Constraint Satisfaction

Moving to a different formalism…

```
   SEND
 + MORE
 ------
  MONEY
```

Consider state space for cryptarithmetic (e.g. DFS).

Is this (DFS) how humans tackle the problem?

Human problem solving appears more **sophisticated**! For example, we derive new constraints on the fly.
→ **little** or **no** search!

Constraint Satisfaction Problems (CSP)

A powerful representation for (discrete) search problems

A **Constraint Satisfaction Problem (CSP)** is defined by:

X is a set of n variables X_1, X_2,…, X_n each defined by a finite domain D_1, D_2,…D_n of possible values.

C is a set of constraints C_1, C_2,…, C_m. Each C_i involves a subset of the variables; specifies the allowable combinations of values for that subset.

A solution is an assignment of values to the variables that satisfies all constraints.

Cryptarithmetic as a CSP

```
   TWO
 + TWO
 -----
  FOUR
```

Variables:

$$T \in \{0,\ldots,9\}; W \in \{0,\ldots,9\}; O \in \{0,\ldots,9\};$$

$$F \in \{0,\ldots,9\}; U \in \{0,\ldots,9\}; R \in \{0,\ldots,9\};$$

$$X_1 \in \{0,\ldots,1\}; X_2 \in \{0,\ldots,1\}; X_3 \in \{0,\ldots,1\};$$ ← Auxiliary variables

Constraints :

$$O + O = R + 10 * X_1$$

$$X_1 + W + W = U + 10 * X_2$$

$$X_2 + T + T = O + 10 * X_3$$

$$X_3 = F$$

each letter has a different digit ($F \neq T, F \neq U$, etc.);

Types of Constraints

Unary Constraints:

Restriction on single variable

Binary Constraints:

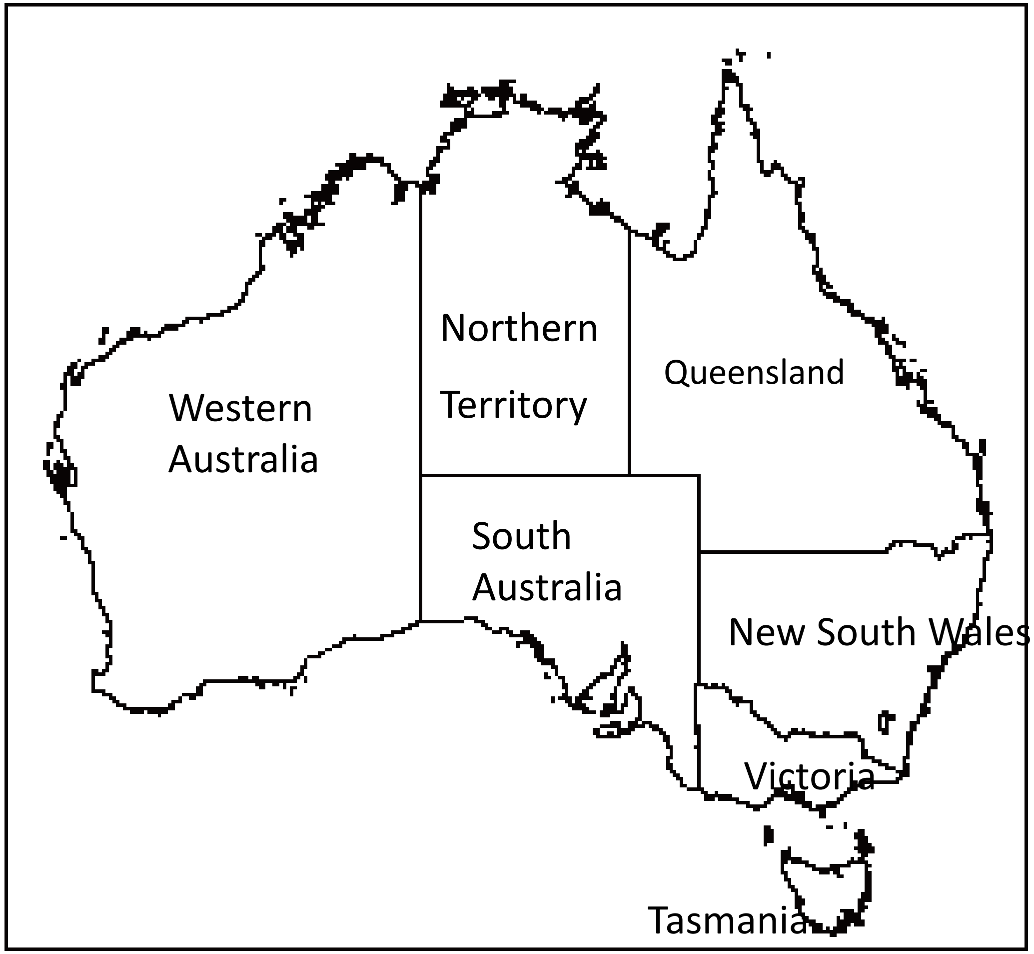
Restriction on pairs of variables

Higher-Order Constraints:

Restriction on more than two variables

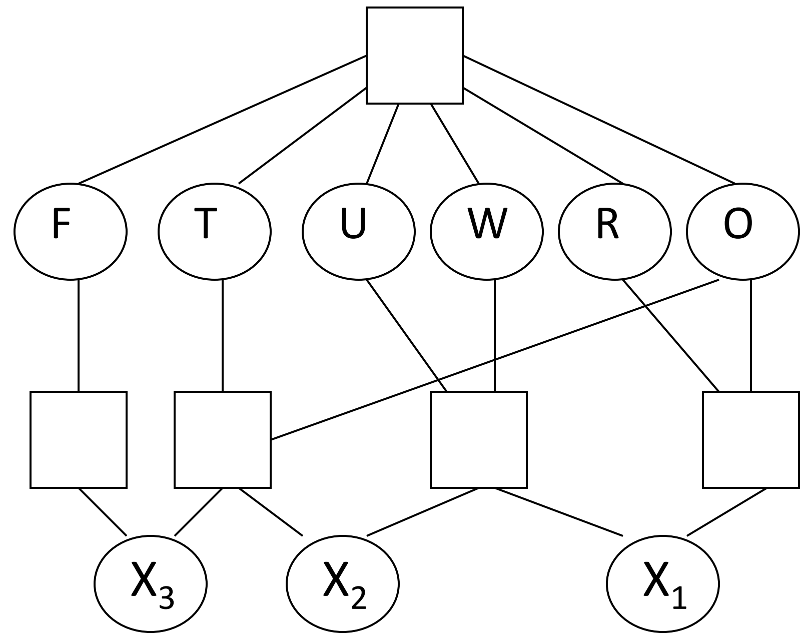
Preferences vs. Constraints

Map Coloring Problem



Constraint Hypergraph

```
   TWO
 + TWO
 -----
  FOUR
```



Types of variables

- Discrete domains
  - Boolean {T,F} ← 3-Sat, K-Sat
  - Finite domains {a,b,c…}
  - Infinite (e.g. all integers)
    - constraints represented using language,
    - e.g. $X<Y$, $Y>Z+5$
- Continuous domains
  - Linear ← linear programming
  - Nonlinear

Constraint Satisfaction Problems (CSP)

For a given CSP the problem is one of the following:

1. find all solutions

2. find one solution
    - · just a feasible solution, or
    - · A "reasonably good" feasible solution, or
    - · the optimal solution given an objective

3. determine if a solution exists

How to View a CSP as a Search Problem?

Initial State - state in which all the variables are unassigned.

Successor function - assign a value to a variable from a set of possible values.

Goal test - check if all the variables are assigned and all the constraints are satisfied.

Path cost - assumes constant cost for each step

Branching Factor

Approach 1- any unassigned variable at a given state can be assigned a value by an operator: branching factor as high as sum of size of all domains.

Approach 2 - since order of variable assignment not relevant, consider as the successors of a node just the different values of a *single* unassigned variable: max branching factor = max size of domain.

Prefer BFS or DFS?

B=BFS **D=DFS**

CSP – Goal Decomposed into Constraints

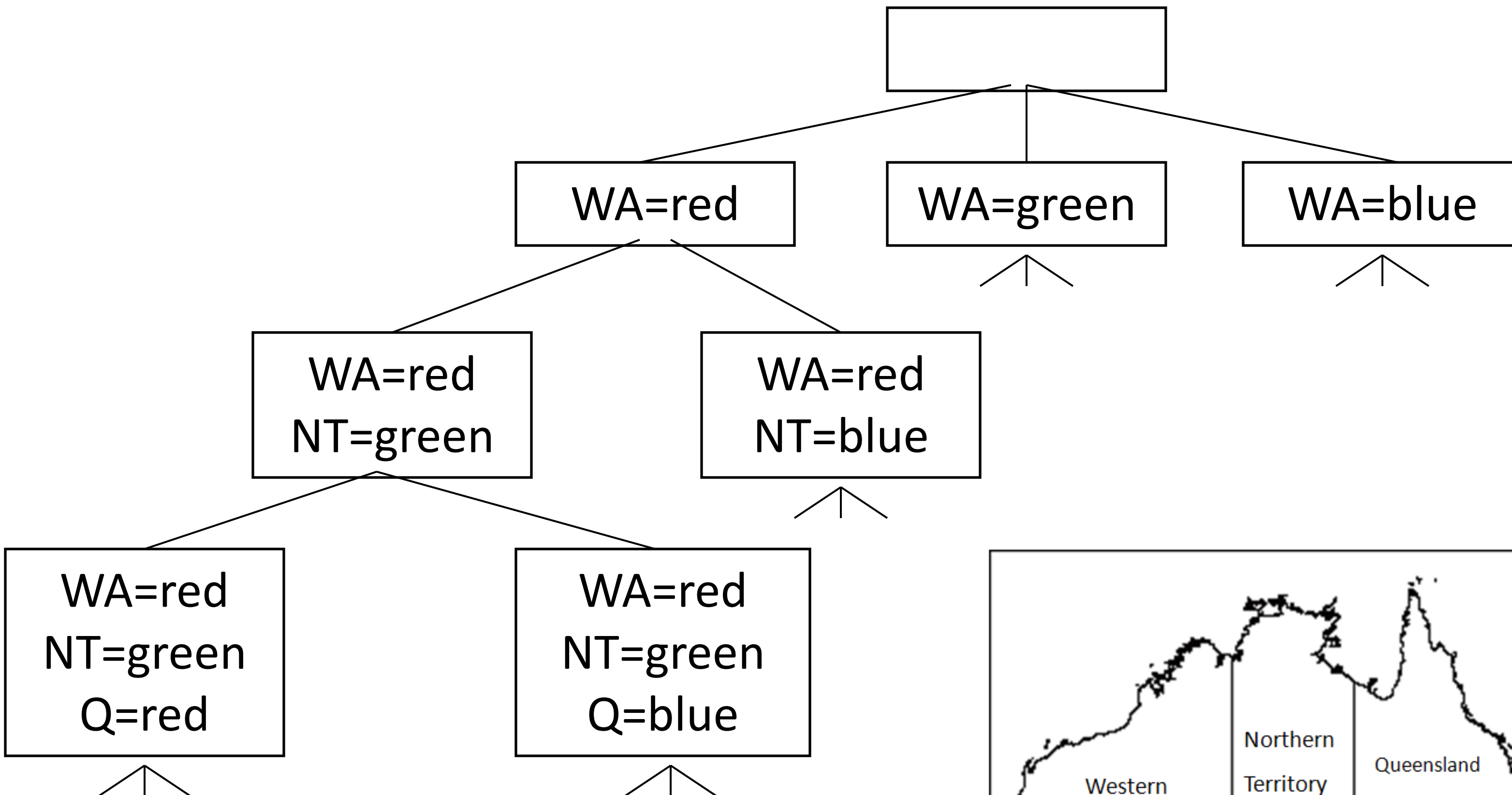
Backtracking Search: a DFS that

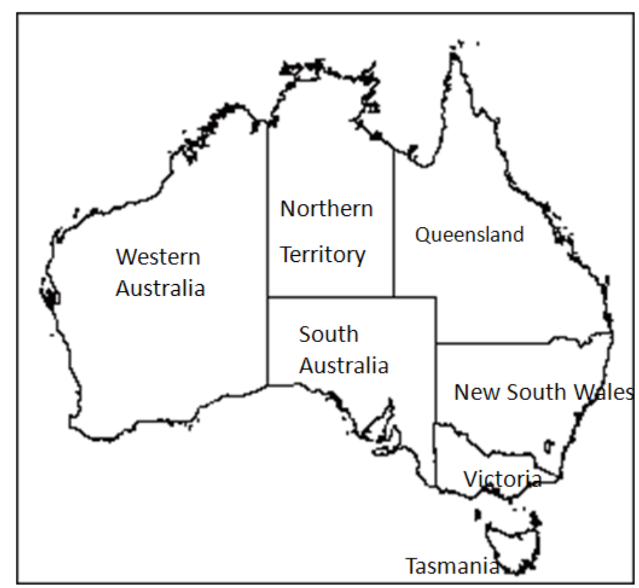
- chooses values for variables one at a time
- checks for *consistency* with the constraints.

Decisions during search:

- Which variable to choose next for assignment.
- Which value to choose next for the variable.

Example


WA=red
WA=green
WA=blue
WA=red
NT=green
WA=red
NT=blue
WA=red
NT=green
Q=red
WA=red
NT=green
Q=blue


Northern
Territory
Queensland
Western
Australia
South
Australia
New South Wales
Victoria
Tasmania

Minimum Remaining Values (MRV)

- Idea: Assign most constrained variable first
- Prune impossible assignments fairly early
- Degree heuristic: choose higher degree first

Which is best order according to MRV heuristic?

- A = NT, SA, WA, Q, NSW, V, T
- B = T, V, SA, NSW, WA, NT, Q
- C = SA, Q, NSW, V, NT, WA, T

Forward Checking

- **Idea:** Reduce domain of unassigned variables based on assigned variables.
  - Each time variable is instantiated, delete from domains of the uninstantiated variables all of those values that conflict with current variable assignment.
- Identify dead ends without having to try them via backtracking
  - E.g. if last variable has zero options, no need to go that deep to find out

General Purpose Heuristics

Variable and value ordering:

Degree heuristic: assign a value to the variable that is involved in the largest number of constraints on other unassigned variables.

Minimum remaining values (MRV): choose the variable with the *fewest* possible values.

Least-constraining value heuristic: choose a value that rules out the smallest number of values in variables connected to the current variable by constraints.

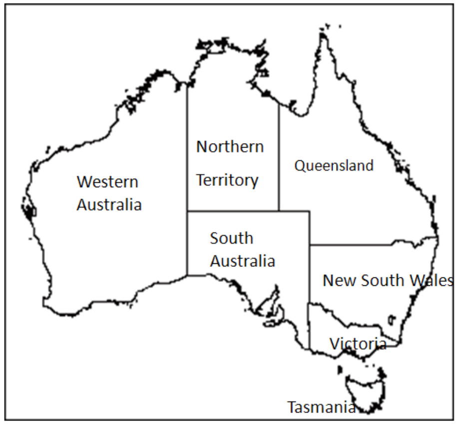
Comparison of CSP Algorithms

Problem	BT	BT+MRV	BT+FC	BT+FC+MRV
USA	(>1,000K)	(>1,000K)	2K	60
N-queens	(>40,000K)	13,500K	(>40,000K)	817K

Constraint Propagation (Arc Consistency)

- **Arc Consistency** - state is arc-consistent, if every variable has some value that is consistent with each of its constraints (consider pairs of variables)

	WA	NT	Q	NSW	V	SA	T
Initial Domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After WA=red	**R**	G B	R G B	R G B	R G B	G B	R G B
After Q=green	**R**	B	**G**	R B	R G B	B	R G B
After V=blue	**R**	B	**G**	R	**B**		R G B



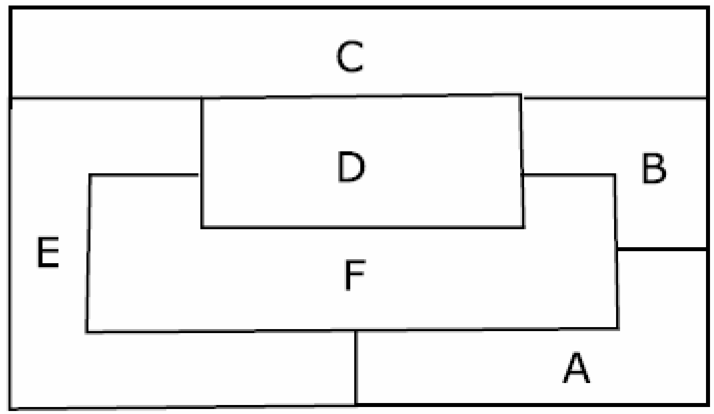
Constraint Propagation (Arc Consistency)

- **Arc Consistency** - state is arc-consistent, if every variable has some value that is consistent with each of its constraints (consider pairs of variables)

Init: Q is queue with all (directed) arcs (X_i,X_j) in CSP
WHILE Q is not empty
- (X_i,X_j) = remove_first(Q)
- FOREACH $\boldsymbol{x \in dom(X_i)}$
    *IF no $\boldsymbol{y \in dom(Xj)}$ satisfies constraint (X_i,X_j)
    ·THEN remove x from $dom(X_i)$
- IF $dom(X_i)$ changed
    *THEN add all arcs $(\boldsymbol{X_k}, \boldsymbol{X_i}) \notin \boldsymbol{Q}$ to $\boldsymbol{Q}$

Example: Arc Consistency

Task: 3-color



Solution:

	A	B	C	D	E	F
	RGB	RGB	RGB	RGB	RGB	RGB
A=R	(R)	GB	RBG	RBG	GB	GB
B=G	(R)	(G)	~~R~~ B	R ~~B~~	G ~~B~~	B

$$D \neq F : D = \{R, \cancel{B}\}$$
$$E \neq F : E = \{G, \cancel{B}\}$$
$$C \neq D : C = \{\cancel{R}, B\}$$

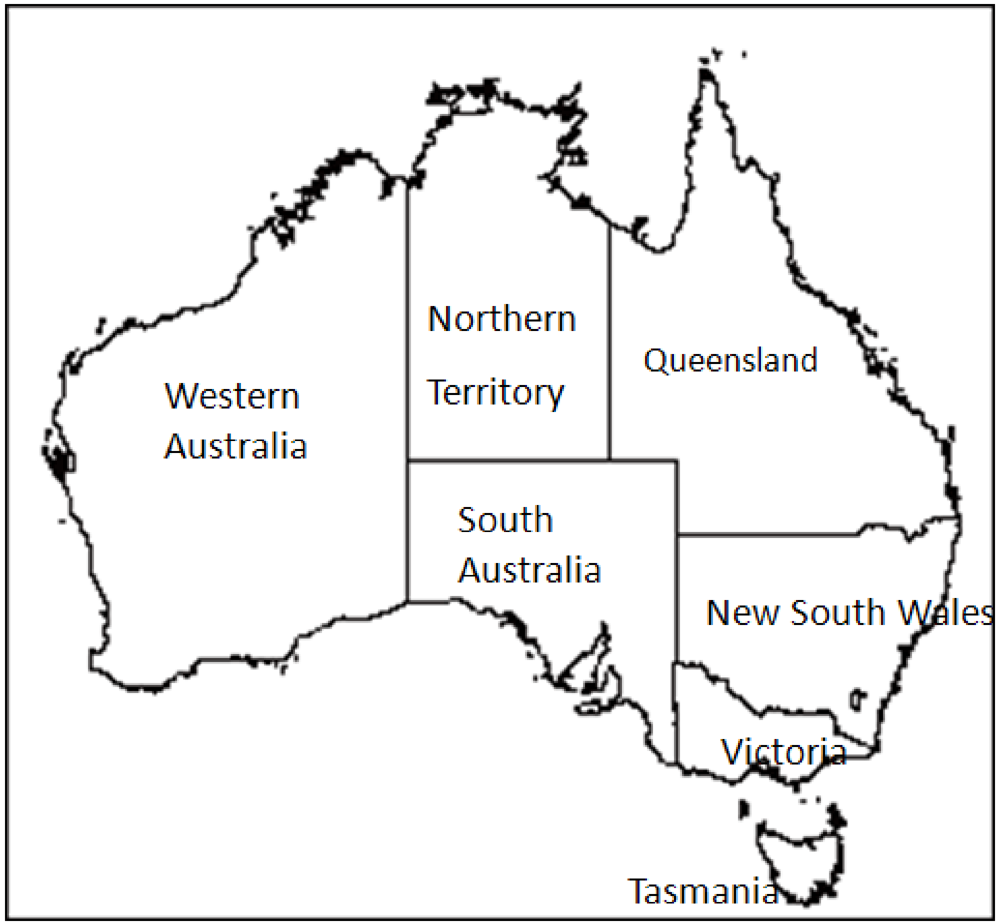
Constraint Propagation (K-Consistency)

- **K-Consistency** generalizes arc-consistency (2-consistency).
- Consistency of groups of K variables.
- Path consistency

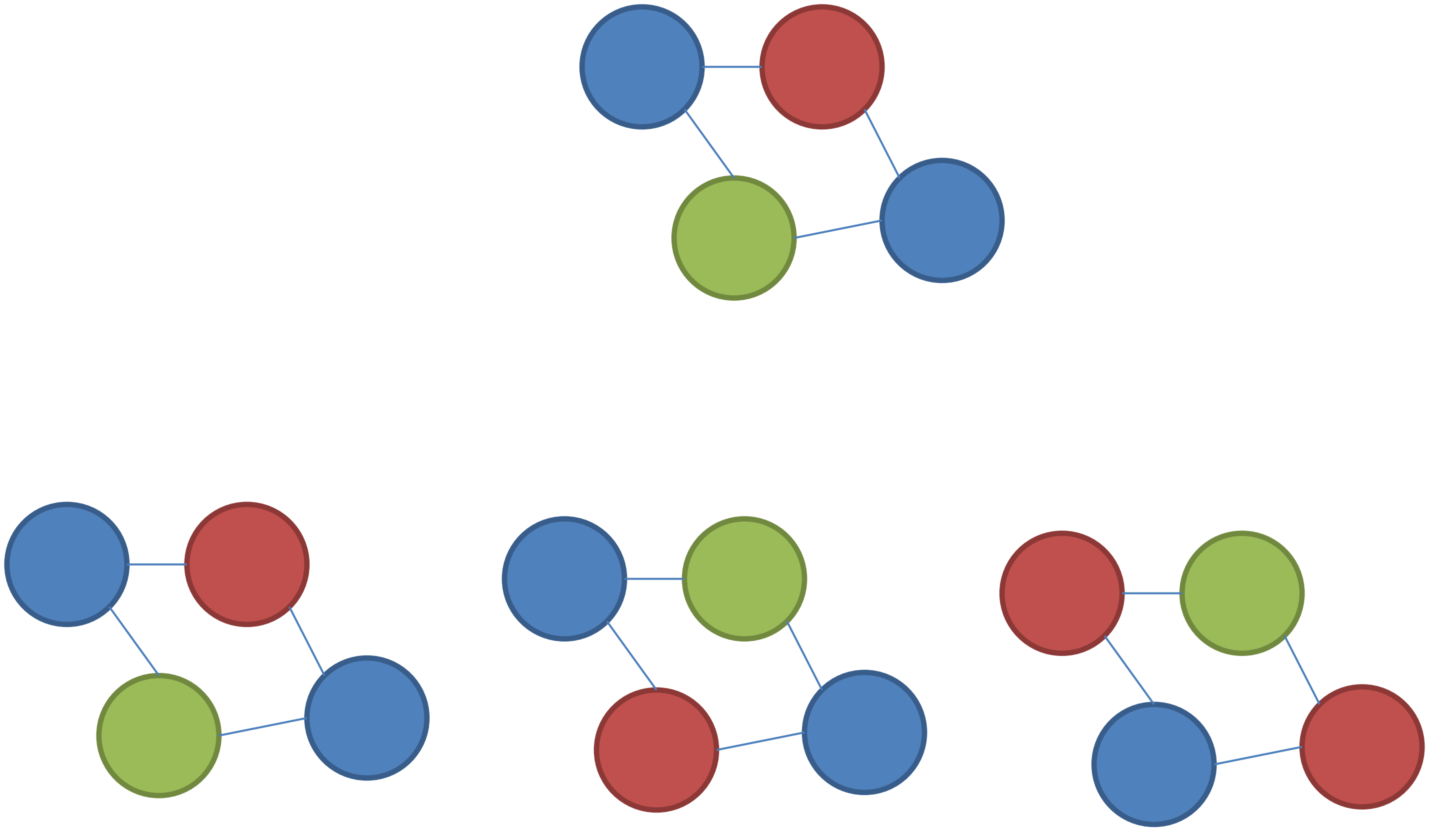
Constraint learning

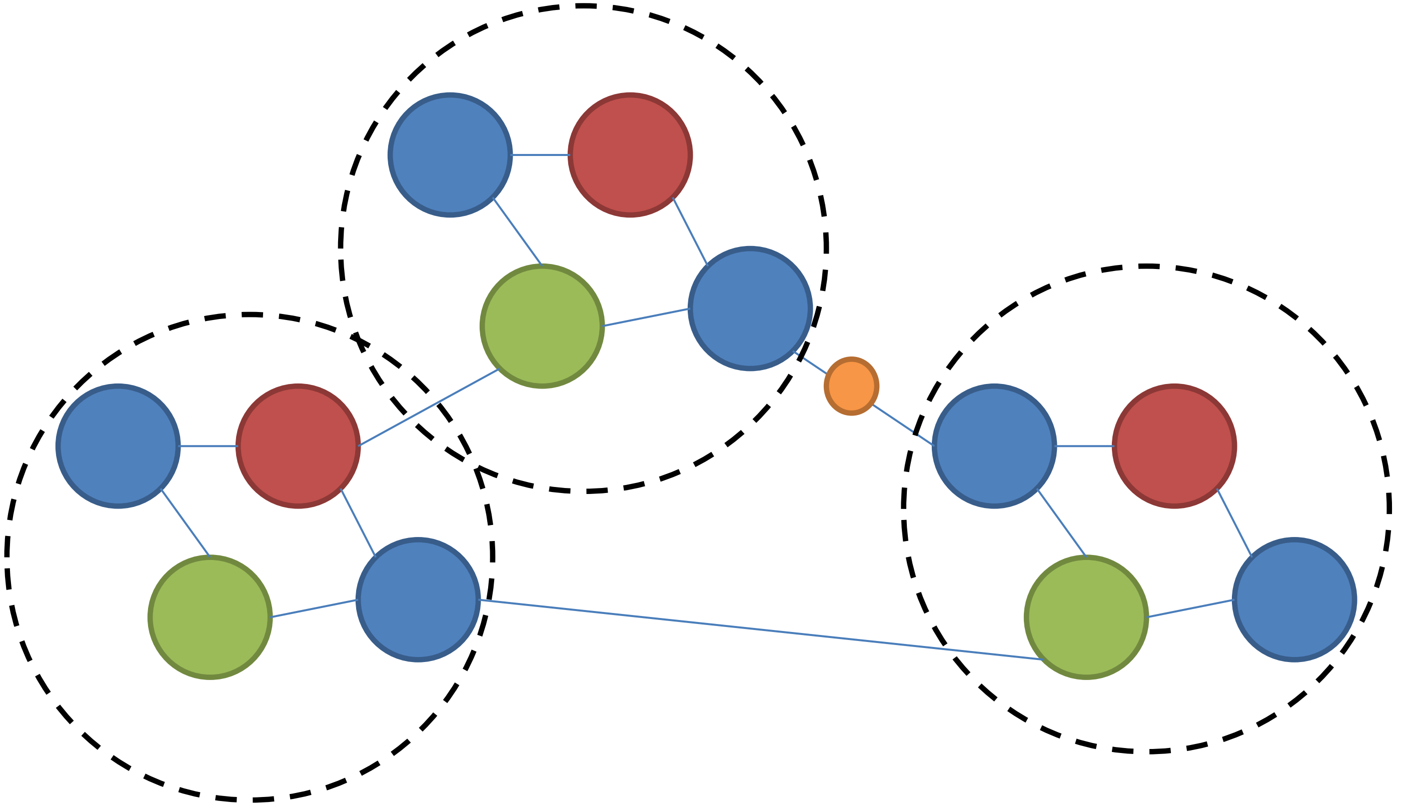
- When assignments fail, is there a way to learn new constraints?
  - Conflict-directed back-jumping looks to find the root cause of a failure and adds it as a new constraint

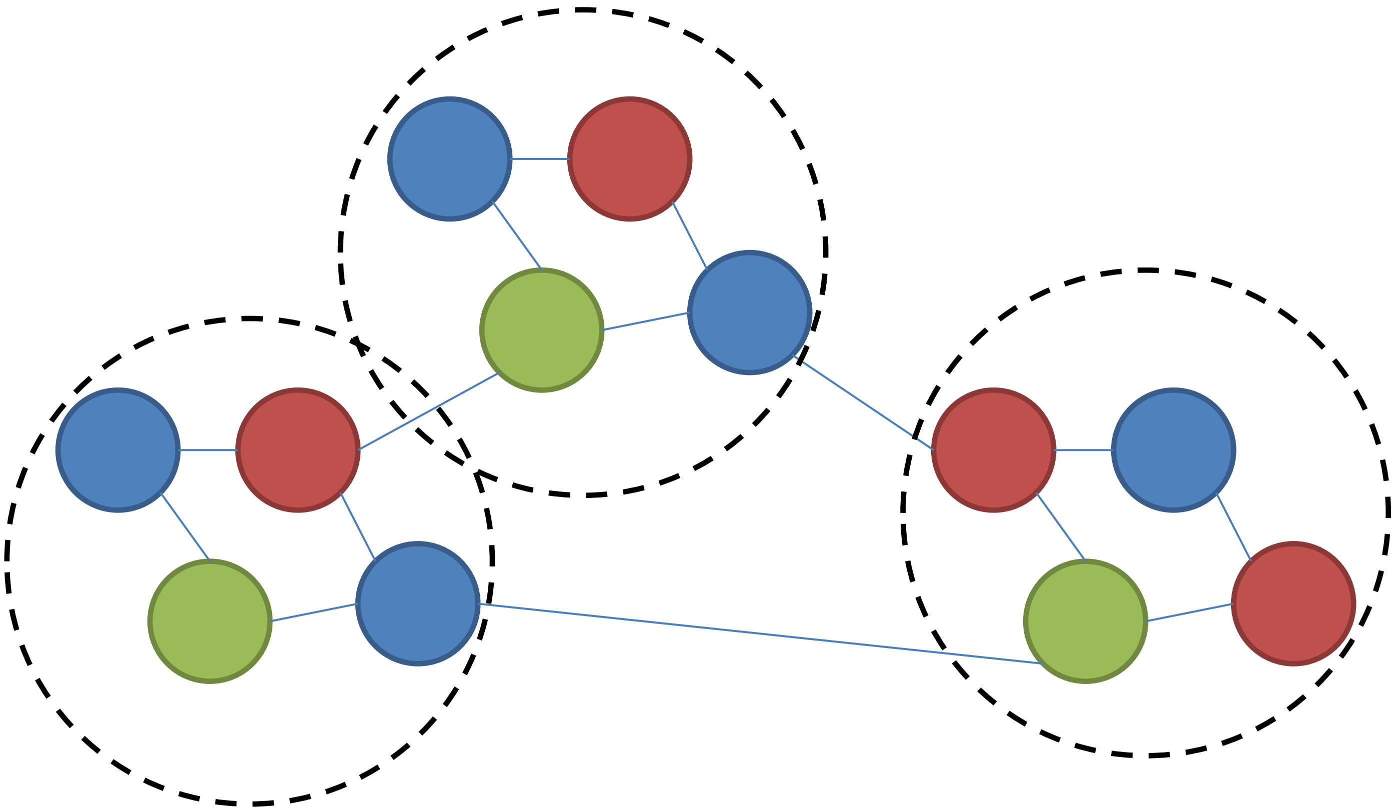
Substructure



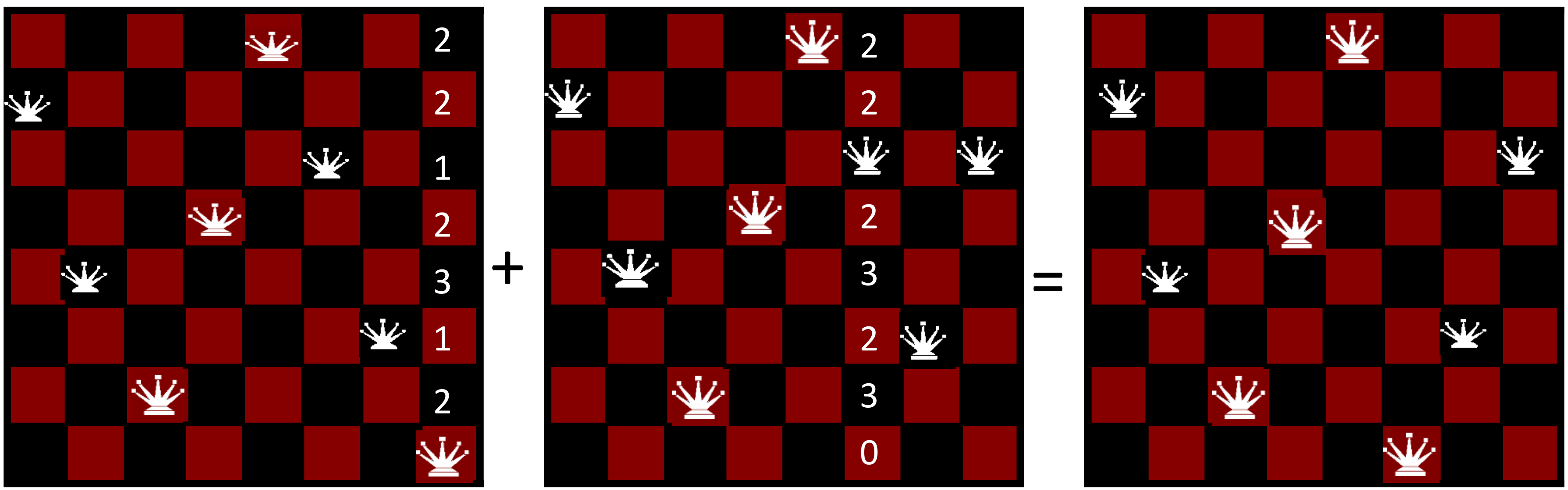
More substructure: Symmetries







Local Search for CSPs



Remarks

Dramatic recent progress in Constraint Satisfaction. Methods can now handle problems with **10,000** to **100,000** variables, and up to **1,000,000** constraints.