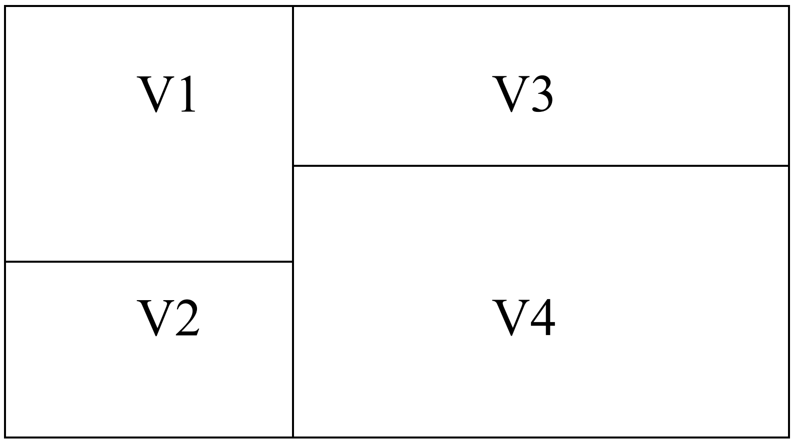
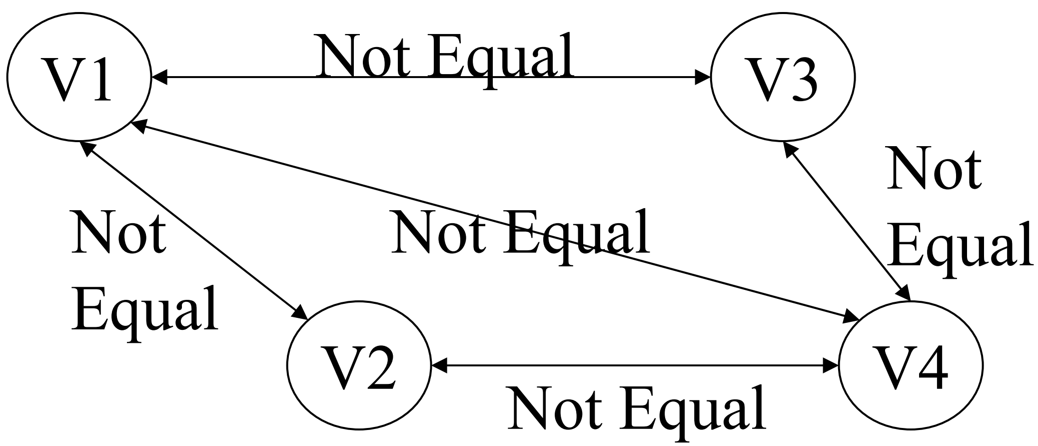
Constraint Satisfaction

6.871 – Lecture 17

An Example Problem: Map Coloring



Domain (V1) = Red, Blue, Green Yellow
Domain (V2) = Red, Blue, Green, Yellow
Domain (V3) = Red, Blue, Green, Yellow
Domain (V4) = Red, Blue, Green, Yellow



Solution Strategy 1: Generate and Test

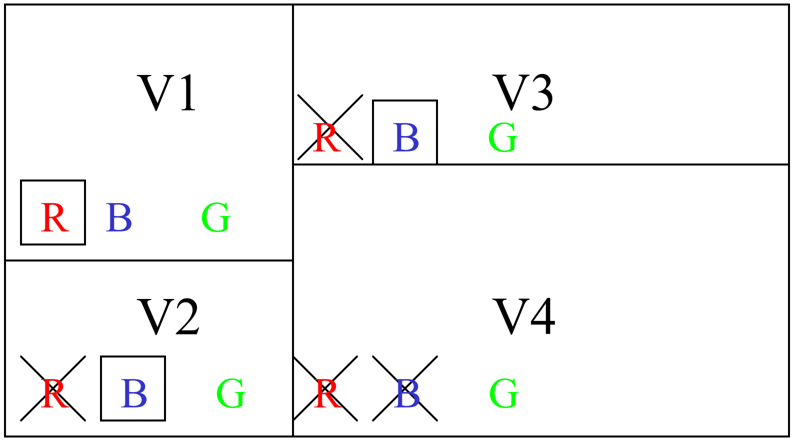
- For Each Variable, Select a Value
  - Then test set of select values for constraint violations
- Iterate until solution found

Select V1 = Red, V2 = Red, V3 = Red, V4 = Red
Test for violated constraints -> Yep.
Generate Next Candidate
Select V1 = Red, V2 = Red, V3 = Red, V4 = Blue
Test for violated constraints -> Yep.

Etc.

Solution Strategy 2: Forward Pruning

- Like G&T
  - But, When choosing a variable, prune out inconsistent choices in connected nodes:



Select V1= Red

Prune V2 = Red

Prune V3 = Red

Prune V4 = Red

Select V2 = Blue

Prune V4 = Blue

Select V3 = Blue

Prune V4 = Blue (already done)

Solution Strategy 3: Backtracking

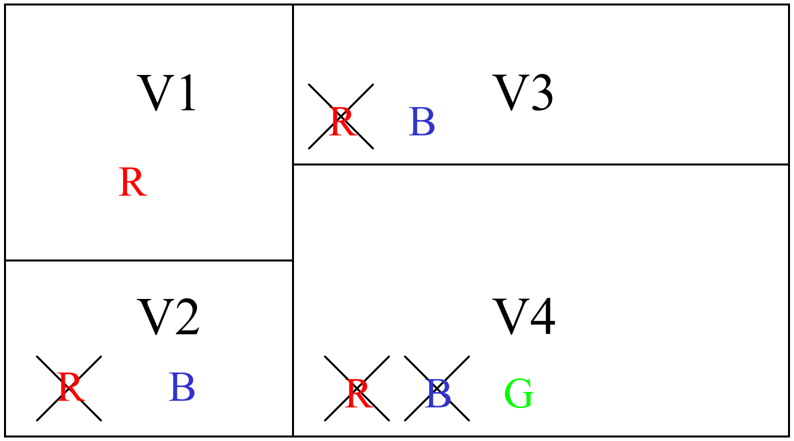
For each variable

Select a value

Check for consistency with previous choices

If not make a different choice

If out of choices backup to previous variable



V1 = Red

V2 = Red,   Inconsistent Try new value

V2 = Blue

V3 = Red, Inconsistent, Try new value

V3 = Blue

V4 = Red, Inconsistent, Try new value

V4 = Blue, Inconsistent, Try new value

V4 = Green

A Backtracking Problem

- Consider MarthaPatrick-Stewart, v2.0
  - Boiling corn on the cob
  - Has two ways to pick up the corn: fingers, serving spoon
  - Has two hands: left and right
  - How to pick up the corn
    - AI to the rescue: use backtracking search.
- What went wrong?

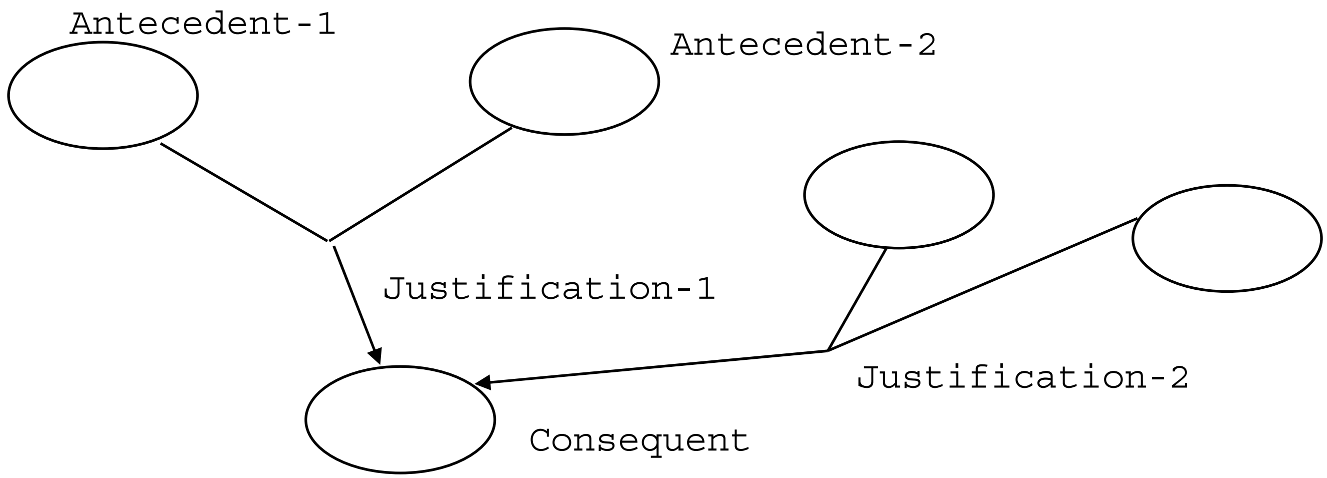
Solution Strategy 4: Intelligent Backtracking

- Like Backtracking
  - Possibly augmented with forward pruning
- When run out of values for a variable backtrack to a *choice related to the problem*, not necessarily to the most recent choice
- Why is this an issue?
  - Because the order of variable assignments isn't necessarily the same as the topology of the constraint network.
  - The constraint network is only a partial order
  - Variable assignments are totally ordered
  - The most recent choice could be on a parallel branch
- How do we deal with this?
  - Dependency networks

Dependency Maintenance

- Analyzing the failed situation is crucial
  - Understand what earlier assumptions led to the failure.
- Truth Maintenance Systems (TMSs) solve this problem.
  - Retraction of Assumptions: remove all (and only) facts which actually depend on the assumption.
  - Explanation: play back the dependencies as a justification for belief
  - Failure analysis: trace a failure back to the set of inconsistent assumptions.
  - Re-establish assumptions: bring back all facts that depended on the re-established assumption.
  - Context Swapping: allow arbitrary sets of assumptions to be retracted and re-established.

Dependency Networks



- Each node is a fact
  - For CSP, a choice of a value for a variable
- Each node is labeled either *in* or *out (believed or not)*
- Certain nodes are *assumptions* (label is supplied externally)
- With each deduction, system creates a justification whose antecedents are all the facts used in making the deduction and whose consequent is the deduced fact.

Consistency of a Dependency Network

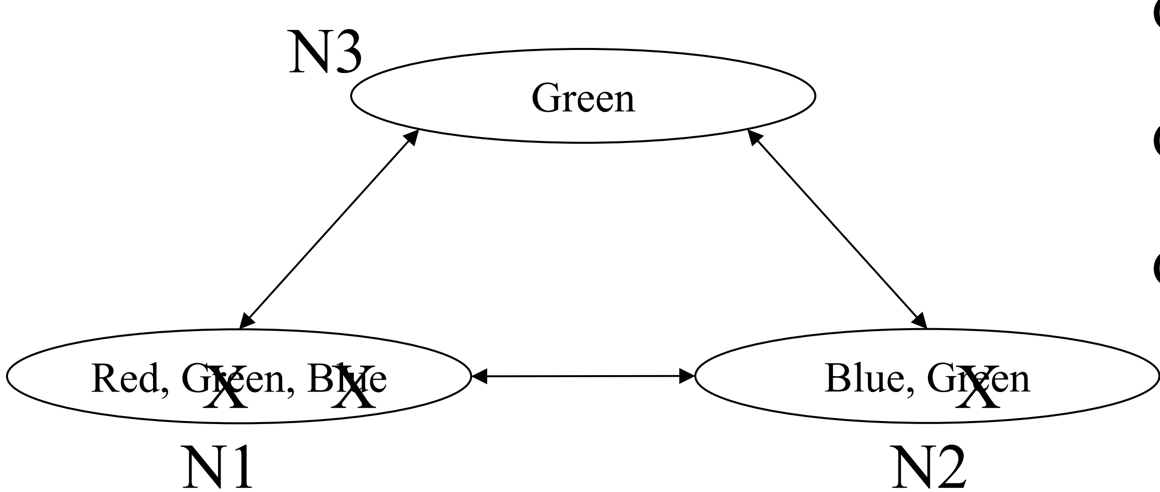
- If all antecedents of a justification are *in* then the justification is *active*.
- If a node is the consequent of an active justification, then the node is *in* .
- If a non-assumption node is not the consequent of an active justification, its label is *out*.
  - i.e., if a node has no justification, we don't believe it

Basic Operations of a TMS

- **Assumption Retraction**, remove the *in* labeling from an assumption. Recalculates status of dependent nodes.
- **Assumption Activation**, add an *in* label to an assumption. Recalculates status of dependent nodes.
- **Find support of a node**: list of all active assumptions currently supporting the node.
- **Handle Contradiction**: Remove a contradiction by finding the assumptions supporting the two contradictory nodes and retracting one of them.
- **Establishing a reasoning context**: Selecting a set of assumptions and giving those (and only those) *in* labels.

Solution Strategy 5:
Arc Consistency

- For each arc A pairing N1 and N2
  - For each value V1 in N1's value set
    - Check that there is a value V2 in N2's value set consistent with V1.
    - If not Prune V1 from N1's value set.
- For each node N1 that was changed:
  - Find all nodes N3 and Arcs A2 such that A2 connects N3 to N1
  - Recheck A2, N3, N1



Check N1-> N2 it's Consistent
Check N2 -> N3
  Delete Green from N2
Check N1 -> N3
  Delete Green from N1
Check N1 -> N2
  Delete Blue from N1

An Example From Logic

- Boolean Constraint Satisfaction

  (Or A B C)

  (Or (not A) C D)

  (Or B (not D))

  (Or (not C) (not B))
- Each variable can be either true or false
- All disjunctions must be true

Solution Strategy 6: Random Sledgehammer

- Assume Boolean valued variables
- Make a random assignment of values to variables
- Find a violated constraint
- Pick one of the values and change it
  - Possibly using a “least constrained” heuristic
- Continue finding and fixing violated constraints
- After exceeding a threshold of steps, try new random seed.
- After exceeding a threshold of seeds, give up.

Constraint Directed Scheduling

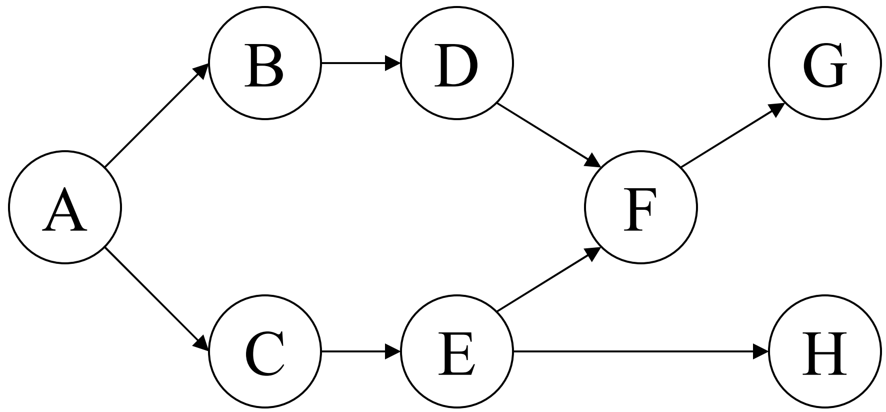
The problem consists of:

- Orders to be satisfied
  - Decomposition of the order into tasks
  - Constraints on task ordering
- Production equipment (with their capacities).
  - Constraints on the use of production equipment to perform specific tasks.
  - Other constraints limiting the use of equipment or the timing of orders.
- Some of the constraints are *hard,* they may not be violated
- Others are *soft,* they may be violated but at some cost

- Problem is to find an assignment of each task to a time slot and an assignment of resources to each task such that no hard constraints are violated and such that cost is minimized.

Precedence Constraints are Hard Constraints

Precedence (PERT) Chart



Constraints:

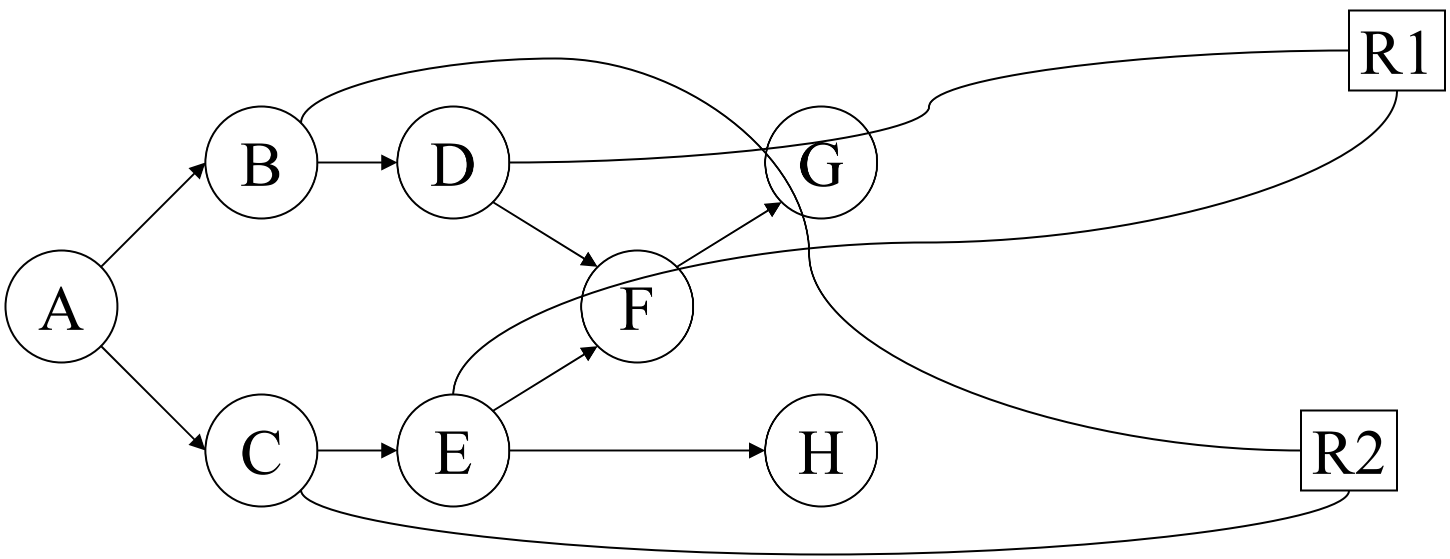
$$EarliestStart(B) \geq EarliestEnd(A)$$

$$EarliestEnd(B) = EarliestStart(B) + Delay(B)$$

$$LatestStart(D) = LatestEnd(D) - Delay(D)$$

$$LatestEnd(B) \leq LatestStart(D)$$

Resource Requirements



$$Overlap(D, E) \rightarrow$$

$$Consumption(D, R1) + Consumption(E, R1) \leq Amount(R1)$$

Other Constraints and Value Functions

- Delivery Dates for each job
- Start Dates for each job
- Production compatibility
  - If A Job uses Resource 1 for step 1 it must use Resource 2 for step 3.
- Personnel restrictions (length of continuous work, schedule)
- Value Function:
  - Work in process
  - Lateness penalties
  - Utilization

Finding a Good Schedule

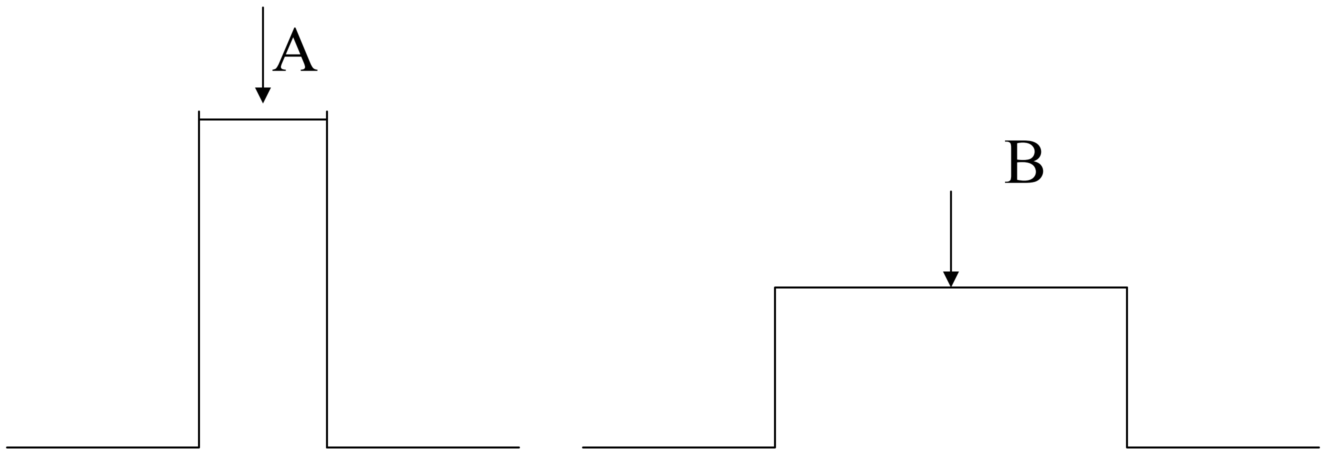
- “Beam Search” for good assignments
- Estimate when each task would “most like to run”
  - (i.e. would incur optimal cost-benefit when viewed in isolation)
- Add up demands for each resource in this schedule
- Find the Most Contended for Resource
  - Find the task competing for this which has minimal slack
  - Assign the resource to that task
- Propagate constraints
  - Assignment of that task
  - Unavailability of resource to other tasks
- Repeat until done.

Why is this difficult?

- Computational Reasons:
  - Coupling of variables through constraints causes a nested search
  - In fact, Boolean constraint satisfaction is classic NP-complete problem
  - <u>Size</u> of Problem leads to <u>Exponential Time</u> for Solution
  - Must resort to heuristics

- Conceptual Reasons:
  - Large collection of different types of constraints
  - Social issues: which one(s) of the incomparable dimensions is to be optimized
  - Dynamism: Things Change (in particular, they break)

What do you want to optimize?



Choose Your portfolio, A or B?

Past the Basic Techniques

- Suppose there is also an evaluation of the solution:
  - Bonuses for early completion of a schedule
  - Some resources cost more than others (relevant if tasks can choose between resources with different costs)
  - Some constraints are “soft” and may be violated with a penalty

- Then we may use any of the techniques to look for an (the) optimal solution
  - Increased complexity and computational time
  - How valuable is the optimum vs. near miss?
  - How fragile is the optimum?

Scheduling as Constraint Satisfaction

- Why is it difficult?
  - Conceptual: Large Number of constraints of different types
  - Social: Whose ox gets gored and who wins
  - Dynamic: Things change and break
  - Computational: Size, realistic problems are very big
- What can we do?
  - Provide a good set of representations for the different types of constraints
    - Be dogged about collecting them
    - Be dogged about keeping them explicit
  - Provide a flexible framework (infrastructure)
  - Make scheduling reactive, incremental, iterative
    - Change driven
    - Seek small change
    - Work from current solution
- Blackboard Style, Incremental solutions

Schedule Repair

- Order Scheduler
- Resource Scheduler
- Right/Left Shifter
- Demand Swapper

Conflict Handling

- Notice the points of Conflict
- Identify the causes of the Conflict
- Diagnose the form of the conflict
- Choose a repair strategy based on the form of the conflict

SAT Problems

- SAT: Given a formula in propositional calculus, is there an assignment to its variables making it true?

- Consider clauses in propositional logic with 3 variables per clause:

- Problem is NP-Complete (Cook 1971)

$$(a \vee \neg b \vee c) \wedge (\neg b \vee d) \wedge (b \vee c \vee e) \wedge \ \ldots\ldots..$$

Figure by MIT OCW.

How Hard is SAT in Practice

- Goldberg (1979) reported very good performance of Davis-Putnam (DP) procedure on random instances.
  - But distribution favored easy instances. (Franco and Paull1983)
- Problem: Many randomly generated SAT problems are surprisingly easy.
- But some are genuinely hard:
  - Job-Shop Scheduling: 10 jobs on 10 machines.
  - Proposed by Fischer and Tompson in **1963**.
  - Solved by Carlier and Pinson in **1990**!
- Open: 15 jobs on 15 machines.

Generating Hard Random Formulas

- Use fixed-clause-length model (Mitchell, Selman, and Levesque 1992)
- Critical parameter: ratio of the number of clauses to the number of variables.
- Hardest 3SAT problems at ratio = 4.3

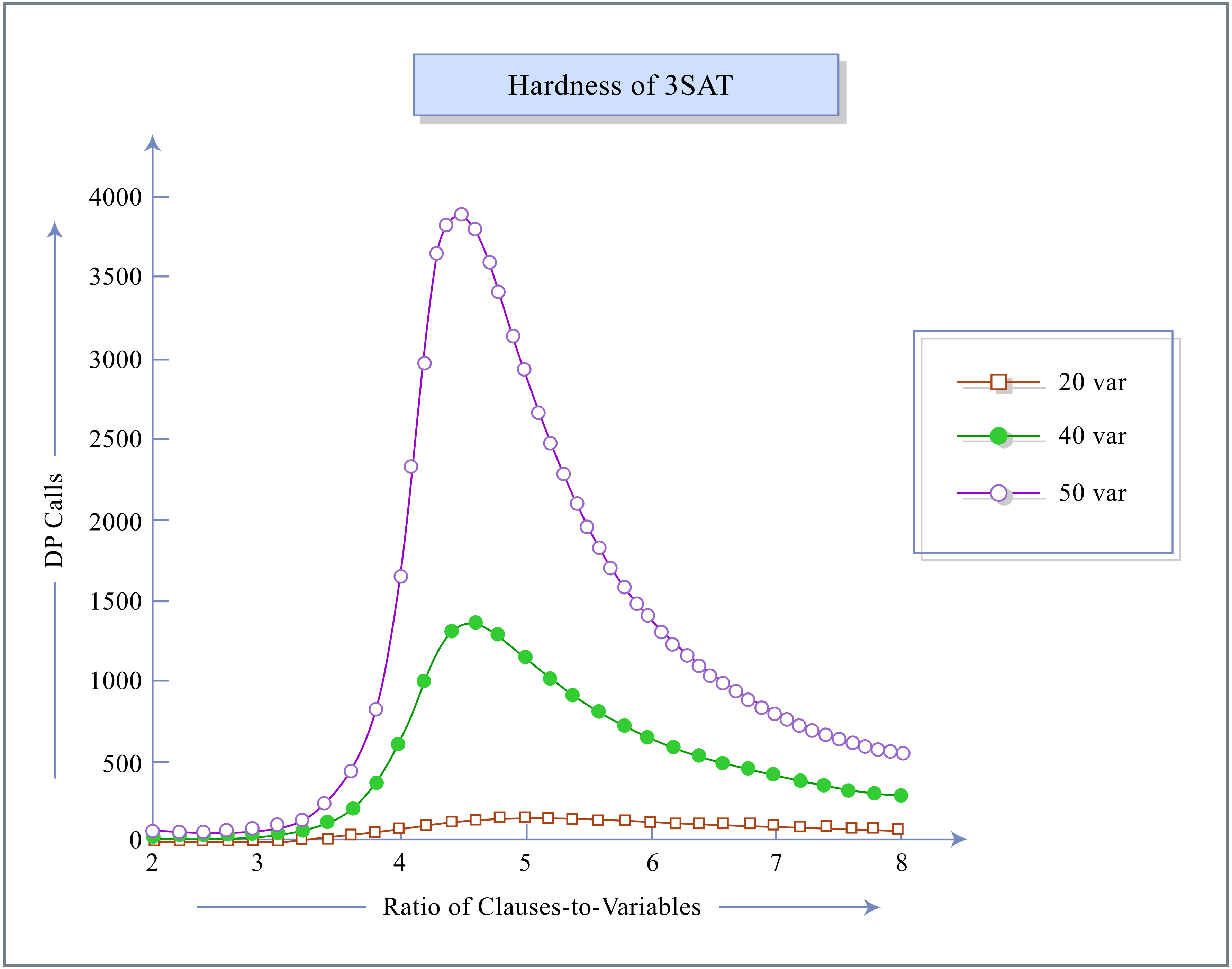
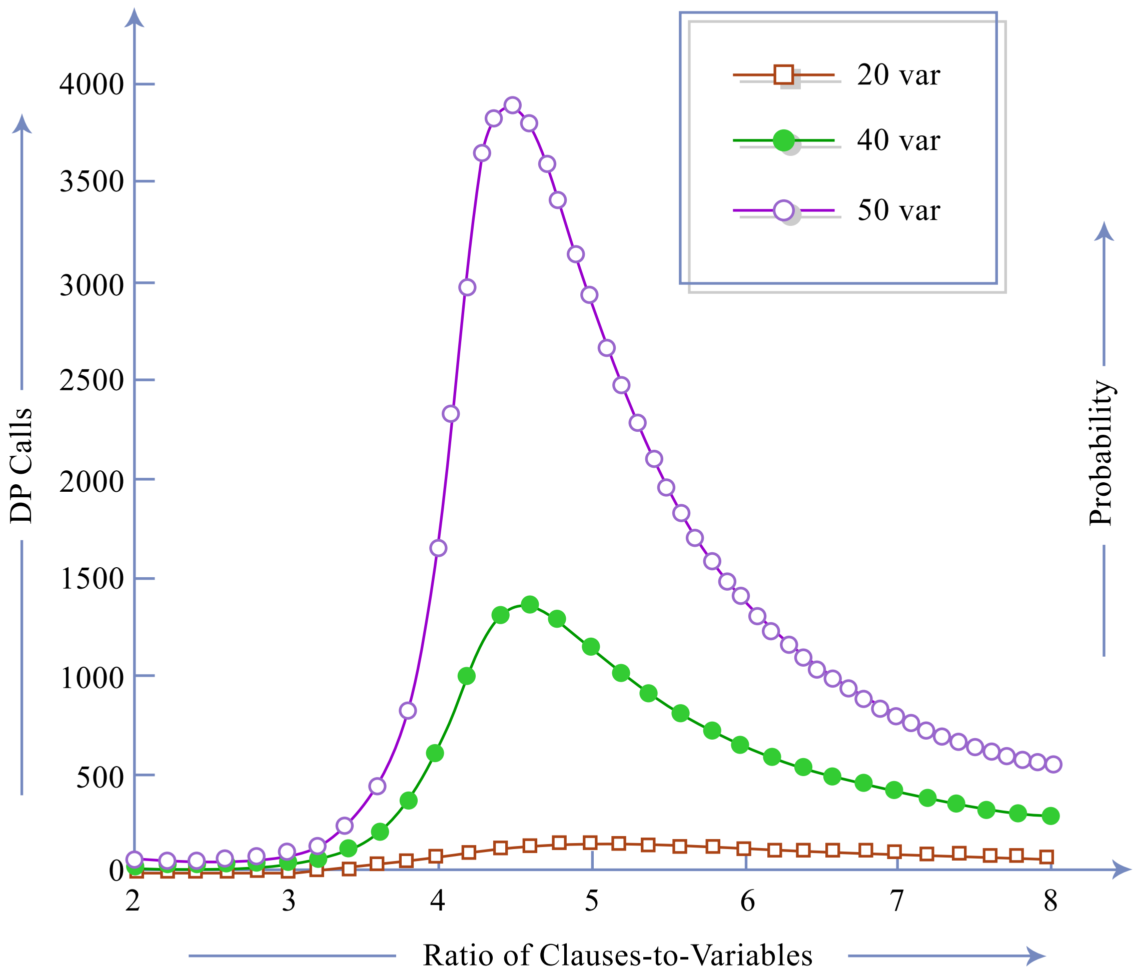


Figure by MIT OCW.

Intuition

- At low ratios:
  - few clauses (constraints)
  - many assignments
  - easily found
- At high ratios:
  - many clauses
  - inconsistencies easily detected

The 4.3 Point



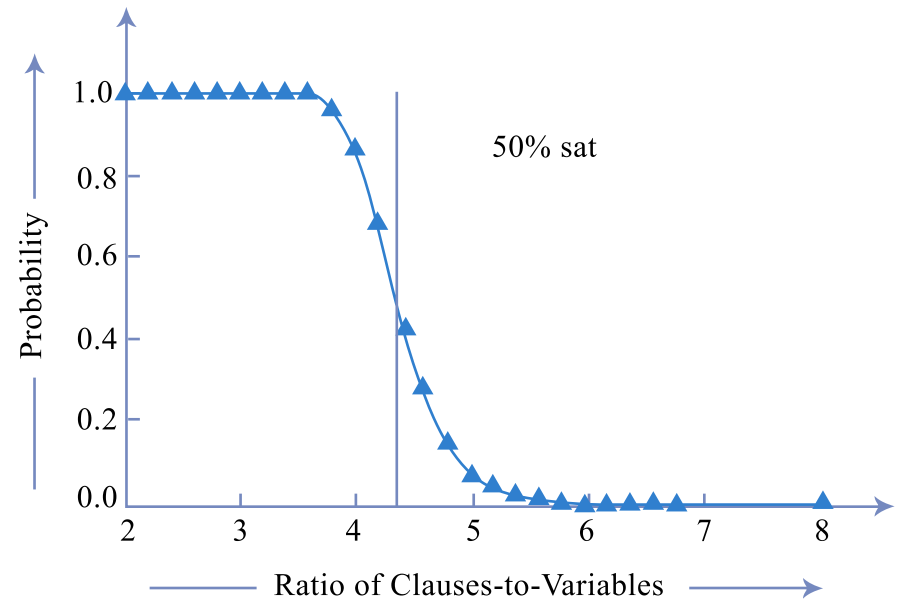


Figure by MIT OCW.

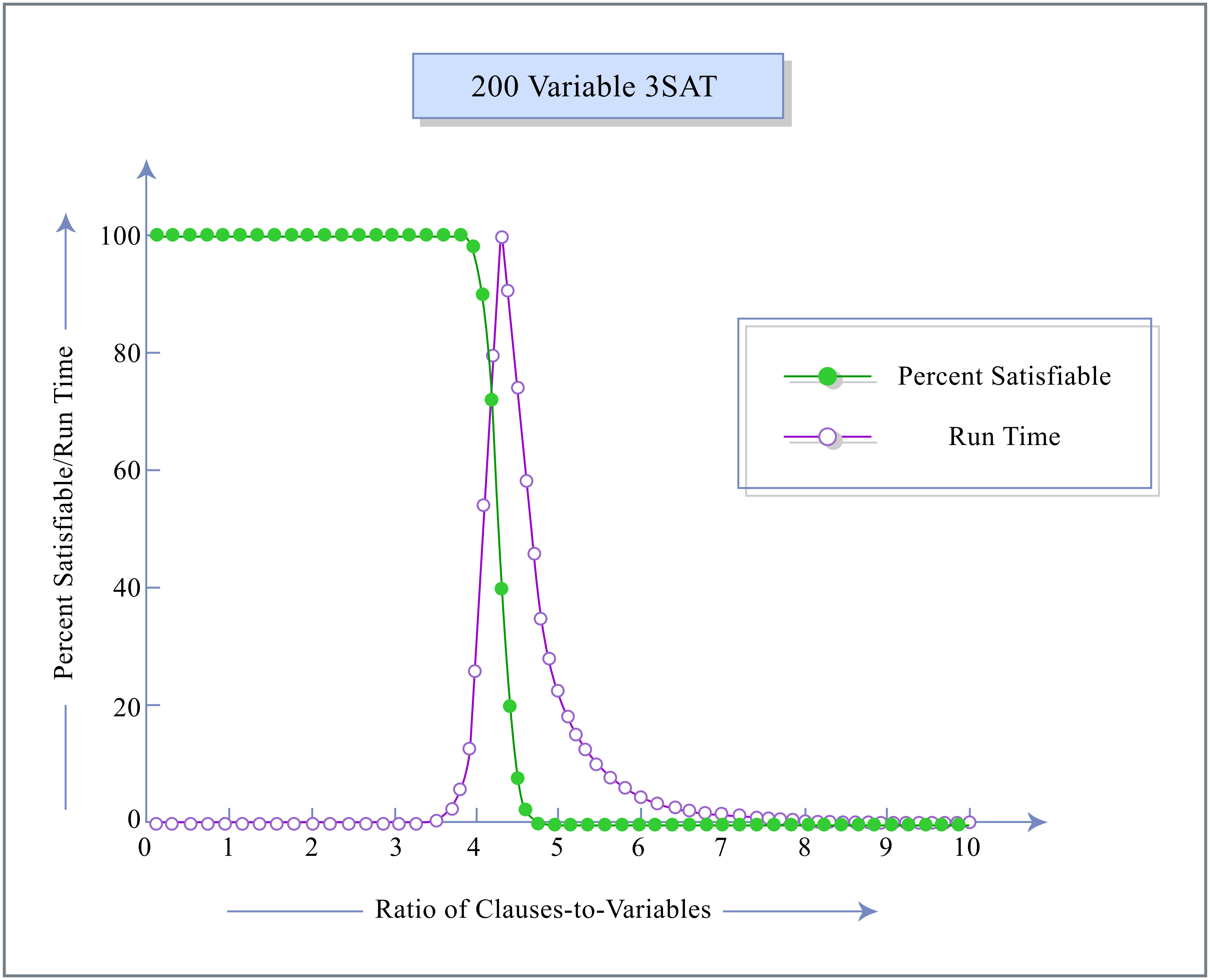
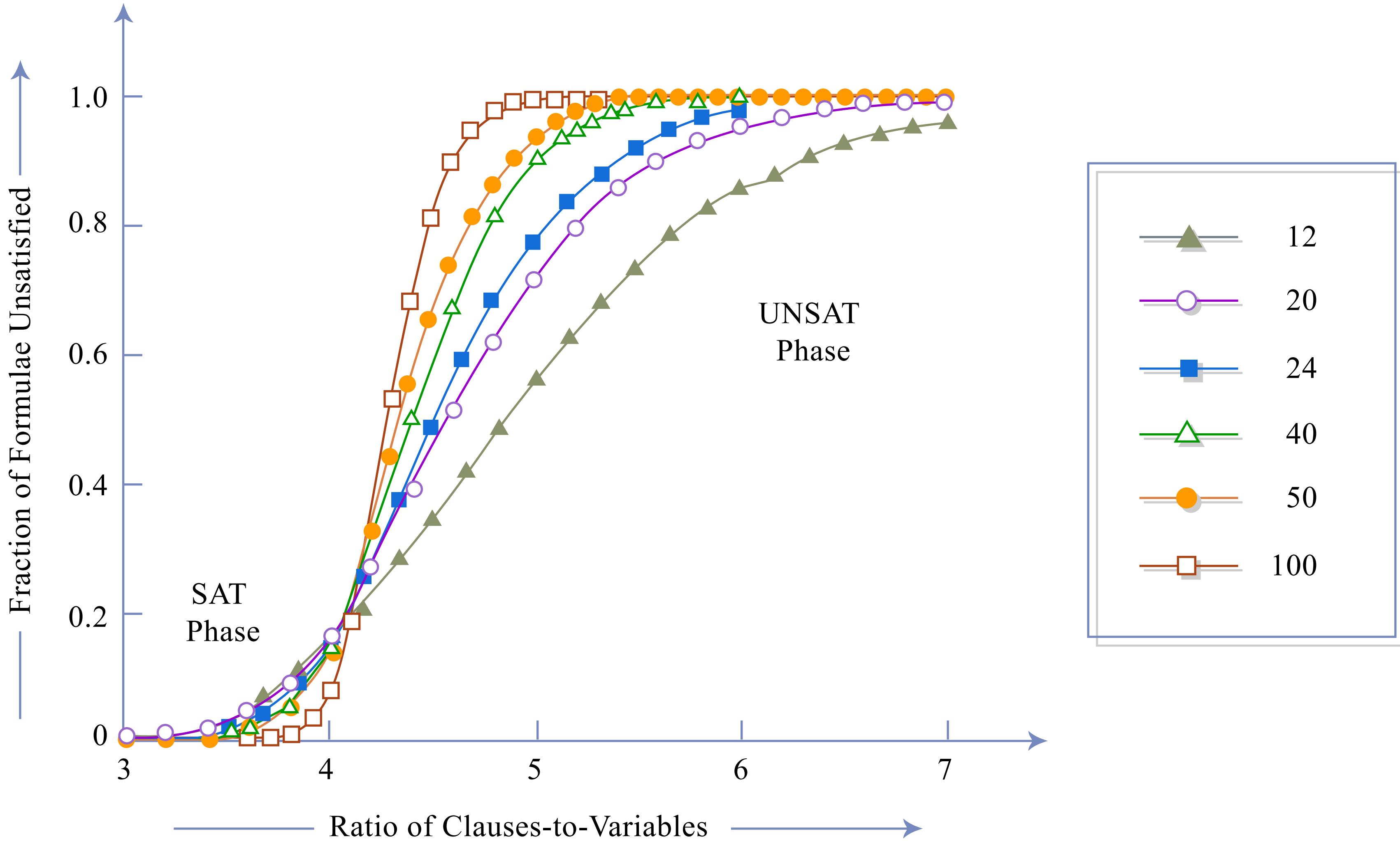


Figure by MIT OCW.

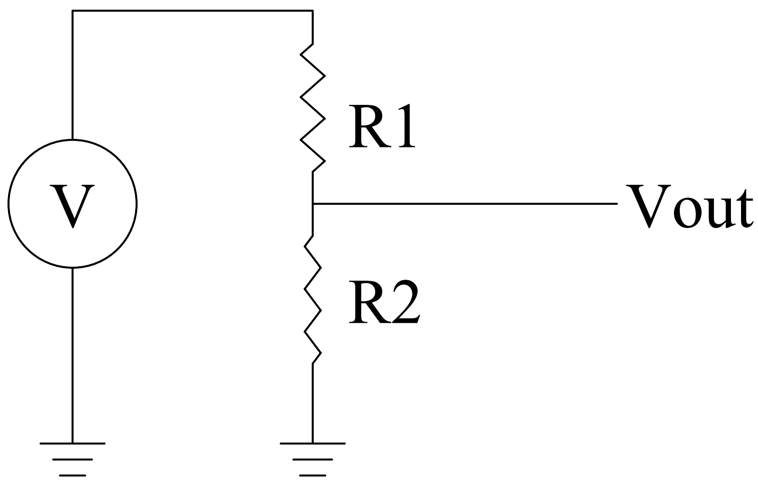
A Closer Look At The 3SAT Phase Transition



Transition sharpens up for higher values of N

Figure by MIT OCW.

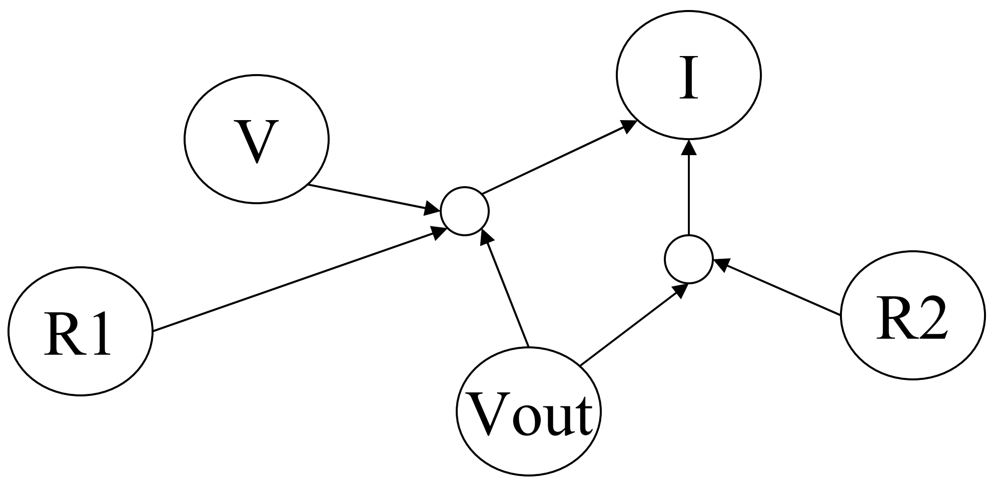
Using Continuous Values: An Example from Electronics



$$I1 = \frac{(V - Vout)}{R1}$$

$$I2 = \frac{Vout}{R2}$$

$$I1 = I2$$



Planning as CSP

- Characterize each operator in terms of the constraints it imposes on the situation before and after its application
- Characterize the world model in terms of the constraints it imposes on variables and their values within a single situation
- Pick a finite number of Plan Steps (n + 1 situations)
  - Or proceed one layer at a time adding as many steps as are consistent
- Reduce to a Boolean CSP
  - Good idea if bounded number of variables and values
  - Consider each assignment of a value to a variable as a distinct proposition
  - Instantiate all the world constraints in each situation
  - Instantiate each operators constraints between each pair of succeeding situations
  - Crunch the resulting CSP

The Blackboard Model

- Origin in speech understanding (but not used there anymore)
- Multiple Levels of Abstraction
- Multiple sources of Knowledge
- Scheduler chooses which activated knowledge source to run next
- Opportunistic behavior
  - E.g. Work from a position of greatest certainty
  - E.g. Work out from the most constrained resource
  - E.g. Work out from the task with least flexibility
- Can be a good organizing principle when multiple KSs are required,no fixed control strategy

Some Example Applications of SAT

- Constraint satisfaction
  - scheduling and planning
  - temporal reasoning (Allen 1983)
  - VLSI design and testing (Larrabee 1992)
- Direct connection to deductive reasoning

  $\Sigma \models \alpha$ iff $\Sigma \cup \{\neg \alpha\}$ is not satisfiable
- Part of other AI reasoning tasks
  - diagnosis / abduction
  - default reasoning
- Learning

Figure by MIT OCW.