CS 480 - Homework #1 Solutions

Aman Gill Kyle Janssen

Spring 2014

1) Show the assertions

Assertion 1. *If* $h_1(n), h_2(n)$ *are admissible then* $h(n) = max(h_1(n), h_2(n))$ *is admissible.*

Proof. Let h_1, h_2 describe admissible heuristics. By definition, $h_1(n) \leq C(n)$ and $h_2(n) \leq C(n)$ where $C(n)$ is the actual cost to get from node n to the goal node. Without loss of generality, assume $h_1 \leq h_2(n)$ for some node $n*$.

if

$$h(n) = max(h_1(n), h_2(n))$$

then

$$h(n*) = h_2(n*)$$

since

$$h_2(n*) \leq C(n*)$$

$$h(n*) \leq C(n*)$$

and $h(n)$ is therefore admissible. □

Assertion 2. *If* $h_1(n), h_2(n)$ *are consistent then* $h(n) = max(h_1(n), h_2(n))$ *is consistent.*

Proof. Let h_1, h_2 describe consistent heuristics. By definition,

$$h_1(n_j) \leq C(n_{j+1}, n_j) + h_1(n_{j+1})$$
$$h_2(n_j) \leq C(n_{j+1}, n_j) + h_2(n_{j+1})$$

where $C(n_{j+1}, n_j)$ is the cost to get from node n_j to n_{j+1}. We will also define

$$h(n) = max(h_1(n_j), h_2(n_j))$$

There will be two cases: one where one heuristic is greater than both nodes, and one where the greater heuristic switches between nodes.

Case 1: Without loss of generality, assume

$$h_1(n_j) \leq h_2(n_j)$$
$$h_1(n_{j+1}) \leq h_2(n_{j+1})$$

then

$$h(n_j) = h_2(n_j) \textit{ and } h(n_{j+1}) = h_2(n_{j+1})$$

Since h_2 is known to be consistent, h is also consistent.

Case 2: Without loss of generality, assume

$$h_1(n_j) \leq h_2(n_j)$$
$$h_2(n_{j+1}) \leq h_1(n_{j+1})$$

then

$$h(n_j) = h_2(n_j) \textit{ and } h(n_{j+1}) = h_1(n_{j+1})$$

additionally,

$$C(n_{j+1}, n_j) + h_2(n_{j+1}) \leq C(n_{j+1}, n_j) + h_1(n_{j+1})$$

since h_2 is consistent,

$$h_2(n_j) \leq C(n_{j+1}, n_j) + h_2(n_{j+1}) \leq C(n_{j+1}, n_j) + h_1(n_{j+1})$$

Therefore, by substitution, h is consistent.

$\square$

2) Exhibit the search tree

- Starting State: $\begin{array}{ccc} 2 & 8 & 3 \\ 1 & 6 & 5 \\ 7 & & 5 \end{array}$

- Goal State: $\begin{array}{ccc} 1 & 2 & 3 \\ 8 & & 4 \\ 7 & 6 & 5 \end{array}$

- Note: The state sets and search paths were generated programmatically. The code is availabe for review on github[?]

1. A^* search algorithm incorporating the Hamming distance heuristic

   (a) A^* search path (max depth of 3):

$$\begin{array}{c}\begin{array}{ccc} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & & 5 \end{array} \\ f(n) = 0 + 4 = 4 \end{array} \rightarrow \begin{array}{c}\begin{array}{ccc} 2 & 8 & 3 \\ 1 & & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 1 + 3 = 4 \end{array} \rightarrow \begin{array}{c}\begin{array}{ccc} 2 & & 3 \\ 1 & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 2 + 3 = 5 \end{array} \rightarrow$$

$$\begin{array}{c}\begin{array}{ccc} & 2 & 3 \\ 1 & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 3 + 2 = 5 \end{array} \rightarrow \begin{array}{c}\begin{array}{ccc} 1 & 2 & 3 \\ & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 4 + 1 = 5 \end{array} \rightarrow \cdots$$

   (b) Current open set:

$$\begin{array}{c}\begin{array}{ccc} 1 & 2 & 3 \\ 7 & 8 & 4 \\ & 6 & 5 \end{array} \\ f(n) = 5 + 2 = 7 \end{array} \quad \begin{array}{c}\begin{array}{ccc} 1 & 2 & 3 \\ 8 & & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 5 + 0 = 5 \end{array} \quad \begin{array}{c}\begin{array}{ccc} 2 & 3 & \\ 1 & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 3 + 4 = 7 \end{array} \quad \begin{array}{c}\begin{array}{ccc} 2 & 8 & 3 \\ & 1 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 2 + 3 = 5 \end{array}$$

$$\begin{array}{c}\begin{array}{ccc} 2 & 8 & 3 \\ 1 & 4 & \\ 7 & 6 & 5 \end{array} \\ f(n) = 2 + 4 = 6 \end{array} \quad \begin{array}{c}\begin{array}{ccc} 2 & 8 & 3 \\ 1 & 6 & 4 \\ & 7 & 5 \end{array} \\ f(n) = 1 + 5 = 6 \end{array} \quad \begin{array}{c}\begin{array}{ccc} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 5 & \end{array} \\ f(n) = 2 + 4 = 6 \end{array}$$

   (c) Current closed set:

$$\begin{array}{c}\begin{array}{ccc} & 2 & 3 \\ 1 & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 3 + 2 = 5 \end{array} \quad \begin{array}{c}\begin{array}{ccc} 1 & 2 & 3 \\ & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 4 + 1 = 5 \end{array} \quad \begin{array}{c}\begin{array}{ccc} 2 & & 3 \\ 1 & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 2 + 3 = 5 \end{array} \quad \begin{array}{c}\begin{array}{ccc} 2 & 8 & 3 \\ 1 & & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 1 + 3 = 4 \end{array}$$

2. A^* search algorithm incorporating the Manhattan distance algorithm

(a) A^* search path (max depth of 3):

$$\begin{array}{c} \begin{array}{ccc} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & & 5 \end{array} \\ f(n) = 0 + 5 = 5 \end{array} \rightarrow \begin{array}{c} \begin{array}{ccc} 2 & 8 & 3 \\ 1 & & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 1 + 4 = 5 \end{array} \rightarrow \begin{array}{c} \begin{array}{ccc} 2 & & 3 \\ 1 & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 2 + 3 = 5 \end{array} \rightarrow$$

$$\begin{array}{c} \begin{array}{ccc} & 2 & 3 \\ 1 & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 3 + 2 = 5 \end{array} \rightarrow \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \\ & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 4 + 1 = 5 \end{array} \rightarrow \quad \ldots$$

(b) Current open set:

$$\begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \\ 7 & 8 & 4 \\ & 6 & 5 \end{array} \\ f(n) = 5 + 2 = 7 \end{array} \quad \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \\ 8 & & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 5 + 0 = 5 \end{array} \quad \begin{array}{c} \begin{array}{ccc} 2 & 3 & \\ 1 & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 3 + 4 = 7 \end{array} \quad \begin{array}{c} \begin{array}{ccc} 2 & 8 & 3 \\ & 1 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 2 + 5 = 7 \end{array}$$

$$\begin{array}{c} \begin{array}{ccc} 2 & 8 & 3 \\ 1 & 4 & \\ 7 & 6 & 5 \end{array} \\ f(n) = 2 + 5 = 7 \end{array} \quad \begin{array}{c} \begin{array}{ccc} 2 & 8 & 3 \\ 1 & 6 & 4 \\ & 7 & 5 \end{array} \\ f(n) = 1 + 6 = 7 \end{array} \quad \begin{array}{c} \begin{array}{ccc} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 5 & \end{array} \\ f(n) = 1 + 6 = 7 \end{array}$$

(c) Current closed set:

$$\begin{array}{c} \begin{array}{ccc} & 2 & 3 \\ 1 & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 3 + 2 = 5 \end{array} \quad \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \\ & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 4 + 1 = 5 \end{array} \quad \begin{array}{c} \begin{array}{ccc} 2 & & 3 \\ 1 & 8 & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 2 + 3 = 5 \end{array} \quad \begin{array}{c} \begin{array}{ccc} 2 & 8 & 3 \\ 1 & & 4 \\ 7 & 6 & 5 \end{array} \\ f(n) = 1 + 4 = 5 \end{array}$$

3) Justify a true/false answer

a) False. For a weighted, directed graph, increasing all of the weights of the edges by some amount A would cause the cost of the shortest path from node s to node g to increase by an amount proportional to the number of steps in that path. If the shortest path is not also the path with the fewest steps and A is of sufficient magnitude, the path with the fewest steps from s to g would become the shortest path.

b) True. When best first search is implemented using an admissible heuristic function, it is the A* algorithm, but unless the heuristic function is also consistent, the possibility remains that a node may be discovered more than once and its cost improved.

4) Pancake sorting problem

a) The heuristic function $h(x) = 1/2$ * the number of breakpoints is admissible. Each step between states costs 1, and each step away from the goal node can increase the number of breakpoints by 2, or $h(x)$ by 1. Because the heuristic and the cost to get to the goal increase at the same rate as we move further from the goal in our state space, $h(x)$ can never overestimate cost, and is therefore admissible.

Furthermore, $h(x)$ is consistent. Because the cost of going from one state to another is always 1, and the maximum increase in $h(x)$ between nodes is 1, $h(x_j) \leq h(x_{j+1}) + 1$ where x_{j+1} is one step closer to the goal than x_j.
b) The search tree using A* with ¡ 1 3 6 4 2 5 ¿ as the initial state. States expanded from the initial state indicate the input required to get to them, as well as their $h, g,$ *and* f A* values. The arrows and letters indicate which states were expanded in each iteration. In the case of a tie, $h(x)$ is used as a tie breaker. The goal node, which is found on the fourth iteration, is underlined. All states that are not struck-out are in the open set when the algorithm finishes and those that are struck-out are in the closed set.

1st iteration:

~~$< 136425 > h(x) : 2, g(x) : 0, f(x) : 2$~~ $< - 1$

2nd iteration:

$1,2 :< 316425 > h(x) : 2, g(x) : 1, f(x) : 3$
$1,3 :< 631425 > h(x) : 2, g(x) : 1, f(x) : 3$
$1,4 :< 463125 > h(x) : 2, g(x) : 1, f(x) : 3$
$1,5 :< 246315 > h(x) : 2, g(x) : 1, f(x) : 3$
$1,6 :< 524631 > h(x) : 2, g(x) : 1, f(x) : 3$
$2,3 :< 163425 > h(x) : 2, g(x) : 1, f(x) : 3$
$2,4 :< 146325 > h(x) : 2, g(x) : 1, f(x) : 3$
$2,5 :< 124635 > h(x) : 2, g(x) : 1, f(x) : 3$
$2,6 :< 152463 > h(x) : 2, g(x) : 1, f(x) : 3$
$3,4 :< 134625 > h(x) : 2, g(x) : 1, f(x) : 3$
~~$3,5 :< 132465 > h(x) : 1, g(x) : 1, f(x) : 2$~~ $< - 2$
$3,6 :< 135246 > h(x) : 2, g(x) : 1, f(x) : 3$
$4,5 :< 136245 > h(x) : 2, g(x) : 1, f(x) : 3$
$4,6 :< 136524 > h(x) : 2, g(x) : 1, f(x) : 3$
$5,6 :< 136452 > h(x) : 2, g(x) : 1, f(x) : 3$

3rd iteration:

$1,2 :< 312465 > h(x) : 1, g(x) : 2, f(x) : 3$
$1,3 :< 231465 > h(x) : 1, g(x) : 2, f(x) : 3$
$1,4 :< 423165 > h(x) : 1, g(x) : 2, f(x) : 3$
$1,5 :< 642315 > h(x) : 2, g(x) : 2, f(x) : 4$
$1,6 :< 564231 > h(x) : 1, g(x) : 2, f(x) : 3$
~~$2,3 :< 123465 > h(x) : 0, g(x) : 2, f(x) : 2$~~ $< - 3$

$2,4 :< 142365 > h(x) : 1, g(x) : 2, f(x) : 3$
$2,5 :< 164235 > h(x) : 2, g(x) : 2, f(x) : 4$
$2,6 :< 156423 > h(x) : 1, g(x) : 2, f(x) : 3$
$3,4 :< 134265 > h(x) : 1, g(x) : 2, f(x) : 3$
$3,6 :< 135642 > h(x) : 2, g(x) : 2, f(x) : 4$
$4,5 :< 132645 > h(x) : 1, g(x) : 2, f(x) : 3$
$4,6 :< 132564 > h(x) : 1, g(x) : 2, f(x) : 3$
$5,6 :< 132456 > h(x) : 1, g(x) : 2, f(x) : 3$

4th iteration:

$1,2 :< 213465 > h(x) : 1, g(x) : 3, f(x) : 4$
$1,3 :< 321465 > h(x) : 1, g(x) : 3, f(x) : 4$
$1,4 :< 432165 > h(x) : 0, g(x) : 3, f(x) : 3$
$1,5 :< 643215 > h(x) : 1, g(x) : 3, f(x) : 4$
$1,6 :< 564321 > h(x) : 0, g(x) : 3, f(x) : 3$
$2,4 :< 143265 > h(x) : 1, g(x) : 3, f(x) : 4$
$2,5 :< 164325 > h(x) : 1, g(x) : 3, f(x) : 4$
$2,6 :< 156432 > h(x) : 1, g(x) : 3, f(x) : 4$
$3,4 :< 124365 > h(x) : 1, g(x) : 3, f(x) : 4$
$3,5 :< 126435 > h(x) : 1, g(x) : 3, f(x) : 4$
$3,6 :< 125643 > h(x) : 1, g(x) : 3, f(x) : 4$
$4,5 :< 123645 > h(x) : 1, g(x) : 3, f(x) : 4$
$4,6 :< 123564 > h(x) : 1, g(x) : 3, f(x) : 4$
$5,6 :< 123456 > h(x) : 0, g(x) : 3, f(x) : 3 <-$ 4 - Goal Node

The code[?] used to produce these results did not run the A^* algorithm, but rather defined the game logic and printed out state and heuristic information.

References

[1] Aman Gill, CS480HW1 Git repository.
https://github.com/WorldDotHack/CS480HW1

[2] Kyle Janssen, PancakeNumbers Git repository.
https://github.com/WizardCode/PancakeNumbers