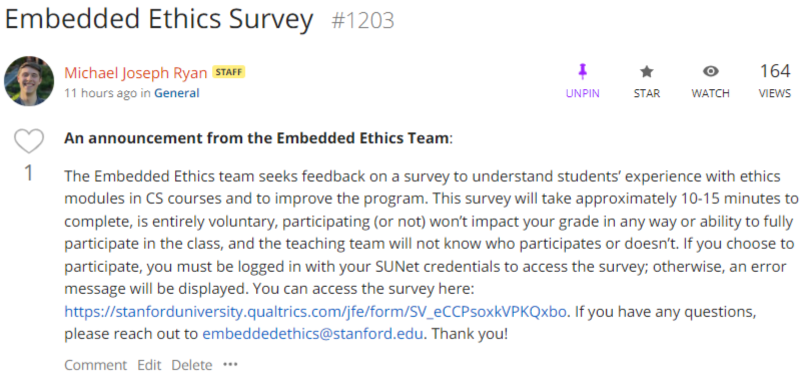
Final Exam Review

Final Exam Review

Embedded Ethics Evaluation Survey

Embedded Ethics Evaluation Survey



Embedded Ethics Survey #1203

Michael Joseph Ryan STAFF
11 hours ago in General

An announcement from the Embedded Ethics Team:

The Embedded Ethics team seeks feedback on a survey to understand students' experience with ethics modules in CS courses and to improve the program. This survey will take approximately 10-15 minutes to complete, is entirely voluntary, participating (or not) won't impact your grade in any way or ability to fully participate in the class, and the teaching team will not know who participates or doesn't. If you choose to participate, you must be logged in with your SUNet credentials to access the survey; otherwise, an error message will be displayed. You can access the survey here: https://stanforduniversity.qualtrics.com/jfe/form/SV_eCCPsoxkVPKQxbo. If you have any questions, please reach out to embeddedethics@stanford.edu. Thank you!

Please take a few minutes to check the pinned post on Ed about the Ethics Survey!

Final Exam Review

Course Modules

Course Modules

- Factor Graphs (Constraint Satisfaction Problems or CSPs)
    - Factors, Satisfiability, Backtracking Search (MCV/LCV), Forward Checking, Arc Consistency, Beam Search, Local Search (ICM)
- Markov Nets
    - Factor Graphs, Assignment Weights, Joint Probability, Marginal Probability, Gibbs Sampling
- Bayesian Nets
    - Probabilistic Inference, HMMs (Forward/Backward), Particle Filtering, Supervised Learning (Maximum Likelihood), Laplace Smoothing, EM
- Logic
    - Natural Language $\iff$ Logic Expressions, Knowledge Base (Entailment, Contradiction, Contingency), Completeness, Soundness

Unlike past final exams, there will be no Ethics problem this time.

Today's Review

Today, we'll cover:

- Aut 2022 Final's CSP Problem
- Winter 2021 Final's Bayesian Net Problem

We will give the problems extra adjustments to cover more concepts from their respective topic (including Markov Nets).

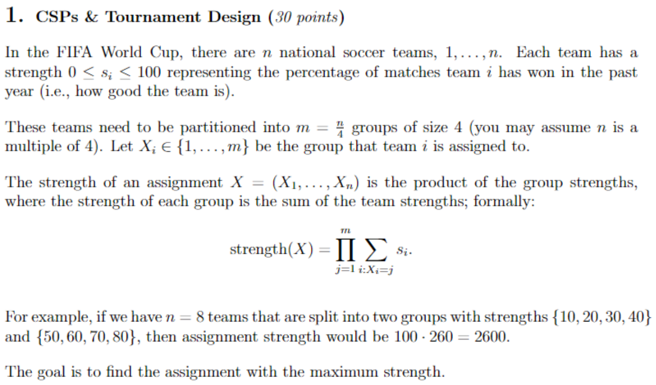
For reviewing Logic:

- Be ready to translate between natural language and logic expressions like you did on your HW!
- Know how to fact check when adding new information to a knowledge base causes (i) entailment vs (ii) contradiction vs (iii) contingency (see Problem Session 9)!
- Understand the difference between completeness & soundness.

Final Exam Review

Constraint Satisfaction Problem (CSP) Review

Breaking down the Problem


1. CSPs & Tournament Design (*30 points*)

In the FIFA World Cup, there are n national soccer teams, $1, \ldots, n$. Each team has a strength $0 \le s_i \le 100$ representing the percentage of matches team i has won in the past year (i.e., how good the team is).

These teams need to be partitioned into $m = \frac{n}{4}$ groups of size 4 (you may assume n is a multiple of 4). Let $X_i \in \{1, \ldots, m\}$ be the group that team i is assigned to.

The strength of an assignment $X = (X_1, \ldots, X_n)$ is the product of the group strengths, where the strength of each group is the sum of the team strengths; formally:

$$\text{strength}(X) = \prod_{j=1}^{m} \sum_{i:X_i=j} s_i.$$

For example, if we have $n = 8$ teams that are split into two groups with strengths $\{10, 20, 30, 40\}$ and $\{50, 60, 70, 80\}$, then assignment strength would be $100 \cdot 260 = 2600$.

The goal is to find the assignment with the maximum strength.

- n teams indexed by i, each with strength $0 \le s_i \le 100$
- Must split teams into groups of 4 for $m = \frac{n}{4}$ total groups, with $X_i = \{1, \ldots, m\}$ being the group that we assign team i
- Strength of an assignment $X = (X_1, \ldots, X_n)$ is $\prod_j^m \sum_{i:X_i=j} s_i$

Defining the Factors

Breaking down the problem:

- n teams indexed by i, each with strength $0 \leq s_i \leq 100$
- Must split teams into groups of 4 for $m = \frac{n}{4}$ total groups, with $X_i = \{1, \ldots, m\}$ being the group that we assign team i
- Strength of an assignment $X = (X_1, \ldots, X_n)$ is $\prod_j^m \sum_{i:X_i=j} s_i$

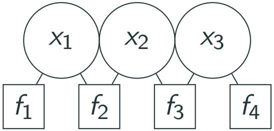
We see that

- The 1st piece of info just specifies variable definitions.
- The 2nd piece of info asks for a boolean constraint or factor.
- The 3rd piece of info asks for a weight factor.

Defining the Factors

Factor Graph

- Variables: $X = (X_1, \dots, X_n)$ where $X_i \in \text{Domain}_i$
- Factors: $f_1, \dots, f_m$, with each $f_j(X) \geq 0$.



- Constraints: Factors that return 0 or 1.

Assignment Weight: Every assignment $x = (x_1, \dots, x_n)$ has

$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

- Consistent if $\text{Weight}(x) > 0$.
- A CSP is satisfiable if $\max_x \text{Weight}(x) > 0$.

Defining the Factors

Breaking down the problem:

- Must split teams into groups of 4 for $m = \frac{n}{4}$ total groups, with $X_i = \{1, \ldots, m\}$ being the group that we assign team i.

We see that

- This piece of info asks for a boolean constraint or factor.

From the previous slide: Every assignment $x = (x_1, \ldots, x_n)$ has

$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

The purpose of a boolean factor is to make the weight 0 if the constraint is violated (evaluates to false) based on an assignment, as an assignment is only consistent if $\text{Weight}(x) > 0$.

Defining the Factors

Breaking down the problem:

- Must split teams into groups of 4 for $m = \frac{n}{4}$ total groups, with $X_i = \{1, \ldots, m\}$ being the group that we assign team i.

We see that

- This piece of info asks for a boolean constraint or factor.

For boolean factors, the indicator function is your best friend!

Define a factor for each group j that checks whether the size of the group is 4, returning 1 or 0 via the indicator function:

$$f_j(X_1, \ldots, X_n) = \mathbb{1}[|\{i : X_i = j\}| = 4]$$

We have m constraint factors in all, each one takes every variable (each representing a team) as arguments, builds a set of all teams assigned to group j, and checks if the set size is 4.

Defining the Factors

Breaking down the problem:

- Must split teams into groups of 4 for $m = \frac{n}{4}$ total groups, with $X_i = \{1, \ldots, m\}$ being the group that we assign team i.

We see that

- This piece of info asks for a boolean constraint or factor.

Define a factor for each group j that checks whether the size of the group is 4, returning 1 or 0 via the indicator function:

$$f_j(X_1, \ldots, X_n) = \mathbb{1}[|\{i : X_i = j\}| = 4]$$

Another way to write this:

$$f_j(X_1, \ldots, X_n) = \mathbb{1}\left[\left(\sum_i^n \mathbb{1}[X_i = j]\right) = 4\right]$$

Both would be accepted answers!

Defining the Factors

Breaking down the problem:

- Strength of an assignment $X = (X_1, \ldots, X_n)$ is $\prod_j^m \sum_{i:X_i=j} s_i$

We see that

- This piece of info asks for a weight factor.

From the previous slide: Every assignment $x = (x_1, \ldots, x_n)$ has

$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

The purpose of a weight factor is to measure how "good an assignment is" so that we can compare assignments when looking for the maximum weight assignment.

Defining the Factors

Breaking down the problem:

- Strength of an assignment $X = (X_1, \ldots, X_n)$ is $\prod_j^m \sum_{i:X_i=j} s_i$

We see that

- This piece of info asks for a weight factor.

From the previous slide: Every assignment $x = (x_1, \ldots, x_n)$ has

$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

We can take advantage of the above definition to write:

$$h_j(X_1, \ldots, X_n) = \sum_{i:X_i=j}^{n} s_i$$

One factor for each group j, summing up the strengths of each team i within the group.

Defining the Factors

Every assignment $x = (x_1, \ldots, x_n)$ has

$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

The purpose of a weight factor is to measure how "good an assignment is" so that we can compare assignments when looking for the maximum weight assignment.

Food for thought: How would you encode factors that you want to minimize?

Suppose instead of measuring each team's strengths, we measure their weaknesses w_i. Then, define the weakness of an assignment $X = (X_1, \ldots, X_n)$ as $\prod_j^m \sum_{i:X_i=j} w_i$.

Write a factor that helps minimize the weakness of an assignment.

Defining the Factors

Food for thought: How would you encode factors that you want to minimize?

Suppose instead of measuring each team's strengths, we measure their weaknesses w_i. Then, define the weakness of an assignment $X = (X_1, \ldots, X_n)$ as $\prod_j^m \sum_{i:X_i=j} w_i$.

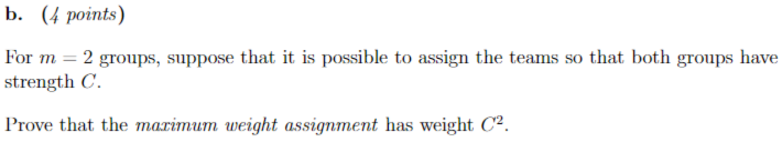
Write a factor that helps minimize the weakness of an assignment.

Use the inverse function!

$$h_j(X_1, \ldots, X_n) = \left(\sum_{i:X_i=j}^{n} w_i \right)^{-1}$$

One factor for each group j, summing up the weakness of each team i within the group and then inverting.

Continuing the Problem


b. (*4 points*)

For $m = 2$ groups, suppose that it is possible to assign the teams so that both groups have strength C.

Prove that the *maximum weight assignment* has weight C^2.

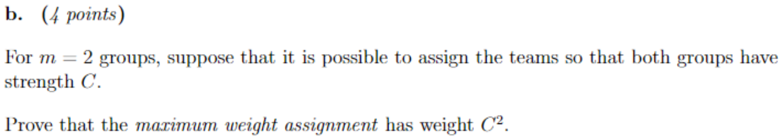
More of a math trick question; need to realize:

- The team strengths all add up to $2C$.
- No matter how you split the teams into 2 groups, you can represent the group strengths as $C - \delta$ and $C + \delta$.
- The total strength of a group is encoded in our factor:

$$h_j(X_1, \ldots, X_n) = \sum_{i:X_i=j}^{n} s_i$$

so we have $h_1 = C - \delta$ and $h_2 = C + \delta$.

Continuing the Problem


b. (*4 points*)

For $m = 2$ groups, suppose that it is possible to assign the teams so that both groups have strength C.

Prove that the *maximum weight assignment* has weight C^2.

- The total strength of a group is encoded in our factor:

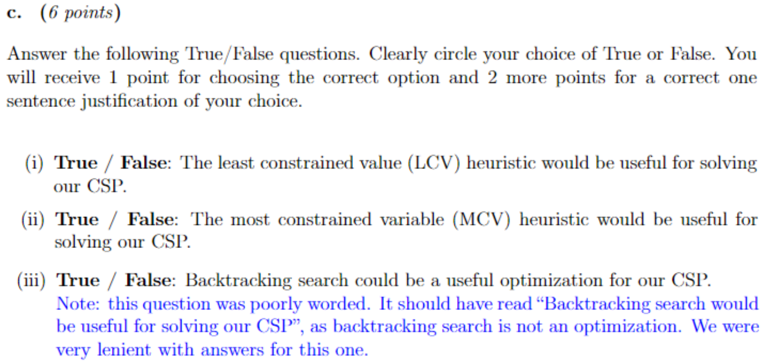
$$h_j(X_1, \ldots, X_n) = \sum_{i:X_i=j}^{n} s_i$$

  so we have $h_1 = C - \delta$ and $h_2 = C + \delta$.
- Weight of an assignment is simply the product of all factors. Our constraint factors always evaluate to 0 or 1, so any non-zero assignment weight is the result of:

$$h_1 h_2 = (C - \delta)(C + \delta) = C^2 - \delta^2$$

  which is always at most C^2.

Dynamic Ordering for Backtracking Search



c. (*6 points*)

Answer the following True/False questions. Clearly circle your choice of True or False. You will receive 1 point for choosing the correct option and 2 more points for a correct one sentence justification of your choice.

(i) **True** / **False**: The least constrained value (LCV) heuristic would be useful for solving our CSP.

(ii) **True** / **False**: The most constrained variable (MCV) heuristic would be useful for solving our CSP.

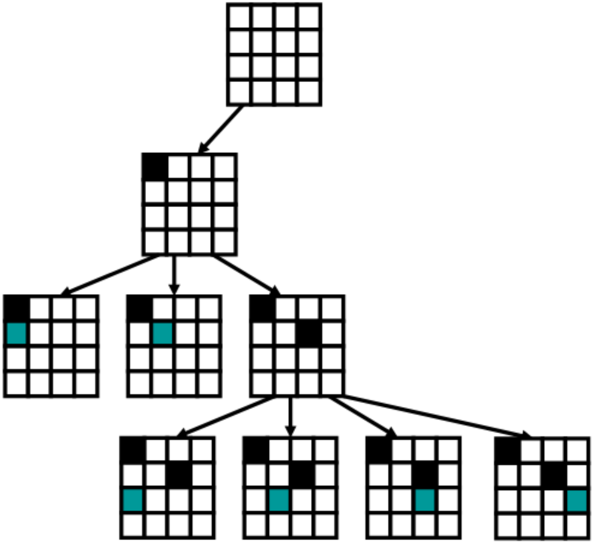
(iii) **True** / **False**: Backtracking search could be a useful optimization for our CSP. Note: this question was poorly worded. It should have read "Backtracking search would be useful for solving our CSP", as backtracking search is not an optimization. We were very lenient with answers for this one.

Recall backtracking search is essentially a brute-force algorithm that eventually finds the optimal (maximum weight) assignment for a CSP (see Problem Session 6 for an example + forward checking).

LCV and MCV are heuristics that speed up backtracking search.

Dynamic Ordering for Backtracking Search

Quick refresher of backtracking search for the 4-Queens problem:



Dynamic Ordering for Backtracking Search

LCV and MCV are heuristics that speed up backtracking search.

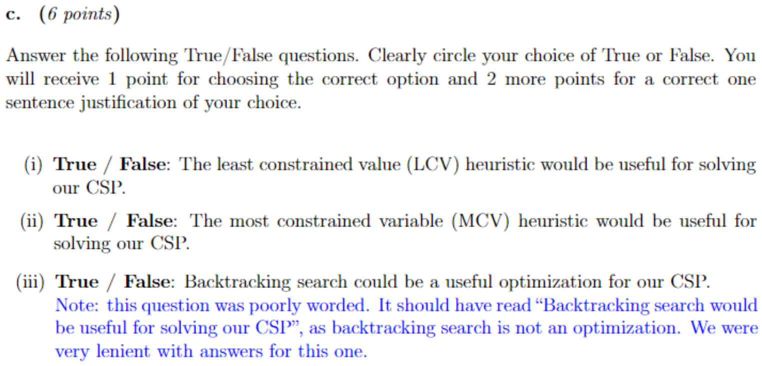
Whenever backtracking search is about to choose the next variable to assign:

- MCV chooses the variable with the smallest domain (smallest number of values that we can assign to it).

Whenever backtracking search is about to decide what value to assign a chosen variable:

- LCV assigns the value that reduces the number of values that we can assign to the other variables the least.
- Arbitrary Example: Suppose assigning $X_1 = 1$ reduces the domain of X_2 to $\{1, 2, 3\}$, but assigning $X_1 = 2$ reduces the domain of X_2 to $\{1, 2, 3, 4\}$; then LCV tries $X_1 = 2$ first.

Dynamic Ordering for Backtracking Search



c. (*6 points*)

Answer the following True/False questions. Clearly circle your choice of True or False. You will receive 1 point for choosing the correct option and 2 more points for a correct one sentence justification of your choice.

(i) **True** / **False**: The least constrained value (LCV) heuristic would be useful for solving our CSP.

(ii) **True** / **False**: The most constrained variable (MCV) heuristic would be useful for solving our CSP.

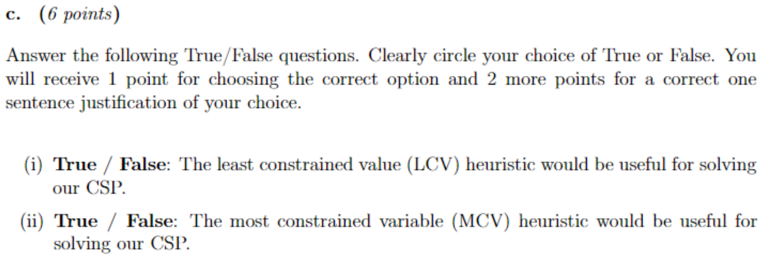
(iii) **True** / **False**: Backtracking search could be a useful optimization for our CSP. Note: this question was poorly worded. It should have read "Backtracking search would be useful for solving our CSP", as backtracking search is not an optimization. We were very lenient with answers for this one.

Back to the problem:

- Backtracking search can always be used to find the optimal assignment. There are no problems here with using it.

The other two parts aren't the best written...

Dynamic Ordering for Backtracking Search



c. (*6 points*)

Answer the following True/False questions. Clearly circle your choice of True or False. You will receive 1 point for choosing the correct option and 2 more points for a correct one sentence justification of your choice.

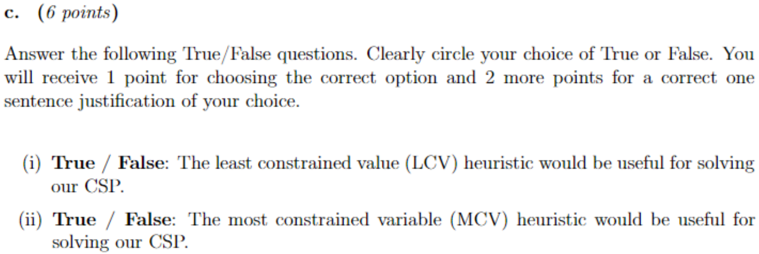
(i) **True** / **False**: The least constrained value (LCV) heuristic would be useful for solving our CSP.

(ii) **True** / **False**: The most constrained variable (MCV) heuristic would be useful for solving our CSP.

What this problem was based on are the following facts:

- LCV does not make sense if not all factors are constraints, because we'll want to try all value assignments for a variable eventually to determine the maximum weight assignment.
- MCV works as long as there are some constraint factors that help prune away the domain of variables.

Dynamic Ordering for Backtracking Search



c. (*6 points*)

Answer the following True/False questions. Clearly circle your choice of True or False. You will receive 1 point for choosing the correct option and 2 more points for a correct one sentence justification of your choice.

(i) **True** / **False**: The least constrained value (LCV) heuristic would be useful for solving our CSP.

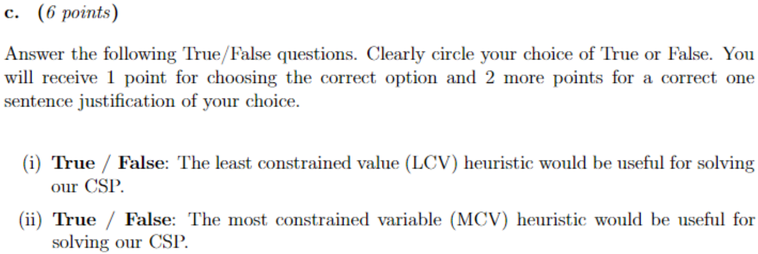
(ii) **True** / **False**: The most constrained variable (MCV) heuristic would be useful for solving our CSP.

For this problem, not all our factors are constraints, so LCV cannot be used, but MCV can be used (technically).

However... MCV wouldn't really give us a speed up here because our constraint factor doesn't actually prune domains:

$$f_j(X_1, \dots, X_n) = \mathbb{1}\left[\left(\sum_i^n \mathbb{1}[X_i = j]\right) = 4\right]$$

Dynamic Ordering for Backtracking Search



c. (*6 points*)

Answer the following True/False questions. Clearly circle your choice of True or False. You will receive 1 point for choosing the correct option and 2 more points for a correct one sentence justification of your choice.

(i) **True** / **False**: The least constrained value (LCV) heuristic would be useful for solving our CSP.

(ii) **True** / **False**: The most constrained variable (MCV) heuristic would be useful for solving our CSP.

Recall our only constraint factor:

$$f_j(X_1, \dots, X_n) = \mathbb{1}\left[\left(\sum_i^n \mathbb{1}[X_i = j]\right) = 4\right]$$

At any given point in our backtracking search, we can always assign a team i to any of the remaining groups with fewer than 4 teams. There are no constraints that make one team more constrained than another in what we can assign.

Dynamic Ordering for Backtracking Search

Recall our only constraint factor:

$$f_j(X_1, \ldots, X_n) = \mathbb{1}\left[\left(\sum_i^n \mathbb{1}[X_i = j]\right) = 4\right]$$

At any given point in our backtracking search, we can always assign a team i to any of the remaining groups with fewer than 4 teams. There are no constraints that make one team more constrained than another in what we can assign.

Food for thought: Come up with a new constraint for a hypothetical scenario that would make MCV more relevant.

Dynamic Ordering for Backtracking Search

Recall our only constraint factor:

$$f_j(X_1, \ldots, X_n) = \mathbb{1}\left[\left(\sum_i^n \mathbb{1}[X_i = j]\right) = 4\right]$$

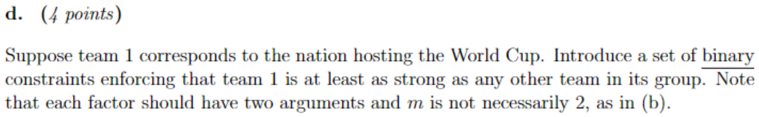
At any given point in our backtracking search, we can always assign a team i to any of the remaining groups with fewer than 4 teams. There are no constraints that make one team more constrained than another in what we can assign.

Food for thought: Come up with a new constraint for a hypothetical scenario that would make MCV more relevant.

Let each group have a pre-assigned leader, and only teams with lower strength than the leader of a group can be put in that group.

Essentially, MCV is useful when there are conflict in assignments!

Continuing the Problem



d. (*4 points*)

Suppose team 1 corresponds to the nation hosting the World Cup. Introduce a set of binary constraints enforcing that team 1 is at least as strong as any other team in its group. Note that each factor should have two arguments and m is not necessarily 2, as in (b).

This is essentially a logic exercise. First, translate the constraint:

- For each team i, that team is either not in the same group as Team 1 OR has a strength less than or equal to s_i.

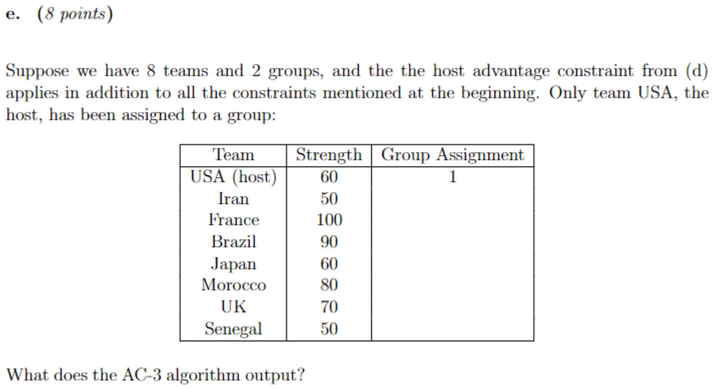
This naturally leads to the expression:

$$X_i \neq X_1 \vee s_1 \geq s_i$$

which can be expressed as a factor via the indicator function:

$$f_i(X_1, X_i) = \mathbb{1}[X_i \neq X_1 \vee s_1 \geq s_i]$$

Arc Consistency



e. (*8 points*)

Suppose we have 8 teams and 2 groups, and the the host advantage constraint from (d) applies in addition to all the constraints mentioned at the beginning. Only team USA, the host, has been assigned to a group:

Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	
France	100	
Brazil	90	
Japan	60	
Morocco	80	
UK	70	
Senegal	50	

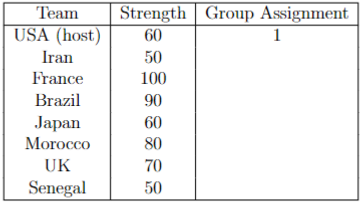
What does the AC-3 algorithm output?

Arc Consistency: A variable X_i is arc consistent with respect to X_j if for each $x_i \in \text{Domain}_i$, there exists $x_j \in \text{Domain}_j$ such that

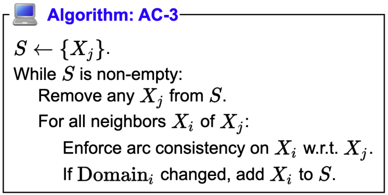
$$f(X_i = x_i, X_j = x_j) \neq 0$$

for all f whose scope contains X_i and X_j.

Arc Consistency



Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	
France	100	
Brazil	90	
Japan	60	
Morocco	80	
UK	70	
Senegal	50	



Algorithm: AC-3

```
$S \leftarrow \{X_j\}$.
While $S$ is non-empty:
    Remove any $X_j$ from $S$.
    For all neighbors $X_i$ of $X_j$:
        Enforce arc consistency on $X_i$ w.r.t. $X_j$.
        If $\mathrm{Domain}_i$ changed, add $X_i$ to $S$.
```

Before AC-3, all teams besides the USA have domains: $\{1, 2\}$.

On the 1st pass of AC-3:

- We examine X_j = USA.
- Iran, Japan, & Senegal all still have domains of $\{1, 2\}$.
- France, Brazil, Morocco, & UK all have their domains reduced to $\{2\}$ because of the Part d constraint.
- Domains of France, Brazil, Morocco, and UK changed, thus we put them back into the set S to examine next.

Arc Consistency

Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	
France	100	
Brazil	90	
Japan	60	
Morocco	80	
UK	70	
Senegal	50	

Algorithm: AC-3

```
$S \leftarrow \{X_j\}$.
While $S$ is non-empty:
    Remove any $X_j$ from $S$.
    For all neighbors $X_i$ of $X_j$:
        Enforce arc consistency on $X_i$ w.r.t. $X_j$.
        If $\mathrm{Domain}_i$ changed, add $X_i$ to $S$.
```

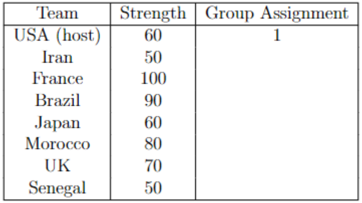
On the 2nd pass of AC-3:

- We examine $X_j = \text{France}$.
- Teams Iran, Japan, and Senegal all have their domains reduced to $\{1\}$ because of our original constraint:

$$f_j(X_1, \ldots, X_n) = \mathbb{1}\left[\left(\sum_i^n \mathbb{1}[X_i = j]\right) = 4\right]$$

- Further passes of AC-3 produce no more changes.

Arc Consistency



Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	
France	100	
Brazil	90	
Japan	60	
Morocco	80	
UK	70	
Senegal	50	

Algorithm: AC-3

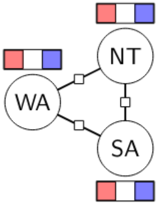
```
$S \leftarrow \{X_j\}$.
While $S$ is non-empty:
    Remove any $X_j$ from $S$.
    For all neighbors $X_i$ of $X_j$:
        Enforce arc consistency on $X_i$ w.r.t. $X_j$.
        If $\mathrm{Domain}_i$ changed, add $X_i$ to $S$.
```

So we have

- Team 1: USA, Iran, Japan, Senegal
- Team 2: France, Brazil, Morocco, UK

In this case, arc consistency also solved our CSP for us!

Arc Consistency



Algorithm: AC-3

```
$S \leftarrow \{X_j\}$.
While $S$ is non-empty:
    Remove any $X_j$ from $S$.
    For all neighbors $X_i$ of $X_j$:
        Enforce arc consistency on $X_i$ w.r.t. $X_j$.
        If $\mathrm{Domain}_i$ changed, add $X_i$ to $S$.
```

But recall from class that AC-3 only looks locally!

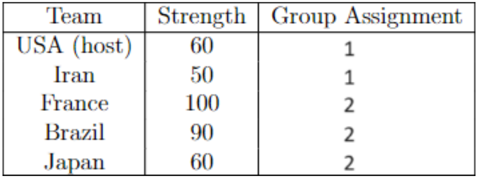
- In the example from class, AC-3 reduced the domains of each node above to 2 assignable colors: red and blue.
- Each node cannot have the same color as a neighboring node, thus, this CSP is not satisfiable!
- Can't assign 2 colors to 3 nodes!
- But AC-3 cannot detect this!

Local Search

Backtracking search and arc consistency help build up assignments.

What if we already have an assignment? Local search to improve!

Consider the modified problem:



Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	1
France	100	2
Brazil	90	2
Japan	60	2

- We want 2 groups, with each group having at least 2 teams.
- We still define the strength of an assignment the same way, thus we still have our factor:

$$h_j(X_1, \ldots, X_n) = \sum_{i:X_i=j}^{n} s_i$$

Local Search

Consider the modified problem:

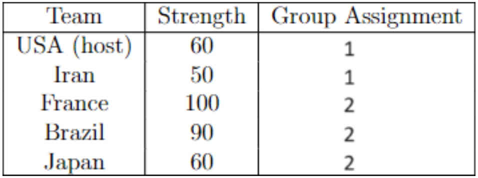
Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	1
France	100	2
Brazil	90	2
Japan	60	2

- We want 2 groups, with each group having at least 2 teams.
- We still define the strength of an assignment the same way, thus we still have our factor:

$$h_j(X_1, \dots, X_n) = \sum_{i:X_i=j}^{n} s_i$$

Suppose we start with the above "random" assignments.
Let's use local search to find a better assignment!

Local Search



Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	1
France	100	2
Brazil	90	2
Japan	60	2

Algorithm: iterated conditional modes (ICM)

```
Initialize $x$ to a random complete assignment
Loop through $i = 1, \dots, n$ until convergence:
    Compute weight of $x_v = x \cup \{X_i : v\}$ for each $v$
    $x \leftarrow x_v$ with highest weight
```

1st iteration:

- USA cannot be reassigned, as Group 1 needs at least 2 teams.
- Iran cannot be reassigned, as Group 1 needs at least 2 teams.
- Reassigning France to Group 1 yields a higher weight:
  $(60 + 50 + 100) * (90 + 60) = 31500$
  vs $(60 + 50) * (100 + 90 + 60) = 27500$
- Reassigning France lowers the size of Group 2 to 2, so Brazil and Japan cannot be reassigned.

Local Search

Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	1
France	100	1
Brazil	90	2
Japan	60	2

Algorithm: iterated conditional modes (ICM)

```
Initialize $x$ to a random complete assignment
Loop through $i = 1, \ldots, n$ until convergence:
    Compute weight of $x_v = x \cup \{X_i : v\}$ for each $v$
    $x \leftarrow x_v$ with highest weight
```

2nd iteration:

- USA can be reassigned, but whether it's in Group 1 or 2 will lead to the same weight total of $150 * 210$. Let it stay in 1.
- Reassigning Iran to Group 2 yields a higher weight: $(60 + 50 + 100) * (90 + 60) = 31500$ vs $(60 + 100) * (50 + 90 + 60) = 32000$
- Group 1 now has only 2 teams, so France stays.
- Continued next slide...

Local Search

Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	1 -> 2
France	100	1
Brazil	90	2
Japan	60	2

Algorithm: iterated conditional modes (ICM)

```
Initialize $x$ to a random complete assignment
Loop through $i = 1, \dots, n$ until convergence:
    Compute weight of $x_v = x \cup \{X_i : v\}$ for each $v$
    $x \leftarrow x_v$ with highest weight
```

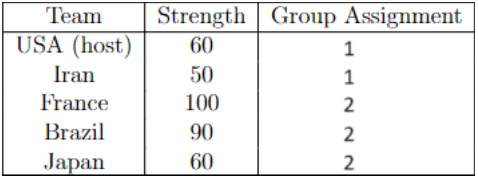
2nd iteration:

- Brazil can be reassigned, but it staying in Group 2 yields a higher weight: $(60 + 100 + 90) * (50 + 60) = 27500$ vs $(60 + 100) * (50 + 90 + 60) = 32000$
- Japan can be reassigned, but it staying in Group 2 yields a higher weight: $(60 + 100 + 60) * (50 + 90) = 30800$ vs $(60 + 100) * (50 + 90 + 60) = 32000$

Any more iterations won't lead to any changes.

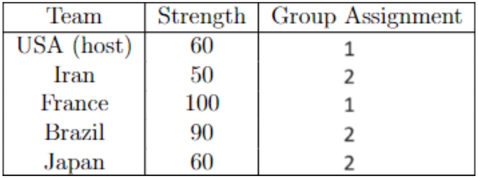
Local Search

And so we went from:



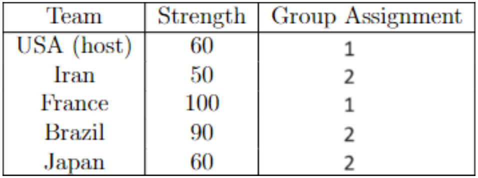
Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	1
France	100	2
Brazil	90	2
Japan	60	2

to



Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	2
France	100	1
Brazil	90	2
Japan	60	2

Local Search



Team	Strength	Group Assignment
USA (host)	60	1
Iran	50	2
France	100	1
Brazil	90	2
Japan	60	2

Note that this isn't the most optimal assignment!

Assigning USA + Iran + Japan to Group 1 and France + Brazil to Group 2 yields a higher weight of 32300!

ICM can get stuck in local maxima!

Final Exam Review

Markov / Bayesian Nets Review