Title: Constraint Satisfaction
AIMA: Chapter 3 Section and Chapter 4, page 114
Web: Introduction to CSPs (Bartak's on-line guide)
Paper: Algorithms for Constraints Satisfaction problems: A Survey"
by Vipin Kumar. AI Magazine, Vol 13, No 1, 32-44, 1992.

Introduction to Artificial Intelligence

CSCE 476-876, Spring 2001

URL: `www.cse.unl.edu/~choueiry/S01-476-876`

Berthe Y. Choueiry (Shu-we-ri)

`choueiry@cse.unl.edu, (402)472-5444`

List of terms for glossary:

Due date: Mon, March 5, 2001

- Anytime algorithm
- 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

Constraint Processing

- Constraint Satisfaction: problem(s) and algorithms
- Constraint Languages: $\left\{ \begin{array}{l} \text{functional} \\ \text{logic} \\ \text{object-oriented} \end{array} \right.$

Constraint Satisfaction Problems (CSP)

- Course: Foundations of Constraint Satisfaction
- CSCE421/821, Fall 2001
- Fall'99 URL: `cse.unl.edu/~choueiry/CSE990-05/`
- Fall'00 URL: `cse.unl.edu/~choueiry/F00-CSCE990/`

Outline

- Definition
- Practical usefulness
- Underlying 'science'
- Commercial successes & job market

What is this about?

Context: You are a senior in college

Problem: You need to register in 4 courses for Spring'2000

Possibilities: Many courses offered in Math, CSE, EE, *etc.*

Constraints: restrict the choices you can make

- *Unary:* Courses have prerequisites you have/don't have
  Courses/instructors you like/dislike
- *Binary:* Courses are scheduled at the same time
- *n-ary:* In CompEng, 4 courses from 5 tracks such as at least 3 tracks are covered

You have choices, but are restricted by constraints

$\longrightarrow$ Make the right decisions

Practical Applications

Adapted from E.C. Freuder

- Radio resource management (RRM)
- Databases (computing joins, view updates)
- Temporal and spatial reasoning
- Planning, scheduling, resource allocation
- Design and configuration
- Graphics, visualization, interfaces
- Hardware verification and software eng.
- HCInteraction and decision support
- Molecular biology
- Robotics, machine vision and computational linguistics
- Transportation
- Qualitative and diagnostic reasoning

Constraint Satisfaction

Given

- A set of variables *4 courses at UNL*
- For each variable, a set of choices (values)
- A set of constraints that restrict the combinations of values the variables can take at the same time

Questions

- Does a solution exist? *classical decision problem*
- How two or more solutions differ? How to change specific choices without perturbing the solution?
- If there is no solution, what are the sources of conflicts? Which constraints should be retracted?
- *etc.*

Constraint Processing is about

- solving a decision problem
- while allowing the user to state arbitrary constraints in an expressive way and
- providing concise and high-level feedback about alternatives and conflicts

Power of Constraints Processing

- flexibility & expressiveness of representations
- interactivity, users can $\left\{ \begin{array}{c} \text{relax} \\ \text{reinforce} \end{array} \right\}$ constraints

Related areas: AI, OR, Algorithmic, DB, Prog. Languages, *etc.*

CSP: Definition

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

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

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

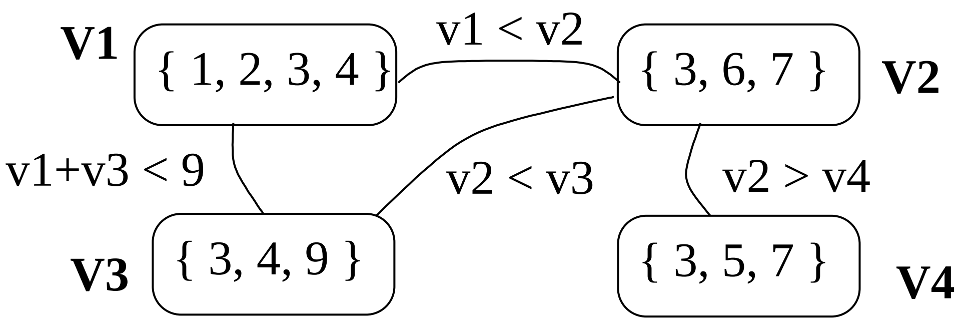
In general, **NP-complete**

Representation

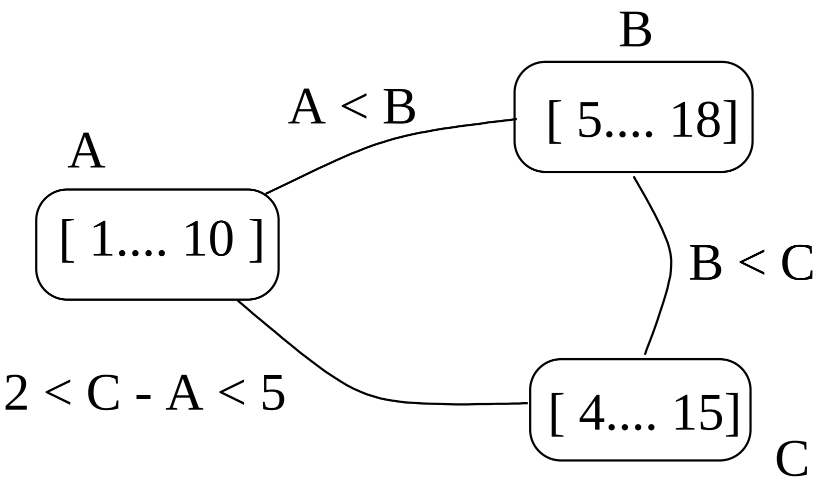
Given $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$
$$\begin{cases} \mathcal{V} = \{V_1, V_2, \ldots, V_n\} \\ \mathcal{D} = \{D_{V_1}, D_{V_2}, \ldots, D_{V_n}\} \\ \mathcal{C} \ \text{set of constraints} \end{cases}$$

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

Constraint **graph**

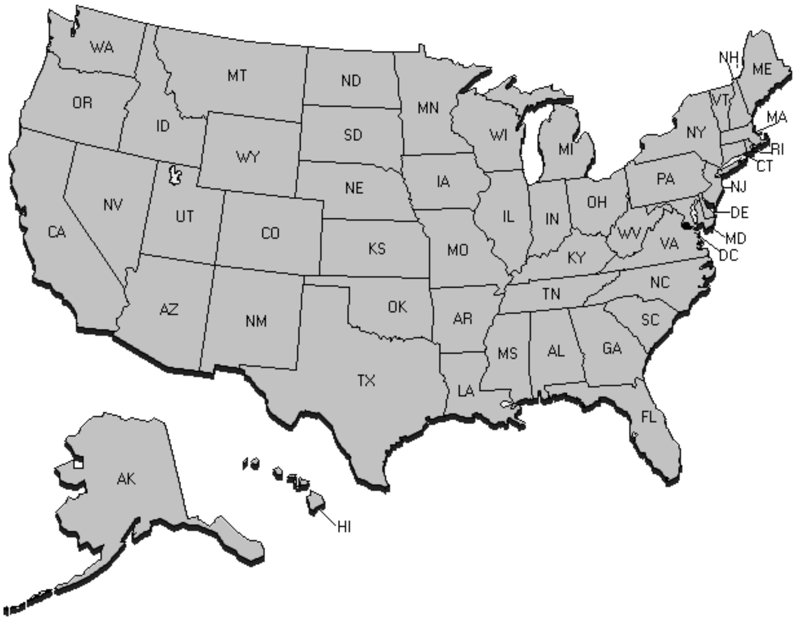


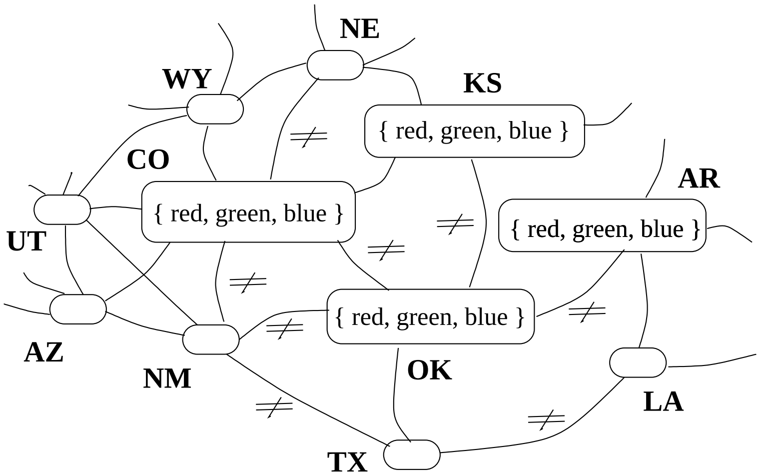
Example I: Temporal reasoning



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





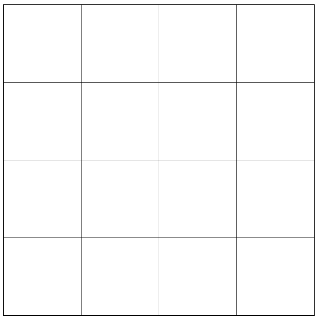
Variables? Domains? Constraints?

Example III: n-queens problem

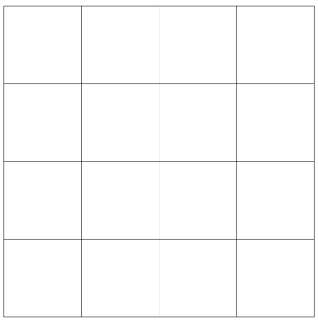
Variables?

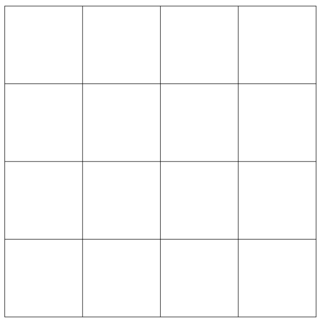
Domains?

Constraints?



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





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?

Search

1. Constructive, systematic
2. Iterative repair

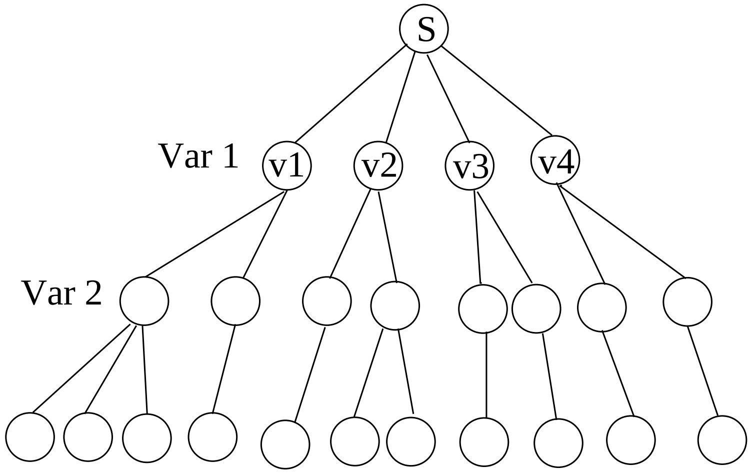
Systematic search

→ 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*

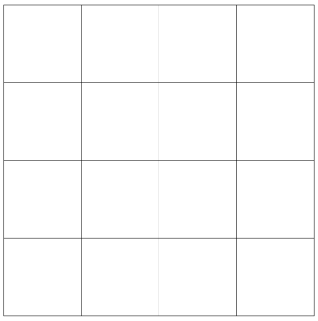
Before looking at search..

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

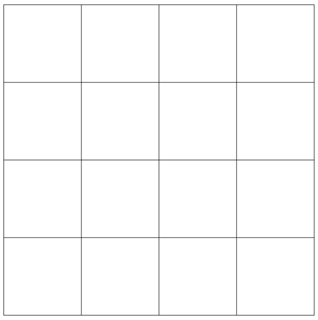


Variables?

Domains?

Size of CSP?

N-**queens**: formulation 2



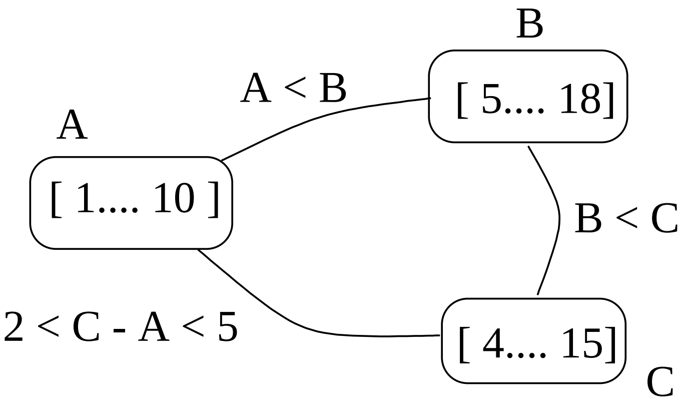
variables?

domains?

size of csp?

Constraint checking

→ arc-consistency



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

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

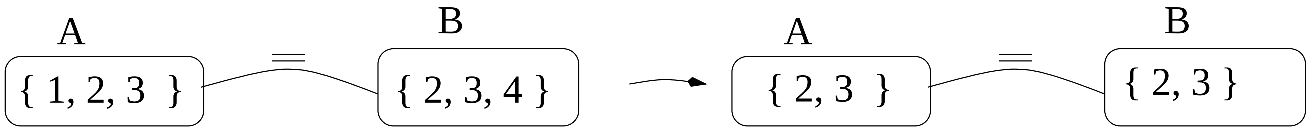
3- B: [5 .. 13]

⟶ 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

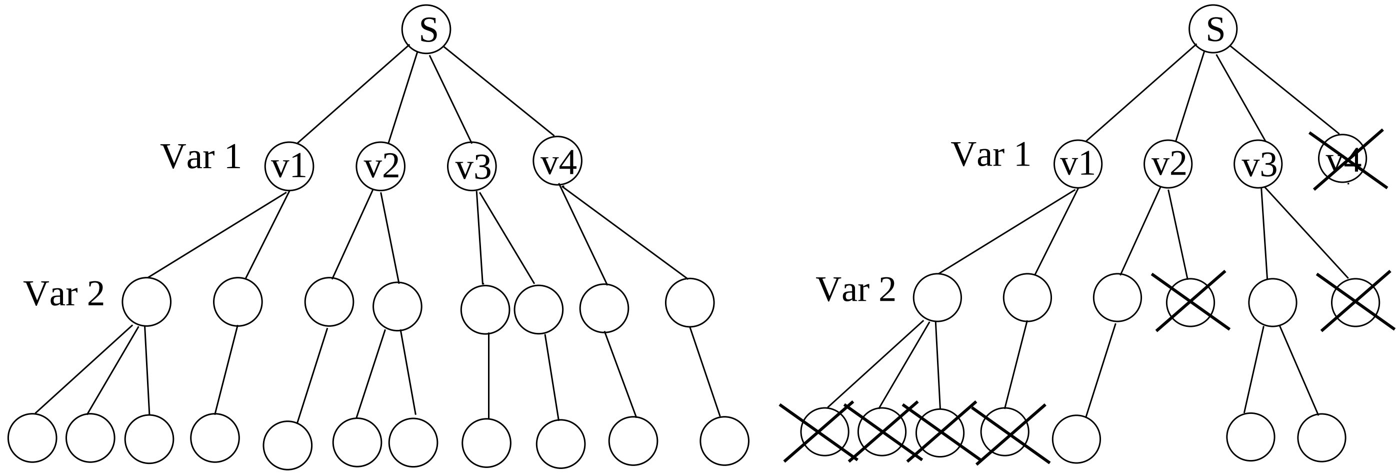


(A=2) ∧ (B=3) still isn't a solution!

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

Systematic search generates d^n possibilities

Are all possibilities acceptable?

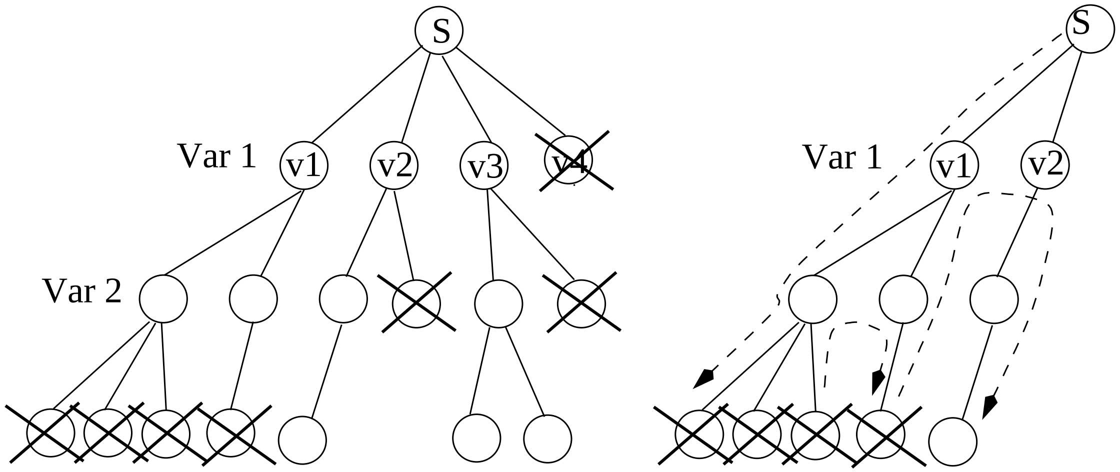


$\rightarrow$ Expand a partial solution only when consistent

$\longrightarrow$ **early pruning**

Search (II) *Chronological backtracking*

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



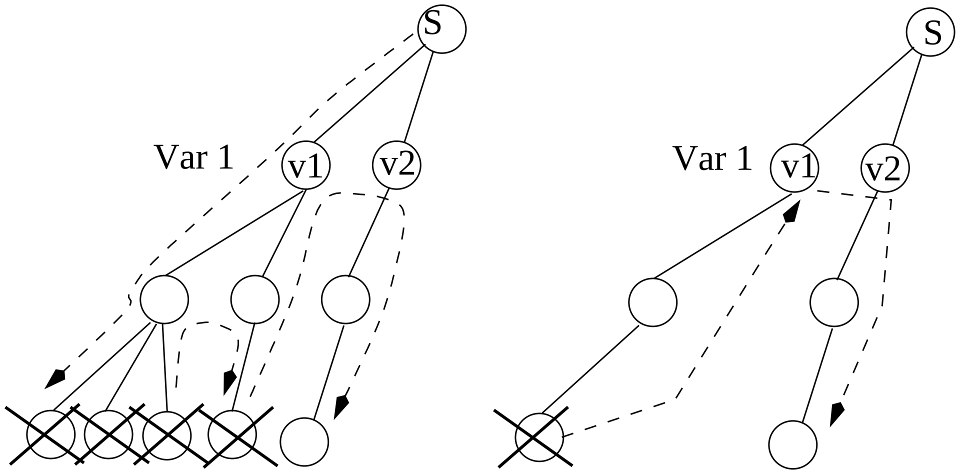
⟶ **Depth-first search & chronological backtracking**

→ DFS: soundness? completeness?

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

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

Backtrack to source of conflict!!



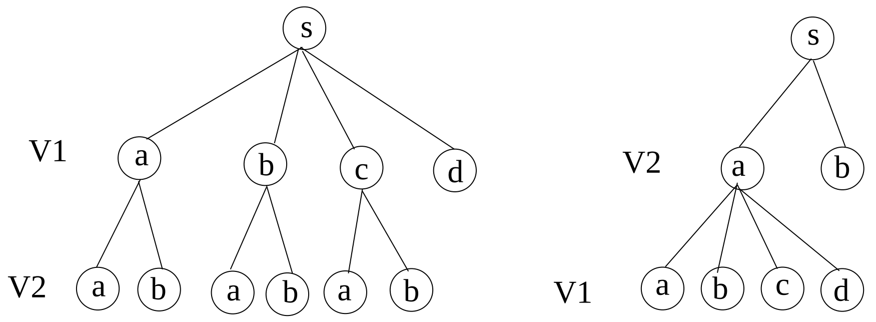
→ Backjumping, conflict-directed backjumping, etc.

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 $\{(V_1 = c), (V_2 = b)\}$

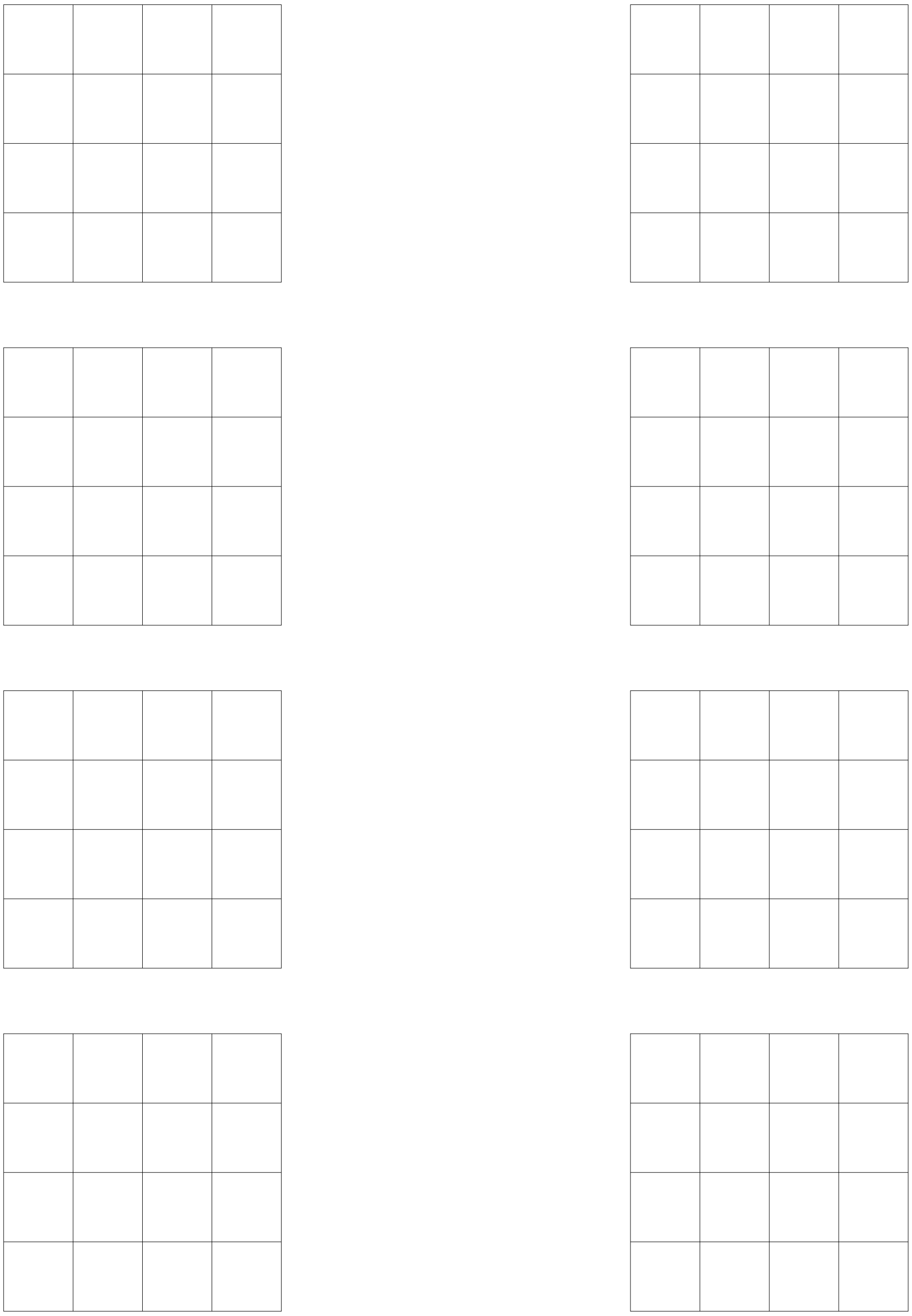


Heuristics:
- most <u>constrained</u> variable first (reduce branching factor)
- most <u>promising</u> value first (find quickly first solution)

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)

Non-systematic search

Iterative repair, local search

modifies a global but **<u>inconsistent</u>** solution

to decrease the number of violated constraints

Examples: Hill climbing, taboo search, simulated annealing,

GSAT, WalkSAT, Genetic Algorithms, etc.

$\longrightarrow$ Incomplete & not sound

Advantage: anytime algorithm

Shortcoming: Cannot tell that no solution exist

Domain types

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

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

Domains:

$\longrightarrow$ Finite (discrete), enumeration works

$\longrightarrow$ Continuous, sophisticated algebraic techniques are needed

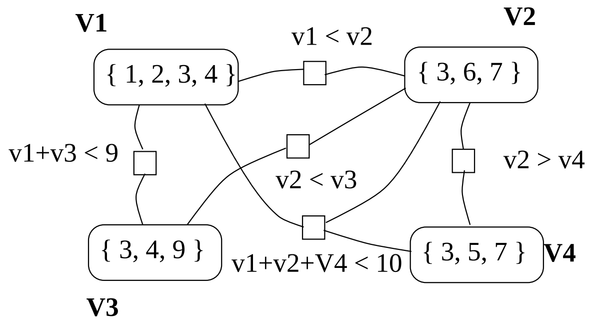
Constraint arity

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

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

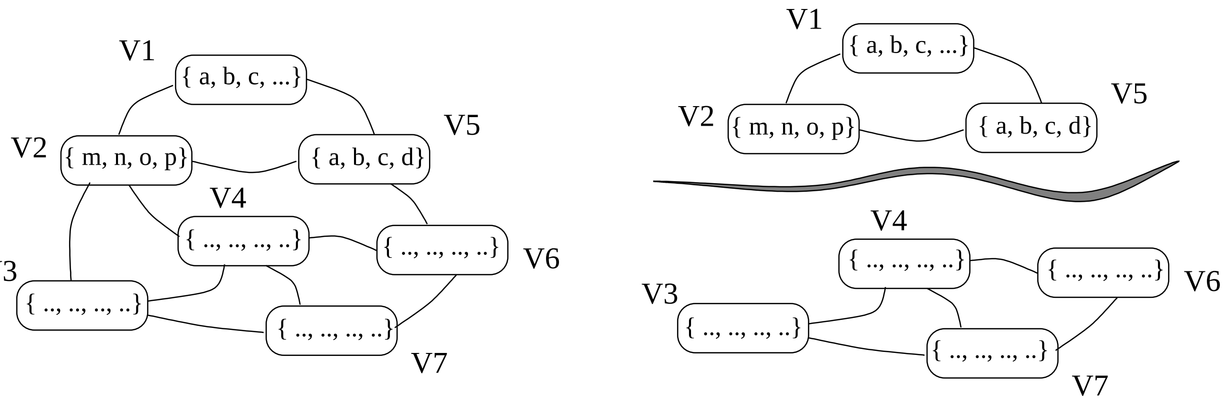
Constraints: universal, unary, binary, ternary, ... global

Constraint network

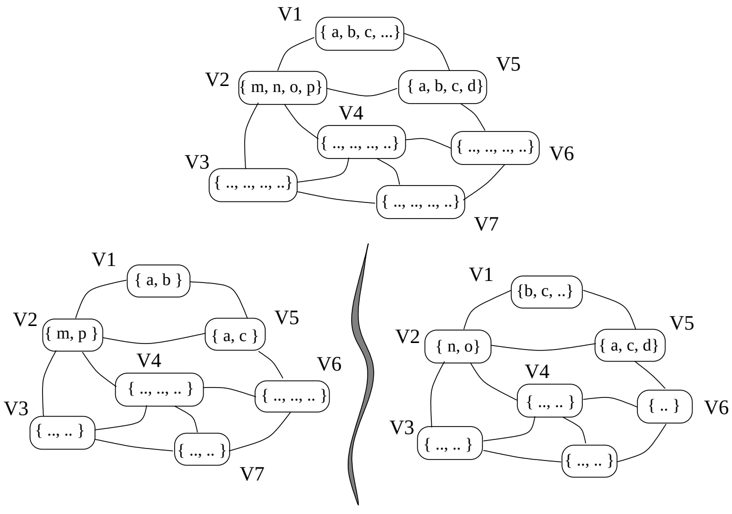


Decomposition $\begin{cases} \text{conjunctive} \\ \text{disjunctive} \end{cases}$

Conjunctive



Disjunctive



Deep analysis

Uncover particular properties, *e.g.*

- bounds the required level of consistency
- predicts ease/difficulty of solving a given instance

- **C-graph topology**
  tree, DAGs, chordal, etc.
- **Constraints type**
  subsets of Allen's relations,
  all-diffs, functional, monotonic, row-convex, etc.
- **Order parameter** (phase transition)

Multi agents

- **Problem:**

  decomposed, replicated

- **Agent:**

  broker, solver, problem, solver+problem

- **Architecture and authority:**

  hierarchical, egalitarian, priority-based (e.g., vote)

- **Communication:**

  agent-to-agent, group, broadcast

- **Interaction strategies:**

  cooperating vs. competing

  transparent vs. secretive

  negotiation + alliance formation

Research directions

Preceding (*i.e.*, search, backtrack, iterative repair, V/V/ordering, consistency checking, decomposition, symmetries & interchangeability, deep analysis) + ...

Evaluation of algorithms:

worst-case analysis vs. empirical studies
random problems?

Cross-fertilization:

SAT, DB, mathematical programming,
interval mathematics, planning, etc.

Modeling & Reformulation

Multi agents:

Distribution and negotiation
→ decomposition & alliance formation

CSP in a nutshell (I)

Solution technique: Search $\begin{cases} \text{constructive} \\ \text{iterative repair} \end{cases}$

Enhancing search: $\begin{cases} \text{intelligent backtrack} \\ \text{variable/value ordering} \\ \text{consistency checking} \\ \text{hybrid search} \\ \heartsuit \ \text{symmetries} \\ \heartsuit \ \text{decomposition} \end{cases}$

CSP in a nutshell (II)

Deep analysis: exploit problem structure
- ♡ graph topology
- ♡ constraint semantics
- phase transition

Research:
- k-ary constraints, soft constraints
- continuous vs. finite domains
- evaluation of algorithms (empirical)
- cross-fertilization (mathematical program.)
- ♡ reformulation and approximation
- ♡ architectures (multi-agent, negotiation)

Constraint Logic Programming (CLP)

A merger of

$\surd$ Constraint solving

$\longrightarrow$ Logic Programming, mostly Horn clauses (*e.g.*, Prolog)

Building blocks

- Constraint: primitives *but also user-defined*
  - cumulative/capacity (linear ineq), MUTEX, cycle, *etc.*
  - domain: Booleans, natural/rational/real numbers, finite
- Rules (declarative): a statement is a conjunction of constraints and is tested for satisfiability before execution proceeds further
- Mechanisms: satisfiability, entailment, delaying constraints

Your future: jobs

Commercial companies: I2 Technologies, Trilogy, Ilog, PeopleSoft/Red Pepper, Carmen Systems (Sweden), J.D. Edwards/Numetrix, <u>COSYTEC</u>, *etc.*

Prestigious research centers: NASA Ames, JPL, BT Labs (UK), IBM Watson+Alamaden, *etc.*

Start your own: Blue Pumpkin, <u>Parc Technologies Ltd</u>, Calico, Firepond, *etc.*

Academic: constraint languages, modeling, representation, automated reasoning, *etc.*

Constraint Processing Techniques are the basis of new languages:

> *Were you to ask me which programming paradigm is likely to gain most in commercial significance over the next 5 years I'd have to pick Constraint Logic Programming (CLP), even though it's perhaps currently one of the least known and understood. That's because CLP has the power to tackle those difficult combinatorial problems encountered for instance in job scheduling, timetabling, and routing which stretch conventional programming techniques beyond their breaking point.*
> *<u>Though CLP is still the subject of intensive research, it's already being used by large corporations</u> such as manufacturers Michelin and Dassault, the French railway authority SNCF, airlines Swissair, SAS and Cathay Pacific, and Hong Kong International Terminals, the world's largest privately-owned container terminal.*

Byte, Dick Pountain