

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory



## Goal-Based Agents Problem Solving as Constraint Satisfaction

Vasant Honavar  
Artificial Intelligence Research Laboratory  
Department of Computer Science  
Bioinformatics and Computational Biology Program  
Center for Computational Intelligence, Learning, & Discovery  
Iowa State University  
honavar@cs.iastate.edu  
[www.cs.iastate.edu/~honavar/](http://www.cs.iastate.edu/~honavar/)  
[www.cild.iastate.edu/](http://www.cild.iastate.edu/)  
[www.bcb.iastate.edu/](http://www.bcb.iastate.edu/)  
[www.igert.iastate.edu](http://www.igert.iastate.edu/)

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Problem solving as constraint satisfaction

- What is a Constraint satisfaction problem?
- Properties of CSP
- Backtracking for CSP
- Local search for CSP
- Problem structure and decomposition
- Constraint propagation

Vasant Honavar, 2006.

---

---

---

---

---

---

---

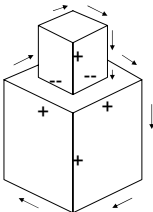
---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Constraint satisfaction problem



- Scene labeling problem (Waltz, 1975)
- Assumptions – trihedral vertices, no shadows, no cracks, general viewing position
- Vertex types: L, T, Y, / \
- Line types: +, -- → ←
- Given a 2-d drawing of a 3-d scene, label the lines so as to make explicit, the 3-d interpretation of the scene

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Constraint

- A constraint is a (typically logical) relationship between variables in a domain
- Constraints are declarative
  - Specify what must hold without specifying how to enforce it
- Each constraint applies to one or more variables
- Constraints are rarely independent of each other
- The result is independent of the order in which the constraints are enforced

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Constraint satisfaction problems

- What is a CSP?
  - Finite set of variables  $V_1, V_2, \dots, V_n$
  - Finite set of variables  $C_1, C_2, \dots, C_m$
  - Non-empty domain of possible values for each variable  $D_{V_1}, D_{V_2}, \dots, D_{V_n}$
  - Each constraint  $C_i$  rules out certain assignments of values to variables e.g.,  $V_1 \neq V_2$
- A *state* is defined as an *assignment* of values to some or all variables.
- *Consistent assignment*: assignment does not violate the constraints

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Constraint satisfaction problems

- An assignment is *complete* when every variable has been assigned a value
- A *solution* to a CSP is a complete assignment that satisfies all constraints.
- Some CSPs require a solution that maximizes an *objective function*.

CSP Applications:

- Scheduling observations on the Hubble Space Telescope
- Temporal reasoning
- Floor planning
- Map coloring
- Cryptography
- 3-d interpretation of line drawings

Vasant Honavar, 2006.

---

---

---

---

---

---

---


---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### CSP example: map coloring



- Variables:  $WA, NT, Q, NSW, V, SA, T$
- Domains:  $D_i = \{red, green, blue\}$
- Constraints: adjacent regions must have different colors.
  - E.g.  $WA \neq NT$  i.e.,
  - E.g.  $(WA, NT) \in \{(red, green), (red, blue), (green, red), \dots\}$

Vasant Honavar, 2006.

---

---

---

---

---

---

---


---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### CSP example: map coloring



- Solutions are assignments satisfying all constraints, e.g.  $\{WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green\}$

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

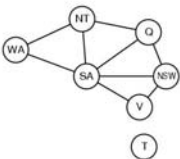
---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Constraint graph

- CSP solvers exploit special properties of CSP (more on this later)
- CSP solvers benefit from
  - Standard representation
  - Generic goal and successor functions
  - Generic heuristics



Constraint graph = nodes are variables, edges show constraints.  
Graph can be used to simplify search.  
e.g. Tasmania is an independent sub problem.

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---


---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Map coloring

Using 3 colors (R, G, & B), color the US map such that no two adjacent states have the same color

- Variables?
- Domains?
- Constraints?



Vasant Honavar, 2006.

---

---

---

---

---

---

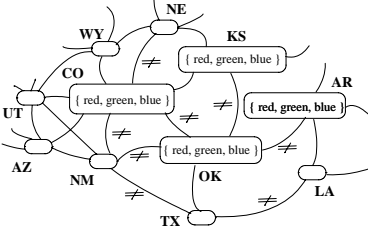
---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Map coloring – Constraint graph representation

Using 3 colors (R, G, & B), color the US map such that no two adjacent states have the same color



Vasant Honavar, 2006.

---

---

---

---

---

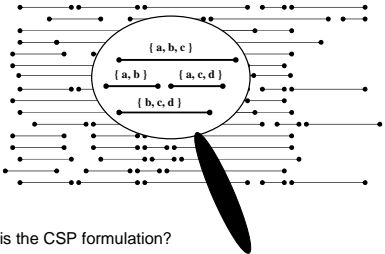
---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Example: Resource Allocation



What is the CSP formulation?

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Resource Allocation – Constraint Graph Representation

**Interval Order**

T1 ← [R1, R3] →  
 T2 ← [R1, R3] →  
 T3 ← [R1, R2, R3] →  
 T4 ← [R1, R3, R4] →  
 T5 ← [R2, R3] →  
 T6 ← [R2, R4] →  
 T7 ← [R2, R4] →

**Constraint Graph**

T1 (R1, R3) ≠ T2 (R1, R3)  
 T1 (R1, R3) ≠ T3 (R1, R2, R3)  
 T1 (R1, R3) ≠ T4 (R1, R3, R4)  
 T2 (R1, R3) ≠ T3 (R1, R2, R3)  
 T2 (R1, R3) ≠ T4 (R1, R3, R4)  
 T3 (R1, R2, R3) ≠ T5 (R2, R3)  
 T3 (R1, R2, R3) ≠ T6 (R2, R4)  
 T3 (R1, R2, R3) ≠ T7 (R2, R4)  
 T4 (R1, R3, R4) ≠ T6 (R2, R4)  
 T4 (R1, R3, R4) ≠ T7 (R2, R4)  
 T5 (R2, R3) ≠ T6 (R2, R4)  
 T5 (R2, R3) ≠ T7 (R2, R4)  
 T6 (R2, R4) ≠ T7 (R2, R4)

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Example: Product Configuration

Train, elevator, car, etc.

- Given:
  - Components and their attributes (variables)
  - Domain covered by each characteristic (values)
  - Relations among the components (constraints)
  - A set of required functionalities (more constraints)
- Find: a product configuration
  - an acceptable combination of components that realizes the required functionalities

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Example: cryptarithmic

TWO  
+ TWO  
FOUR

Variables:  $F, T, U, W, R, O, X_1, X_2, X_3$   
 Domains:  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$   
 Constraints  
 $allDiff(F, T, U, W, R, O)$   
 $O + O = R + 10 \cdot X_1$ , etc.

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Varieties of CSPs

- Discrete variables
  - Finite domains; size  $d \Rightarrow O(d^n)$  possible assignments
    - E.g. Boolean CSPs, include Boolean satisfiability (NP-complete).
  - Infinite domains (integers, strings, etc.)
    - E.g. job scheduling, variables are start/end days for each job
    - Need a constraint language to express constraints e.g.  $StartJob_1 + 5 \leq StartJob_2$ .
    - Linear constraints solvable, nonlinear undecidable
- Continuous variables
  - e.g. start/end times for Hubble Telescope observations
  - Linear constraints solvable in poly time by LP methods

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Varieties of constraints

- Unary constraints involve a single variable
  - e.g.  $SA \neq green$
- Binary constraints involve pairs of variables
  - e.g.  $SA \neq WA$
- Higher-order constraints involve 3 or more variables.
  - e.g. cryptarithmic column constraints
- Preference (soft constraints) e.g. *red* is better than *green* often represented by a cost for each variable assignment
  - constrained optimization problems

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### CSP as a standard search problem

- A CSP can easily expressed as a standard search problem
- *Initial State*:
  - the empty assignment {}.
- *Successor function*:
  - Assign value to unassigned variable provided that there is no conflict
- *Goal test*:
  - the current assignment is complete
- *Arc cost*:
  - constant cost for each step

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### CSP as a standard search problem

- Solution is found at depth  $n$  (if there are  $n$  variables).
  - Hence depth first search can be used
- Path is irrelevant
- Branching factor  $b$  at the top level is  $nd$ .
- $b=(n-1)d$  at depth  $l$ , hence  $n!d^n$  leaves ( $d^n$  complete assignments)

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Commutativity

- CSPs are commutative.
  - The order of any given set of actions has no effect on the outcome
  - Example:
    - choose colors for Australian territories one at a time
    - [WA=red then NT=green] same as [NT=green then WA=red]
    - All CSP search algorithms consider a single variable assignment at a time  $\Rightarrow$  there are  $d^n$  leaves

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Backtracking search

- Depth-first search
- Chooses values for one variable at a time and backtracks when a variable has no legal values left to assign
- General performance not good
  - (see table p. 143 of text)

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Backtracking search

```

function BACKTRACKING-SEARCH(csp) return a solution or failure
return RECURSIVE-BACKTRACKING({}, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) return a solution or failure
if assignment is complete then return assignment
var ← SELECT-UNASSIGNED-VARIABLE (VARIABLES[csp], assignment, csp)
for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
  if value is consistent with assignment according to CONSTRAINTS[csp]
  then
    add {var=value} to assignment
    result ← RECURSIVE-BACKTRACKING(assignment, csp)
    if result ≠ failure then return result
    remove {var=value} from assignment
return failure

```

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Backtracking example

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Backtracking example

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Backtracking example

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Backtracking example

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Subtrees have similar topologies

- Consider 4 variables  $W, X, Y, Z$  with domains of cardinality 4, 3, 2 and 1 respectively

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Variable ordering and Search space size

Variable ordering W, X, Y, Z

Search space size  
 $= (4)(3)(2)(1) + (4)(3)(2) + (4)(3) + (4)$   
 $= 24 + 24 + 12 + 4$   
 $= 64$

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Variable ordering and Search space size

Variable ordering Z, Y, X, W

Search space size  
 $= (4)(3)(2)(1) + (3)(2)(1) + (2)(1) + 1$   
 $= 24 + 6 + 2 + 1 = 33$

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Backtracking search

- Standard backtracking fails to exploit special properties of CSP
  - Subtrees have similar topologies:
  - Search space has minimal size under a certain ordering of variables (most constrained to least constrained)

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Backtracking search

- **Subtrees have similar topologies:**
  - If we find an assignment of values to a pair of variables is incompatible, it should rule out all assignments that include the incompatible partial assignment
- **Search space has minimal size under a certain ordering of variables** (most constrained to least constrained)
  - Choose ordering of variables to assign that minimizes the search space size

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Improving backtracking efficiency

- Which variable should be assigned next?
- In what order should its values be tried?
- Can we detect inevitable failure early?
- Can we take advantage of problem structure?

Vasant Honavar, 2006.

---

---

---

---

---

---

---


---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Minimum remaining values



```

var ← SELECT-UNASSIGNED-
VARIABLE(VARIABLES[csp],assignment,csp)

```

- A.k.a. most constrained variable heuristic
- *Rule:* choose variable with the fewest legal moves
- *Which variable shall we try first?*

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Degree heuristic

- Use degree heuristic
- *Rule*: select variable that is involved in the largest number of constraints on other unassigned variables.
- Degree heuristic is very useful as a tie breaker.
- *In what order should its values be tried?*

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Least constraining value

- Least constraining value heuristic
- *Rule*: given a variable choose the least constraining value i.e. the one that leaves the maximum flexibility for subsequent variable assignments.

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Forward checking

- Can we detect inevitable failure early?  
– *And avoid it later?*
- *Forward checking idea*: keep track of remaining legal values for unassigned variables.
- Terminate search when any variable has no legal values

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Forward checking

- Assign {WA=red}
- Effects on other variables connected by constraints with WA
  - NT can no longer be red
  - SA can no longer be red

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Forward checking

- Assign {Q=green}
- Effects on other variables connected by constraints with WA
  - NT can no longer be green
  - NSW can no longer be green
  - SA can no longer be green
- MRV heuristic will automatically select NT and SA next, why?

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Forward checking

- If V is assigned blue
- Effects on other variables connected by constraints with WA
  - SA is empty
  - NSW can no longer be blue
- FC has detected that partial assignment is inconsistent with the constraints and backtracking can occur

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### 4 Queens Revisited – Importance of representation

- **N-queens**: formulation 1
  - Variables? **4 Rows**
  - Domains? **4 Column positions**
  - Size of CSP?  $4 \times 4 \times 4 \times 4 = 4^4 = 2^8$

	1	2	3	4
$V_1$				
$V_2$				
$V_3$				
$V_4$				

- **N-queens**: formulation 2
  - Variables? **16 Cells**
  - Domains? **{0,1}**
  - Size of CSP?  $2^{16}$

$V_{11}$	$V_{12}$	$V_{13}$	$V_{14}$
$V_{21}$	$V_{22}$	$V_{23}$	$V_{24}$
$V_{31}$	$V_{32}$	$V_{33}$	$V_{34}$
$V_{41}$	$V_{42}$	$V_{43}$	$V_{44}$

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

Variables – columns  
Values – rows

	1	2	3	4
1				
2				
3				
4				

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

Search Tree:

- X1: {1,2,3,4}
- X2: { , 3,4}
- X3: { , 2, , 4}
- X4: { , 2,3, }

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

Search Tree:

- X1: {1,2,3,4}
- X2: { , 3,4}
- X3: { , 2, , 4}
- X4: { , 2,3, }

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

Search Tree:

- X1: {1,2,3,4}
- X2: { , 3,4}
- X3: { , , , }
- X4: { , 2,3, }

Note X3 has empty domain  
 Cannot backtrack to X2 (X2=4 is ruled out)  
 Must backtrack to X1

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

```

    graph TD
      X1["X1  
{ ,2,3,4}"] --- X2["X2  
{1,2,3,4}"]
      X1 --- X3["X3  
{1,2,3,4}"]
      X1 --- X4["X4  
{1,2,3,4}"]
      X2 --- X3
      X2 --- X4
      X3 --- X4
  
```

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

```

    graph TD
      X1["X1  
{ ,2,3,4}"] --- X2["X2  
{ , , ,4}"]
      X1 --- X3["X3  
{1, ,3, }"]
      X1 --- X4["X4  
{1, ,3,4}"]
      X2 --- X3
      X2 --- X4
      X3 --- X4
  
```

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

```

    graph TD
      X1["X1  
{ ,2,3,4}"] --- X2["X2  
{ , , ,4}"]
      X1 --- X3["X3  
{1, ,3, }"]
      X1 --- X4["X4  
{1, ,3,4}"]
      X2 --- X3
      X2 --- X4
      X3 --- X4
  
```

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

The diagram illustrates the 4-Queens problem. On the left is a 4x4 grid with columns labeled 1 to 4 and rows labeled 1 to 4. Queens are placed at (1,2), (2,3), (3,3), and (4,2). On the right is a search tree with four nodes:
 

- X1: { ,2,3,4}
- X2: { , , ,4}
- X3: {1, , , }
- X4: {1, ,3, }

 Edges connect X1 to X3 and X4, and X2 to X3 and X4.

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

The diagram illustrates the 4-Queens problem. On the left is a 4x4 grid with columns labeled 1 to 4 and rows labeled 1 to 4. Queens are placed at (1,2), (2,3), (3,3), and (4,2). On the right is a search tree with four nodes:
 

- X1: { ,2,3,4}
- X2: { , , ,4}
- X3: {1, , , }
- X4: {1, ,3, }

 Edges connect X1 to X3 and X4, and X2 to X3 and X4.

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

The diagram illustrates the 4-Queens problem. On the left is a 4x4 grid with columns labeled 1 to 4 and rows labeled 1 to 4. Queens are placed at (1,2), (2,3), (3,3), and (4,2). On the right is a search tree with four nodes:
 

- X1: { ,2,3,4}
- X2: { , , ,4}
- X3: {1, , , }
- X4: { , ,3, }

 Edges connect X1 to X3 and X4, and X2 to X3 and X4.

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Example: 4-Queens Problem

```

    graph TD
      X1["X1  
{ ,2,3,4}"] --- X2["X2  
{ , , ,4}"]
      X1 --- X3["X3  
{1, , , }"]
      X1 --- X4["X4  
{ , ,3, }"]
      X2 --- X3
      X2 --- X4
      X3 --- X4
  
```

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Constraint propagation

- Solving CSPs with combination of heuristics plus forward checking is more efficient than either approach alone
- Constraint propagation repeatedly enforces constraints locally

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Arc consistency

- $X \rightarrow Y$  is consistent iff  
for every value  $x$  of  $X$  there is some allowed  $y$
- $SA \rightarrow NSW$  is consistent iff  
 $SA=blue$  and  $NSW=red$

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Arc consistency

- $X \rightarrow Y$  is consistent iff for every value  $x$  of  $X$  there is some allowed  $y$
- $NSW \rightarrow SA$  is consistent iff
  - $NSW=red$  and  $SA=blue$
  - $NSW=blue$  and  $SA=???$

Arc can be made consistent by removing *blue* from *NSW*

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Arc consistency

- Arc can be made consistent by removing *blue* from *NSW*
- RECHECK neighbours !!
  - Remove red from *V*

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Arc consistency

- Arc can be made consistent by removing *blue* from *NSW*
- RECHECK neighbours !!
  - Remove red from *V*
- Arc consistency detects failure earlier than FC
- Can be run as a preprocessor or after each assignment.
  - Repeated until no inconsistency remains

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Arc consistency algorithm

```

function AC-3(csp) return the CSP, possibly with reduced domains
inputs: csp, a binary csp with variables  $\{X_1, X_2, \dots, X_n\}$ 
local variables: queue, a queue of arcs initially the arcs in csp

while queue is not empty do
   $(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\textit{queue})$ 
  if REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) then
    for each  $X_k$  in NEIGHBORS[ $X_j$ ] do
      add  $(X_i, X_k)$  to queue
function REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) return true iff we remove a
value
  removed  $\leftarrow$  false
  for each x in DOMAIN[ $X_i$ ] do
    if no value y in DOMAIN[ $X_j$ ] allows (x,y) to satisfy the constraints between
 $X_i$  and  $X_j$ 
    then delete x from DOMAIN[ $X_i$ ]; removed  $\leftarrow$  true
  return removed

```

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### K-consistency

- Arc consistency does not detect all inconsistencies:
  - Partial assignment  $\{WA=red, NSW=red\}$  is inconsistent.
- Stronger forms of propagation can be defined using the notion of k-consistency.
- A CSP is k-consistent if for any set of k-1 variables and for any consistent assignment to those variables, a consistent value can always be assigned to any kth variable.
  - E.g. 1-consistency or node-consistency
  - E.g. 2-consistency or arc-consistency
  - E.g. 3-consistency or path-consistency

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### K-consistency

- A graph is strongly k-consistent if
  - It is k-consistent and
  - Is also (k-1) consistent, (k-2) consistent, ... all the way down to 1-consistent.
- This is ideal since a solution can be found in time  $O(nd)$  instead of  $O(n^2d^3)$
- YET *no free lunch*: any algorithm for establishing n - consistency must take time exponential in n, in the worst case

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Further variants

- Checking special constraints efficiently
  - Checking Alldif(...) constraint *E.g. {WA=red, NSW=red}*
- Intelligent backtracking
  - Standard form is chronological backtracking i.e. try different value for preceding variable.
  - More intelligent backtrack to conflict set.
    - Set of variables that caused the failure or set of previously assigned variables that are connected to X by constraints.
    - Backjumping moves back to most recent element of the conflict set.
    - Forward checking can be used to determine conflict set.

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Local search for CSP

- Use complete-state representation
- For CSPs
  - allow states with unsatisfied constraints
  - operators **reassign** variable values
- Variable selection: randomly select any conflicted variable
- Value selection: *min-conflicts heuristic*
  - Select new value that results in a minimum number of conflicts with the other variables

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Local search for CSP

```

function MIN-CONFLICTS(csp, max_steps) return solution or failure
inputs: csp, a constraint satisfaction problem
       max_steps, the number of steps allowed before giving up

current ← an initial complete assignment for csp
for i = 1 to max_steps do
  if current is a solution for csp then return current
  var ← a randomly chosen, conflicted variable from
  VARIABLES[csp]
  value ← the value v for var that minimize
  CONFLICTS(var, v, current, csp)
  set var = value in current
return failure
  
```

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Min-conflicts example 1

h=5      h=3      h=1

- Use of min-conflicts heuristic in hill-climbing

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Min-conflicts example 2

- A two-stage solution for an 8-queens problem using min-conflicts heuristic
- At each stage a queen is chosen for reassignment in its column.
- The algorithm moves the queen to the min-conflict square breaking ties randomly

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Advantages of local search

- The runtime of min-conflicts is roughly independent of problem size.
  - Solving the millions-queen problem in roughly 50 steps.
- Local search can be used in an online setting.
  - Backtrack search requires more time

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Problem structure

- How can the problem structure help to find a solution quickly?
- Identify relatively independent subproblems:
  - Coloring Tasmania and mainland are independent subproblems
  - Identifiable as connected components of constrained graph
- Improves performance

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Problem structure

- Suppose each problem has  $c$  variables out of a total of  $n$ .
- Worst case solution cost is  $O(n/c d^c)$ , i.e. linear in  $n$ 
  - Instead of  $O(d^n)$ , exponential in  $n$
- E.g.  $n=80, c=20, d=2$ 
  - $2^{80} = 4$  billion years at 1 million nodes/sec.
  - $4 * 2^{20} = .4$  second at 1 million nodes/sec

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Tree-structured CSPs

- Theorem: if the constraint graph has no loops then CSP can be solved in  $O(nd^2)$  time
- Compare difference with general CSP, where worst case is  $O(d^n)$

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Tree-structured CSPs

- In many cases subproblems of a CSP are connected as a tree
- Any tree-structured CSP can be solved in time linear in the number of variables.
  - Choose a variable as root, order variables from root to leaves such that every node's parent precedes it in the ordering. (label var from  $X_1$  to  $X_n$ )
  - For  $j$  from  $n$  down to 2, apply REMOVE-INCONSISTENT-VALUES(Parent( $X_j$ ),  $X_j$ )
  - For  $j$  from 1 to  $n$  assign  $X_j$  consistently with Parent( $X_j$ )

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Nearly tree-structured CSPs

- Can more general constraint graphs be reduced to trees?
- Two approaches:
  - Remove certain nodes
  - Collapse certain nodes

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science Artificial Intelligence Research Laboratory

### Nearly tree-structured CSPs – Cutset Conditioning

- Idea: assign values to some variables so that the remaining variables form a tree.
- Assume that we assign  $\{SA=x\} \leftarrow$  cycle cutset
  - And remove any values from the other variables that are inconsistent.
  - The selected value for SA could be the wrong one so we have to try all of them

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Nearly tree-structured CSPs

- Feasible if cycle cutset is small.
- Finding the smallest cycle cutset is NP-hard
  - Approximation algorithms exist

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Nearly tree-structured CSPs

- Tree decomposition of the constraint graph in a set of connected subproblems
- Solve each subproblem independently
- Combine resulting solutions
- Necessary conditions:
  - Every variable appears in at least one of the subproblems
  - If two variables are connected in the original problem, they must appear together in at least one subproblem
  - If a variable appears in two subproblems, it must appear in each node on the path

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---

Iowa State University Department of Computer Science  
Artificial Intelligence Research Laboratory

### Summary

- CSP is a special kind of search problem:
  - states defined by values of a fixed set of variables,
  - goal test defined by constraints on variable values
- CSP solution exploits the special properties of CSP
- Variable ordering and value selection heuristics help significantly
- Forward checking prevents assignments that lead to failure
- Tree structured CSPs can be solved in linear time
- Iterative min-conflicts is usually effective in practice

Vasant Honavar, 2006.

---

---

---

---

---

---

---

---