Title: CSP 101 (cont.)

Material: Constraint Satisfaction, Dechter MIT Encyclopedia
CSP, An Overview, Pedro Meseguer

Foundations of Constraint Satisfaction
CSCE421/821, Fall 2002
`www.cse.unl.edu/~choueiry/F02-421-821/`

Berthe Y. Choueiry (Shu-we-ri)
Ferguson Hall, Room 104
`choueiry@cse.unl.edu`, Tel: (402)472-5444

Domain types

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

$$C_{V_a, V_b, \ldots, V_i} = \{(x, y, \ldots, z)\} \subseteq D_{V_a} \times D_{V_b} \times \ldots \times D_{V_i}$$

Domains:

$\longrightarrow$ Finite (discrete), enumeration works

$\longrightarrow$ Continuous, sophisticated algebraic techniques are needed

Terminology

Given $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C}) \ldots$

Query:

- Find a solution
- Find all solutions
- Find number of solutions
- Can you extend a partial solution, *etc.*

Constraints:

- *arity:* universal, unary, binary, ternary, ... global
- *scope:* set of variables to which the constraint applies
- *definition:* intension, extension
- *implementation:* predicate, set of tuples, binary matrix, etc.

Representation:

Macrostructure $\mathcal{G}(\mathcal{P})$:

- constraint graph for binary constraints
- constraint network for non-binary constraints

Micro-structure $\mu(\mathcal{P})$:

Co-microstructure co-$\mu(\mathcal{P})$:

Complexity of CSP

Characterization:
Decision problem. In general, NP-complete by reduction from 3SAT

Steps for proving NP-completeness:
Given a problem Π in **NP**, show that some already know **NP-complete** problem Π' can be transformed to Π.

1. Show that Π is in NP
2. Select a known **NP-complete** problem Π'
3. Construct a transformation f from Π' to Π', and
4. Prove that f is a polynomial transformation

(check Chapter 3 of Garey & Johnson)

What is SAT?

SAT?

Given a sentence:

- *Sentence:* conjunction of clauses
  $(c_1 \vee \neg c_4 \vee c_5 \vee c_6) \wedge (c_2 \vee \neg c_3) \wedge (\neg c_4)$
- *Clause:* disjunction of literals
  $(c_2 \vee \neg c_3)$
- *Literal:* a term or its negation
  c_1, $\neg c_6$
- *Term:* boolean variable
  $c_1 = 1 \Leftrightarrow \neg c_1 = 0$

Question: Find an assignment of truth values to Boolean variables such the sentence is satisfied.

Problem reduction

Example: CSP into SAT *(proves nothing, just an exercise)*

Notation: variable-value pair = VVP

- VVP $\rightarrow$ term.
  E.g., $V_1 = \{a, b, c, d\}$ and $V_2 = \{a, b, c\}$ becomes
  $x_1 = (V_1, a)$, $x_2 = (V_1, b)$, $x_3 = (V_1, c)$, $x_4 = (V_1, d)$, and
  $x_5 = (V_2, a)$, $x_6 = (V_2, b)$, $x_7 = (V_2, c)$
- VVPs of a variable $\rightarrow$ disjunction of terms.
  E.g., $x_1 \vee x_2 \vee x_3 \vee x_4$
- (optional) At most one VVP per variable.
  $(\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3) \wedge$
  $(\neg x_2 \vee \neg x_4) \wedge \ldots$ for every combinations of 2 values in the domain of a variable

Example: CSP into SAT (cont')

- Constraint: $C_{V_1,V_2} = \{(a, a), (a, b), (b, c), (c, b), (d, a)\}$
  Way 1: One clause per inconsistent tuple
  E.g., $\neg x_1 \vee \neg x_7$
  How many?

  Way 2: tuple $\rightarrow$ conjunction of terms.
  E.g., $x_1 \wedge x_5$
  constraint $\rightarrow$ disjunction of these conjunctions
  E.g., $(x_1 \wedge x_5) \vee (x_1 \wedge x_6) \vee (x_2 \wedge x_7) \vee (x_3 \wedge x_6) \vee (x_4 \wedge x_5)$
  $\longrightarrow$ transform into conjunctive normal form (CNF)
- Question: find a truth assignment of the boolean variables such as the sentence is satisfied.

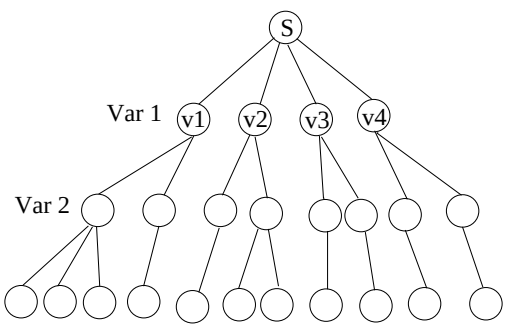
How to solve a CSP?

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 checking/filtering, constraint
   propagation to reduce size of search space

Importance of modeling

N-queens: formulation 1

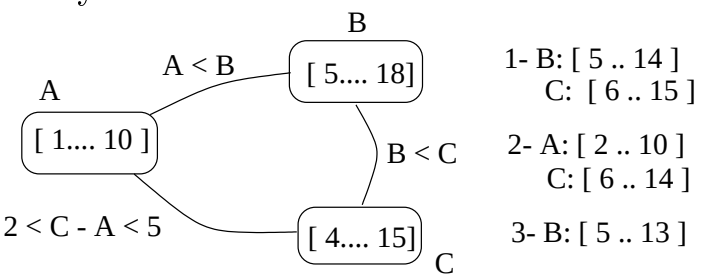
Variables?
Domains?
Size of CSP?

N-queens: formulation 2

variables?
domains?
size of csp?

Constraint checking

→ arc-consistency



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

⟶ Constraint filtering, constraint checking, etc..
eliminate non-acceptable tuples prior to search

→ Warning: arc-consistency does not solve the problem

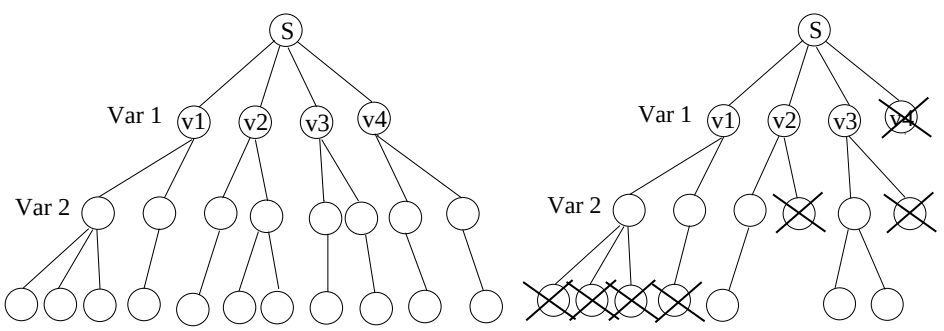
(A=2) ∧ (B=3) still isn't a solution!

Systematic search (I)

Back-checking

Systematic search generates d^n possibilities

Are all possibilities acceptable?

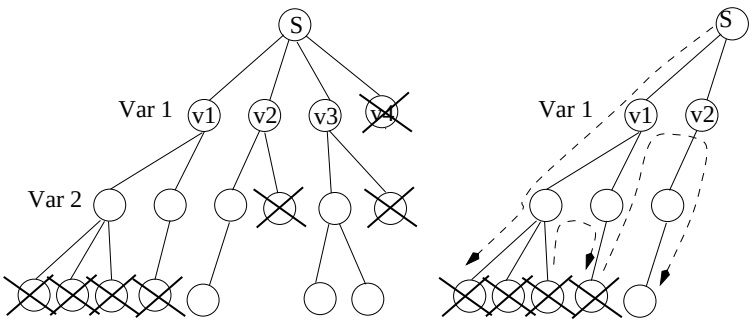


→ Expand a partial solution only when consistent

⟶ **early pruning**

Systematic search (II) *Chronological backtracking*

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



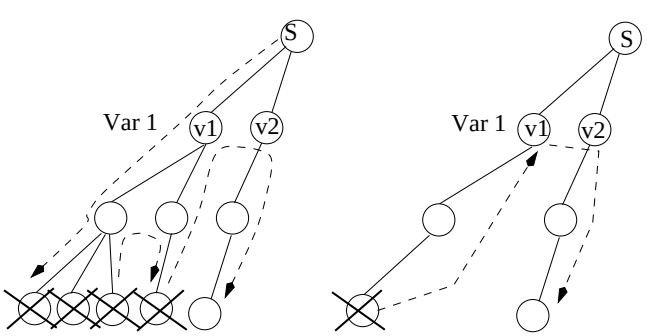
⟶ **Depth-first search & chronological backtracking**

→ DFS: soundness? completeness?

Systematic search (III) *Intelligent backtracking*

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

Backtrack to source of conflict!!



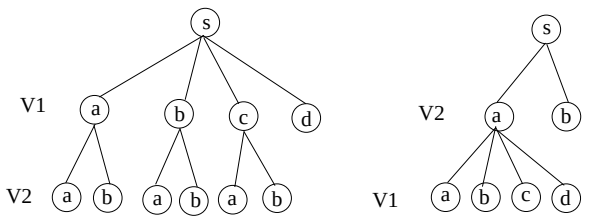
→ Backjumping, conflict-directed backjumping, etc.

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

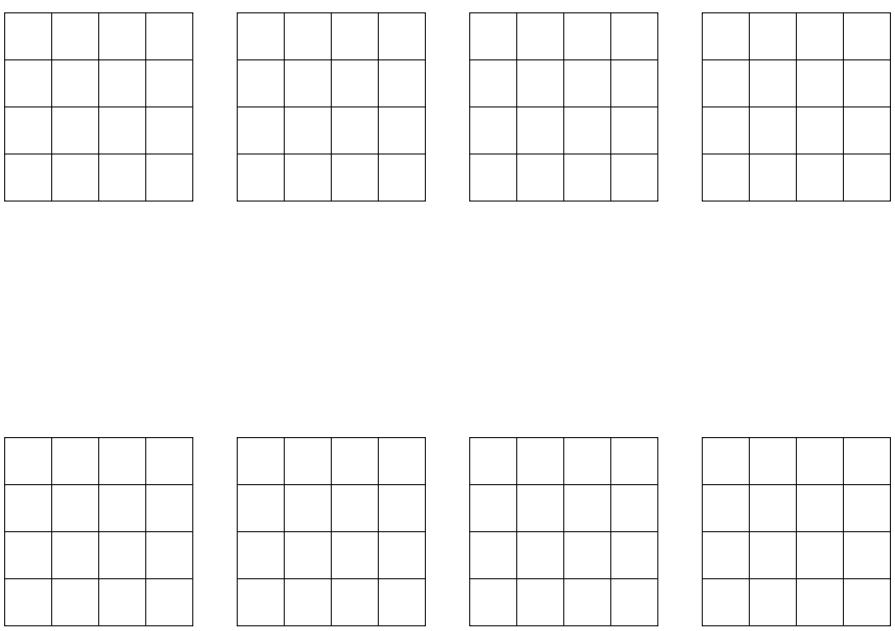


Heuristics: { most constrained variable first (reduce branching factor); most promising value first (find quickly first solution) }

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

Systematic search (V) *Forward checking*

Why not filter right away effects of an action?



Backtrack search

- Constructive, systematic, exhaustive
- In general sound and complete
- Back-checking
- Backtracking: chronological vs. intelligent
- Ordering heuristics:
  static, dynamic variable, dynamic variable-value pairs,
- Looking ahead:
  Partial (Forward checking, FC), directional arc-consistency,
  Full (a.k.a. Maintaining Arc-consistency or MAC),

CSP: a decision problem (NP-complete)

1- Modeling:
— abstraction and reformulation

2- Preprocessing techniques:
— eliminate non-acceptable tuples prior to search

3- Systematic search:
— potentially d^n paths of fixed lengths
— chronological backtracking
— intelligent backtracking
— variable/value ordering heuristics

4- Search 'hybrids':
— Mixing consistency checking with search: look-ahead strategies

Non-systematic search

Iterative repair, local search
modifies a global but **inconsistent** 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

Decomposition

- conjunctive
- disjunctive
- partition constraints (DB)

Conjunctive

Disjunctive

$\longrightarrow$ Properties of disjunctive decompositions

Properties of disjunctive decompositions
(ignore for now)

$$\mathcal{P} \longrightarrow \{ \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_i \}$$

Conjunctive vs. disjunctive

Consistent	No constraint is removed
Simplifying	$\text{Size}(\mathcal{P}_i) < \text{Size}(\mathcal{P})$
Semi-complete	At least one solution is kept
Complete	No solution is lost
Redundant	Solutions replicated in $\{\mathcal{P}_i\}$
Reducible	$\sum_i \text{Size}(\mathcal{P}_i)$ may be $< \text{Size}(\mathcal{P})$

Deep analysis

Uncover particular properties, *e.g.*
- bound the required level of consistency (islands of tractability)
- predict ease/difficulty of solving a given instance

- **C-graph structure, 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

1. **Computational tasks:**
   problem decomposed, replicated
2. **Types of Agent:**
   broker, solver, problem, solver+problem
3. **Architecture and authority:**
   hierarchical, egalitarian, priority-based (e.g., vote)
4. **Nature of Communication:**
   agent-to-agent, group, broadcast
5. **Interaction strategies:**
   cooperating vs. competing
   transparent vs. secretive
   negotiation + alliance formation

Computational tasks

At one end of the spectrum, agents may be involved in solving heterogeneous, distinct and completely independent problems and request other agents to supply specific functionalities for the completion of the individual tasks.

At the other end of the spectrum, the same problem could be replicated and assigned to all agents, which can then share their individual results with other agents incrementally in order to speed up the execution of the global computational task.

Types of agent

An agent in the system can be any of the following:

- an agent that collects data from the environment and formulate the CSP
- a reasoning module with specific computational characteristics (e.g., various search algorithms)
- brokers that facilitate matching between a service seeker and a number of service providers (e.g., CORBA brokers).

Agent architecture and how is authority granted

Agents could be organized in a strict hierarchy in which a given agent has full control over the activities of the agents that lie underneath it in the hierarchy. It decides how the lower level agents may cooperate while ensuring coordination with the higher-level agent.

Agents could be in a strictly flat structure competing for services and rewards, either chaotically or according to some strict priority policy, based on voting or time-responsiveness.

Nature of the communication environment

Communications among agents may be conducted according to:

- a one-to-one schema
- multi-cast (i.e., one-to-group), or
- broadcast, where all agents in the environment have access to the content of the communicated information.

Type of supported interactions

Agents may be cooperative, pooling their resources and capabilities to achieve a common, global objective, or they could be competitive trying to win rewards and optimize their individual gain.

They could also adopt a midway strategy, dynamically forming coalitions and gathering support to acquire more resources and realize greater gains.

Also, agents may be transparent about their intentions, resources, needs, and constraints or may be secretive, hiding one or the other of their strengths or weaknesses.

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.

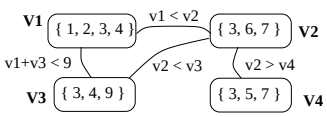
Modeling & Reformulation

Multi agents:

Distribution and negotiation
→ decomposition & alliance formation

CSP in a nutshell (I)

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



Constraint graph, constraint network
Finite domains
Binary constraints, universal constraints.

Examples: map coloring, puzzles, resource allocation, temporal reasoning, product configuration, databases, spreadsheets, graphical layouts, graphical user-interfaces, bioinformatics, etc.

CSP in a nutshell (II)

Solution technique: Search
- constructive
- iterative repair

Enhancing search:
- intelligent backtrack
- variable/value ordering
- consistency checking
- hybrid search
- ♡ symmetries
- ♡ decomposition

CSP in a nutshell (III)

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

√ Constraint solving

⟶ 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

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.*
> *Though CLP is still the subject of intensive research, it's already being used by large corporations 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*