Solving a CSP

Outline

I. Binary CSP

II. Constraint Propagation

III. Backtracking search

* Figures/images are from the textbook site (or by the instructor).

I. Binary CSP

A *unary constraint* restricts the value of a single variable.

$$\langle (SA), SA \neq green \rangle$$

I. Binary CSP

A *unary constraint* restricts the value of a single variable.

$$\langle (SA), SA \neq green \rangle$$

A *binary constraint* relates two variables.

$$SA \neq WA$$

I. Binary CSP

A *unary constraint* restricts the value of a single variable.

$$\langle (SA), SA \neq green \rangle$$

A *binary constraint* relates two variables.

$$SA \neq WA$$

A higher order constraint relate more variables.

Ternary constraint example: $\langle (X, Y, Z), X < Y < Z$ or $X > Y > Z \rangle$

I. Binary CSP

A *unary constraint* restricts the value of a single variable.

$$\langle (SA), SA \neq green \rangle$$

A *binary constraint* relates two variables.

$$SA \neq WA$$

A higher order constraint relate more variables.

Ternary constraint example: $\langle (X, Y, Z), X < Y < Z$ or $X > Y > Z \rangle$

A *global constraint* involves an arbitrary number of variables (but not necessarily all the variables).

I. Binary CSP

A *unary constraint* restricts the value of a single variable.

$$\langle (SA), SA \neq green \rangle$$

A *binary constraint* relates two variables.

$$SA \neq WA$$

A higher order constraint relate more variables.

Ternary constraint example: $\langle (X, Y, Z), X < Y < Z$ or $X > Y > Z \rangle$

A *global constraint* involves an arbitrary number of variables (but not necessarily all the variables).

Alldiff$(v_1, \ldots, v_k)$: variables $v_1, \ldots, v_k$ must have different values.

e.g., Sudoku (all variables in a row, column, or 3×3 box).

Cryptarithmetic Puzzle

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

Cryptarithmetic Puzzle

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

Addition constraints on the four columns:

$$O + O = R + 10 \cdot C_1$$
$$C_1 + W + W = U + 10 \cdot C_2$$
$$C_2 + T + T = O + 10 \cdot C_3$$
$$C_3 = F$$

Cryptarithmetic Puzzle

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

Addition constraints on the four columns:

$$
\begin{aligned}
O + O &= R + 10 \cdot \boxed{C_1} \\
C_1 + W + W &= U + 10 \cdot \boxed{C_2} \\
C_2 + T + T &= O + 10 \cdot \boxed{C_3} \\
C_3 &= F
\end{aligned}
$$

carry-overs into the tens, hundreds, and thousands, resp.

Cryptarithmetic Puzzle

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

Addition constraints on the four columns:

$$\begin{aligned} O + O &= R + 10 \cdot \boxed{C_1} \\ C_1 + W + W &= U + 10 \cdot \boxed{C_2} \\ C_2 + T + T &= O + 10 \cdot \boxed{C_3} \\ C_3 &= F \end{aligned}$$

carry-overs into the tens, hundreds, and thousands, resp.

A *hypergraph* generalizes a graph in that an edge (called an *hyperedge*) can connect more than two vertices.

Cryptarithmetic Puzzle

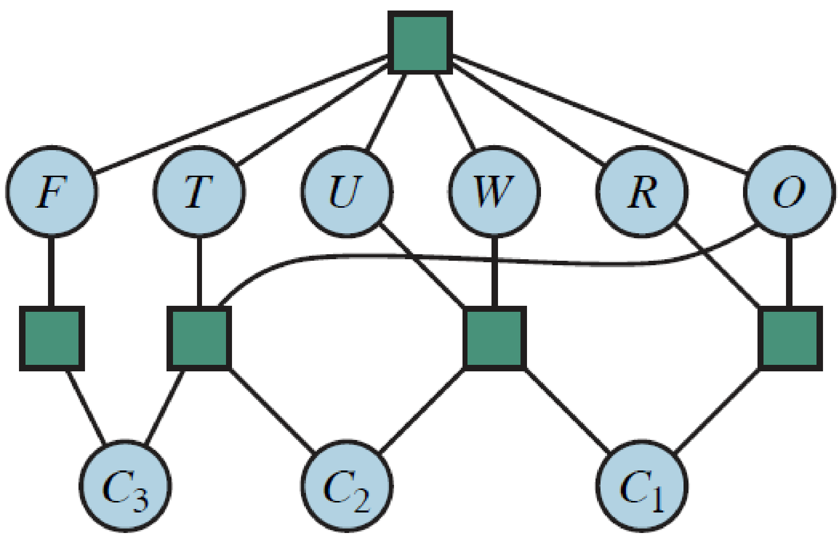
$$
\begin{array}{cccc}
 & T & W & O \\
+ & T & W & O \\
\hline
F & O & U & R
\end{array}
$$

Addition constraints on the four columns:

$$
\begin{aligned}
O + O &= R + 10 \cdot \boxed{C_1} \\
C_1 + W + W &= U + 10 \cdot \boxed{C_2} \\
C_2 + T + T &= O + 10 \cdot \boxed{C_3} \\
C_3 &= F
\end{aligned}
$$

carry-overs into the tens, hundreds, and thousands, resp.

A *hypergraph* generalizes a graph in that an edge (called an *hyperedge*) can connect more than two vertices.



Constraint hypergraph

Cryptarithmetic Puzzle

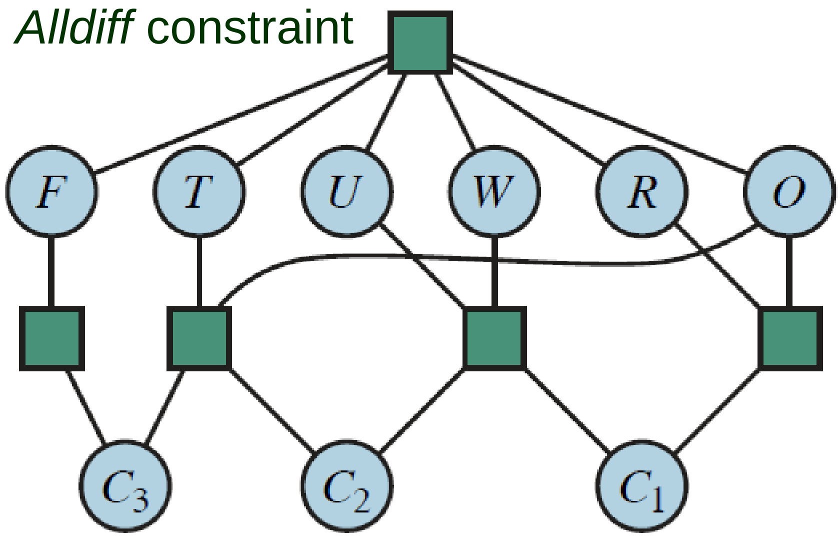
$$
\begin{array}{cccc}
 & T & W & O \\
+ & T & W & O \\
\hline
F & O & U & R
\end{array}
$$

Addition constraints on the four columns:

$$
\begin{aligned}
O + O &= R + 10 \cdot C_1 \\
C_1 + W + W &= U + 10 \cdot C_2 \\
C_2 + T + T &= O + 10 \cdot C_3 \\
C_3 &= F
\end{aligned}
$$

carry-overs into the tens, hundreds, and thousands, resp.

Alldiff constraint



Constraint hypergraph

A *hypergraph* generalizes a graph in that an edge (called an *hyperedge*) can connect more than two vertices.

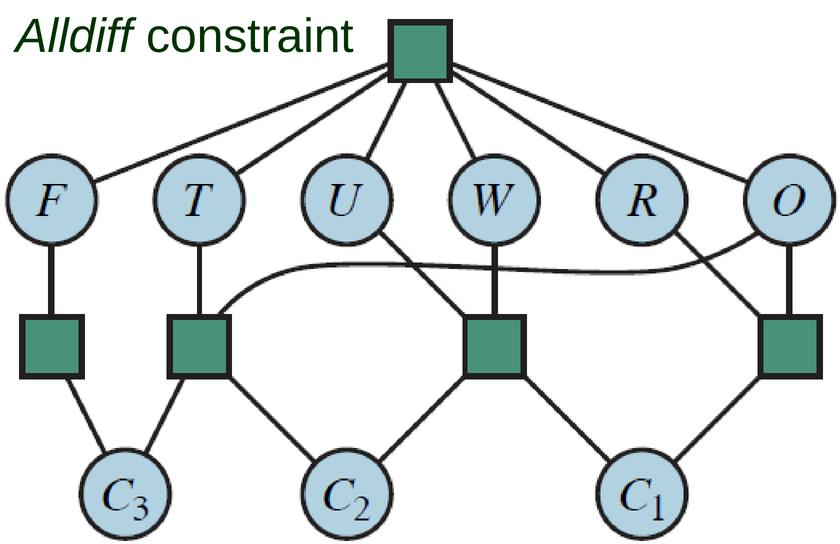
Cryptarithmetic Puzzle

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

Addition constraints on the four columns:

$$\begin{aligned} O + O &= R + 10 \cdot C_1 \\ C_1 + W + W &= U + 10 \cdot C_2 \\ C_2 + T + T &= O + 10 \cdot C_3 \\ C_3 &= F \end{aligned}$$

carry-overs into the tens, hundreds, and thousands, resp.



Constraint hypergraph

A *hypergraph* generalizes a graph in that an edge (called an *hyperedge*) can connect more than two vertices.

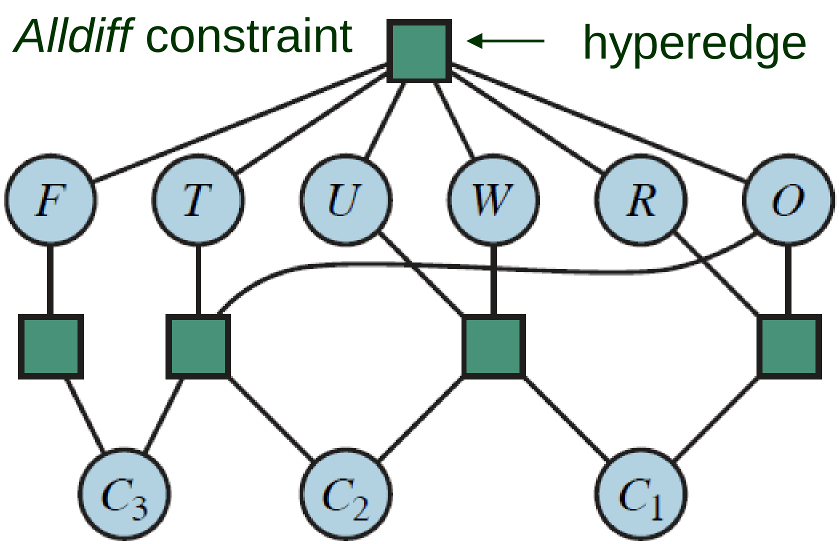
Cryptarithmetic Puzzle

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

Addition constraints on the four columns:

$$
\begin{aligned}
O + O &= R + 10 \cdot C_1 \\
C_1 + W + W &= U + 10 \cdot C_2 \\
C_2 + T + T &= O + 10 \cdot C_3 \\
C_3 &= F
\end{aligned}
$$

carry-overs into the tens, hundreds, and thousands, resp.



Constraint hypergraph

A *hypergraph* generalizes a graph in that an edge (called an *hyperedge*) can connect more than two vertices.

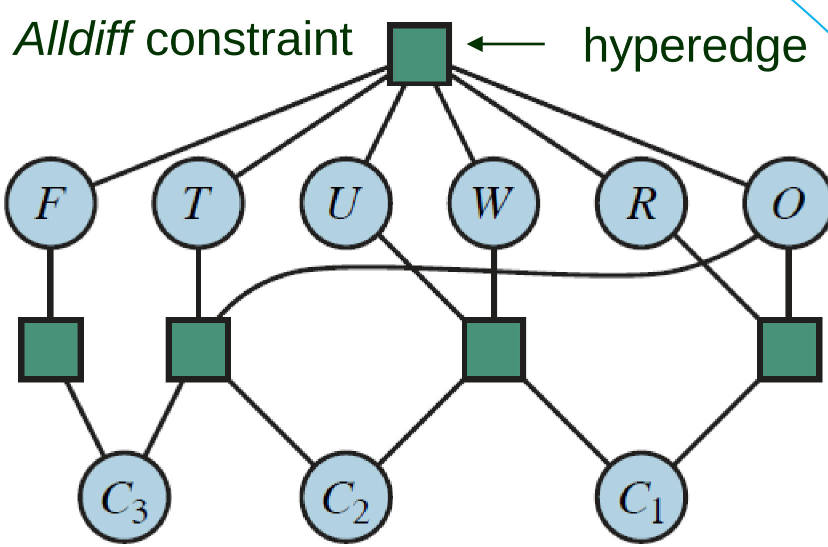
Cryptarithmetic Puzzle

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

Addition constraints on the four columns:

$$
\begin{aligned}
O + O &= R + 10 \cdot C_1 \\
C_1 + W + W &= U + 10 \cdot C_2 \\
C_2 + T + T &= O + 10 \cdot C_3 \\
C_3 &= F
\end{aligned}
$$

carry-overs into the tens, hundreds, and thousands, resp.



Constraint hypergraph

A *hypergraph* generalizes a graph in that an edge (called an *hyperedge*) can connect more than two vertices.

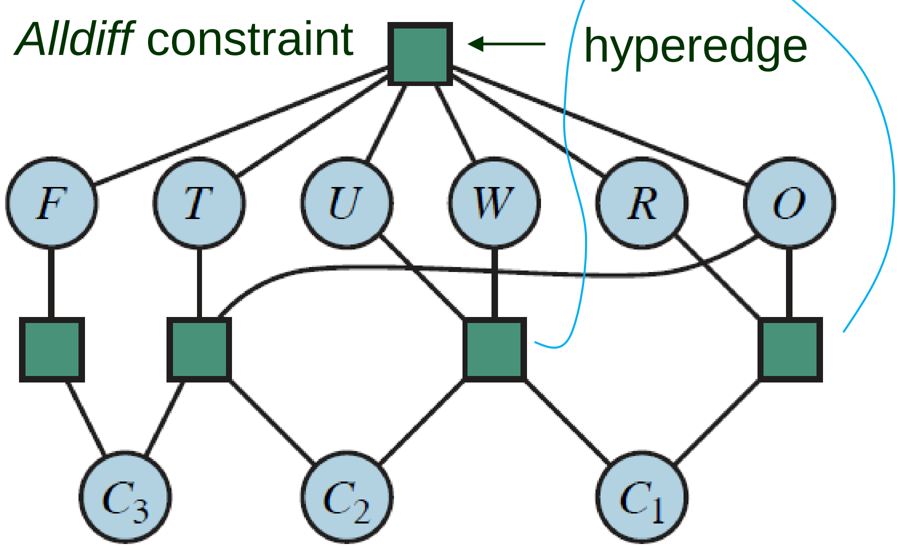
Cryptarithmetic Puzzle

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

Addition constraints on the four columns:

$$
\begin{aligned}
O + O &= R + 10 \cdot C_1 \\
C_1 + W + W &= U + 10 \cdot C_2 \\
C_2 + T + T &= O + 10 \cdot C_3 \\
C_3 &= F
\end{aligned}
$$

carry-overs into the tens, hundreds, and thousands, resp.



Constraint hypergraph

A *hypergraph* generalizes a graph in that an edge (called an *hyperedge*) can connect more than two vertices.

Transformation to Binary Constraints

Any n-ary constraint can be reduced to a set of binary constraints by introducing auxiliary variables.

Transformation to Binary Constraints

Any n-ary constraint can be reduced to a set of binary constraints by introducing auxiliary variables.

$$\Downarrow$$

Any CSP can be transformed into a binary CSP.

Transformation to Binary Constraints

Any n-ary constraint can be reduced to a set of binary constraints by introducing auxiliary variables.

$$\Downarrow$$

Any CSP can be transformed into a binary CSP.

Advantages of a global constraint such as *Alldiff*:

- Easier and less error-prone to write.
- Allowing efficient special-purpose inference algorithms.

II. Constraint Propagation

Use constraints to reduce the number of legal values for a variable, which in turn reduce those for another variable, and so on.

II. Constraint Propagation

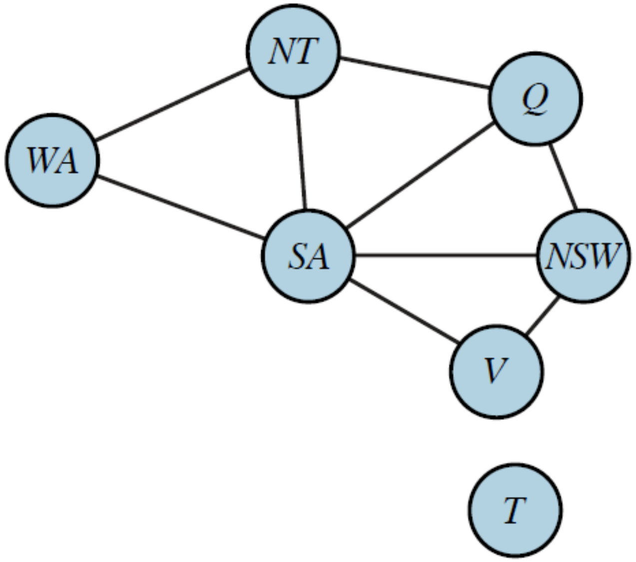
Use constraints to reduce the number of legal values for a variable, which in turn reduce those for another variable, and so on.

Enforcement of local consistency can cause inconsistent values to be eliminated throughout the constraint graph.

II. Constraint Propagation

Use constraints to reduce the number of legal values for a variable, which in turn reduce those for another variable, and so on.

Enforcement of local consistency can cause inconsistent values to be eliminated throughout the constraint graph.

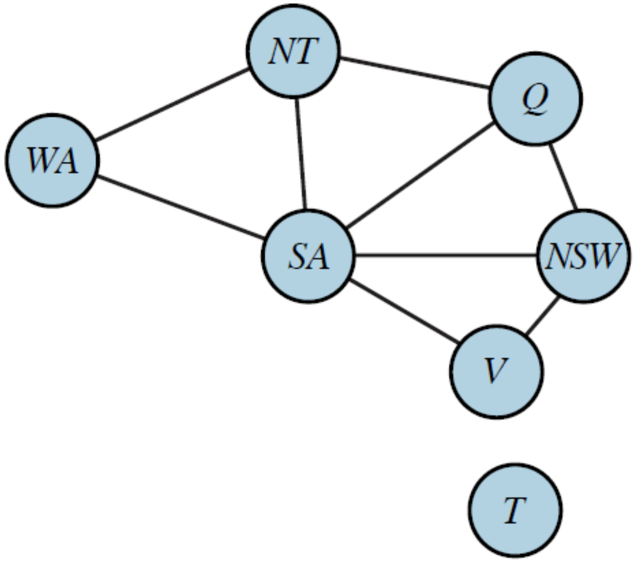


- Node consistency

II. Constraint Propagation

Use constraints to reduce the number of legal values for a variable, which in turn reduce those for another variable, and so on.

Enforcement of local consistency can cause inconsistent values to be eliminated throughout the constraint graph.

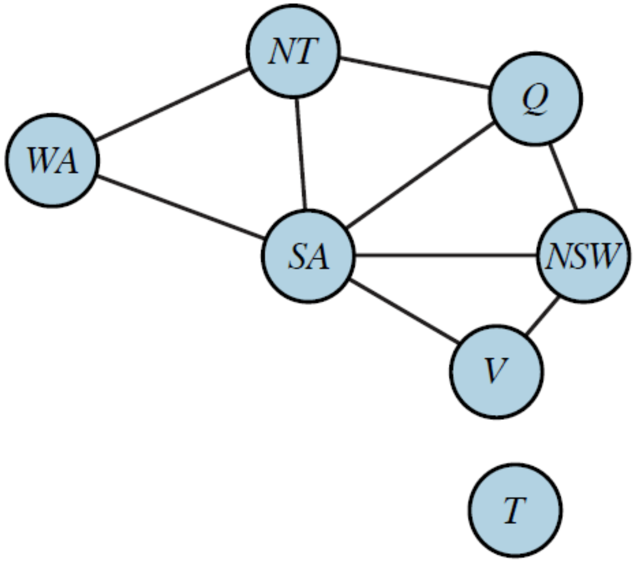


- Node consistency

II. Constraint Propagation

Use constraints to reduce the number of legal values for a variable, which in turn reduce those for another variable, and so on.

Enforcement of local consistency can cause inconsistent values to be eliminated throughout the constraint graph.



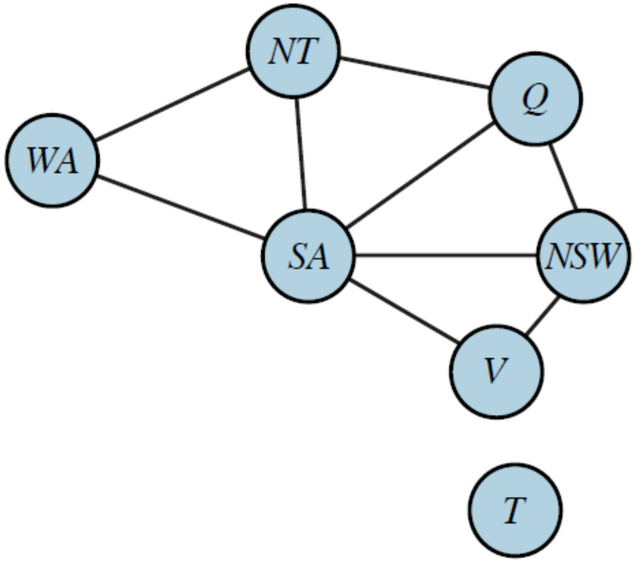
- Node consistency

  Suppose South Australians dislike green.

II. Constraint Propagation

Use constraints to reduce the number of legal values for a variable, which in turn reduce those for another variable, and so on.

Enforcement of local consistency can cause inconsistent values to be eliminated throughout the constraint graph.



- Node consistency

  Suppose South Australians dislike green.

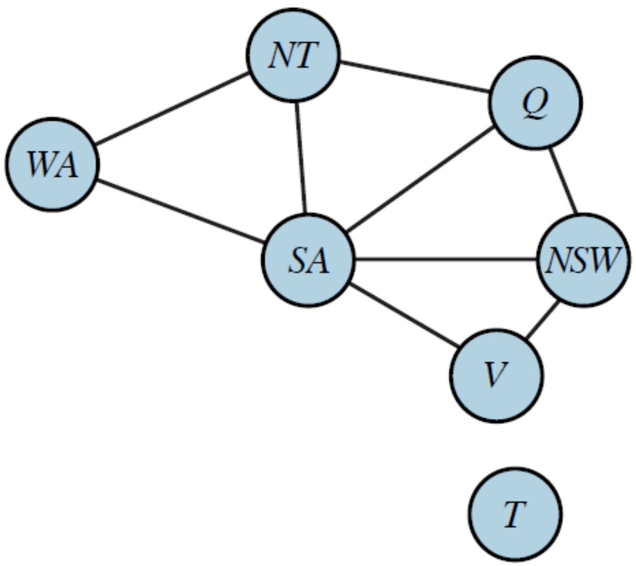
  ⇩

  Domain for *SA*: {red, blue}

II. Constraint Propagation

Use constraints to reduce the number of legal values for a variable, which in turn reduce those for another variable, and so on.

Enforcement of local consistency can cause inconsistent values to be eliminated throughout the constraint graph.



- Node consistency

Suppose South Australians dislike green.

$\Downarrow$

Domain for *SA*: {red, blue}

Eliminate all the unary constraints by reducing the corresponding variables at the start.

Arc Consistency

A variable X_i is *arc-consistent* with respect to (w.r.t.) another variable X_j if for every value in D_i there exists a value in D_j that satisfies the binary constraint on the arc (i.e., edge (X_i, X_j)).

Arc Consistency

A variable X_i is *arc-consistent* with respect to (w.r.t.) another variable X_j if for every value in D_i there exists a value in D_j that satisfies the binary constraint on the arc (i.e., edge (X_i, X_j)).

The variable is *arc-consistent* if it is so w.r.t every other variable that it shares a binary constraint with.

Arc Consistency

A variable X_i is *arc-consistent* with respect to (w.r.t.) another variable X_j if for every value in D_i there exists a value in D_j that satisfies the binary constraint on the arc (i.e., edge (X_i, X_j)).

The variable is *arc-consistent* if it is so w.r.t every other variable that it shares a binary constraint with.

The constraint graph is *arc-consistent* if every variable is arc consistent with every other variable.

Application of Arc Consistency

Example 1

$$Y = X^2 \quad \text{where } X, Y \in \{0, 1, \ldots, 9\}$$

Application of Arc Consistency

Example 1

$$Y = X^2 \quad \text{where } X, Y \in \{0,1, \dots, 9\}$$

X arc consistent w.r.t. Y

Application of Arc Consistency

Example 1

$$Y = X^2 \quad \text{where } X, Y \in \{0, 1, \ldots, 9\}$$

X arc consistent w.r.t. Y $\Longrightarrow$ Reduced domain of X: $\{0, 1, 2, 3\}$

Application of Arc Consistency

Example 1

$$Y = X^2 \quad \text{where } X, Y \in \{0,1,\dots,9\}$$

X arc consistent w.r.t. Y $\Longrightarrow$ Reduced domain of X: $\{0, 1, 2, 3\}$

Y arc consistent w.r.t. X

Application of Arc Consistency

Example 1

$$Y = X^2 \quad \text{where } X, Y \in \{0, 1, \ldots, 9\}$$

X arc consistent w.r.t. Y $\Longrightarrow$ Reduced domain of X: $\{0, 1, 2, 3\}$

Y arc consistent w.r.t. X $\Longrightarrow$ Reduced domain of Y: $\{0, 1, 4, 9\}$

Application of Arc Consistency

Example 1

$$Y = X^2 \quad \text{where } X, Y \in \{0,1, \dots, 9\}$$

X arc consistent w.r.t. Y $\Longrightarrow$ Reduced domain of X: $\{0, 1, 2, 3\}$

Y arc consistent w.r.t. X $\Longrightarrow$ Reduced domain of Y: $\{0, 1, 4, 9\}$

Example 2 Australia map-coloring

Application of Arc Consistency

Example 1

$$Y = X^2 \quad \text{where } X, Y \in \{0,1, \ldots, 9\}$$

X arc consistent w.r.t. Y $\Longrightarrow$ Reduced domain of X: $\{0, 1, 2, 3\}$

Y arc consistent w.r.t. X $\Longrightarrow$ Reduced domain of Y: $\{0, 1, 4, 9\}$

Example 2 Australia map-coloring

$SA \neq WA$

Application of Arc Consistency

Example 1

$$Y = X^2 \quad \text{where } X, Y \in \{0,1,...,9\}$$

X arc consistent w.r.t. Y $\Longrightarrow$ Reduced domain of X: $\{0, 1, 2, 3\}$

Y arc consistent w.r.t. X $\Longrightarrow$ Reduced domain of Y: $\{0, 1, 4, 9\}$

Example 2 Australia map-coloring

$SA \neq WA$

No effect on the domain {*red*, *green*, *blue*} of each variable.

Arc-Consistency Algorithm AC-3

function AC-3(*csp*) **returns** false if an inconsistency is found and true otherwise
  queue ← a queue of arcs, initially all the arcs in *csp*

  while *queue* is not empty **do**
    $(X_i,\ X_j) \leftarrow$ POP(*queue*)
    if REVISE(*csp*, $X_i,\ X_j$) **then**
      if size of $D_i\ =\ 0$ **then return** *false*
      for each X_k **in** X_i.NEIGHBORS - $\{X_j\}$ **do**
        add $(X_k,\ X_i)$ to *queue*
  return *true*

// propagate to other variables sharing a
// constraint with X_i since its domain has
// reduced.

function REVISE(*csp*, $X_i,\ X_j$) **returns** true iff we revise the domain of X_i
  revised ← *false*
  for each x **in** D_i **do**
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j **then**
      delete x from D_i
      revised ← *true*
  return *revised*

Arc-Consistency Algorithm AC-3

function AC-3(*csp*) **returns** false if an inconsistency is found and true otherwise
  queue ← a $\boxed{\text{queue}}$ of arcs, initially all the arcs in *csp*

  while *queue* is not empty **do**
    $(X_i, X_j) \leftarrow$ POP(*queue*)
    if REVISE(*csp*, X_i, X_j) **then**
      if size of $D_i = 0$ **then return** *false*
      for each X_k **in** X_i.NEIGHBORS - $\{X_j\}$ **do**
        add (X_k, X_i) to *queue*
  return *true*

// propagate to other variables sharing a
// constraint with X_i since its domain has
// reduced.

function REVISE(*csp*, X_i, X_j) **returns** true iff we revise the domain of X_i
  revised ← *false*
  for each x **in** D_i **do**
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j **then**
      delete x from D_i
      revised ← *true*
  return *revised*

Domain size $\leq d$, c binary constraints.

- Each arc inserted into the queue d times.
- Each checking takes $O(d^2)$ time.

Arc-Consistency Algorithm AC-3

function AC-3(*csp*) **returns** false if an inconsistency is found and true otherwise
  queue ← a queue of arcs, initially all the arcs in *csp*

  while *queue* is not empty **do**
    $(X_i,\ X_j) \leftarrow$ POP(*queue*)
    if REVISE(*csp*, X_i, X_j) **then**
      if size of $D_i = 0$ **then return** *false*
      for each X_k **in** X_i.NEIGHBORS - $\{X_j\}$ **do**
        add $(X_k,\ X_i)$ to *queue*
  return *true*

// propagate to other variables sharing a
// constraint with X_i since its domain has
// reduced.

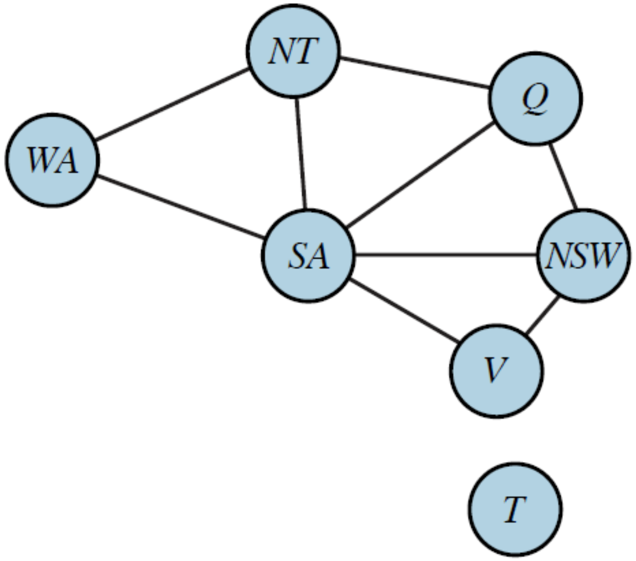
function REVISE(*csp*, X_i, X_j) **returns** true iff we revise the domain of X_i
  revised ← *false*
  for each x **in** D_i **do**
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j **then**
      delete x from D_i
      revised ← *true*
  return *revised*

Domain size $\leq d$, c binary constraints.

- Each arc inserted into the queue d times.
- Each checking takes $O(d^2)$ time.

$O(cd^3)$

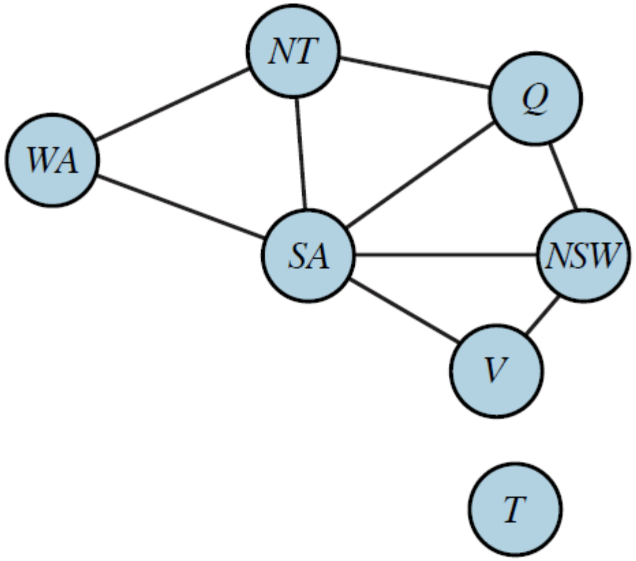
Path Consistency



Arc consistency reduces the domains.

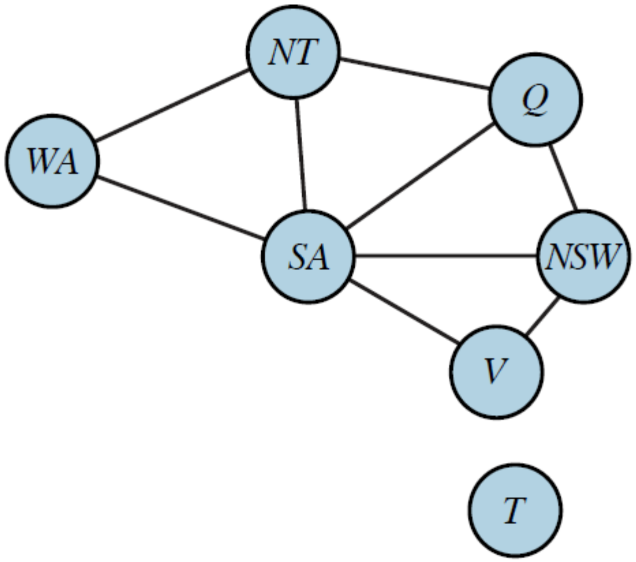
Path Consistency

Arc consistency reduces the domains.



No solution if using two colors even though every constraint can be satisfied individually.

Path Consistency

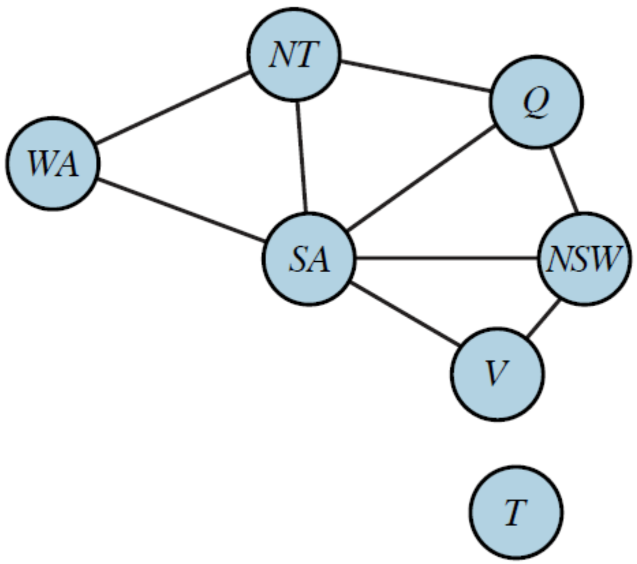


Arc consistency reduces the domains.

Path consistency checks implicit constraints inferable across triples of variables along a path.

No solution if using two colors even though every constraint can be satisfied individually.

Path Consistency

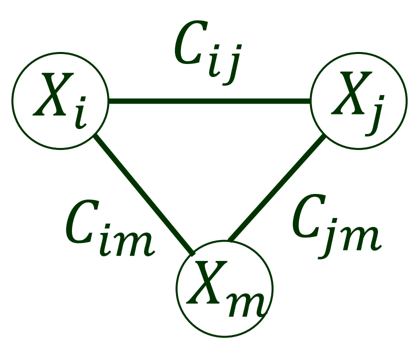


No solution if using two colors even though every constraint can be satisfied individually.

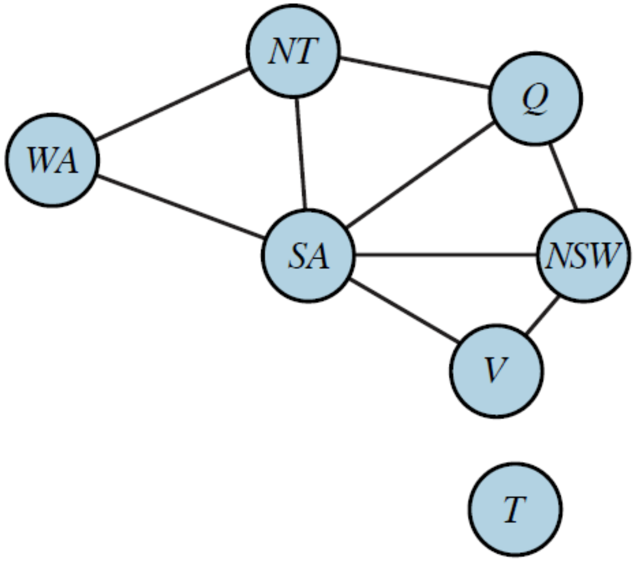
Arc consistency reduces the domains.

Path consistency checks implicit constraints inferable across triples of variables along a path.

$\{X_i, X_j\}$ is *path-consistent* w.r.t. X_m if for every assignment to X_i, X_j consistent with their constraint C_{ij} (if exists), there exists an assignment to X_m that satisfies the constraints C_{im} and C_{jm} between them and X_m.



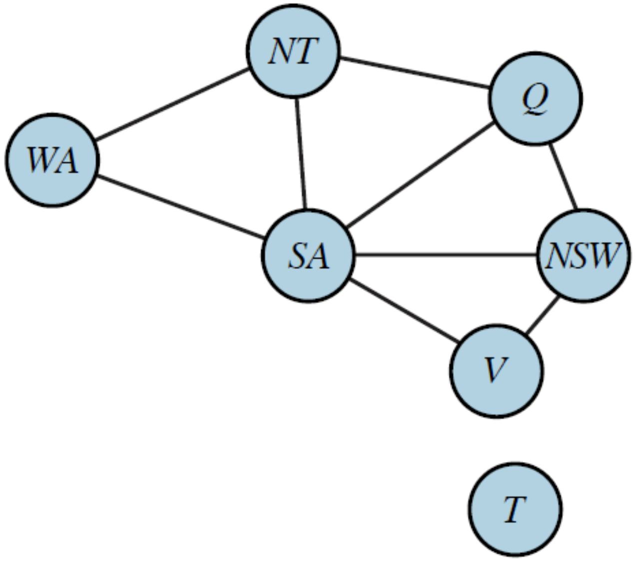
Application of Path Consistency



Make {*WA*, *SA*} path-consistent w.r.t. *NT*.

Assume two colors only: *red*, *blue*.

Application of Path Consistency



Make {*WA*, *SA*} path-consistent w.r.t. *NT*.

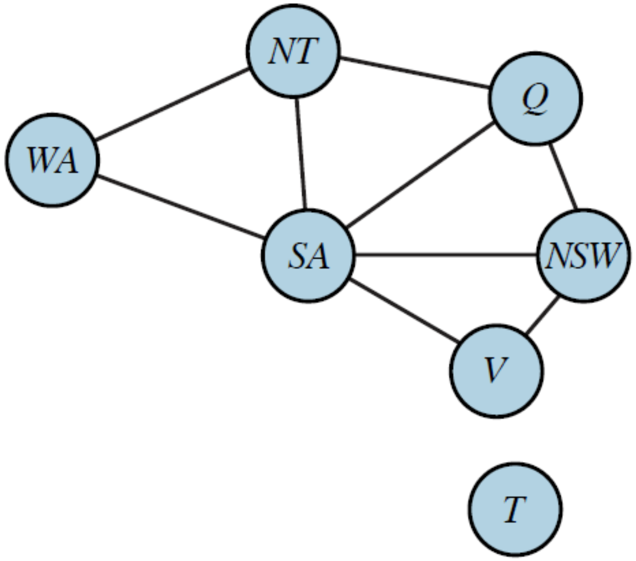
Assume two colors only: *red*, *blue*.

Only two possible assignments:

{*WA* = *red*, *SA* = *blue*}

{*WA* = *blue*, *SA* = *red*}

Application of Path Consistency



Make {*WA*, *SA*} path-consistent w.r.t. *NT*.

Assume two colors only: *red*, *blue*.

Only two possible assignments:

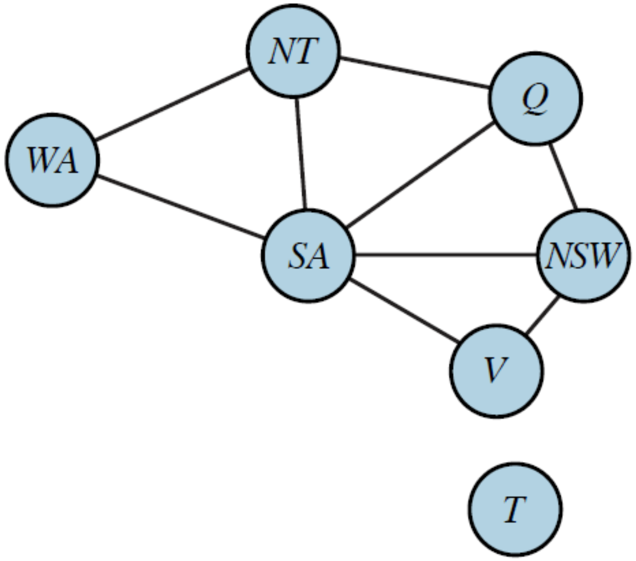
{*WA* = *red*, *SA* = *blue*}

{*WA* = *blue*, *SA* = *red*}

$\Downarrow$

No assignment for *NT* in either case!

Application of Path Consistency



Make {*WA*, *SA*} path-consistent w.r.t. *NT*.

Assume two colors only: *red*, *blue*.

Only two possible assignments:

{*WA* = *red*, *SA* = *blue*}

{*WA* = *blue*, *SA* = *red*}

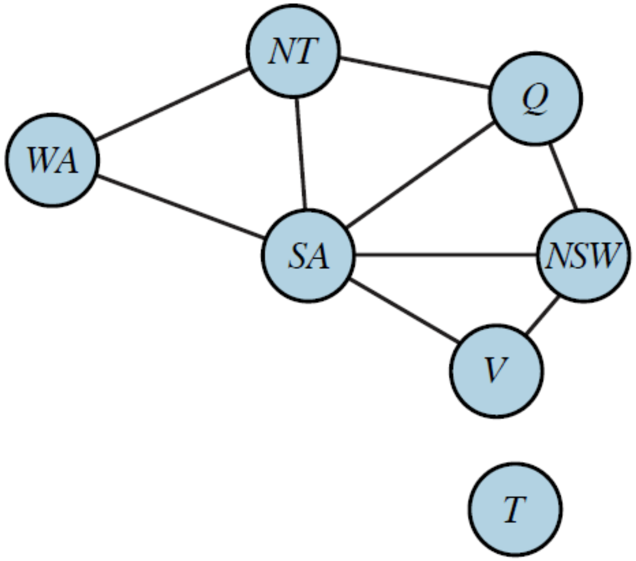
⇩

No assignment for *NT* in either case!

⇩

Eliminate both assignments to *WA* and *SA*.

Application of Path Consistency



Make {*WA*, *SA*} path-consistent w.r.t. *NT*.

Assume two colors only: *red*, *blue*.

Only two possible assignments:

{*WA* = *red*, *SA* = *blue*}

{*WA* = *blue*, *SA* = *red*}

⇩

No assignment for *NT* in either case!

⇩

Eliminate both assignments to *WA* and *SA*.

⇩

No solution to the problem.

Bounds Propagation

Two flights F_1 and F_2 have domains:

$$D_1 = [0,165] \text{ and } D_2 = [0,385]$$

max capacity

Bounds Propagation

Two flights F_1 and F_2 have domains:

$$D_1 = [0{,}165] \text{ and } D_2 = [0{,}385]$$

max capacity

Constraint: the two flights together carry 420 people.

Bounds Propagation

Two flights F_1 and F_2 have domains:

$$D_1 = [0,165] \text{ and } D_2 = [0,385]$$

max capacity

Constraint: the two flights together carry 420 people.

$\Downarrow$ domain reduction

$$D_1 = [420 - 385,165] = [35,165]$$
$$D_2 = [420 - 165,385] = [255,385]$$

Bounds Propagation

Two flights F_1 and F_2 have domains:

$$D_1 = [0{,}165] \text{ and } D_2 = [0{,}385]$$

max capacity

Constraint: the two flights together carry 420 people.

$\Downarrow$ domain reduction

$$D_1 = [420 - 385{,}165] = [35{,}165]$$
$$D_2 = [420 - 165{,}385] = [255{,}385]$$

Bounds-consistent if for any two variables X, Y, whether X takes on its lower- or upper-bound values, there exists some value of Y that satisfies the constraint between X and Y.

Bounds Propagation

Two flights F_1 and F_2 have domains:

$$D_1 = [0,165] \text{ and } D_2 = [0,385]$$

max capacity

Constraint: the two flights together carry 420 people.

$\Downarrow$ domain reduction

$$D_1 = [420 - 385,165] = [35,165]$$
$$D_2 = [420 - 165,385] = [255,385]$$

Bounds-consistent if for any two variables X, Y, whether X takes on its lower- or upper-bound values, there exists some value of Y that satisfies the constraint between X and Y.

* By the above definition, for some value of X between its lower and upper bounds, there may not exist a value of Y to meet the constraint (e.g., a nonlinear constraint).

Bounds Propagation

Two flights F_1 and F_2 have domains:

$D_1 = [0,165]$ and $D_2 = [0,385]$

max capacity

Constraint: the two flights together carry 420 people.

$\Downarrow$ domain reduction

$$D_1 = [420 - 385,165] = [35,165]$$
$$D_2 = [420 - 165,385] = [255,385]$$

Bounds-consistent if for any two variables X, Y, whether X takes on its lower- or upper-bound values, there exists some value of Y that satisfies the constraint between X and Y.

* By the above definition, for some value of X between its lower and upper bounds, there may not exist a value of Y to meet the constraint (e.g., a nonlinear constraint).

Sudoku

Fill the digits 1 to 9 in a 9×9 grid such that no digit appears twice in any row, column, or 3×3 box.

	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		

Sudoku

Fill the digits 1 to 9 in a 9×9 grid such that no digit appears twice in any row, column, or 3×3 box.

	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		

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

Sudoku as a CSP

	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		

Variables: $A1, \dots, A9, B1, \dots I9$

Domain: $D = \{1,2,\dots,9\}$

Sudoku as a CSP

	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		

Variables: $A1, \dots, A9, B1, \dots I9$

Domain: $D = \{1,2, \dots, 9\}$

27 *Alldiff* constraints:

Sudoku as a CSP

	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		

Variables: $A1, \dots, A9, B1, \dots I9$

Domain: $D = \{1,2, \dots, 9\}$

27 *Alldiff* constraints:

Alldiff$(A1, A2, A3, A4, A5, A6, A7, A8, A9)$

$\vdots$

Alldiff$(I1, I2, I3, I4, I5, I6, I7, I8, I9)$

Sudoku as a CSP

	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		

Variables: $A1, \dots, A9, B1, \dots I9$

Domain: $D = \{1, 2, \dots, 9\}$

27 *Alldiff* constraints:

Alldiff$(A1, A2, A3, A4, A5, A6, A7, A8, A9)$

$\vdots$

Alldiff$(I1, I2, I3, I4, I5, I6, I7, I8, I9)$

Alldiff$(A1, B1, C1, D1, E1, F1, G1, H1, I1)$

$\vdots$

Alldiff$(A9, B9, C9, D9, E9, F9, G9, H9, I9)$

Sudoku as a CSP

	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		

Variables: $A1, \ldots, A9, B1, \ldots I9$

Domain: $D = \{1,2, \ldots, 9\}$

27 *Alldiff* constraints:

Alldiff$(A1, A2, A3, A4, A5, A6, A7, A8, A9)$

$\vdots$

Alldiff$(I1, I2, I3, I4, I5, I6, I7, I8, I9)$

Alldiff$(A1, B1, C1, D1, E1, F1, G1, H1, I1)$

$\vdots$

Alldiff$(A9, B9, C9, D9, E9, F9, G9, H9, I9)$

Alldiff$(A1, A2, A3, B1, B2, B3, C1, C2, C3)$

$\vdots$

Alldiff$(A7, A8, A9, B7, B8, B9, C7, C8, C9)$

Sudoku as a CSP

	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		

Variables: $A1, \ldots, A9, B1, \ldots I9$

Domain: $D = \{1,2, \ldots, 9\}$

27 *Alldiff* constraints:

Alldiff$(A1, A2, A3, A4, A5, A6, A7, A8, A9)$

$\vdots$

Alldiff$(I1, I2, I3, I4, I5, I6, I7, I8, I9)$

Alldiff$(A1, B1, C1, D1, E1, F1, G1, H1, I1)$

$\vdots$

Alldiff$(A9, B9, C9, D9, E9, F9, G9, H9, I9)$

Alldiff$(A1, A2, A3, B1, B2, B3, C1, C2, C3)$

$\vdots$

Alldiff$(A7, A8, A9, B7, B8, B9, C7, C8, C9)$

A CSP solver can handle thousands of puzzles per second!

Sudoku as a CSP

	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		

Variables: $A1, \ldots, A9, B1, \ldots I9$

Domain: $D = \{1,2, \ldots, 9\}$

27 *Alldiff* constraints:

Alldiff$(A1, A2, A3, A4, A5, A6, A7, A8, A9)$

$\vdots$

Alldiff$(I1, I2, I3, I4, I5, I6, I7, I8, I9)$

Alldiff$(A1, B1, C1, D1, E1, F1, G1, H1, I1)$

$\vdots$

Alldiff$(A9, B9, C9, D9, E9, F9, G9, H9, I9)$

Alldiff$(A1, A2, A3, B1, B2, B3, C1, C2, C3)$

$\vdots$

Alldiff$(A7, A8, A9, B7, B8, B9, C7, C8, C9)$

A CSP solver can handle thousands of puzzles per second!
Only the simplest ones can be solved by AC-3.

Example of CSP Solution

	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		

Example of CSP Solution

	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		

Consider E6:

Example of CSP Solution

	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		

Consider E6:

$$D_{E6} \leftarrow \{1,2, \ldots, 9\}$$

Example of CSP Solution

	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		

Consider E6:

$$D_{E6} \leftarrow \{1,2,\ldots,9\}$$

$\Downarrow$ constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$

$$= \{3,4,5,6,9\}$$

Example of CSP Solution

	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		

Consider E6:

$$D_{E6} \leftarrow \{1,2,\ldots,9\}$$

$\Downarrow$ constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$

$$= \{3,4,5,6,9\}$$

$\Downarrow$ constraints in the column

$$D_{E6} \leftarrow D_{E6} \setminus \{2,3,5,6,8,9\}$$

$$= \{4\}$$

Example of CSP Solution

	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					4			8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

Consider E6:

$$D_{E6} \leftarrow \{1,2,\ldots,9\}$$

$\Downarrow$ constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$

$$= \{3,4,5,6,9\}$$

$\Downarrow$ constraints in the column

$$D_{E6} \leftarrow D_{E6} \setminus \{2,3,5,6,8,9\}$$

$$= \{4\}$$

Example of CSP Solution

	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					4			8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1	?	3		

Consider E6:

$$D_{E6} \leftarrow \{1,2,\ldots,9\}$$

$\Downarrow$ constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$

$$= \{3,4,5,6,9\}$$

$\Downarrow$ constraints in the column

$$D_{E6} \leftarrow D_{E6} \setminus \{2,3,5,6,8,9\}$$

$$= \{4\}$$

Consider I6:

Example of CSP Solution

	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					4			8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1	?	3		

Consider E6:

$$D_{E6} \leftarrow \{1,2,\dots,9\}$$

constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$

$$= \{3,4,5,6,9\}$$

constraints in the column

$$D_{E6} \leftarrow D_{E6} \setminus \{2,3,5,6,8,9\}$$

$$= \{4\}$$

Consider I6: $D_{I6} \leftarrow \{1,2,\dots,9\}$

Example of CSP Solution

	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					4			8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1	?	3		

Consider E6:

$$D_{E6} \leftarrow \{1,2,\dots,9\}$$

$\Downarrow$ constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$

$$= \{3,4,5,6,9\}$$

$\Downarrow$ constraints in the column

$$D_{E6} \leftarrow D_{E6} \setminus \{2,3,5,6,8,9\}$$

$$= \{4\}$$

Consider I6: $D_{I6} \leftarrow \{1,2,\dots,9\}$

$\Downarrow$ constraints in the column

$$D_{I6} \leftarrow D_{I6} \setminus \{2,3,4,5,6,8,9\} = \{1,7\}$$

Example of CSP Solution

	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					4			8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1	?	3		

Consider E6:

$$D_{E6} \leftarrow \{1,2,\ldots,9\}$$

$\Downarrow$ constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$

$$= \{3,4,5,6,9\}$$

$\Downarrow$ constraints in the column

$$D_{E6} \leftarrow D_{E6} \setminus \{2,3,5,6,8,9\}$$

$$= \{4\}$$

Consider I6:

$$D_{I6} \leftarrow \{1,2,\ldots,9\}$$

$\Downarrow$ constraints in the column

$$D_{I6} \leftarrow D_{I6} \setminus \{2,3,4,5,6,8,9\} = \{1,7\}$$

$\Downarrow$ constraints in the box

$$D_{I6} \leftarrow D_{I6} \setminus \{1,2,3,6,9\} = \{7\}$$

Example of CSP Solution

	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					4			8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1	7	3		

Consider E6:

$$D_{E6} \leftarrow \{1,2,\dots,9\}$$

$\Downarrow$ constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$

$$= \{3,4,5,6,9\}$$

$\Downarrow$ constraints in the column

$$D_{E6} \leftarrow D_{E6} \setminus \{2,3,5,6,8,9\}$$

$$= \{4\}$$

Consider I6:

$$D_{I6} \leftarrow \{1,2,\dots,9\}$$

$\Downarrow$ constraints in the column

$$D_{I6} \leftarrow D_{I6} \setminus \{2,3,4,5,6,8,9\} = \{1,7\}$$

$\Downarrow$ constraints in the box

$$D_{I6} \leftarrow D_{I6} \setminus \{1,2,3,6,9\} = \{7\}$$

Example of CSP Solution

	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					4			8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1	7	3		

Consider E6:

$$D_{E6} \leftarrow \{1,2,\dots,9\}$$

$\Downarrow$ constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$

$$= \{3,4,5,6,9\}$$

$\Downarrow$ constraints in the column

$$D_{E6} \leftarrow D_{E6} \setminus \{2,3,5,6,8,9\}$$

$$= \{4\}$$

Consider A6:

Consider I6:

$$D_{I6} \leftarrow \{1,2,\dots,9\}$$

$\Downarrow$ constraints in the column

$$D_{I6} \leftarrow D_{I6} \setminus \{2,3,4,5,6,8,9\} = \{1,7\}$$

$\Downarrow$ constraints in the box

$$D_{I6} \leftarrow D_{I6} \setminus \{1,2,3,6,9\} = \{7\}$$

Example of CSP Solution

	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					4			8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1	7	3		

Consider E6:

$$D_{E6} \leftarrow \{1,2,\ldots,9\}$$

$\Downarrow$ constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$
$$= \{3,4,5,6,9\}$$

$\Downarrow$ constraints in the column

$$D_{E6} \leftarrow D_{E6} \setminus \{2,3,5,6,8,9\}$$
$$= \{4\}$$

Consider A6:

$$D_{A6} \leftarrow \{1,2,\ldots,9\}$$

Consider I6:

$$D_{I6} \leftarrow \{1,2,\ldots,9\}$$

$\Downarrow$ constraints in the column

$$D_{I6} \leftarrow D_{I6} \setminus \{2,3,4,5,6,8,9\} = \{1,7\}$$

$\Downarrow$ constraints in the box

$$D_{I6} \leftarrow D_{I6} \setminus \{1,2,3,6,9\} = \{7\}$$

Example of CSP Solution

	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					4			8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1	7	3		

Consider E6:

$$D_{E6} \leftarrow \{1,2,\dots,9\}$$

$\Downarrow$ constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$
$$= \{3,4,5,6,9\}$$

$\Downarrow$ constraints in the column

$$D_{E6} \leftarrow D_{E6} \setminus \{2,3,5,6,8,9\}$$
$$= \{4\}$$

Consider I6:

$$D_{I6} \leftarrow \{1,2,\dots,9\}$$

$\Downarrow$ constraints in the column

$$D_{I6} \leftarrow D_{I6} \setminus \{2,3,4,5,6,8,9\} = \{1,7\}$$

$\Downarrow$ constraints in the box

$$D_{I6} \leftarrow D_{I6} \setminus \{1,2,3,6,9\} = \{7\}$$

Consider A6:

$$D_{A6} \leftarrow \{1,2,\dots,9\}$$

$\Downarrow$ constraints in the column

$$D_{A6} \leftarrow \{1\}$$

Example of CSP Solution

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

Consider E6:

$$D_{E6} \leftarrow \{1,2,\dots,9\}$$

$\Downarrow$ constraints in the box

$$D_{E6} \leftarrow D_{E6} \setminus \{1,2,7,8\}$$

$$= \{3,4,5,6,9\}$$

$\Downarrow$ constraints in the column

$$D_{E6} \leftarrow D_{E6} \setminus \{2,3,5,6,8,9\}$$

$$= \{4\}$$

Consider I6:

$$D_{I6} \leftarrow \{1,2,\dots,9\}$$

$\Downarrow$ constraints in the column

$$D_{I6} \leftarrow D_{I6} \setminus \{2,3,4,5,6,8,9\} = \{1,7\}$$

$\Downarrow$ constraints in the box

$$D_{I6} \leftarrow D_{I6} \setminus \{1,2,3,6,9\} = \{7\}$$

Consider A6:

$$D_{A6} \leftarrow \{1,2,\dots,9\}$$

$\Downarrow$ constraints in the column

$$D_{A6} \leftarrow \{1\}$$

III. Backtracking Search

- Constraint propagation often ends with partial solutions.
- Backtracking search can be employed to extend them to full solutions.

III. Backtracking Search

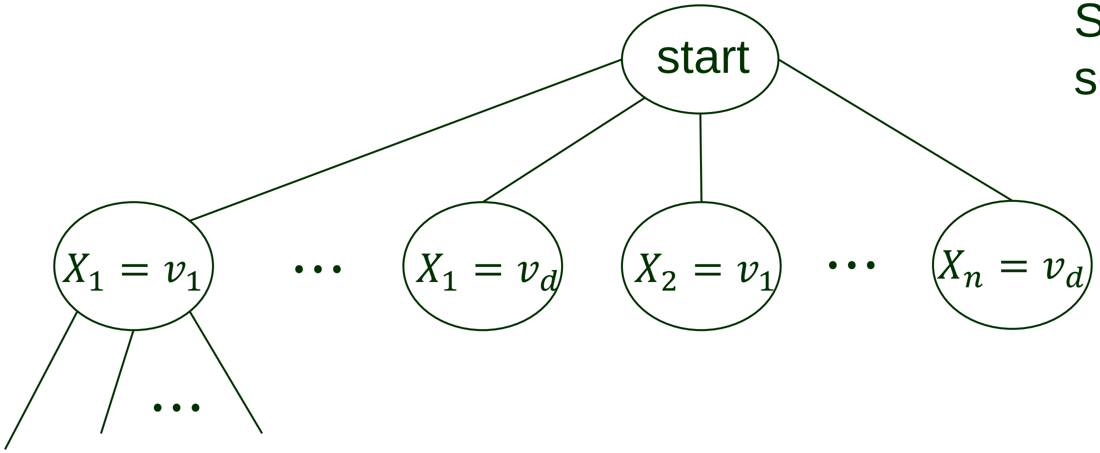
- Constraint propagation often ends with partial solutions.
- Backtracking search can be employed to extend them to full solutions.

Solve a CSP using depth-limited search.

n variables of domain size d

III. Backtracking Search

- Constraint propagation often ends with partial solutions.
- Backtracking search can be employed to extend them to full solutions.

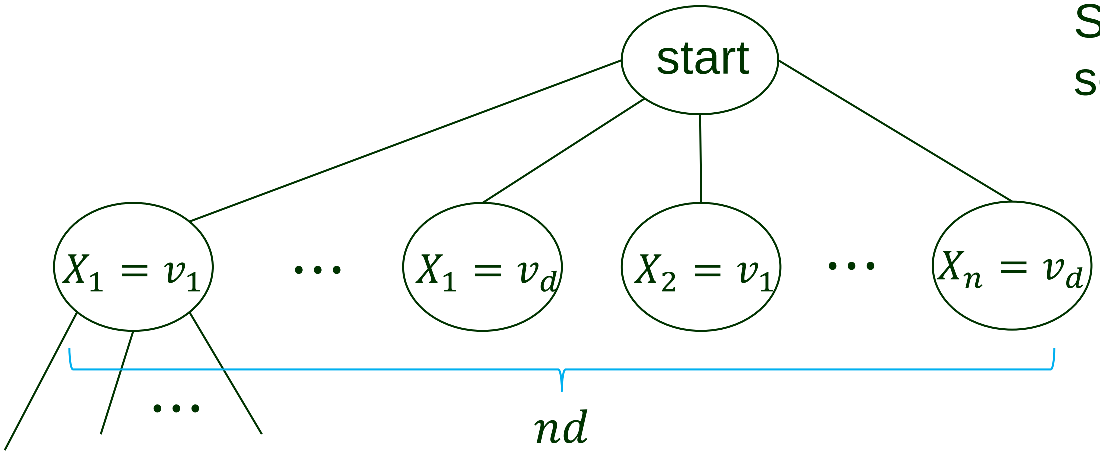


Solve a CSP using depth-limited search.

n variables of domain size d

III. Backtracking Search

- Constraint propagation often ends with partial solutions.
- Backtracking search can be employed to extend them to full solutions.

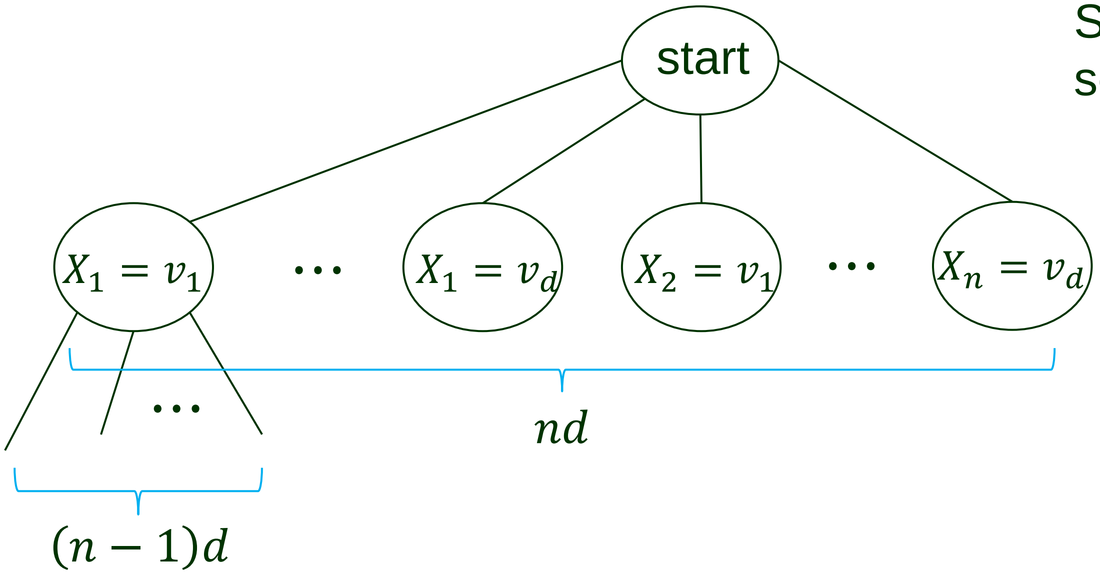


Solve a CSP using depth-limited search.

n variables of domain size d

III. Backtracking Search

- Constraint propagation often ends with partial solutions.
- Backtracking search can be employed to extend them to full solutions.

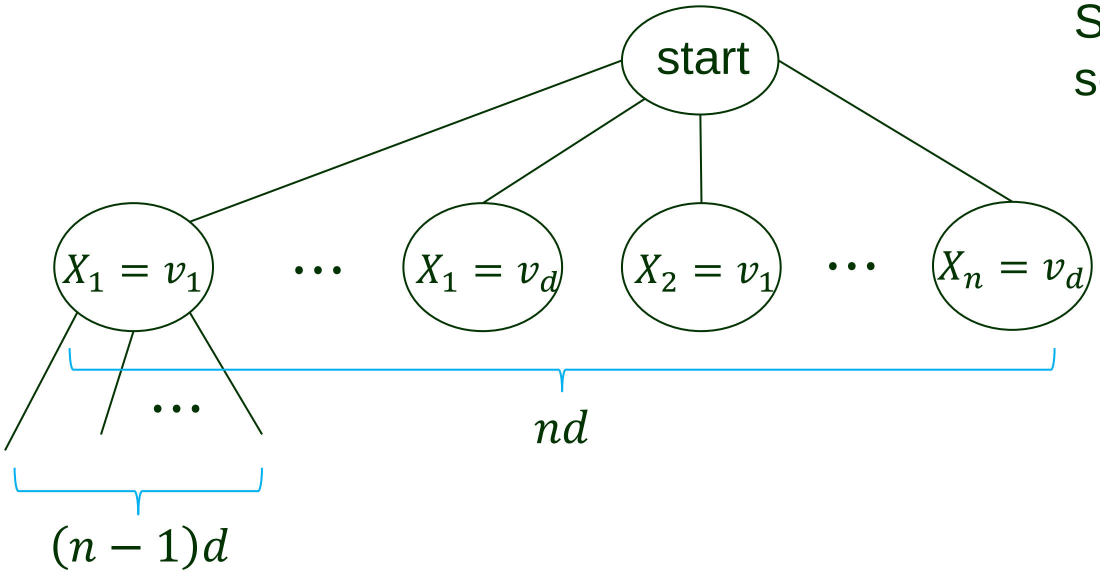


Solve a CSP using depth-limited search.

n variables of
domain size d

III. Backtracking Search

- Constraint propagation often ends with partial solutions.
- Backtracking search can be employed to extend them to full solutions.



Solve a CSP using depth-limited search.

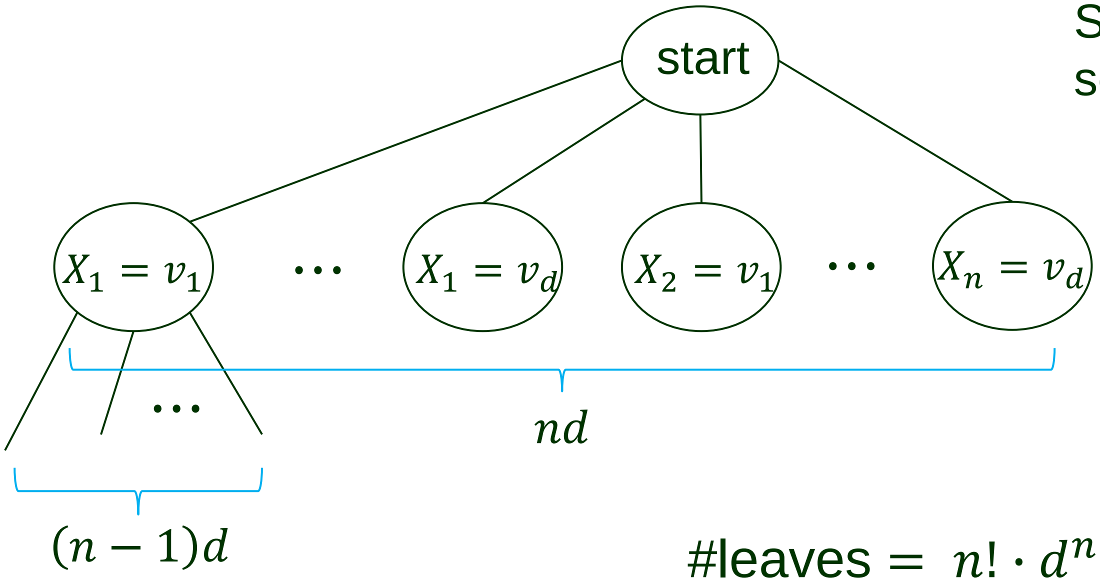
n variables of domain size d

Branching factor at depth i:

$$(n - i + 1)!$$

III. Backtracking Search

- Constraint propagation often ends with partial solutions.
- Backtracking search can be employed to extend them to full solutions.



Solve a CSP using depth-limited search.

n variables of
domain size d

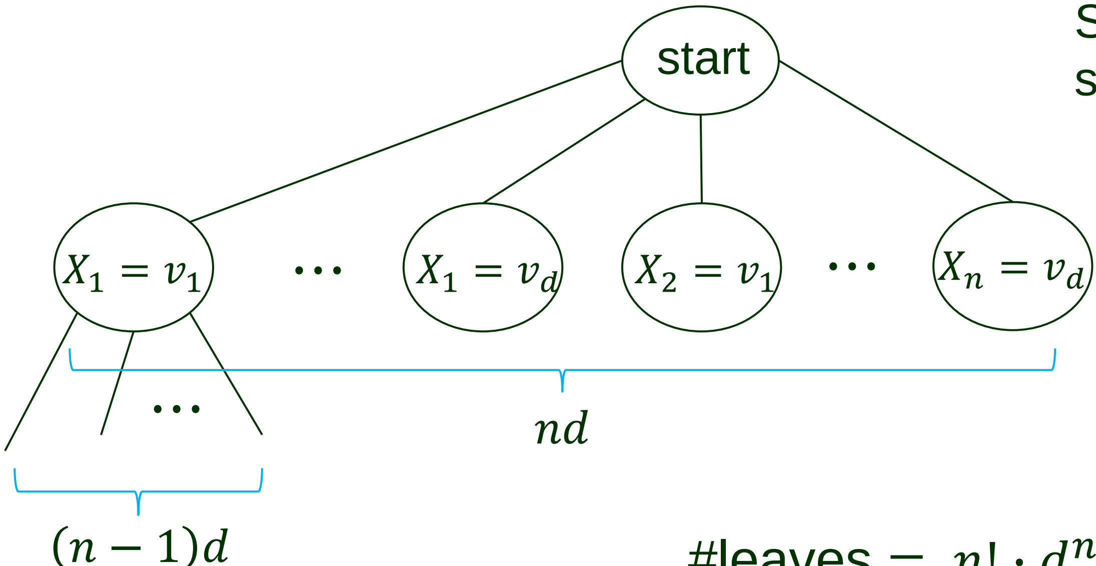
Branching factor at depth i:

$$(n-i+1)!$$

#leaves = $n! \cdot d^n$

III. Backtracking Search

- Constraint propagation often ends with partial solutions.
- Backtracking search can be employed to extend them to full solutions.



Solve a CSP using depth-limited search.

n variables of
domain size d

Branching factor at depth i:

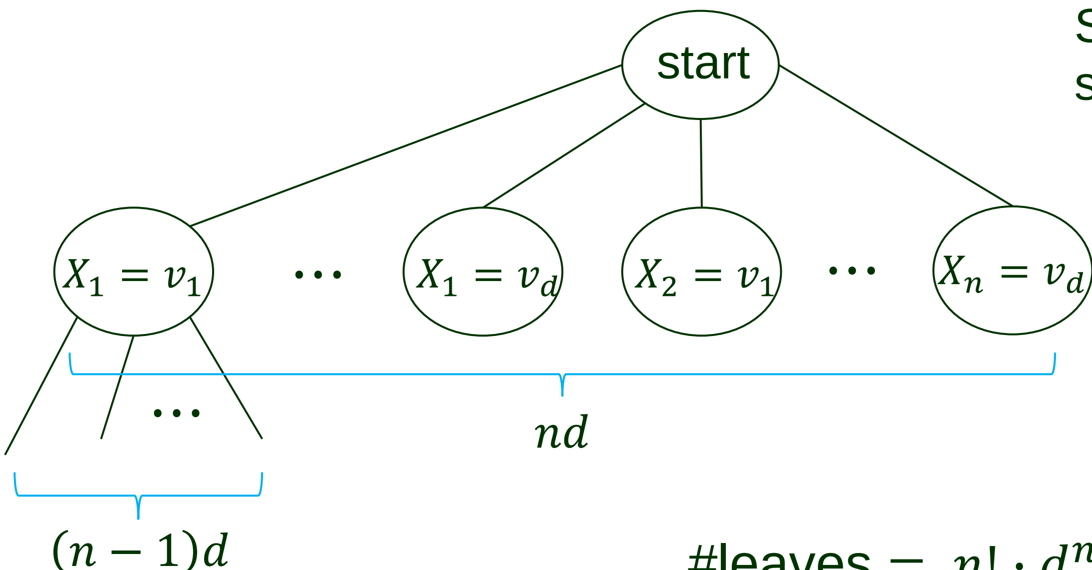
$$(n-i+1)!$$

$$\#\text{leaves} = n! \cdot d^n$$

But #assignments $= d^n$.

III. Backtracking Search

- Constraint propagation often ends with partial solutions.
- Backtracking search can be employed to extend them to full solutions.



Solve a CSP using depth-limited search.

n variables of
domain size d

Branching factor at depth i:

$$(n-i+1)!$$

#leaves = $n! \cdot d^n$

But #assignments = d^n.

How to get back to d^n?

Commutativity of CSP

A problem is *commutative* if the order of application of any given set of actions does not matter.

Commutativity of CSP

A problem is *commutative* if the order of application of any given set of actions does not matter.

assignments in a CSP

Commutativity of CSP

A problem is *commutative* if the order of application of any given set of actions does not matter.

assignments in a CSP

No difference between

Step 1: NSW = *red*
Step 2: SA = *blue*

and

Step 1: SA = *blue*
Step 2: NSW = *red*

Commutativity of CSP

A problem is *commutative* if the order of application of any given set of actions does not matter.

assignments in a CSP

No difference between

Step 1: NSW = *red* Step 2: SA = *blue*	and	Step 1: SA = *blue* Step 2: NSW = *red*

Need only consider a single variable at each node.

Commutativity of CSP

A problem is *commutative* if the order of application of any given set of actions does not matter.

assignments in a CSP

No difference between

Step 1: NSW = *red*
Step 2: SA = *blue*

and

Step 1: SA = *blue*
Step 2: NSW = *red*

Need only consider a single variable at each node.

At the root choose between

SA = *blue*, SA = *red*, and SA = *green*

Commutativity of CSP

A problem is *commutative* if the order of application of any given set of actions does not matter.

assignments in a CSP

No difference between

Step 1: *NSW* = *red*
Step 2: *SA* = *blue*

and

Step 1: *SA* = *blue*
Step 2: *NSW* = *red*

Need only consider a single variable at each node.

At the root choose between

SA = *blue*, *SA* = *red*, and *SA* = *green*

but not between

SA = *blue* and *NSW* = *red*

Backtracking Algorithm

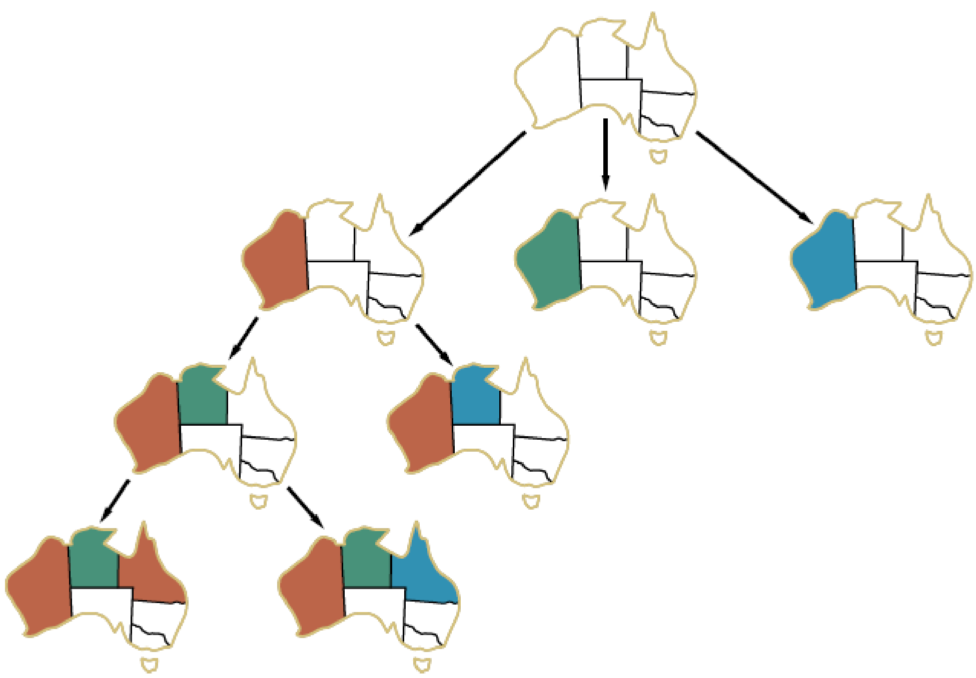
- Repeatedly chooses an unassigned variable X_i.
- Tries all values $v_j \in D_i$ (its domain).

  - Add $X_i = v_j$ to the partial solution (after consistency checking).

  - Try to extend it into a solution via a recursive call (in which another unassigned variable will be considered)

Backtracking Algorithm

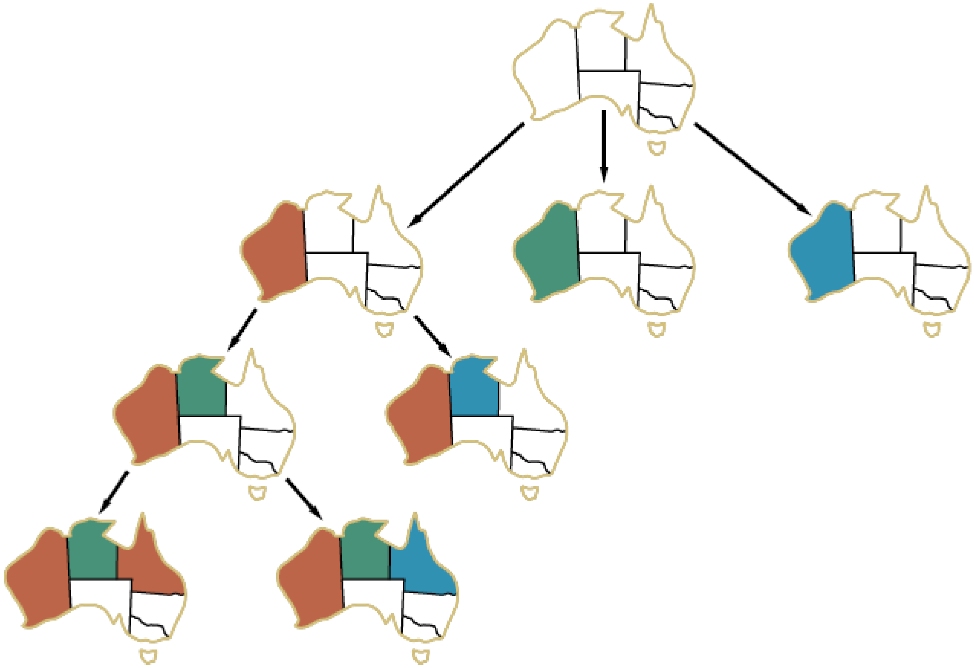
- Repeatedly chooses an unassigned variable X_i.
- Tries all values $v_j \in D_i$ (its domain).



- Add $X_i = v_j$ to the partial solution (after consistency checking).
- Try to extend it into a solution via a recursive call (in which another unassigned variable will be considered)

Backtracking Algorithm

- Repeatedly chooses an unassigned variable X_i.
- Tries all values $v_j \in D_i$ (its domain).



- Add $X_i = v_j$ to the partial solution (after consistency checking).
- Try to extend it into a solution via a recursive call (in which another unassigned variable will be considered)