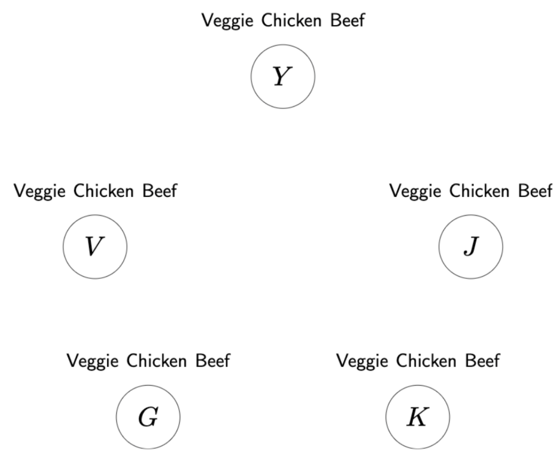
CS221 Problem Workout Solutions

Week 6

1) [CA session] Problem 1

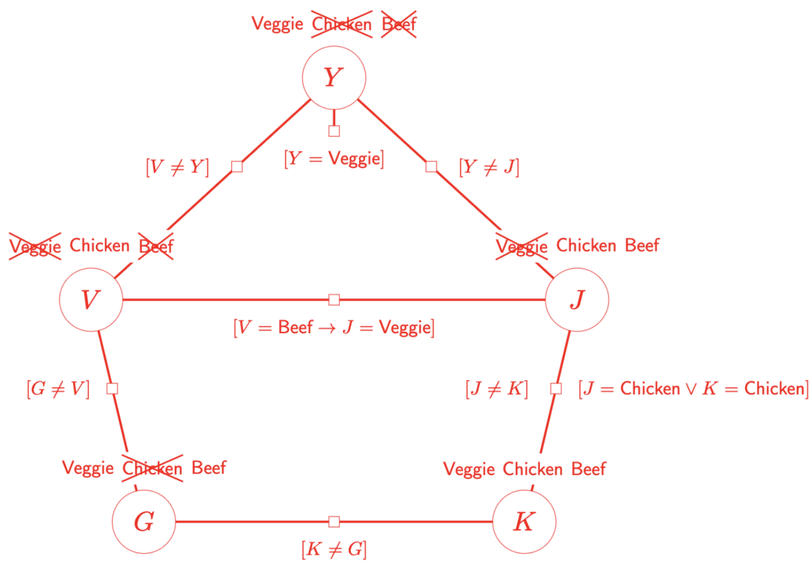
(a) You and your friends (Veronica, Jarvis, Gabriela, Kanti) sit around a table like this: There are three dishes on the menu: the vegetarian deep dish pizza, the



chicken quesadilla, and the beef cheeseburger. Each person will order exactly one dish. But what started out as a simple dinner has quickly turned into a logistical nightmare because of all the constraints you and your friends impose upon yourselves:

i. Each person must order something different than the people sitting immediately next to them.
ii. You (Y) are vegetarian.
iii. If Veronica (V) orders beef, then Jarvis (J) will order veggie.
iv. Kanti (K) and Jarvis (J) cannot both get non-chicken dishes.

Draw the potentials for the above constraints and write the propositional formula above each potential (e.g., $[Y = \text{Veggie}]$). Then for each pair of variables, enforce arc consistency in both directions, crossing out the appropriate values from the domains.

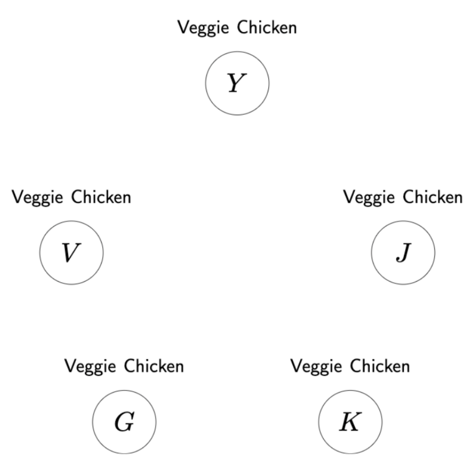


(b) Your server comes by your table and tells you that they are out of beef today, so you all decide to rework your constraints. Now they are:

i. There is a preference for people sitting next to each other to order different dishes. Formally, we have 5 potentials: $f(Y, J) = \mathbf{1}[Y \neq J] + 1$, $f(J, K) = \mathbf{1}[J \neq K] + 1$, etc.

ii. You (Y) are vegetarian.

With the 2 constraints above, what is a maximum weight assignment and what is its weight? If there are many assignments with the same maximum weight, give any one. For convenience, the updated table is given below.



Solution The maximum weight 16. Many answers can be correct. They just need to satisfy the following: 1. Y=Veggie; 2. Only one pair of adjacent people have the same dish. One max weight assignment is: Y=Veggie, V=Chicken, G=Veggie, K=Chicken, and J=Chicken

2) **[Breakout] Problem 2** Which of the following scenarios can be modeled as a two-player zero-sum game? Select all that apply.

(a) **Confession or Silence**: Two prisoners are interrogates separately. if both prisoners confess, both are sentences to 5 years; If neither of them confess. both are sentenced to 1 year; If one prisoner confesses and the other does not, the the one that confesses is left free, while the other is sentenced to 10 years.

(b) **Game of Go**: Two players compete against each other in a five-round Go match. Whoever wins at least three rounds wins the match.

(c) **Dividing the Cake**: Alice and Bob are sharing one cake. Alice first divides the cake into two pieces, then Bob chooses one piece and leaves Alice with the other piece. Both of them want to maximize their share of cake.

(d) **Stock Market**: Parag and Elon each own 100 shares of Twitter stock. Parag first decides whether to buy or sell 1 share of his stock, then Elon decides whether to buy or sell 1 share of his stock. Both of them want to maximize their financial profit from the transaction.

Solution (B) (C)

3) **[Breakouts] Problem 3**

Optimal Driving Corporation (ODC) has hired you as a consultant to help design optimal algorithms for their new fleet of self-driving cars. They are currently focused on a particular stretch of freeway with H lanes and length W (see figure below). There are $K+1$ cars on the road: Your car (the one being controlled algorithmically) starts at position $x_0^{(0)} = (1,1)$, and the K other cars start at positions $x_1^{(0)}, \dots, x_K^{(0)}$.

Each car (yours and others) has the same set of four possible actions (the successor states below are for the car in $(5,2)$ from the figure below):

- $+(1,0)$: Move forward one step (ending up in $(6,2)$)
- $+(2,0)$: Move forward two steps (ending up in $(7,2)$)
- $+(1,-1)$: Move forward one step and to the left (ending up in $(6,1)$)
- $+(1,1)$: Move forward one step and to the right (ending up in $(6,3)$)

A car cannot take an action that takes it out of one of the H lanes; for example, a car in $(1,3)$ cannot take action $+(1,1)$. If a car leaves this stretch of freeway (e.g., taking action $+(1,0)$ from $(7,2)$), then the car is teleported to a special end position $\odot$. Once in $\odot$, any action a car takes will keep it in $\odot$ and incur 0 cost. All cars take turns moving, starting with your car, followed by cars 1 through K, and then your car, etc. The game ends when all cars have left the stretch of freeway (moved into position $\odot$). Each action (except for those taken from position $\odot$) has a base cost of 1. There are also a set of potholes P. If at the end of an action, a car lands on a pothole or in a position occupied by another car, it incurs an additional cost of c (and does not affect any other cars). Each car's goal is to minimize its own total cost. For example, if your car takes actions that lead to the following state sequence: $(1,1) \to (2,2) \to (4,2) \to (6,2) \to (7,3)$, the cost incurred by you is $1+1+(1+c)+1 = 4+c$. Note that cost is only incurred based on where a car lands, so the cost of $(2,2) \to (4,2)$ above is 1, even though there is a pothole at $(3,2)$.

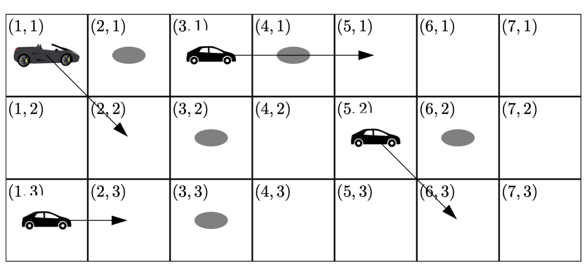


Figure 1: An example of the driving scenario, where we have a freeway of length $W = 7$ and $H = 3$ lanes. Your car starts at $x_0^{(0)} = (1,1)$ and the other $K = 3$ cars start in $x_1^{(0)} = (1,3)$, $x_2^{(0)} = (3,1)$, and $x_3^{(0)} = (5,2)$. The potholes $P = \{(2,1),(3,2),(3,3),(4,1),(6,2)\}$ are shown as gray ellipses. The arrows show one possible action that each of the cars can take.

In practice, we don't know what policies the other cars could be facing. For each setting, specify what is the most specific type of model that could be used to represent it (e.g., search problem, MDP, two-player zero-sum game, CSP, etc.). For example, if something is a search problem, don't say MDP, even though all search problems are special cases of MDPs. What inference algorithm would you use for the model (e.g., uniform cost search, minimax, etc.)? If we have not covered inference algorithms for a particular model, say so. Simply state the model and inference algorithm, and briefly justify your answer.

(a) Each other car's policy chooses the action a that minimizes its immediate cost (not your cost) plus the distance from $x_i + a$ to (W, H). Any ties are broken randomly.

Solution The greedy policy is a known stochastic policy. Due to the randomness, it is not a search problem, but can be cast as an MDP. The default algorithm for computing optimal policies is value iteration. But because the MDP is acyclic, we could also compute this using a recursive dynamic programming.

(b) Each other car's policy is optimally minimizing your cost (which might be the case if you had a siren on your car).

Solution All cars are now trying to minimize your cost, so this is a search problem. Which can be solved using UCS or A* (all costs are non-negative), or dynamic programming (since the state graph is acyclic).

(c) Each other car's policy is optimally maximizing your cost.

Solution This is a classic turn-based zero-sum game (very similar to Pac-Man with cars instead of ghosts). We would compute the recurrence using dynamic programming. Minimax is another option.

(d) Each other car's policy is trying to minimize its own cost.

Solution This is a turn-based non-zero-sum game. These games in general don't have optimal policies, but merely Nash equilibria. How to compute them is outside the scope of this class, so any answer for the inference algorithm is okay. One such algorithm to solve for the Nash equilibria is the Lemke-Howson algorithm.

4) **[CA session] Problem 4** Farmer Kim wants to install a set of sprinklers to water all his crops in the most cost-effective manner and has hired you as a consultant. Specifically, he has a rectangular plot of land, which is broken into $W \times H$ cells. For each cell (i, j), let $C_{i,j} \in \{0, 1\}$ denote whether there are crops in that cell that need watering. In each cell (i, j), he can either install ($X_{i,j} = 1$) or not install ($X_{i,j} = 0$) a sprinkler. Each sprinkler has a range of R, which means that any cell within Manhattan distance of R gets watered. The maintenance cost of the sprinklers is the sum of the Manhattan distances from each sprinkler to his home located at $(1, 1)$. Recall that the Manhattan distance between (a_1, b_1) and (a_2, b_2) is $|a_1 - a_2| + |b_1 - b_2|$. Naturally, Farmer Kim wants the maintenance cost to be as small as possible given that all crops are watered. See figure below for an example.

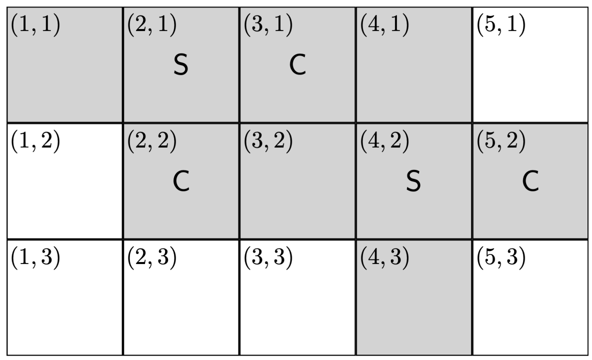


Figure 2: An example of a farm with $W = 5$ and $H = 3$. Each cell (i, j) is marked 'C' if there are crops there that need watering ($C_{i,j} = 1$). An example of a sprinkler installation is given: a cell (i, j) is marked with 'S' if we are placing a sprinkler there ($X_{i,j} = 1$). Here, the sprinkler range is $R = 1$, and the cells that are shaded are the ones covered by some sprinkler. In this case, the sprinkler installation is valid (all crops are watered), and the total maintenance cost is $1 + 4 = 5$.

Farmer Kim actually took CS221 years ago, and remembered a few things. He says: "I think this should be formulated as a factor graph. The variables should be $X_{i,j} \in \{0, 1\}$ for each cell (i, j). But here's where my memory gets foggy. What should the factors be?" Let $X = \{X_{i,j}\}$ denote a full assignment to all variables $X_{i,j}$. Your job is to define two types of factors:

- $f_{i,j}$: ensures any crops in (i, j) are watered,
- f_{cost}: encodes the maintenance cost,

so that a maximum weight assignment corresponds to a valid sprinkler installation with minimum maintenance cost.

$f_{i,j}(X) =$

Solution For each cell (i, j), let $f_{i,j}$ encode whether the crops (if they exist) in (i, j) are watered:

$$f_{i,j}(X) = \left[C_{i,j} = 0 \text{ or } \left(\min_{i',j':X_{i',j'}=1} |i' - i| + |j' - j| \right) \leq R \right]. \tag{1}$$

The first part encodes there being no crops in cell (i, j). The second part encodes that if there is a crop, the closest sprinkler should be at most R Manhattan distance away.

$f_{\text{cost}}(X) =$

Solution We define the next factor to the exponentiated negative minimum cost, so that the factor is non-negative and that maximizing the weight corresponds to minimizing the maintenance cost:

$$f_{\text{cost}}(X) = \exp\left(- \sum_{i',j':X_{i',j'}=1} |i' - 1| + |j' - 1| \right). \tag{2}$$

Any answer where the factor is non-negative and maximizing the weight corresponds to minimizing the maintenance cost is accepted. The answer should also account for the case where there are no sprinklers. Below is another acceptable answer:

$$f_{\text{cost}}(X) = \begin{cases} 1 & \text{if } X_{i',j'} = 0 \text{ for all } i', j' \\ \left(\sum_{i',j':X_{i',j'}=1} |i' - 1| + |j' - 1| \right)^{-1} & \text{otherwise} \end{cases}$$