CS 188
Fall 2019

Exam Prep 2 Solutions

Note: we have provided more material than can be covered in discussion that can be used as practice at home.

Q1. Formulation: Holiday Shopping

You are programming a holiday shopping robot that will drive from store to store in order to buy all the gifts on your shopping list. You have a set of N gifts $G = \{g_1, g_2, \ldots g_N\}$ that must be purchased. There are M stores, $S = \{s_1, s_2, \ldots s_M\}$ each of which stocks a known inventory of items: we write $g_k \in s_i$ if store s_i stocks gift g_k. Shops may cover more than one gift on your list and will never be out of the items they stock. Your home is the store s_1, which stocks no items.

The actions you will consider are travel-and-buy actions in which the robot travels from its current location s_i to another store s_j in the fastest possible way and buys whatever items remaining on the shopping list that are sold at s_j. The time to travel-and-buy from s_i to s_j is $t(s_i, s_j)$. You may assume all travel-and-buy actions represent shortest paths, so there is no faster way to get between s_i and s_j via some other store. The robot begins at your home with no gifts purchased. You want it to buy all the items in as short a time as possible and return home.

For this planning problem, you use a state space where each state is a pair (s, u) where s is the current location and u is the set of unpurchased gifts on your list (so $g \in u$ indicates that gift g has not yet been purchased).

(a) How large is the state space in terms of the quantities defined above?

$M \times 2^N$. You are in one of M places (simple index from 1 to M), and have not purchased some subset of N items (binary vector of size N).

(b) For each of the following heuristics, which apply to states (s, u), circle whether it is admissible, consistent, neither, or both. Assume that the minimum of an empty set is zero.

([neither] / admissible / consistent / both)	The shortest time from the current location to any other store: $\min_{s' \neq s} t(s, s')$
(neither / admissible / consistent / [both])	The time to get home from the current location: $t(s, s_1)$
(neither / admissible / consistent / [both])	The shortest time to get to any store selling any unpurchased gift: $\min_{g \in u}(\min_{s': g \in s'} t(s, s'))$
(neither / [admissible] / consistent / both)	The shortest time to get home from any store selling any unpurchased gift: $\min_{g \in u}(\min_{s': g \in s'} t(s', s_1))$
([neither] / admissible / consistent / both)	The total time to get each unpurchased gift individually: $\sum_{g \in u}(\min_{s': g \in s'} t(s, s'))$
([neither] / admissible / consistent / both)	The number of unpurchased gifts times the shortest store-to-store time: $\lvert u \rvert(\min_{s_i, s_j \neq s_i} t(s_i, s_j))$

Remember, a consistent heuristic doesn't decrease from state to state by more than it actually costs to get from state to state. And of course, a heuristic is admissible if it is consistent. If you're confused, remember: the problem defines the minimum of an empty set as 0.

1. This heuristic does not return 0 in the goal state (s_1, $\emptyset$), since it gives the minimum distance to any store *other than the current one*.
2. We'll always need to get home from any state; the distance to home from home is 0; and this heuristic

does not decrease by more than it costs to get from state to state.

3. We'll always need to get that last unpurchased item, and taking the min distance store guarantees that we underestimate how much distance we actually have to travel. It is consistent because the heuristic never diminishes by more than what is travelled.
4. We'll always need to get home from getting the last unpurchased item, and taking the min underestimates the actual requirement. What makes this heuristic inconsistent is that when we visit the last store to pick up the last unfinished item, the value of the heuristic goes to 0. Let's say the graph looks like this: s_3 `--1-->` s_2 `----5---->` s_1, with s_2 containing the last item. From s_3, the heuristic is 5, but from s_2, the heuristic is now 0, meaning that traveling from s_3 to s_2 decreases the heuristic by 5 but the actual cost is only 1.
5. This can overestimate the actual amount of work required.
6. Same.

You have waited until very late to do your shopping, so you decide to send an swarm of R robot minions to shop in parallel. Each robot moves at the same speed, so the same store-to-store times apply. The problem is now to have all robots start at home, end at home, and for each item to have been bought by at least one robot (you don't have to worry about whether duplicates get bought). Hint: consider that robots may not all arrive at stores in sync.

(c) Give a minimal state space for this search problem (be formal and precise!)

We need the location of each robot at each time. At a given time, a robot can either be at one of M stores, or in any of $(T-1)M$ transition locations, where T is the maximum travel distance between two stores. Thus, the location of each robot takes $(MT)^R$. We also need the set of items purchased (2^N). Therefore, the size of each state is: $(MT)^R \times 2^N$.

One final task remains: you still must find your younger brother a stuffed Woozle, the hot new children's toy. Unfortunately, no store is guaranteed to stock one. Instead, each store s_i has an initial probability p_i of still having a Woozle available. Moreover, that probability drops exponentially as other buyers scoop them up, so after t time has passed, s_i's probability has dropped to $\beta^t p_i$. You cannot simply try a store repeatedly; once it is out of stock, that store will stay out of stock. Worse, you only have a single robot that can handle this kind of uncertainty! Phrase the problem as a single-agent MDP for planning a search policy for just this one gift (no shopping lists). You receive a single reward of +1 upon successfully buying a Woozle, at which point the MDP ends (don't worry about getting home); all other rewards are zeros. You may assume a discount of 1.

(d) Give a minimal state space for this MDP (be formal and precise!)

Which stores have been checked: 2^M
Whether Woozle has been bought: 2
Current time: T.

We may also want to keep track of the current location (M), but since there is no reward for traveling, we don't have to model that aspect of the problem.

Q2. CSPs: Properties

(a) 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.

True **False**

(b) 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? (circle one)

0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞

In general, the search might have to examine all possible assignments.

(c) 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? (circle one)

0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞

The MRV and LCV heuristics are often helpful to guide the search, but are not guaranteed to reduce backtracking in the worst case.

In fact, CSP solving is NP-complete, so any polynomial-time method for solving general CSPs would consititute a proof of $P = NP$ (worth a million dollars from the Clay Mathematics Institute!).

(d) What is the maximum number of times a backtracking search algorithm might have to backtrack in a *tree-structured* CSP, if it is running arc consistency and using an optimal variable ordering? (circle one)

0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞

Applying arc consistency to a tree-structured CSP guarantees that no backtracking is required, if variables are assigned starting at the root and moving down towards the leaves.

Q3. CSPs: Time Management

Two of our TAs, Arjun and Dave, are making their schedules for a busy morning. There are five tasks to be carried out:

(F) Pick up food for the group's research seminar, which, sadly, takes one precious hour.
(H) Prepare homework questions, which takes 2 consecutive hours.
(P) Prepare the PR2 (robot that Pieter uses for research) for a group of preschoolers' visit, which takes one hour.
(S) Lead the research seminar, which takes one hour.
(T) Teach the preschoolers about the PR2 robot, which takes 2 consecutive hours.

The schedule consists of one-hour slots: 8am-9am, 9am-10am, 10am-11am, 11am-12pm. The requirements for the schedule are as follows:

1. In any given time slot each TA can do at most one task (F, H, P, S, T).
2. The PR2 preparation (P) should happen before teaching the preschoolers (T).
3. The food should be picked up (F) before the seminar (S).
4. The seminar (S) should be finished by 10am.
5. Arjun is going to deal with food pick up (F) since he has a car.
6. The TA not leading the seminar (S) should still attend, and hence cannot perform another task (F, T, P, H) during the seminar.
7. The seminar (S) leader does not teach the preschoolers (T).
8. The TA who teaches the preschoolers (T) must also prepare the PR2 robot (P).
9. Preparing homework questions (H) takes 2 consecutive hours, and hence should start at or before 10am.
10. Teaching the preschoolers (T) takes 2 consecutive hours, and hence should start at or before 10am.

To formalize this problem as a CSP, use the variables F, H, P, S and T. The values they take on indicate the TA responsible for it, and the starting time slot during which the task is carried out (for a task that spans 2 hours, the variable represents the starting time, but keep in mind that the TA will be occupied for the next hour also - make sure you enforce constraint (a)!). Hence there are eight possible values for each variable, which we will denote by A8, A9, A10, A11, D8, D9, D10, D11, where the letter corresponds to the TA and the number corresponds to the time slot. For example, assigning the value of A8 to a variables means that this task is carried about by Arjun from 8am to 9am.

(a) What is the size of the state space for this CSP?
8^5.

(b) Which of the statements above include unary constraints?
(d), (e), (i), (j). (i) and (j) are both unary constraints, and binary constraints in a single sentence.

(c) In the table below, enforce all unary constraints by crossing out values in the table on the left below. If you made a mistake, cross out the whole table and use the right one.

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

(d) Start from the table above, select the variable S and assign the value A9 to it. Perform forward checking by crossing out values in the table below. Again the table on the right is for you to use in case you believe you made a mistake.

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

(e) Based on the result of (d), what variable will we choose to assign next based on the MRV heuristic (breaking ties alphabetically)? Assign the first possible value to this variable, and perform forward checking by crossing out values in the table below. Again the table on the right is for you to use in case you believe you made a mistake.

Variable F is selected and gets assigned value A8.

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

Have we arrived at a dead end (i.e., has any of the domains become empty)?

No.

(f) We return to the result from enforcing just the unary constraints, which we did in (c). Select the variable S and assign the value A9. Enforce arc consistency by crossing out values in the table below.

F	A8	A9	A10	A11	D8	D9	D10	D11
H	A8	A9	A10	A11	D8	D9	D10	D11
P	A8	A9	A10	A11	D8	D9	D10	D11
S	A8	A9	A10	A11	D8	D9	D10	D11
T	A8	A9	A10	A11	D8	D9	D10	D11

(g) Compare your answers to (d) and to (f). Does arc consistency remove more values or less values than forward checking does? Explain why.

Arc consistency removes more values. It's because AC checks consistency between any pair of variables, while FC only checks the relationship between pairs of assigned and unassigned variables.

(h) Check your answer to (f). Without backtracking, does any solution exist along this path? Provide the solution(s) or state that there is none.

AC along this path gives 1 solution: F: A8 H: A10 P: D8 S: A9 T: D10