Harvard CS 182: Artificial Intelligence October 15, 2022

Section 6: Fall 2022 Midterm Review Solutions

Authors: Janani Sekar, Mujin Kwun

Practice Problem Solutions

Search Problems

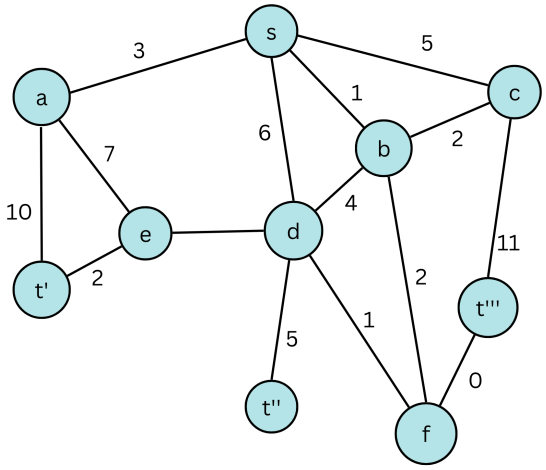
1. True of False

   - Iterative deepening search is guaranteed to expand at least as many nodes than BFS (on any graph where the goal is not on the root or the first level). True, because iterative deepening search is like BFS in that it traverses the tree by level, but it runs DFS to that particular level. This means that sometimes it expands the same node twice to find the same node that BFS would have found by expanding only once. The main advantage of iterative deepening search is that it is less memory intensive than BFS.
   - A* search with a heuristic that is not completely admissible may still find the shortest path to the goal state. True, consider a heuristic that is inadmissible because it overestimates the the cost of the cheapest path to the goal for a single node that isn't on the optimal path to begin with. A* would still find an optimal solution.
   - A* search with the heuristic $h(n) = 0$ is equivalent to the uniform-cost search This is just true by definition of $f(x)$, which equals $g(x) + h(x)$, where $g(x)$ is the true cost by which uniform cost search decides which node to expand.
   - If there are two admissible heuristic functions, h_1 and $h2$, for the same search problem, then the heuristic $h_3 = (h_1 + h_2)/2$ is admissible. True, the average of two admissible heuristics is going to be less than the bigger of the two heuristics, which guarantees that $h3$ is an underestimate of the true cost, and therefore admissible.
   - Greedy (best-first) search, which expands by $h(n)$, is both complete and optimal when the heuristic is admissible and the path cost never decreases. False, because if $f(x)$ is an increasing function of depth, with a heuristic such as the trivial heuristic, greedy search may expand to a node very deep in the search tree, when there is an optimal solution with a lower path cost.
   - For a search problem, the path returned by uniform cost search may change if we add a positive constant C to every step cost. Consider that there are two paths

from the start state s to the goal t, $s \to x \to t$ and $s \to t$. $c(s,x) = 1, c(x,t) = 1$, and $c(s,t) = 3$. So the optimal path is through x. Now, if we add 2 to each of the costs, the optimal path is directly from s to t. Since uniform cost search finds the optimal path, its path will change.

2. Informed Search:

   For a given search problem, we have the following state space representation, where the blue nodes are all the possible states that can be visited, and the numbers along the edges are the true cost of traveling to a state along the edge (e.g. the cost of going from node a to node e is 7). The start state is node s at the top of the graph, and there are 3 possible goal states t', t'', t''', which may or may not be optimal.



   We are given a heuristic h, which evaluates the cost of traveling to each node as the following:

$$h(s) = 8$$
$$h(a) = 9$$
$$h(b) = 1$$
$$h(c) = 3$$
$$h(d) = 4$$
$$h(e) = 1$$
$$h(f) = 5$$
$$h(t') = h(t'') = h(t''') = 0$$

   Using the given heuristic and the actual edge costs provided in the graph above, evaluate the order in which the nodes are expanded and which identify which goal is reached using:

- Greedy Search:
- A* Graph Search

Greedy search: Goal reached is t''', states visited are s, b, c, t'''

A* Tree search: Goal reached is t''', states visited are s, b, c, f, t'''

3. Heuristics:

   Consider the following modified version of the Tower of Hanoi problem, where there are 3 disks of different sizes, and 3 possible pegs they can be placed on. The rules of the game are that a disk may only be moved if it is on the top of a stack, and it can only be placed on an empty peg or a larger disk. The goal is to move all 3 disks to the 3rd peg.

   To formulate this as a search problem, let the disks be called S (small), M (medium), and L (large). We will represent the states as 3 ordered sets (one for each peg, called 1, 2, and 3), where the first element of the set is the top disk, and the last element the bottom disk. (Obviously, the sets can also be empty.)

   We also assume that S weighs 1 unit, M weighs 2 units, and L weighs 3 units. The cost of a move is the weight of the disk, multiplied by the distance of the move (1 for a neighboring peg, 2 for a peg 2 steps away).

   For this search problem, which of the following heuristics is admissible? Additionally, of the admissible heuristics, which one dominates the rest?

   - The number of disks on peg 3, Not admissible. Think about the case where the 2 large disks are on peg three and the short disk is on peg 2, so one move away from solving the problem, or the goal state. This heuristic would evaluate to 2, which is greater than 1.
   - The number of disks NOT on peg 3 Admissible. The number of moves left to solve the game, or find the goal state, is at least the number of disks missing from peg 3.
   - The weight of the disks NOT on peg 3 Admissible. Instead of thinking about a particular state as a certain number of moves away from the goal state, we can think of it as a certain amount of weight away from the goal state.
   - The weight of the disks NOT on peg 3 multiplied by their distances from peg 3. Admissible. For example, if the large disk is missing from peg 3 and is on peg 1, the cost of moving the large disk is at least 2 (for its distance) * 3 (for its weight). This heuristic dominates the rest, since it accounts for the cost of an action and the weight of the disk, unlike the rest.

Constraint Satisfaction Problems

1. True or False

- Value and variable ordering can decrease the number of expansions performed when using search to determine that a CSP has no solution. False, when there is no solution to a CSP, backtracking search will have to iterate through every possible assignment before returning no solution.
- Value and variable ordering can decrease the number of expansions performed when using search to determine that a CSP has a solution. True, you can think of this like the DFS question on Problem Set 1. If the search algorithm gets lucky and the first assignment it picks solves the CSP, then the number of expansions performed are much fewer.

2. Formulating a CSP

   The Harvard Administration wants to hear student opinions on a given issue and is organizing group meetings with Dean Khurana. There are n groups with a total capacity of m students across all groups. They want to hear from students in all 12 houses. We want to assign enough group spots to students in each house (e.g. bigger houses should get more spots).

   We also want to make sure all the students from the same house are in the same group. We will assume there are 12 houses. We will define s_i as the number of spots that house i gets, and g_i as the number of spots in group i.

   Formulate this problem as a CSP by defining your variables, domains, and constraints.

   There are many possible ways to configure this problem as a CSP, but here is one approach: Variables: G_{ij} and S_{ij} for assigning students to groups and spots, with $i \in \{1...12\}$, $j \in \{1...s_i\}$. Domains are $G_{ij} : \{1...n\}, S_{ij} : \{1...m\}$. Constraints are that $\forall i, j, k : G_{ij} = G_{ik}$ since two students from the same house i must be in the same group and $\forall i, j, k : S_{ij} \neq S_{ik}$ since no two students can occupy the same spot.

3. Solving a CSP

   Halloween is coming up and there is going to be a potluck! A is hosting the potluck and asks her guests to pick one of the following items to bring:

   - Pumpkin pie
   - Spooky jello
   - Caramel apples
   - Candy corn
   - Mysterious punch
   - Gummy worms

   A has invited her friends B, C, D, and E. All 5 of the potluck attendees (A included) are responsible for choosing 1 item from the list above. However, there are a few constraints on what they can bring:

   - C and D are very competitive, so if C brings the same item as another friend, it cannot be D.

- E is allergic to pumpkins, so he can't bring pie.
- D can't bring jello, apples, or worms.
- C can't bring jello, candy corn, or worms.
- B can't bring punch or worms.
- A will only bring an item that is earlier in the list than B.
- A also can't bring worms.
- B and C will only bring items that start with the same letter (e.g. candy corn and caramel apples).
- D will only bring an item that is later in the list than E.
- A and B want to be unique and therefore will not bring the same item as anyone else.

Which of these constraints are unary constraints and which are binary constraints?

Unary constraints: 2, 3, 4, 5, 7

Binary constraints: all of the rest.

Set up the CSP by defining the variables and domains after enforcing all of the unary constraints.

Domains: A: [pie, jello, caramel apples, corn, punch] B: [pie, jello, caramel apples, corn] C: [pie, caramel apples, punch] D: [pie, corn, punch] E: [jello, caramel apples, corn, punch, worms]

What values remain in each of the domains after enforcing arc-consistency? Domains: A: [pie, jello] B: [corn] C: [caramel apples] D: [punch] E: [jello, caramel apples]

Convex Optimization

Convex Sets and Convex Functions

1. A norm is any function $\|*\| : \mathbb{R}^n \to \mathbb{R}$ such that:
   (a) $\|x\| \geq 0$
   (b) $\|x\| = 0$ if and only if $x = 0$
   (c) $\|ax\| = |a|\|x\|, \ \forall a \in \mathbb{R}$
   (d) $\|x + y\| \leq \|x\| + \|y\|$

   Show that the unit norm ball , $\{x | \|x\| \leq 1\}$ is convex, regardless of norm chosen.

Let C be a unit norm ball: $C = \{x | \|x\| \leq 1\}$. Take $x, y \in C$ and $\theta \in [0, 1]$. Then, to show convexity, we must check that $\theta x + (1-\theta)y \in C$ as well

$$\begin{aligned}
\|\theta x + (1-\theta)y\| &\leq |\theta x\| + \|(1-\theta)y\| && \text{(Using (d) above)} \\
&= \theta\|x\| + (1-\theta)\|y\| \\
&= 2 \cdot ((xy)^2 + (zw)^2) && \text{(part c)} \\
&\leq \theta + (1-\theta) && \text{(unit norm ball definition)} \\
&= 1
\end{aligned}$$

We have shown that $\theta x + (1-\theta)y \leq 1$. Therefore $\theta x + (1-\theta)y \in C$ and C is convex ■

2. Let $f, g : \mathbb{R} \to \mathbb{R}$ and h be the composition: $h(x) = f(g(x))$. State whether h is convex or concave in each of these scenarios and explain why

   (a) f is monotonically increasing and both f and g are convex

   Take $x, y \in \mathbb{R}$ and $\theta \in [0, 1]$. We want to show that $(f \circ g)(\theta x + (1-\theta)y) \leq \theta(f \circ g)(x) + (1-\theta)(f \circ g)(y)$

   $$\begin{aligned}
   (f \circ g)(\theta x + (1-\theta)y) &= f(g(\theta x + (1-\theta)y)) \\
   &\leq f(\theta g(x) + (1-\theta)g(y)) && \text{(g is convex and f is nondecreasing)} \\
   &\leq \theta f(g(x)) + (1-\theta)f(g(y)) && \text{(f convex)} \\
   &= \theta(f \circ g)(x) + (1-\theta)(f \circ g)(y)
   \end{aligned}$$

   ■

   (b) f is monotonically decreasing, f is convex and g is concave

   Take $x, y \in \mathbb{R}$ and $\theta \in [0, 1]$. We want to show that $(f \circ g)(\theta x + (1-\theta)y) \leq \theta(f \circ g)(x) + (1-\theta)(f \circ g)(y)$

   $$\begin{aligned}
   (f \circ g)(\theta x + (1-\theta)y) &= f(g(\theta x + (1-\theta)y)) \\
   &\leq f(\theta g(x) + (1-\theta)g(y)) && \text{(g is concave f is decreasing)} \\
   &\leq \theta f(g(x)) + (1-\theta)f(g(y)) && \text{(f convex)} \\
   &= \theta(f \circ g)(x) + (1-\theta)(f \circ g)(y)
   \end{aligned}$$

   ■

   (c) f is monotonically decreasing, f is concave and g is convex

   h is concave Take $x, y \in \mathbb{R}$ and $\theta \in [0, 1]$. We want to show that $(f \circ g)(\theta x + (1-\theta)y) \geq \theta(f \circ g)(x) + (1-\theta)(f \circ g)(y)$

   $$\begin{aligned}
   (f \circ g)(\theta x + (1-\theta)y) &= f(g(\theta x + (1-\theta)y)) \\
   &\geq f(\theta g(x) + (1-\theta)g(y) && \text{(g is convex f is decreasing)} \\
   &\geq \theta f(g(x)) + (1-\theta)f(g(y)) && \text{(f concave)} \\
   &= \theta(f \circ g)(x) + (1-\theta)(f \circ g)(y)
   \end{aligned}$$

3. Using your results from part 2, is the following function convex or concave?

$$f(x) = \frac{1}{log(x)}, \text{ for } x > 1$$

convex. $\frac{1}{x}$ is convex and decreasing (for $x > 0$) and $log(x)$ is concave for x greater than 1

Integer Programming

Felonious Gru decides to abandon his life of crime and rebrands himself as Farmer Gru, starting a new farm: Rich Minion Fields. He needs help choosing which crops to grow. Each crop has a starting cost, failure risk, and expected profit (as a percentage of starting cost) included below

- Spinach: starting cost: 1.3, profit rate: 10%, failure risk 6%
- Potato: starting cost: 0.8, profit rate: 20%, failure risk 4%
- Rhubarb: starting cost: 0.6, profit rate: 20%, failure risk 6%
- Cavendish Banana: starting cost: 1.8, profit rate: 10%, failure risk 5%
- Red Banana: starting cost: 1.2, profit rate: 10%, failure risk 5%
- Blue Banana: starting cost: 2.4, profit rate: 10%, failure risk 4%

Gru is strapped for cash and can't afford to spend more than 4 units of money. Additionally, Gru's business partner, Dr. Nefario, refuses to agree to a crop portfolio with an average failure risk of over 5%. Gru needs help finding an optimal selection of crops. Formulate this as an integer programming problem

Solution:

Let $x_1, x_2, x_3, x_4, x_5, x_6$ represent whether spinach, potato, rhubarb, c. banana, r. banana, or b. banana are chosen. For example, let $x_1 = 1$ if spinach is chosen and 0 otherwise.

The expression we wish to maximize is the total profit (sum of profit rate*initial costs):

MAX $0.13x_1 + 0.16x_2 + 0.12x_3 + 0.18x_4 + 0.12x_5 + 0.24x_6$

Such that the total cost is not over 4 and the average failure risk is not over 5%:

$1.3x_1 + 0.8x_2 + 0.6x_3 + 1.8x_4 + 1.2x_5 + 2.4x_6 \leq 4$

$0.06x_1 + 0.04x_2 + 0.06x_3 + 0.05x_4 + 0.05x_5 + 0.04x_6 \leq 0.05(x_1 + x_2 + x_3 + x_4 + x_5 + x_6)$

$x_1, x_2, x_3, x_4, x_5, x_6 \in \{0, 1\}$