EEL5840: Elements of Machine Intelligence

Announcements

- Reading Assignment:
  > Nilsson chapter 9
- Announcements:
  > Tentative 2nd Exam Date:
    - 12/03/15 (Thursday)
  > LISP Project due 12/01/15
- Today’s Handouts in WWW:
  > Outline Class 20
- Web Site
  > www.mil.ufl.edu/eel5840
  > Software and Notes

EEL5840: Elements of Machine Intelligence

Today’s Menu

- Finish Class 19 Slides 23-31
- Heuristic Search (Chapter 9)
  \[ \Rightarrow \text{Algorithm A*} \]
  \[ \Rightarrow \text{Admissibility of A*} \]
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 A
Let $f(n) = g(n) + h(n)$ in step 8 of GRAPH-SEARCH where
$g(n):$ estimate of the cost of a minimum length path $s \rightarrow n$
$h(n):$ estimate of the cost of a minimum length path $n \rightarrow t$
also, step 7 guarantees that $g(n)$ can never increase.
{See slides 24-25 in class 19, we only keep the minimum path to the start node, path $n \rightarrow s$}

In the example:
$c(n_1,n_2) =$ cost from $n_1 \rightarrow n_2$
$g(n) =$ $c(n_1,n) = c(n_1,n_2) + c(n_2,n) = c_1 + c_2$
$h^*(n) =$ cost from $n \rightarrow t$ (the actual but unknown cost from $n \rightarrow t$)
h(n) is an estimate of $h^*(n)$. When we are at node $n$ we have not finished the problem and we do not yet have the real $c(n,t)$. We say that $f(n)$ is the cost of a minimal cost path constrained through node $n$.
Here, $c_6 < c_3$ and $c_3 \leq c_6 + c_7$ by the triangle inequality.

DEFINITIONS
- $f(n):$ estimate of the cost of a path $s \rightarrow t$ through node $n$
  $c(s,t)=c(s,n)+c(n,t)$
- $k(n_i,n_j):$ actual cost of a minimal-cost path $n_i \rightarrow n_j$
- $h^*(n):$ cost of a minimal cost path from node $n$ to a goal, i.e.,
  $\min(k(n,t_1), k(n,t_2), \ldots, k(n,t_i))$ and any node that achieves $h^*(n)$ is a node in the optimal path.
- $g^*(n):$ cost of an optimal path from $s \rightarrow n$ (the shortest path $s \rightarrow n$)
  $= k(s,n) \forall n$ accessible from $s$
- $f^*(n):$ (actual cost of an optimal path $s \rightarrow n$) + (cost of an optimal path $n \rightarrow t$) {the path $s \rightarrow t$ is constrained to go through $n$}
- $f(n) = g(n) + h(n) \forall n$ and $f^*(n) = g^*(n) + h^*(n) \forall n$

Now when $n=s$;
- $f^*(s) = g^*(s) + h^*(s) = h^*(s)$ since by definition $g^*(s)=0=k(s,s)$
- $h^*(s):$ actual cost of an unconstrained path $s \rightarrow t$
EEL5840: Elements of Machine Intelligence

Algorithm A

Let $g(n)$ be an estimate of $g^*(n)$ and $h(n)$ be an estimate of $h^*(n)$. Then $f(n)$ will be an estimate of $f^*(n)$ and further let

\[ E \{f(n) - f^*(n)\} = 0 \quad \text{and} \quad \text{Variance}\{f(n)\} \text{ be non-increasing} \]

We say that $f(n)$ is a consistent estimate of $f^*(n)$ and we expect algorithm A to yield good results.

Q: Are there any properties of $f$, $g$, or $h$ that insure optimal results?

We also note:

- If $h(n)=0$ and $g(n)=\text{depth}(n)$ algorithm A yields BFS.
- If $g(n)=0$ algorithm A yields pure heuristic search (DFS).
- $f(n) = \alpha g(n) + \beta h(n) = \alpha \{\text{breadth component}\} + \beta \{\text{depth component}\}$
- And $\{\alpha, \beta\}=1$ in Algorithm A but it can be more general.

Algorithm A*

ALGORITHM A*

Let $f(n) = g(n) + h(n)$ in step 8 of GRAPH-SEARCH where:

- $g(n)$: estimate of the cost of a minimum length path $s \rightarrow n$
- $h(n)$: estimate of the cost of a minimum length path $n \rightarrow t$

and $0 \leq h(n) \leq h^*(n)$ and step 7 guarantees that $g^*(n) \leq g(n)$

- Algorithm A* always finds the optimal path from $s \rightarrow t$.
- Notice that $h(n) \leq h^*(n)$ means that $h(n)$ underestimates the actual optimal cost $h^*(n)$ and thus, $h(n)$ is a conservative estimate of $h^*(n)$.
- Since $h(n)=0$ is an underestimate of $h^*(n)$ then A* with $g(n)=\text{depth}(n)$ always finds the optimal path $s \rightarrow t$. That is, BFS, is optimal.
- On the other hand $g(n) = 0$, that is heuristic search (DFS), does not carry the same guarantee as we will see later.
EEL5840: Elements of Machine Intelligence

Admissibility of A*

Admissible: An algorithm is admissible if for any graph it always terminates in an optimal path from \( s \rightarrow t \) if such a path exists. Admissibility implies GRAPH-SEARCH must terminate!

Result 1 GRAPH-SEARCH always terminates for finite graphs

Proof: In every cycle of the algorithm we remove a node from OPEN and only a finite number of new successors are added. Since the graph is finite, we ultimately run out of new successors and we will either terminate in step 5 by finding a goal or in step 3 by running out of nodes.

Can we show that A* terminates even for infinite graphs if a path from \( s \rightarrow t \) exists?

Suppose A* does not terminate, that is we never quit adding nodes to OPEN. Then even the smallest \( f(n) \) will eventually grow to \( \infty \).

Why? \( f(n) = g(n) + h(n) \) & since \( g(n) \) is a depth component \( g(n) \rightarrow \infty \) as \( n \rightarrow \infty \) {infinite graphs have infinite depth}

Recall by definition \( c(n_i, n_j) \geq e > 0 \) (\( e \) is a small positive number) and step 7 of GRAPH-SEARCH guarantees that \( g(n) \geq g^*(n) \) or \( g^*(n) \leq g(n) \) and \( 0 \leq h(n) \leq h^*(n) \) in A*

Let \( d'(n) \): length of the shortest path \( s \rightarrow n \)

\[ \therefore g(n) \geq g^*(n) \geq d'(n)e \] and since \( f(n) = g(n) + h(n) \) and \( h(n) \geq 0 \)

then \( f(n) \geq g(n) \geq d'(n)e \)

{every node \( n \) on OPEN is at least as large as \( d'(n)e \)}
Admissibility of A*

But if A* does not terminate and we never quit adding nodes to OPEN, then \( f(n) \to \infty \) since \( d(n) \to \infty \) as \( n \to \infty \). These are large \( f \)-value nodes added to the existing nodes in OPEN.

To show A* terminates we will now show that there is always one node \( n' \) on OPEN that has a finite value given by \( f(n) \leq f^*(s) \)

Let a path \( s \to n_k \) be optimal and ordered, that is

\[
\text{path}^*(s \to n_k) = \{ s=n_0, n_1, n_2, \ldots, n_{k-1}, n_k \}
\]

Before termination let \( n' \) be the 1st node on OPEN that is a member of the path \( s \to n_k \).

Q: Is there such a member \( n' \)?
A: Yes! Why? To start with \( s=n_0 \) is a member of OPEN and after we enter the loop \( n_k \) cannot be a member of CLOSED (else we’ve terminated!)

Algorithm A*

Example:

Let \( a's=\{0.5\} \ b's=\{1.2, 1, 1.5\} \ c's=\{1, 0.5, 0.75\} \)

Initially: OPEN =\{s\} CLOSED=\{\}

1st Pass: Expand s,
M=\( \Gamma(s)=\{l_1,m_1,r_1\} \)
OPEN =\{l_1,m_1,r_1\}
G=\{s, l_1, m_1, r_1\}
Pointers=\{nil, s, s, s\}
CLOSED=\{s\}
\{0.5,1,1.2\}
\therefore \text{ Expand } l_1

Example:

Let \( a's=\{0.5\} \ b's=\{1.2, 1, 1.5\} \ c's=\{1, 0.5, 0.75\} \)

Initially: OPEN =\{s\} CLOSED=\{\}

1st Pass: Expand s,
M=\( \Gamma(s)=\{l_1,m_1,r_1\} \)
OPEN =\{l_1,m_1,r_1\}
G=\{s, l_1, m_1, r_1\}
Pointers=\{nil, s, s, s\}
CLOSED=\{s\}
\{0.5,1,1.2\}
\therefore \text{ Expand } l_1
Admissibility of A*

2nd Pass: Expand \( l_1 \), \( M = \Gamma(l_1) = \{ l_2 \} \) CLOSED = \{ s, l_1 \}
\[ G = \{ s, l_1, m_1, r_1, l_2 \} \]
Pointers = \{ nil, s, s, s, \}
OPEN = \{ l_2, m_1, r_1 \} F = \{ 1,1,1,2 \} \implies \text{Expand} \ l_2

3rd Pass: Expand \( l_2 \), \( M = \Gamma(l_2) = \{ l_3 \} \) CLOSED = \{ s, l_1, l_2 \}
\[ G = \{ s, l_1, m_1, r_1, l_2, l_3 \} \]
Pointers = \{ nil, s, s, s, l_1, l_2 \}
OPEN = \{ m_1, r_1, l_3 \} F = \{ 1,1,2,1,5 \} \implies \text{Expand} \ m_1

4th Pass: Expand \( m_1 \), \( M = \Gamma(m_1) = \{ m_2 \} \) CLOSED = \{ s, l_1, l_2, m_1 \}
\[ G = \{ s, l_1, m_1, r_1, l_2, l_3, m_2 \} \]
Pointers = \{ nil, s, s, s, l_1, l_2, m_1 \}
OPEN = \{ r_1, m_2, l_3 \} F = \{ 1,2,1,5,1,5 \} \implies \text{Expand} \ r_1

5th Pass: Expand \( r_1 \), \( M = \Gamma(r_1) = \{ r_2 \} \) CLOSED = \{ s, l_1, l_2, m_1, r_1 \}
\[ G = \{ s, l_1, m_1, r_1, l_2, l_3, m_2, r_2 \} \]
Pointers = \{ nil, s, s, s, s, l_1, l_2, m_1, r_1 \}
OPEN = \{ m_2, l_3, r_2 \} F = \{ 1,5,1,5,2,2 \} \implies \text{Expand} \ m_2

NOTE: At all times in OPEN there is a node \( n' \), which is a member of the optimal path \( \ast = \{ s, m_1, m_2, t \} \). Also note that the \( f \) value of the nodes in path \( \ast \) is \( f^\ast(s) = c_1 + c_2 + c_3 \)

6th Pass: Expand \( m_2 \), \( M = \Gamma(m_2) = \{ t \} \) CLOSED = \{ s, l_1, l_2, m_1, m_2 \}
\[ G = \{ s, l_1, m_1, r_1, l_2, l_3, m_2, r_2, t \} \]
Pointers = \{ nil, s, s, s, l_1, l_2, m_1, r_1, m_2 \}
OPEN = \{ l_3, r_2, t \} F = \{ 1,5,2,2,2,2,5 \} \implies \text{Expand} \ l_3

7th Pass: Expand \( l_3 \), \( M = \Gamma(l_3) = \{ l_4 \} \) CLOSED = \{ s, l_1, l_2, m_1, r_1, m_2, l_3 \}
\[ G = \{ s, l_1, m_1, r_1, l_2, l_3, m_2, r_2, t, l_4 \} \]
Pointers = \{ nil, s, s, s, s, l_1, l_2, m_1, r_1, m_2, l_3 \}
OPEN = \{ l_4, l_5, t \} F = \{ 2,2,1,5,2,2,2,5 \} \implies \text{Expand} \ l_4

8th Pass: Expand \( l_4 \), \( M = \Gamma(l_4) = \{ l_5 \} \) CLOSED = \{ s, l_1, l_2, m_1, r_1, m_2, l_3, l_4 \}
\[ G = \{ s, l_1, m_1, r_1, l_2, l_3, m_2, r_2, t, l_5 \} \]
Pointers = \{ nil, s, s, s, s, l_1, l_2, m_1, r_1, m_2, l_3, l_4 \}
OPEN = \{ l_5, t \} F = \{ 2,2,2,2,2,5 \} \implies \text{Expand} \ l_5

9th Pass: Expand \( l_5 \), \( M = \Gamma(l_5) = \{ t \} \) CLOSED = \{ s, l_1, l_2, m_1, r_1, m_2, l_3, l_4, t \}
\[ G = \{ s, l_1, m_1, r_1, l_2, l_3, m_2, r_2, t, l_5 \} \]
Pointers = \{ nil, s, s, s, l_1, l_2, m_1, r_1, m_2, l_3, l_4 \}
OPEN = \{ t, l_5 \} F = \{ 2,2,2,2,5 \} \implies \text{Success!} \text{ Found } t \text{ Path is } s \rightarrow m_2 \rightarrow m_1 \rightarrow t

NOTE: In step 9 above when \( n = t \) is found as a successor of \( r_2 \) step 7 of the algorithm checks and finds that \( n = t \) is already in OPEN and keeps the shortest path (the cost \( c_1 + c_2 + c_3 < b_1 + b_2 + b_3 \)) to \( n = t \) which is still through \( m_i \) and not \( r_z \).
The End!