CS 188
Fall 2020

# Regular Discussion 2 Solutions

## CSPs

CSPs are defined by three factors:

1. *Variables* - CSPs possess a set of $N$ variables $X_1, ..., X_N$ that can each take on a single value from some defined set of values.
2. *Domain* - A set $\{x_1, ..., x_d\}$ representing all possible values that a CSP variable can take on.
3. *Constraints* - Constraints define restrictions on the values of variables, potentially with regard to other variables.

CSPs are often represented as constraint graphs, where nodes represent variables and edges represent constraints between them.

- *Unary Constraints* - Unary constraints involve a single variable in the CSP. They are not represented in constraint graphs, instead simply being used to prune the domain of the variable they constrain when necessary.
- *Binary Constraints* - Binary constraints involve two variables. They're represented in constraint graphs as traditional graph edges.
- *Higher-order Constraints* - Constraints involving three or more variables can also be represented with edges in a CSP graph.

In **forward checking**, whenever a value is assigned to a variable $X_i$, forward checking prunes the domains of unassigned variables that share a constraint with $X_i$ that would violate the constraint if assigned. The idea of forward checking can be generalized into the principle of **arc consistency**. For arc consistency, we interpret each undirected edge of the constraint graph for a CSP as two directed edges pointing in opposite directions. Each of these directed edges is called an **arc**. The arc consistency algorithm works as follows:

- Begin by storing all arcs in the constraint graph for the CSP in a queue $Q$.
- Iteratively remove arcs from $Q$ and enforce the condition that in each removed arc $X_i \longrightarrow X_j$, for every remaining value $v$ for the tail variable $X_i$, there is at least one remaining value $w$ for the head variable $X_j$ such that $X_i = v, X_j = w$ does not violate any constraints. If some value $v$ for $X_i$ would not work with any of the remaining values for $X_j$, we remove $v$ from the set of possible values for $X_i$.
- If at least one value is removed for $X_i$ when enforcing arc consistency for an arc $X_i \longrightarrow X_j$, add arcs of the form $X_k \longrightarrow X_i$ to $Q$, for all unassigned variables $X_k$. If an arc $X_k \longrightarrow X_i$ is already in $Q$ during this step, it doesn't need to be added again.
- Continue until $Q$ is empty, or the domain of some variable is empty and triggers a backtrack.

We've delineated that when solving a CSP, we fix some ordering for both the variables and values involved. In practice, it's often much more effective to compute the next variable and corresponding value "on the fly" with two broad principles, **minimum remaining values** and **least constraining value**:

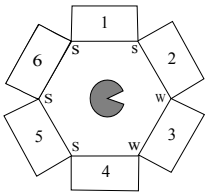
- *Minimum Remaining Values (MRV)* - When selecting which variable to assign next, using an MRV policy chooses whichever unassigned variable has the fewest valid remaining values (the *most constrained variable*).
- *Least Constraining Value (LCV)* - Similarly, when selecting which value to assign next, a good policy to implement is to select the value that prunes the fewest values from the domains of the remaining unassigned values.

# 1 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. Unfortunately, Pacman cannot measure 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.

**(a)** 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 $X_6 = P$,
$\forall i, j$ s.t. Adj$(i, j)$ $\neg(X_i = E$ and $X_j = E)$

Unary:

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

Note: This is just one of many solutions. The answers below will be based on this formulation.

**(b)** Suppose we assign $X_1$ to $E$. Perform forward checking after this assignment. Also, enforce unary constraints.

| | | | |
|---|---|---|---|
| $X_1$ | | | E |
| $X_2$ | | | |
| $X_3$ | | G | E |
| $X_4$ | | G | E |
| $X_5$ | P | G | E |
| $X_6$ | P | | |

**(c)** Suppose forward checking returns the following set of possible assignments:

| | | | |
|---|---|---|---|
| $X_1$ | P | | |
| $X_2$ | | G | E |
| $X_3$ | | G | E |
| $X_4$ | | G | E |
| $X_5$ | P | | |
| $X_6$ | P | G | E |

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

$X_1$ or $X_5$ (tie breaking)

**(d)** 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)

# 2 CSP: Air Traffic Control

We have five planes: A, B, C, D, and E and two runways: international and domestic. We would like to schedule a time slot and runway for each aircraft to **either** land or take off. We have four time slots: $\{1, 2, 3, 4\}$ for each runway, during which we can schedule a landing or take off of a plane. We must find an assignment that meets the following constraints:

- Plane B has lost an engine and must land in time slot 1.
- Plane D can only arrive at the airport to land during or after time slot 3.
- Plane A is running low on fuel but can last until at most time slot 2.
- Plane D must land before plane C takes off, because some passengers must transfer from D to C.
- No two aircrafts can reserve the same time slot for the same runway.

**(a)** Complete the formulation of this problem as a CSP in terms of variables, domains, and constraints (both unary and binary). Constraints should be expressed implicitly using mathematical or logical notation rather than with words.

**Variables:** A, B, C, D, E for each plane.

**Domains:** a tuple *(runway type, time slot)* for runway type $\in \{international, domestic\}$ and time slot $\in \{1, 2, 3, 4\}$.

**Constraints:**

$$B[1] = 1 \qquad A[1] \leq 2$$
$$D[1] \geq 3 \qquad D[1] < C[1]$$
$$A \neq B \neq C \neq D \neq E$$

**(b)** For the following subparts, we add the following two constraints:

- Planes A, B, and C cater to international flights and can only use the international runway.
- Planes D and E cater to domestic flights and can only use the domestic runway.

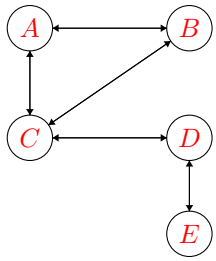
**(i)** With the addition of the two constraints above, we completely reformulate the CSP. You are given the variables and domains of the new formulation. Complete the constraint graph for this problem given the original constraints and the two added ones.

**Variables:** A, B, C, D, E for each plane.

**Domains:** $\{1, 2, 3, 4\}$

**Explanation of Constraint Graph:**
We can now encode the runway information into the identity of the variable, since each runway has more than enough time slots for the planes it serves. We represent the non-colliding time slot constraint as a binary constraint between the planes that use the same runways.

**Constraint Graph:**




**(ii)** What are the domains of the variables after enforcing arc-consistency? Begin by enforcing unary constraints. (Cross out values that are no longer in the domain.)

| | | | | |
|---|---|---|---|---|
| A | 1 | 2 | 3 | 4 |
| B | 1 | 2 | 3 | 4 |
| C | 1 | 2 | 3 | 4 |
| D | 1 | 2 | 3 | 4 |
| E | 1 | 2 | 3 | 4 |

**(iii)** Arc-consistency can be rather expensive to enforce, and we believe that we can obtain faster solutions using only **forward-checking** on our variable assignments. Using the Minimum Remaining Values heuristic, perform backtracking search on the graph, breaking ties by picking lower values and characters first. List the $(variable, assignment)$ pairs in the order they occur (including the assignments that are reverted upon reaching a dead end). Enforce unary constraints before starting the search.

(You don't have to use this table, it won't be graded.)

| | | | | |
|---|---|---|---|---|
| A | 1 | 2 | 3 | 4 |
| B | 1 | 2 | 3 | 4 |
| C | 1 | 2 | 3 | 4 |
| D | 1 | 2 | 3 | 4 |
| E | 1 | 2 | 3 | 4 |

**Answer:** (B, 1), (A, 2), (C, 3), (C, 4), (D, 3), (E, 1)