Constraint Satisfaction Problem

29/02/2024

Koustav Rudra

CSPs: Recap

- CSP Structure
  - Variables
  - Domains
  - Constraints
    - Implicit (code to compute)
    - Explicit (list of legal tuples)
    - Unary / Binary / n-ary
  - Goals
    - Find any solution
    - Find optimal solution

CSP Solver

- Backtracking give huge <u>gain in speed</u>

- Ordering
  - Which variable should be processed next (MRV)?
  - In what order values of the chosen variable be tried (LCV)?

- Filtering
  - Can we detect eventual failure early?
  - Arc consistency

NP-hard

- Structure
  - Can we exploit the problem structure?

CSP: Efficient Solver

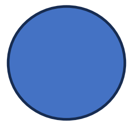
29/02/2024

Koustav Rudra

K-Consistency

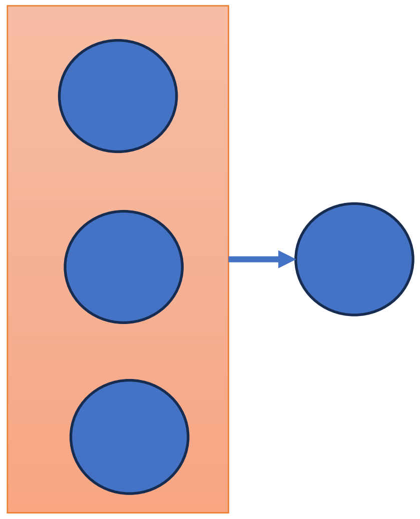
- Consistency ➔ Non-violence of constraints

- Degrees of consistency
  - 1-consistency (node consistency)
    - Unary constraints



  - 2-consistency (Arc consistency)
    - Any consistent assignment to one can be extended to other
    - Binary

  - K-consistency
    - Any consistent assignment to K-1 nodes can be extended to the Kth node
    - If we have satisfactory assignments for K-1 nodes
      - We can find a satisfactory assignment for Kth node

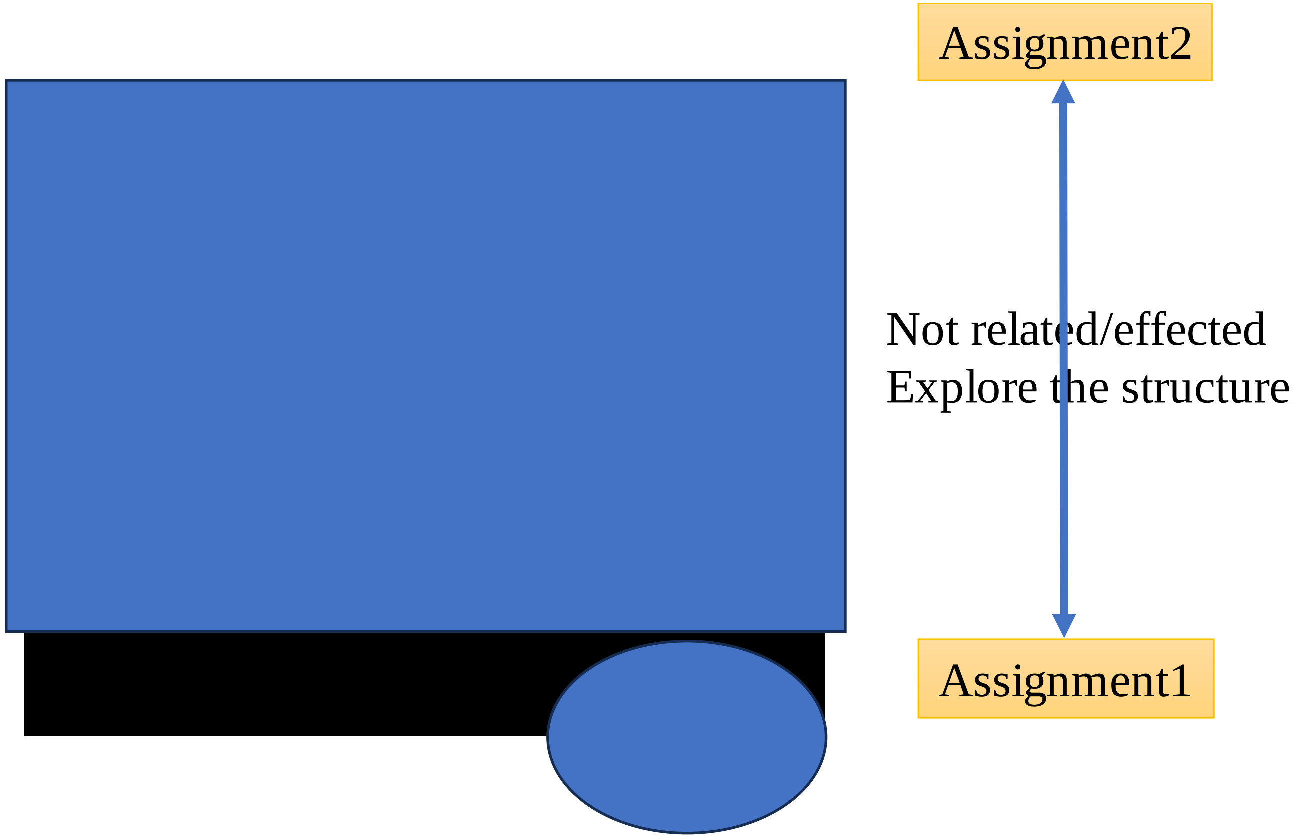


Strong K-Consistency

- Strong K-consistent ➔ K-1, K-2,…,1 consistent
- Strong N-consistency ensures solution without backtracking [N variable CSP]
  - Choose random assignment of any variable
  - Choose a new variable
  - 2-consistency ➔ there is a choice consistent with the first
  - Choose another variable
  - 3-consistency ➔ there is a choice consistent with the first two
  - …
- What is the limitation?
  - Enforcing strong N-consistency as hard as having the solution
- Trade-off between arc-consistency and K-consistency
  - E.g., 3-consistency aka Path-consistency

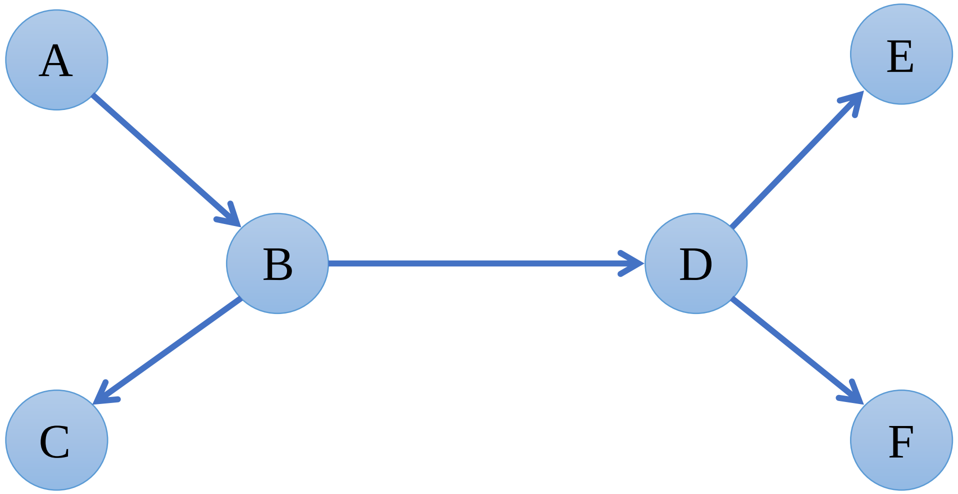
Problem Structure

- Independent Subproblems
  - Mainland and Tasmania do not interact
- How to identify independent subproblems?
  - Connected components of constraint graph
- What is the benefit?
  - Without decomposition running time: $O(d^n)$
  - Let n variables broken into subproblem of c variables
  - Worst case: $O(\frac{n}{c}d^c)$ ➔ Linear in n
- Let n=100, c=20, d=2
  - Without decomposition: 2^{100}
  - With decomposition: $5 * 2^{20}$



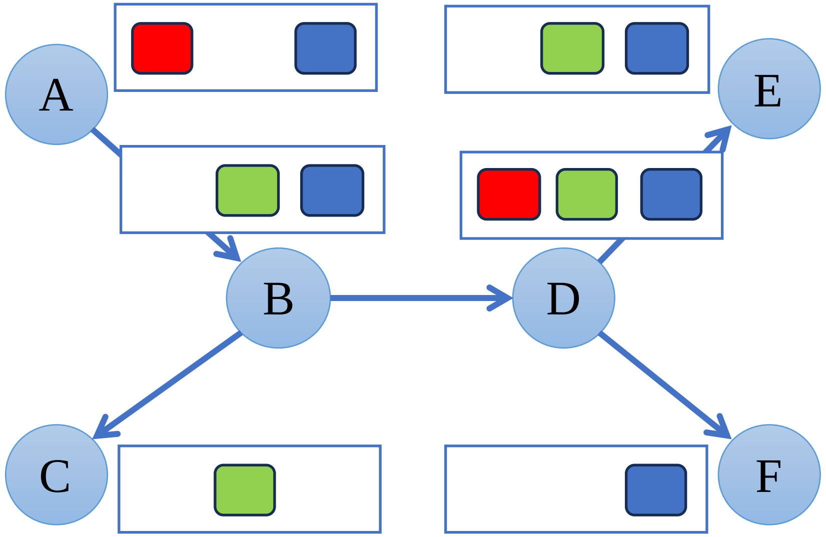
Very uncommon to find a problem structure

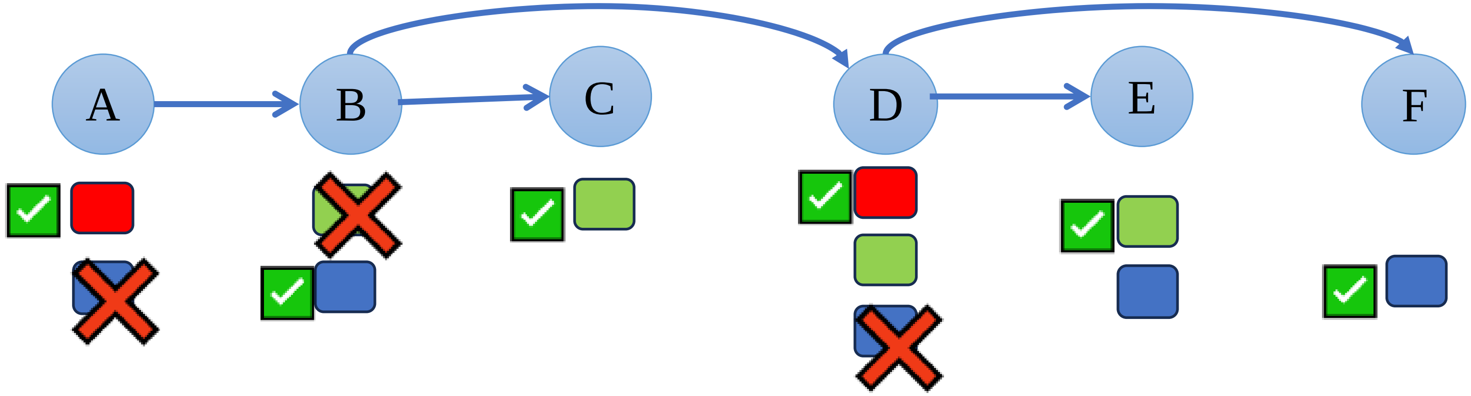
Tree Structured CSP



- No loop
- If constraint graph has no loop, the CSP could be solved in $O(nd^2)$ in worst case

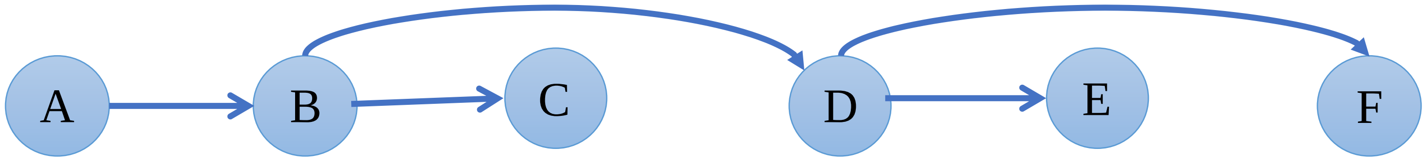
Tree Structured CSP





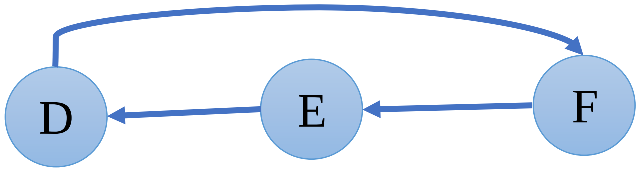
- Order:
    - Choose a Root variable
    - Order other variables in such a way that parents precede children
- Remove Backward:
    - For i=n to 2, REMOVEINCONSISTENT($Parent[x_i], x_i$)
- Assign Forward:
    - For i=1 to n, Assign x_i consistently with $Parent[x_i]$
- Runtime: $O(nd^2)$ Forward Pass: n, Backward Pass: n, Comparison: d^2

Tree Structured CSP



- After Backward pass all the root-to-leaf arcs are consistent
  - After a backward pass, each x→y has been made arc consistent
  - Y’s children have been processed before y
  - Y’s options can’t be reduced

- Forward assignment will not backtrack

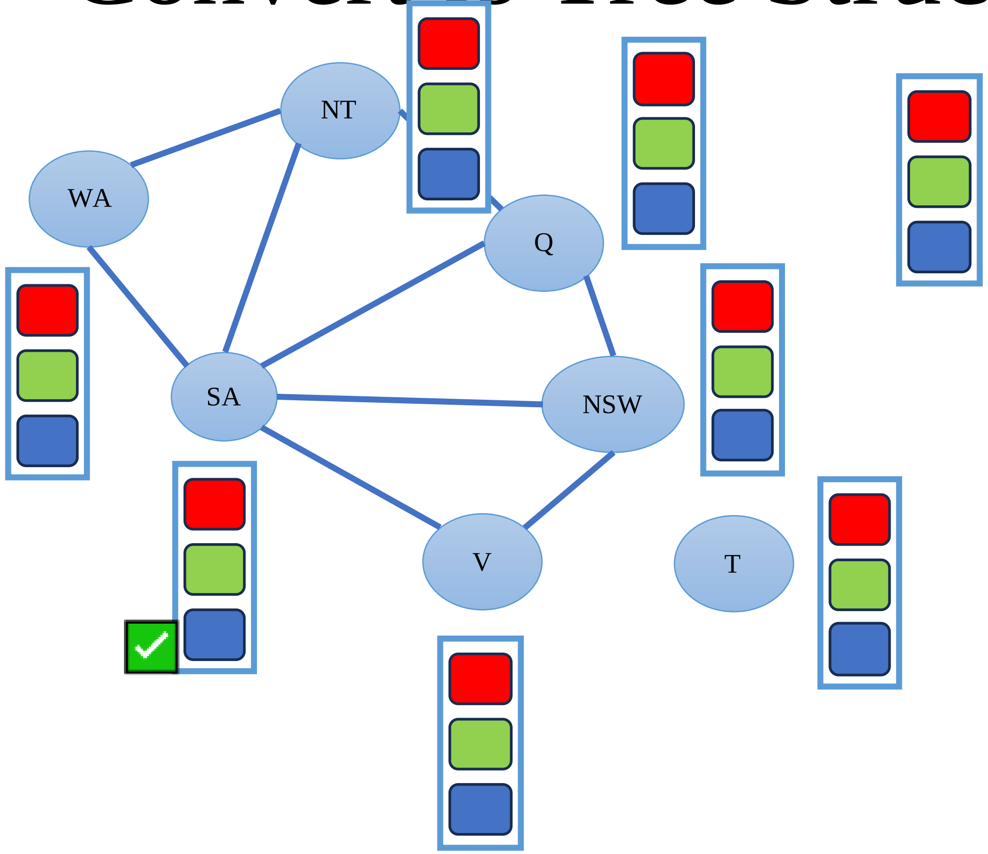


- Does not work on constraint graphs with cycles?

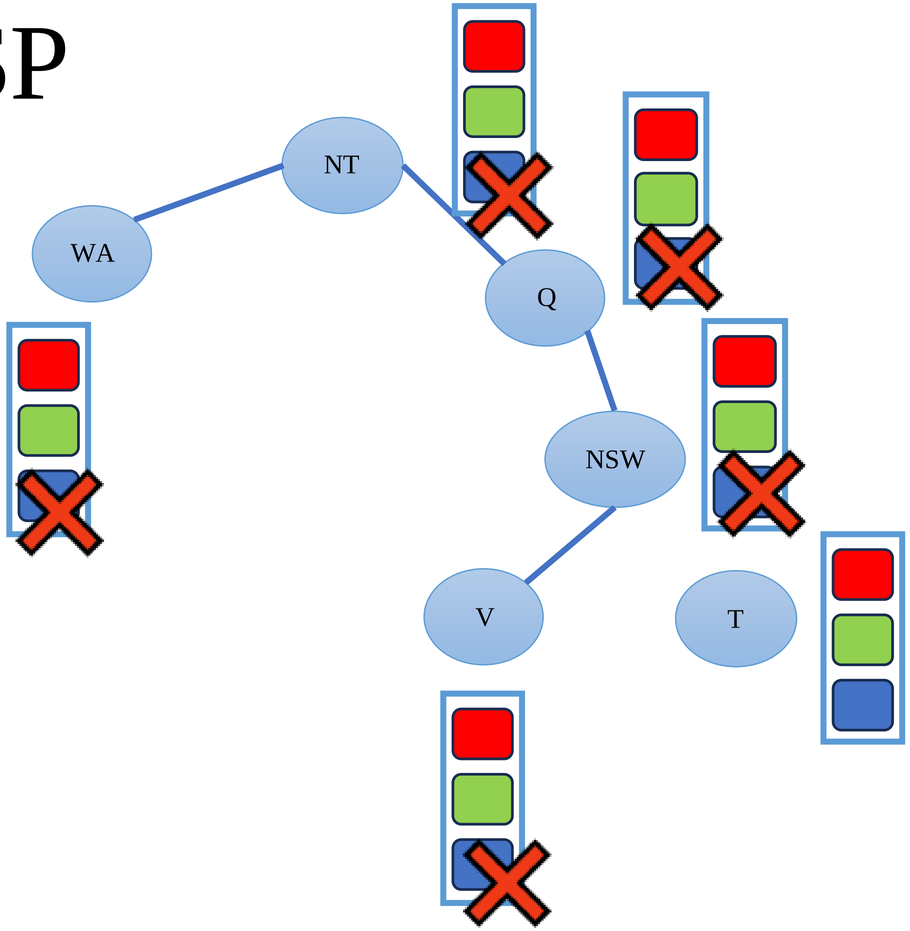
- Tree Structured CSP is not common

- How can we convert a CSP to Tree Structured CSP?

Convert to Tree Structured CSP



CUTSET={SA}



- Conditioning: Forcefully initiate a variable and prune the domains of the neighbours
- CUTSET-CONDITIONING:
    - Obtain a CUTSET of variables
    - Removing those will leave the constraint graph a tree
    - Instantiate (in all ways) the CUTSET

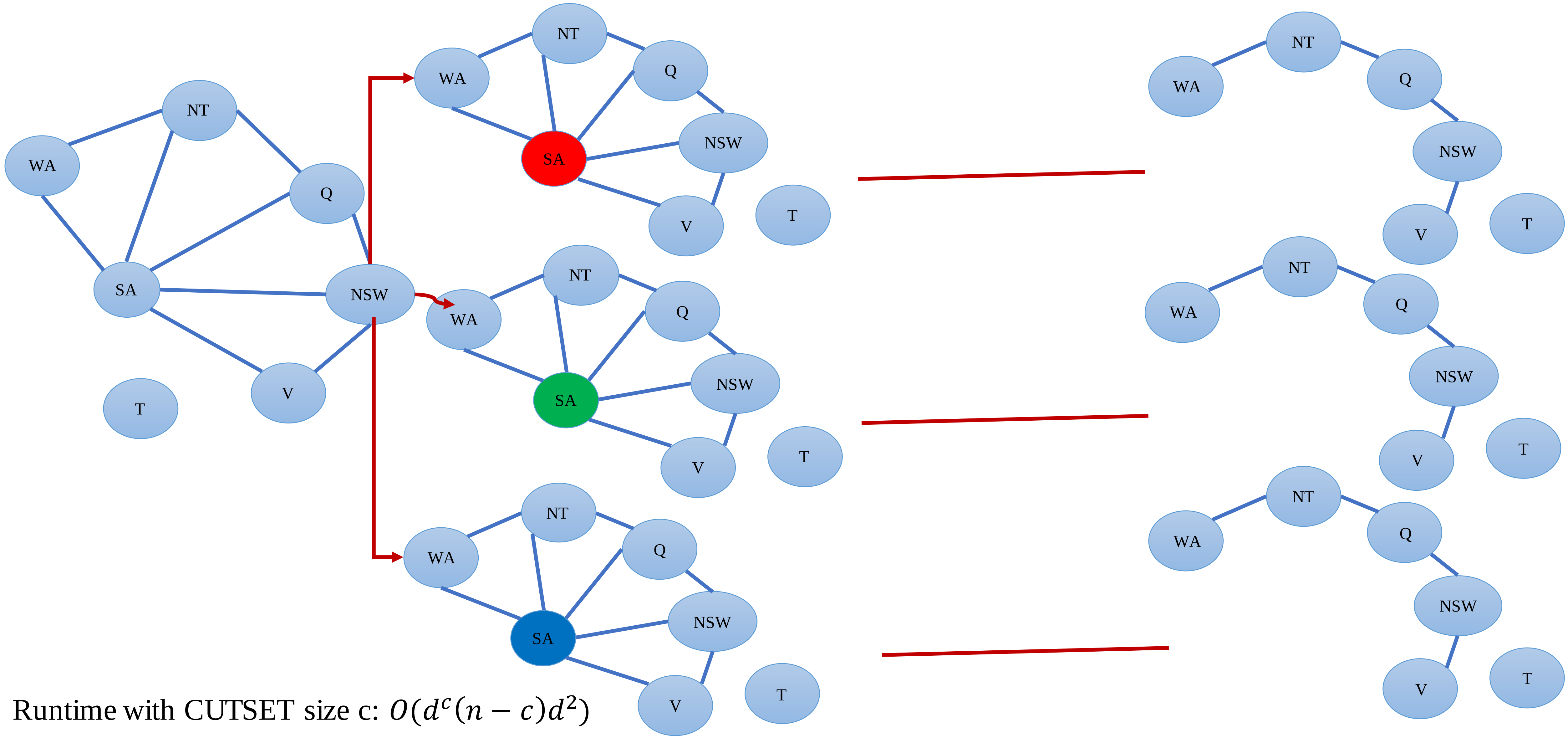
CUTSET Conditioning

Choose a CUTSET

Instantiate CUTSET

Compute Residual

Solve Residual TreeCSP



Runtime with CUTSET size c: $O(d^c(n-c)d^2)$

Thank You