval)

University of Florida
EEL 5840 - Class #12 - Fall 2009
© Dr. A. Amato, Author

9

EEL5840: Elements of Machine Intelligence

SubjuGator

Heuristic Search

- Iterative Deepening
 - Enjoys the same linear memory requirements of DFS while guaranteeing that a goal node of minimal depth will be found- memory grows linearly with the depth of the goal.
 - Successive depth-first searches are conducted-each with a depth bound increasing by 1-until a goal node is found.

Depth bound = 1 Depth bound = 2 Depth bound = 3 Depth bound = 4

University of Florida
EEL 5840 - Class #12 - Fall 2009
© Dr. A. Amato, Author

10

EEL5840: Elements of Machine Intelligence

SubjuGator

Heuristic Search

- How many nodes are expanded by BFS?
Assuming uniform branching factor b , with a goal at depth d
 $1 + b + b^2 + \dots + b^d = (b^{d+1} - 1)/(b - 1)$
- How many nodes are expanded by iterative deepening?
Down to level j we have $N_{id} = (b^{j+1} - 1)/(b - 1)$
At worst, we must conduct d such searches for the goal at depth d
 $N_{id} = \sum_{j=0}^d (b^{j+1} - 1)/(b - 1) = (b^{d+2} - 2b - bd + d + 1)/(b - 1)^2$
for large d this reduces to
 $N_{id}/N_{bf} \approx b/(b - 1)$
For $b=10$ and large d this is about 1.11% or ID expands about 11% more nodes than BFS

University of Florida
EEL 5840 - Class #12 - Fall 2009
© Dr. A. Amato, Author

11

EEL5840: Elements of Machine Intelligence


SubjuGator

Heuristic Search

- IDA* (Korf, 1985)
 - > Allows us to find minimal cost paths with memory that grows linearly with the depth of the goal.
 - > Execute a series of depth-first searches. In the first search we establish a cost cutoff equal to $f(n_0) = g(n_0) + h(n_0) = h(n_0)$ with $n_0 = s$.
 - > Expand nodes in a DFS fashion—backtrack whenever the f value of a successor of an expanded node exceeds the cutoff value.
 - > If this terminates at a goal node, then you have a minimal path, else the cost of an optimal path must exceed the cutoff value.
 - > Use as your next cutoff value the value of a node visited but not expanded
 - > IDA* does have to repeat node expansions, but there are potential tradeoffs involving reduced memory requirements and ease of implementation

University of Florida
EEL 5840 - Class #12 - Fall 2009
© Dr. A. Amato, Author

12



EEL5840: Elements of Machine Intelligence

The End!

University of Florida
EEL 5840 - Class #2 - Fall 2009
© Dr. A. Aravamudan

13