Constraint sat. prob. (Ch. 6)
Announcements

Writing 3 assigned Wed.
  -Find papers (like writ 2) for project
  -scholar.google.com is your friend!
A constraint satisfaction problem is when there are a number of variables in a domain with some restrictions.

A consistent assignment of variables has no violated constraints.

A complete assignment of variables has no unassigned variables.

(A solution is complete and consistent)
CSP

Map coloring is a famous CSP problem
Variables: each state/country
Domain: \{yellow, blue, green, purple\} (here)
Constraints: No adjacent variables same color
CSP

partial and not consistent

Consistent and complete
Another common use of CSP is job scheduling.
CSP

Suppose we have 3 jobs: $J_1$, $J_2$, $J_3$
If $J_1$ takes 20 time units to complete, $J_2$ takes 30 and $J_3$ takes 15 but $J_1$ must be done before $J_3$

How to write this as a boolean expression? (jobs cannot be scheduled at the same time)
Suppose we have 3 jobs: $J_1$, $J_2$, $J_3$

If $J_1$ takes 20 time units to complete, $J_2$ takes 30 and $J_3$ takes 15 but $J_1$ must be done before $J_3$

We can represent this as (and them together):

$J_1 \& J_2: (J_1 + 20 \leq J_2 \text{ or } J_2 + 30 \leq J_1)$

$J_1 \& J_3: (J_1 + 20 \leq J_3)$

$J_2 \& J_3: (J_2 + 30 \leq J_3 \text{ or } J_3 + 15 \leq J_2)$
Types of constraints

A **unary** constraint is for a single variable (i.e. $J_1$ cannot start before time 5)

**Binary** constraints are between two variables (i.e. $J_1$ starts before $J_2$)

All constraints can be broken down into using only binary and unary
Types of constraints

K-consistency is:
For any consistent sets size (k-1), there exists a valid value for any other variable (not in set)

1-consistency: All values in the domain satisfy the variable's unary constraints

2-consistency: All binary values are in domain

3-consistency: Given consistent 2 variables, there is a value for a third variable (i.e. if \{A,B\} is consistent, then exists \( C \) s.t. \( \{A,C\} \& \{B,C\} \))
Types of constraints

For example, 1-consistent means you can pick 0 consistent variables (if you pick nothing it is always consistent) then any assignment to a new variable is also consistent.

This boils down to saying you can pick any valid pick of a single variable in isolation.

In other words, you satisfy the unary constraints.
Types of constraints

2-consistent means you pick a valid value from the domain for one variable and see if there is any valid assignment for a second variable.

3-consistent means you pick a valid pair of values for 2 variables and see if there is any valid assignment for a third variable.

If you are unable to find a valid assignment for the last variable, it is not consistent.
Types of constraints

Rules: 1. Tasmania cannot be red
2. Neighboring providences cannot share colors

2 Colors: red green
Types of constraints

WA = \{\text{red, green}\}
NT = \{\text{red, green}\}
Q = \{\text{red, green}\}
SA = \{\text{red, green}\}
NSW = \{\text{red, green}\}
V = \{\text{red, green}\}
T = \{\text{red, green}\}

Not 1-consistent as we need T to not be red (i.e. rule #2 eliminates T=\text{red})
Types of constraints

WA = NT = Q = SA = NSW = V
= \{\text{red, green}\}
T = \{\text{green}\}

1-consistent now

Also 2-consistent, for example:
Pick WA as “set k-1”, then try to pick NT...
If WA=green, then we can make NT=red
if WA=red, NT=green (true for all pairs)
Types of constraints

WA = NT = Q = SA = NSW = V
= \{\text{red, green}\}
T = \{\text{green}\}

Not 3-consistent!

Pick (WA, SA) and add NT... If NT=green, will not work with either: (WA=red, SA=green) or (WA=green, SA=red)... NT=red also will not work, so NT's domain is empty and not 3-cons.
Types of constraints

Try to do this job problem with: J1, J2 and J3
(Domains are positive integers)

Jobs cannot overlap
J3 takes 3 time units
J2 takes 2 time units
J1 takes 1 time unit
J1 must happen before J3
J2 cannot happen at time 1
All jobs must finish by time 7
(i.e. you can start J2 at time 5 but not at time 6)
Applying constraints

We can repeatedly apply our constraint rules to shrink the domain of variables (we just shrunk NT's domain to nothing)

This reduces the size of the domain, making it easier to check:

- If the domain size is zero, there are no solutions for this problem
- If the domain size is one, this variable must take on that value (the only one in domain)
Applying constraints

AC-3 checks all 2-consistency constraints:

1. Add all binary constraints to queue
2. Pick a binary constraint \((X_i, Y_j)\) from queue
3. If \(x\) in \(\text{domain}(X_i)\) and no consistent \(y\) in \(\text{domain}(Y_j)\), then remove \(x\) from \(\text{domain}(X_i)\)
4. If you removed in step 3, update all other binary constraints involving \(X_i\) (i.e. \((X_i, X_k)\))
5. Goto step 2 until queue empty
Applying constraints

Some problems can be solved by applying constraint restrictions (such as sudoku) (i.e. the size of domain is one after reduction)

Harder problems this is insufficient and we will need to search to find a solution

Which is what we will do... now
CSP vs. search

Let us go back to Australia coloring:

How can you color using search techniques?
CSP vs. search

We can use an incremental approach:

State = currently colored provinces (and their color choices)

Action = add a new color to any province that does not conflict with the constraints

Goal: To find a state where all provinces are colored
CSP vs. search

Is there a problem?
CSP vs. search

Is there a problem?

Let \( d = \) domain size (number of colorings), \( n = \) number of variables (provinces)

The number of leaves are \( n! * d^n \)

However, there are only \( d^n \) possible states in the CSP so there must be a lot of duplicate leaves (not including mid-tree parts)
CSP vs. search

CSP assumes one thing general search does not: the order of actions does not matter.

In CSP, we can assign a value to a variable at any time and in any order without changing the problem (all we care about is the end state).

So all we need to do is limit our search to one variable per depth, and we will have a match with CSP of \( d^n \) leaves (all combinations).
Let's apply CSP modified DFS on Australia:
(assign values&variables in alphabetical order)

1\textsuperscript{st}: blue
2\textsuperscript{nd}: green
3\textsuperscript{rd}: red
CSP vs. search

NSW:

NT:

Q:

SA: X X X X X X

T: ...

Nothing colored

NSW red
CSP vs. search

STOP PICKING BLUE EVERY TIME!!!!