Constraint Satisfaction Problems (CSPs)

Outline

I. Job shop scheduling

II. The diet problem and linear programming

III. Binary CSPs

IV. Constraint Propagation

* Figures/images are from the textbook site (or by the instructor). Otherwise, the source is cited unless such citation would make little sense due to the triviality of generating the image.

I. Example 2: Job-Shop Scheduling

Car assembly with 15 tasks:

- install axles (front and back): 2
- affix wheels (right and left, front and back): 4
- tighten nuts for each wheel: 4
- affix hubcaps: 4
- inspect the final assembly: 1

I. Example 2: Job-Shop Scheduling

Car assembly with 15 tasks:

- install axles (front and back): 2
- affix wheels (right and left, front and back): 4
- tighten nuts for each wheel: 4
- affix hubcaps: 4
- inspect the final assembly: 1

$$\mathcal{X} = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{LF}, Wheel_{RB}, Wheel_{LB}, Nuts_{RF}, Nuts_{LF}, Nuts_{RB}, Nuts_{LB}, Cap_{RF}, Cap_{LF}, Cap_{RB}, Cap_{LB}, Inspect\}.$$

I. Example 2: Job-Shop Scheduling

Car assembly with 15 tasks:

- install axles (front and back): 2
- affix wheels (right and left, front and back): 4
- tighten nuts for each wheel: 4
- affix hubcaps: 4
- inspect the final assembly: 1

$$\mathcal{X} = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{LF}, Wheel_{RB}, Wheel_{LB}, Nuts_{RF}, Nuts_{LF}, Nuts_{RB}, Nuts_{LB}, Cap_{RF}, Cap_{LF}, Cap_{RB}, Cap_{LB}, Inspect\}.$$

$\mathcal{D}$: time space for each variable.

I. Example 2: Job-Shop Scheduling

Car assembly with 15 tasks:

- install axles (front and back): 2
- affix wheels (right and left, front and back): 4
- tighten nuts for each wheel: 4
- affix hubcaps: 4
- inspect the final assembly: 1

$$\mathcal{X} = \{Axle_F, Axle_B, Wheel_{RF}, Wheel_{LF}, Wheel_{RB}, Wheel_{LB}, Nuts_{RF}, Nuts_{LF}, Nuts_{RB}, Nuts_{LB}, Cap_{RF}, Cap_{LF}, Cap_{RB}, Cap_{LB}, Inspect\}.$$

$\mathcal{D}$: time space for each variable.

$Axle_F =$ starting time for installation of the front axle.

Precedence Constraints

$$T_1 + d_1 \leq T_2$$

starting time of task T_1 duration of task T_1

Precedence Constraints

$$T_1 + d_1 \leq T_2$$

starting time of task T_1 duration of task T_1

- The axles have to be in place before the wheels are put on (axle installation takes 10 minutes).

$$Axle_F + 10 \leq Wheel_{RF} \qquad Axle_F + 10 \leq Wheel_{LF}$$

$$Axle_B + 10 \leq Wheel_{RB} \qquad Axle_B + 10 \leq Wheel_{LB}$$

Precedence Constraints

$$T_1 + d_1 \leq T_2$$

starting time of task T_1 — duration of task T_1

- ♣ The axles have to be in place before the wheels are put on (axle installation takes 10 minutes).

$$Axle_F + 10 \leq Wheel_{RF} \qquad Axle_F + 10 \leq Wheel_{LF}$$

$$Axle_B + 10 \leq Wheel_{RB} \qquad Axle_B + 10 \leq Wheel_{LB}$$

- ♣ Affix each wheel (1 minutes), then tighten the nuts (2 minutes), and finally attach the hubcap (1 minute, but not represented)

$$Wheel_{RF} + 1 \leq Nuts_{RF} \qquad Nuts_{RF} + 2 \leq Cap_{RF}$$

$$Wheel_{LF} + 1 \leq Nuts_{LF} \qquad Nuts_{LF} + 2 \leq Cap_{LF}$$

$$Wheel_{RB} + 1 \leq Nuts_{RB} \qquad Nuts_{RB} + 2 \leq Cap_{RB}$$

$$Wheel_{LB} + 1 \leq Nuts_{LB} \qquad Nuts_{LB} + 2 \leq Cap_{LB}$$

More Constraints

- Inspection comes last and take 3 minutes

$X + d_X \leq Inspect$ for every $X \in \mathcal{X}$

duration of task X

More Constraints

- ♣ Inspection comes last and take 3 minutes

$$X + d_X \leq Inspect \quad \text{for every } X \in \mathcal{X}$$

duration of task X

- ♣ The whole assembly must be finished in 30 minutes

More Constraints

- Inspection comes last and take 3 minutes

$$X + d_X \leq Inspect \quad \text{for every } X \in \mathcal{X}$$

duration of task X

- The whole assembly must be finished in 30 minutes

Limit the domain of all variables (discretization)

$$\mathcal{D} = \{1, 2, \ldots, 27\}$$

Disjunctive Constraint

Four workers installing wheels have to share one tool for axle installment.

Disjunctive Constraint

Four workers installing wheels have to share one tool for axle installment.

⇓

$Axle_F$ and $Axle_B$ must not overlap in time.

Disjunctive Constraint

Four workers installing wheels have to share one tool for axle installment.

$\Downarrow$

$Axle_F$ and $Axle_B$ must not overlap in time.

$\Downarrow$

$(Axle_F + 10 \leq Axle_B)$ or $(Axle_B + 10 \leq Axle_F)$

Discrete and Continuous Domains

- Discrete, finite domains:

  Map coloring, 8-queens, scheduling (with time limits).

Discrete and Continuous Domains

- Discrete, finite domains:

  Map coloring, 8-queens, scheduling (with time limits).

- Discrete, infinite domains:

  Implicit constraints only, e.g., $T_1 + d_1 \leq T_2$

Discrete and Continuous Domains

- Discrete, finite domains:

  Map coloring, 8-queens, scheduling (with time limits).

- Discrete, infinite domains:

  Implicit constraints only, e.g., $T_1 + d_1 \leq T_2$

- Continuous domains:

  Scheduling of experiments on the Hubble Telescope,

Discrete and Continuous Domains

- Discrete, finite domains:

  Map coloring, 8-queens, scheduling (with time limits).

- Discrete, infinite domains:

  Implicit constraints only, e.g., $T_1 + d_1 \leq T_2$

- Continuous domains:

  Scheduling of experiments on the Hubble Telescope,

  linear programming.

II. The Diet Problem*

How much money to spend in order to get what Polly needs every day?

- energy (2,000 kcal)
- protein (55 g)
- calcium (800 mg)

* V. Chvatal. *Linear Programming.* W. H. Freeman and Company, 1983.

II. The Diet Problem*

How much money to spend in order to get what Polly needs every day?

- energy (2,000 kcal)
- protein (55 g)
- calcium (800 mg)

Food	Serving size	Energy (kcal)	Protein (g)	Calcium (mg)	Price per serving (cents)
Oat meal	28 g	110	4	2	3
Chicken	100 g	205	32	12	24
Eggs	2 large	160	13	54	13
Whole milk	237 cc	160	8	285	9
Cherry pie	170 g	420	4	22	20
Pork with beans	260 g	260	14	80	19

* V. Chvatal. *Linear Programming.* W. H. Freeman and Company, 1983.

Daily Serving Limits

	Servings at most per day
Oatmeal	4
Chicken	3
Eggs	2
Milk	8
Cherry pie	2
Pork with beans	2

Task: Design the *most economical* menu.

Formulating the Problem

x_1: servings of oatmeal
x_2: servings of chicken
x_3: servings of eggs
x_4: servings of whole milk
x_5: servings of cherry pie
x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Formulating the Problem

x_1: servings of oatmeal
x_2: servings of chicken
x_3: servings of eggs
x_4: servings of whole milk
x_5: servings of cherry pie
x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

$$\min \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$$

subject to

and

Formulating the Problem

x_1: servings of oatmeal
x_2: servings of chicken
x_3: servings of eggs
x_4: servings of whole milk
x_5: servings of cherry pie
x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\min \quad \overbrace{3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6}$$

subject to

and

Formulating the Problem

x_1: servings of oatmeal
x_2: servings of chicken
x_3: servings of eggs
x_4: servings of whole milk
x_5: servings of cherry pie
x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\min \quad \underbrace{3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6}$$

$$\begin{aligned} \text{subject to} \quad & 0 \leq x_1 \leq 4 \\ & 0 \leq x_2 \leq 3 \\ & 0 \leq x_3 \leq 2 \\ & 0 \leq x_4 \leq 8 \\ & 0 \leq x_5 \leq 2 \\ & 0 \leq x_6 \leq 2 \\ \text{and} \quad & \end{aligned}$$

Formulating the Problem

x_1: servings of oatmeal
x_2: servings of chicken
x_3: servings of eggs
x_4: servings of whole milk
x_5: servings of cherry pie
x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\min \quad \overbrace{3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6}$$

$$\text{subject to} \quad \left.\begin{array}{l} 0 \le x_1 \le 4 \\ 0 \le x_2 \le 3 \\ 0 \le x_3 \le 2 \\ 0 \le x_4 \le 8 \\ 0 \le x_5 \le 2 \\ 0 \le x_6 \le 2 \end{array}\right\}$$

Servings-per-day limits

and

Formulating the Problem

x_1: servings of oatmeal
x_2: servings of chicken
x_3: servings of eggs
x_4: servings of whole milk
x_5: servings of cherry pie
x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\min \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$$

subject to

$$0 \le x_1 \le 4$$
$$0 \le x_2 \le 3$$
$$0 \le x_3 \le 2$$
$$0 \le x_4 \le 8$$
$$0 \le x_5 \le 2$$
$$0 \le x_6 \le 2$$

Servings-per-day limits

and

energy $\quad 110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 \ge 2000$

protein $\quad 4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 \ge 55$

calcium $\quad 2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 \ge 800$

Formulating the Problem

x_1: servings of oatmeal
x_2: servings of chicken
x_3: servings of eggs
x_4: servings of whole milk
x_5: servings of cherry pie
x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\min \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$$

$$\text{subject to} \quad 0 \le x_1 \le 4, \quad 0 \le x_2 \le 3, \quad 0 \le x_3 \le 2, \quad 0 \le x_4 \le 8, \quad 0 \le x_5 \le 2, \quad 0 \le x_6 \le 2$$

Servings-per-day limits

and

energy $\quad 110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 \ge 2000$

protein $\quad 4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 \ge 55$

calcium $\quad 2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 \ge 800$

Constraints (linear)

Formulating the Problem

x_1: servings of oatmeal
x_2: servings of chicken
x_3: servings of eggs
x_4: servings of whole milk
x_5: servings of cherry pie
x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\min \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$$

subject to

$$0 \le x_1 \le 4$$
$$0 \le x_2 \le 3$$
$$0 \le x_3 \le 2$$
$$0 \le x_4 \le 8$$
$$0 \le x_5 \le 2$$
$$0 \le x_6 \le 2$$

Servings-per-day limits

Linear Program!

and

energy $110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 \ge 2000$

protein $4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 \ge 55$

calcium $2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 \ge 800$

Constraints (linear)

Linear Programming (LP)

Max $c_1 x_1 + c_2 x_2 + \cdots + c_d\, x_d$

subject to
$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1d}x_d \leq b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2d}x_d \leq b_2$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nd}x_d \leq b_n$$

Linear Programming (LP)

Max $c_1 x_1 + c_2 x_2 + \cdots + c_d\, x_d$

subject to

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1d}x_d \le b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2d}x_d \le b_2$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nd}x_d \le b_n$$

$\boldsymbol{x} = (x_1, x_2, \dots, x_d)$

$\boldsymbol{c} = (c_1, c_2, \dots, c_d)$

$\boldsymbol{b} = (b_1, b_2, \dots, b_n)$

Linear Programming (LP)

Max $c_1 x_1 + c_2 x_2 + \cdots + c_d\, x_d \qquad = \boldsymbol{c}\boldsymbol{x}^{\boldsymbol{T}}$

subject to
$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1d}x_d \le b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2d}x_d \le b_2$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nd}x_d \le b_n$$

$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$

$\boldsymbol{c} = (c_1, c_2, \ldots, c_d)$

$\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$

Linear Programming (LP)

Max $c_1 x_1 + c_2 x_2 + \cdots + c_d\, x_d \qquad = \boldsymbol{c}\boldsymbol{x}^{\boldsymbol{T}}$

subject to

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ \cdots + a_{1d}x_d \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ \cdots + a_{2d}x_d \leq b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ \cdots + a_{nd}x_d \leq b_n \end{array}\right\} \quad A\boldsymbol{x}^T \leq \boldsymbol{b}^T$$

$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$

$\boldsymbol{c} = (c_1, c_2, \ldots, c_d)$

$\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$

Linear Programming (LP)

Max $c_1 x_1 + c_2 x_2 + \cdots + c_d\, x_d$ $= \boldsymbol{c}\boldsymbol{x}^T$

subject to

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1d}x_d \le b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2d}x_d \le b_2 \\ \quad\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nd}x_d \le b_n \end{array}\right\} \quad A\boldsymbol{x}^T \le \boldsymbol{b}^T$$

$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$

$\boldsymbol{c} = (c_1, c_2, \ldots, c_d)$

$\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$

Solvable in time polynomial in d.

Linear Programming (LP)

Max $c_1 x_1 + c_2 x_2 + \cdots + c_d\, x_d \qquad = \boldsymbol{c}\boldsymbol{x}^T$

subject to

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1d}x_d \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2d}x_d \leq b_2 \\ \qquad\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nd}x_d \leq b_n \end{array}\right\} \quad A\boldsymbol{x}^T \leq \boldsymbol{b}^T$$

$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$

$\boldsymbol{c} = (c_1, c_2, \ldots, c_d)$

$\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$

Solvable in time polynomial in d.

Simplex method $O(2^d)$

(best performance in practice)

III. Binary CSP

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

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

III. Binary CSP

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

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

A *binary constraint* relates two variables.

$$SA \neq WA$$

III. 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 $X > Y > Z \rangle$

III. 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).

III. 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$(v_1, \dots, v_k)$: variables $v_1, \dots, v_k$ must have different values.

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

Cryptarithmetic Puzzle

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

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

	T	W	O
$+$	T	W	O
F	O	U	R

Addition constraints on the four columns:

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

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

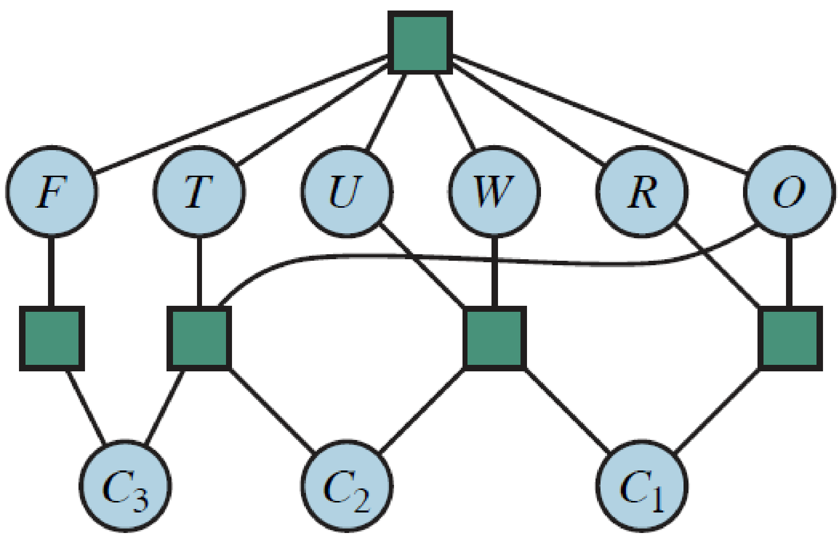
Cryptarithmetic Puzzle

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

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.



Constraint hypergraph

Cryptarithmetic Puzzle

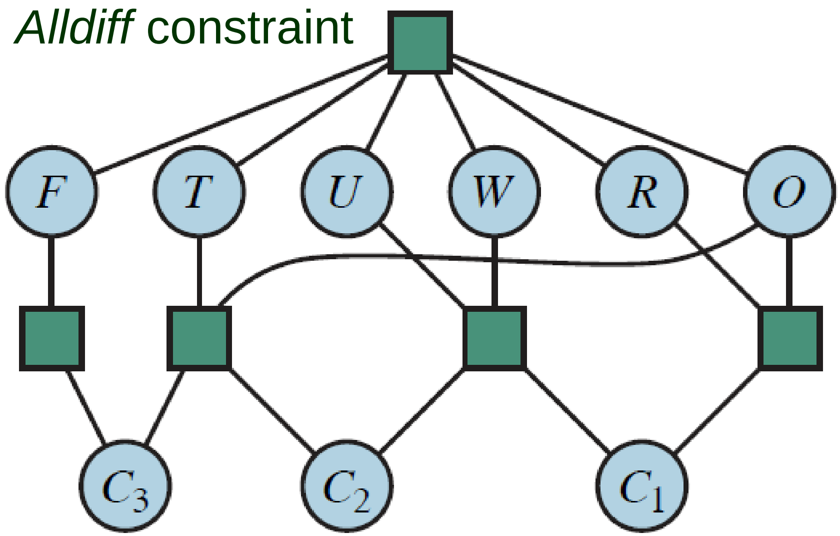
$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$

Addition constraints on the four columns:

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

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

Alldiff constraint



Constraint hypergraph

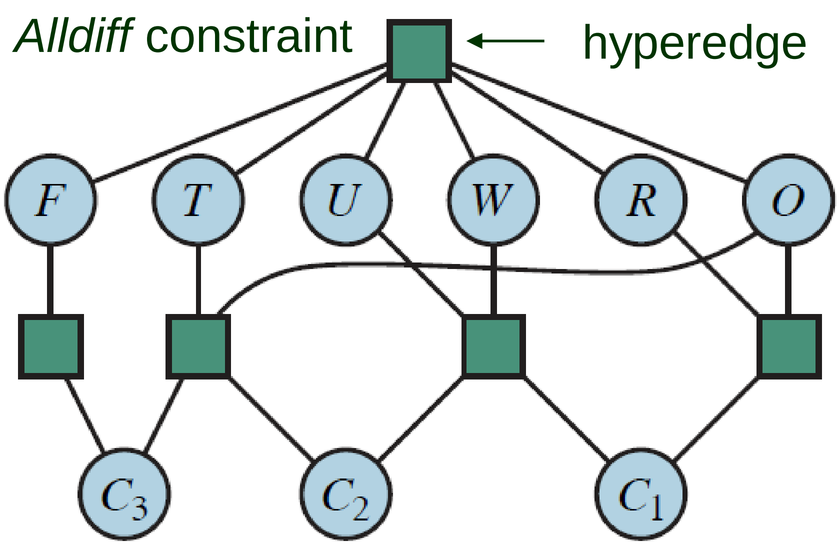
Cryptarithmetic Puzzle

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

Addition constraints on the four columns:

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

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



Constraint hypergraph

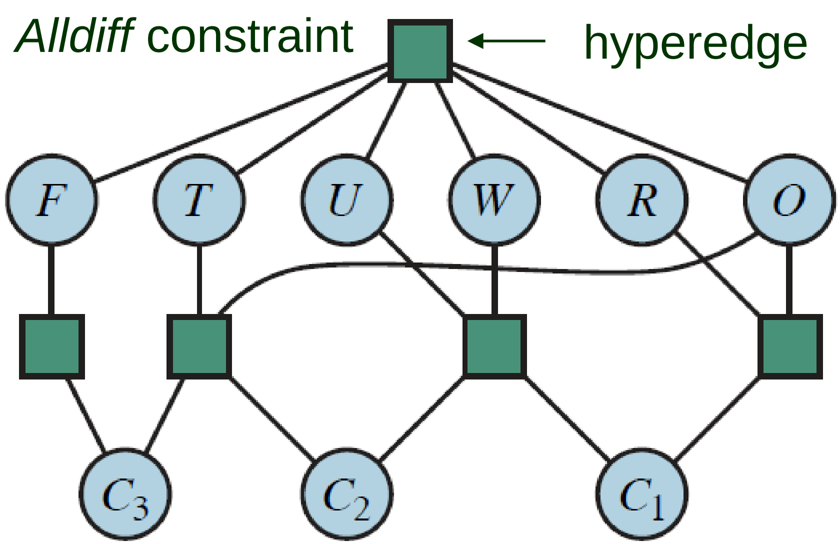
Cryptarithmetic Puzzle

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

Addition constraints on the four columns:

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

C_1, C_2, C_3: 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.

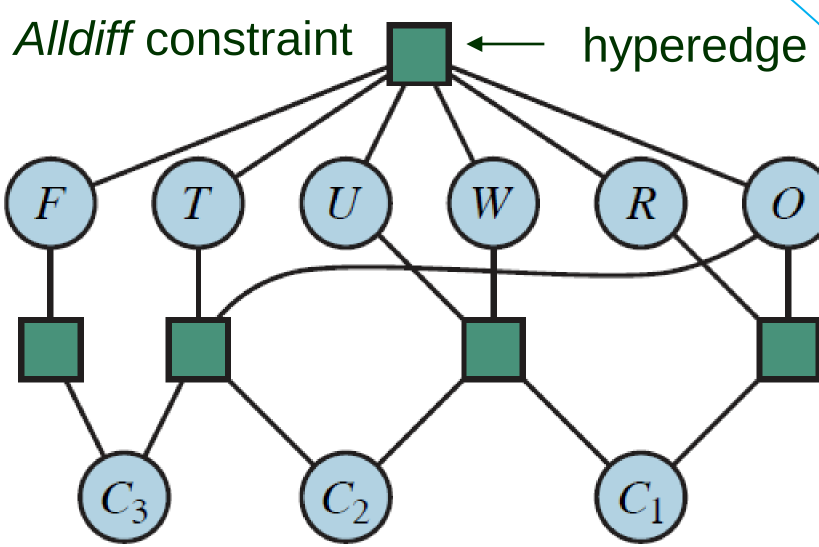
Cryptarithmetic Puzzle

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

Addition constraints on the four columns:

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

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



Constraint hypergraph

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

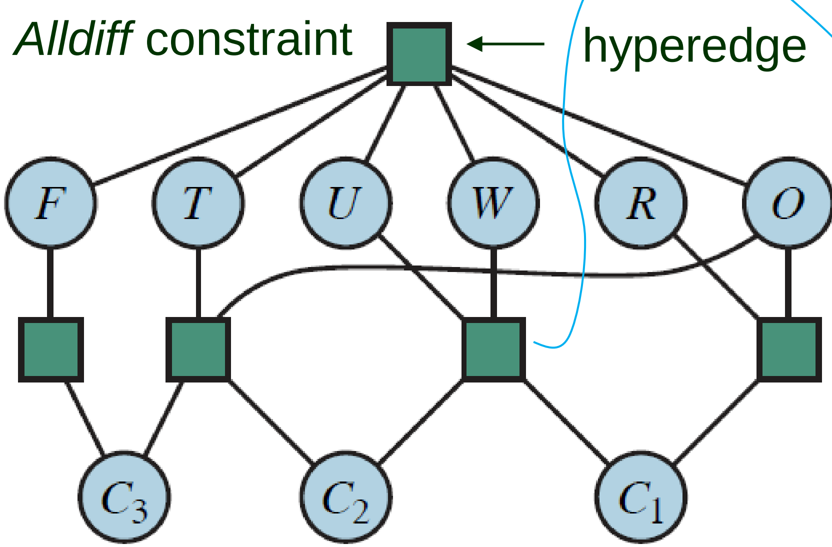
Cryptarithmetic Puzzle

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

Addition constraints on the four columns:

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

C_1, C_2, C_3: 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

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.

⇓

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.

IV. 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.

IV. Constraint Propagation

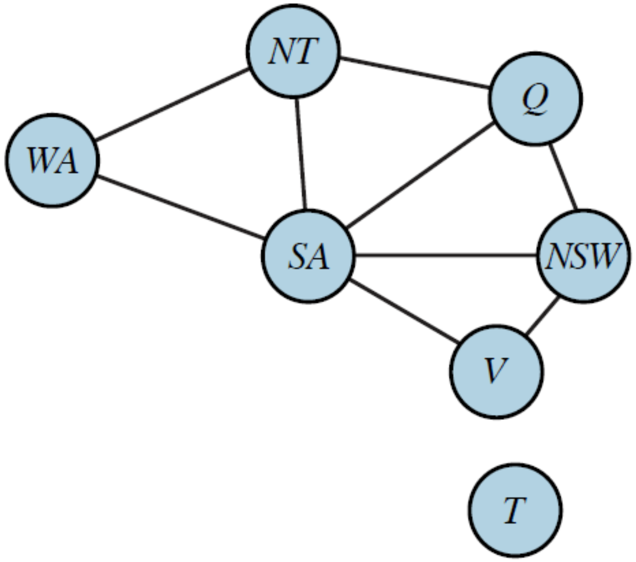
Use constraints to reduce the number of legal values for a variable, which in turn reduce those for another variable, and so on.

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

IV. 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.

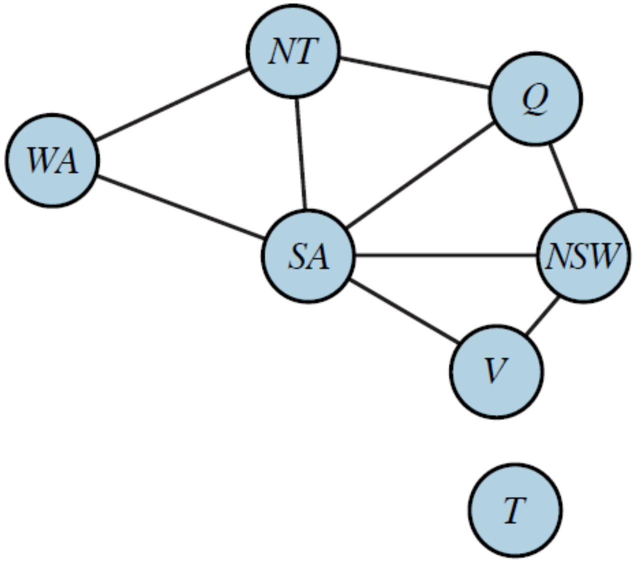


- Node consistency

IV. 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.



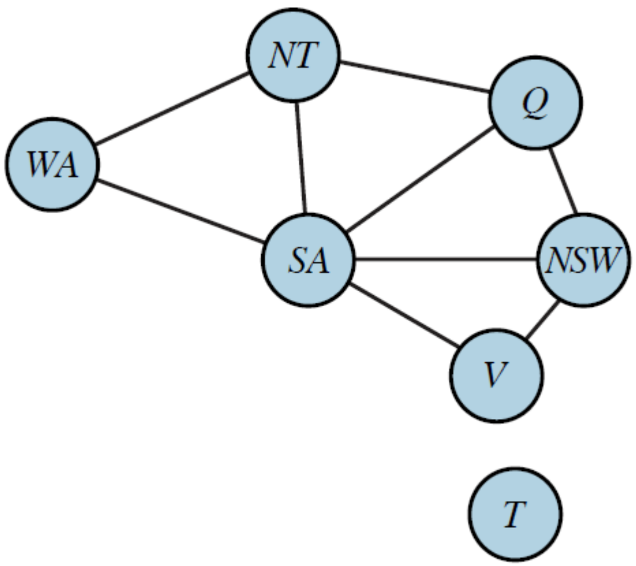
- Node consistency

  Suppose South Australians dislike green.

IV. 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.

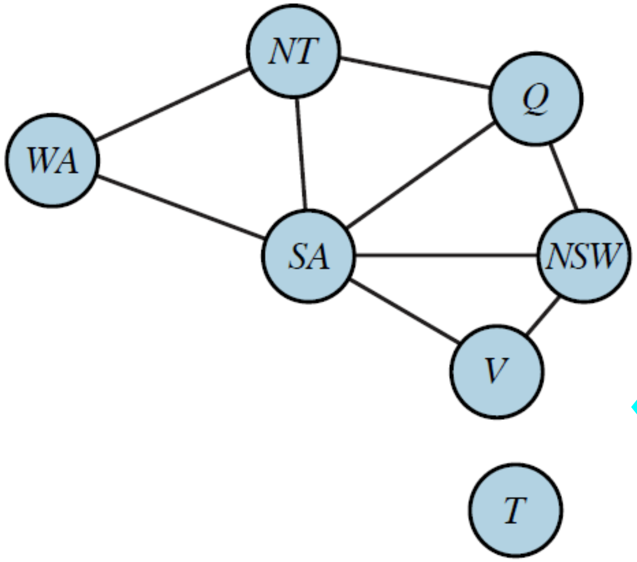
  ⇓

  Domain for *SA*: {red, blue}

IV. 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}

♦ Eliminate all the unary constraints by reducing the domains of the involved variables at the start.

♦ A variable is *node-consistent* if all the values in its domain satisfy its unary constraints.

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

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

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

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

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

Y arc-consistent w.r.t. X

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 ⟹ Reduced domain of X: $\{0, 1, 2, 3\}$

Y arc-consistent w.r.t. X ⟹ 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

$Y = X^2$ 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

$Y = X^2$ 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≠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) ← 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   // propagate to other variables sharing a
        add (X_k, X_i) to queue                  // constraint with X_i since its domain has
  return true                                    // 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 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 $\leq d$ times (X_i has $\leq d$ values to delete).
- 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*

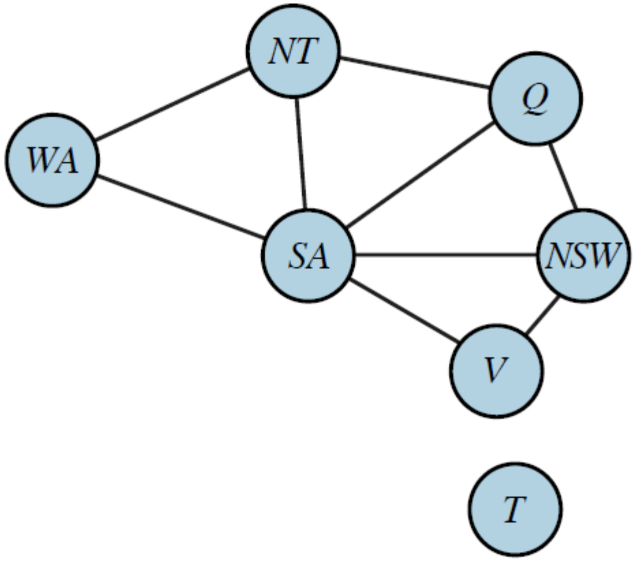
Domain size $\leq d$, c binary constraints.

- Each arc inserted into the queue $\leq d$ times (X_i has $\leq d$ values to delete).
- Each checking takes $O(d^2)$ time.

$$O(cd^3)$$

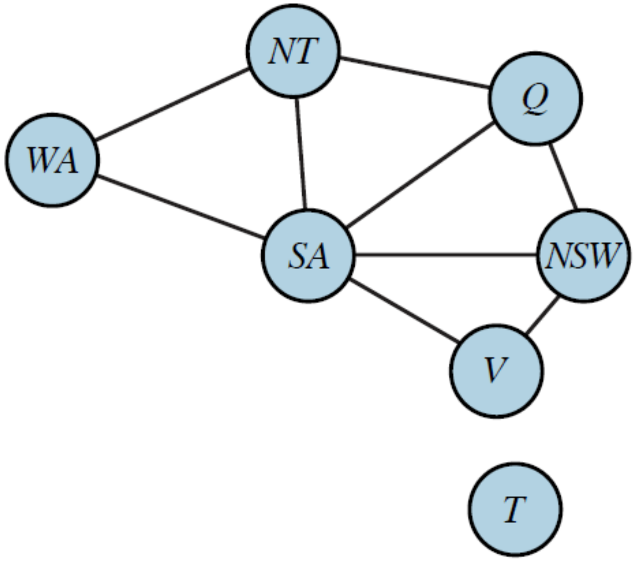
Path Consistency

Arc consistency reduces the domains.



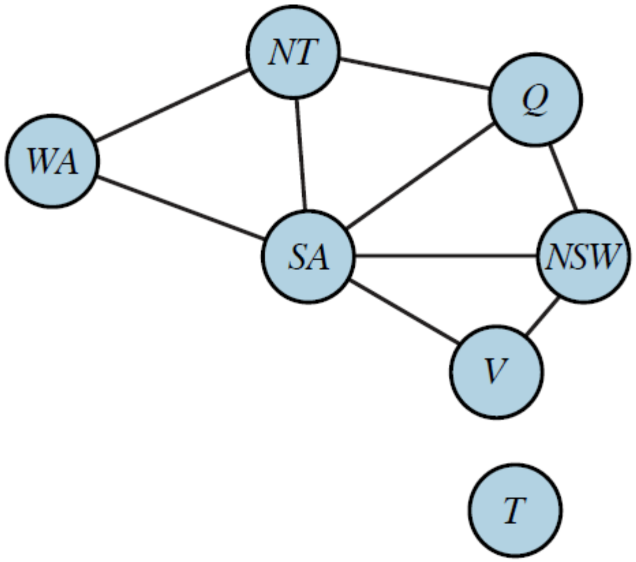
Path Consistency

Arc consistency reduces the domains.



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

Path Consistency

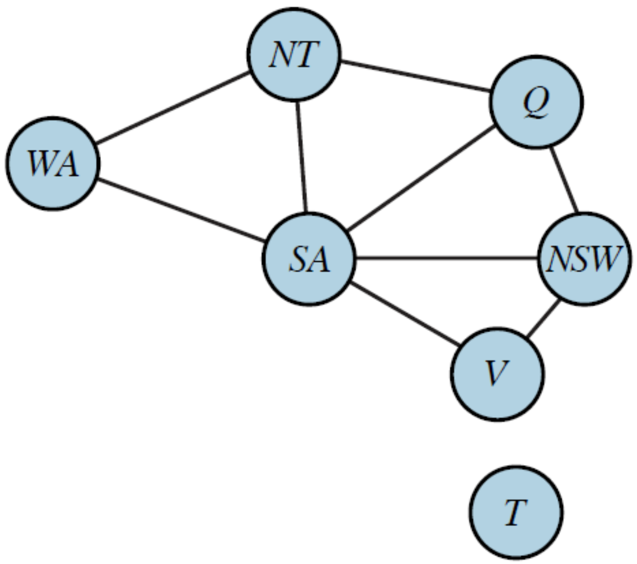


No solution if using two colors even though every constraint can be satisfied individually.

Arc consistency reduces the domains.

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

Path Consistency

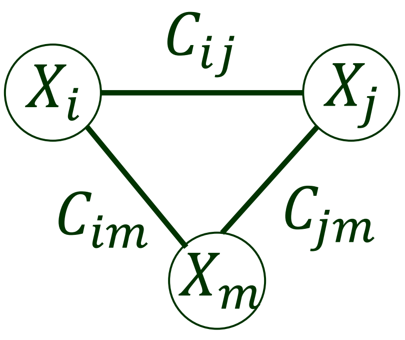


No solution if using two colors even though every constraint can be satisfied individually.

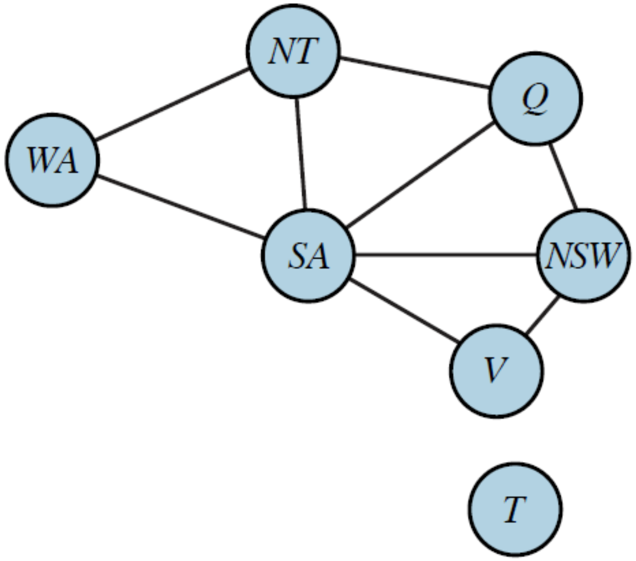
Arc consistency reduces the domains.

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.



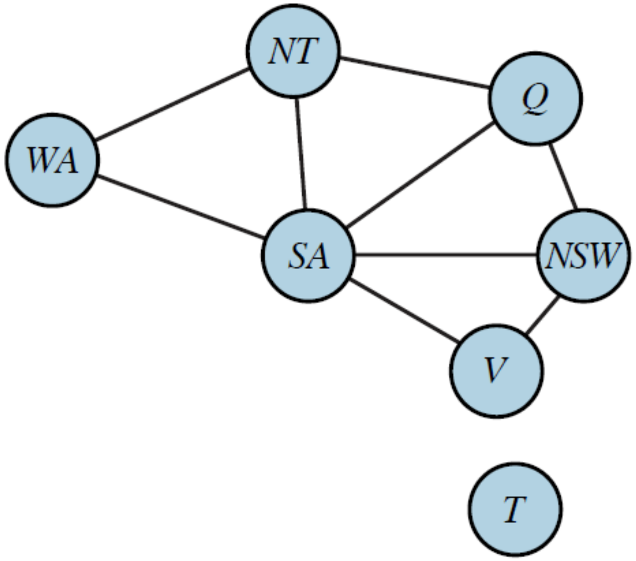
Application of Path Consistency



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*.

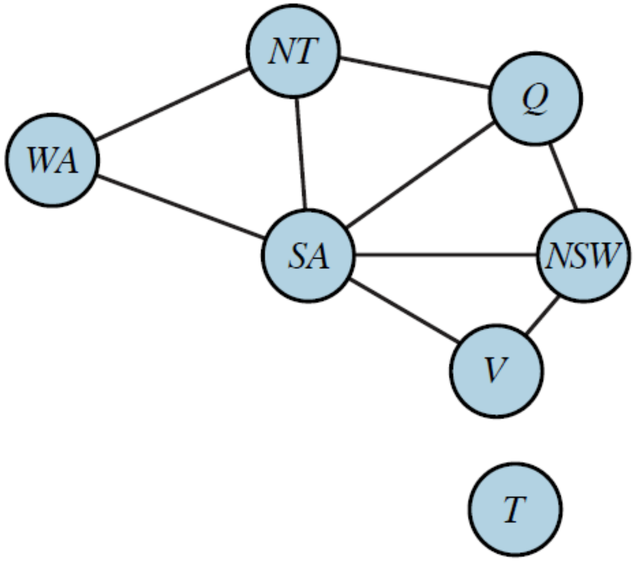
Assume two colors only: *red*, *blue*.

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:

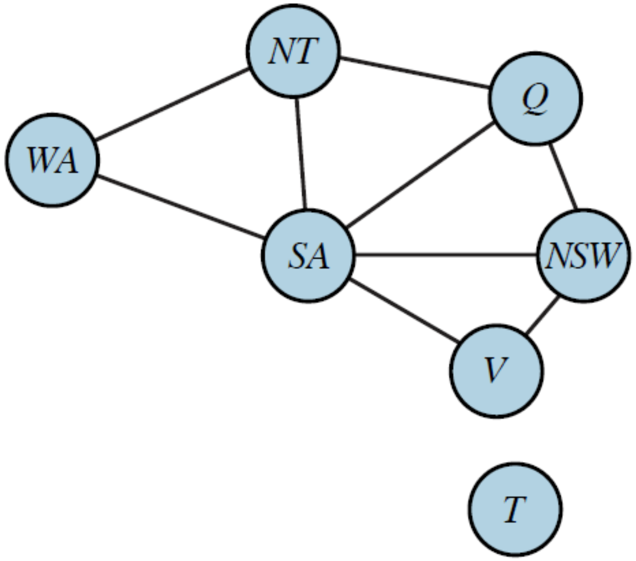
{*WA* = *red*, *SA* = *blue*}

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

⇓

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*}

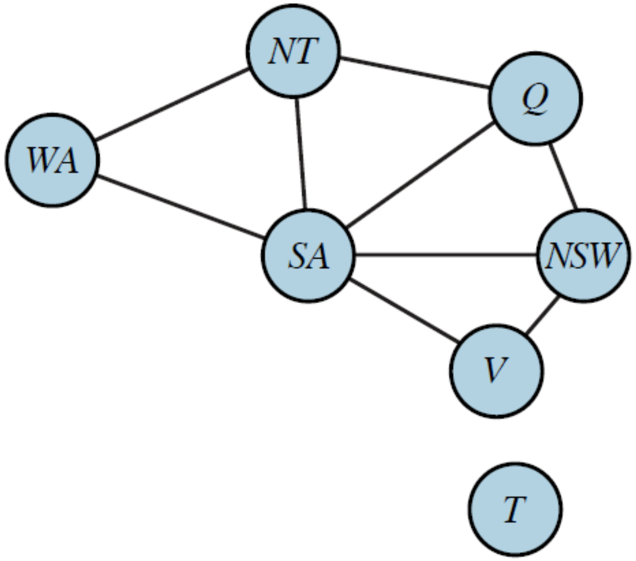
⇩

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.

⇓ 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.

⇩ 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.

⇩ 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: $A1, \ldots, A9, B1, \ldots I9$

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

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

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

27 *Alldiff* constraints:

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

⋮

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

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

27 *Alldiff* constraints:

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

⋮

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

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

⋮

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

⋮

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

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

⋮

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

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

⋮

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

⋮

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

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

⋮

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

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

⋮

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

⋮

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

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

⋮

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

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

⋮

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:

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

⇓ constraints in the box

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

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

⇓ constraints in the column

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

$$= \{4\}$$

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:

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

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

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

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

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:

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

$\Downarrow$ 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:

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

⇓ 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 A6:

Consider I6:

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

⇓ constraints in the column

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

⇓ 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 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

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

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:

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

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 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

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

Consider A6:

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

$\Downarrow$ 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:

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

$\Downarrow$ 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

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

Consider A6:

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

$\Downarrow$ constraints in the column

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