Constraint Satisfaction Problems (CSPs)

Outline

I. Definition of a CSP

II. Map coloring

III. Job shop scheduling

IV. The diet problem and linear programming

* 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. Definition of a CSP

- A set of variables $\mathcal{X} = \{X_1, \dots, X_n\}$.

- A set of domains $\mathcal{D} = \{D_1, \dots, D_n\}$.

  Domain $D_i = \{v_1, \dots, v_{k_i}\}$ is the set of allowable values for the variable X_i.

  e.g., $\{true, false\}$ for a Boolean variable.

- A set of constraints $\mathcal{C} = \{C_1, \dots, C_m\}$ that specifies allowable combination of values.

Relation

- C_j: $\langle (v_i, v_j)$, relation$\rangle$

  - a set of tuple of values for v_i and v_j

    If $D_1 = D_2 = \{1, 2, 3\}$, the relation "X_1 is greater than X_2":

$$\langle (X_1, X_2), \{(3,1), (3,2), (2,1)\} \rangle$$

Relation

- C_j: $\langle (v_i, v_j)$, relation$\rangle$

  - a set of tuple of values for v_i and v_j

    If $D_1 = D_2 = \{1, 2, 3\}$, the relation "X_1 is greater than X_2":

    $$\langle (X_1, X_2), \{(3,1), (3,2), (2,1)\} \rangle$$

  - a function that checks if a tuple satisfies the relation

    $$\langle (X_1, X_2), X_1 > X_2 \rangle$$

Assignments

$\{X_i = v_i, X_j = v_j, \ldots\}$

Assignments

$$\{X_i = v_i, X_j = v_j, \ldots\}$$

An assignment is

- *consistent* if no constraint is violated.

Assignments

$$\{X_i = v_i, X_j = v_j, \ldots\}$$

An assignment is

- *consistent* if no constraint is violated.

Assignments

$$\{X_i = v_i, X_j = v_j, \dots\}$$

An assignment is

- *consistent* if no constraint is violated.
- *complete* if every variable is assigned a value.
- *partial* if some variables are unassigned.

Assignments

$$\{X_i = v_i, X_j = v_j, \dots\}$$

An assignment is

- *consistent* if no constraint is violated.
- *complete* if every variable is assigned a value.
- *partial* if some variables are unassigned.

A *solution* to a CSP is a consistent, complete assignment.

A *partial solution* is a partial assignment that is consistent.

Assignments

$$\{X_i = v_i, X_j = v_j, \dots\}$$

An assignment is

- *consistent* if no constraint is violated.
- *complete* if every variable is assigned a value.
- *partial* if some variables are unassigned.

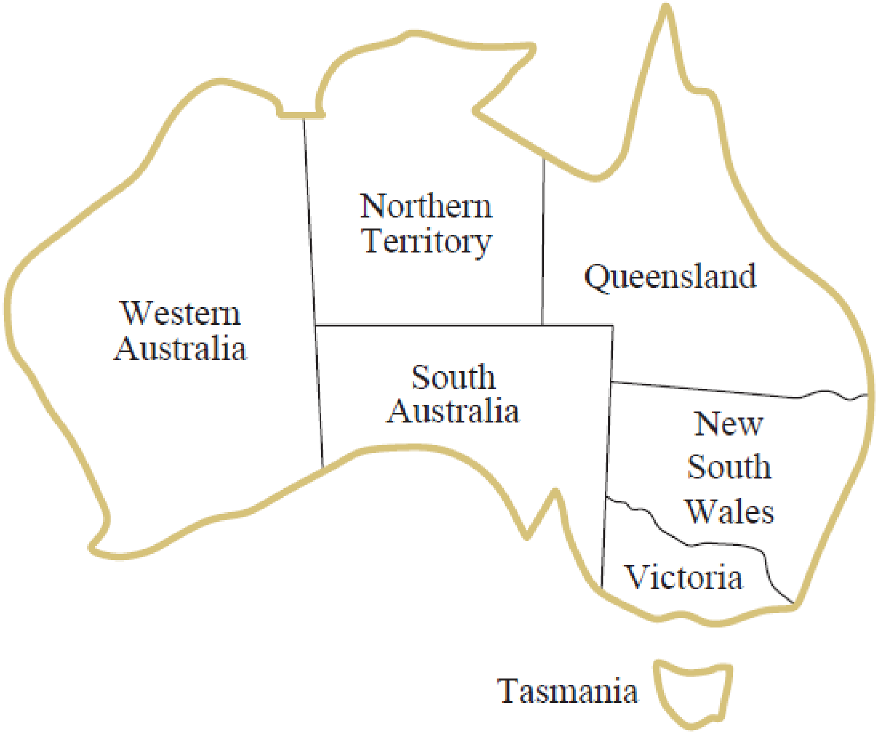
A *solution* to a CSP is a consistent, complete assignment.

A *partial solution* is a partial assignment that is consistent.

Solving a CSP is NP-complete in general!

II. Example 1: Map Coloring

Color the regions of Australia in red, green, or blue such that no two neighboring regions share the same color.

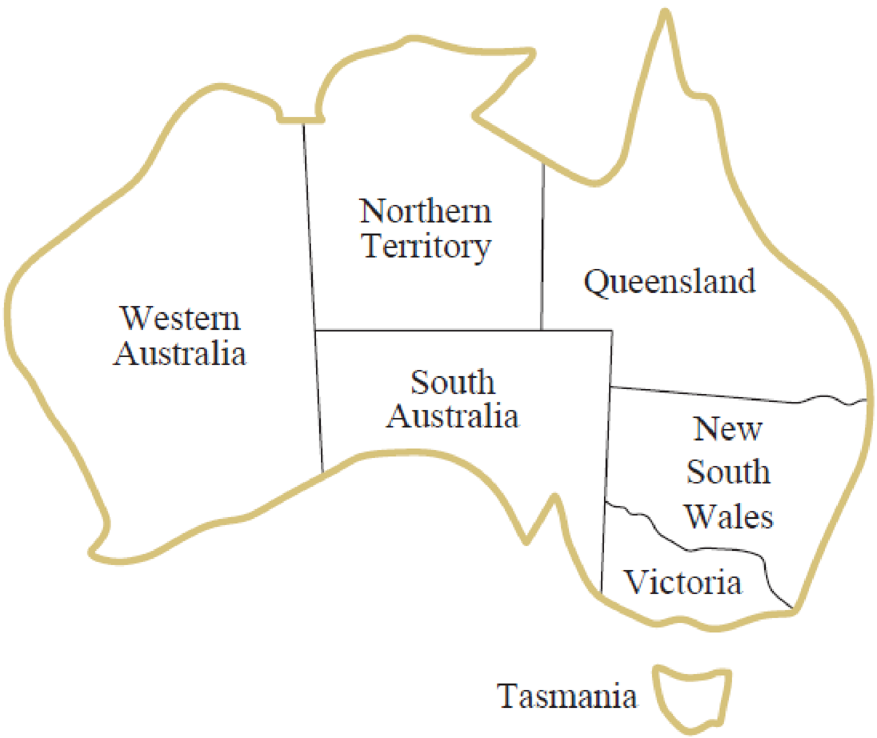


Variables: $\mathcal{X} = \{WA, NT, Q, NSW, V, SA, T\}$

Domains: $D_i = \{red, green, blue\}$

Map Coloring (cont'd)

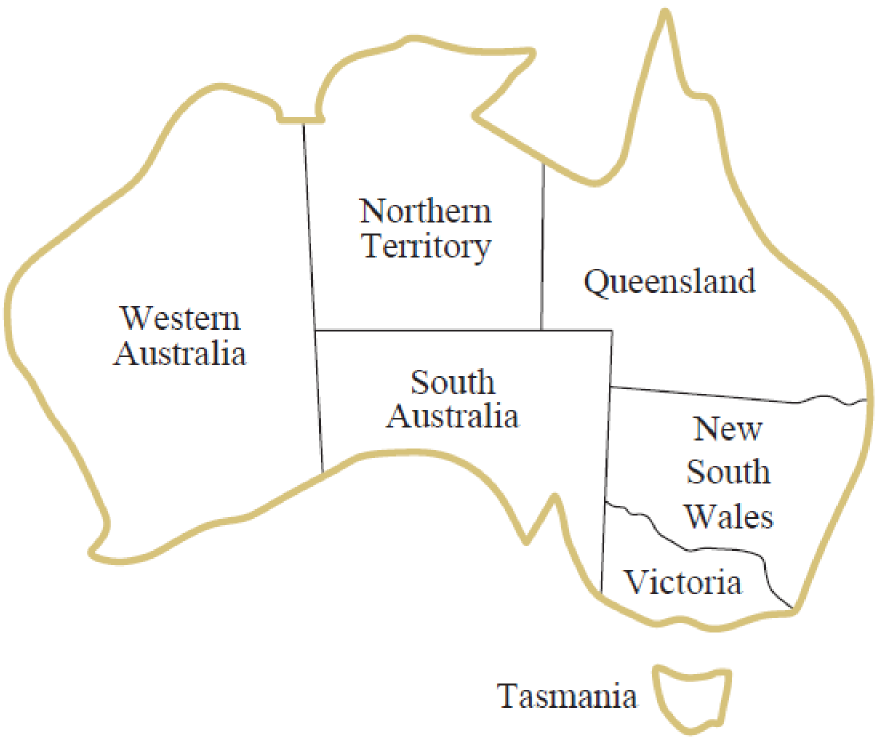
Constraints: $\mathcal{C} = \{SA \neq WA,\ SA \neq NT,\ SA \neq Q,\ SA \neq NSW,\ SA \neq V$
$WA \neq NT,\ NT \neq Q,\ Q \neq NSW,\ NSW \neq V\}$



Map Coloring (cont'd)

Constraints: $\mathcal{C} = \{SA \neq WA,\ SA \neq NT,\ SA \neq Q,\ SA \neq NSW,\ SA \neq V$
$WA \neq NT,\ NT \neq Q,\ Q \neq NSW,\ NSW \neq V\}$

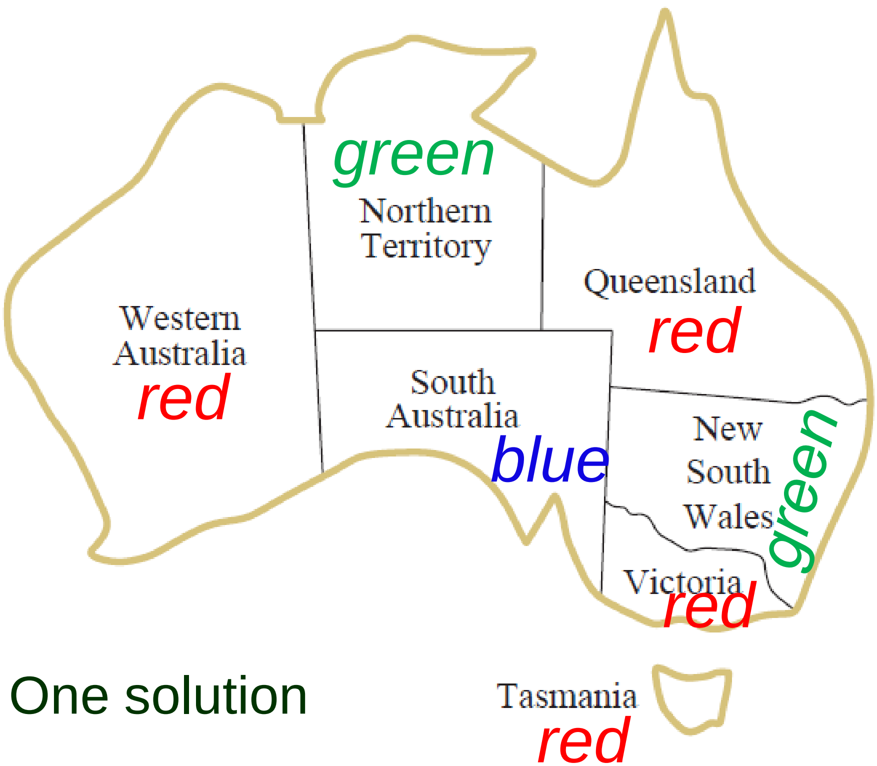
$SA \neq WA$ by enumeration: { (*red*, *green*), (*red*, *blue*), (*green*, *red*),
(*green*, *blue*), (*blue*, *red*), (*blue*, *green*)}



Map Coloring (cont'd)

Constraints: $\mathcal{C} = \{SA \neq WA,\ SA \neq NT,\ SA \neq Q,\ SA \neq NSW,\ SA \neq V$
$WA \neq NT,\ NT \neq Q,\ Q \neq NSW,\ NSW \neq V\}$

$SA \neq WA$ by enumeration: { (*red*, *green*), (*red*, *blue*), (*green*, *red*), (*green*, *blue*), (*blue*, *red*), (*blue*, *green*)}

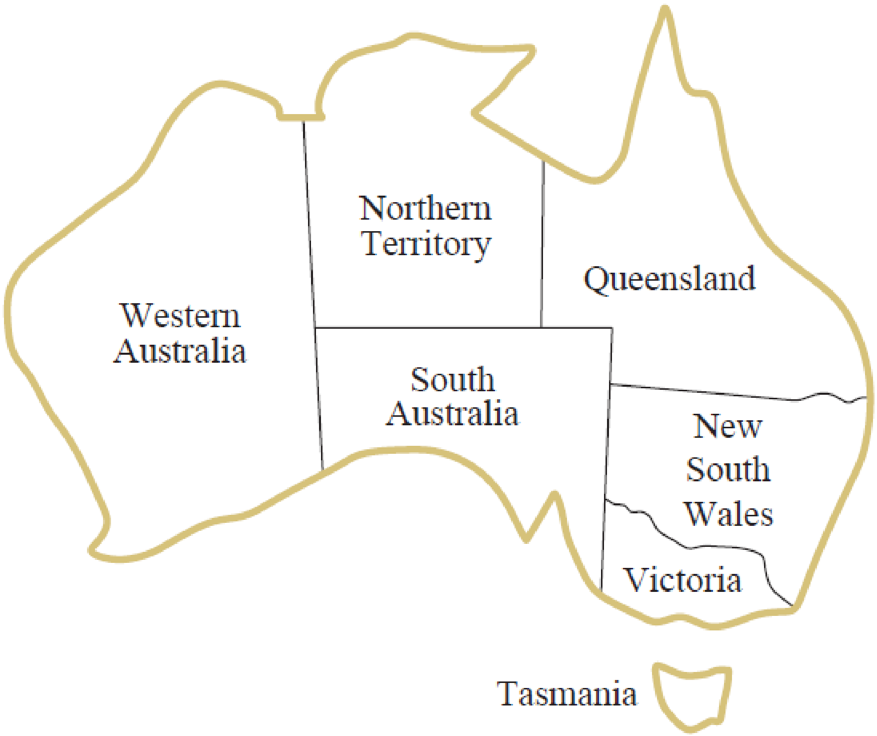


One solution

Constraint Graph

Binary constraints only.

variable ↔ vertex
constraint ↔ edge

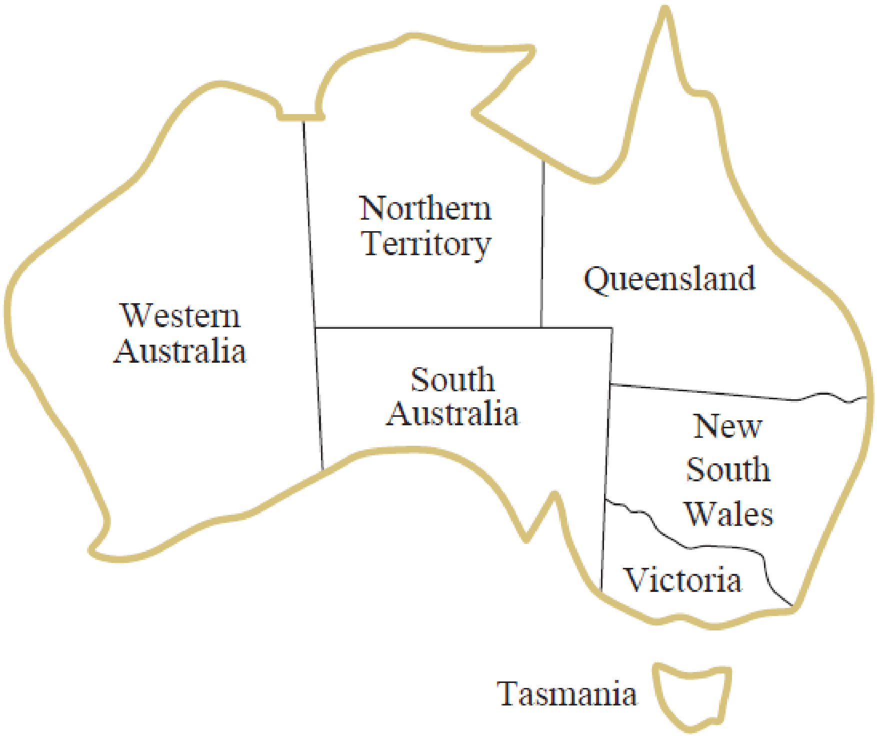


Constraint Graph

Binary constraints only.

variable ⟷ vertex
constraint ⟷ edge

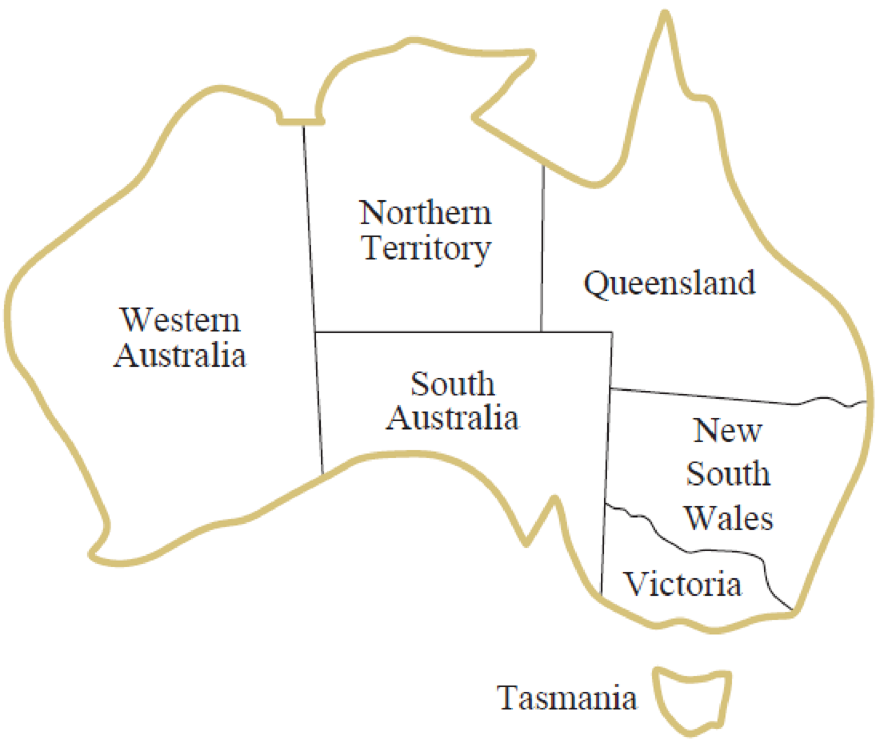
Constraint graph



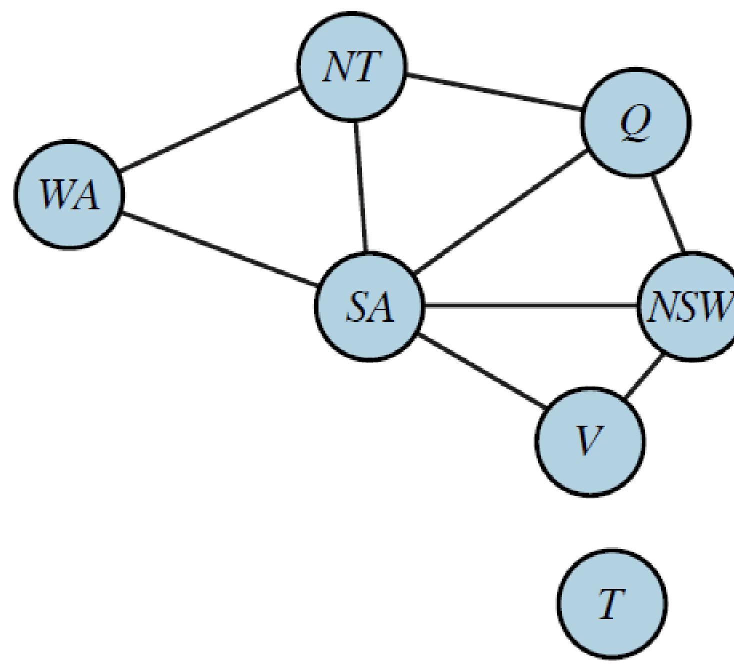
Constraint Graph

Binary constraints only.

variable ⟷ vertex
constraint ⟷ edge

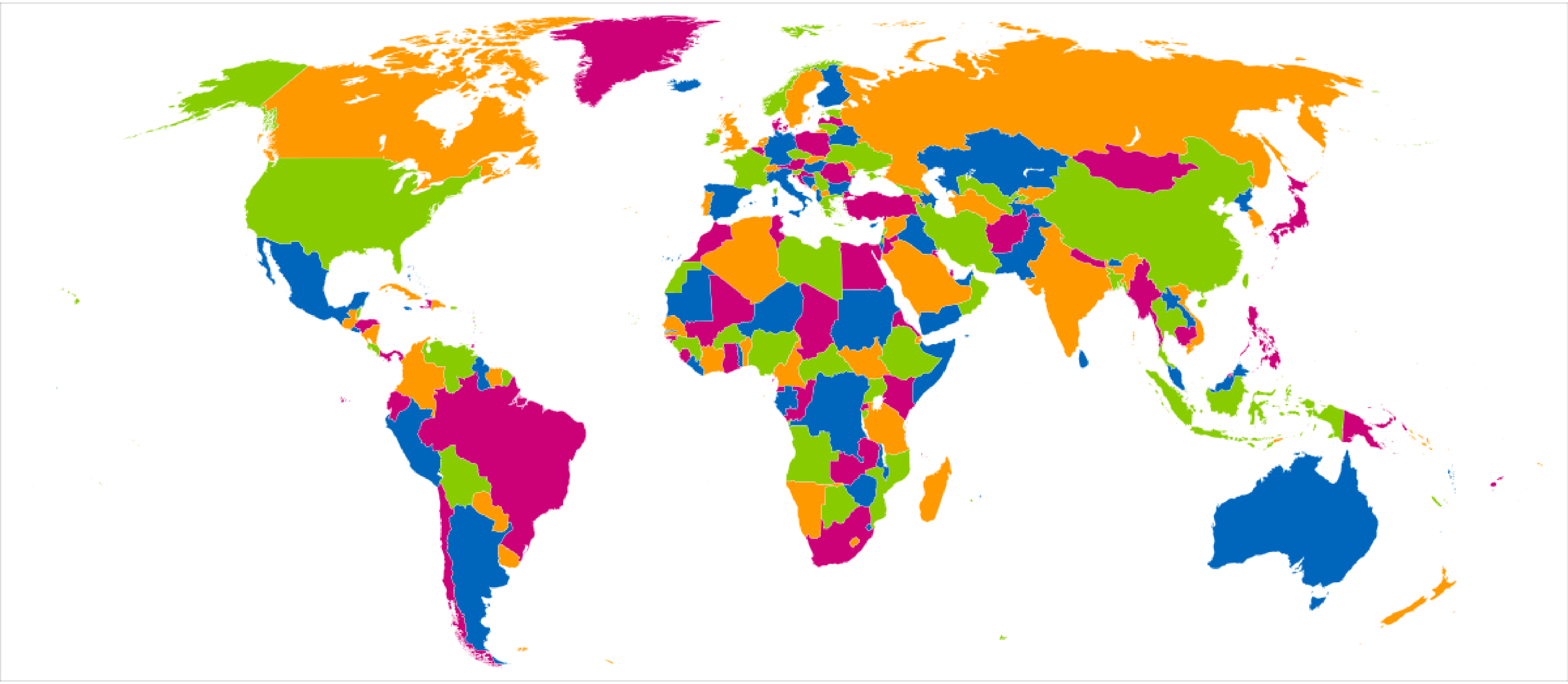


Constraint graph



Four-Color Theorem

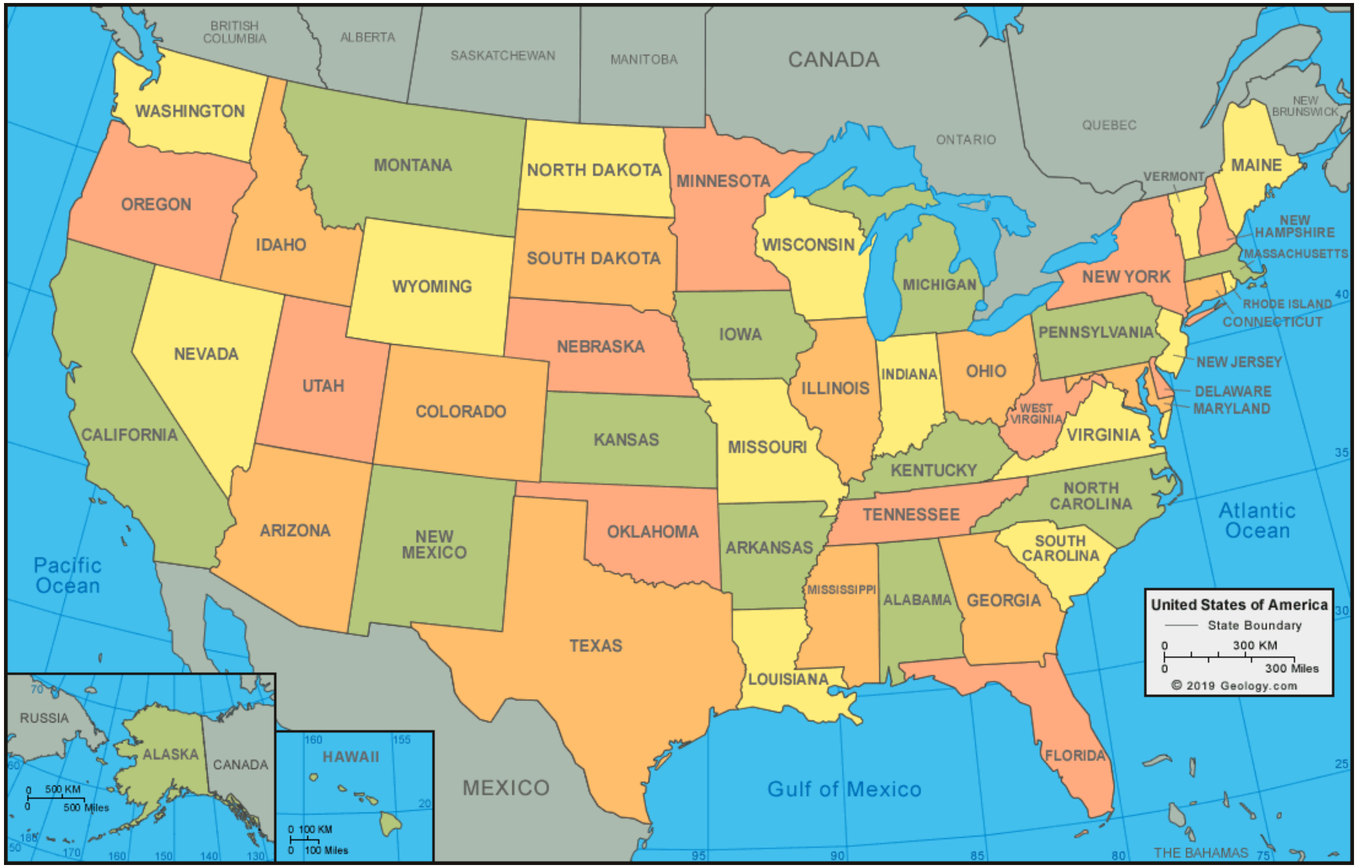
Theorem Any map in a plane can be colored using four colors in such a way that regions sharing a common boundary (other than a single point) do not share the same color.



* Images from https://commons.wikimedia.org/wiki/File:World_map_with_four_colours.svg and https://geology.com/world/the-united-states-of-america-satellite-image.shtml.

Four-Color Theorem

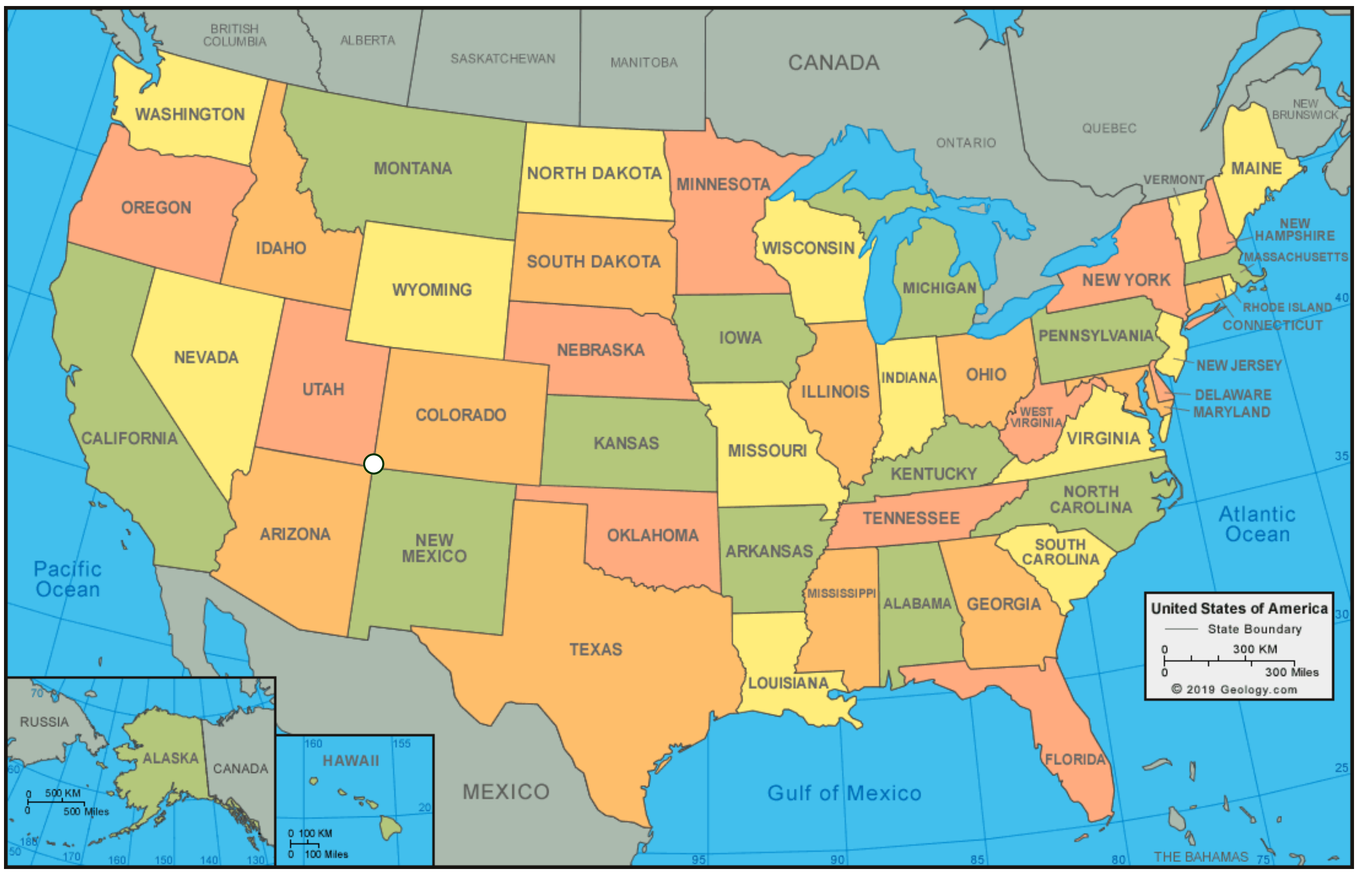
Theorem Any map in a plane can be colored using four colors in such a way that regions sharing a common boundary (other than a single point) do not share the same color.



* Images from https://commons.wikimedia.org/wiki/File:World_map_with_four_colours.svg and https://geology.com/world/the-united-states-of-america-satellite-image.shtml.

Four-Color Theorem

Theorem Any map in a plane can be colored using four colors in such a way that regions sharing a common boundary (other than a single point) do not share the same color.



* Images from https://commons.wikimedia.org/wiki/File:World_map_with_four_colours.svg and https://geology.com/world/the-united-states-of-america-satellite-image.shtml.

Why Formulate a CSP?

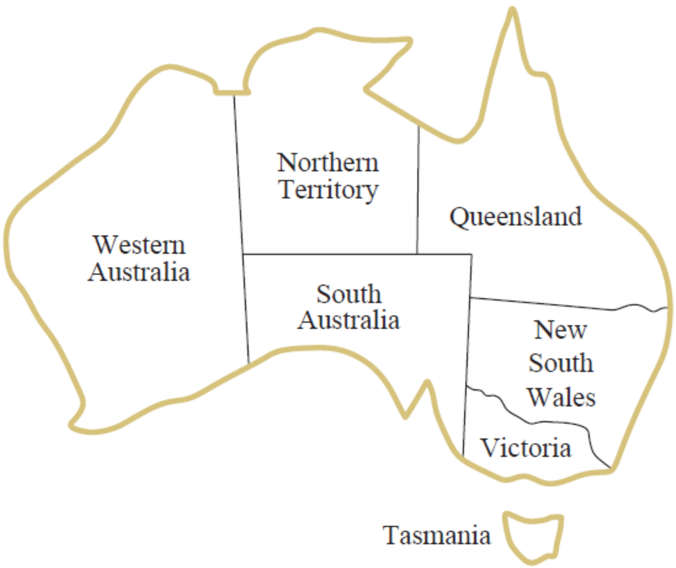
- A natural representation for a wide variety of problem.
- Fast and efficient CSP solvers.

Why Formulate a CSP?

- A natural representation for a wide variety of problem.
- Fast and efficient CSP solvers.
- Ability of a CSP solver to reduce the search space significantly.

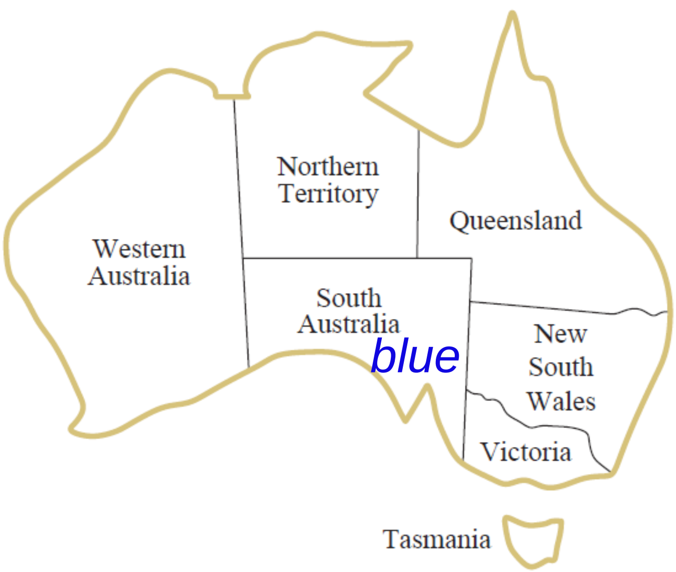
Why Formulate a CSP?

- A natural representation for a wide variety of problem.
- Fast and efficient CSP solvers.
- Ability of a CSP solver to reduce the search space significantly.



Why Formulate a CSP?

- A natural representation for a wide variety of problem.
- Fast and efficient CSP solvers.
- Ability of a CSP solver to reduce the search space significantly.



{*SA* = *blue*}

Why Formulate a CSP?

- A natural representation for a wide variety of problem.
- Fast and efficient CSP solvers.
- Ability of a CSP solver to reduce the search space significantly.

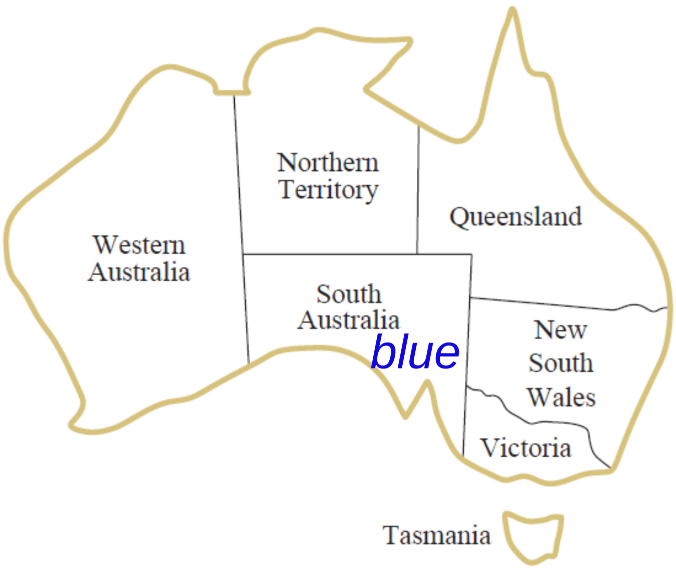
{*SA* = *blue*}

$\Downarrow$

blue cannot be assigned to *WA*, *NT, Q, NSW*, and *V*.

Why Formulate a CSP?

- A natural representation for a wide variety of problem.
- Fast and efficient CSP solvers.
- Ability of a CSP solver to reduce the search space significantly.



{*SA* = *blue*}

$\Downarrow$

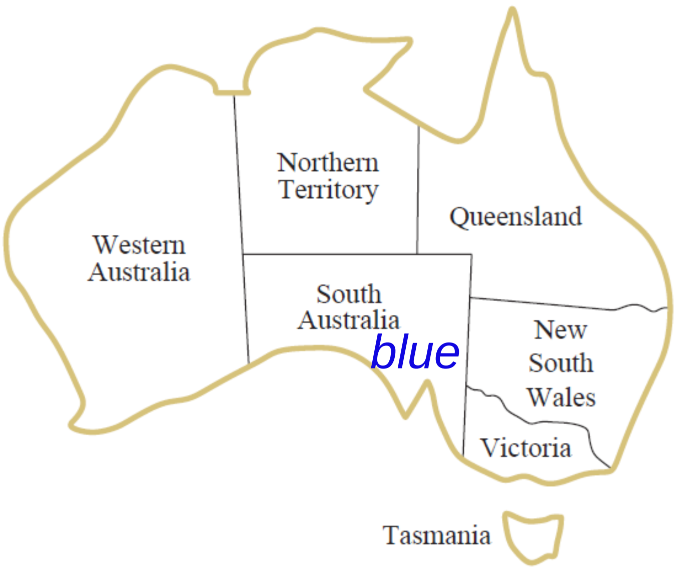
blue cannot be assigned to *WA*, *NT*, *Q*, *NSW*, and *V*.

$\Downarrow$

$2^5 = 32$ assignments to the five regions.

Why Formulate a CSP?

- A natural representation for a wide variety of problem.
- Fast and efficient CSP solvers.
- Ability of a CSP solver to reduce the search space significantly.



{*SA* = *blue*}

$\Downarrow$

blue cannot be assigned to *WA*, *NT*, *Q*, *NSW*, and *V*.

$\Downarrow$

$2^5 = 32$ assignments to the five regions.

A reduction from $3^5 = 243$ assignments by a search procedure not using the constraint.

Why the Search Space is Reduced?

- Atomic state-space search

Is this specific state a goal?

If not, what about this one?

Why the Search Space is Reduced?

- Atomic state-space search

  Is this specific state a goal?

  If not, what about this one?

- CSP

  Upon violation by a partial assignment

  - discard its further refinements,
  - see which variables cause the violation,
  - focus on those variables that matter.

Why the Search Space is Reduced?

- Atomic state-space search

Is this specific state a goal?
If not, what about this one?

- CSP

Upon violation by a partial assignment

- ♦ discard its further refinements,
- ♦ see which variables cause the violation,
- ♦ focus on those variables that matter.

III. 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

III. 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

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

III. 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.

III. 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, 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

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

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

$$X + d_X \leq Inspect$$ 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.

$\Downarrow$

$Axle_F$ 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.

IV. 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.

IV. 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

$$\min \quad \overbrace{3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6}^{\text{Objective function (linear)}}$$

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

$$
\begin{aligned}
\text{min} \quad & \overbrace{3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6}^{\text{Objective function (linear)}} \\
\text{subject to} \quad & 0 \le x_1 \le 4 \\
& 0 \le x_2 \le 3 \\
& 0 \le x_3 \le 2 \\
& 0 \le x_4 \le 8 \\
& 0 \le x_5 \le 2 \\
& 0 \le x_6 \le 2 \\
\text{and} \quad &
\end{aligned}
$$

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

$$\text{min} \quad \overbrace{3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6}^{\text{Objective function (linear)}}$$

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

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

$$\text{min} \quad \overbrace{3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6}^{\text{Objective function (linear)}}$$

$$\text{subject to} \quad \left.\begin{aligned} 0 &\le x_1 \le 4 \\ 0 &\le x_2 \le 3 \\ 0 &\le x_3 \le 2 \\ 0 &\le x_4 \le 8 \\ 0 &\le x_5 \le 2 \\ 0 &\le x_6 \le 2 \end{aligned}\right\} \text{Servings-per-day limits}$$

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

$$\min \quad \overbrace{3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6}^{\text{Objective function (linear)}}$$

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

and

$$\left.\begin{array}{ll} \text{energy} & 110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 \ge 2000 \\ \text{protein} & 4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 \ge 55 \\ \text{calcium} & 2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 \ge 800 \end{array}\right\} \text{Constraints (linear)}$$

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)

$$\min \quad \overbrace{3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6}^{\text{Objective function (linear)}}$$

Linear Program!

$$\text{subject to} \quad \left.\begin{aligned} &0 \le x_1 \le 4 \\ &0 \le x_2 \le 3 \\ &0 \le x_3 \le 2 \\ &0 \le x_4 \le 8 \\ &0 \le x_5 \le 2 \\ &0 \le x_6 \le 2 \end{aligned}\right\} \text{Servings-per-day limits}$$

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

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

Linear Programming (LP)

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

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

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

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

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

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

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

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

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

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

Linear Programming (LP)

Max $c_1x_1 + c_2x_2 + \cdots + c_d\, x_d \qquad = \boldsymbol{cx}^{\boldsymbol{T}}$

subject to

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

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

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

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

Linear Programming (LP)

Max $c_1x_1 + c_2x_2 + \cdots + c_d\, x_d \qquad = \boldsymbol{cx^T}$

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

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

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

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

Solvable in time polynomial in d.

Linear Programming (LP)

Max $c_1x_1 + c_2x_2 + \cdots + c_d\,x_d$ $\qquad = \boldsymbol{c}\boldsymbol{x}^T$

subject to

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

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

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

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

Solvable in time polynomial in d.

Linear Programming (LP)

Max $c_1x_1 + c_2x_2 + \cdots + c_d\, x_d \qquad = \boldsymbol{c}\boldsymbol{x}^{\boldsymbol{T}}$

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

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

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

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

Solvable in time polynomial in d.

Simplex method $O(2^d)$
(best performance in practice)

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

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

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

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

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

Solvable in time polynomial in d.

Simplex method $O(2^d)$

(best performance in practice)