Theory and Practice of Constraint Satisfaction

- Welcome
- Sign-up sheet
- Check `www.cse.unl.edu/~choueiry/CSE990-05/`

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

Admin

Time: M-W: 3:30–4.45 p.m.

Office hours: M-W: 10:30 a.m.–12:00 p.m.

Course:
- Lectures
- Presentations by students
- Summary of presentations by students

Rules:
- Absence: 2 sessions max.
- Collaboration: encouraged, except for..

Grading:
- Up to 80% from menu: Presentation 20%,
  Summary: 5%, Project: 30%, Paper: 30%.
- Programming assignments: 10%
- Participation/interaction: 10%

\+ Bonus for 100% attendance
\+ Bonus for 'riddles'

Content of the course

- Introduction 1 or 2 sessions
- Foundations and basic mechanisms : 12 sessions
- Decomposition: 5 sessions
- Abstraction & reformulation: 6 sessions
- Extensions: 2 sessions
- Alternative approaches: 1 session

Required reading + additional advised reading

Course support

- Tsang
- Papers from: WWW, course webpage, library,
  electronic reserves, me etc.

Pointers to resources

Textbook

Tsang'93 (out of print)

Web resources

Archives at Constraint Computation Center, UNH

Conferences

CP (+WSs), IJCAI, ECAI, NCAI (AAAI), FLAIRS, etc.

Journals

Constraints, AIJ, JACM, AI+Math, etc.

Jobs

- Commercial companies: i2 Technologies, Trilogy, Ilog, PeopleSoft, Carmen Systems (Sweden), etc.
- Prestigious RCs: NASA Ames, JPL, BT Labs (UK), etc.
- Start your own: MS not yet aware..

Constraint Satisfaction 101

- Definition & characterization
- Applications
- Solution techniques
- Issues & research directions

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**

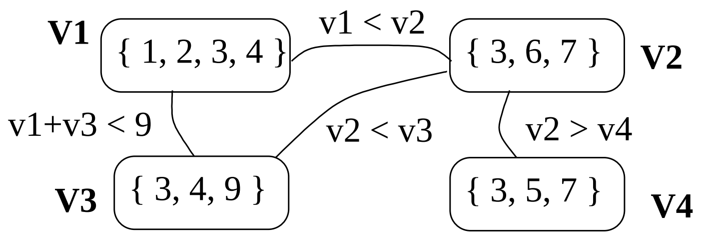
Riddle 1: Express $\mathcal{D}$ in terms of $\mathcal{C}$
Reply by email.
Winner: first correct answer

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**, network



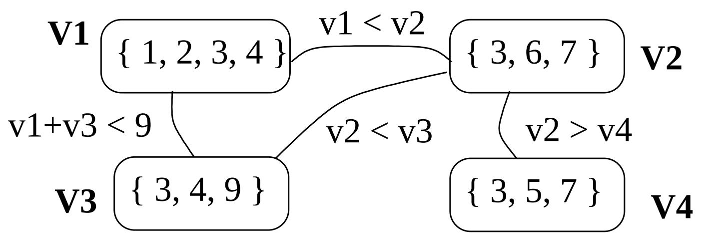
Domains

Finite (discrete), infinite (discrete/continuous)

Constraints

Binary (graph), k-ary (hyper-graph)

Constraint **graph**, network



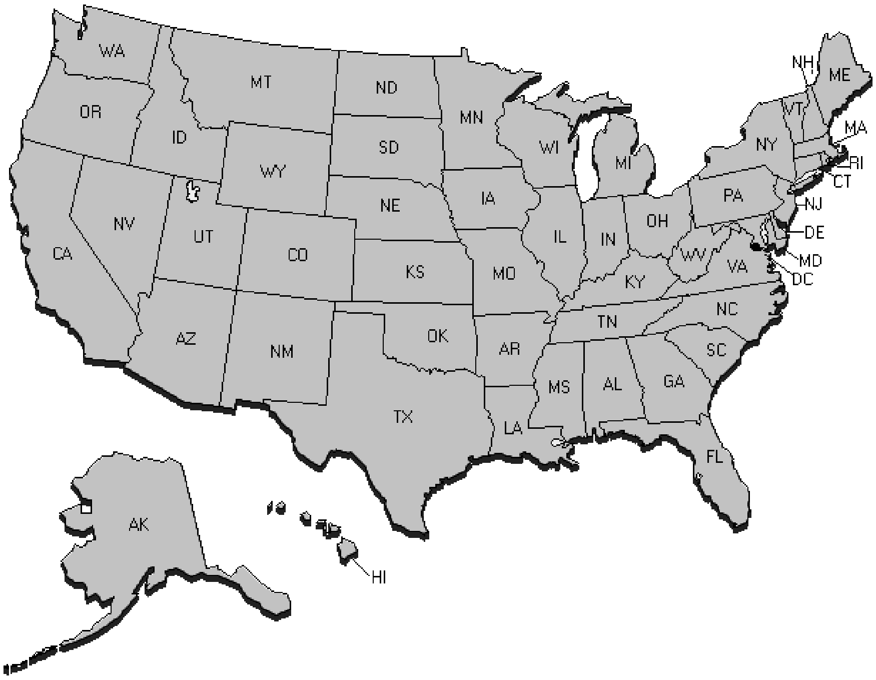
Riddle 2: Using two types of nodes, propose an alternative representation of a constraint network that allows the representation of non-binary constraints using edges instead of hyper-edges.

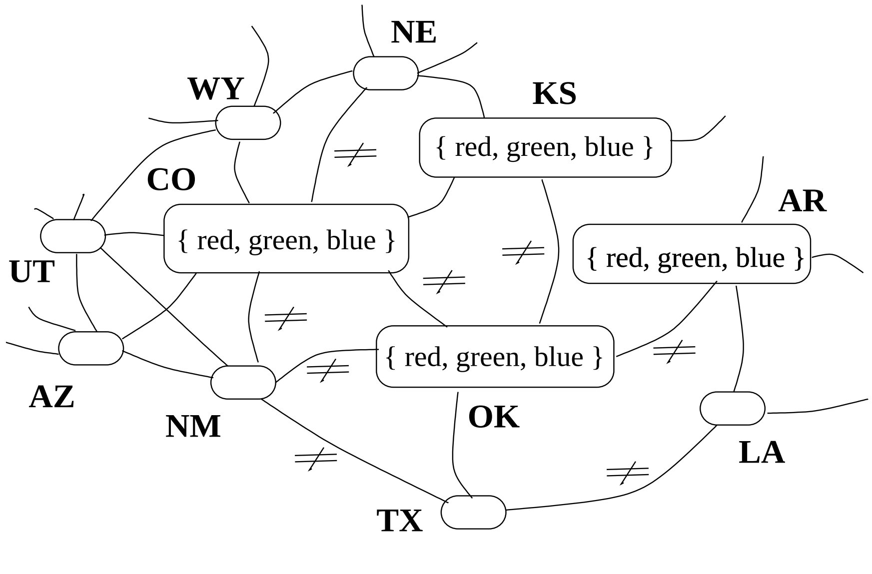
Use it to represent the CSP above with this *additional* constraint:

$$V_1 + V_2 + V_4 < 10$$

I. Map coloring

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





II. Puzzle

Four musicians, Fred, Ike, Mike, and Sal play bass, drums, guitar and keyboard, not necessarily in that order. They have 4 successful songs, "Blue Sky", "Happy Song", "Gentle Rhythm" and "Nice Melody." Ike and Mike are, in one order or the other, the composer of "Nice Melody" and the keyboardist, etc..

Formulation 1

Variables: Bass, Drums, Guitar, Keyboard, Blue Sky,
Happy Song, Gentle Rhythm and Nice Melody.

Domains: { Fred, Ike, Mike, Sal }

Constraints: ...

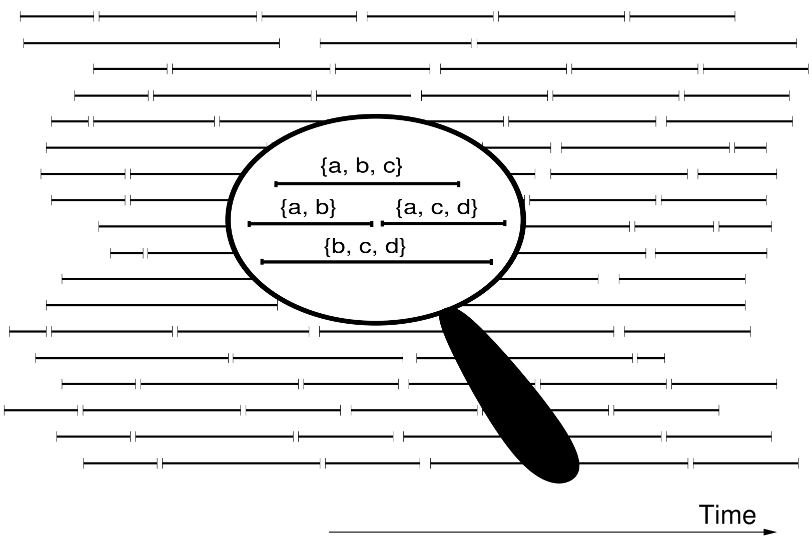
Formulation 2

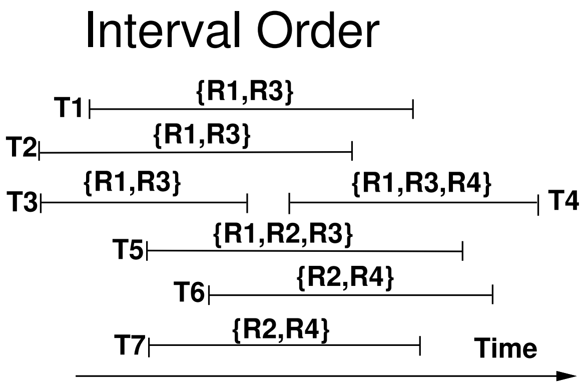
Variables: Fred's-instrument, Ike's-instrument, ..
Domains: { bass, drums, guitar, keyboard }

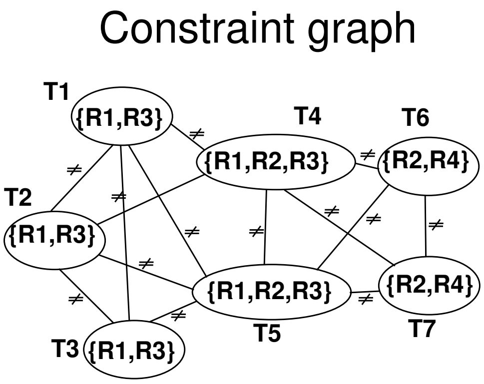
Variables: Fred's-song, Ikes's-song, ..
Domains: { Blue Sky, Happy Song, Gentle Rhythm, Nice Melody}

Constraints: ...

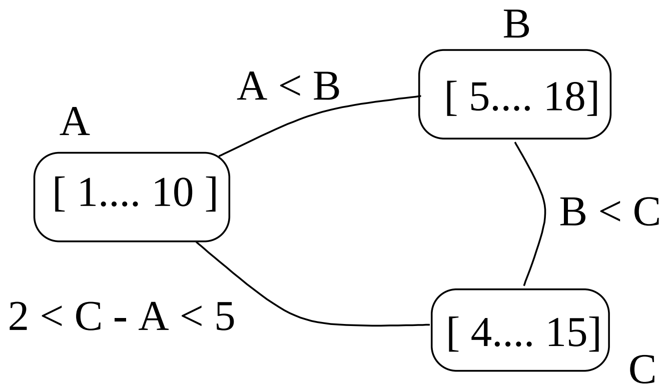
III. Resource Allocation







IV. Temporal reasoning



V. Product configuration

train, elevator, car, etc.

Given: components and their attributes (variables)
domain covered by each characteristic (values)
relations among the components (constraints)
a set of required functionalities (more constraints)

Find: a product configuration
i.e., an acceptable combination of components
that realizes the required functionalities

VI. Databases

- Join operation in relational DB is a CSP
- View update is a CSP

VII. Interactive systems

Dataflow constraints

- Spreadsheets
- Graphical layout systems and animation
- Graphical user interfaces

VIII. Molecular biology (bioinformatics)

Threading

IX. etc.

Advantages:

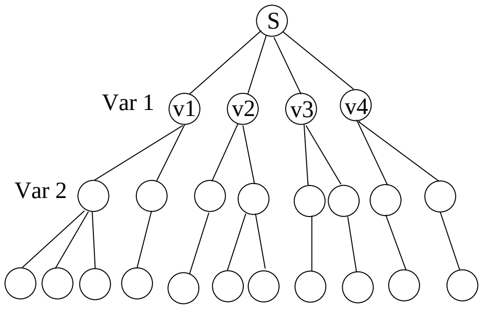
Flexible, expressive, compact representation

Related areas:

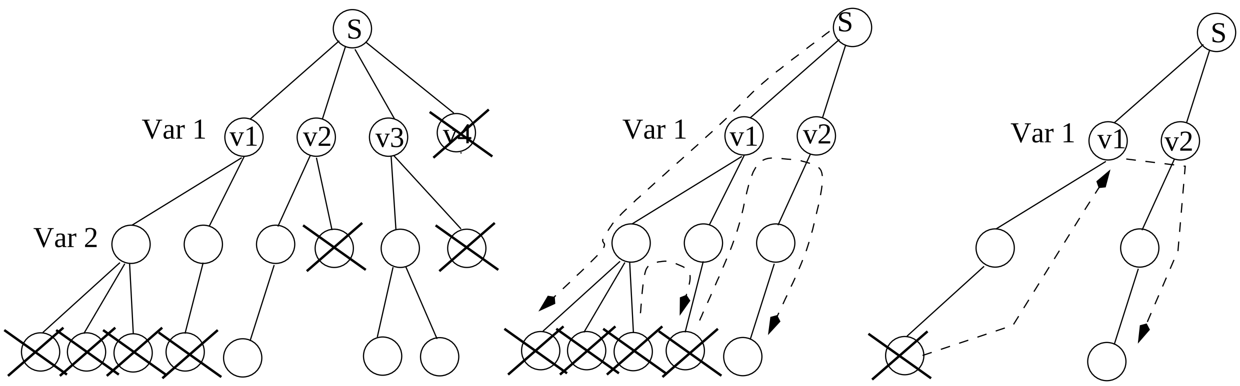
AI, OR, Algorithmics, DB, Prog. Languages

Solution technique: Search $\begin{cases} \text{Constructive} \\ \text{Iterative repair} \end{cases}$

Constructive, generate d^n possibilities



- Expand a partial solution only when consistent
- Depth first: chronological, intelligent backtracking



$\longrightarrow$ Complete & sound

Solution technique (II)

Iterative repair (local search)
modifies a global but **inconsistent** solution to decrease the number of violated constraints

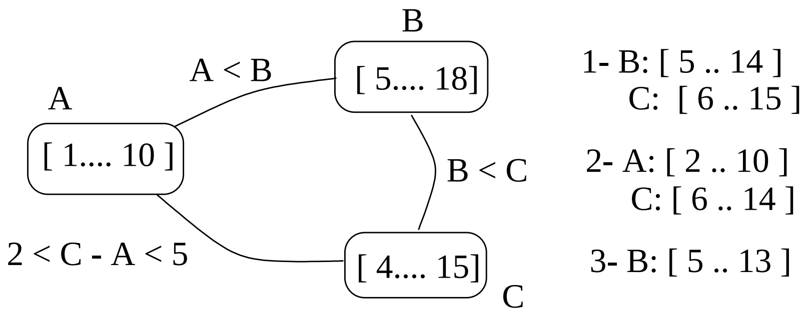
Examples: Simulated annealing, hill climbing, taboo, GSAT, Genetic Algorithms, etc.

$\longrightarrow$ Incomplete & not sound

Enhancements

- Variable/value ordering (static/dynamic)
  to control branching factor

- Consistency checking: arc-consistency



  AC-1, AC-2, ..., AC-7,
  3-consistency, k-consistency ($k \leq n$)

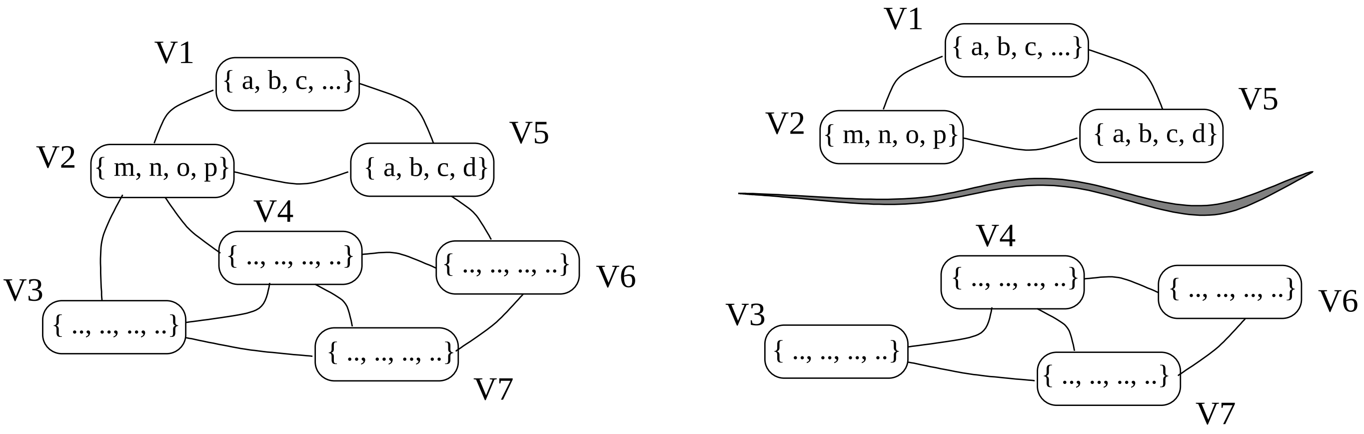
- Consistency checking + Search
  forward checking (FC), FCarc, FCpath, etc.

- Symmetries and interchangeability
  e.g., pigeon hole

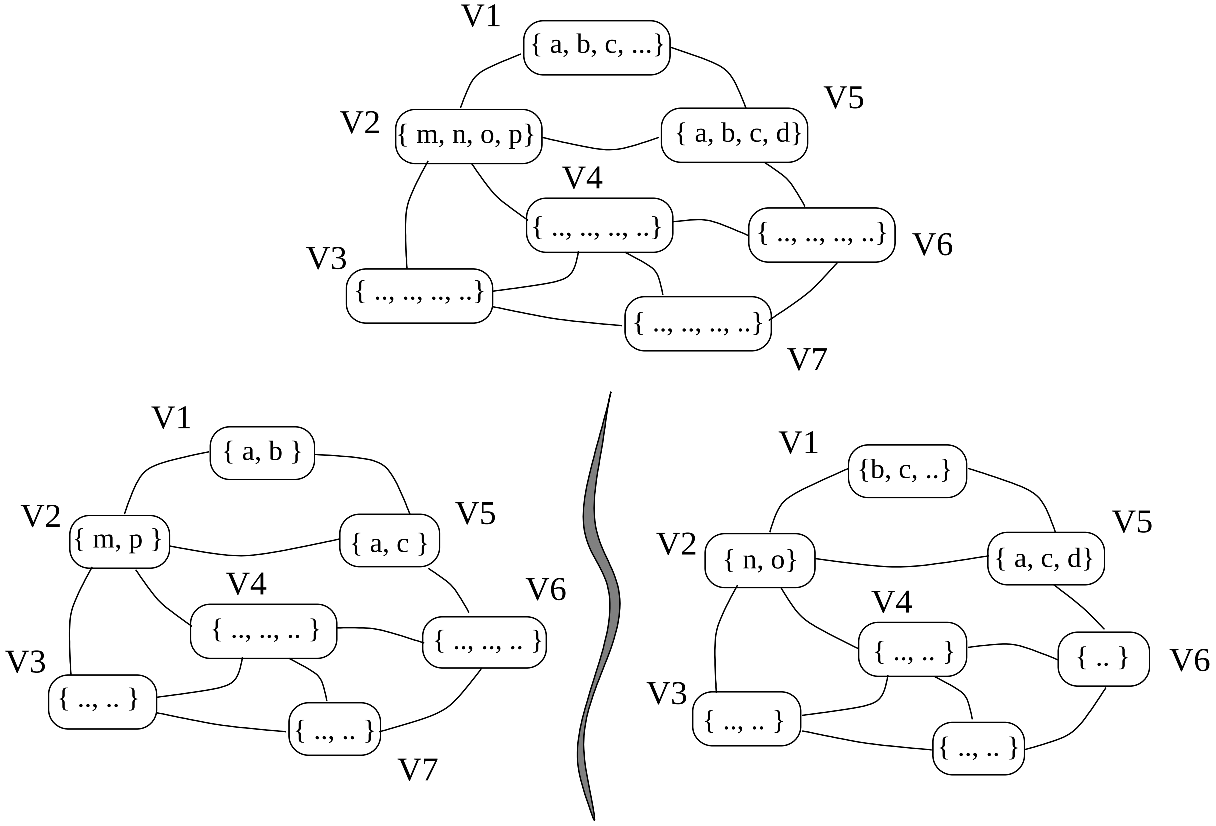
- Decomposition: disjunctive, conjunctive

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

Conjunctive



Disjunctive



Properties

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

Conjunctive vs. disjunctive

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

Deep analysis

Uncover particular properties, *e.g.*
- bounds the required level of consistency
- predicts ease/difficulty of solving a given instance

- **C-graph structure**
  tree, DAGs, chordal, etc.

- **Constraints type**
  subsets of Allen's relations,
  all-diffs, functional,
  monotonic, row-convex, etc.

- **Order parameter** (phase transition)

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, empirical studies
  random problems?

- **Cross-fertilization**:
  SAT, DB, math. programming,
  interval math, planning, etc.

- **Modeling & Reformulation**

- **Multi-agents**:
  Distribution and negotiation
  $\rightarrow$ decomposition & alliance formation

Multi-agent

- Problem:
  decomposed, replicated

- Agent:
  broker, solver, problem, solver+problem

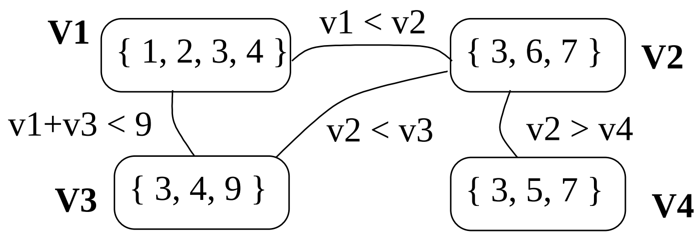
- Architecture and authority:
  hierarchical, egalitarian
  priority-based (e.g., vote)

- Communication:
  agent-to-agent, group, broadcast

- Interaction strategies:
  cooperative, competitive
  transparent, secretive

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.
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
- forward-checking
- symmetries (interchangeability)
- decomposition

Deep analysis:
- graph structure
- constraints semantics
- phase transition

Open:
- k-ary constraints
- evaluation of algorithms
- cross-fertilization (mathematical program.)
- reformulation and approximation
- architectures (multi-agent, negotiation)