Title: Solving Problems by Searching

AIMA: Chapter 3 (Sections 3.6 and 3.7)

Introduction to Artificial Intelligence

CSCE 476-876, Spring 2000

www.cse.unl.edu/~cse476

Berthe Y. Choueiry

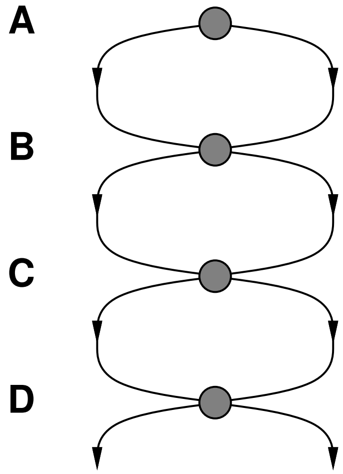
choueiry@cse.unl.edu, (402)476-5444

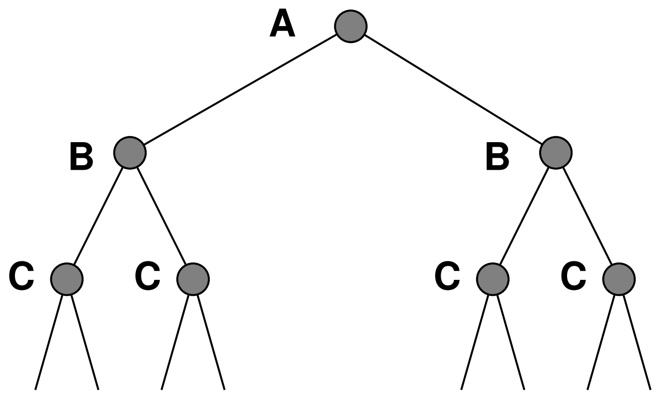
Loops: Avoid repeated states (I)

Avoid expanding states that have already been visited

Valid for both infinite and finite trees

Example: $\begin{cases} m \text{ maximum depth} \\ m+1 \text{ states} \\ 2^m \text{ possible branches (paths)} \end{cases}$

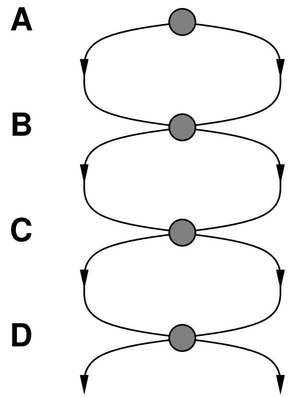


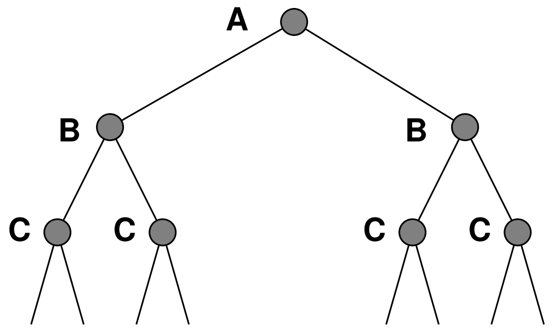


Loops: (II)

3 Strategies:
1. Do not return to the state you come from
2. Do not return to any of ancestor state
3. Do not return to any previously visited state

   ...... required space?





$\longrightarrow$ Implementation: hash table.

Trade-off cost of storing+checking vs. cost of searching

Summary

Path: sequence of actions leading from one state to another

Partial solution: a path from an initial state to another state

Search: develop a sets of partial solutions

- Search tree & its components (node, root, leaves, fringe)
- Data structure for a search node
- Search space vs. state space
- Node expansion, queue order
- Search types: uniformed vs. heuristic
- 6 uniformed search strategies
- 4 criteria for evaluating & comparing search strategies

Constraint Satisfaction Problems (CSP)

$\longrightarrow$ Course: Foundations of Constraint Satisfaction

$\longrightarrow$ CSCE-990, Fall 2000

$\longrightarrow$ Fall'99 URL: `cse.unl.edu/~choueiry/CSE990-05/`

$\longrightarrow$ Instructor: Choueiry

$\longrightarrow$ Interested? $\begin{cases} \text{Independent study} \\ \text{MS or PhD thesis} \end{cases}$

CSP: Definition

Given $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C}) \begin{cases} \mathcal{V} = \{V_1, V_2, \ldots, V_n\} \\ \mathcal{D} = \{D_{V_1}, D_{V_2}, \ldots, D_{V_n}\} \\ \mathcal{C} \text{ set of constraints} \end{cases}$

$$C_{V_k,V_l,V_m} = \{\,(x, y, z)\} \subseteq D_{V_k} \times D_{V_l} \times D_{V_m}$$

Query: can we find one value for each variable
such that all constraints are satisfied?

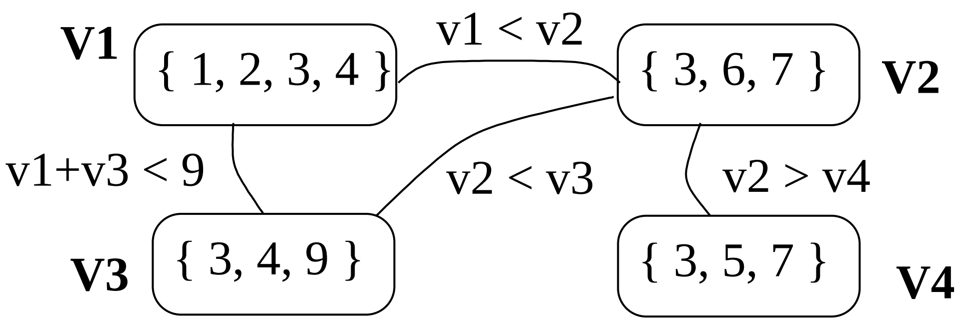
In general, **NP-complete**

Representation

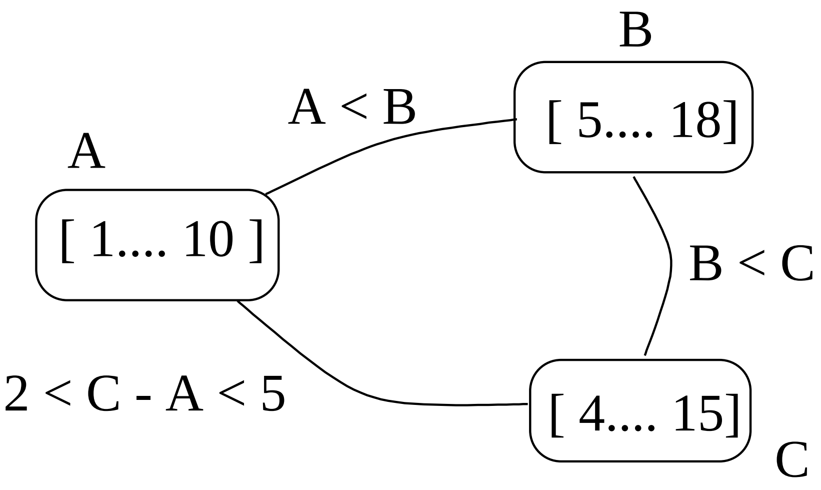
Given $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C}) \begin{cases} \mathcal{V} = \{V_1, V_2, \ldots, V_n\} \\ \mathcal{D} = \{D_{V_1}, D_{V_2}, \ldots, D_{V_n}\} \\ \mathcal{C} \;\; \text{set of constraints} \end{cases}$

$$C_{V_k, V_l, V_m} = \{ (x, y, z)\} \subseteq D_{V_k} \times D_{V_l} \times D_{V_m}$$

Constraint **graph**

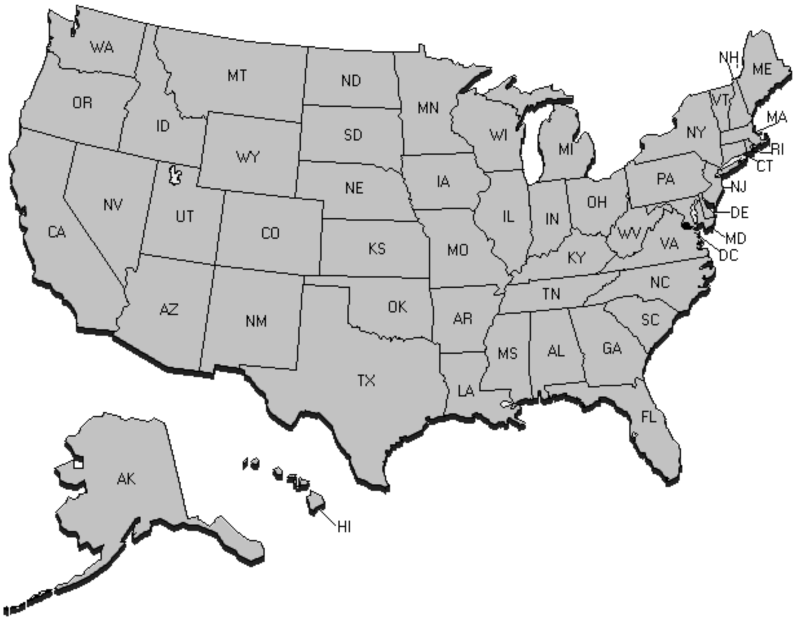


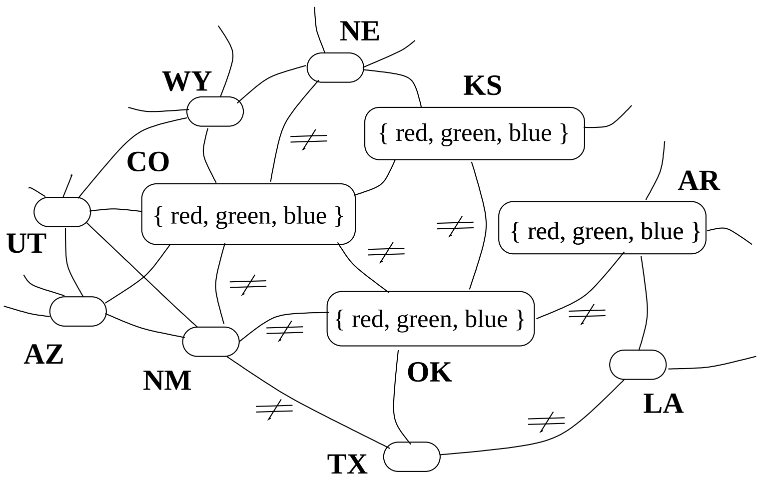
Example I: Temporal reasoning



Example II: Map coloring

Using 3 colors (R, G, & B), color the US map such that no two adjacent states do have the same color





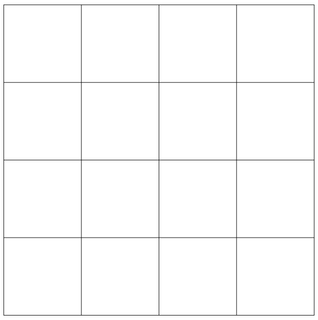
Variables? Domains? Constraints?

Example III: n-queens problem

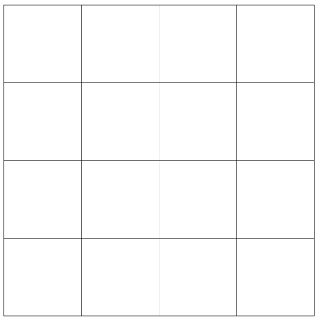
Variables?

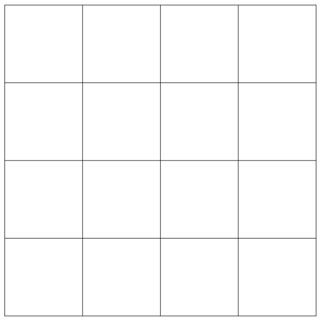
Domains?

Constraints?



Incremental formulation $\begin{cases} \text{State?} \\ \\ \text{Action?} \end{cases}$





In general

What is a state in a CSP?

What is an action in a CSP?

What is the goal state in a CSP?

What is a state in a CSP?

→ A set of instantiated variables
and a set of uninstantiated variables

What is an action in a CSP?

→ The instantiation of one more variables
(instantiation = assignment of a value to a variable)

What is the goal state in a CSP?

→ All variables are instantiated such that
all constraints are satisfied

How to solve a CSP?

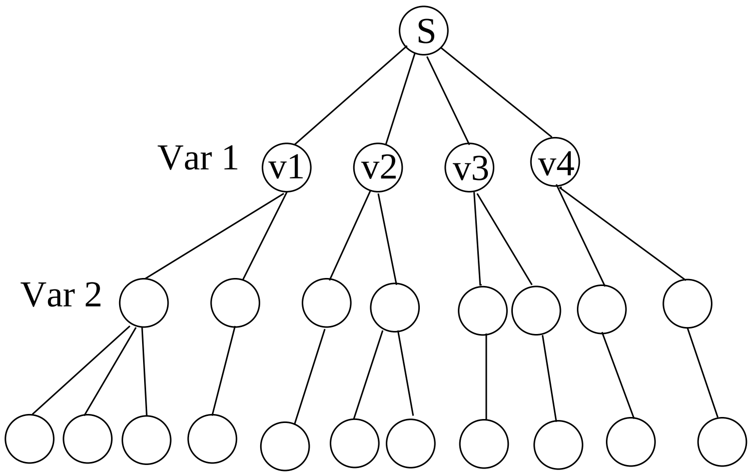
Systematic search!

→ Starting from a root node

→ Consider all values for a variable V_1

→ For every value for V_1, consider all values for V_2

→ etc..



For n variables, each of domain size d:

- Maximum depth? *fixed!*
- Maximum number of paths? *size of search space, size of CSP*

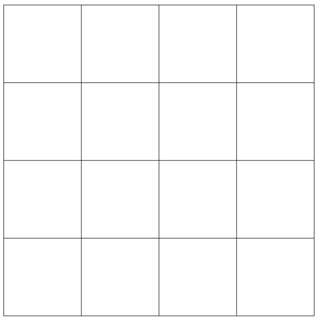
Before looking at search..

Consider

1. Importance of modeling/formulating
   to control the size of the search space
2. Preprocessing: consistency filtering
   to reduce size of search space

Importance of modeling

N-**queens**: formulation 1

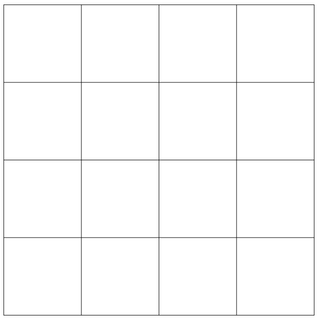


Variables?

Domains?

Size of CSP?

N-**queens**: formulation 2



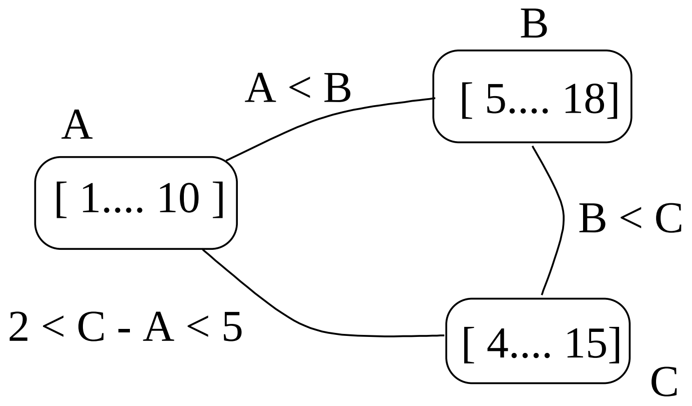
variables?

domains?

size of csp?

Constraint checking

→ arc-consistency



1- B: [5 .. 14]
C: [6 .. 15]

2- A: [2 .. 10]
C: [6 .. 14]

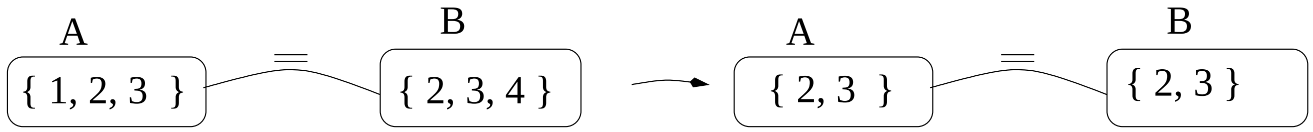
3- B: [5 .. 13]

⟶ 3-consistency, k-consistency ($k \leq n$)

⟶ Constraint filtering, constraint checking, etc..

eliminate non-acceptable tuples <u>prior</u> to search

→ Warning: arc-consistency does not solve the problem

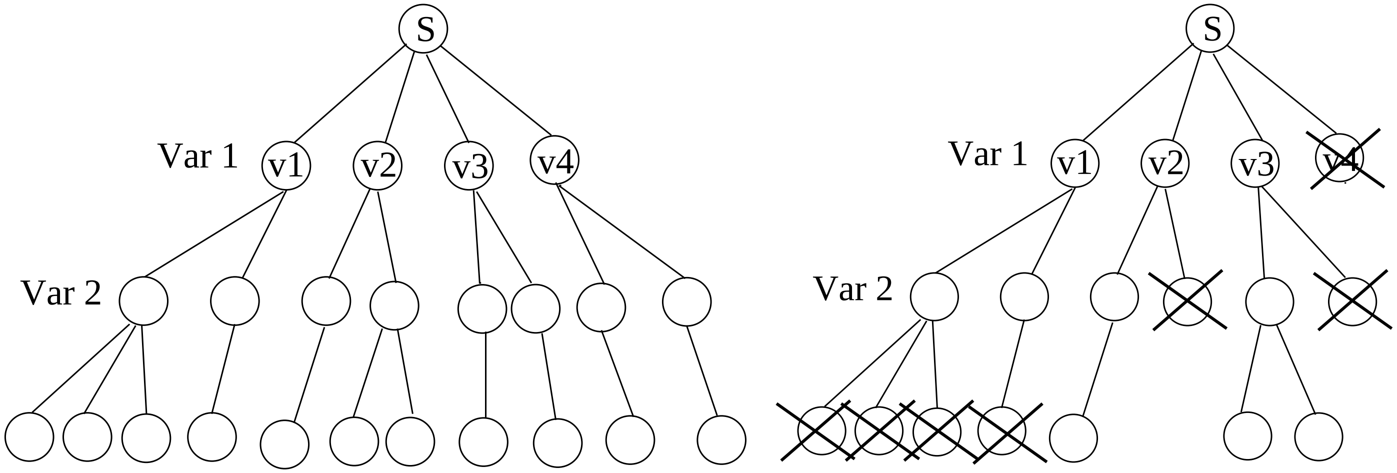


(A=2) ∧ (B=3) still isn't a solution!

Search (I) ***Back-checking***

Systematic search generates d^n possibilities

Are all possibilities acceptable?

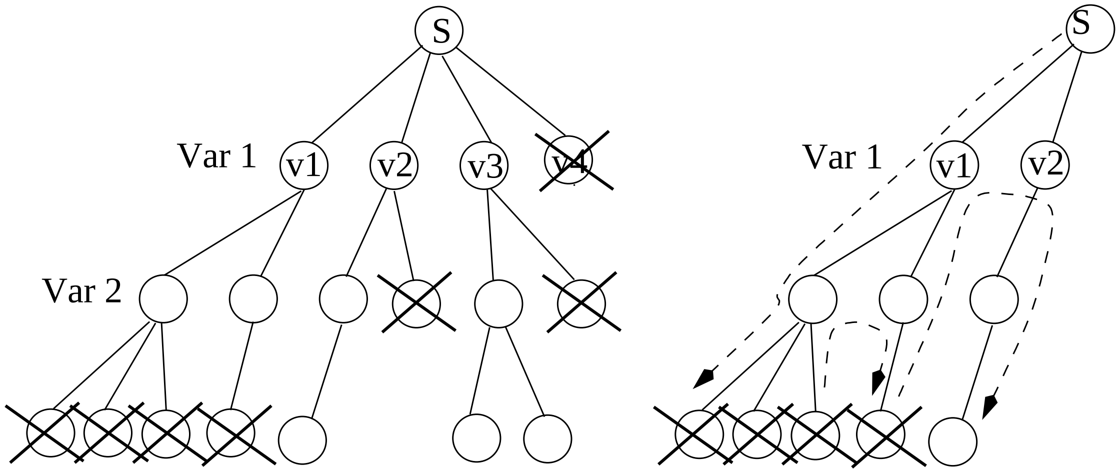


$\rightarrow$ Expand a partial solution only when consistent

$\longrightarrow$ **early pruning**

Search (II) ***Chronological backtracking***

What if only <u>one</u> solution is needed?



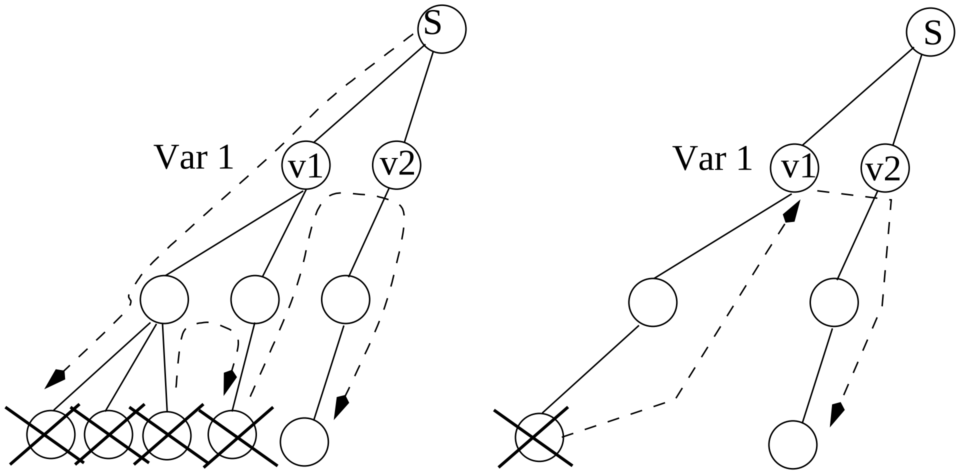
$\longrightarrow$ **Depth-first search & chronological backtracking**

$\rightarrow$ DFS: soundness? completeness?

Search (III) ***Intelligent backtracking***

What if the reason for failure was higher up in the tree?

Backtrack to source of conflict!!



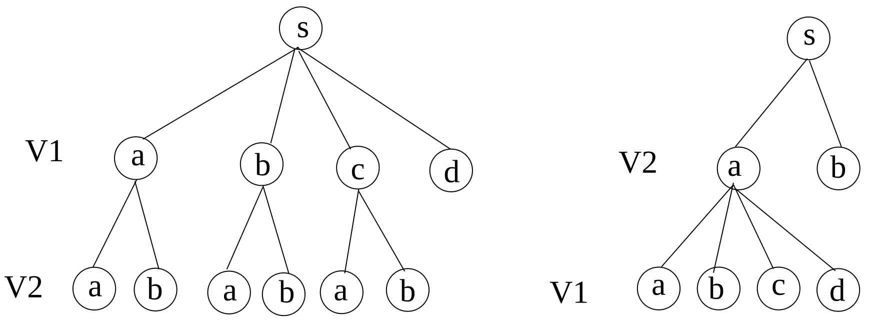
$\rightarrow$ Backjumping, conflict-directed backjumping, etc.

Search (IV) ***Ordering heuristics***

Which variable to expand first?

Exp: $V_1, V_2,\ D_{V_1} = \{a, b, c, d\}, D_{V_2} = \{a, b\}$

Sol: $\{(V_1 = c), (V_2 = a)\}$ and $\{(V_1 = c), (V_2 = b)\}$

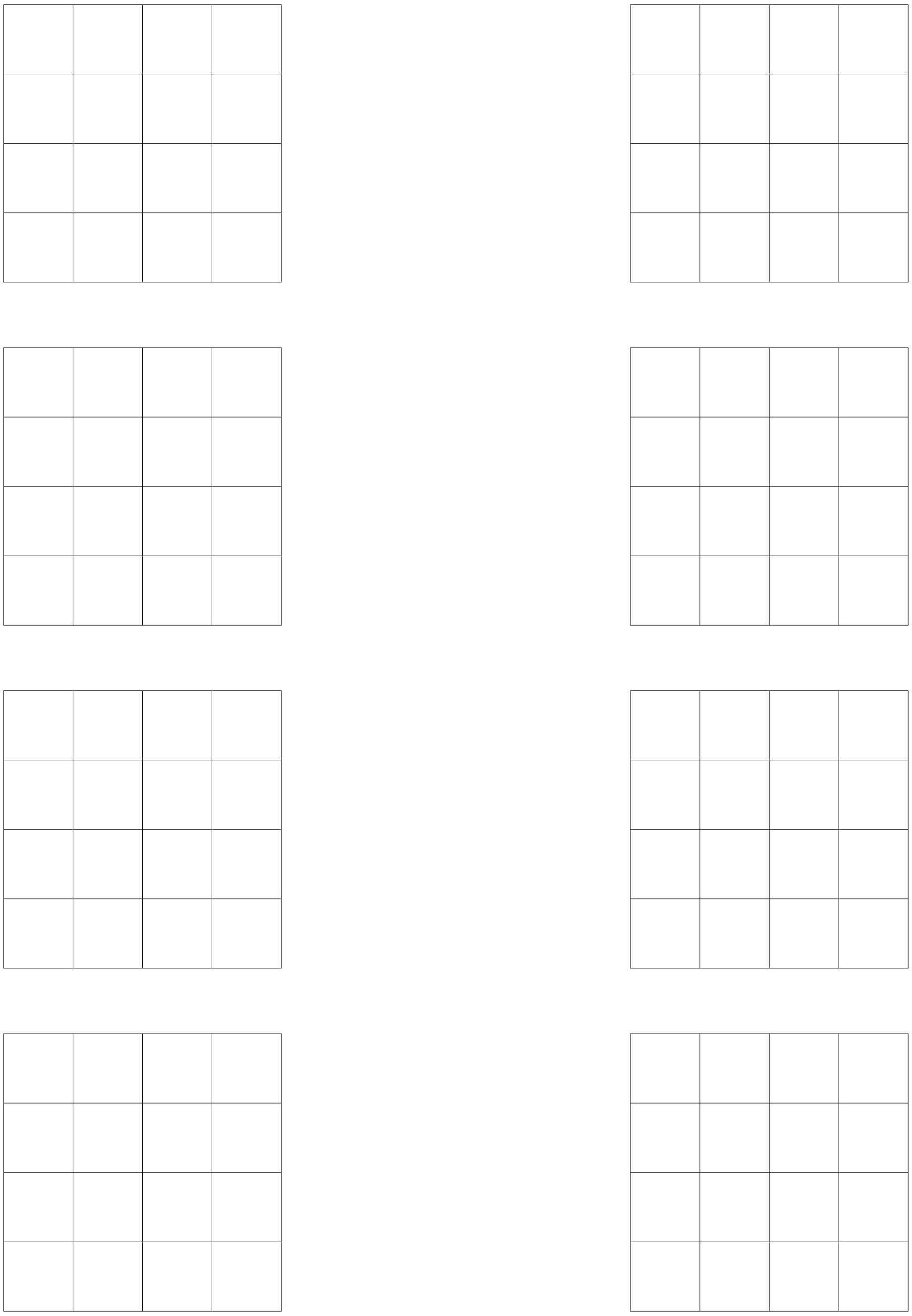


Heuristics: {
- most <u>constrained</u> variable first (reduce branching factor)
- most <u>promising</u> value first (find quickly first solution)

Strategies for { variable ordering / value ordering } could be { static / dynamic }

Search (V) ***Forward checking***

Why not filter right away effects of an action?



CSP: a decision problem (NP-complete)

1- Modeling:

— abstraction and reformulation

2- Preprocessing techniques:

— eliminate non-acceptable tuples prior to search

3- Search:

— potentially d^n paths of fixed lengths

— chronological backtracking

— intelligent backtracking

— variable/value ordering heuristics

— forward checking

4- Search 'hybrids':

— Mixing preprocessing with forward checking (MAC)

Domain types

$$\textbf{Given } \mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C}) \begin{cases} \mathcal{V} = \{V_1, V_2, \dots, V_n\} \\ \mathcal{D} = \{D_{V_1}, D_{V_2}, \dots, D_{V_n}\} \\ \mathcal{C} \text{ set of constraints} \end{cases}$$

$$C_{V_k,V_l,V_m} = \{\,(x, y, z)\} \subseteq D_{V_k} \times D_{V_l} \times D_{V_m}$$

Domains:

$\longrightarrow$ Finite (discrete), enumeration works

$\longrightarrow$ Continuous, sophisticated algebraic techniques are needed

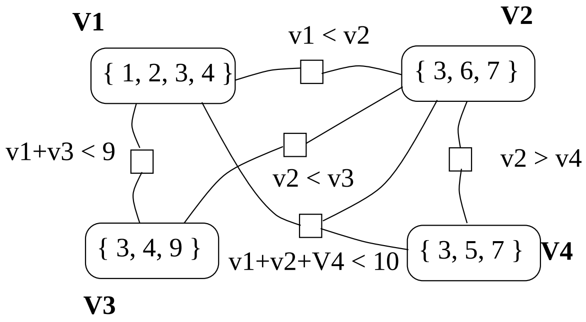
Constraint arity

Given $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ $\begin{cases} \mathcal{V} = \{V_1, V_2, \ldots, V_n\} \\ \mathcal{D} = \{D_{V_1}, D_{V_2}, \ldots, D_{V_n}\} \\ \mathcal{C} \;\text{ set of constraints} \end{cases}$

$$C_{V_k, V_l, V_m} = \{\, (x, y, z)\} \subseteq D_{V_k} \times D_{V_l} \times D_{V_m}$$

Constraints: universal, unary, binary, ternary, ... global

Constraint network



List of terms for glossary:

Due date: Mon, Jan 31, 2000

- Arc consistency
- Binary constraint
- Chronological backtracking
- Constraint (in a CSP)
- Constraint arity
- Constraint Satisfaction Problem (not AIMA!)
- Forward checking
- Global constraint
- Instantiated variable
- Intelligent backtracking
- Variable domain
- Variable/value ordering heuristic
- Ternary constraint
- Unary constraint
- Universal constraint