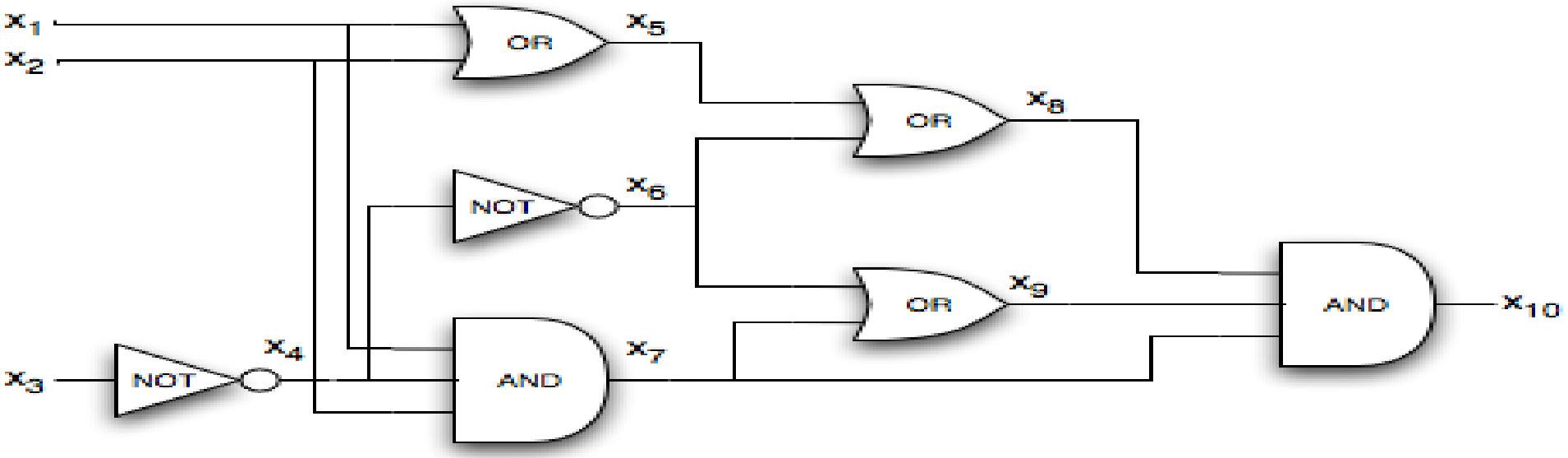
Karlsruhe Institute of Technology

Practical SAT Solving

Lecture 3

Carsten Sinz, Tomáš Balyo | May 2, 2016

NSTITUTE FOR THEORETICAL COMPUTER SCIENCE



Homework Points

- You get homework points for doing homework and showing up with them on the exercises
- You will have the possiblity to collect at least 100 points during the semester (plus a lot more bonus points)
- You must collect at least 50 points to pass the exercises and be allowed to participate in the oral exam.
- Extra homework points improve your grade:
  - $\geq$ 70 points improves grade by 0.3
  - $\geq$ 90 points improves grade by 0.7
  - $\geq$ 110 points improves grade by 1.0

Lecture Outline

- Planning as Satisfiability
- Random Formulas

What is Planning

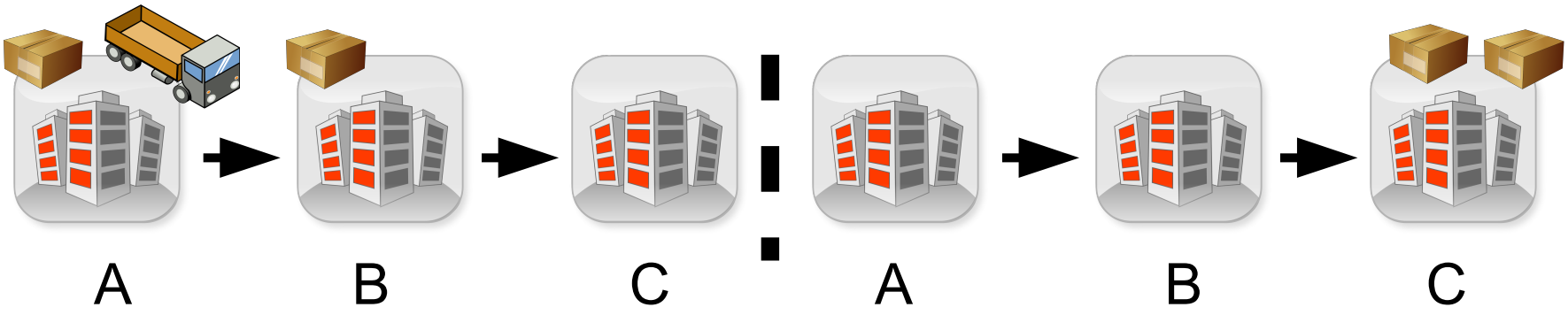
Informal Definition

Planning is the process of finding a plan, i.e., a sequence of actions that changes the state of the world from some initial state to a desired (goal) state.

Examples

- Delivering some packages
- Building a submarine
- Robot motion planning
- Fulfilling a scientific goal by an autonomous space probe

Trucking Example



- Initial State
  - There is a truck and a package in city A
  - There is a package in city B
- Goal
  - There are two packages in city C
- Possible Actions
  - (Un)loading packages from/on the truck, driving between cities

Formalizing Planning

Planning Problem Definition

A planning problem instance is Π is a tuple $(\mathcal{X}, \mathcal{A}, s_I, s_G)$ where

- $\mathcal{X}$ is a set of multivalued variables with finite domains.
  - each variable $x \in \mathcal{X}$ has a finite possible set of values $dom(x)$
- $\mathcal{A}$ is a set actions. Each action $a \in \mathcal{A}$ is a tuple $(pre(a), eff(a))$
  - $pre(a)$ is a set of preconditions of action a
  - $eff(a)$ is a set of effects of action a
  - both are sets of equalities of the form $x = v$ where $x \in \mathcal{X}$ and $v \in dom(x)$
- s_I is the initial state, it is a **full** assignment of the variables in $\mathcal{X}$
- s_G is the set of goal conditions, it is a set of equalities (same as $pre(a)$ and $eff(a)$)

Formalizing Planning II

World State

A state is full assignment of the variables in $\mathcal{X}$ (each variable $x \in \mathcal{X}$ has exactly one value assigned from its domain $dom(x)$. A state can be represented as a set of equalities.

The initial state s_I is a state. A state s is a goal state if $s_G \subseteq s$

Applicable Actions

An action $a \in \mathcal{A}$ is applicable in the state s if $pre(a) \subseteq s$

Applying an Action

When an action $a \in \mathcal{A}$ is applied in the state s it changes to the state s' such that $eff(a) \subseteq s'$ and the difference between s and s' is minimal (only variables used in $eff(a)$ are changed).

Formalizing Planning III

A Plan

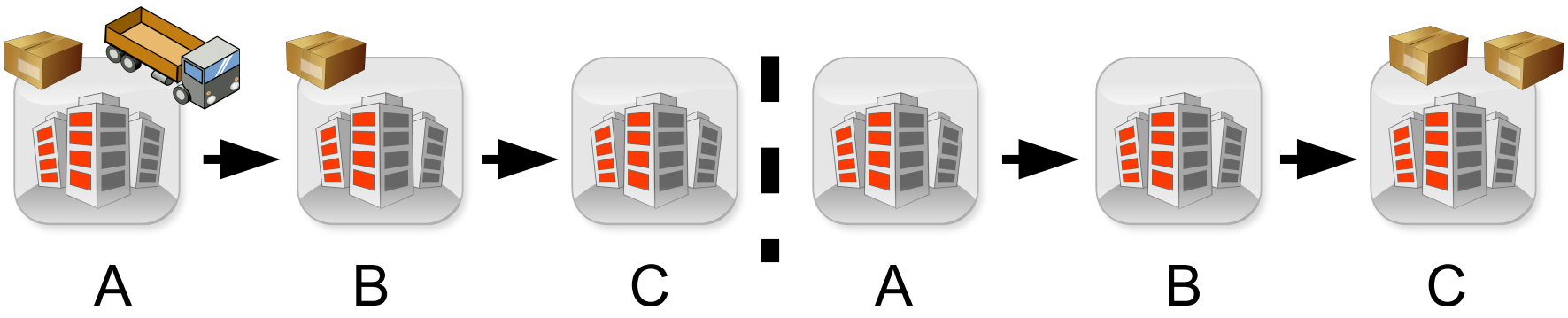
A plan for P for a planning problem $\Pi = (\mathcal{X}, \mathcal{A}, s_I, s_G)$ is sequence of actions $a_1, a_2, \ldots a_n$ such that

- $\forall i \; a_i \in \mathcal{A}$
- let $s_1 = s_I$ and $s_{i+1} = \mathit{apply}(s_i, a_i)$
- a_i is applicable in s_i
- $s_G \subseteq s_{n+1}$

If $P = \{a_1, a_2, \ldots a_n\}$ then n is the lenght of the plan P.

An optimal plan is a plan of shortest length.

Trucking Example



- variables: Truck Location T, $dom(T) = \{A, B, C\}$, Package Locations P_1 and P_2, $dom(P_1) = dom(P_2) = \{A, B, C, T\}$
- Initial state: $\{T = A, P_1 = A, P_2 = B\}$
- Goal: $\{P_1 = C, P_2 = C\}$
- Actions: $load(P_i, L) = (\{T = L, P_i = L\}, \{P_i = T\})$
  $unload(P_i, L) = (\{T = L, P_i = T\}, \{P_i = L\})$
  $drive(L_1, L_2) = (\{T = L_1\}, \{T = L_2\})$ where $i \in \{1, 2\}$ and $L, L_1, L_2 \in \{A, B, C\}$

Trucking Example

World State

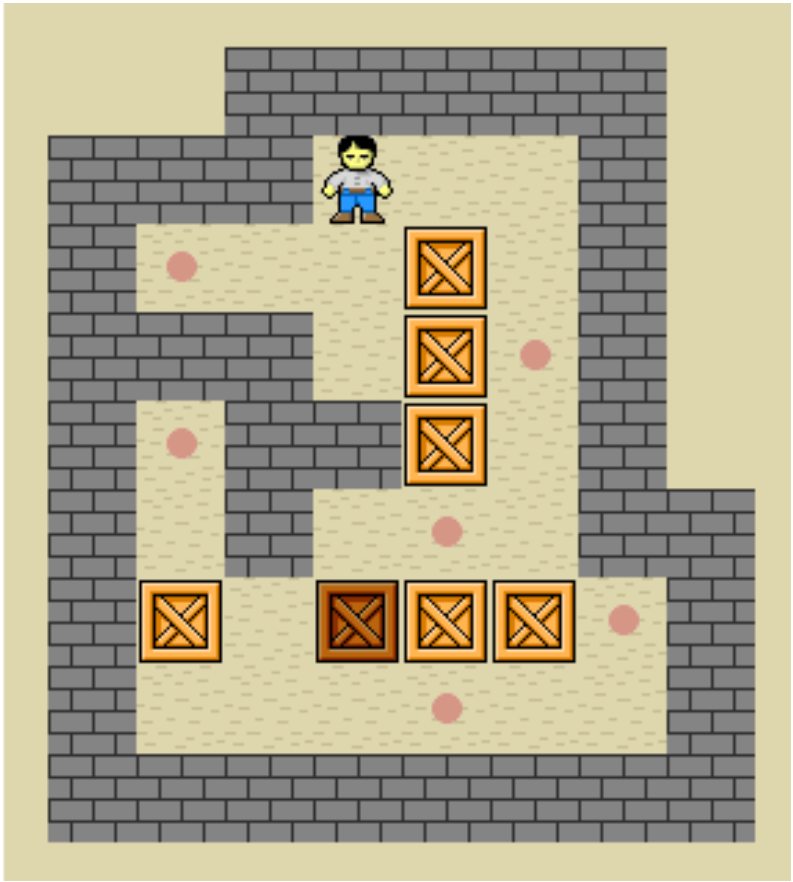
- $T = A, P_1 = A, P_2 = B$
- $T = A, P_1 = T, P_2 = B$
- $T = B, P_1 = T, P_2 = B$
- $T = B, P_1 = T, P_2 = T$
- $T = C, P_1 = T, P_2 = T$
- $T = C, P_1 = C, P_2 = C$

The Plan

- $load(P_1, A)$
- $drive(A, B)$
- $load(P_2, B)$
- $drive(B, C)$
- $unload(P_1, C)$, $unload(P_1, C)$

Sokoban Example

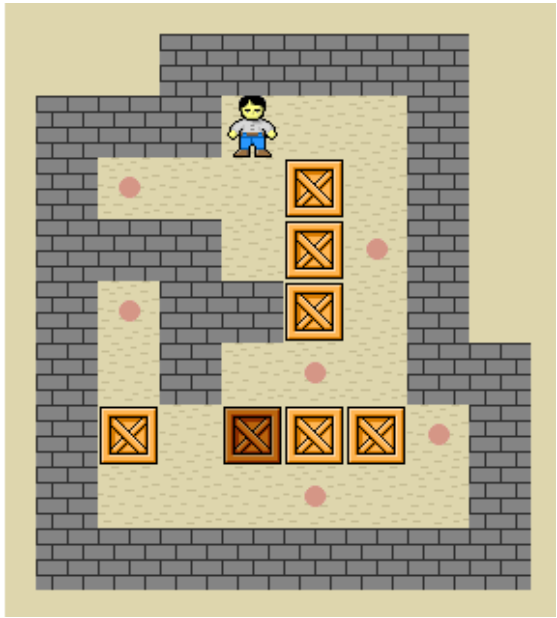
- Initial State
  - There is a worker and a bunch of boxes
- Goal
  - All the boxes must in goal positions
- Possible Actions
  - moving with the worker
  - pushing boxes
- Forbidden
  - to pull boxes
  - move throuhg walls or boxes



http://wki.pe/Sokoban

Encoding Sokoban

- Variables – For each location we have variable, the domain is WORKER, BOX, EMPTY
- Initial State – assign values based on the picture
- Goal – goal position variables have value BOX



  - Actions – move and push for each possible location
  - $push(L_1, L_2, L_3) = (\{L_1 = W, L_2 = B, L_3 = E\}, \{L_1 = E, L_2 = W, L_3 = B\})$
  - $move(L_1, L_2) = (\{L_1 = W, L_2 = E\}, \{L_1 = E, L_2 = W\})$

Encoding Planning into CNF

Is that even possible?

Encoding Planning into CNF

- We cannot encode the existence of a plan in general
- But we can encode the existence of plan up to some length

Encoding Planning into CNF

- We cannot encode the existence of a plan in general
- But we can encode the existence of plan up to some length

SATPLAN Algorithm

- INPUT: a planning problem Π
- OUTPUT: a plan P

for $m := 1, 2, \ldots$ **do**
  $F =$ encodePlanExists(Π, m)
  if solver.isSat(F) **then**
    return extractPlan(Π, m, solver.solution)

Encoding Planning into CNF

The Task

Given a planning problem instance $\Pi = (\mathcal{X}, \mathcal{A}, s_I, s_G)$ and $k \in \mathbb{N}$ construct a CNF formula F such that F is satisfiable if and only if there is plan of length k for Π.

Encoding Planning into CNF

The Task

Given a planning problem instance $\Pi = (\mathcal{X}, \mathcal{A}, s_I, s_G)$ and $k \in \mathbb{N}$ construct a CNF formula F such that F is satisfiable if and only if there is plan of length k for Π.

We will need two kinds of variables

- Variables to encode the actions:
  a_i^t for each $t \in \{1, \ldots, k\}$ and $a_i \in \mathcal{A}$
- Variables to encode the states:
  $b_{x=v}^t$ for each $t \in \{1, \ldots, k+1\}$, $x \in \mathcal{X}$ and $v \in dom(x)$

In total we have $k|\mathcal{A}| + (k+1)\sum_{x \in \mathcal{X}} dom(x)$ variables

Encoding Planning into CNF

We will need six kinds of clauses

- The first state is the initial state
- The goal conditions are satisfied in the end
- Each state variable has at least one value
- Each state variable has at most one value
- If an action is applied it must be applicable
- If an action is applied its effects are applied in the next step
- State variables cannot change without an action between steps
- At most one action is used in each step

Encoding Planning into CNF

The first state is the initial state

$$\begin{gathered}(b^1_{x=v}) \\ \forall(x = v) \in s_I\end{gathered} \tag{1}$$

The goal conditions are satisfied in the end

$$\begin{gathered}(b^{n+1}_{x=v}) \\ \forall(x = v) \in s_G\end{gathered} \tag{2}$$

Encoding Planning into CNF

Each state variable has at least one value

$$\begin{gathered}(b^t_{x=v_1} \vee b^t_{x=v_2} \vee \cdots \vee b^t_{x=v_d}) \\ \forall x \in X,\ \mathrm{dom}(x) = \{v_1, v_2, \ldots, v_d\},\ \forall t \in \{1, \ldots, k+1\}\end{gathered} \tag{3}$$

Each state variable has at most one value

$$\begin{gathered}(\neg b^t_{x=v_i} \vee \neg b^t_{x=v_j}) \\ \forall x \in X,\ v_i \neq v_j, \{v_i, v_j\} \subseteq \mathrm{dom}(x),\ \forall t \in \{1, \ldots, k+1\}\end{gathered} \tag{4}$$

Encoding Planning into CNF

If an action is applied it must be applicable

$$(\neg a^t \vee b^t_{x=v}) \quad \forall a \in \mathcal{A},\ \forall (x = v) \in \text{pre}(a),\ \forall t \in \{1, \ldots, k\} \tag{5}$$

If an action is applied its effects are applied in the next step

$$(\neg a^t \vee b^{t+1}_{x=v}) \quad \forall a \in \mathcal{A},\ \forall (x = v) \in \text{eff}(a),\ \forall t \in \{1, \ldots, k\} \tag{6}$$

Encoding Planning into CNF

State variables cannot change without an action between steps

$$(\neg b_{x=v}^{t+1} \vee b_{x=v}^{t} \vee a_{s_1}^{t} \vee \cdots \vee a_{s_j}^{t})$$

$$\forall x \in X,\ \forall v \in \mathrm{dom}(x),\ \mathrm{support}(x = v) = \{a_{s_1}, \ldots, a_{s_j}\},\ \forall t \in \{1, \ldots, k\} \tag{7}$$

By $\mathrm{support}(x = v) \subseteq \mathcal{A}$ we mean the set of *supporting actions* of the assignment $x = v$, i.e., the set of actions that have $x = v$ as one of their effects.

Encoding Planning into CNF

At most one action is used in each step

$$\begin{gathered}(\neg a_i^t \vee \neg a_j^t) \\ \forall \{a_i, a_j\} \subseteq \mathcal{A},\ a_i \neq a_j\ \forall t \in \{1, \ldots, k\}\end{gathered} \tag{8}$$

Encoding Planning into CNF

The Task Solved

Given a planning problem instance $\Pi = (\mathcal{X}, \mathcal{A}, s_I, s_G)$ and $k \in \mathbb{N}$ a CNF formula F, which is a conjunction of all the above described clauses is satisfiable if and only if there is plan of length k for Π.

Optimizations

- Better encoding of at-most-one
- Allowing several actions in each step
- Encoding variable transitions instead of variable values

Random Formulas

Goldbergs Random Formulas – 1979

Generate m claues on n variables such that for each clause c each variable x_i is in c as positive literal with probability $\frac{1}{3}$, as a negative literal with probablity $\frac{1}{3}$ and not at all with probability $\frac{1}{3}$.

Random Formulas

Goldbergs Random Formulas – 1979

Generate m claues on n variables such that for each clause c each variable x_i is in c as positive literal with probability $\frac{1}{3}$, as a negative literal with probablity $\frac{1}{3}$ and not at all with probability $\frac{1}{3}$.

- Goldberg showed that these instances can be solved in polynomial time (using the DPLL algorithm) and therefore ”SAT is easy”.

Random Formulas

Goldbergs Random Formulas – 1979

Generate m claues on n variables such that for each clause c each variable x_i is in c as positive literal with probability $\frac{1}{3}$, as a negative literal with probablity $\frac{1}{3}$ and not at all with probability $\frac{1}{3}$.

- Goldberg showed that these instances can be solved in polynomial time (using the DPLL algorithm) and therefore ”SAT is easy”.
- in 1983 Franco and Paull pointed out, that these instances are so easy that just quessing assignemts solves it in $\mathcal{O}(1)$ expected time

Random k-SAT Formulas

Uniform Random k-SAT – 1983 Franco and Paull

Uniform random k-SAT formulas are formed by selecting uniformly and independently m clauses of length k from the set of all $2^k \binom{n}{k}$ clauses on a given set of n variables.

Random k-SAT Formulas

Uniform Random k-SAT – 1983 Franco and Paull

Uniform random k-SAT formulas are formed by selecting uniformly and independently m clauses of length k from the set of all $2^k \binom{n}{k}$ clauses on a given set of n variables.

Phase Transition Hypothesis

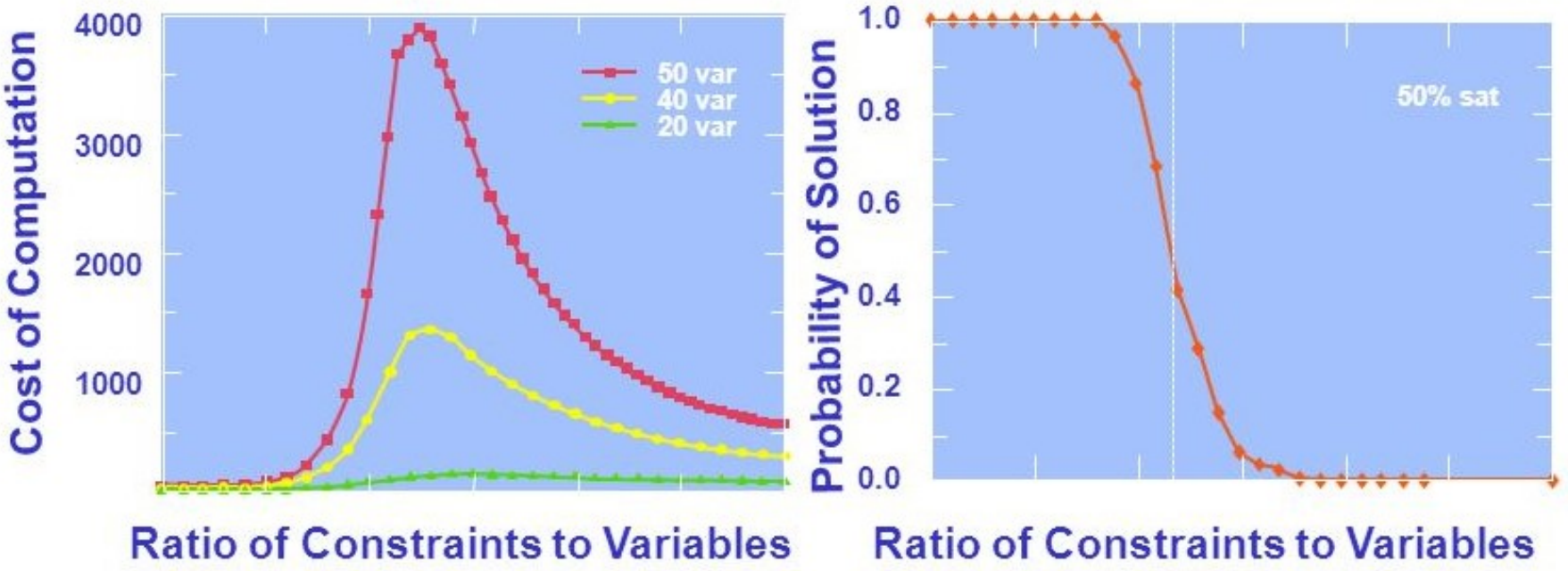
Let $\alpha = \frac{m}{n}$ be the ratio of clauses and variables of a k-SAT formula. For each k there exists a constant α_k such that

- A random k-SAT formula with $\alpha < \alpha_k$ is satisfiable w.h.p.
- A random k-SAT formula with $\alpha > \alpha_k$ is unsatisfiable w.h.p.

w.h.p. means ”with high probability”, the hypothesis is proven only for 2-SAT where $\alpha_2 = 1$.

Phase Transition – 3-SAT

- Selman *et al.* (1994):



- From experiments we deduce that α_3 is around 4.25
- Myth: phase transition is where the hardest formulas are

Satisfiable random formulas

- How to generate a random satisfiable formula?

Satisfiable random formulas

- How to generate a random satisfiable formula?
- Hidden Solution Algorithm
  - Take a truth assignment ϕ
  - Generate random clauses but only add those that satisfy ϕ

Satisfiable random formulas

- How to generate a random satisfiable formula?
- Hidden Solution Algorithm
  - Take a truth assignment ϕ
  - Generate random clauses but only add those that satisfy ϕ
- Two Hidden Algorithm
  - Same as previous but with 2 assignments ϕ and $\overline{\phi}$

Satisfiable random formulas

- How to generate a random satisfiable formula?
- Hidden Solution Algorithm
  - Take a truth assignment ϕ
  - Generate random clauses but only add those that satisfy ϕ
- Two Hidden Algorithm
  - Same as previous but with 2 assignments ϕ and $\overline{\phi}$
- Clause Distribution Control

Clause Distribution Control Algorithm

CDC Algorithm

- INPUT: a truth assignment ϕ on n variables
- OUTPUT: a random k-SAT CNF satisfied by ϕ

$F = \emptyset$
while $|F| < r * n$ **do**
    $C =$ a random k-clause on n variables
    $s =$ number of literals in C satisfied by ϕ
    $r =$ a random real number $(0, 1)$
    if $s > 0$ and $p_s > r$ **then**
        $F = F \cup \{C\}$
return F

$r, p_1, p_2, \ldots, p_k$ are parameters of the CDC algorithm

Choosing CDC parameters p_i and r?

- Barthel et. al algorithm
  - $0.077 < p_1 < 0.25$
  - $p_2 = (1 - 4p_1)/6$
  - $p_3 = (1 + 2p_1)/6$
  - $r > 4.25$
- Q-Hidden algorithm
  - $p_i = q^i$ for a parameter q
  - r not specified
- Automatic configuration based algorithm
  - use and automatic configuration tool and a solver to optimize p_i and r against

Are you touched by SAT?