CS335 Introduction to AI

Constraint Satisfaction Problems (CSP)

Francisco Iacobelli

Department of Computer Science
Northeastern Illinois University

March 19, 2024

Constraint Satisfaction Probelm

Example

- Color each region either red,green or blue
- No adjacent region can have the same color

Constraint Satisfaction Problems

CSPs

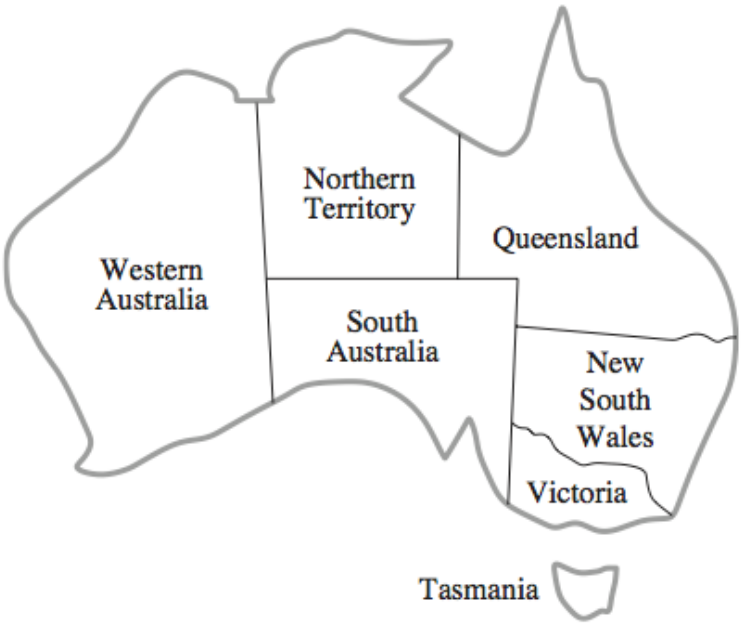
- So far, states evaluated by heuristics and goal
- CSP = factored representation of state
  - set of variables with a value
  - allows for more efficient algorithms
  - We want to find any solution or that there's none

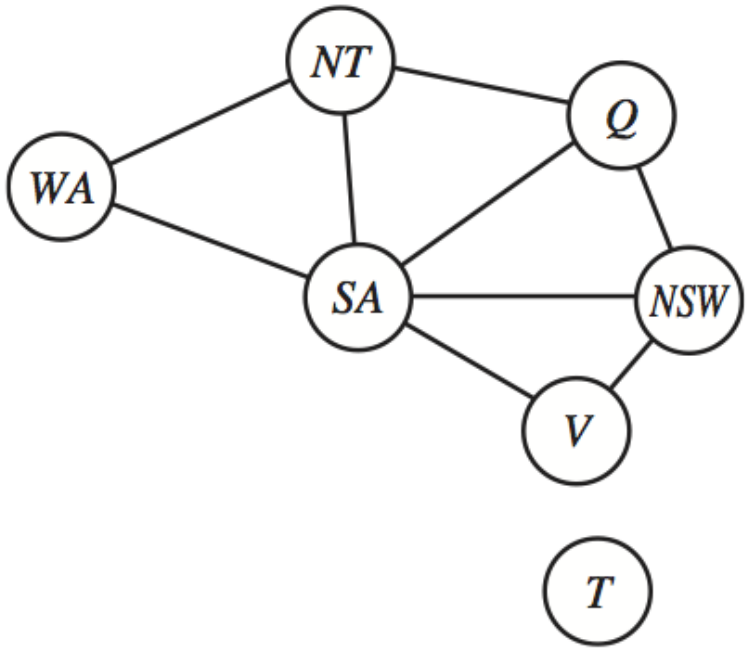
- X, a set of variables, $\{X_1, \ldots, X_n\}$
- D, a set of **domains** for each X. $\{D_1, \ldots, D_n\}$
- C, a set of constraints

- X, a set of variables, $\{X_1, \ldots, X_n\}$
- D, a set of **domains** for each X. $\{D_1, \ldots, D_n\}$
  - $D_i = \{v_1, \ldots, v_n\}$ for X_i
- C, a set of constraints
  - $C_i =< scope, rel >$

Constraint Satisfaction Probelm

Map Coloring





- Color each region either red,green or blue
- No adjacent region can have the same color

- $X = \{SA, NSW, NT, Q, WA, V, T\}$
- $D = \{red, blue, green\}$ for each $X_i \in X\}$
- $C = \{\langle(\forall X_i, X_j such\ that\ X_i\ touches\ X_j), (Color(X_i) \neq Color(X_j))\rangle\}$

Constraint Satisfaction Problem

From Math to Pseudo Code

1. $X = \{SA, NSW, NT, Q, WA, V, T\}$
2. $D = \{red, blue, green\}$ for each $X_i \in X\}$
3. $C = \{\langle(\forall X_i, X_j \textit{such that } X_i \textit{ touches } X_j), (Color(X_i) \neq Color(X_j))\rangle\}$

For each item ask: Do the elements of the set imply an action?

If **no**, then they are simple `list`s. Determine the types of the elements.

If **yes**, then, for each action, how many parameters are needed? of what types?

Then ask: does this item modify any previous data? if so, rethink the data type.

Constraint Satisfaction Problem

From Math to Pseudo Code

1. $X = \{SA, NSW, NT, Q, WA, V, T\}$
2. $D = \{red, blue, green\}$ for each $X_i \in X\}$
3. $C = \{\langle(\forall X_i, X_j \text{such that } X_i \text{ touches } X_j), (Color(X_i) \neq Color(X_j))\rangle\}$

The first one seems like a list of `string`s.

```
X = ['SA','NSW','NT','Q','WA','V','T']
```

Item #2 seems to be a `list`, but it affects the item #1. Each element has its possible colors.

Let's change X to a structure that associates provinces with colors. Maybe:

```
  // define the object
  object State (name):
     current_color=''
     possible_colors = ['red','blue','green']
 // instantiate elements of type 'State'
  SA = State('SA')
  NSW = State('NSW')
  etc.
// Re-create our initial list.
  X = [SA,NSW,NT,Q,...etc.]
```

Constraint Satisfaction Problems

From Math to pseudocode

1. $X = \{SA, NSW, NT, Q, WA, V, T\}$
2. $D = \{red, blue, green\}$ for each $X_i \in X\}$
3. $C = \{\langle(\forall X_i, X_j \textit{such that } X_i \textit{ touches } X_j), (Color(X_i) \neq Color(X_j))\rangle\}$

Item #3 is a set of actions that check that **two provinces** that are *adjacent* do not have the same colors.

```
function c1(x1,x2) return boolean
    if x1 touches x2 then
        return x1.current_color is not x2.current_color
    return True
```

Scheduling Classes

Can you formulate it in terms of variables, domains and constraints?

- $X =?$
- $D =?$
- $C =?$

Can you formulate it in terms of variables, domains and constraints?

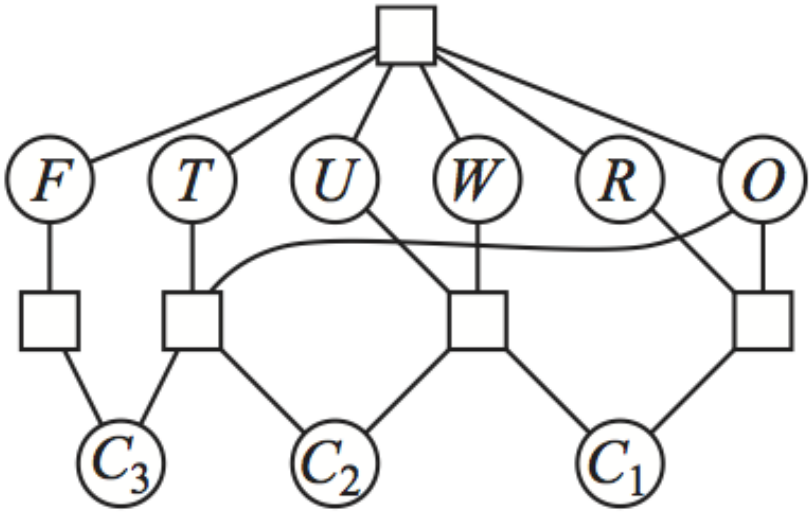
- $X = \{CS235, CS355, CS101, etc.\}$
- $D = \{Mon9am, Mon11am, Mon1pm \ldots Fri4pm, Fri6pm\}$
- $C =$
  - $\forall i, j$ if x_i, x_j are co-requisites, then $Ti \neq Tj$

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

Constraint kind: *AllDiff*

27 *AllDiff*s

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$



Constraints:

- $AllDiff(F, T, U, W, R, O)$
- $O + O = R + 10 \times C_1$
- $C_1 + W + W = U + 10 \times C_2$
- $C_2 + T + T = O + 10 \times C_3$
- $C_3 = F$

CSPs Formally

Kinds of Constraints

- **Unary**: Involve a single variable ($SA \neq green$)
- **Binary**: Involve a pair of variables ($SA \neq WA$)
- **Higher Order**: Involve 3 or more (Cryptarithmetic's)

Solving CSPs

Search:Depth Limited

- CSP with n variables with domain size d
- Branching factor at top = nd
- At next level: $(n-1)d$
- In the end $n!d^n$ leaves. But only d^n possible complete assignments

Solving CSPs

Backtracking

- Variable assignments are commutative:
  $(WA = red \Rightarrow NT = green) \iff (NT = green \Rightarrow WA = red)$
- Only need to consider assignments to a single variable at each node
- Backtracks when variable has no legal value
- Can solve n-queens for $n \approx 25$

Solving CSPs

Backtracking

```
function BacktrackingSearch(csp)
    return Backtrack({},csp) //returns solution or failure

function Backtrack(assignment,csp)
    if assignment is complete return assignment
    u_var=SelectUnassignedVariable(csp)
    for each value in OrderDomainValues(u_var,assignment,csp) do
        if isConsistent(value,assignment)
            add {u_var=value} to assignment
            inferences = Inference(csp,u_var,value) //maybe with F.C.
            if inferences!=failure
                add inferences to assignment
                result = Backtrack(assignment,csp)
                if result != failure then
                    return result
        remove {u_var=value} and inferences from assignment
    return failure
```

Backtracking

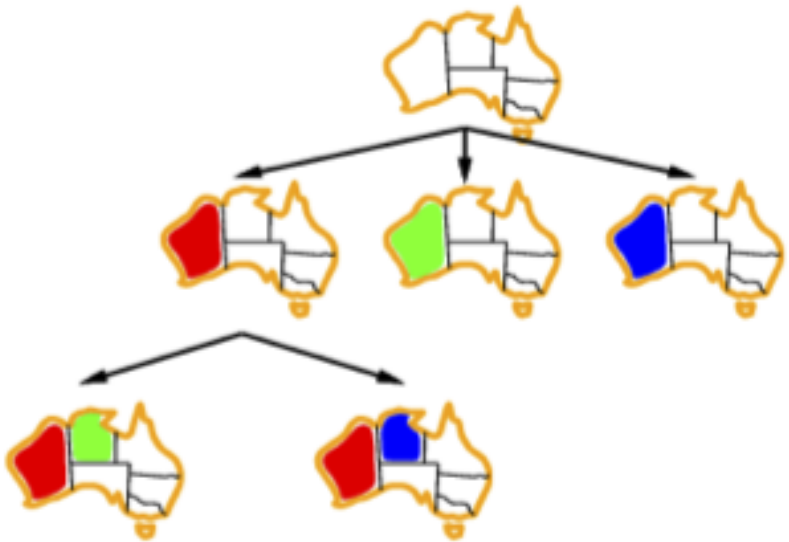
Example (from Marc Erich Latoshik)

Backtracking

Example

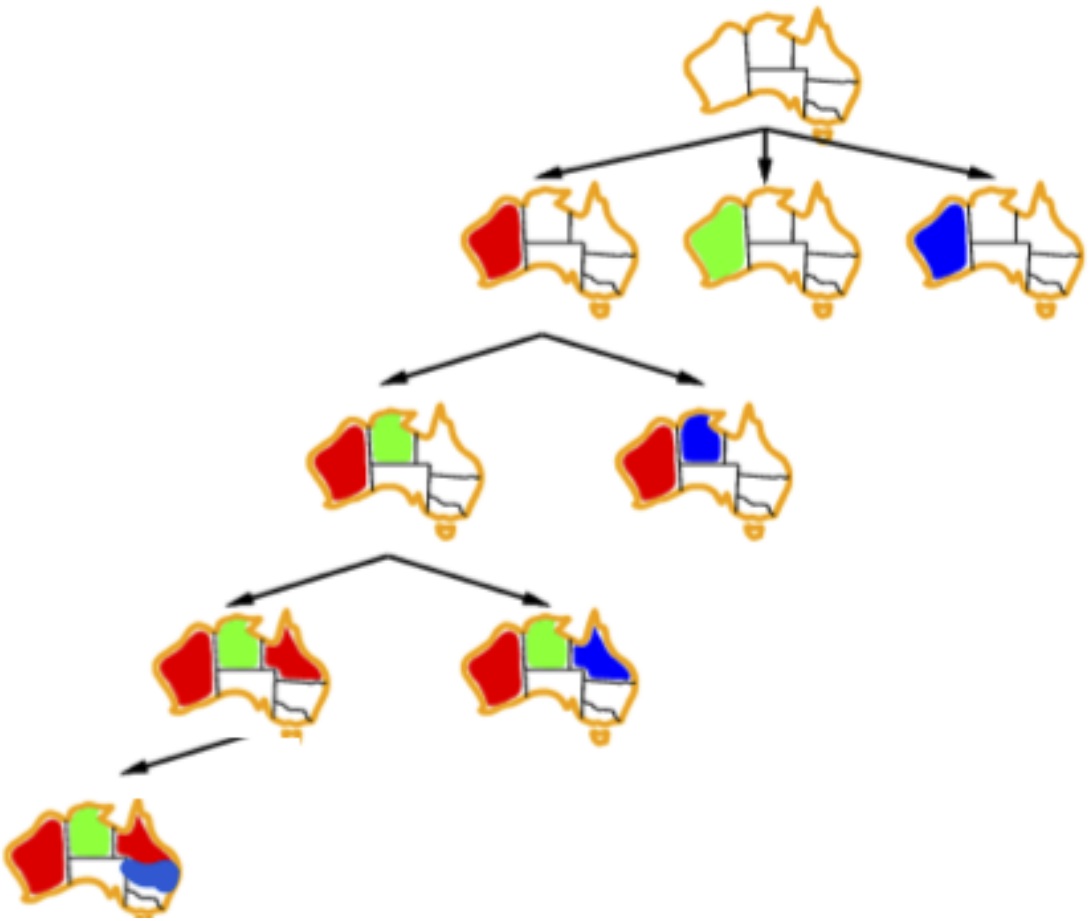
Backtracking

Example



Backtracking

Example



Backtracking

Heuristics and Considerations

- Really important how to choose the next variable:
  - Minimum Remaining Values (MRV) heuristic (var w/fewest legal values)
  - Degree Heuristics (var involved in most constraints)
  - Least constraining value (prefers var flexibility for the future)
- Check for constraint consistency; i.e. Inference with AC-3 (arc consistency)

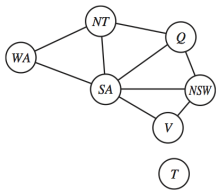
Can be used to check consistency (inferences), or with minor modifications, it can be the basis to solve small CSPs.

```
function ForwardChecking(csp) //returns a new domain for each
    var
    for each variable X in csp do
        for each unassigned variable Y connected to X do
            for each value d in Domain(Y)
                if d is inconsistent with Value(X)
                    Domain(Y)={Domain(Y)-d}
    return csp //whith modified domains
```

If we randomly assign X a value from the *Domain*(X) after each iteration, we can have a brute force CSP.

Inference in CSP

Example

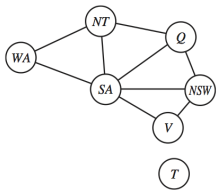


Domains

WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB

Inference in CSP

Example

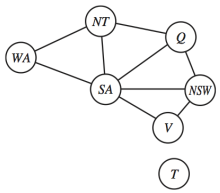


	WA	NT	Q	NSW	V	SA	T
Domains	RGB	RGB	RGB	RGB	RGB	RGB	RGB
After WA	(R)	GB	RGB	RGB	RGB	GB	RGB

No possible assignments for SA, we try other assignments.

Inference in CSP

Example

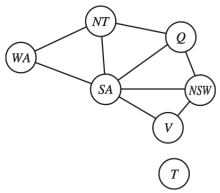


	WA	NT	Q	NSW	V	SA	T
Domains	RGB	RGB	RGB	RGB	RGB	RGB	RGB
After WA	(R)	GB	RGB	RGB	RGB	GB	RGB
After Q	(R)	B	(G)	RB	RGB	B	RGB

No possible assignments for SA, we try other assignments.

Inference in CSP

Example



	WA	NT	Q	NSW	V	SA	T
Domains	RGB	RGB	RGB	RGB	RGB	RGB	RGB
After WA	(R)	GB	RGB	RGB	RGB	GB	RGB
After Q	(R)	B	(G)	RB	RGB	B	RGB
After V	(R)	B	(G)	R	(B)		RGB

No possible assignments for SA, we try other assignments.

Local Search in CSPs

Heuristics

- Local search use a complete state formulation
- Initial assignment
- Change one variable at a time using heuristics

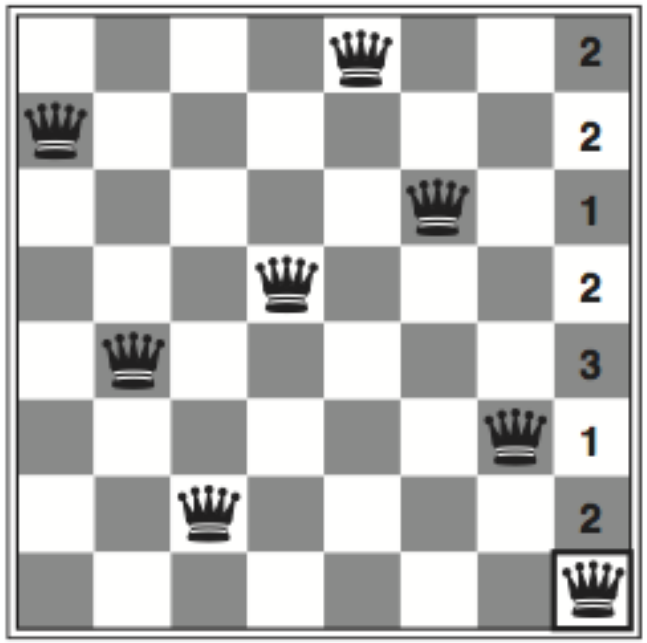
Local Search in CSPs

Min-Conflicts

```
function MinConflicts(csp,max_steps)
// csp, max_steps is num of steps before giving up
    current = an initial assignment for csp
    for i=1 to max_steps do
        if current is a solution for csp
            return current
        var = a randomly chosen conflicted variable in csp
        value = the value v for var that minimizes Conflicts
        set var = value in current
    return failure
```

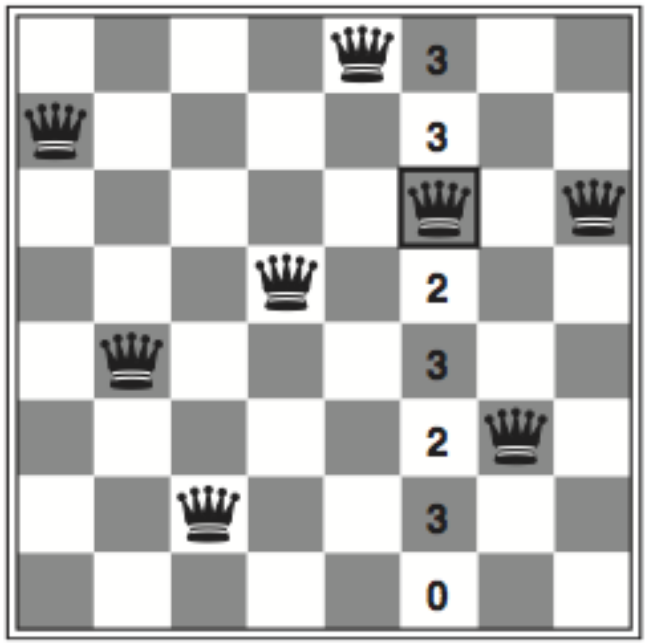
Local Search in CSPs

Example



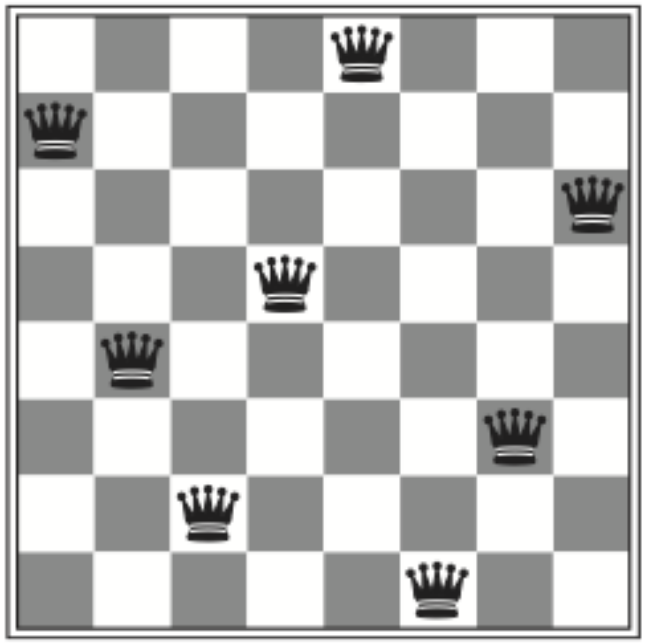
Local Search in CSPs

Example



Local Search in CSPs

Example



Median number of consistency checks over five runs

From Russel and Norvig

Problem	Backtracking	Fwd. Checking	FC+MRV	Min Conflicts
n-queens	> 40,000K	> 40,000K	817K	4K
USA states	> 1,000K	2K	60	64
Zebra	3,859K	35K	500	2K