University of Waterloo
School of Computer Science
CS 486/686, SAMPLE Midterm Examination
Winter 2024

Name:

Waterloo Student ID:

- **Instructor: Jesse Hoey & Josh Jung**
- **Date: Any day before Feb 9th 630pm**
- **Location: Anywhere you want**
- **Time: Anytime before Feb 9th, 630pm**
- **There are 4 questions on this exam**
- **There are 12 pages in this exam (including cover page and two additional pages at the end for work that does not fit in the spaces provided)**
- **There are a total of 100 marks on the exam**
- **You have 90 minutes to complete the exam**
- **ONLY non-programmable calculators are allowed**

Question	1	2	3	4	TOTAL
Marks	20	20	30	30	100
Score					

Question 1: Short Answers (20 marks out of a total of 100 marks)

WRITE ANSWERS IN THE SPACES PROVIDED AFTER EACH QUESTION

(1a) [**4 marks**] Complete the following sentence: A* search uses a priority queue of nodes ranked by ______________, and so is a mixture of ______________ and best-first search.

> **SOLUTION**:
> cost to the node + heuristic from node to goal, lowest-cost-first

(1b) [**4 marks**] A heuristic for a search problem is an estimate of the true cost to the goal. Is the following statement true or false: *an admissible heuristic is always greater than the true cost.* Briefly explain

> **SOLUTION**:
> false. It must be less than or equal to the true cost

(1c) [**4 marks**] True or False: in a deterministic planner, causal rules specify when a feature keeps its value when not acted upon. Briefly explain why.

> **SOLUTION**:
> False. Causal rules specify when things change as a result of an action.

(1d) [**4 marks**] Why is variable elimination for constraint satisfaction problems intractable?

> **SOLUTION**:
> It is hard to find the optimal elimination ordering.

(1e) [**4 marks**] Give one reason why Heuristic Depth-First Search is often used in practice.

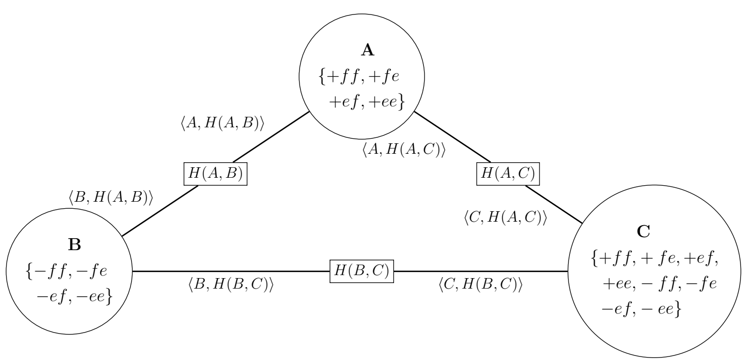
> **SOLUTION**:
> Space complexity

Question 2: Constraint Satisfaction (30 marks marks out of a total of 100 marks)

The KoobeCaf™social network consists of a number of *members* who can be connected (or not) to each other. Connected members on KoobeCaf™can be either both *friends* (f) or *enemies* (e) with each other. Members come in two types: positive (+) and negative (−). While connected members of different types *must* be enemies, members of the same type can be friends or enemies. A *stable* social network is one in which all members satisfy these constraints. This can be represented as a constraint satisfaction problem (CSP) as follows:

- **variables and domains**: Each KoobeCaf™member is represented with a tuple containing: their type (+ or −) and whether they are friends (f) or enemies (e) with each of their (up to $N-1$) connections. Thus, a member b of a network with three fully connected members a, b, c (in that order), would be represented with a variable $B = T_b X_a X_c$ where $T_b \in \{+, -\}$ is b's type and $X_a, X_c \in \{f, e\}$ are b's relationship with a and c, respectively. Thus, B has domain $\{+ff, +fe, +ef, +ee, -ff, -fe, -ef, -ee\}$, where $-fe$ means b is type negative, friends with a, but enemies with c. The order of the friend/enemy relations in each tuple is alphabetic.
- **constraints**: A binary constraint between each pair of connected members that requires the two members to (1) have the same relation (e.g. in the 3-member network A,B,C above, if $A = {*}{*}f$ then $C = {*}f{*}$ where $*$ means either value); and (2) be of the same type OR be enemies. Call this constraint $H(\cdot,\cdot)$. For example, with three fully connected members represented by variables A, B, C: $H(A,C) = True$ for $\{A, C\} \in \{\{+{*}f, +f{*}\}, \{+{*}e, +e{*}\}, \{-{*}f, -f{*}\}, \{-{*}e, -e{*}\}, \{+{*}e, -e{*}\}, \{-{*}e, +e{*}\}\}$.

This graph shows a KoobeCaf™network with three members: a, b and c, in which a is positive (+), and b is negative (−) and domain values of the wrong type have been removed from the domains of A and B.



(2a) [**25 marks**] Using AC-3, make the CSP shown above arc-consistent by filling in the table on the facing page, in which each iteration is on a single line in the table and a check-mark is under each constraint that is consistent (i.e. the arc is **not** in the

(Question 3 CONTINUES ON NEXT PAGE...)

TDA queue), and the domain values remaining for each variable are shown under each variable name. Always choose the left-most (in the table) inconsistent constraint to make consistent next. You may not need all rows, but you will not need more rows. Don't add redundant arcs back on the TDA (after changing the domain of X for the arc $\langle X, c(X,Y)\rangle$, you only add arcs $\langle Z, c'(Z,X)\rangle$ to TDA where $\mathbf{Z} \neq \mathbf{Y}$).

$\mathbf{A} = T_a X_b X_c$ $T_a = +$ $X_b, X_c \in \{f, e\}$	$\langle A, H(A,B)\rangle$	$\langle B, H(A,B)\rangle$	$\mathbf{B} = T_b X_a X_c$ $T_b = -$ $X_a, X_c \in \{f, e\}$	$\langle B, H(B,C)\rangle$	$\langle C, H(B,C)\rangle$	$\mathbf{C} = T_c X_a X_b$ $T_c \in \{+, -\}$ $X_a, X_b \in \{f, e\}$	$\langle C, H(A,C)\rangle$	$\langle A, H(A,C)\rangle$
$+ff$ $+fe$ $+ef$ $+ee$			$-ff$ $-fe$ $-ef$ $-ee$			$+ff$ $+fe$ $+ef$ $+ee$ $-ff$ $-fe$ $-ef$ $-ee$		
$+ef$ $+ee$	✓							

SOLUTION:

10 things change (shown in green) - 2.5 marks per item changed

A	$\langle A, H(A,B)\rangle$	$\langle B, H(A,B)\rangle$	**B**	$\langle B, H(B,C)\rangle$	$\langle C, H(B,C)\rangle$	**C**	$\langle C, H(A,C)\rangle$	$\langle A, H(A,C)\rangle$
$+ff$ $+fe$ $+ef$ $+ee$			$-ff$ $-fe$ $-ef$ $-ee$			$+ff$ $+fe$ $+ef$ $+ee$ $-ff$ $-fe$ $-ef$ $-ee$		
$+ef$ $+ee$	$\checkmark$							
	$\checkmark$	$\checkmark$	$-ef$ $-ee$					
	$\checkmark$	$\checkmark$		$\checkmark$				
	$\checkmark$	$\checkmark$		$\checkmark$	$\checkmark$	$+fe$ $+ee$ $-ff$ $-fe$ $-ef$ $-ee$		
	$\checkmark$	$\checkmark$			$\checkmark$	$+fe$ $+ee$ $-ef$ $-ee$	$\checkmark$	
	$\checkmark$	$\checkmark$		$\checkmark$	$\checkmark$		$\checkmark$	
	$\checkmark$	$\checkmark$		$\checkmark$	$\checkmark$		$\checkmark$	$\checkmark$

(2b) [**5 marks**] Based only on your arc-consistent network (last line of the table), can you guarantee that there is a solution (a stable network)? Briefly explain why or why not.

SOLUTION:
No. You can't tell because it might not be globally consistent.

Question 3: Search (30 marks marks out of a total of 100 marks)

WRITE ANSWERS IN THE SPACES PROVIDED AFTER EACH QUESTION

An artificial intelligence conference has $N_c + N_s$ attendees, of which N_c do research on connectionist AI ("connectionists") and N_s do research on symbolic AI "symbolicists". On their way to the conference reception, they come across a river with no bridge that they must cross. There is a boat that can carry only 2 people at a time. If more connectionists than symbolicists are left on either side of the river, the connectionists will start a fight with the symbolicists. At least one person must be in the boat on each trip (to paddle the boat). The goal is to move everyone to the other side of the river in the shortest number of trips without a fight starting. Assume $N_c < N_s$ to start with.

The state space can be easily described by the number of connectionists and symbolicists on each side, plus the location of the boat. The cost function, $g(n)$ is simply 1 for each trip across the river. A simple heuristic $h(n)$ for this problem is to relax it by ignoring fights that break out.

(3a) [**10 marks**] Give a formal (mathematical) definition of a state, neighborhood function, cost function and the heuristic function described above.

SOLUTION:
There could be many ways to do this but a tuple $n = \langle N_s, N_c, N_s', N_c', B\rangle$ where N_s, N_c is the number of symbolicists and connectionists on the arrival side and N_s', N_c' is the number of symbolicists and connectionists on the destination side, and $B \in \{a, d\}$ is the location of the boat. There is a neighbour of a node for each possible transfer of people from B to the opposite side ($B = a \to B = d$ or $B = d \to B = a$), where "possible" means one of 1 or 2 connectionists, 1 or 2 symbolicists, or 1 of each. We will say a node has no neighbours (it is a terminal state) if $N_c > N_s$ or $N_c' > N_s'$. The cost function is 1 for each edge and the heuristic is different for each value of B as follows:

($B = a$): $h(n) = (N_c + N_s - 1) \times 2 - 1 = (N_c + N_s) \times 2 - 3$ since it will take this many trips to get all $N_c + N_s$ people across: 2 people per trip from $a \to d$ and 1 per trip from $d \to a$. The $(N_C + N_s - 1)$ factor is because the last trip moves 2 people across the river. The -1 factor is because on the last trip, you don't need to send the boat back.

($B = d$): $h(n) = (N_c + N_s - 1 + 1) \times 2 - 1 + 1 = (N_c + N_s) \times 2$ since the boat has to first carry one person back and then we are in the (B=a) situation again but with one more person on side a.

These two can be combined into a single equation

$$h(n) = (N_c + N_s - 1\delta(B, a)) \times 2 - 1\delta(B, a) = (N_c + N_s) \times 2 - 3\delta(B, a)$$

where

$$\delta(B, a) = \begin{cases} 1 & \text{if} \quad B = a \\ 0 & \text{if} \quad B = d \end{cases}$$

explanation: The quickest way without worrying about fights is to take 2 people over and one comes back. So every 2 trips, one person gets transferred. Therefore you might think its N*2 if N is the number of people in total. However, the last trip over transfers two people completely, so really one person gets carried "for free" so its (N-1)*2. Further, the last person doesn't have to go back at the end, so its (N-1)*2-1

the simple way to think about the destination side heuristic is that you have to bring one person back to the arrival side, take one extra trip, and then you have N+1 persons on the arrival side and can use that heuristic, so its

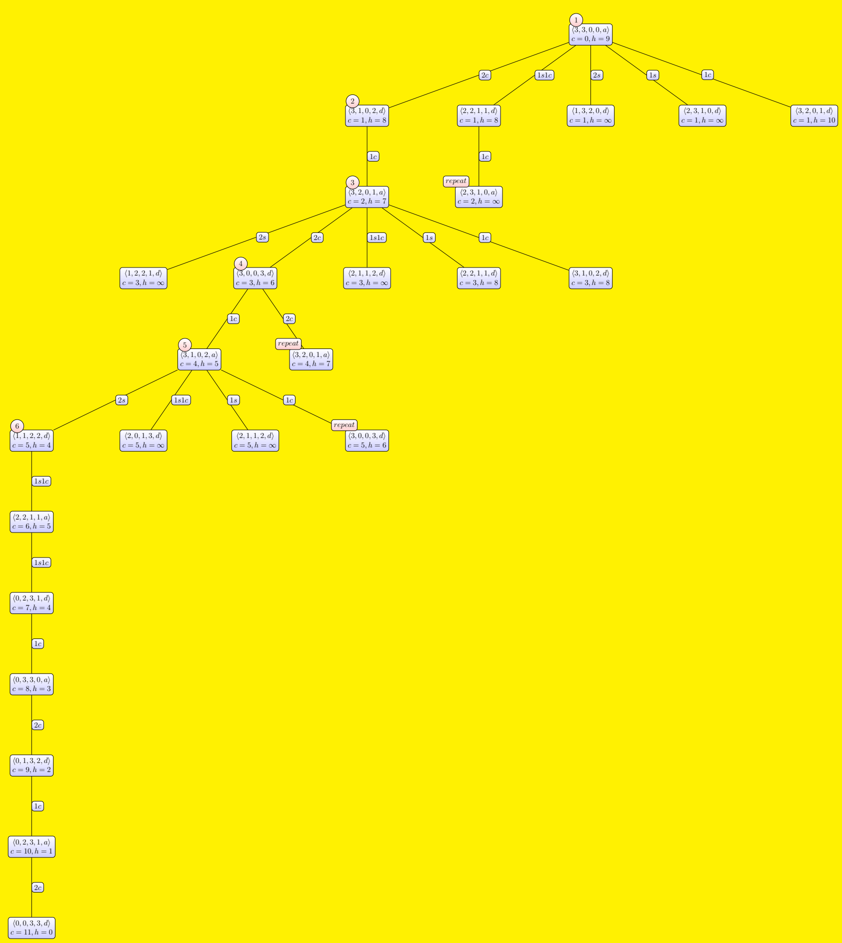
(N+1-1)*2 - 1 +1 = 2N

(3b) [**5 marks**] Is the $h(n)$ defined above admissible? Carefully explain why or why not.

SOLUTION:
It is admissible since the boat can't move unless there is one person in it, and carries at most 2 people, so the maximum transfer rate is 1 person per round trip.

(4c) [**15 marks**] Show part of the search graph that results from applying algorithm A* for the problem with $N_c = N_s = 3$ starting from the configuration where everyone is on one side, with the heuristic $h(n)$ and cost $g(n)$ defined above. Specifically, continue until you have expanded (generated the successors of) four nodes. Break ties arbitrarily. Number the four nodes you expand to indicate the order in which they are expanded ("1","2","3","4"). Then, number (with "5") the next leaf that you would expand if you were to continue with A* search on this graph. Label each node with its $g(n)$ value and its $h(n)$ value. You should label all nodes, not just the ones you expand. Do not add a node to the tree if it is already in the tree somewhere with a lower or equal value of $f(n) = g(n) + h(n)$. Branches of your search graph should terminate (have an infinite heuristic value) if the connectionists outnumber the symbolicists on either side of the river.

SOLUTION:
In this graph, I show a couple of repeated nodes which are optional and I expand the last node all in a single path from the node marked (6) only to show the solution



Question 4: Decision Trees (20 marks out of a total of 100 marks)

You are trying to build a system to predict whether a stock price will rise or fall, using the following dataset of historical measurements of four attributes: the `season`, whether an `election` is taking place, whether the price of oil (`oil_price`) has risen or fallen recently whether the price of the stock (`stock_price`) in question rose or fell.

`election`	`season`	`oil_price`	`stock_price`
no	winter	rise	rise
no	summer	fall	fall
no	summer	rise	rise
yes	winter	rise	fall
no	winter	fall	fall
yes	summer	rise	rise
yes	summer	fall	fall
yes	winter	fall	fall

(4a) [**15 marks**] Using information gain (weighted by fractions of documents) as your feature selection mechanism, construct a decision tree that predicts if the stock price will rise or fall based on the attributes above. Continue splitting until no more attributes are available, or until all data agrees on the target attribute (stock price change). Show your work and draw a graph of your final decision tree.

SOLUTION:

(i) Round 1: Initial information is 3/8 rise, 5/8 fall so $I(3/8, 5/8) = 0.95$

- election has 4/4 split yes/no. For election=no, 2/2 split so I(0.5,0.5)=1, for election=yes, 3/1 split, so $I(1/4, 3/4) = 0.8$, so the $I(split) = 0.5 * 1 + 0.5 * 0.8 = 0.9$
- season is the same as election
- oil price has 4/4 split. For oil price=rise, 3/1 split so I=0.8, for oil-price=afll, 4/0 split so I=0, so $I(split) = 0.8 * 0.5 = 0.4$ and this has maximum information gain.

(ii) Round 2: for oil-price=fall, stock-price is always fall, so this is a leaf. For oil-price=rise,

- election has 2/2 split, for election=no, I(0,1)=0, for election=yes, I(0.5,0.5)=1, so I(split)=0.5
- season is the same I=0 for season=summer so I(split)=0.5

(iii) Round 3: which ever one is left over perfectly predicts the data so I(split)=0

(4b) [**3 marks**] What prediction would your decision tree give for the following test data

election	season	oil_price	stock_price
no	summer	fall	
no	winter	rise	
yes	summer	rise	

SOLUTION:
a=fall, b=rise, c=rise

(4c) [**2 marks**] Suppose now you received the following data points and added them to the data in Q(1), and re-learned the decision tree using information gain to split.

election	season	oil_price	stock_price
yes	summer	rise	fall
yes	winter	rise	rise

Would your decision tree change? If so, what would the change be? Would your classification of the data in (b) change? If so, show the new classification. If not, explain why not.

SOLUTION:
If season was chosen first in (ii) above, then, the tree would change since election will now be chosen first. Otherwise, only the leaves will change. Now the prediciton for (a) and (b) would be the same, but for (c) would be a random choice between rise/fall, or a probability of 0.5 rise/fall.

ADDITIONAL WORK - Clearly state which question you are working on

ADDITIONAL WORK - Clearly state which question you are working on