

Q1. Searching with Heuristics

Consider the A* searching process on the connected undirected graph, with starting node S and the goal node G. Suppose the cost for each connection edge is **always positive**. We define $h^*(X)$ as the shortest (optimal) distance to G from a node X.

Answer Questions (a), (b) and (c). You may want to solve Questions (a) and (b) at the same time.

(a) Suppose h is an **admissible** heuristic, and we conduct A* **tree search** using heuristic h' and finally find a solution. Let C be the cost of the found path (directed by h' , defined in part (a)) from S to G

(i) Choose **one best** answer for each condition below.

1. If $h'(X) = \frac{1}{2}h(X)$ for all Node X , then $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$
2. If $h'(X) = \frac{h(X)+h^*(X)}{2}$ for all Node X , then $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$
3. If $h'(X) = h(X) + h^*(X)$ for all Node X , then $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$
4. If we define the set $K(X)$ for a node X as all its neighbor nodes Y satisfying $h^*(X) > h^*(Y)$, and the following always holds

$$h'(X) \leq \begin{cases} \min_{Y \in K(X)} h'(Y) - h(Y) + h(X) & \text{if } K(X) \neq \emptyset \\ h(X) & \text{if } K(X) = \emptyset \end{cases}$$

then,

- $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$

5. If K is the same as above, we have

$$h'(X) = \begin{cases} \min_{Y \in K(X)} h(Y) + \text{cost}(X, Y) & \text{if } K(X) \neq \emptyset \\ h(X) & \text{if } K(X) = \emptyset \end{cases}$$

where $\text{cost}(X, Y)$ is the cost of the edge connecting X and Y ,

then,

- $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$

6. If $h'(X) = \min_{Y \in K(X) \cup \{X\}} h(Y)$ (K is the same as above), $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$

(ii) In which of the conditions above, h' is still **admissible** and for sure to dominate h ? Check all that apply. Remember we say h_1 dominates h_2 when $h_1(X) \geq h_2(X)$ holds for all X . 1 2 3 4 5 6

(b) Suppose h is a **consistent** heuristic, and we conduct A* **graph search** using heuristic h' and finally find a solution.

(i) Answer exactly the same questions for each conditions in Question (a)(i).

1. $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$
2. $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$
3. $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$
4. $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$
5. $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$
6. $C = h^*(S)$ $C > h^*(S)$ $C \geq h^*(S)$

(ii) In which of the conditions above, h' is still **consistent** and for sure to dominate h ? Check all that apply.

- 1 2 3 4 5 6

(c) Suppose h is an **admissible** heuristic, and we conduct A* **tree search** using heuristic h' and finally find a solution.

If $\epsilon > 0$, and X_0 is a node in the graph, and h' is a heuristic such that

$$h'(X) = \begin{cases} h(X) & \text{if } X = X_0 \\ h(X) + \epsilon & \text{otherwise} \end{cases}$$

- Alice claims h' can be inadmissible, and hence $C = h^*(S)$ does not always hold.
- Bob instead thinks the node expansion order directed by h' is the same as the heuristic h'' , where

$$h''(X) = \begin{cases} h(X) - \epsilon & \text{if } X = X_0 \\ h(X) & \text{if otherwise} \end{cases}$$

Since h'' is admissible and will lead to $C = h^*(S)$, and so does h' . Hence, $C = h^*(S)$ always holds.

The two conclusions (underlined) apparently contradict with each other, and **only exactly one of them are correct and the other is wrong**. Choose the **best** explanation from below - which student's conclusion is wrong, and why are they wrong?

- Alice's conclusion is wrong, because the heuristic h' is always admissible.
- Alice's conclusion is wrong, because an inadmissible heuristics does not necessarily always lead to the failure of the optimality when conducting A* tree search.
- Alice's conclusion is wrong, because of another reason that is not listed above.
- Bob's conclusion is wrong, because the node visiting expansion ordering of h'' during searching might not be the same as h' .
- Bob's conclusion is wrong, because the heuristic h'' might lead to an incomplete search, regardless of its optimally property.
- Bob's conclusion is wrong, because of another reason that is not listed above.

Q2. CSPs

In this question, you are trying to find a four-digit number satisfying the following conditions:

1. the number is odd,
2. the number only contains the digits 1, 2, 3, 4, and 5,
3. each digit (except the leftmost) is strictly larger than the digit to its left.

(a) CSPs

We will model this as a CSP where the variables are the four digits of our number, and the domains are the five digits we can choose from. The last variable only has 1, 3, and 5 in its domain since the number must be odd. The constraints are defined to reflect the third condition above. Thus before we start executing any algorithms, the domains are

1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 3 4 5
-----------	-----------	-----------	---------

- (i)** Before assigning anything, enforce arc consistency. Write the values remaining in the domain of each variable after arc consistency is enforced.

--	--	--	--

- (ii)** With the domains you wrote in the previous part, which variable will the MRV (Minimum Remaining Value) heuristic choose to assign a value to first? If there is a tie, choose the leftmost variable.

- The first digit (leftmost)
- The second digit
- The third digit
- The fourth digit (rightmost)

- (iii)** Now suppose we assign to the leftmost digit first. Assuming we will continue filtering by enforcing arc consistency, which value will LCV (Least Constraining Value) choose to assign to the leftmost digit? **Break ties from large (5) to small (1).**

- 1
- 2
- 3
- 4
- 5

- (iv)** Now suppose we are running min-conflicts to try to solve this CSP. If we start with the number 1332, what will our number be after one iteration of min-conflicts? Break variable selection ties from left to right, and **break value selection ties from small (1) to large (5).**

- (b) The following questions are completely unrelated to the above parts. Assume for these following questions, there are only binary constraints unless otherwise specified.
- (i) [true or false] When enforcing arc consistency in a CSP, the set of values which remain when the algorithm terminates does not depend on the order in which arcs are processed from the queue.
 - (ii) [true or false] Once arc consistency is enforced as a pre-processing step, forward checking can be used during backtracking search to maintain arc consistency for all variables.
 - (iii) In a general CSP with n variables, each taking d possible values, what is the worst case time complexity of enforcing arc consistency using the AC-3 method discussed in class?
 0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞
 - (iv) In a general CSP with n variables, each taking d possible values, what is the maximum number of times a backtracking search algorithm might have to backtrack (i.e. the number of the times it generates an assignment, partial or complete, that violates the constraints) before finding a solution or concluding that none exists?
 0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞
 - (v) What is the maximum number of times a backtracking search algorithm might have to backtrack in a general CSP, if it is running arc consistency and applying the MRV and LCV heuristics?
 0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞