Lecture 9

Constraint Satisfaction Problem

Lusi Li
Department of Computer Science
ODU

Reading for This Class:
Chapter 6, Russell and Norvig

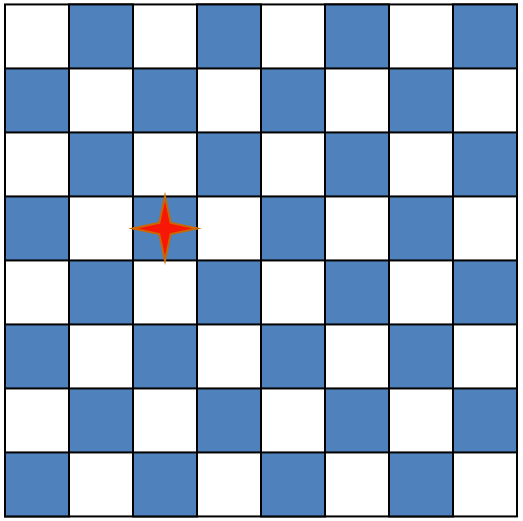
Review

- Last Class
  - Genetic Algorithms
    - Representation
    - Selection
    - Crossover
    - Mutation
- This Class
  - Constraint Satisfaction Problems
- Next Class
  - Backtracking Algorithm

Intro Example: 8-Queens

If we use the incremental approach:

- **Initial state: no queens on board**
- **Each action: add a queen to any empty position on the board**
- **Goal test: 8 queens on board and none attacked**

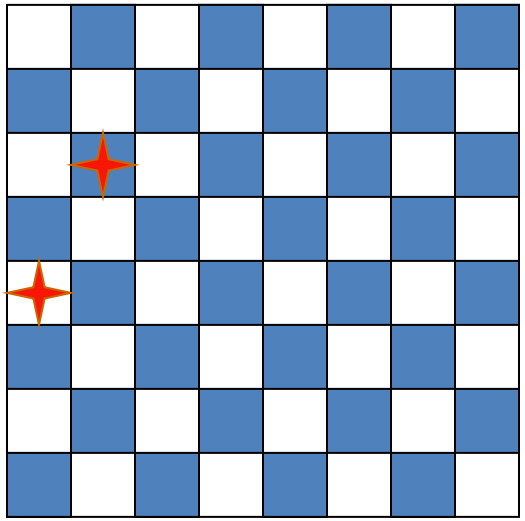


- **Purely generate-and-test**
- **The "search" tree is only used to enumerate all possible 64 x 63 x…x 57 ~ 3 x 10^{14} combinations**

Intro Example: 8-Queens

If we use the incremental approach:

- **Initial state: no queens on board**
- **Each action: add a queen to any empty position in leftmost empty column on board**
- **Goal test: 8 queens on board and none attacked**



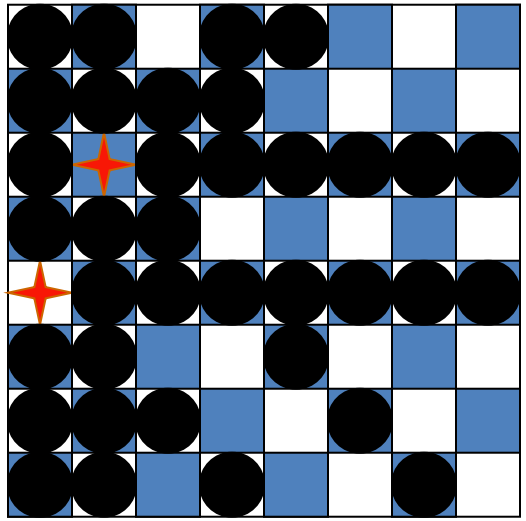
Another form of generate-and-test, with no redundancies

→ "only" $8^8 \sim 1.7 \times 10^7$ combinations

Intro Example: 8-Queens

If we use the incremental approach:

- **Initial state: no queens on board**
- **Each action: add a queen to any allowed position in leftmost empty column on board that will not attack the existing queens**
- **Goal test: 8 queens on board and none attacked**



2057 combinations!!!

What is Needed?

- Not just a successor function and goal test
- But also a means to propagate the constraints imposed by one queen on the others and an early failure test
- → Explicit representation of constraints and constraint manipulation algorithms

Constraint Satisfaction Problems (CSPs)

- A CSP is defined as a triple $<V, D, C>$
  - finite set of variables $V = \{V_1, V_2, \ldots, V_n\}$
  - non-empty domains of possible values for each variable

    $D = \{D_{V1}, D_{V2}, \ldots, D_{Vn}\}$
  - finite set of constraints $C = \{C_1, C_2, \ldots, C_m\}$ that specify allowable combinations of values

- Each constraint consists of a pair <scope, relation>
  - scope is a variable tuple
  - relation is a list of possible values for tuple

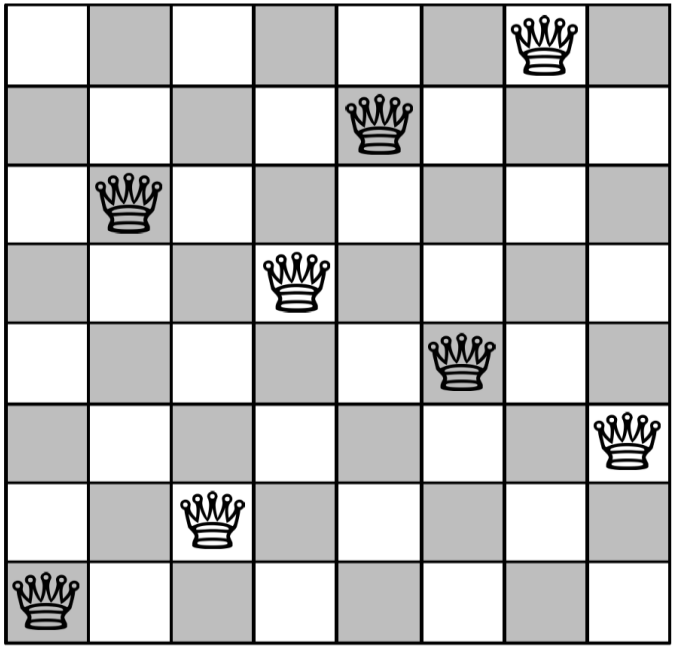
    (e.g., $<(V_2,V_3),\{(R,B),(B,R)\}>$)

    or functional relation (e.g., $<(V_2,V_3),V_2 \neq V_3>$, $<(V_1, V_4), V_1 > V_4 + 5>$)

CSPs (continued)

- A state is an assignment of values to some or all variables.
  - An assignment is *complete* when every variable has a value.
  - An assignment is *partial* when some variables have no values.
- Consistent assignment
  - assignment does not violate the constraints
- Goal test is a set of constraints specifying allowable combinations of values for subsets of variables
- A solution to a CSP is a complete and consistent assignment.
- Some CSPs require a solution that maximizes an *objective function*.

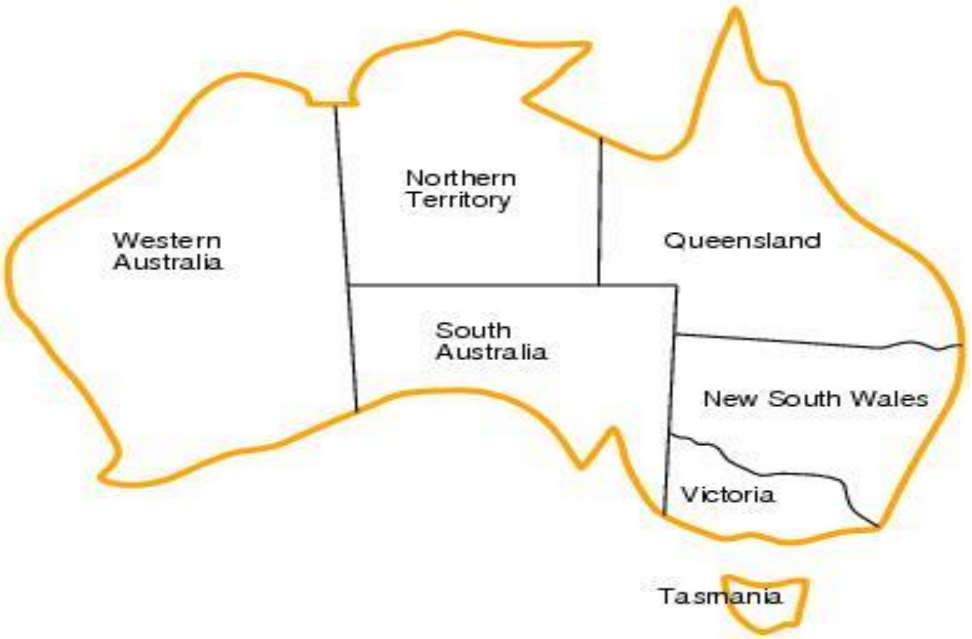
CSP Example: 8-Queens



- Variables: $Q = \{Q_1, Q_2, \ldots, Q_8\}$, each Q_i means a queen in column i
- Domains: $D_i = \{1,2,\ldots,8\}$ *for* Q_i, the row position
- Constraints: no queens attack each other
  - no two queens can be in the same row: $Q_i \neq Q_j$ for $i \neq j$
  - no two queens can be in the same diagonal: $|Q_i - Q_j| \neq |i - j|$

CSP Example: Map Coloring

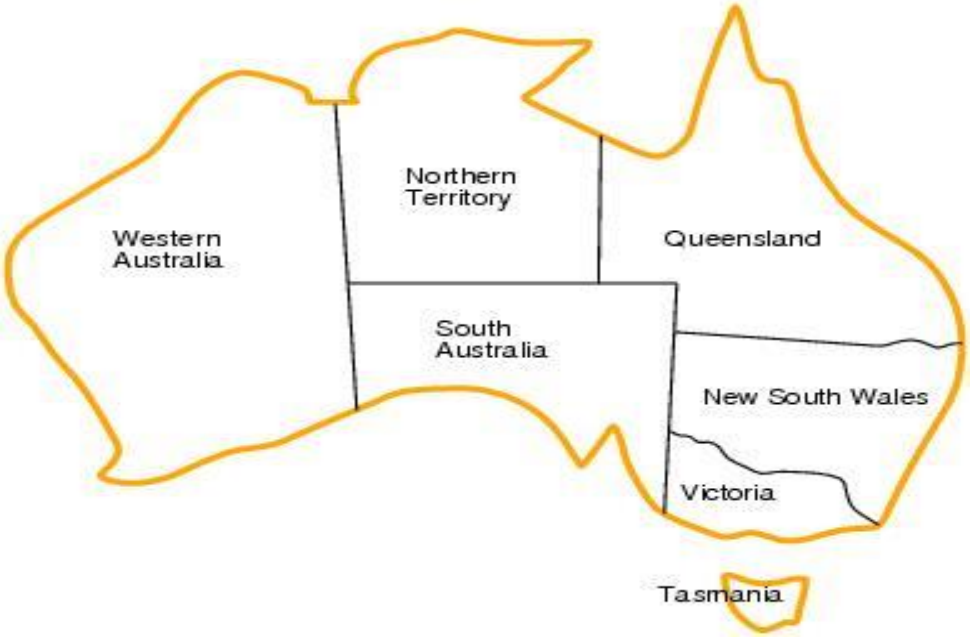
Given 3 different colors {red, green, blue}, color a map so that adjacent regions are different colors



- **Variables?**
- **Domains?**
- **Constraints?**

CSP Example: Map Coloring

Given 3 different colors {red, green, blue}, color a map so that adjacent regions are different colors

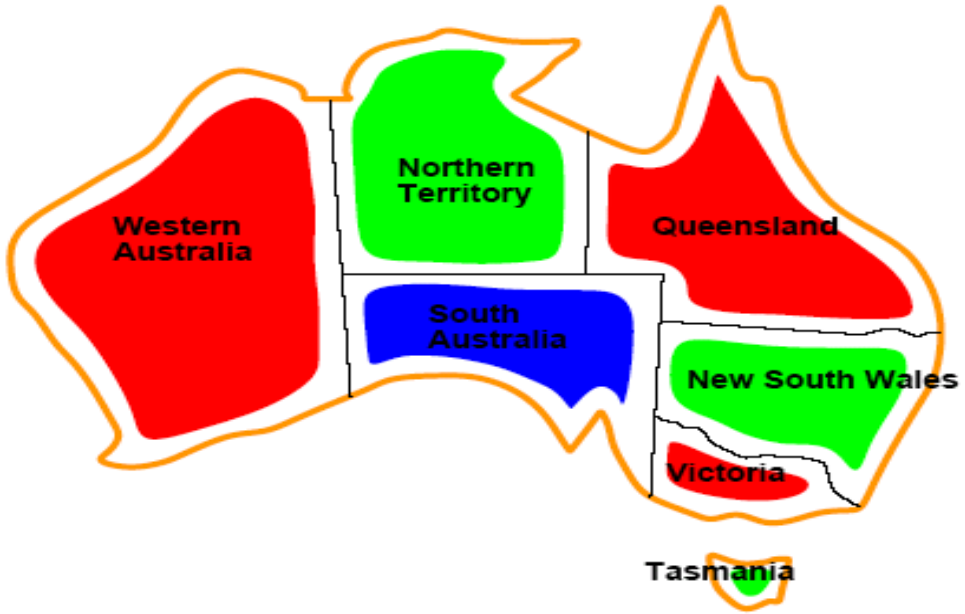


- **Variables: V = {*WA, NT, Q, NSW, V, SA, T*}**
- **Domains: D_i = *{red, green, blue}***
- **Constraints: adjacent regions must have different colors.**
  - C = {WA≠NT, WA≠SA, NT≠SA, NT≠Q, SA≠Q, SA≠NSW, SA≠V, Q≠NSW, NSW≠V}

CSP Example: Map Coloring

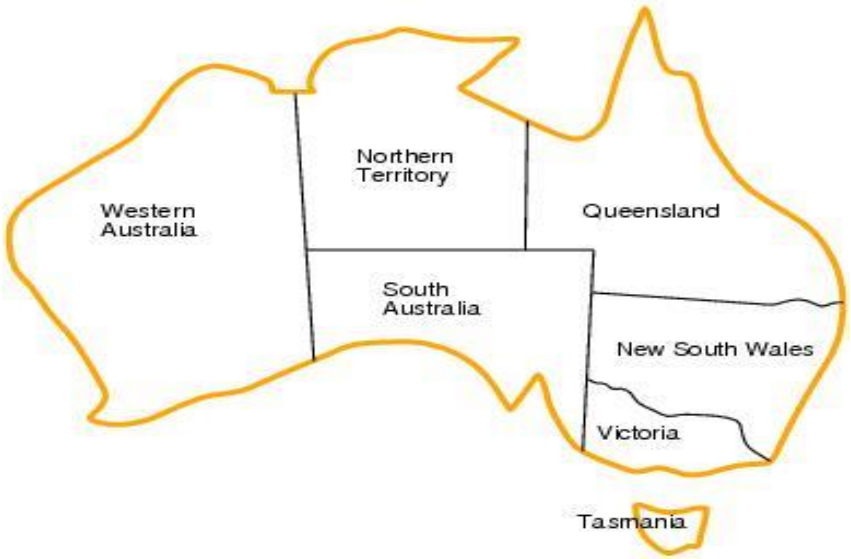
- **Solution**
  - **Color assignments satisfying all constraints, e.g.**

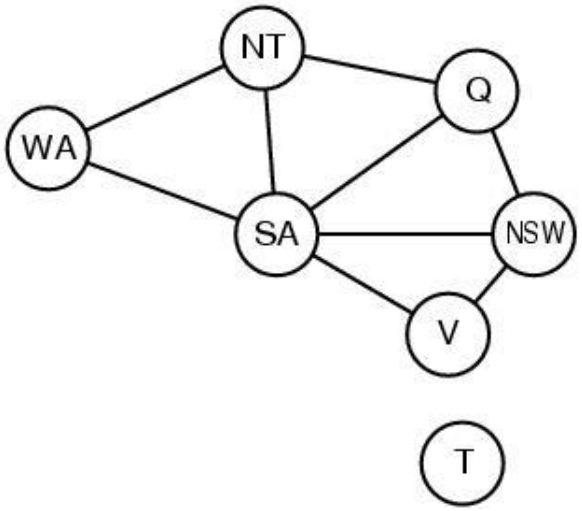
{WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green}



Constraint Graph

- **Constraint graph for binary CSP**
  - **Nodes are variables**
  - **Arcs are binary constraints**
- **Graph can be used to speed up search**

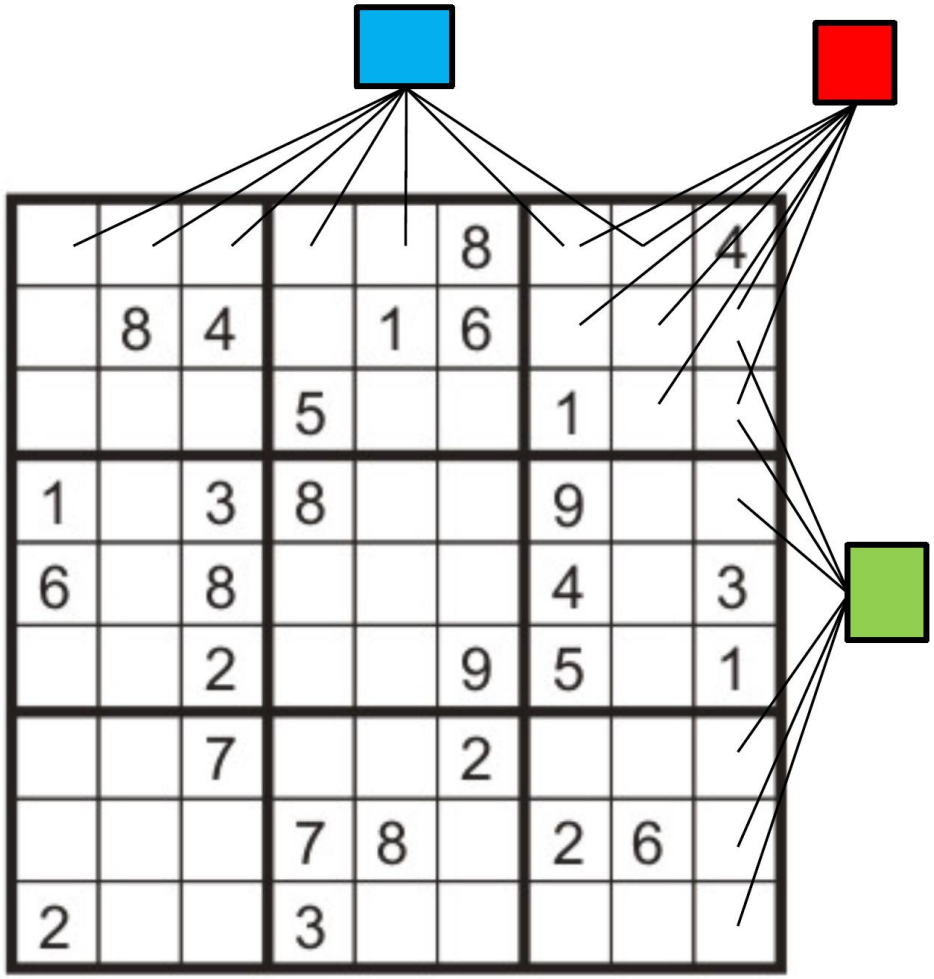




WA≠NT, WA≠SA, NT≠SA, NT≠Q, SA≠Q, SA≠NSW, SA≠V,Q≠NSW, NSW≠V

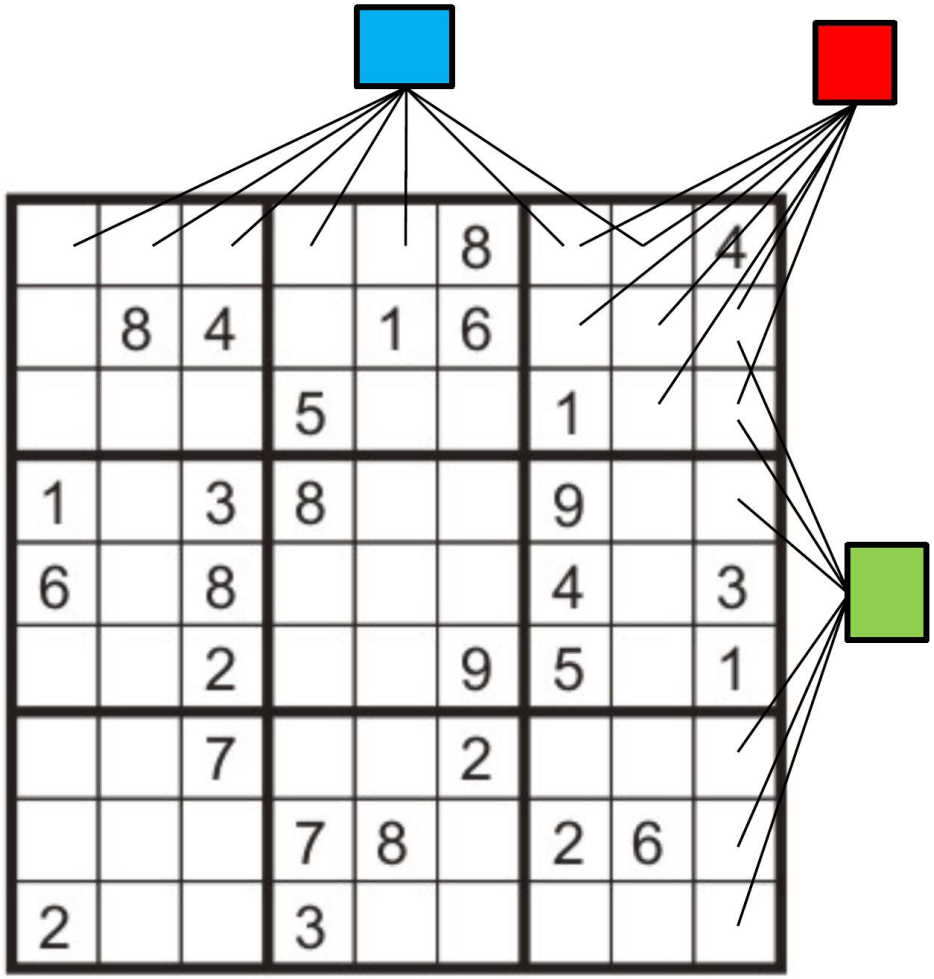
CSP Example: Sudoku

- **Variables?**
- **Domains?**
- **Constraints?**



CSP Example: Sudoku

- **Variables: X_{ij}, each open square**
- **Domains: $D_i = \{1,2,\ldots,9\}$**
- **Constraints:**
  - **9-way all diff col**
  - **9-way all diff row**
  - **9-way all diff box**



CSP Example: Class Scheduling

- 4 required classes to graduate
  - A: Algorithms
  - B: Bayesian Learning
  - C: Computer Programming
  - D: Distributed Computing
- A few restrictions
  - Algorithms must be taken same semester as Distributed computing
  - Computer programming is a prereq for Distributed computing and Bayesian learning, so it must be taken in an earlier semester
  - Algorithms and Bayesian Learning are always offered at the same time, so they cannot be taken the same semester
- 3 semesters (semester 1,2,3) when can take classes

CSP Example: Class Scheduling

- 4 required classes to graduate: A, B, C, D
- A must be taken same semester as D
- C is a prereq for D and B so must take C earlier than D & B
- A & B are always offered at the same time, so they cannot be taken the same semester
- 3 semesters (semester 1,2,3) when can take classes

- **Define CSP**
  - **Variables: V = {A, B, C, D}**
  - **Domains: V_i = {1, 2, 3}**
  - **Constraints: A = D, C < B, C < D, A ≠ B**

CSP Example: Task Scheduling

We have four tasks T1, T2, T3, T4 and takes duration 10, 20, 30, 15 min to complete. Suppose all tasks must be done in 45 min.

The time constraints are as follows:

- **T1 must be done during T3**
- **T2 must be achieved before T1 starts**
- **T2 must overlap with T3**
- **T4 must start after T1 is complete**

- **Variables?**
- **Domains?**
- **Constraints?**

CSP Example: Task Scheduling

We have four tasks T1, T2, T3, T4 and takes duration 10, 20, 30, 15 min to complete. Suppose all tasks must be done in 45 min.

The time constraints are as follows:

- **T1 must be done during T3**
- **T2 must be achieved before T1 starts**
- **T2 must overlap with T3**
- **T4 must start after T1 is complete**

- **Variables: T1, T2, T3, T4**
- **Domains: D_i = {0:45} (start time)**
- **Constraints:** $T1 > T3, T1 + 10 < T3 + 30$

  $T2 + 20 \leq T1$

  $T2 + 20 \geq T3$

  $T1 + 10 \leq T4$

Varieties of CSPs

- **Discrete variables**
  - **Finite domains: size $d \Rightarrow O(d^n)$ complete assignments.**
    - **E.g. Map coloring, Sudoku, N-queens**
  - **Infinite domains (integers, strings, etc.)**
    - **E.g. job scheduling, variables are start/end days for each job**
    - **Need a constraint language e.g., $StartJob_1 + 5 \leq StartJob_3$.**
    - **Linear constraints: solvable**
    - **Nonlinear: no general algorithm**
- **Continuous variables**
  - **e.g. building an airline schedule**
  - **Linear constraints solvable in polynomial time by LP methods.**

Varieties of constraints

- **Unary constraints involve a single variable.**
  - **e.g. *SA* ≠ *green***
- **Binary constraints involve pairs of variables.**
  - **e.g. *SA* ≠ *WA***
- **Higher-order constraints involve 3 or more variables.**
  - **Can always be represented by multiple binary constraints**
- **Global constraints involve an arbitrary number of variables**
  - **E.g., Alldiff, all of the variables involved in the constraint must have different values**
- **Preference (soft constraints)**
  - **e.g. *red* is better than *green***
  - **often can be represented by a cost for each variable assignment**
  - **combination of optimization with CSPs**

Binary Constraint Transformation Example

- Unary constraints: Just delete values from variable's domain.
- Higher order (3 variables or more): reduce to binary constraints.
- Simple example:
  - Three example variables: X, Y, Z.
  - Domains: Dx = {1,2,3}, Dy = {1,2,3}, Dz = {1,2,3}.
  - Constraints: C[X,Y,Z] = {$X+Y=Z$} = {(1,1,2), (1,2,3), (2,1,3)},
    plus many other variables and constraints elsewhere in the CSP.

  - Create a new variable, W, taking values as triples (3-tuples).
  - Domain of W is Dw = {(1,1,2), (1,2,3), (2,1,3)}.
    - Dw is exactly the tuples that satisfy the higher order constraint.
  - Create three new constraints:
    - C[X,W] = { [1, (1,1,2)], [1, (1,2,3)], [2, (2,1,3)] }.
    - C[Y,W] = { [1, (1,1,2)], [2, (1,2,3)], [1, (2,1,3)] }.
    - C[Z,W] = { [2, (1,1,2)], [3, (1,2,3)], [3, (2,1,3)] }.
  - Other constraints elsewhere involving X, Y, or Z are unaffected.

Commutativity of CSP

The order in which values are assigned to variables is irrelevant to the final assignment, hence:

1. **Generate successors of a node by considering assignments for only one variable**

2. **Do not store the path to node**

Summary

- **Constraint Satisfaction Problem**

What I want you to do

- Review Chapter 6
- Work on your assignment and submit it by Feb. 15th