

CSC 665: Artificial Intelligence

Constraint Satisfaction Problems

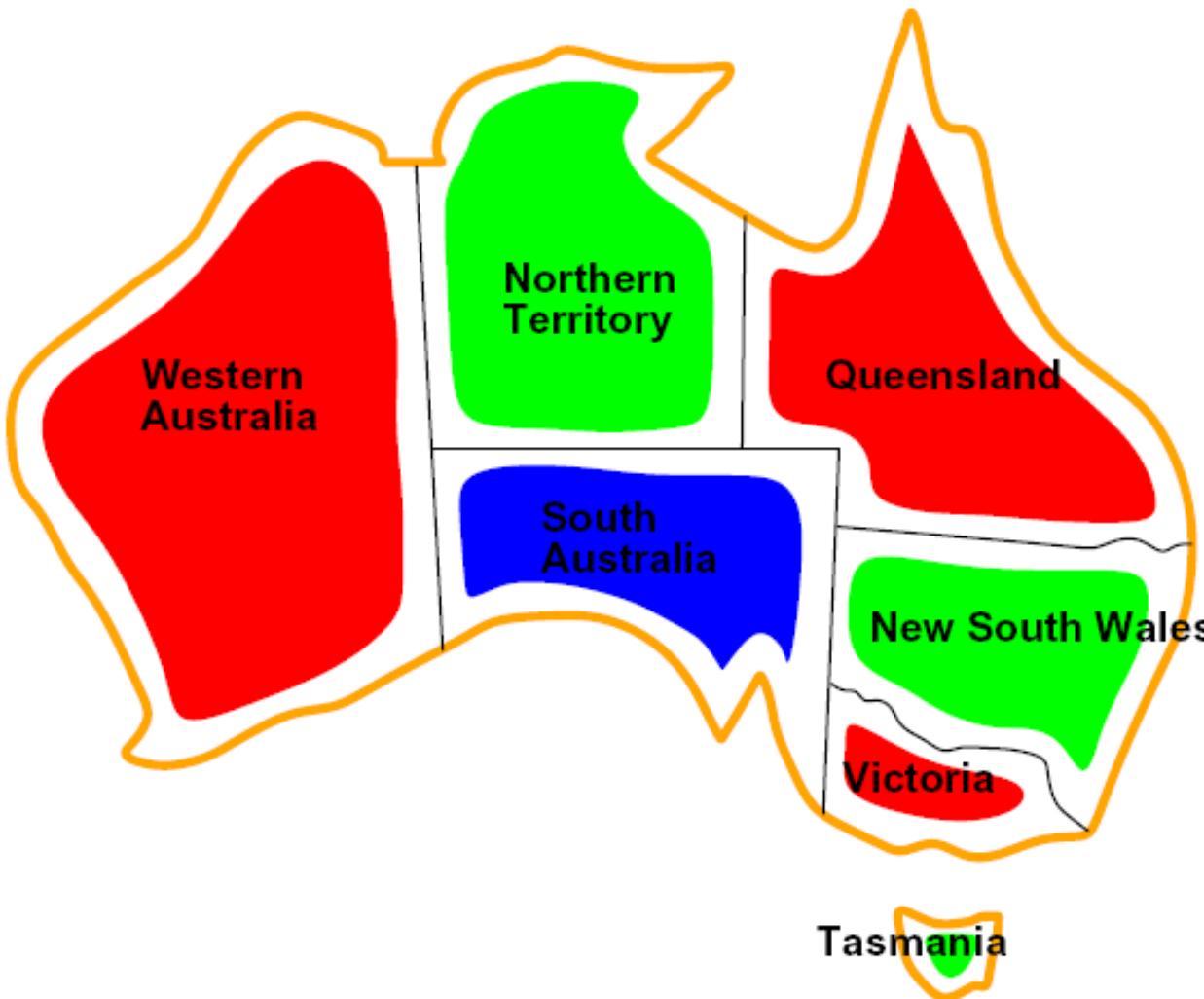
Instructor: Pooyan Fazli

San Francisco State University

Constraint Satisfaction Problems

- Constraint satisfaction problems (CSPs):
 - A special subset of search problems
 - State is defined by a set of variables X_i and each variable has a domain D_i
 - Goal test is a set of constraints specifying allowable combinations of values for subsets of variables

CSP Examples



Example: Map Coloring

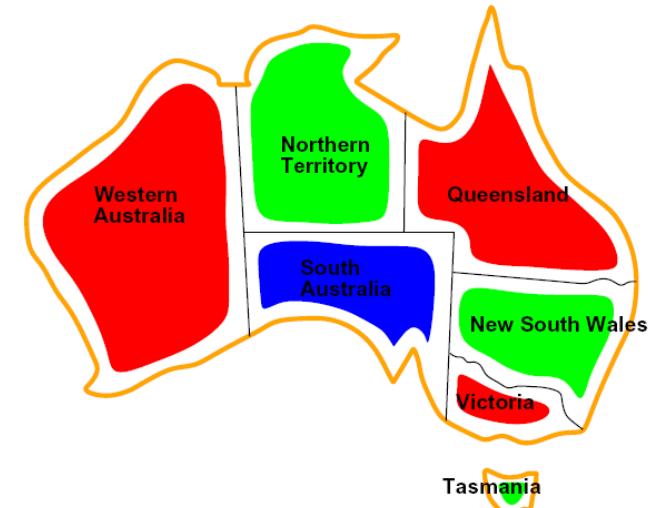
- Variables: WA, NT, Q, NSW, V, SA, T
- Domains: $D = \{\text{red, green, blue}\}$
- Constraints: adjacent regions must have different colors

Implicit: $\text{WA} \neq \text{NT}$

Explicit: $(\text{WA}, \text{NT}) \in \{(\text{red, green}), (\text{red, blue}), \dots\}$

- Solutions are assignments satisfying all constraints, e.g.:

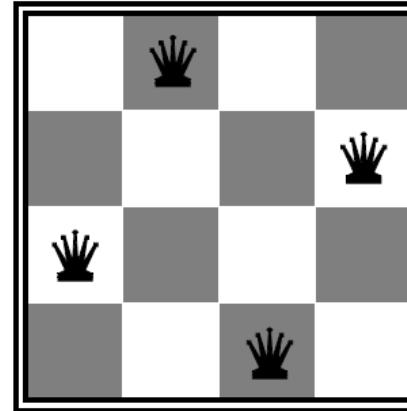
$\{\text{WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green}\}$



Example: N-Queens

- Formulation 1:

- Variables: X_{ij}
- Domains: $\{0, 1\}$
- Constraints



$$\forall i, j, k \quad (X_{ij}, X_{ik}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\forall i, j, k \quad (X_{ij}, X_{kj}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\forall i, j, k \quad (X_{ij}, X_{i+k, j+k}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\forall i, j, k \quad (X_{ij}, X_{i+k, j-k}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\sum_{i,j} X_{ij} = N$$

Example: N-Queens

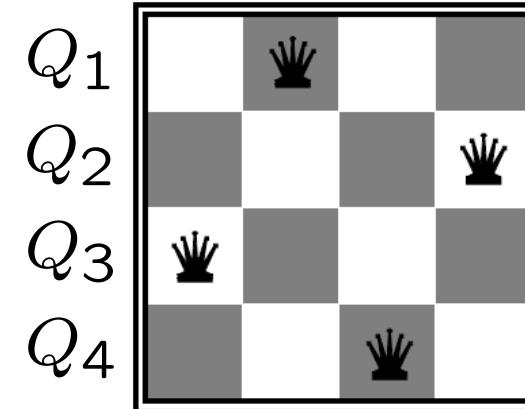
- Formulation 2:

- Variables: Q_k (Rows)
- Domains: $\{1, 2, 3, \dots, N\}$
Queen position at each row
- Constraints:

Implicit: $\forall i, j \text{ non-threatening}(Q_i, Q_j)$

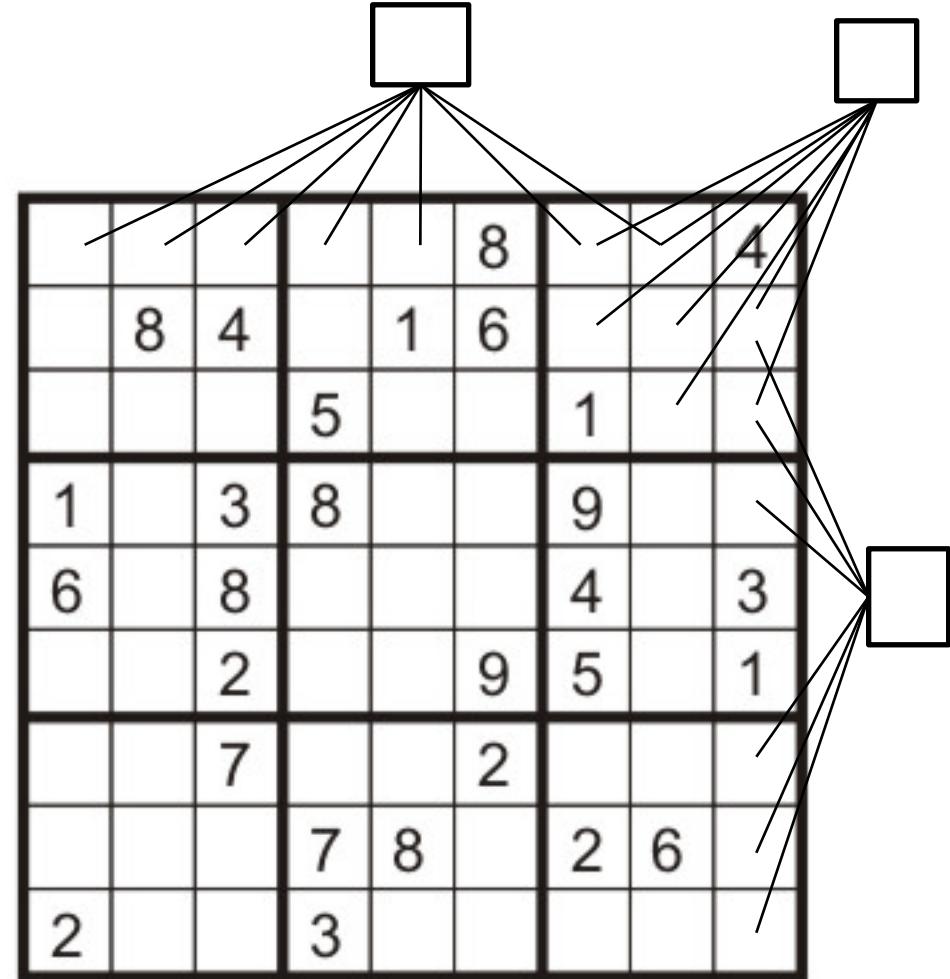
Explicit: $(Q_1, Q_2) \in \{(1, 3), (1, 4), \dots\}$

...



Example: Sudoku

- **Variables:**
 - Each (open) square
- **Domains:**
 - $\{1, 2, \dots, 9\}$
- **Constraints:**
 - 9-way alldiff for each column
 - 9-way alldiff for each row
 - 9-way alldiff for each region



Constraint Satisfaction Problems: Definition

Definition:

A Constraint Satisfaction Problem (CSP) consists of:

- a set of variables V
- a domain $\text{dom}(V)$ for each variable $V \in V$
- a set of constraints C

Simple example:

- $V = \{V_1\}$
 - $\text{dom}(V_1) = \{1,2,3,4\}$
- $C = \{C_1, C_2\}$
 - $C_1: V_1 \neq 2$
 - $C_2: V_1 > 1$

Another example:

- $V = \{V_1, V_2\}$
 - $\text{dom}(V_1) = \{1,2,3\}$
 - $\text{dom}(V_2) = \{1,2\}$
- $C = \{C_1, C_2, C_3\}$
 - $C_1: V_2 \neq 2$
 - $C_2: V_1 + V_2 < 5$
 - $C_3: V_1 > V_2$

Models of a CSP

Definition:

A **model** of a CSP is an assignment of values to all of its variables that **satisfies** all of its constraints.

Simple example:

- $V = \{V_1\}$
 - $\text{dom}(V_1) = \{1,2,3,4\}$
- $C = \{C_1, C_2\}$
 - $C_1: V_1 \neq 2$
 - $C_2: V_1 > 1$

All models for this CSP:

- { $V_1 = 3$ }
- { $V_1 = 4$ }

Models of a CSP

- $\mathcal{V} = \{V_1, V_2\}$
 - $\text{dom}(V_1) = \{1, 2, 3\}$
 - $\text{dom}(V_2) = \{1, 2\}$
- $C = \{C_1, C_2, C_3\}$
 - $C_1: V_2 \neq 2$
 - $C_2: V_1 + V_2 < 5$
 - $C_3: V_1 > V_2$

Which are models for this CSP?

{ $V_1=1, V_2=1$ }

{ $V_1=2, V_2=1$ }

{ $V_1=3, V_2=1$ }

{ $V_1=3, V_2=2$ }



Varieties of Variables

- Discrete Variables
 - Finite domains
 - Size d means $O(d^n)$ complete assignments
 - Infinite domains (integers, strings, etc.)
- Continuous variables
 - E.g., start/end times for Hubble Telescope observations

Varieties of Constraints

- **Varieties of Constraints**

- Unary constraints involve a single variable (equivalent to reducing domains), e.g.:

$SA \neq \text{green}$

- Binary constraints involve pairs of variables, e.g.:

$SA \neq WA$

- Higher-order constraints involve 3 or more variables

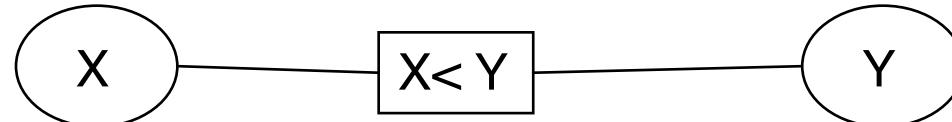
Constraint Network: Definition

Definition: A **constraint network** is defined by a graph, with

- one **node** for every **variable** (drawn as **circle**)
- one **node** for every **constraint** (drawn as **rectangle**)
- **Edges/arcs** running between **variable nodes** and **constraint nodes** whenever a given variable is involved in a given constraint.

•

- Two variables X and Y
- One constraint: $X < Y$



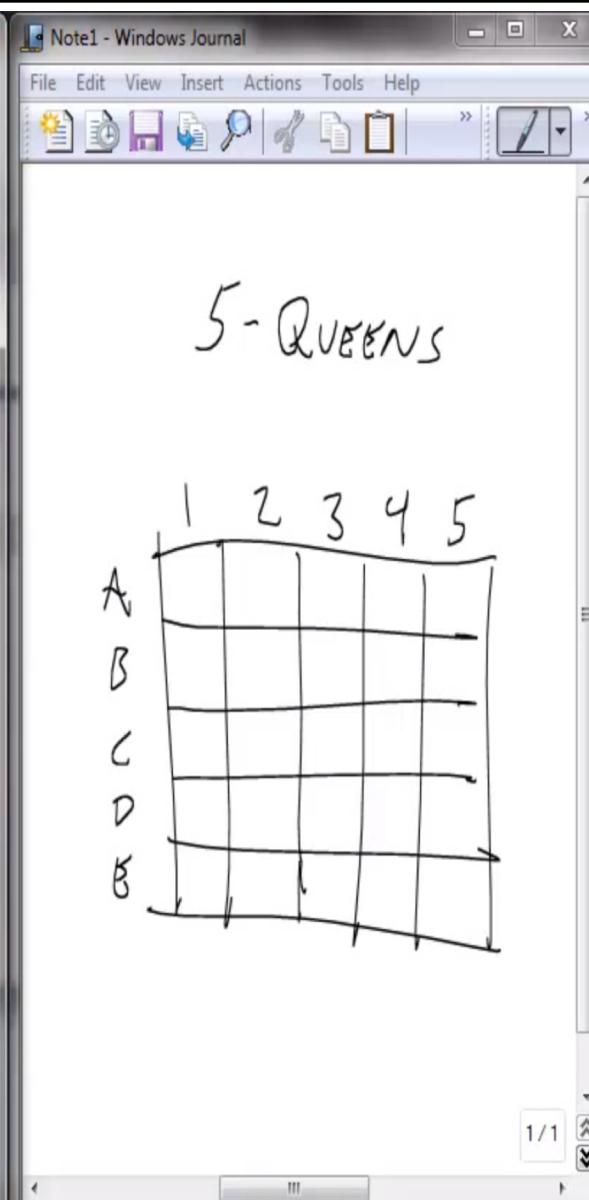
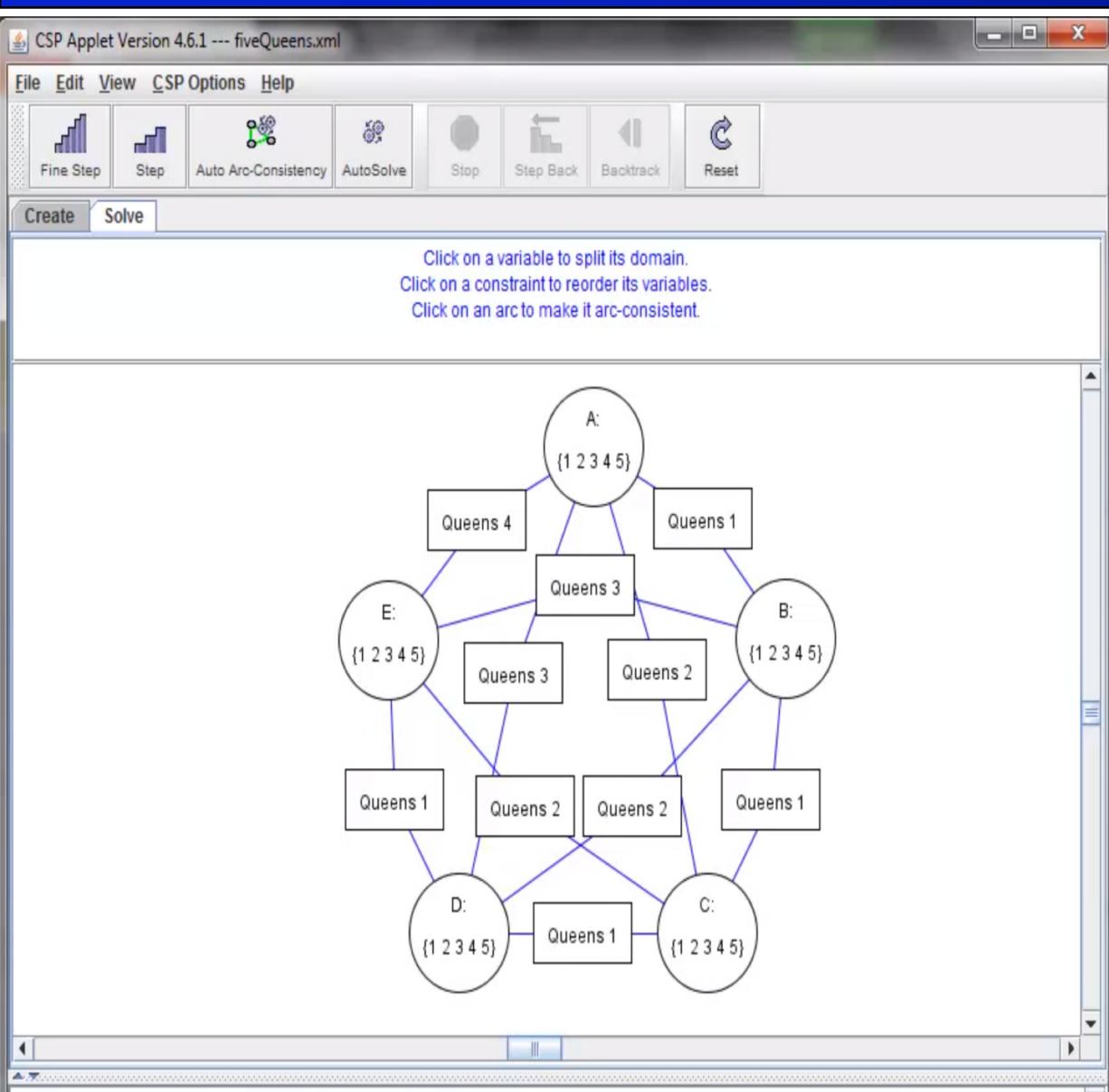
Constraint Network: Definition

Definition: A **constraint network** is defined by a graph, with

- one **node** for every **variable** (drawn as **circle**)
- one **node** for every **constraint** (drawn as **rectangle**)
- **Edges/arcs** running between **variable nodes** and **constraint nodes** whenever a given variable is involved in a given constraint.

- Whiteboard example: 3 Variables A,B,C
 - 3 Constraints: $A < B$, $B < C$, $A + 3 = C$
 - 6 edges/arcs in the constraint network:
 - $\langle A, A < B \rangle$, $\langle B, A < B \rangle$
 - $\langle B, B < C \rangle$, $\langle C, B < C \rangle$
 - $\langle A, A + 3 = C \rangle$, $\langle C, A + 3 = C \rangle$

Alspace CSP Demos



<http://www.aispace.org/>

Real-World CSPs

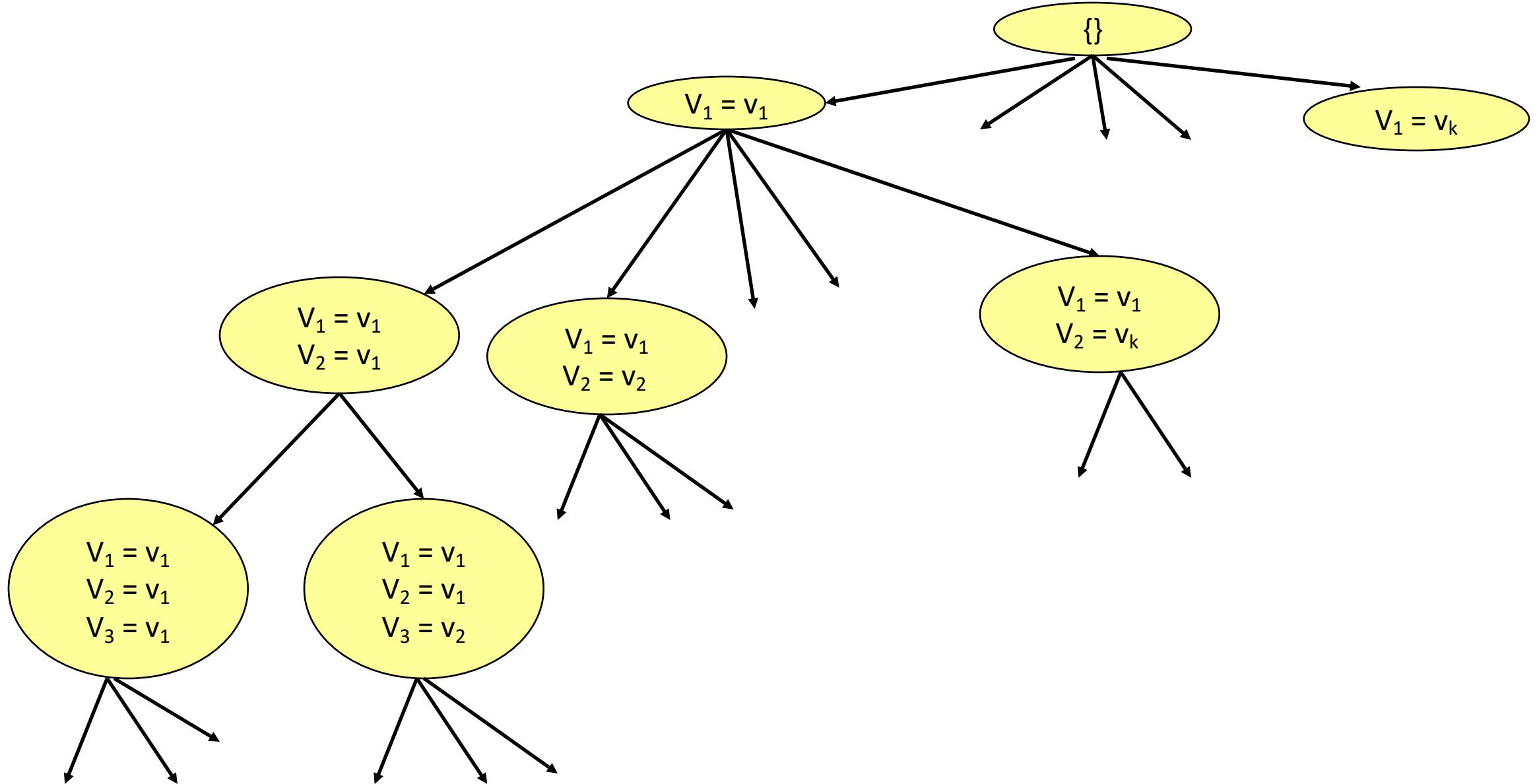
- Assignment problems: e.g., who teaches what class
 - Timetabling problems: e.g., which class is offered when and where?
 - Hardware configuration
 - Transportation scheduling
 - Factory scheduling
 - Circuit layout
 - Fault diagnosis
 - ... lots more!
-
- Many real-world problems involve real-valued variables...

Solving CSPs

Standard Search Formulation

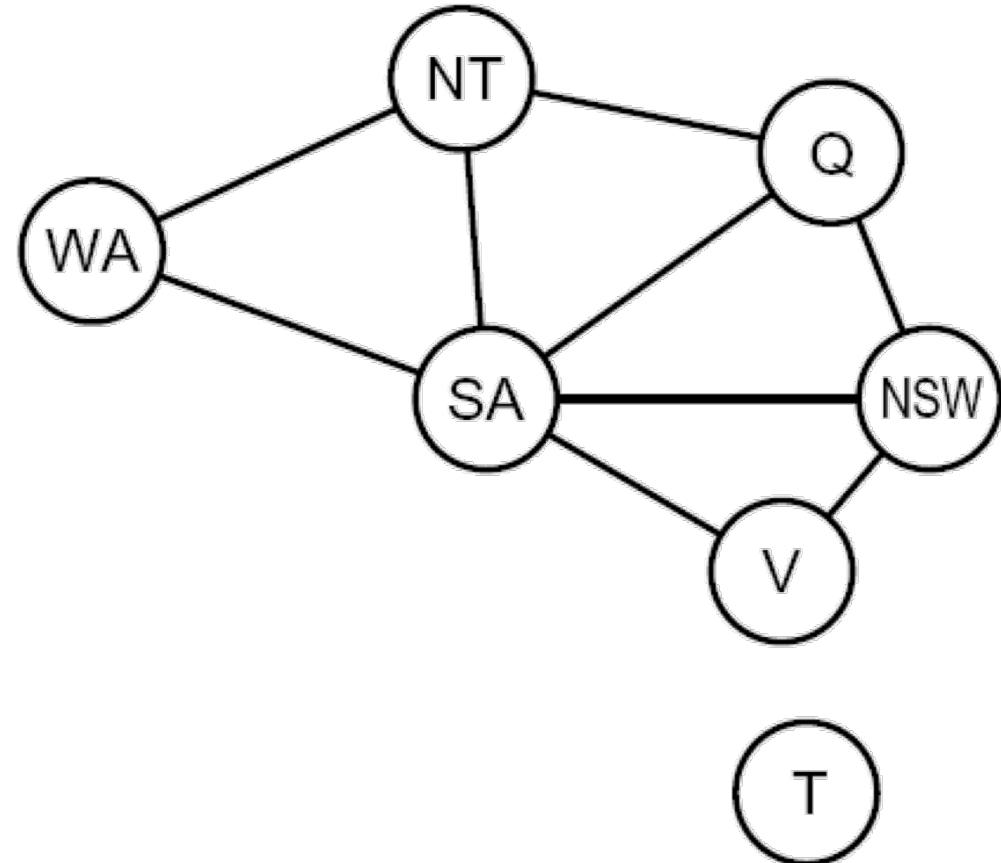
- Standard search formulation of CSPs
- States defined by the values assigned so far (partial assignments)
 - **Initial state**: the empty assignment, {}
 - **Successor function**: assign a value to an unassigned variable
 - **Goal test**: the current assignment is complete and satisfies all constraints
- We'll start with the straightforward, naïve approach, then improve it

Standard Search Formulation: Example

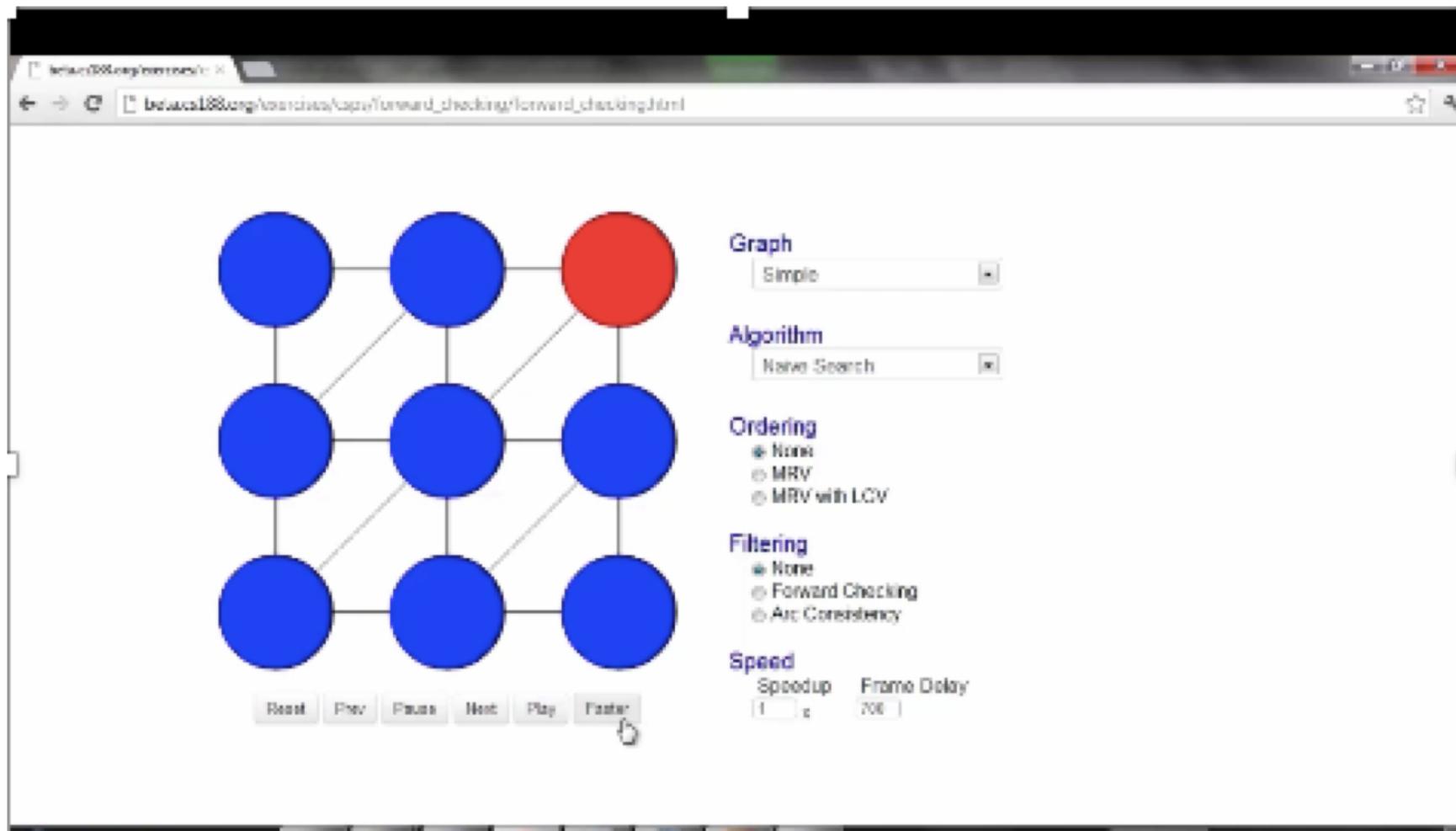


Search Methods

- What would BFS do?
- What would DFS do?



Video of Demo Coloring -- DFS



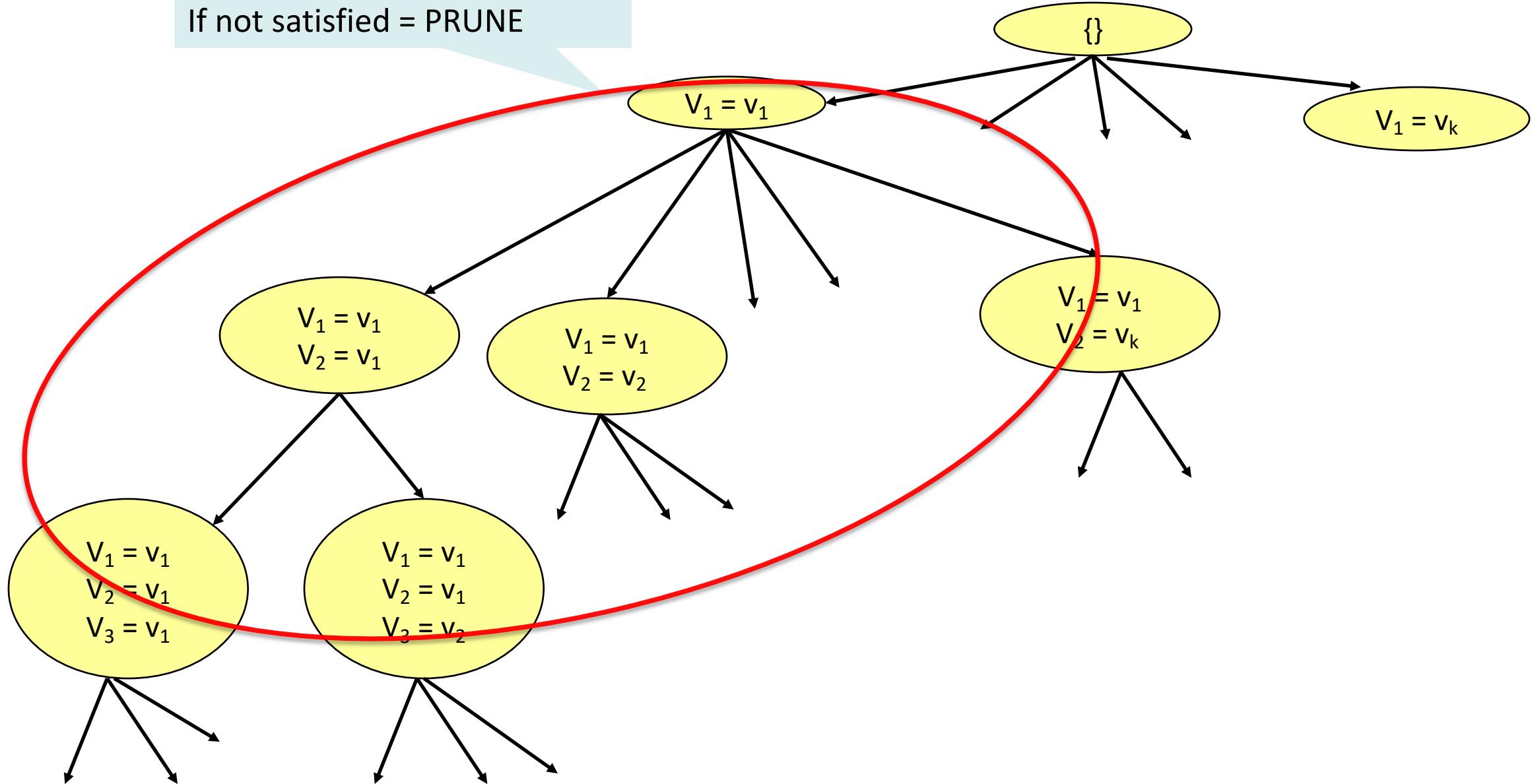
Backtracking Search

Backtracking Search

- Backtracking search is the basic uninformed algorithm for solving CSPs
- Idea: Check constraints as you go
 - I.e. consider only values which do not conflict previous assignments
 - Might have to do some computation to check the constraints
 - “Incremental goal test”
- Depth-first search with this improvement is called *backtracking search*
- Can solve n-queens for $n \approx 25$

Backtracking Example

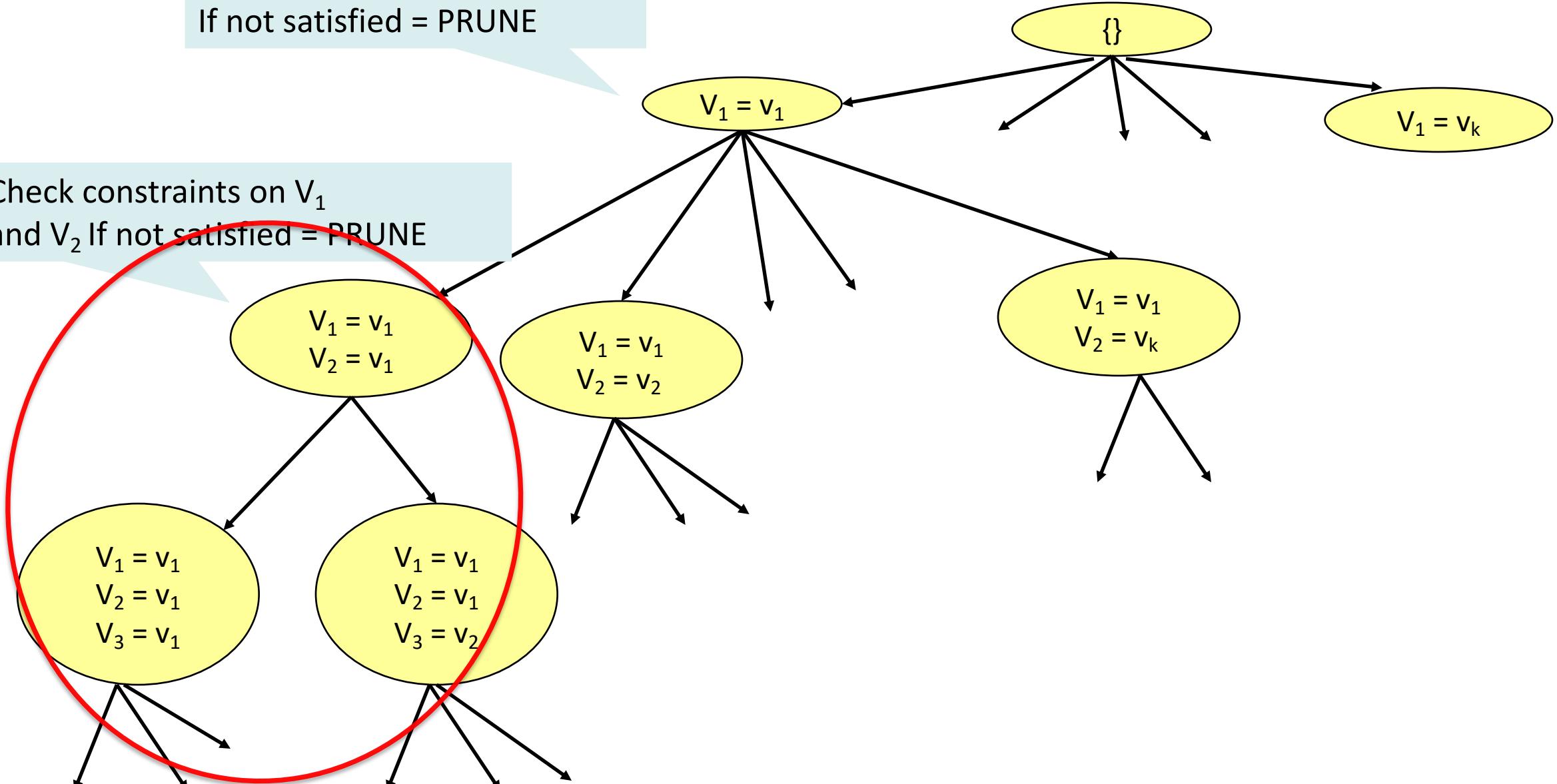
Check unary constraints on V_1
If not satisfied = PRUNE



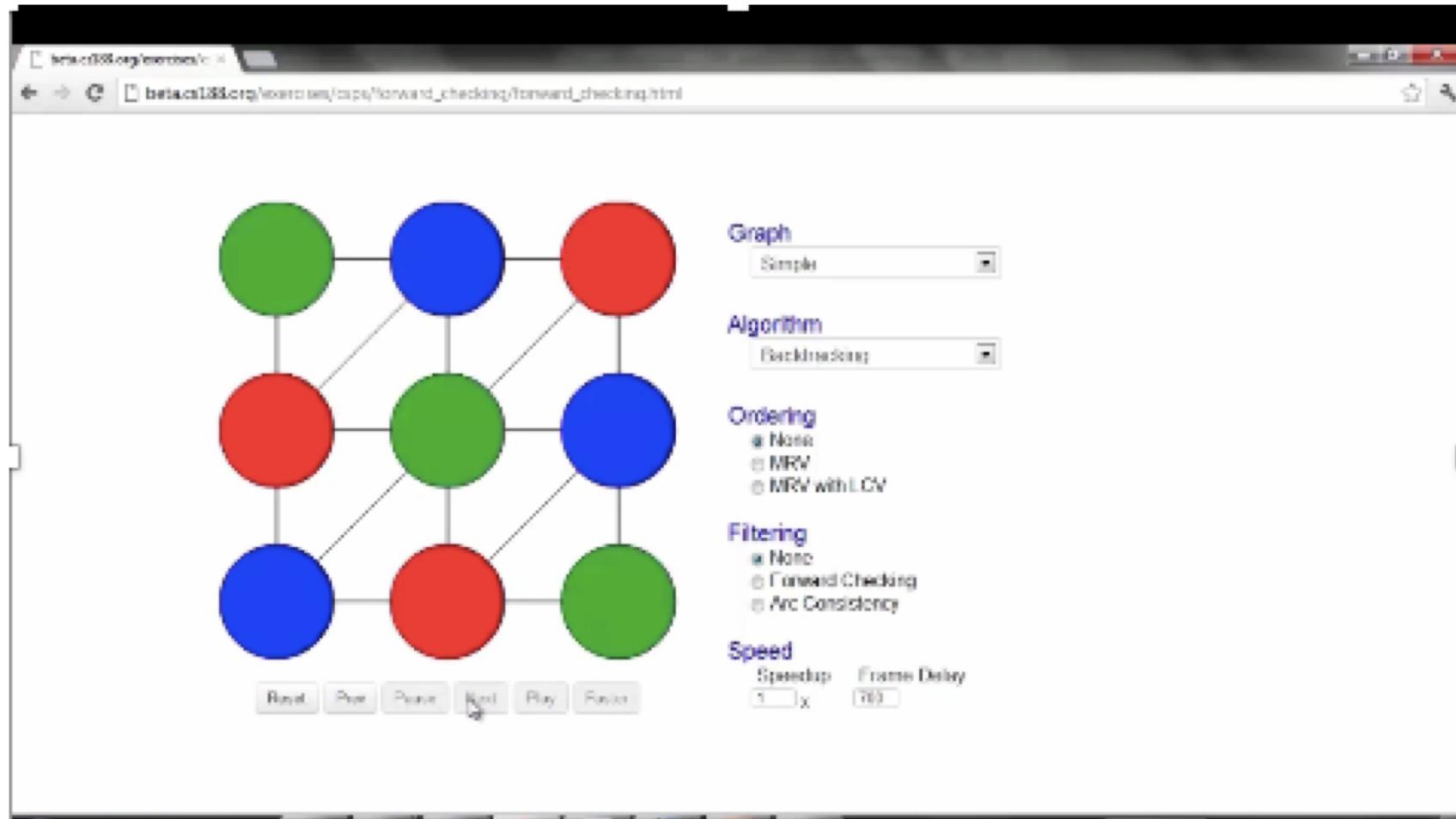
Backtracking Example

Check unary constraints on V_1
If not satisfied = PRUNE

Check constraints on V_1
and V_2 If not satisfied = PRUNE



Video of Demo Coloring – Backtracking



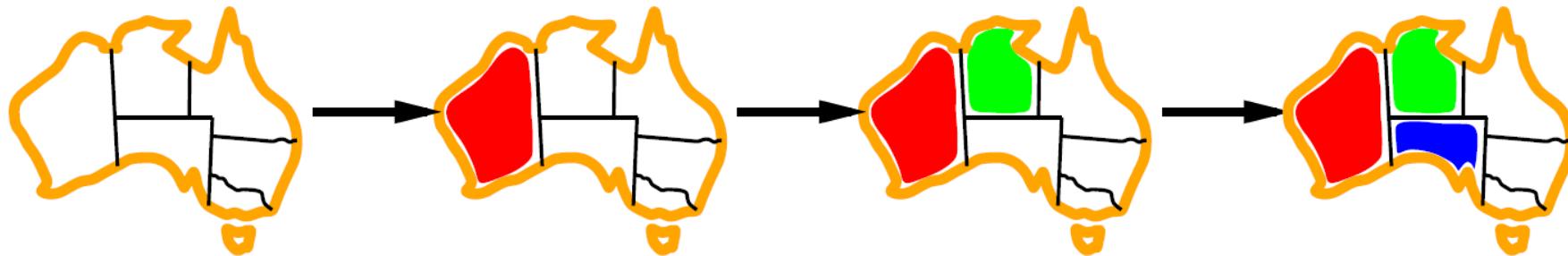
Improving Backtracking

- General-purpose ideas give huge gains in speed
- Ordering:
 - Which variable should be assigned next?
 - In what order should its values be tried?
- Filtering: Can we detect inevitable failure early?

Ordering

Ordering: Minimum Remaining Values

- **Variable Ordering: Minimum Remaining Values (MRV):**
 - Choose the variable with the fewest legal values left in its domain



- Why min rather than max?
- Also called “most constrained variable”
- “Fail-fast” ordering

Ordering: Least Constraining Value

- **Value Ordering: Least Constraining Value**
 - Given a choice of variable, choose the *least constraining value*
 - I.e., the one that rules out the fewest values in the remaining variables
 - Note that it may take some computation to determine this!
- **Why least rather than most?**
 - Tries to avoid failure by assigning values that leave maximal flexibility for the remaining variables.
- Combining these ordering ideas makes 1000 queens feasible



Filtering

Can we detect inevitable failure early?

Node Consistency

- Prune the domains as much as possible before searching for a solution.

Definition: A variable is **Node Consistent** if no value of its domain is ruled impossible by any unary constraints.

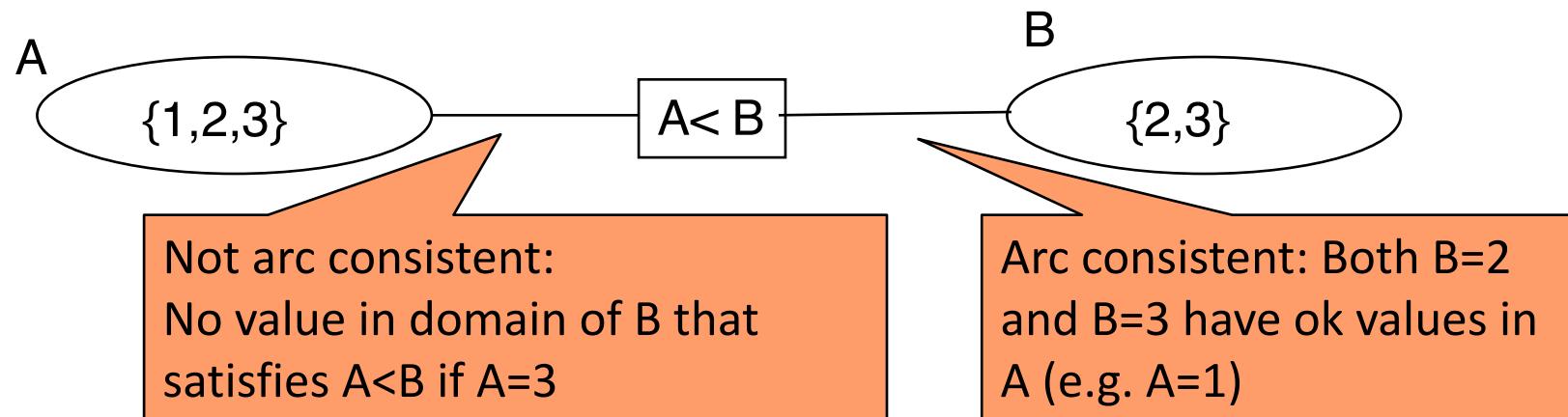
- Example: $\text{dom}(V_2) = \{1, 2, 3, 4\}$. $V_2 \neq 2$
- Variable V_2 is not node consistent.
 - It is node consistent once we remove 2 from its domain.
- Trivial for unary constraints. Trickier for k-ary ones.

Arc Consistency

Definitions:

An arc $\langle X, c(X,Y) \rangle$ is arc consistent if for each value x in $\text{dom}(X)$ there is some value y in $\text{dom}(Y)$ such that $c(X,Y)$ is satisfied.

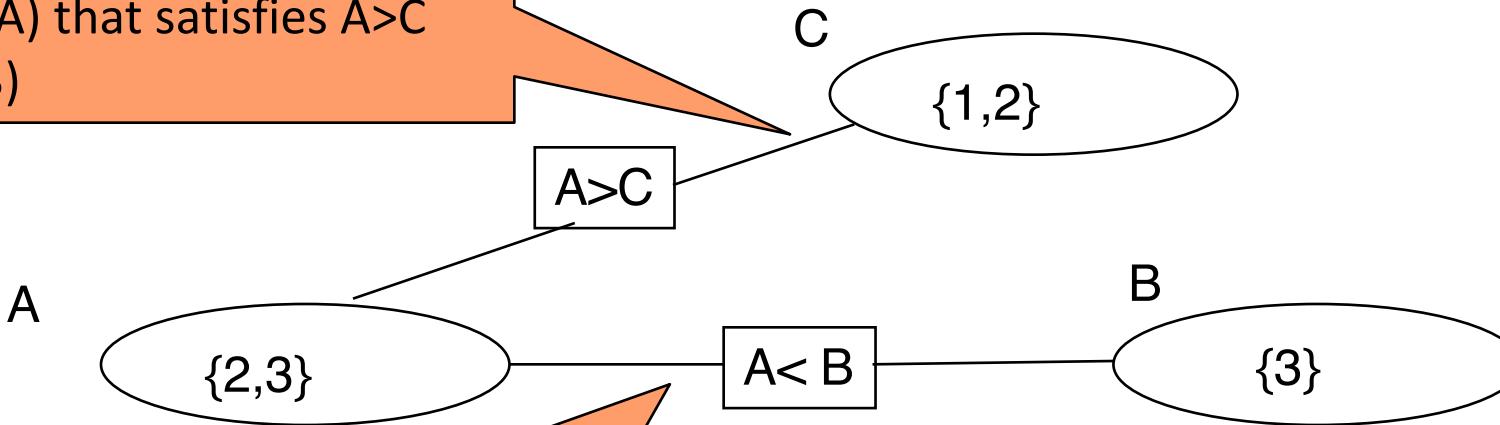
A network is arc consistent if all its arcs are arc consistent.



Arc Consistency

Arc consistent:

For each value in $\text{dom}(C)$, there is one in $\text{dom}(A)$ that satisfies $A > C$ (namely $A=3$)

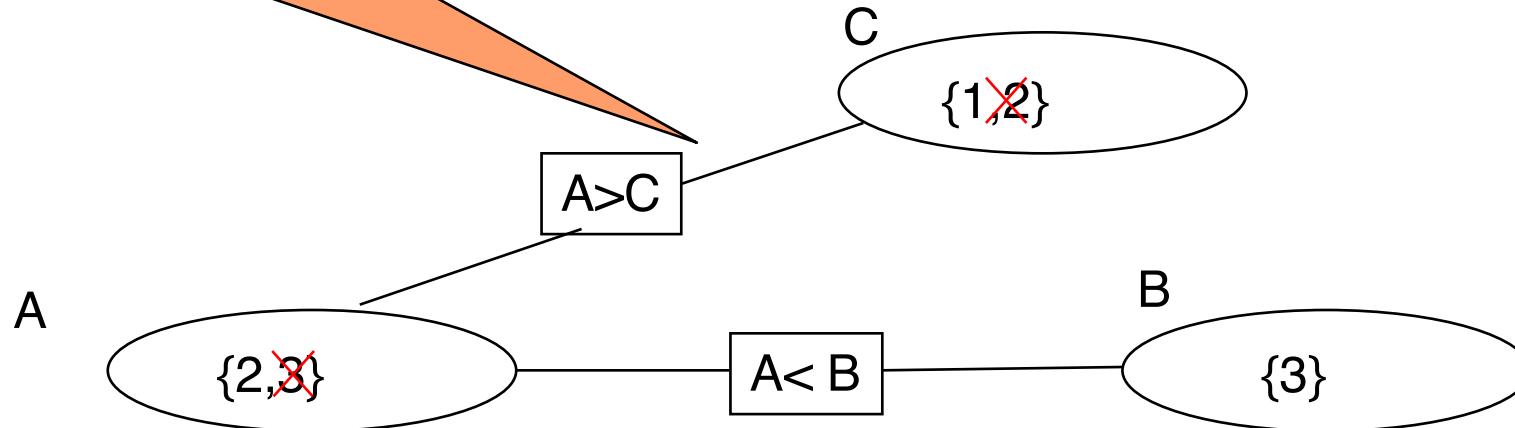


Not arc consistent:

No value in domain of B that satisfies $A < B$ if $A=3$

Arc Consistency

Not arc consistent anymore:
For C=2, there is no value in
 $\text{dom}(A)$ that satisfies $A > C$



Arc Consistency: High Level Strategy

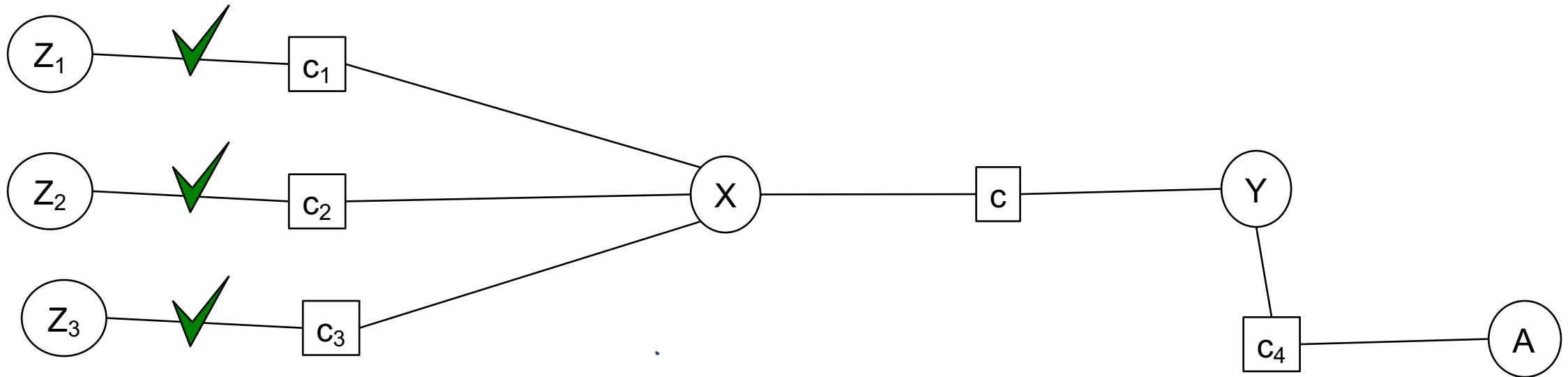
- Consider the arcs in turn, making each arc consistent
 - Reconsider arcs that could be made inconsistent again by this pruning of the domains
- Eventually reach a ‘fixed point’: all arcs consistent
- Run ‘Simple Problem 1’ in Alspace for an example:



- Arc Consistency can be run as a preprocessor or after each assignment.

Arc Consistency Quiz

- When we reduce the domain of a variable X to make an arc $\langle X, c \rangle$ arc consistent, which arcs do we need to reconsider?



- You do not need to reconsider other arcs
 - If arc $\langle Y, c \rangle$ was arc consistent before, it will still be arc consistent
 - If an arc $\langle X, c' \rangle$ was arc consistent before, it will still be arc consistent
 - Nothing changes for arcs of constraints not involving X

Arc Consistency Algorithm: Interpreting Outcomes

- Three possible outcomes
(when all arcs are arc consistent):
 - Each domain has a single value
 - We have a (unique) solution.
 - At least one domain is empty
 - No solution! All values are ruled out for this variable.
 - Some domains have more than one value
 - There may be a solution, multiple ones, or none
 - Need to solve this new CSP (usually simpler) problem:
same constraints, domains have been reduced

Arc Consistency Quiz

Can we have an arc consistent network with non-empty domains that has no solution?

YES



NO

- Example: vars A, B, C with domain {1, 2} and constraints $A = B$, $B = C$, $A \neq C$
- Or see Alspace CSP applet “Simple Problem 2”

Domain Splitting (or Case Analysis)

- Arc consistency ends: Some domains have more than one value → may or may not have a solution
 - A. Apply Backtracking Search or
 - B. Split the problem in a number of disjoint cases:

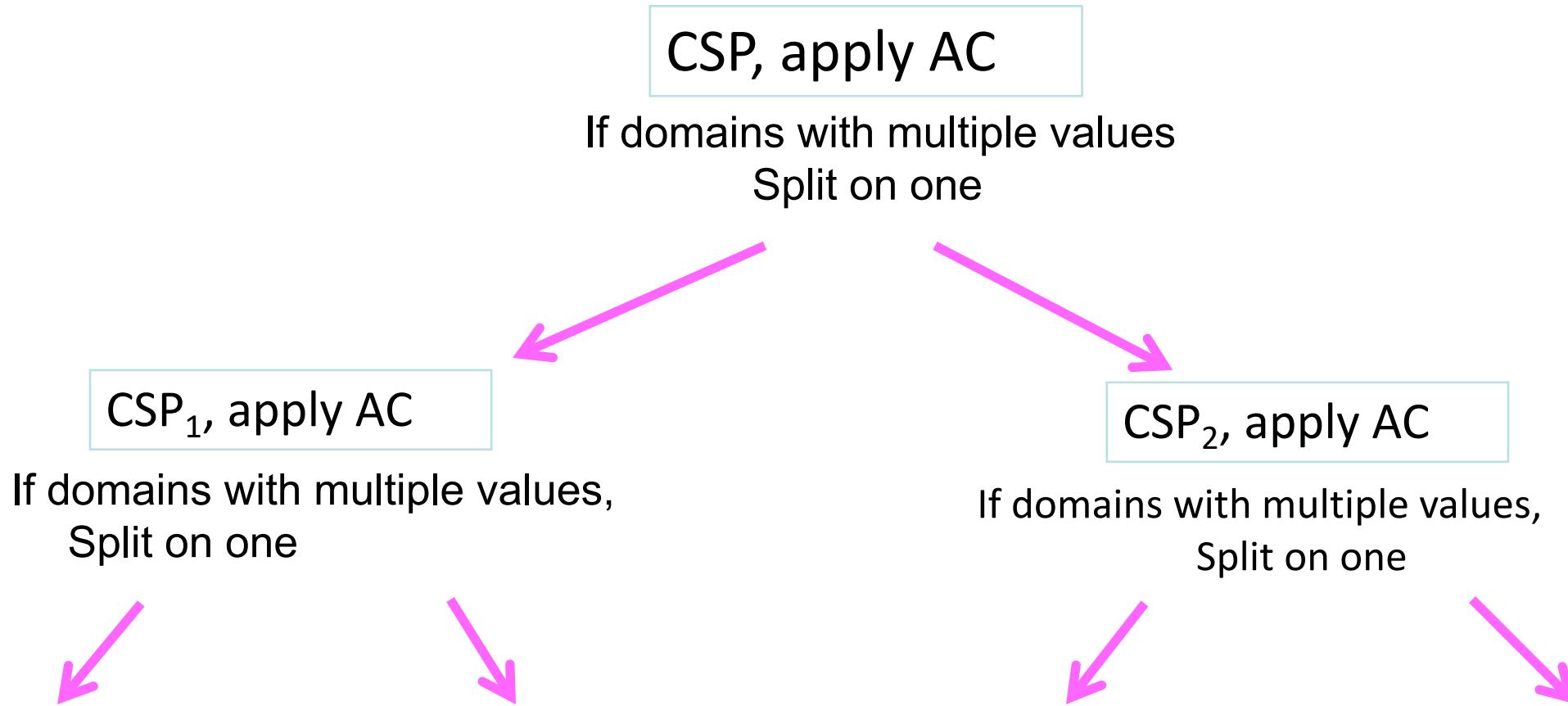
CSP with $\text{dom}(X) = \{x_1, x_2, x_3, x_4\}$ becomes

CSP_1 with $\text{dom}(X) = \{x_1, x_2\}$ and

CSP_2 with $\text{dom}(X) = \{x_3, x_4\}$

- Solution to CSP is the **union** of solutions to CSP_i

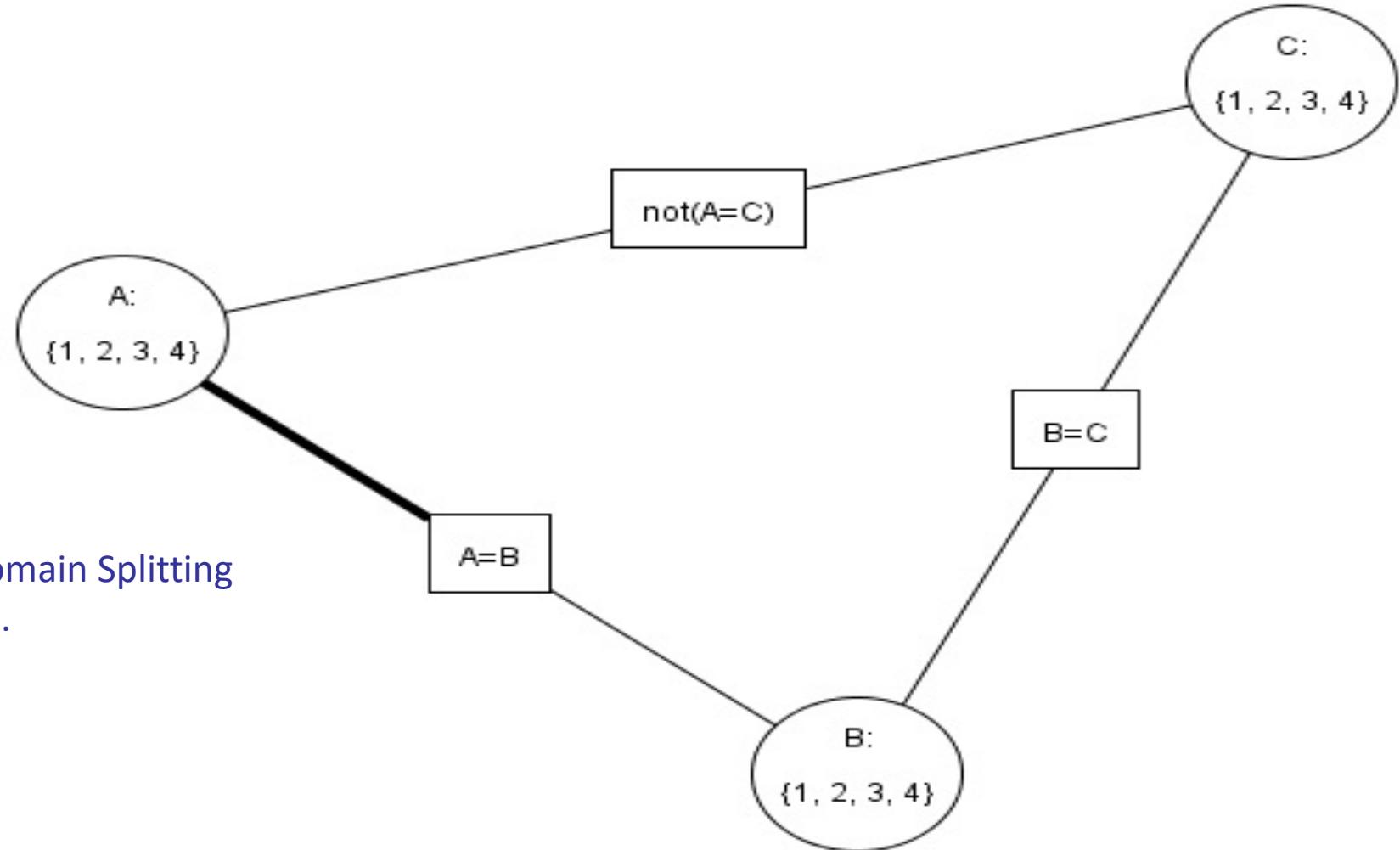
Searching by Domain Splitting



Example: ‘Simple Problem 2’ in Alspace

- 3 variables: A, B, C
- Domains: all {1,2,3,4}
- $A=B$, $B=C$, $A \neq C$

- Trace Arc Consistency + Domain Splitting
for this network in Alspace.



Example: ‘Simple Problem 2’ in Alspace

3 variables: A, B, C

Domains: all $\{1,2,3,4\}$

$A=B$, $B=C$, $A \neq C$

$(\{1,2,3,4\}, \{1,2,3,4\}, \{1,2,3,4\})$

\downarrow AC (arc consistency)

$(\{1,2,3,4\}, \{1,2,3,4\}, \{1,2,3,4\})$

$A \in \{1,3\}$

$A \in \{2,4\}$

$(\{1,3\}, \{1,2,3,4\}, \{1,2,3,4\})$

$(\{2,4\}, \{1,2,3,4\}, \{1,2,3,4\})$

\downarrow AC

\downarrow AC

$(\{1,3\}, \{1,3\}, \{1,3\})$

$(\{2,4\}, \{2,4\}, \{2,4\})$

$B \in \{1\}$

$B \in \{3\}$

$B \in \{2\}$

$B \in \{4\}$

$(\{1,3\}, \{1\}, \{1,3\})$

$(\{1,3\}, \{3\}, \{1,3\})$

$(\{2,4\}, \{2\}, \{2,4\})$

$(\{2,4\}, \{4\}, \{2,4\})$

\downarrow AC

\downarrow AC

\downarrow AC

\downarrow AC

$(\{\}, \{\}, \{\})$

$(\{\}, \{\}, \{\})$

$(\{\}, \{\}, \{\})$

$(\{\}, \{\}, \{\})$

No solution

No solution

No solution

No solution

Example: ‘Simple Problem 2’ in Alspace

3 variables: A, B, C

Domains: all $\{1,2,3,4\}$

$A=B$, $B=C$, $A=C$

$(\{1,2,3,4\}, \{1,2,3,4\}, \{1,2,3,4\})$

↓ AC (arc consistency)

$(\{1,2,3,4\}, \{1,2,3,4\}, \{1,2,3,4\})$

$A \in \{1,3\}$

$A \in \{2,4\}$

$(\{1,3\}, \{1,2,3,4\}, \{1,2,3,4\})$

$(\{2,4\}, \{1,2,3,4\}, \{1,2,3,4\})$

↓ AC

↓ AC

$(\{1,3\}, \{1,3\}, \{1,3\})$

$(\{2,4\}, \{2,4\}, \{2,4\})$

$B \in \{1\}$

$B \in \{3\}$

$B \in \{2\}$

$B \in \{4\}$

$(\{1,3\}, \{1\}, \{1,3\})$

$(\{1,3\}, \{3\}, \{1,3\})$

$(\{2,4\}, \{2\}, \{2,4\})$

$(\{2,4\}, \{4\}, \{2,4\})$

↓ AC

↓ AC

↓ AC

↓ AC

$(\{1\}, \{1\}, \{1\})$

$(\{3\}, \{3\}, \{3\})$

$(\{2\}, \{2\}, \{2\})$

$(\{4\}, \{4\}, \{4\})$

Solution

Solution

Solution

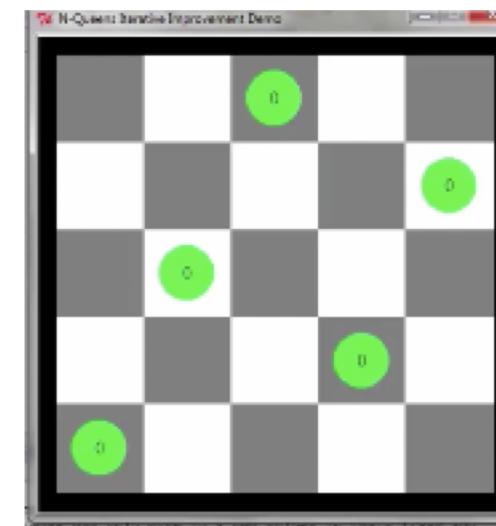
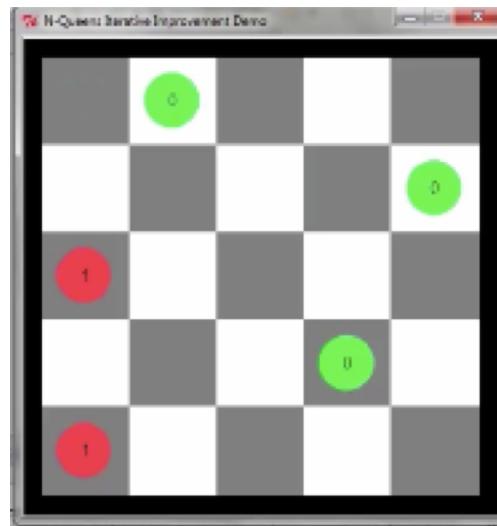
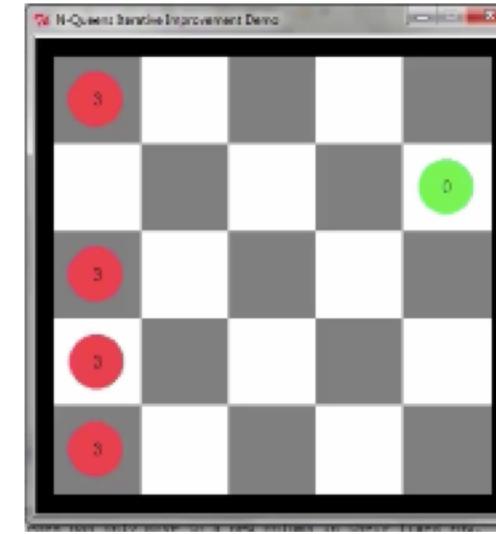
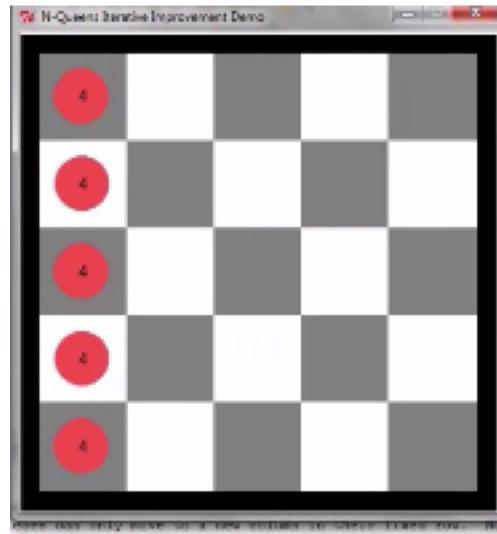
Solution

A quick note on local search for CSPs

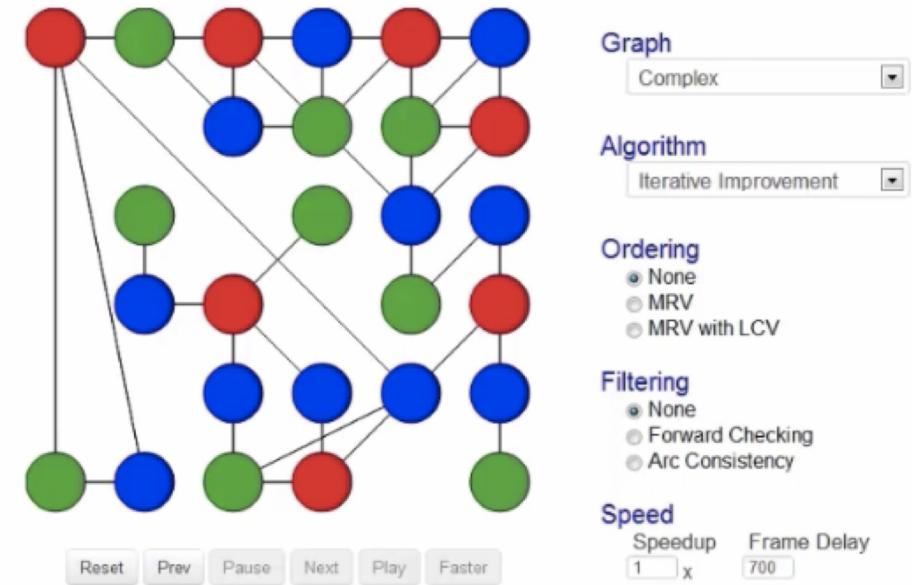
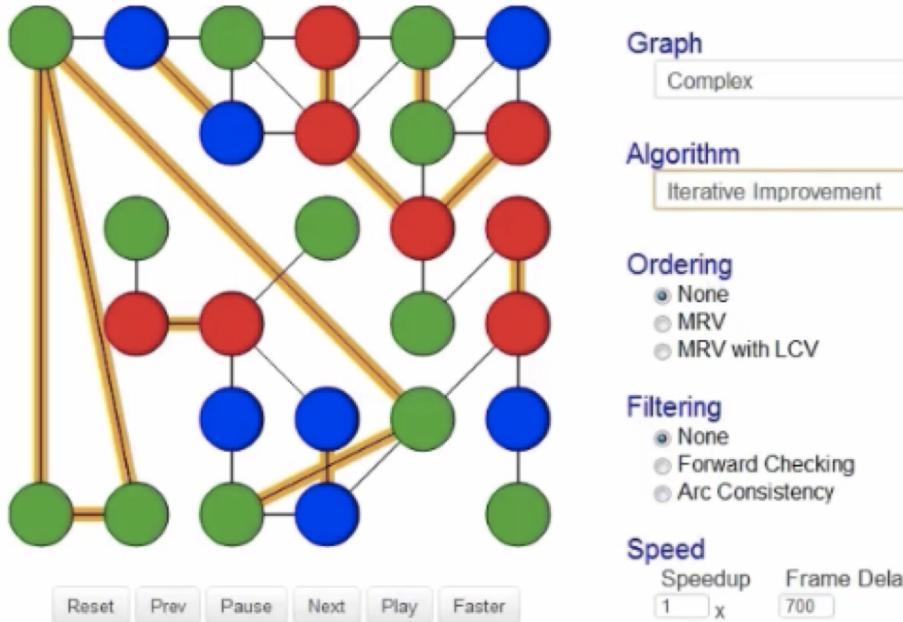
Greedy Descent with Min-Conflict Heuristic

- Algorithm: While not solved,
 - Variable selection: ***randomly select*** any conflicted variable
 - Value selection: min-conflicts heuristic:
 - Choose a value that violates the fewest constraints
 - Break ties randomly
- One of the best local search techniques for CSP solving
- Given random initial state, can solve n-queens in almost constant time for arbitrary n with high probability (e.g., n = 10,000,000)!

Demo Min-Conflict in N-Queens



Demo Min-Conflict – Coloring



Constraint Optimization Problems

- Constraint Satisfaction Problems
 - Hard constraints: need to satisfy all of them
 - All models are equally good
- Constraint Optimization Problems
 - Hard constraints: need to satisfy all of them
 - Soft constraints: need to satisfy them as well as possible
 - Can have weighted constraints
 - Minimize $h(n) = \text{sum of weights of constraints unsatisfied in } n$
 - Hard constraints have a very large weight
 - Some soft constraints can be more important than other soft constraints → larger weight
 - All local search methods work just as well for constraint optimization
 - all they need is an evaluation function h

Example for Constraint Optimization Problem

Exam scheduling

- Hard constraints:
 - Cannot have an exam in too small a room
 - Cannot have multiple exams in the same room in the same time slot
 - ...
- Soft constraints
 - Students should not have multiple exams on the same day
 - It would be nice if students had their exams spread out
 - ...

Local Search Generality: Dynamically Changing CSPs

- The problem may change over time
 - Particularly important in scheduling
 - E.g., schedule for airline:
 - Thousands of flights and thousands of personnel assignments
 - A storm can render the schedule infeasible
- Goal: Repair the schedule with minimum number of changes
 - Often easy for Local Search starting from the current schedule
 - Other techniques usually:
 - Require more time
 - Might find solutions requiring many more changes

Summary: CSPs

- CSPs are a special kind of search problem:
 - States are partial assignments
 - Goal test defined by constraints
- Basic solution: backtracking search
- Speed-ups:
 - Ordering
 - Filtering
- Greedy descent with min-conflict heuristic is often effective in practice

Reading

- Read Chapters 6, 6.1, 6.2, 6.2.1, 6.2.2, 6.2.6, 6.3, 6.3.1, 6.4 in the AIMA textbook
- Read Chapters 4.6 in the P&M textbook (Domain Splitting)