CS3243 Introduction to Artificial Intelligence

AY2021/2022 Semester 1

Tutorial 2: Informed Search

Important Instructions:

- **Assignment 2** consists of **Question 6** and **Question 7** from this tutorial.
- Your solutions for the above questions may be handwritten or typewritten, but handwritten solutions must be legible for marks to be awarded. If you are submitting handwritten solutions, please append the question paper in front of your solutions.
- You are to submit your solutions on LumiNUS by **Week 5 Monday (6 September), 2359 hours**, under **Tutorial Submission → Tutorial 2**.
- Questions marked with an asterisk (*) are harder and unlikely to appear in tests/exams.

Note: you may discuss the content of the questions with your classmates, but you must work out and write up your solution individually - solutions that are plagiarised will be heavily penalised.

Tutorial Questions

(1) (a) Under the context of informed search, state the difference between the terms $f(n)$, $\hat{f}(n)$, $\hat{f}_{\text{pop}}(n)$, $g(n)$, $\hat{g}(n)$, and $h(n)$.

Solution: The function $\hat{g}(n)$ denotes the current best known path cost from the initial state to node n. Note that the value of $\hat{g}(n)$ is updated during the execution of the algorithm. The function $g(n)$ denotes the minimum path cost from an initial state to state n.

The function $h(n)$, known as a heuristic, denotes the approximated path cost from state n to the (nearest) goal state.

The functions $f(n)$ and $\hat{f}(n)$ are evaluation functions that are adopted by the informed search algorithm in question. For example, in the case of the greedy best-first search algorithm, $f(n) = \hat{f}(n) = h(n)$; in the case of the A^* search algorithm, $f(n) = g(n) + h(n)$ and $\hat{f}(n) = \hat{g}(n) + h(n)$. $\hat{f}_{\text{pop}}(n)$ denotes the value of $\hat{f}(n)$ when it is popped from the frontier.

(b) Prove that the graph-based variant of the A^* **Search** algorithm is optimal when a consistent heuristic is utilized.

Solution: Mathematically, we would want to prove that $\hat{f}_{\text{pop}}(s_g) = f(s_g)$, i.e. when the goal node s_g is popped from the frontier, we would have found the optimal path to it. Let

$$s_0, s_1, \ldots, s_{g-1}, s_g$$

be the path from the start node s_0 leading to the goal node s_g.

Base case: $\hat{f}_{\text{pop}}(s_0) = f(s_0) = h(s_0)$.

Induction step: Assume that for all $s_0, s_1, \ldots, s_k$, $\hat{f}_{\text{pop}}(s_i) = f(s_i)$. We know that

$$\begin{aligned}\hat{f}_{\text{pop}}(s_{k+1}) &= \hat{g}_{\text{pop}}(s_{k+1}) + h(s_{k+1}) \\ &\geq g(s_{k+1}) + h(s_{k+1}) \\ &= f(s_{k+1})\end{aligned} \tag{2.1}$$

In order to make sure that each s_{k+1} is only explored after when we pop s_k, the condition of $f(s_i) \leq f(s_{i+1})$ is required, leading to the need for the consistency of h. By popping s_k, we have:

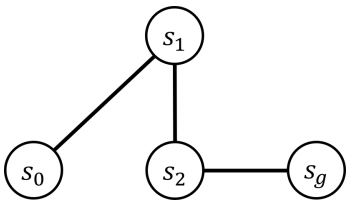
$$\begin{aligned}\hat{f}_{\text{pop}}(s_{k+1}) &= \min\{\hat{f}(s_{k+1}), \hat{g}_{\text{pop}}(s_k) + c(s_k, s_{k+1}) + h(s_{k+1})\} \\ &\leq \hat{g}_{\text{pop}}(s_k) + c(s_k, s_{k+1}) + h(s_{k+1}) \\ &= g(s_k) + c(s_k, s_{k+1}) + h(s_{k+1}) && \text{from our IH} \\ &= g(s_{k+1}) + h(s_{k+1}) \\ &= f(s_{k+1})\end{aligned} \tag{2.2}$$

From equations 2.1 and 2.2, we obtain $\hat{f}_{\text{pop}}(s_{k+1}) = f(s_{k+1})$. Hence, by induction, whenever we pop a node from the frontier, the optimal path to the node would have been found.

(2) (a) Provide a counter-example to show that the tree-based variant of the **Greedy Best-First Search** algorithm is incomplete.

Solution: Consider the following search space, where:

$$\begin{aligned}h(s_0) &= 3 \\ h(s_1) &= 4 \\ h(s_2) &= 5 \\ h(s_g) &= 0\end{aligned}$$



Each time s_0 is explored, we add s_1 to the front of the frontier, and each time s_1 is explored, we add s_0 to the front of the frontier. Notice that s_2 is never at the front of the frontier. This causes the greedy best-first search algorithm to continuously loop over s_0 and s_1.

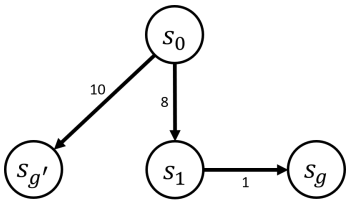
(b) Briefly explain why the graph-based variant of the **Greedy Best-First Search** algorithm is complete.

Solution: Assuming a finite search space, the graph-based variant of the greedy best-first search algorithm will eventually visit all states within the search space, and thus find a goal state.

(c) Provide a counter-example to show that neither variant of the **Greedy Best-First Search** algorithm is optimal.

Solution: Consider the following search space, where:

$$h(s_0) = 9$$
$$h(s_1) = 1$$
$$h(s_g) = 0$$
$$h(s_{g'}) = 0$$



With either variant of the greedy best-first search algorithm, when s_0 is explored, $s_{g'}$ would be added to the front of the frontier and then explored next, resulting in the algorithm returning the non-optimal $s_0 \rightarrow s_{g'}$ path.

(3) (a) Given that a heuristic h is such that $h(s_g) = 0$ where s_g is any goal state, prove that if h is consistent, then it must be admissible.

Solution: The proof can be done by induction on $k(s_i)$, which denotes the number of actions required to reach the goal from a node s_i to the goal node s_g.

Base case: ($k = 1$, i.e. the node s_i is one step away from s_g) Since the heuristic function h is consistent,

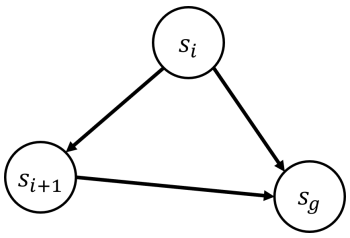
$$h(s_i) \leq c(s_i, s_g) + h(s_g)$$

Since $h(s_g) = 0$,

$$h(s_i) \leq c(s_i, s_g) = h^*(s_i)$$

Therefore, h is admissible.

Induction step:



Suppose that our assumption holds for every node that is $k-1$ actions away from s_g, and let us observe a node s_i that is k actions away from s_g; that is, the optimal path from s_i to s_g has $k > 1$ steps. We can write the optimal path from s_i to s_g as

$$s_i \rightarrow s_{i+1} \rightarrow \ldots \rightarrow s_{g-1} \rightarrow s_g$$

Since h is consistent, we have

$$h(s_i) \leq c(s_i, s_{i+1}) + h(s_{i+1})$$

Now, note that since s_{i+1} is on a least-cost path from s_i to s_g, we must have that the path $s_{i+1} \to s_{i+2} \to \ldots \to s_{g-1} \to s_g$ is a least-cost path from s_{i+1} to s_g as well. By our induction hypothesis, we have

$$h(s_{i+1}) \leq h^*(s_{i+1})$$

Combining the two inequalities, we have

$$h(s_i) \leq c(s_i, s_{i+1}) + h^*(s_{i+1})$$

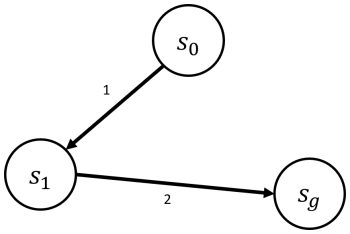
Note that $h^*(s_{i+1})$ is the cost of the optimal path from s_{i+1} to s_g; by our previous observation (that $s_{i+1} \to s_{i+2} \to \ldots \to s_{g-1} \to s_g$ is an least-cost path from s_{i+1} to s_g), we have that the cost of the optimal path from s_i to s_g, i.e. $h^*(s_i)$, equals $c(s_i, s_{i+1}) + h^*(s_i)$, which concludes the proof.

(b) Give an example of an admissible heuristic function that is not consistent.

Solution: An example of an admissible heuristic function that is not consistent is

$$h(s_0) = 3$$
$$h(s_1) = 1$$
$$h(s_g) = 0$$

for the following graph:



h is admissible since

$$h(s_0) \leq h^*(s_0) = 1 + 2 = 3$$
$$h(s_1) \leq h^*(s_1) = 2$$

However, h is not consistent since $3 = h(s_0) > c(s_0, s_1) + h(s_1) = 1 + 1 = 2$.

(4) Suppose that the A^* **Search** algorithm utilizes

$$\hat{f}(n) = w \times \hat{g}(n) + (1 - w) \times h(n)$$

where $0 \leq w \leq 1$, instead of $\hat{f}(n) = \hat{g}(n) + h(n)$. For any value of w, an optimal solution will be found whenever h is a consistent heuristic.

Determine if the statement above is True or False, and provide a rationale.

Solution: This is false. When $w = 0$, this is essentially greedy best-first search, which is suboptimal (see question 2c).

*(5) We have seen various search strategies in class, and analyzed their worst-case running time. Prove that *any deterministic search algorithm* will, in the worst case, search the entire state space. More formally, prove the following theorem:

Theorem 2.1

Let $\mathcal{A}$ be some complete, deterministic search algorithm. Then, for any search problem defined by a finite connected graph $G = (V, E)$, where V is the set of possible states and E is the set of transition edges between them, there exists a choice of start node s_0 and goal node s_g such that $\mathcal{A}$ searches through the entire graph G.

Solution: Let us begin by running $\mathcal{A}$ on the graph G, without setting any goal node at all. That is, there are no goal nodes at all in G. In this case, the algorithm $\mathcal{A}$ will return "False" when it explores the entire set V. Let $H_t(\mathcal{A}, s_0) \subseteq V$ be the set of nodes that $\mathcal{A}$ explores if it starts at s_0, and does not encounter a goal node at steps $1, 2, \ldots, t$ (at $t = 1$, we have $H_1(\mathcal{A}, s_0) = \{s_0\}$). We also let v_t be the node that $\mathcal{A}$ selects at time t, given that it has observed the set $H_{t-1}(\mathcal{A}, s_0)$ so far. We note that it is entirely possible that $\mathcal{A}$ selects $v_t \in H_{t-1}(\mathcal{A}, s_0)$. However, we make a simple observation: the sequence $(H_t(\mathcal{A}, s_0))_{t=1}^{\infty}$ is weakly increasing in size, and there exists some time t^* such that for all $t > t^*$, $H_t(\mathcal{A}, s_0) = V$. In other words, since $\mathcal{A}$ is a complete search algorithm, it will continue exploring the nodes in G until all nodes have been explored. Let us assume that t^* is the first time step for which $H_t(\mathcal{A}, s_0) = V$. In other words, at time $t^* - 1$, $|H_{t^*-1}(\mathcal{A}, s_0)| = |V| - 1$.

We now set the goal node to be v_{t^*}. From our previous argument, we know that when $\mathcal{A}$ starts at s_0, it will explore a set of size $|V| - 1$ before reaching v_{t^*}, realizing that it is a goal node and terminating. In other words, for any node s_0, if we select a goal node according to the above procedure, the algorithm $\mathcal{A}$ will exhaustively search through the entire graph before reaching a goal node.

Here is another, inductive proof. Let us set the goal node to be some arbitrary node g_1. If $\mathcal{A}$ searches through the entire graph G when g_1 is the goal, we are done. Otherwise, let U_1 be the set of unexplored nodes when g_1 is the goal node. We take an arbitrary node $g_2 \in U_1$ to be the goal. Since $\mathcal{A}$ is deterministic and complete, it will run the same search order as it has previously ran when g_1 was the goal, and then search through the nodes in U_1 until it reaches g_2. If it searched through all the nodes in U_1 as well, we are done. Otherwise, we repeat.

In general, suppose that we have set g_t to be the goal node, and that $\mathcal{A}$ did not search through the entire graph until it reached g_t. Let U_t be the set of unsearched nodes when g_t is the goal node. We set g_{t+1} to be some arbitrary node in U_t and rerun $\mathcal{A}$. Since $\mathcal{A}$ is deterministic, we know that when g_{t+1} is the goal node, we have $U_{t+1} \subset U_t$. Since $U_1 \supset U_2 \supset \ldots \supset U_t$, and the number of nodes in G is finite, there exists some iteration t^* such that $U_{t^*} = \varnothing$. Thus, g_{t^*} is a goal node for which $\mathcal{A}$ searches through the entire graph.

Assignment Questions

(6) (a) In the search problem below, we have listed 5 heuristics. Indicate whether each heuristic is admissible and/or consistent in the table below.

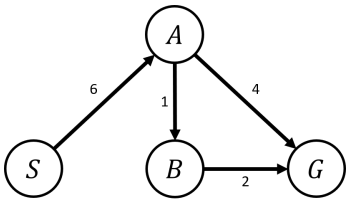


Figure 2.1. Graph for question 6.

	S	A	B	G	Admissible	Consistent
h_1	0	0	0	0	True	True
h_2	8	1	1	0	True	False
h_3	9	3	2	0	True	True
h_4	6	3	1	0	True	False
h_5	8	4	2	0	False	False

(b) Write out the order of the nodes that is explored by A^* **Search** with graph-based implementation, when using heuristic h_4. You must express your answer in the same manner as in question (1).

Solution: $S - A - B - G$

(c) Which heuristic would you use? Explain why.

Solution: The heuristic h_3 corresponds to the exact cost from each node to the goal node (i.e. $h_3 = h^*$), and therefore it is the optimal heuristic.

(d) Prove or disprove the following statement:

The heuristic $h(n) = \max\{h_3(n), h_5(n)\}$ is admissible.

Solution: This is false, since $4 = h(A) > h^*(A) = 3$.

(7) Assume there exists just a single goal node s_g, and an initial node s_0. Assume further that there is a path from s_0 to s_g.

Suppose you have the function $OPT(s_i)$, which returns the optimal path cost from a node s_i to the goal node s_g. Recall that a path is defined as a sequence of nodes, e.g. if $\pi = s_0, s_1, s_2, s_3$, we say that nodes s_0, s_1, s_2, s_3 lie on π.

We are given a heuristic function h such that

$$h(s_i) = \begin{cases} \frac{OPT(s_i)}{2} & \text{if } s_i \text{ lies on an optimal path from } s_0 \text{ to } s_g \\ k(s), \text{ where } k(s) > OPT(s) & \text{if no optimal path from } s_0 \text{ to } s_g \text{ contains } s_i \end{cases}$$

Observe that when s_i does not lie on any optimal path from s_0 to s_g, $h(s_i)$ can take any value that is larger than $OPT(S_i)$. Prove or disprove the following statement:

*An A^* **Search** algorithm with graph-based implementation will always find an optimal path from s_0 to s_g with the above heuristic function h.*

You may not assume anything about the graph structure, other than the existence of the initial node s_0 and the goal node s_g. Also, no further assumptions or restrictions on h are allowed.

Solution: Taking the optimal path from s_0 to s_g to be

$$\pi^* : s_0, s_1, \ldots, s_k, s_{k+1}, \ldots, s_g$$

If we can prove that:

(a) $f(s_0) \leq f(s_1) \leq \ldots \leq f(s_g)$, and

(b) $\hat{f}_{\text{pop}}(s_i) = f(s_i)$

then, we could use these conditions to show that A^* graph search with the heuristic will be optimal.

Note that it suffices to show that condition 1 holds for the given heuristic *along its optimal path*. This is because we only require that s_k is popped before s_{k+1} if s_k lies before s_{k+1} along the optimal path.

We know condition 1 holds along the optimal path because:

$$\begin{aligned} \forall s_i \in \pi^*, h(s_i) &= \frac{OPT(s_i)}{2} \\ &= \frac{c(s_i, s_{i+1}) + OPT(s_{i+1})}{2} \\ &< c(s_i, s_{i+1}) + \frac{OPT(s_{i+1})}{2} \\ &= c(s_i, s_{i+1}) + h(s_{i+1}) \end{aligned}$$

Condition 2 follows from induction. The base case is $\hat{f}_{\text{pop}}(s_0) = f(s_0)$, and we assume that for all $s_0, s_1, ..., s_k$, $\hat{f}_{\text{pop}}(s_k) = f(s_k)$. We know that when a node s_k along the optimal path is popped, we have

$$\begin{aligned} \hat{f}_{\text{pop}}(s_{k+1}) &= \min\{\hat{f}(s_{k+1}), \hat{g}_{\text{pop}}(s_k) + c(s_k, s_{k+1}) + h(s_{k+1})\} \\ &\leq \hat{g}_{\text{pop}}(s_k) + c(s_k, s_{k+1}) + h(s_{k+1}) \\ &= g(s_k) + c(s_k, s_{k+1}) + h(s_{k+1}) \\ &= f(s_{k+1}) \end{aligned}$$

Thus, when s_{k+1} is popped next, we have $\hat{f}_{\text{pop}}(s_{k+1}) = f(s_{k+1})$.