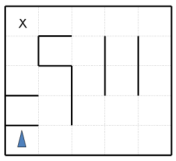
CS188 Fall 2018 Section 2: Graph Search + CSPs

1 Search and Heuristics

Imagine a car-like agent wishes to exit a maze like the one shown below:



The agent is directional and at all times faces some direction $d \in (N, S, E, W)$. With a single action, the agent can *either* move forward at an adjustable velocity v *or* turn. The turning actions are *left* and *right*, which change the agent's direction by 90 degrees. Turning is only permitted when the velocity is zero (and leaves it at zero). The moving actions are *fast* and *slow*. *Fast* increments the velocity by 1 and *slow* decrements the velocity by 1; in both cases the agent then moves a number of squares equal to its NEW adjusted velocity. Any action that would result in a collision with a wall crashes the agent and is illegal. Any action that would reduce v below 0 or above a maximum speed $V_{\max}$ is also illegal. The agent's goal is to find a plan which parks it (stationary) on the exit square using as few actions (time steps) as possible.

As an example: if the agent shown were initially stationary, it might first turn to the east using (*right*), then move one square east using *fast*, then two more squares east using *fast* again. The agent will of course have to *slow* to turn.

1. If the grid is M by N, what is the size of the state space? Justify your answer. You should assume that all configurations are reachable from the start state.

   The size of the state space is $4MN(V_{max} + 1)$. The state representation is (direction facing, x, y, speed). Note that the speed can take any value in $\{0, ..., V_{max}\}$.

2. Is the Manhattan distance from the agent's location to the exit's location admissible? Why or why not?

   No, Manhattan distance is not an admissible heuristic. The agent can move at an average speed of greater than 1 (by first speeding up to V_{max} and then slowing down to 0 as it reaches the goal), and so can reach the goal in less time steps than there are squares between it and the goal. A specific example: the target is 6 squares away, and the agent's velocity is already 4. By taking only 4 *slow* actions, it reaches the goal with a velocity of 0.

3. State and justify a non-trivial admissible heuristic for this problem which is not the Manhattan distance to the exit.

   There are many answers to this question. Here are a few, in order of weakest to strongest:

   (a) The number of turns required for the agent to face the goal.

   (b) Consider a relaxation of the problem where there are no walls, the agent can turn and change speed arbitrarily. In this relaxed problem, the agent would move with V_{max}, and then suddenly stop at the goal, thus taking $d_{manhattan}/V_{max}$ time.

   (c) We can improve the above relaxation by accounting for the deceleration dynamics. In this case the agent will have to slow down to 0 when it is about to reach the goal. Note that this heuristic will always return a greater value than the previous one, but is still not an overestimate of the true cost to reach the goal. We can say that this heuristic *dominates* the previous one.

4. If we used an inadmissible heuristic in A* graph search, would the search be complete? Would it be optimal?

   If the heuristic function is bounded, then A* graph search would visit all the nodes eventually, and would find a path to the goal state if there exists one. An inadmissible heuristic does not guarantee optimality as it can make the good optimal goal look as though it is very far off, and take you to a suboptimal goal.

5. If we used an *admissible* heuristic in A* graph search, is it guaranteed to return an optimal solution? What if the heuristic was consistent?

   Admissible heuristics do not necessarily guarantee optimality; they are only guaranteed to return an optimal solution if they are consistent as well.

6. Give a general advantage that an inadmissible heuristic might have over an admissible one.

   The time to solve an A* search problem is a function of two factors: the number of nodes expanded, and the time spent per node. An inadmissible heuristic may be faster to compute, leading to a solution that is obtained faster due to less time spent per node. It can also be a closer estimate to the actual cost function (even though at times it will overestimate!), thus expanding less nodes. We lose the guarantee of optimality by using an inadmissible heuristic. But sometimes we may be okay with finding a suboptimal solution to a search problem.

2 Course Scheduling

You are in charge of scheduling for computer science classes that meet Mondays, Wednesdays and Fridays. There are 5 classes that meet on these days and 3 professors who will be teaching these classes. You are constrained by the fact that each professor can only teach one class at a time.

The classes are:

1. Class 1 - Intro to Programming: meets from 8:00-9:00am
2. Class 2 - Intro to Artificial Intelligence: meets from 8:30-9:30am
3. Class 3 - Natural Language Processing: meets from 9:00-10:00am
4. Class 4 - Computer Vision: meets from 9:00-10:00am
5. Class 5 - Machine Learning: meets from 10:30-11:30am

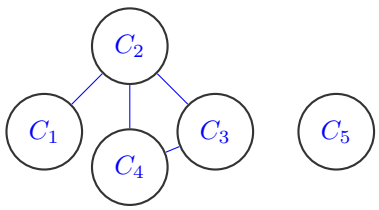
The professors are:

1. Professor A, who is qualified to teach Classes 1, 2, and 5.
2. Professor B, who is qualified to teach Classes 3, 4, and 5.
3. Professor C, who is qualified to teach Classes 1, 3, and 4.

1. Formulate this problem as a CSP problem in which there is one variable per class, stating the domains (after enforcing unary constraints), and binary constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit.

Variables	Domains (or unary constraints)	Binary Constraints
C_1	{A, C}	$C_1 \neq C_2$
C_2	{A}	$C_2 \neq C_3$
C_3	{B, C}	$C_2 \neq C_4$
C_4	{B, C}	$C_3 \neq C_4$
C_5	{A, B}	

2. Draw the constraint graph associated with your CSP.

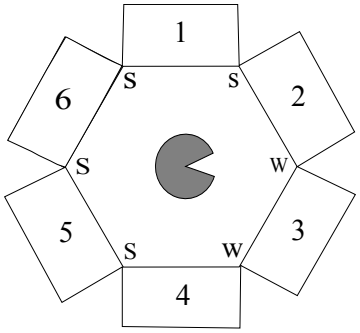


3 (Optional) CSPs: Trapped Pacman

Pacman is trapped! He is surrounded by mysterious corridors, each of which leads to either a pit (P), a ghost (G), or an exit (E). In order to escape, he needs to figure out which corridors, if any, lead to an exit and freedom, rather than the certain doom of a pit or a ghost.

The one sign of what lies behind the corridors is the wind: a pit produces a strong breeze (S) and an exit produces a weak breeze (W), while a ghost doesn't produce any breeze at all. Unfortuantely, Pacman cannot measure the the strength of the breeze at a specific corridor. Instead, he can stand *between* two adjacent corridors and feel the max of the two breezes. For example, if he stands between a pit and an exit he will sense a strong (S) breeze, while if he stands between an exit and a ghost, he will sense a weak (W) breeze. The measurements for all intersections are shown in the figure below.

Also, while the total number of exits might be zero, one, or more, Pacman knows that two neighboring squares will *not* both be exits.



Pacman models this problem using variables X_i for each corridor i and domains P, G, and E.

1. State the binary and/or unary constraints for this CSP (either implicitly or explicitly).

   Binary:

   $X_1 = P$ or $X_2 = P$, $X_2 = E$ or $X_3 = E$,
   $X_3 = E$ or $X_4 = E$, $X_4 = P$ or $X_5 = P$,
   $X_5 = P$ or $X_6 = P$, $X_1 = P$ or $X_6 = P$,
   $\forall i, j, Adj(i, j) and \neg(X_i = E and X_j = E)$

   Unary:

   $X_2 \neq P$,
   $X_3 \neq P$,
   $X_4 \neq P$

2. Cross out the values from the domains of the variables that will be deleted in enforcing arc consistency.

X_1	P		
X_2		G	E
X_3		G	E
X_4		G	E
X_5	P		
X_6	P	G	E

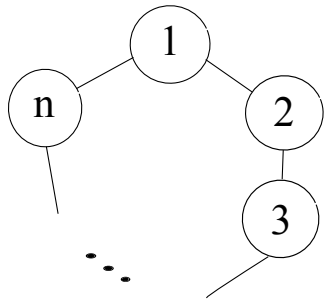
3. According to MRV, which variable or variables could the solver assign first?

   X_1 or X_5 (tie breaking)

4. Assume that Pacman knows that $X_6 = G$. List all the solutions of this CSP or write *none* if no solutions exist.

   (P,E,G,E,P,G)
   (P,G,E,G,P,G)

The CSP described above has a circular structure with 6 variables. Now consider a CSP forming a circular structure that has n variables ($n > 2$), as shown below. Also assume that the domain of each variable has cardinality d.



5. Explain precisely how to solve this general class of circle-structured CSPs efficiently (i.e. in time linear in the number of variables), using methods covered in class. Your answer should be at most two sentences.

   We fix X_j for some j and assign it a value from its domain (i.e. use cutset conditioning on one variable). The rest of the CSP now forms a tree structure, which can be efficiently solved without backtracking by enforcing arc consistency. We try all possible values for our selected variable X_j until we find a solution.

6. 2 If standard backtracking search were run on a circle-structured graph, enforcing arc consistency at every step, what, if anything, can be said about the worst-case backtracking behavior (e.g. number of times the search could backtrack)?

   A tree structured CSP can be solved without any backtracking. Thus, the above circle-structured CSP can be solved after backtracking at most d times, since we might have to try up to d values for X_j before finding a solution.