Harvard CS 182: Artificial Intelligence October 15, 2022

Section 6: Fall 2022 Midterm Review

Authors: Janani Sekar, Mujin Kwun

Conceptual Review

Search Problems

Every search problem has:

- A set of *states*, where the *start state* is the state where we begin our search.
- At least one *goal state*, or a state that meets all the constraints of the problem and ends the search algorithm.
- A *successor function* that maps a state to successor states. For each possible successor state, the function also tells us the action required to get to this next state and its cost. It returns an action, successor state, cost triplet.

We explore both tree search and graph search. You want to be familiar with the differences between tree search and graph search and how each works.

We also distinguish uninformed and informed search problems. In uninformed search, we cannot identify any characteristics about a state other than whether or not it is a goal state.

There are 4 search algorithms we look at in *uninformed search*, summarized below:

Algorithm	**Complete?**	**Optimal?**	**Time**	**Space**
BFS	Yes	Not Really	$\Theta(b^d)$	$\Theta(b^d)$
UCS	Sort Of	Yes	$\Theta(b^{\frac{C^*}{\epsilon}})$	$\Theta(b^{\frac{C^*}{\epsilon}})$
DFS	No	No	$\Theta(b^m)$	$\Theta(bm)$
IDS	Yes	No	$\Theta(b^d)$	$\Theta(bd)$

Here, b = the branching factor, d = distance from the start node, C^* = the cost of an optimal solution, ϵ = the cost per action, and m = the maximum depth of the state space.

Informed search algorithms use information beyond that given in the problem statement via a heuristic function. We look at a few informed search algorithms:

- *Greedy Search*: At any given state, expand the node with the lowest heuristic estimate $h(x)$

- *A* Tree Search*: Combine the heuristic $h(x)$ from greedy search with the cost summation, $g(x)$ to create, $f(x) = h(x) + g(x)$. We expand the node with the lowest $f(x)$.
- *A* Graph Search*: Expands using the same $f(x)$ as A* tree search but keeps track of nodes that have already been expanded in an explored set. Never expands the same node more than once.

In terms of choosing heuristics, we consider the following properties:

- A heuristic h *dominates* the heuristic h' if and only if $\forall x$, $h(x) \geq h'(x)$.
- A heuristic is *admissible* if, $\forall$ nodes x, $h(x) \leq h^*(x)$, where $h^*(x)$ represents the cost of the optimal path to a goal (i.e. it either underestimates or exactly knows the cost of the optimal path).
- A heuristic is *consistent* if, for any two nodes x, y, $h(x) \leq c(x, y) + h(y)$ where $c(x, y)$ is the cost of the cheapest path between x and y (i.e. taking a detour is more costly then remaining on the current path. Consistency implies admissibility when $h(t) = 0$ for all goals t).

You will want to know the following theorem and its proof (not included here) for the optimality of A* tree search with an admissible heuristic.

Theorem 1 *A* tree search with an admissible heuristic returns and optimal solution.*

You should also know the following theorem about the optimality of A* graph search, but will not be required to prove this.

Theorem 2 *A* graph search with a consistent heuristic returns and optimal solution.*

Motion Planning

Motion planning problems can be formulated as search problems, and we can use many of the above strategies to solve motion planning problems, which typically involve finding a path through a given space. In the motion planning problem, we're given the following:

- A robot with configuration space C
- A set of obstacles C_{obs}
- An initial configuration q_{init}
- Goal configuration q_{goal}.

We can break C into C_{free} and C_{obs}, or free space and obstacles. One way to discretize a free space is to decompose it into cells. Letting cells be nodes and letting adjacencies between cells be edges, we can then obtain a graph.

Visibility graphs are another way to discretize C. In a visibility graph, nodes are the vertices of the obstacles, and edges are paths between nodes that do not enter the interior of any obstacle, which are polygonal.

Lemma 3 *Consider a set S of disjoint polygonal obstacles, and start and goal positions q_{start} and q_{goal} respectively. Any shortest path between q_{start} and q_{goal} is a polygonal path where the inner vertices are vertices of S.*

You should know the geometric proof to this lemma, and be familiar with the general idea of path-finding through a free space with a visibility graph, as in Problem Set 1.

We also look at solving motion-planning problems through sampling-based methods, as such Probabilistic RoadMap (PRM) and Rapidly-exploring Random Trees (RRT).

Constraint Satisfaction Problems

A constraint satisfaction problem, or CSP, has 3 main components:

- Decision variables: $\{X_1, \ldots, X_n\}$ i.e. assignments that need to be made.
- Domains for each variable denoted: $\{D_1, \ldots, D_n\}$ i.e. plausible values for each decision variable.
- A set of constraints defined on subsets of variables i.e. equalities and inequalities that limit how we assign values to decision variables.

Types of Constraints

- Unary constraint (involving 1 decision variable) e.g. $X_1 = 3$
- Binary constraints (involving 2 decision variables) e.g. $X_1 + X_2 \leq 5$
- Other constraints may involve more than 2 decision variables e.g. $X_1 + X_2 - X_3 = 0$

Arc consistency is a way to reduce the choices we have for each decision variable by ruling out certain assignments that violate constraints.

- A variable X_i is arc-consistent with respect to variable X_j if for every value in D_i there exists a value in D_j that satisfies the binary constraint on (X_i, X_j).

- A CSP is arc-consistent if every variable is arc-consistent with respect to every other variable.
- Arc-consistency may sometimes not reduce variable domains at all. Sometimes, it may reduce domains to the extent that there is only 1 remaining value per domain, solving the CSP.

Backtracking search is a search algorithm used to solve CSPs by searching through combinations of possible assignments to the decision variables until it finds a solution that satisfies all constraints. Note that vanilla backtracking search does not necessarily utilize heuristics and forward-checking, but backtracking search can be combined with heuristics and inference (e.g. forward checking) to reduce the average search time required.

Convex Optimization

1. Convex Sets

Definition: A set $\mathcal{F} \subseteq \mathbb{R}^n$ is *convex* if for all $\mathbf{x}, \mathbf{y} \in \mathcal{F}$ and $\theta \in [0, 1]$, $\theta\mathbf{x} + (1-\theta)\mathbf{y} \in \mathcal{F}$. In other words, a set is convex if and only if all convex combinations of any two elements results in another element contained within the set.

Intuitively, for any two points in a convex set, any point on the line segment between them must also be in the set

What we have shown previously:

- The intersection of any collection convex sets is a convex set
- The union of convex sets is not necessarily convex
- A set is convex if and only if its intersection with any line is convex

2. Convex Functions

Definition: A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if and only if for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $\theta \in [0, 1]$,

$$f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y})$$

For functions f that are twice differentiable, $f''(x) \geq 0$ for all $x \in \mathbb{R}$

Examples:

- Exponential Function: $f(x) = e^{ax}$

$$f''(x) = a^2 e^{ax} \geq 0 \quad \text{for all } x \in \mathbb{R}.$$

- Euclidean Norm: $f(\mathbf{x}) = ||\mathbf{x}||_2 = \sqrt{\sum_{i=1}^{n}(x_i)^2}$.
- Weighted Sums of Convex Functions

3. Convex Optimization Problem

Definition: A *convex optimization problem* is a specialization of a general optimization problem $\min_{\mathbf{x}} f(\mathbf{x})$ such that $\mathbf{x} \in \mathcal{F}$ where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is a *convex function*, and the feasible region $\mathcal{F}$ is a *convex set*.

Examples

- Linear Programming: The linear programming problem can be formulated as finding

  $$\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x} \qquad \text{such that } A\mathbf{x} = \mathbf{a} \text{ and } B\mathbf{x} \leq \mathbf{b},$$

  where $\mathbf{x} \in \mathbb{R}^n$ is the optimization variable, and $\mathbf{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $\mathbf{a} \in \mathbb{R}^m$, $B \in \mathbb{R}^{k\times n}$, $\mathbf{b} \in \mathbb{R}^k$ are the problem data.

- Weber Point
- Max Flow

Integer Programming

Find $x_1, \ldots, x_l$ to maximize:

$$\max \sum_{j=1}^{l} c_j x_j$$

such that

$$\forall i \in [k], \sum_{j=1}^{l} a_{ij} x_j \leq b_i$$
$$\forall j \in [l], x_j \in \mathbb{Z}.$$

In other words, the Integer Programming Optimization Problem asks to find values for all x_i such that all k constraints are satisfied and the optimization function is maximized.

Integer programming is not convex because the possible solutions are discrete points, so the feasible set is not convex.

Game Theory

1. Normal Form Game

The normal form of a game is defined as:

- Set of players $N = \{1, \ldots, n\}$
- Strategy set S: The strategies that the players can take
- Utility function of player i: $u_i : S^n : R$, the utility player i will have when each $j \in N$ plays the strategy $s_j \in S$

Definition 4 *Dominance refers to a strategy $s = (s_1, \ldots, s_n)$ where each player i chooses a strategy s_i that maximizes their utility u_i regardless of the strategy s_j used by any other player $j \in N \setminus \{i\}$.*

2. Nash Equilibrium

Definition 5 *A vector of strategies $\boldsymbol{s} = (s_1, \ldots, s_n) \in S^n$ such that for all $i \in N$ and for all $s_i' \in S$ the following is true: $u_i(\boldsymbol{s}) \geq u_i(s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n)$*

In other words, in a Nash equilibrium, no player wants to unilaterally deviate from their current strategy.

3. Hotelling Model

Imagine that we can visualize political spectrum on the real line below.

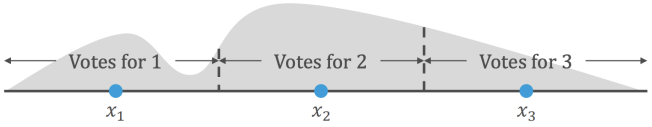


Figure 1: Political Spectrum

There is a nonatomic distribution of voters with each voter taking a point on the real line. Similarly, candidates are players who take positions on the spectrum, $x_1, x_2,$ and x_3 in the example above. Each candidate attracts voters with positions closest to them on the real line with ties being split equally.

Consider the case with two players/candidates x_1 and x_2:

1. Let their individual utilities be 1 if they win, 0 if they lose, and $\frac{1}{2}$ if they tie.
2. Let m represent the median *voter* position (for simplicity assume that it is unique.)

Depending on x_2's position, x_1's best position is as follows:

- If $x_2 < m$, the best response for player 1 is to take any position x_1 such that:

$$x_1 > x_2 \text{ and } \frac{x_1 + x_2}{2} < m$$

- Similarly, if $x_2 > m$, the best response for player 1 is to take any position x_1 such that:

$$x_1 < x_2 \text{ and } \frac{x_1 + x_2}{2} > m$$

- Finally, if $x_2 = m$, the best response for player 1 is to take any position x_1 such that:

$$x_1 = m$$

Therefore, the unique Nash equilibrium is at (m,m)

4. Mixed Strategies

A mixed strategy is a probability distribution over (pure) strategies.

- Distribution of strategies:

$$x_i(s_i) = P(i \text{ plays } s_i)$$

- Utility of player $i \in N$:

$$u_i(x_1, ..., x_n) = \sum_{(s_1,...,s_n) \in S^n} u_i(s_1, ..., s_n) \prod_{j=1}^{n} x_j(s_j)$$

  which is the expectation over the distribution of mixed strategies.

Practice Problems

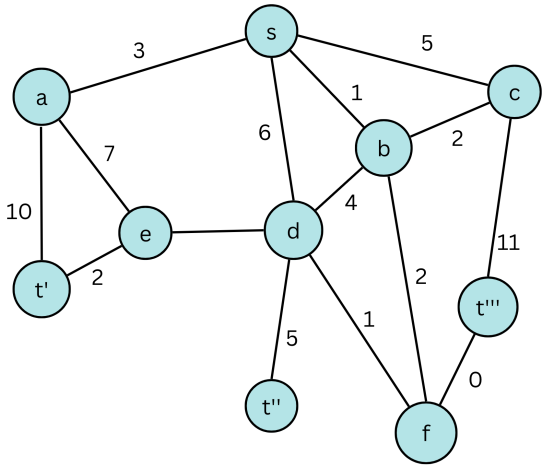
Search Problems

1. True of False

- Iterative deepening search is guaranteed to expand more nodes than BFS (on any graph whose root is not the goal)
- A* search with a heuristic that is not completely admissible may still find the shortest path to the goal state
- A* search with the heuristic $h(n) = 0$ is equivalent to the uniform-cost search
- If there are two sets of admissible heuristic values, h_1 $h2$, for the same search problem, then the heuristic $h_3 = (h_1 + h_2)/2$ is consistent and admissible.
- Greedy best-first search (sort queue by $h(n)$) is both complete and optimal when the heuristic is admissible and the path cost never decreases.
- For a search problem, the path returned by uniform cost search may change if we add a positive constant C to every step cost.

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* Tree Search

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.

   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
   - The number of disks NOT on peg
   - The weight of the disks NOT on peg
   - The weight of the disks NOT on peg 3 multiplied by their distances from peg 3.

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.
   - Value and variable ordering can decrease the number of expansions performed when using search to determine that a CSP has a solution.

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.

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?

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

   What values remain in each of the domains after enforcing arc-consistency?

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.

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

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

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

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