Probability in the analysis of CNF Satisfiability algorithms and properties

Outline:

1. Why is it worthwhile?
2. Some problems with its use
3. Digest of types of results
4. Some handy tools of analysis
5. Using the tools to obtain results

Why is it worthwhile?:

The questions:

Why are some problems so difficult?

Can algorithms be developed to make them easier?

Why is it worthwhile?:

The questions:

- Why are some problems so difficult?
- Can algorithms be developed to make them easier?

Why probability?

- Results and process tend to draw out intuition
- Can explain good or bad algorithm behaviour
- Affords comparison of otherwise incomparable classes of SAT

Preliminaries:

Only talking about CNF

A *variable* looks like this: v_9, takes a *value* from the set $\{0,1\}$

Preliminaries:

Only talking about CNF

A *variable* looks like this: v_9, takes a *value* from the set $\{0,1\}$

A *positive literal* looks like a variable, a *negated literal*: $\neg v_9$

Preliminaries:

Only talking about CNF

A *variable* looks like this: v_9, takes a *value* from the set $\{0,1\}$

A *positive literal* looks like a variable, a *negated literal*: $\neg v_9$

A *clause* is a set of literals, looking like this: $\{ \neg v_0, \neg v_2, v_5, v_9\}$

Preliminaries:

Only talking about CNF

A *variable* looks like this: v_9, takes a *value* from the set $\{0,1\}$

A *positive literal* looks like a variable, a *negated literal*: $\neg v_9$

A *clause* is a set of literals, looking like this: $\{ \neg v_0, \neg v_2, v_5, v_9\}$

The *width* of a clause is the number of literals it contains

Preliminaries:

Only talking about CNF

A *variable* looks like this: v_9, takes a *value* from the set $\{0,1\}$

A *positive literal* looks like a variable, a *negated literal*: $\neg v_9$

A *clause* is a set of literals, looking like this: $\{ \neg v_0, \neg v_2, v_5, v_9\}$

The *width* of a clause is the number of literals it contains

An *instance* of SAT is a set of clauses, looking like this:

$\{ v_0, \neg v_1, v_6 \}, \{ \neg v_1, v_5 \}, \{ \neg v_1, \neg v_3, \neg v_4 \}, \{ \neg v_0, \neg v_2, v_5, v_9\}, \{ v_9\}$

Preliminaries:

Only talking about CNF

A *variable* looks like this: v_9, takes a *value* from the set $\{0,1\}$

A *positive literal* looks like a variable, a *negated literal*: $\neg v_9$

A *clause* is a set of literals, looking like this: $\{ \neg v_0, \neg v_2, v_5, v_9\}$

The *width* of a clause is the number of literals it contains

An *instance* of SAT is a set of clauses, looking like this:

$\{ v_0, \neg v_1, v_6 \}, \{ \neg v_1, v_5 \}, \{ \neg v_1, \neg v_3, \neg v_4 \}, \{ \neg v_0, \neg v_2, v_5, v_9\}, \{ v_9\}$

Use φ to represent an instance of SAT

An assignment of values satisfying φ is a *model* for φ

Preliminaries:

Only talking about CNF

A *variable* looks like this: v_9, takes a *value* from the set $\{0,1\}$

A *positive literal* looks like a variable, a *negated literal*: $\neg v_9$

A *clause* is a set of literals, looking like this: $\{ \neg v_0, \neg v_2, v_5, v_9\}$

The *width* of a clause is the number of literals it contains

An *instance* of SAT is a set of clauses, looking like this:

$\{ v_0, \neg v_1, v_6 \}, \{ \neg v_1, v_5 \}, \{ \neg v_1, \neg v_3, \neg v_4 \}, \{ \neg v_0, \neg v_2, v_5, v_9\}, \{ v_9\}$

Use φ to represent an instance of SAT

An assignment of values satisfying φ is a *model* for φ

Many algorithms (DPLL variants) use this update operation:

$\varphi = \{ c - \{\neg v\} : c \in \varphi , v \notin c \}$ (set $v = 1$)

$\varphi = \{ c - \{v\} : c \in \varphi , \neg v \notin c \}$ (set $v = 0$)

Some problems:

1. Must assume an input distribution, often not reflecting reality

Some problems:

1. Must assume an input distribution, often not reflecting reality

2. Analysis can be difficult or impossible: algorithmic steps may change distribution significantly (as yet, tools are limited)

Some problems:

1. Must assume an input distribution, often not reflecting reality

2. Analysis can be difficult or impossible: algorithmic steps may change distribution significantly (as yet, tools are limited)

3. Can yield misleading results

Example: A probabilistic model for generating random formulas

Given a set of literals: $L = \{ v_1 , \neg v_1, \ldots , v_n , \neg v_n \}$ and $0 \leq p \leq 1$

Construct a clause c: for all $l \in L$ add v_l , $\neg v_l$ to c , each with probability p

Construct a formula: a collection of m clauses constructed independently

Example: A probabilistic model for generating random formulas

Given a set of literals: $L=\{ v_1 , \neg v_1, \dots , v_n , \neg v_n \}$ and $0 \leq p \leq 1$

Construct a clause c: for all $l \in L$ add v_l , $\neg v_l$ to c , each with probability p

Construct a formula: a collection of m clauses constructed independently

Justification: if $p=1/3$ all formulas are equally likely (unbiased in a way)

Example: A probabilistic model for generating random formulas

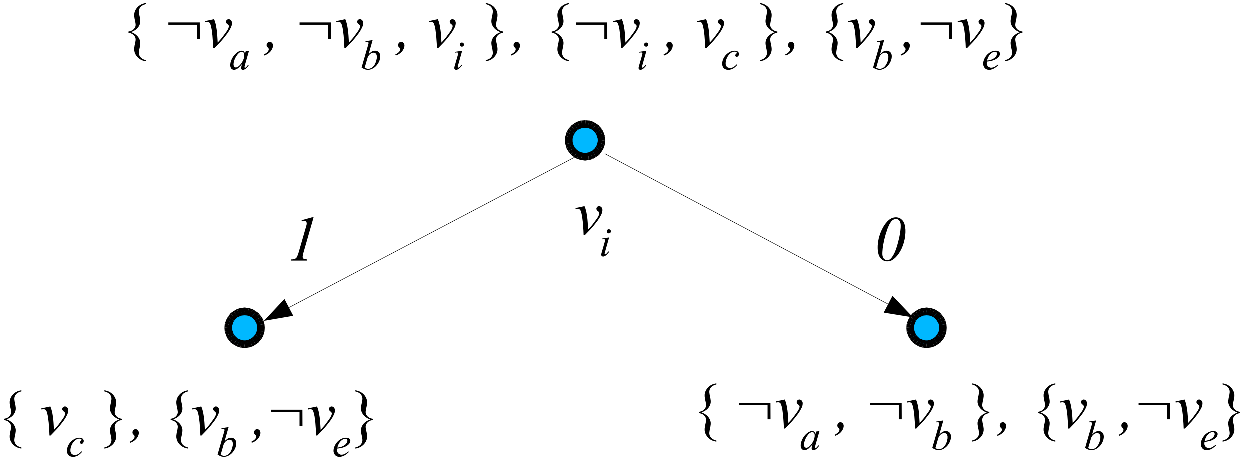
Given a set of literals: $L = \{ v_1, \neg v_1, \ldots, v_n, \neg v_n \}$ and $0 \leq p \leq 1$

Construct a clause c: for all $l \in L$ add v_l, $\neg v_l$ to c, each with probability p

Construct a formula: a collection of m clauses constructed independently

Justification: if $p=1/3$ all formulas are equally likely (unbiased in a way)

Applied to Davis-Putnam variant:



$\{ \neg v_a, \neg v_b, v_i \}, \{ \neg v_i, v_c \}, \{ v_b, \neg v_e \}$

$\{ v_c \}, \{ v_b, \neg v_e \}$

$\{ \neg v_a, \neg v_b \}, \{ v_b, \neg v_e \}$

Analysis sketch:

Given m' clauses. Average number of clauses removed when a value is assigned = $p \cdot m'$

Let $T(i)$ be the average number of clauses remaining on the ith iteration

Then $T(0) = m$

$$T(i) = (1-p)T(i-1)$$

For what i does this number hit 1?

Analysis sketch:

Given m' clauses. Average number of clauses removed when a value is assigned $= p \cdot m'$

Let $T(i)$ be the average number of clauses remaining on the ith iteration

Then $T(0) = m$

$$T(i) = (1-p)T(i-1)$$

For what i does this number hit 1?

When $1 = m(1-p)^i$

or $lg(m) = -i \cdot lg(1-p)$

$$i = lg(m) / -lg(1-p)$$

So, size of search space is $2^i \approx 2^{lg(m)/p} = m^c$, c constant if $p=1/3$

Analysis sketch:

Given m' clauses. Average number of clauses removed when a value is assigned = $p \cdot m'$

Let $T(i)$ be the average number of clauses remaining on the ith iteration

Then $T(0) = m$

$T(i) = (1-p)T(i-1)$

For what i does this number hit 1?

When $1 = m(1-p)^i$

or $lg(m) = -i \cdot lg(1-p)$

$i = lg(m) / -lg(1-p)$

So, size of search space is $2^i \approx 2^{lg(m)/p} = m^c$, c constant if $p=1/3$

Conclusion: *Davis-Putnam is fantastic, on the average!*

Problems with the analysis:

1. The input model is funky!
   The probability that a random assignment satisfies a random formula is

$$(1 - (1 - p)^{2n})^{m} \approx e^{-me^{-2pn}}$$

which tends to 1 if $ln(m) / p \cdot n = o(1)$ which means the average width of a clause can be at least $ln(m)$. With $p=1/3$, holds if $n > ln(m)$.

Problems with the analysis:

1. The input model is funky!
   The probability that a random assignment satisfies a random formula is

$$(1-(1-p)^{2n})^{m} \approx e^{-me^{-2pn}}$$

   which tends to 1 if $ln(m) / p \cdot n = o(1)$ which means the average width of a clause can be at least $ln(m)$. With $p=1/3$, holds if $n > ln(m)$.

2. If average clause width is constant (i.e. $p = c/n$, c a constant) then average search space size is

$$2^{-lg(m)/lg(1-p)} \approx 2^{lg(m)/(c'/n)} = 2^{n \cdot lg(m)/c'}$$

   Exponential!

Types of results: Random instance has...

1. some exploitable structural property with high probability

... under some conditions

Types of results: Random instance has...

1. some exploitable structural property with high probability

2. some intractable property for some algorithm w.h.p.

... under some conditions

Types of results: Random instance has...

1. some exploitable structural property with high probability

2. some intractable property for some algorithm w.h.p.

3. some sharp or coarse phase transition property

... under some conditions

Types of results: Random instance has...

1. some exploitable structural property with high probability

2. some intractable property for some algorithm w.h.p.

3. some sharp or coarse phase transition property

4. 1st (discontinuous) or 2nd (continuous) order phase trans.

... under some conditions

Types of results: Random instance has...

1. some exploitable structural property with high probability

2. some intractable property for some algorithm w.h.p.

3. some sharp or coarse phase transition property

4. 1st (discontinuous) or 2nd (continuous) order phase trans.

5. a high probability of being solved by some algorithm

... under some conditions

Types of results: Random instance has...

1. some exploitable structural property with high probability
2. some intractable property for some algorithm w.h.p.
3. some sharp or coarse phase transition property
4. 1st (discontinuous) or 2nd (continuous) order phase trans.
5. a high probability of being solved by some algorithm
6. a much greater chance of being in one class as opposed to another class

... under some conditions

Input Distributions: (CNF)

Constant density: $(G^{p}_{m,n})$

Given a set of literals: $L=\{ v_1 , \neg v_1, \dots , v_n , \neg v_n \}, 0 \leq p \leq 1$

Construct a clause c: for all $l \in L$ add v_l , $\neg v_l$ to c , each with probability p

Construct a formula: a collection of m clauses constructed independently

Random formula φ: $\varphi \in G^{p}_{m,n}$

Input Distributions: (CNF)

Constant density: $(G^{p}_{m,n})$

Given a set of literals: $L=\{ v_1 , \neg v_1, \dots , v_n , \neg v_n \}, 0\leq p\leq 1$

Construct a clause c: for all $l \in L$ add v_l , $\neg v_l$ to c , each with probability p

Construct a formula: a collection of m clauses constructed independently

Constant width: $(M^{k}_{m,n})$

Given a set of variables: $V=\{ v_1 , v_2, \dots , v_n \}$, integer k

Let $H = \{ c \subseteq V : |c| = k \}$

Construct a clause c: choose c uniformly and independently from H,

for all $v \in c$, change v to $\neg v$ independently with probability $1/2$.

Construct a formula: a collection of m clauses constructed independently

Random formula φ: $\varphi \in M^{k}_{m,n}$ or $\varphi \in G^{p}_{m,n}$

Tools:

1. First moment method
2. Second moment method
3. Flow analysis (differential equations)
4. Freidgut, sharp phase transitions
5. Eigenvalues
6. Martingales

First Moment Method (used to show $Pr\ (P) \rightarrow 0$ as $n \rightarrow \infty$):

X: positive integer-valued random variable with distribution $p_x(t)$

$$Pr\ (X \geq 1) = \int_{t=1}^{\infty} p_x(t)dt < \int_{t=1}^{\infty} t \cdot p_x(t)dt < E\{X\}$$

Let X be a count of some entity and suppose some property P holds if and only if $X \geq 1$.

Then $Pr\ (P) < E\{X\}$

So, if $E\{X\} \rightarrow 0$ as $n \rightarrow \infty$, then $Pr(P) \rightarrow 0$ as $n \rightarrow \infty$.

Example:

Let $\varphi \in M^k_{m,n}$

Let P be the property that φ has a model.

Show $Pr\ (P) \to 0$ if $m/n > -ln(2)\ /\ ln(1\text{-}2^{-k})$

Example:

Let $\varphi \in M^{k}_{m,n}$

Let P be the property that φ has a model.

Show $Pr\ (P) \rightarrow 0$ if $m/n > -ln(2)\ /\ ln(1\text{-}2^{-k})$

Let X be the number of models for φ.

$$\text{Let } X_i = \begin{cases} 1 \text{ if } i^{th} \text{ assignment is a model for } \varphi \\ 0 \text{ if } i^{th} \text{ assignment is not a model for } \varphi \end{cases}$$

$Pr\ (X_i = 1) = ((2^k\text{-}1)\ /\ 2^k)^m$

$X = \sum_i X_i$, so $E\{X\} = 2^n E\{X_i\} = 2^n(1 - 2^{-k})^m$

$Pr\ (\exists\ a\ model\ for\ \varphi)\ <\ 2^n(1 - 2^{-k})^m$

$Pr\ (\varphi\ has\ a\ model) \rightarrow 0$ if $m/n > -ln(2)\ /\ ln(1\text{-}2^{-k})$

For $k=3$ φ has no model w.h.p. when $m/n > 5.19$

Second Moment Method (used to show $Pr\,(P) \to 1$ as $n \to \infty$):

Let $\varphi \in M^{k}_{m,n}$

Let P be some property

Let a "witness" be a structure whose presence in φ implies property P holds

Let $W = \{\, w : w \textit{ is a witness for } P\}$.

Suppose $Pr\,(w) \equiv q$ is independent of w for all witnesses.

$$\text{Let } X_w = \begin{cases} 1 & \text{if } w \subseteq \varphi \text{ and is a witness for } P \\ 0 & \text{otherwise.} \end{cases}$$

Then $E\{X_w\} = q$ and $var(X_w) = E\{(X_w - q)^2\} = q(1 - q)$.

Define $X = \sum_{w \in W} X_w$ and let $\mu = E\{X\} = q|W|$ and $\sigma^2 = var(X)$.

Consider Chebyshev's inequality: $Pr\,(X = 0) \leqslant \sigma^2/\mu^2$.

If events $w \subseteq \varphi$ are independent, then $\sigma^2 = q|W|(1\text{-}q) = O(\mu)$

But this is not usually the case!

Show that if independence does not hold, it is weak enough that

$$\sigma^2 = o(\mu^2)$$

Start with one witness *w* chosen arbitrarily
Let *A(w)* be all witnesses other than *w* sharing at least one clause with *w*
Let *D(w)* be all witnesses having no clause in common with *w*

$$\sigma^2 \;=\; \mu(1\text{-}q) + \mu\left(\sum_{z\in A(w)}\Big(Pr(z|w) - q\Big) + \sum_{z\in D(w)}\Big(Pr(z|w) - q\Big)\right)$$

Need $|A(w)| \ll |D(w)|$ or little "overlap" among witnesses

If $\mu \rightarrow \infty$ as $n \rightarrow \infty$ and $\sum_{z\in A(w)} Pr\,(z|w) = \mathrm{o}(\mu)$ for arbitrary *w*
Then $Pr\,(P) \rightarrow 1$ as $n \rightarrow \infty$.

2nd Moment Method Example:

Let $\varphi \in M^k_{m,n}$

Let P be the property that φ has no model.

Let a *witness* be a model for φ.

Let X be the number of models for φ.

Let $X_w = \begin{cases} 1 \text{ if witness } w \text{ satisfies } \varphi \\ 0 \text{ if witness } w \text{ does not satisfy } \varphi \end{cases}$

$X = \sum_w X_w$, so $\sigma^2 = (m^2 / n)\ \Theta(\mu^2)$

But if $m/n > 1$ the variance is too high and 2nd moment method cannot be used.

2^{nd} Moment Method Example:

Let $\varphi \in M^{k}_{m,n}$

Define an *equivalence cycle* to be a collection of clauses such as

$\{v_1, \neg v_2\}, \{v_2, \neg v_3\}, \ldots, \{v_x, v_1\}, \{\neg v_1, \neg v_{x+1}\}, \ldots, \{v_{2x-1}, \neg v_1\}$

and suppose $x = \Theta ln(n)$.

From each of a subset φ' of clauses of φ, remove all but two literals from each and suppose the result is an equivalence cycle.

Then φ' is a *witness* to the property that φ is not SLUR

Let X be the number of equivalence cycles in φ of length parm x.

$\sum_{z\in A(w)} Pr(z|w) < \sum_{1\leqslant q\leqslant x} |A_q(w)| Pr(z|w,q)$ (overlap of q clauses)

$= \left(\frac{x}{n} + \left(\frac{mk(k-1)}{2n}\right)^{-x}\right)\mu = o(\mu)$ if $m/n > 2/k(k-1)$, $k>2$

Flow Analysis:

In Davis-Putnam-Loveland-Logemann variants, when setting a variable some clauses disappear and some get their width reduced by one.

Let's consider straight-line (non-backtracking) variants.

```
Let $\varphi$ be a CNF Boolean expression
Repeat the following until $\varphi = \emptyset$ or $\emptyset \in \varphi$:
    Choose a literal $v$ or $\neg v$ occurring in $\varphi$
    $\varphi = \{ c - \{\neg v\} : c \in \varphi, v \notin c\}$  or
    $\varphi = \{ c - \{v\} : c \in \varphi, \neg v \notin c\}$
If $\varphi = \emptyset$ then report a solution is found otherwise give up
```

Under what conditions does this give up with probability tending to 0?

Answer depends on the way literals are chosen

Flow Analysis:

Let $\varphi \in M^{k}_{m,n}$

$C_i(j)$: The collection of clauses of width i remaining in φ after iteration j.

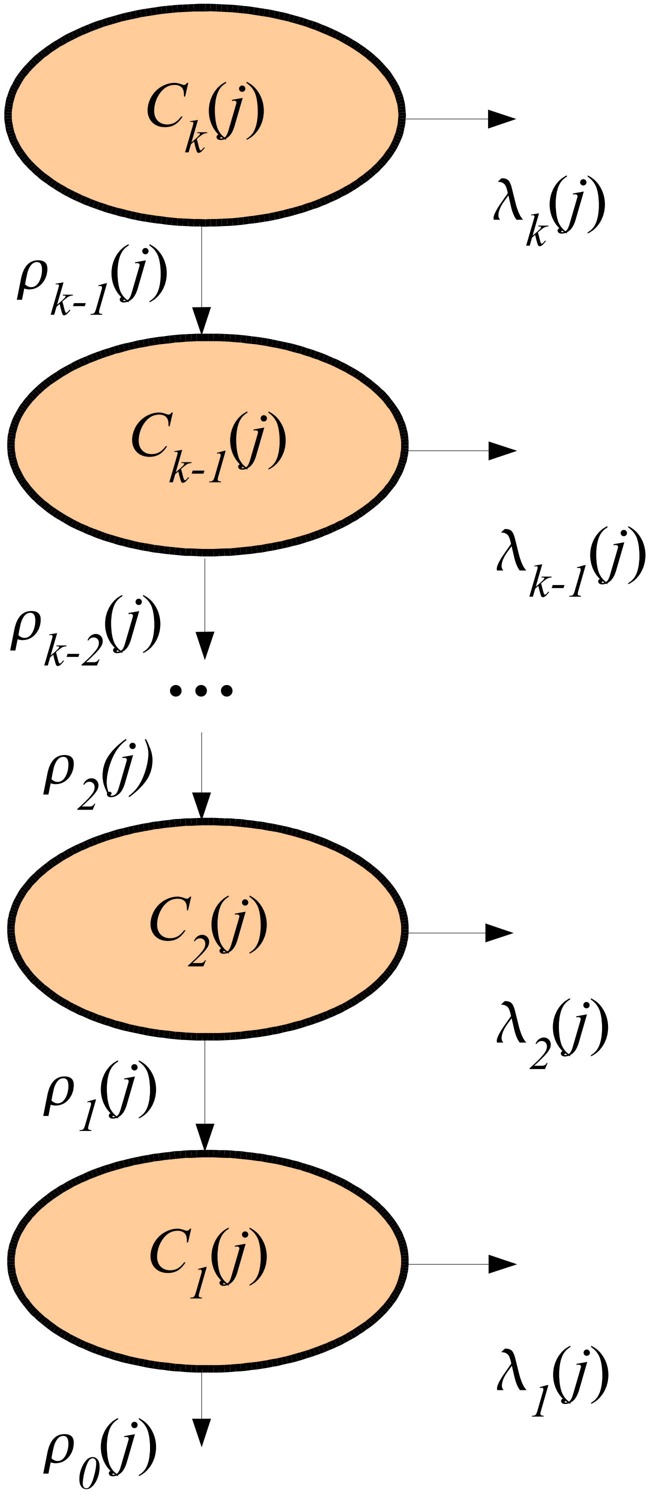
$C_k(0) = m$ and $\forall_{0\leq i<k}\, C_i(0) = 0$

Define $m_i(j) = |C_i(j)|$

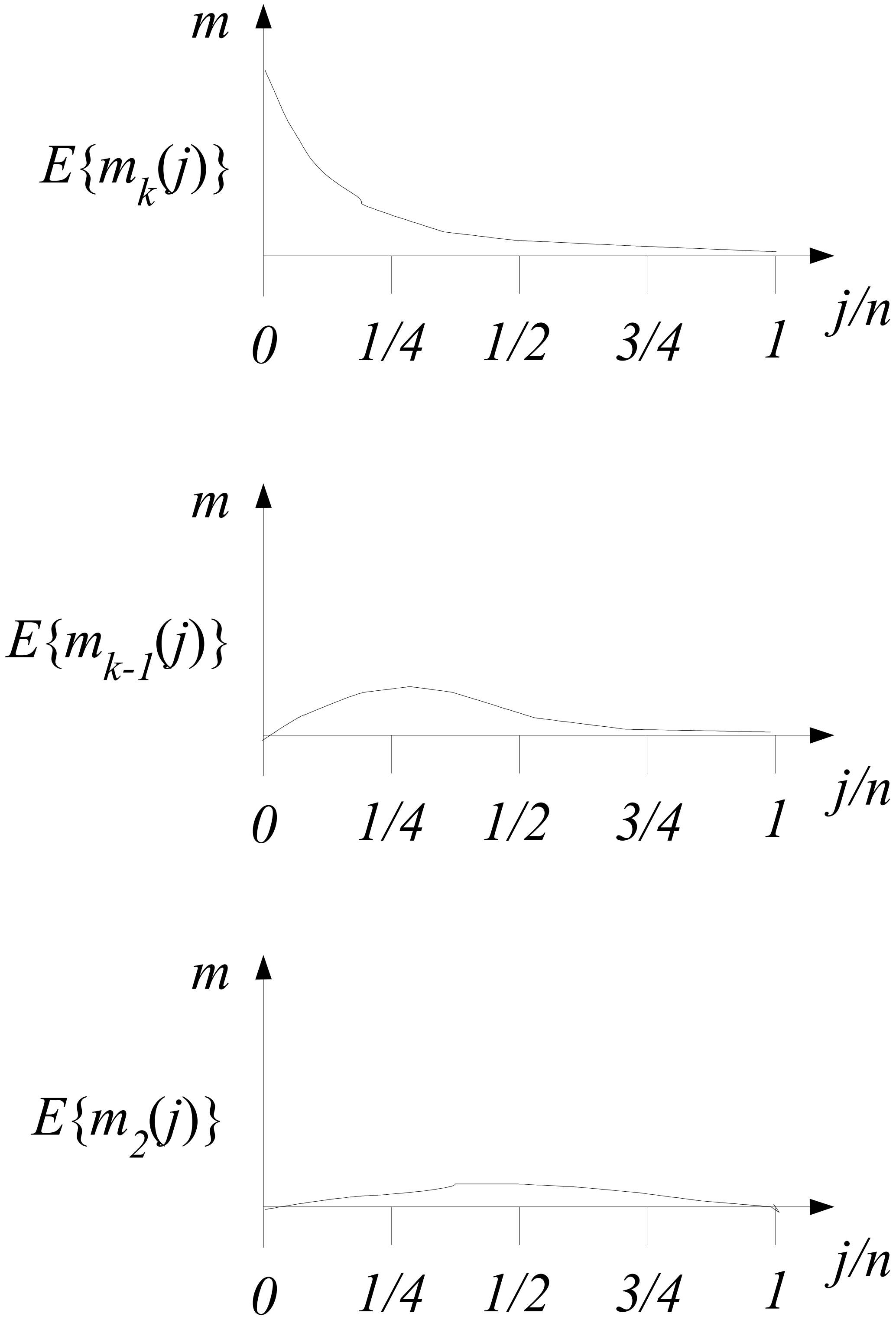
Define $\lambda_i(j)$: Flow of clauses becoming satisfied on iteration j.

Define $\rho_i(j)$: Flow of clauses reduced by one literal on iteration j.

The idea: design a heuristic that keeps the accumulation of clauses in $C_0(j)$ equal to 0 with high probability.



$C_k(j)$
$\lambda_k(j)$
$\rho_{k-1}(j)$
$C_{k-1}(j)$
$\lambda_{k-1}(j)$
$\rho_{k-2}(j)$
…
$\rho_2(j)$
$C_2(j)$
$\lambda_2(j)$
$\rho_1(j)$
$C_1(j)$
$\lambda_1(j)$
$\rho_0(j)$



m
$E\{m_k(j)\}$
0
1/4
1/2
3/4
1
j/n
m
$E\{m_{k-1}(j)\}$
0
1/4
1/2
3/4
1
j/n
m
$E\{m_2(j)\}$
0
1/4
1/2
3/4
1
j/n

Example:

Unit clause heuristic: when there is a clause with one unassigned variable remaining, set the value of such a variable so as to satisfy clause.

Intuitively: If the clause flow $\rho_1(j) < 1$ then any accumulation of unit clauses can be prevented and no clauses will ever be eliminated

Example:

Unit clause heuristic: when there is a clause with one unassigned variable remaining, set the value of such a variable so as to satisfy clause.

Intuitively: If the clause flow $\rho_1(j) < 1$ then any accumulation of unit clauses can be prevented and no clauses will ever be eliminated

Write difference equations describing flows:

$$m_i(j+1) = m_i(j) + \rho_i(j) - \lambda_i(j) \qquad \forall 1 \le i \le k,\ 0 < j < n$$

Example:

Unit clause heuristic: when there is a clause with one unassigned variable remaining, set the value of such a variable so as to satisfy clause.

Intuitively: If the clause flow $\rho_1(j) < 1$ then any accumulation of unit clauses can be prevented and no clauses will ever be eliminated

Write difference equations describing flows:

$$m_i(j+1) = m_i(j) + \rho_i(j) - \lambda_i(j) \qquad \forall 1 \leq i \leq k,\ 0 < j < n$$

Take expectations:

$$E\{m_i(j+1)\} = E\{m_i(j)\} + E\{\rho_i(j)\} - E\{\lambda_i(j)\} \qquad \forall 1 \leq i \leq k,\ 0 < j < n$$

Suppose statistical independence holds:

$\forall\, 1 < i \leq k,\ 0 < j < n$

$$E\{\lambda_i(j)\} = E\{E\{\lambda_i(j) \mid m_i(j)\}\} = E\left\{\frac{i \cdot m_i(j)}{n-j}\right\} = \frac{i \cdot E\{m_i(j)\}}{n-j}$$

$\forall\, 1 \leq i < k,\ 0 < j < n$

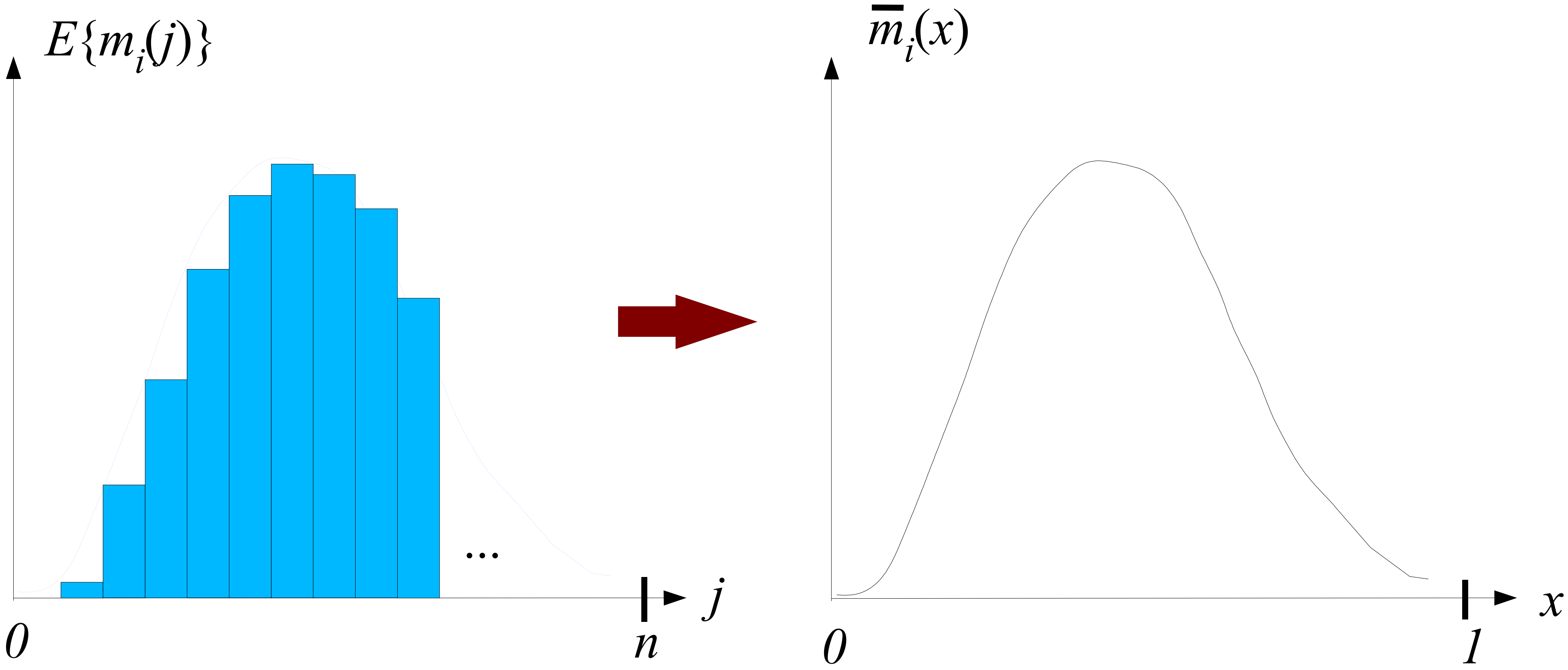
$$E\{\rho_i(j)\} = E\{E\{\rho_i(j) \mid m_{i+1}(j)\}\} = E\left\{\frac{(i+1) \cdot m_{i+1}(j)}{2(n-j)}\right\} = \frac{(i+1) \cdot E\{m_{i+1}(j)\}}{2(n-j)}$$

$$E\{\rho_k(j)\} = 0$$

Hence:

$$E\{m_i(j+1) - m_i(j)\} = \frac{(i+1) \cdot E\{m_{i+1}(j)\}}{2(n-j)} - \frac{i \cdot E\{m_i(j)\}}{n-j}$$

$$E\{m_k(j+1) - m_k(j)\} = -\frac{k \cdot E\{m_k(j)\}}{n-j}$$


$E\{m_i(j)\}$
...
0
n
j
$\overline{m}_i(x)$
0
1
x

Turning into differential equations: $(x = j/n)$

$$\frac{d\bar{m}_i(x)}{dx} = \frac{(i+1)\bar{m}_{i+1}(x)}{2(1-x)n} - \frac{i\cdot\bar{m}_i(x)}{(1-x)n} \qquad \forall\ 1 < i < k$$

$$\frac{d\bar{m}_k(x)}{dx} = - \frac{k\cdot\bar{m}_k(x)}{(1-x)n}$$

Boundary conditions:

$\bar{m}_k(0) = m/n,\ \ \bar{m}_i(0) = 0$ for $0 < i < k$

Solution:

$$\bar{m}_i(x) = \frac{1}{2^{k-i}}\binom{k}{i}(1-x)^i\,(x)^{k-i}\,m/n$$

Translating:

$$E\{m_i(x)\} = \frac{1}{2^{k-i}}\binom{k}{i}(1-j/n)^i\,(j/n)^{k-i}\,m$$

The important flow:

$$E\{\rho_1(j)\} = \frac{E\{m_2(j)\}}{n-j} = \frac{1}{2^{k-2}}\binom{k}{2}(1-j/n)^2\,(j/n)^{k-2}\,m$$

Take the derivative with respect to *j* and set to 0 to find loc. of maximum:

$$j^* = \frac{k-2}{k-1}n$$

So:

$$\mathrm{E}\{\rho_1(j^*)\} < 1 \text{ when } m/n < \frac{2^{k-1}}{k}\left(\frac{k-1}{k-2}\right)^{k-1}$$

For $k=3$ this is 2.666.

Conclusion: unit-clause algorithm succeeds with probability bounded by a constant when $m/n < 2.666$

What makes this work?

1. The clausal distribution is the same, up to parameters, at each level.

   if $\varphi' \in C_i(j)$ then $\varphi' \in M^{\,i}_{|C_i(j)|,n-j}$

What makes this work?

1. The clausal distribution is the same, up to parameters, at each level.

   if $\varphi' \in C_i(j)$ then $\varphi' \in M^{i}_{|C_i(j)|,n-j}$

2. There is pretty much never a sudden spurious flow.

   $Pr(||C_i(j+1)| - |C_i(j)|| > n^{1/5}) = o(n^{-3})$

What makes this work?

1. The clausal distribution is the same, up to parameters, at each level.

   if $\varphi' \in C_i(j)$ then $\varphi' \in M^{\,i}_{|C_i(j)|,n-j}$

2. There is pretty much never a sudden spurious flow.

   $Pr(||C_i(j+1)| - |C_i(j)|| > n^{1/5}) = o(n^{-3})$

3. The average flow change is pretty smooth.

   $E\{|C_i(j+1)| - |C_i(j)|\} = f_i\,(j/n, |C_0(j)|/n, \ldots , |C_k(j)|/n) + o(1)$

   and

   f_i is continuous and

   $|f_i\,(u_1, \ldots , u_{k+2}) - f_i\,(v_1, \ldots , v_{k+2})| \leq L \sum_{1 \leq i \leq j} |u_i - v_i|$

Is a better result possible?

Sure! Adjust the flows using a heuristic which chooses values so as to increase flow "out" and decrease flow "down"

Is a better result possible?

Sure! Adjust the flows using a heuristic which chooses values so as to increase flow "out" and decrease flow "down"

Examples:

1. If there is no unit clause, choose a variable with greatest difference between number of occurrences of positive and negative literals and a value that satisfies most of the clauses it is contained in.

Is a better result possible?

Sure! Adjust the flows using a heuristic which chooses values so as to increase flow "out" and decrease flow "down"

Examples:

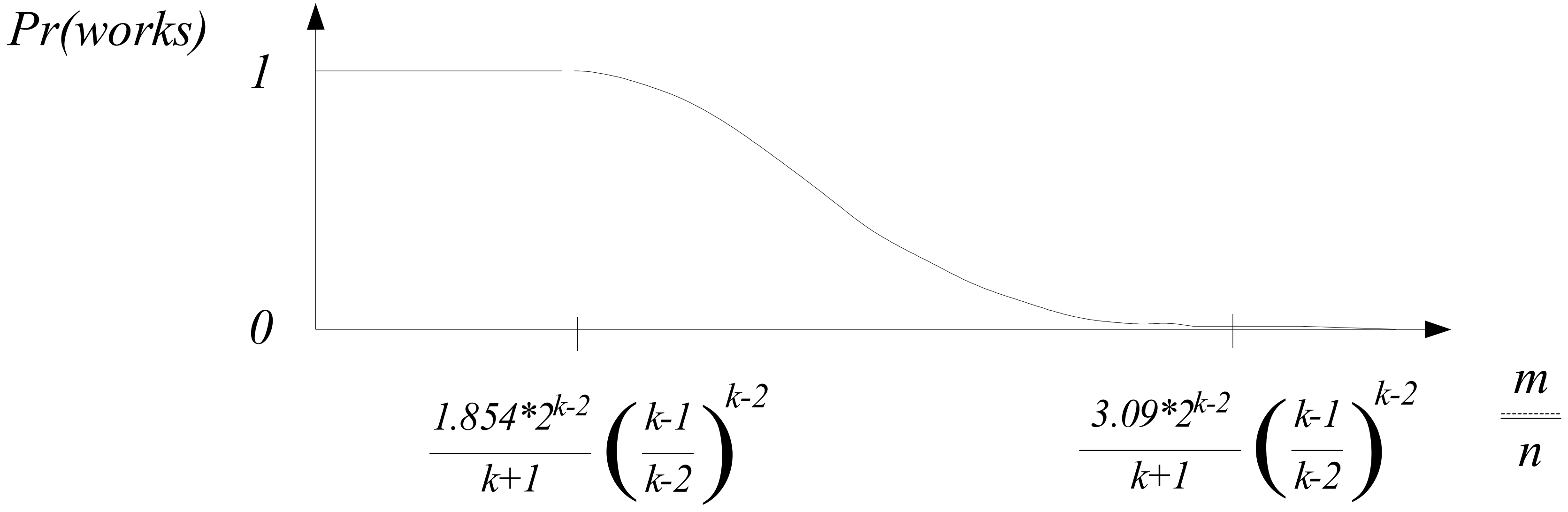
1. If there is no unit clause, choose a variable with greatest difference between number of occurrences of positive and negative literals and a value that satisfies most of the clauses it is contained in.
2. If there is no unit clause, choose a variable from a smallest clause and a value that satisfies that clause.

Is a better result possible?

Sure! Adjust the flows using a heuristic which chooses values so as to increase flow "out" and decrease flow "down"

Examples:

1. If there is no unit clause, choose a variable with greatest difference between number of occurrences of positive and negative literals and a value that satisfies most of the clauses it is contained in.
2. If there is no unit clause, choose a variable from a smallest clause and a value that satisfies that clause.



Is there a limit to all this?

Sure! For $\varphi \in M^{3}_{m,n}$ no algorithm good for $m/n > 3.22$

Optimal algorithm: *picking single variable at a time*:

```
Repeat the following until either $\emptyset \in \varphi$ or $\varphi = \emptyset$:
    if there is a unit clause $\{v\}$ or $\{\neg v\}$ in $\varphi$ choose $v = 1|0$, resp.
    otherwise,
    choose 2-clause $\{v, w\}$ or $\{v, \neg w\}$, or $\{\neg v, w\}$ or $\{\neg v, \neg w\}$ at random,
    select $v$, $\neg v$, $w$ or $\neg w$ at random and temporarily assign the value to
    the chosen variable which satisfies its clause. Now fix the assignment:
        Let $M_3(j)$ = # 3-clauses satisfied - # literals falsified in 3-clauses
            $M_2(j)$ = # 2-clauses satisfied - # literals falsified in 2-clauses
            $d_3(j) = m_3(j)/(n-j)$
            $d_2(j) = m_2(j)/(n-j)$   // recall = $\rho_1(j)$, should be < 1
            $\theta(j) = (1.5 \cdot d_3(j) - 2 \cdot d_2(j) + \gamma^*)/(1 - d_2(j))$
            Reverse the assignment if $\theta(j) > M_3(j)/M_2(j)$
```

θ: Gain by free moves in suppressing 2-clauses to scarcity of free moves

Trade fewer 2-clauses ($M_3(j)$) for more future possibilities ($M_2(j)$)

What about picking two literals at a time?

Worse!   For $\varphi \in M^{3}_{m,n}$ no algorithm good for *m/n > 3.19!*

What about picking two literals at a time?

Worse! For $\varphi \in M^{3}_{m,n}$ no algorithm good for *m/n > 3.19!*

Reasons: More control over the reduction of 3-clauses and increase of 2-clauses but number of variables decreases more rapidly. Since critical numbers are $d_3(j)$ and $d_2(j)$, improvement in numerator is offset by increase in denominator.

What about picking two literals at a time?

Worse! For $\varphi \in M^{3}_{m,n}$ no algorithm good for $m/n > 3.19!$

Reasons: More control over the reduction of 3-clauses and increase of 2-clauses but number of variables decreases more rapidly. Since critical numbers are $d_3(j)$ and $d_2(j)$, improvement in numerator is offset by increase in denominator.

In single variable choice algorithm, revealing two variables gives four choices, in two-variable choice algorithm we get three choices since one choice would falsify a clause.

What about picking two literals at a time?

Worse! For $\varphi \in M^{3}_{m,n}$ no algorithm good for $m/n > 3.19!$

Reasons: More control over the reduction of 3-clauses and increase of 2-clauses but number of variables decreases more rapidly. Since critical numbers are $d_3(j)$ and $d_2(j)$, improvement in numerator is offset by increase in denominator.

In single variable choice algorithm, revealing two variables gives four choices, in two-variable choice algorithm we get three choices since one choice would falsify a clause.

What about a mixed algorithm?

Still not great! For $\varphi \in M^{3}_{m,n}$ no algorithm good for $m/n > 3.26!$

Are there natural heuristics which are outside this framework?

Sure!

1. A pure literal: v appears in clauses and $\neg v$ does not or
   $\neg v$ appears in clauses and v does not.

Algorithm: On a free move, find a pure literal and remove
all clauses containing it.

Are there natural heuristics which are outside this framework?

Sure!

1. A pure literal: v appears in clauses and $\neg v$ does not or
   $\neg v$ appears in clauses and v does not.

   Algorithm: On a free move, find a pure literal and remove
   all clauses containing it.

2. Maximize expected number of models with given partial assignment.
   Let t be a partial assignment including a new assignment to v
   Suppose t has i variables assigned values
   $Pr\,(t \text{ falsifies a clause } c \mid |c| = j,\ t \text{ fixes all literals in } c) = 2^{-j}$
   $Pr\,(t \text{ fixes all literals in } c \mid |c| = j) = \binom{i}{j}/\binom{n}{j}$
   $Pr\,(|c| = j) = 1$ if $j=k$, 0 otherwise (suppose)
   $f_{t,c} = Pr\,(t \text{ falsifies } c) = 2^{-k}\binom{i}{k}/\binom{n}{k}$
   weight of $t = (1 - f_{t,c})^m$ (looks like $Pr\,(t$ satisfies all clauses$)$)

Are we stuck? Are myopic results all we can hope for?

Nope! We have a new tool!

Algorithms that make use of the number of appearances of literals in 3-clauses and 2-clauses, separately, are allowed!

Are we stuck? Are myopic results all we can hope for?

Nope! We have a new tool!

Algorithms that make use of the number of appearances of literals in 3-clauses and 2-clauses, separately, are allowed!

A Greedy Algorithm:

```
Repeat the following until $\varphi = \emptyset$ or $\emptyset \in \varphi$:
  Let $v$ or $\neg v$ be a literal occurring at least as frequently in $\varphi$ as any
  $\varphi = \{ c - \{\neg v\} : c \in \varphi, v \notin c\}$ or
  $\varphi = \{ c - \{v\} : c \in \varphi, \neg v \notin c\}$
  Repeat the following until no unit clauses exist in $\varphi$:
    Select a unit clause $\{v\}$ or $\{\neg v\}$ and set $v$ to $1$ or $0$ and
    $\varphi = \{ c - \{\neg v\} : c \in \varphi, v \notin c\}$ or
    $\varphi = \{ c - \{v\} : c \in \varphi, \neg v \notin c\}$
If $\varphi = \emptyset$ then report a solution is found otherwise give up
```

Are we stuck? Are myopic results all we can hope for?

Nope! We have a new tool!

Algorithms that make use of the number of appearances of literals in 3-clauses and 2-clauses, separately, are allowed!

A Greedy Algorithm:

```
Repeat the following until $\varphi = \emptyset$ or $\emptyset \in \varphi$:
    Let $v$ or $\neg v$ be a literal occurring at least as frequently in $\varphi$ as any
    $\varphi = \{ c - \{\neg v\} : c \in \varphi, v \notin c\}$ or
    $\varphi = \{ c - \{v\} : c \in \varphi, \neg v \notin c\}$
    Repeat the following until no unit clauses exist in $\varphi$:
        Select a unit clause $\{v\}$ or $\{\neg v\}$ and set $v$ to $1$ or $0$ and
        $\varphi = \{ c - \{\neg v\} : c \in \varphi, v \notin c\}$ or
        $\varphi = \{ c - \{v\} : c \in \varphi, \neg v \notin c\}$
If $\varphi = \emptyset$ then report a solution is found otherwise give up
```

Pr (Greedy algorithm succeeds) > 0 if $m/n < 3.42$

Friedgut & others:

Lift constant, bounded probability results to almost always results

Let s,t be vectors in $\{0,1\}^N$

Say $s \geq t$ if $s_i \geq t_i$ holds for all $i=0,...,N$

Call vector set E monotone increasing if $s \in E$, $t \geq s$, then $t \in E$

E expresses some *"property"* like connectedness

Friedgut & others:

Lift constant, bounded probability results to almost always results

Let s,t be vectors in $\{0,1\}^N$

Say $s \geq t$ if $s_i \geq t_i$ holds for all $i=0,\ldots,N$

Call vector set E monotone increasing if $s \in E$, $t \geq s$, then $t \in E$

E expresses some *"property"* like connectedness

Let $w(s)$ be the Hamming weight of s

Define $\mu_{\mathrm{p}}(\{s\}) = (1-p)^{N-w(s)}p^{w(s)},\ 0 \leq p \leq 1$

Define $\mu_p(E) = \sum_{s\in E}\mu_p(\{s\})$

What happens to $\mu_p(E)$ as a function of p?

Friedgut & others:

Lift constant, bounded probability results to almost always results

Let s,t be vectors in $\{0,1\}^N$

Say $s \geq t$ if $s_i \geq t_i$ holds for all $i=0,\ldots,N$

Call vector set E monotone increasing if $s \in E$, $t \geq s$, then $t \in E$

E expresses some *"property"* like connectedness

Let $w(s)$ be the Hamming weight of s

Define $\mu_{\mathrm{p}}(\{s\}) = (1-p)^{N-w(s)}p^{w(s)}$, $0 \leq p \leq 1$

Define $\mu_p(E) = \sum_{s\in E}\mu_p(\{s\})$

What happens to $\mu_p(E)$ as a function of p?

Set $f_N(p) = \mu_p(E_N)$ and let $p_c(N) = f_N^{-1}(c)$

if $lim_{N\rightarrow\infty}(p_{1-\epsilon}(N)-p_\epsilon(N))/p_{1/2}(N) = 0$, have a sharp threshold

if $lim_{N\rightarrow\infty}(p_{1-\epsilon}(N)-p_\epsilon(N))/p_{1/2}(N) > \mathrm{c}$, have a coarse threshold

Friedgut & others:

Lift constant, bounded probability results to almost always results

Let s,t be vectors in $\{0,1\}^N$

Say $s \geq t$ if $s_i \geq t_i$ holds for all $i=0,\ldots,N$

Call vector set E monotone increasing if $s \in E$, $t \geq s$, then $t \in E$

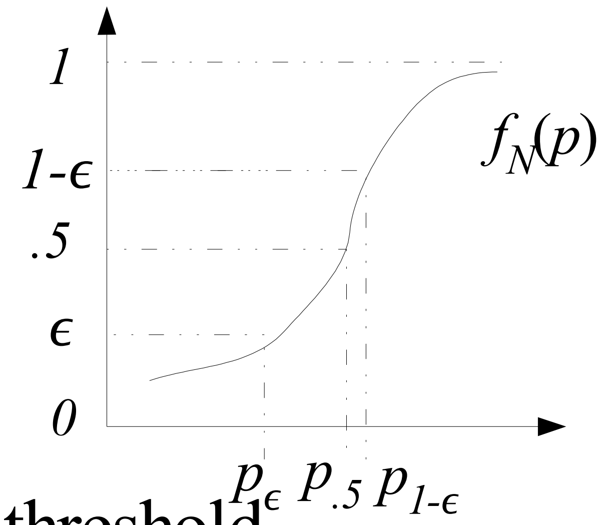
E expresses some *"property"* like connectedness

Let $w(s)$ be the Hamming weight of s

Define $\mu_p(\{s\}) = (1-p)^{N-w(s)} p^{w(s)}$, $0 \leq p \leq 1$

Define $\mu_p(E) = \sum_{s \in E} \mu_p(\{s\})$

What happens to $\mu_p(E)$ as a function of p?



Set $f_N(p) = \mu_p(E_N)$ and let $p_c(N) = f_N^{-1}(c)$

if $\lim_{N \rightarrow \infty} (p_{1-\epsilon}(N) - p_\epsilon(N))/p_{1/2}(N) = 0$, have a sharp threshold

if $\lim_{N \rightarrow \infty} (p_{1-\epsilon}(N) - p_\epsilon(N))/p_{1/2}(N) > c$, have a coarse threshold

Friedgut & others:

If E is monotone such that $p_{1/2}(N) = o(1)$ and the following hold then E has a sharp threshold:

1. For each $c \in (0,1)$ and all positive K,
   $lim_{N \rightarrow \infty} \mu_{p_c(N)}(s \geq m,\ m \in E$ and $w(m) \leq K) = 0$

   (Few elements of E of bounded Hamming weight appear)

2. For each $c \in (0,1)$, all positive K, and all $s_0 \notin E$ with $w(s_0)=K$,
   $lim_{N \rightarrow \infty} \mu_{p_c(N)}(s \in E \mid s \geq s_0) = c$

   (Probability of being in E is not affected much by conditioning on a bounded weight element not in E)

Friedgut & others:

If E is monotone such that $p_{1/2}(N) = o(1)$ and the following hold then E has a sharp threshold:

1. For each $c \in (0,1)$ and all positive K,

$$lim_{N \to \infty} \mu_{p_c(N)}(s \geq m,\ m \text{ is minimal for } E \text{ and } w(m) \leq K) = 0$$

(Few elements of E of bounded Hamming weight appear)

2. For each $c \in (0,1)$, all positive K, and all $s_0 \notin E$ with $w(s_0)=K$,

$$lim_{N \to \infty} \mu_{p_c(N)}(s \in E,\ s \oplus s_0 \notin E \mid s \geq s_0) = 0$$

(Probability of being in E is not affected much by conditioning on a bounded weight element not in E)

Friedgut & others:

Example:

Assume $\varphi \in M^{k}_{cn,n}$

Let the property be φ has no model

There exists a sequence $r_k(n)$ such that for any $\epsilon > 0$,

$lim_{n \rightarrow \infty} Pr(\varphi$ has no model $\mid c = r_k(n) - \epsilon) = 0$ and

$lim_{n \rightarrow \infty} Pr(\varphi$ has no model $\mid c = r_k(n) + \epsilon) = 1$

So previous bounded probability algorithm results are lifted to w.h.p.

Eigenvalues:

Can be used to certify the non-existence of a model for φ (non-resolution alg)

Eigenvalues:

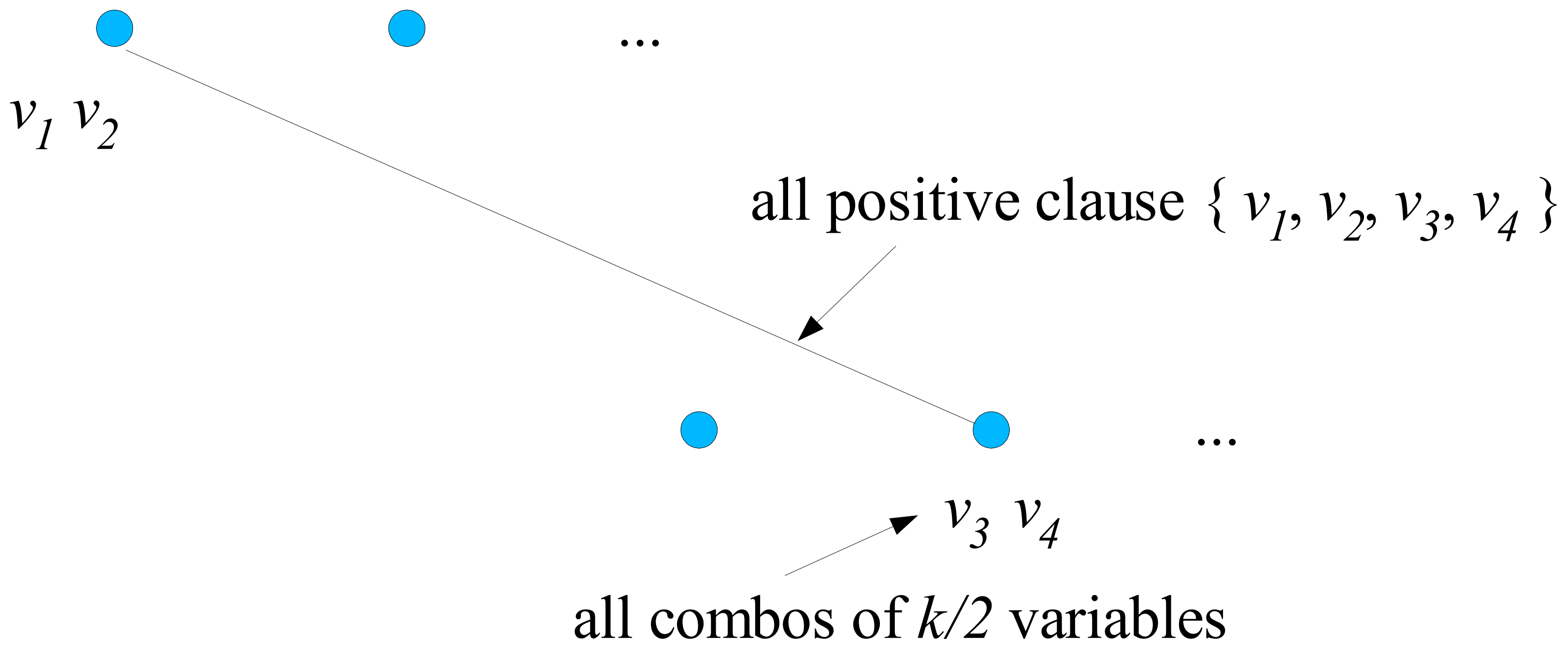
Can be used to certify the non-existence of a model for φ (non-resolution alg)

If $\varphi \in M^k_{m,n}$ has a model then there is a subset V' of $n/2$ variables such that either φ has no all positive clause or no all negative clause taken strictly from V'.

Eigenvalues:

Can be used to certify the non-existence of a model for φ (non-resolution alg)

If $\varphi \in M^{k}_{m,n}$ has a model then there is a subset V' of $n/2$ variables such that either φ has no all positive clause or no all negative clause taken strictly from V'. Suppose $k=4$:



...
$v_1\ v_2$
all positive clause $\{ v_1, v_2, v_3, v_4 \}$
...
$v_3\ v_4$
all combos of $k/2$ variables

Another graph for all negative clauses.

If φ has a model, one graph has Independent Set $> (n/2)^{k/2} = (1/2)^{k/2}|V|$

Eigenvalues:

Let G=(V,E) be an undirected graph with n vertices and m edges

Define $n\times n$ matrix $A_{G,p}$: $a_{i,j} = 1$ iff $\{i,j\} \notin$ E; $1\text{-}1/p$ otherwise, for $0\leq p\leq 1$.

If $\lambda_1(A_{G,p})$ is the largest eigenvalue of $A_{G,p}$ and $\alpha(G)$ is the largest independent set of G then $\lambda_1(A_{G,p}) > \alpha(G)$

Martingales:

Even when dependencies creep in, the probability may still be concentrated around the mean effectively squeezing the probability out of the problem.

Example:

A literal is pure in φ if its complement does not appear in φ

Suppose φ has p pure literals

Let $x_1, \dots , x_{2n-p}$ be # occurrences of all literals in $\varphi \in M^k_{m,n}$

Let $\phi_1\ \phi_2 \dots \phi_{3m}$ be a permutation of the $3m$ literals of φ

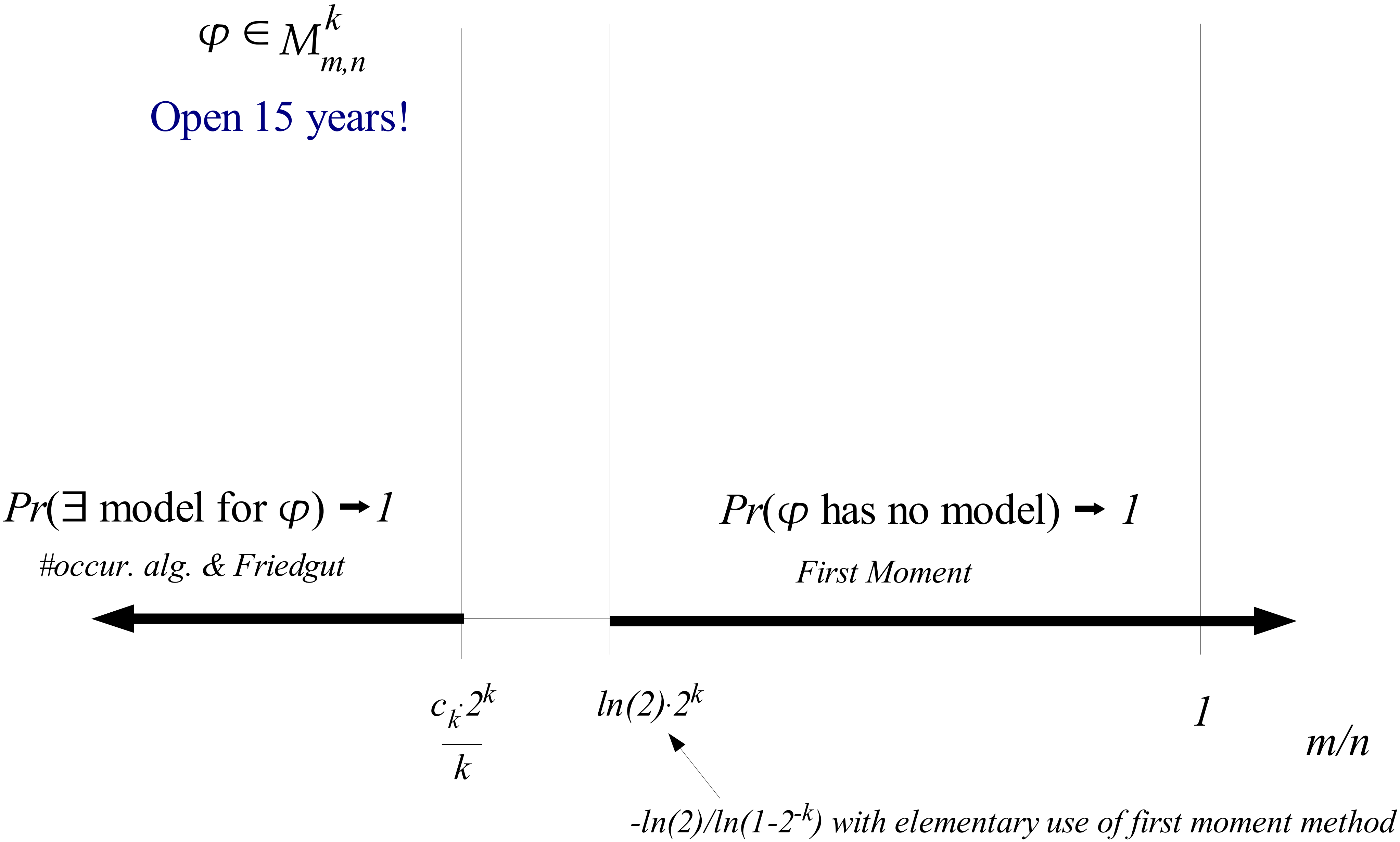
Let m' be the number of clauses remaining after clauses containing p pure literals are removed

Interchanging ϕ_i and ϕ_j cannot change m' by more than 1

Hence $Pr(|m' - E\{m'\}| \geq t \mid x_1, \dots , x_{2n-p}) \leq 2 \cdot \mathrm{e}^{-2t\,/3_2m}$

Pure literal algorithm succeeds on 3-SAT w.h.p when $m/n < 1.63$

Application ⇨ Phase transitions ⇨ *k*-SAT, where?



$\varphi \in M^{k}_{m,n}$

Open 15 years!

$Pr(\exists$ model for $\varphi) \to 1$

#occur. alg. & Friedgut

$Pr(\varphi$ has no model$) \to 1$

First Moment

$\frac{c_k \cdot 2^k}{k}$

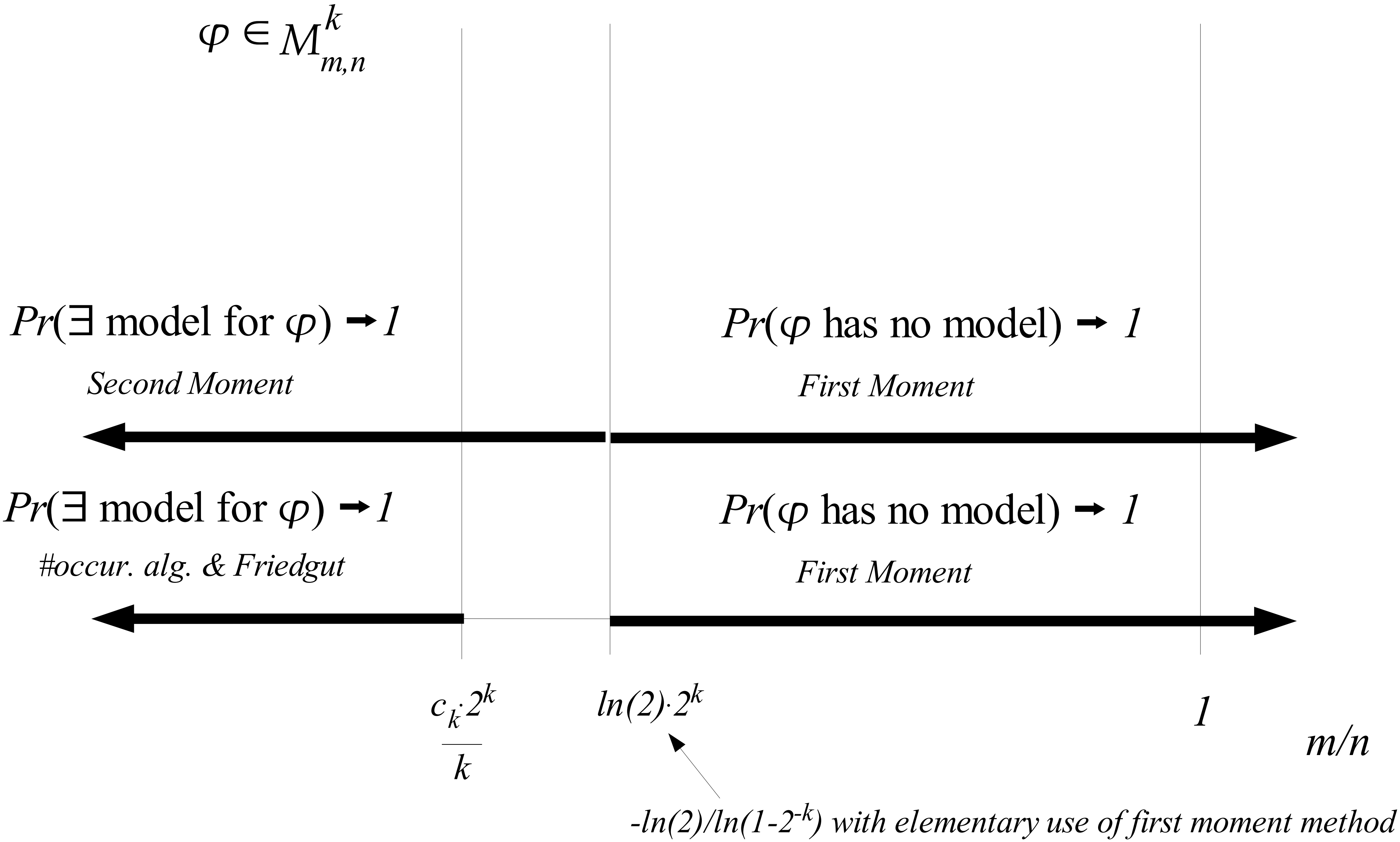
$ln(2) \cdot 2^k$

1

m/n

$-ln(2)/ln(1-2^{-k})$ *with elementary use of first moment method*

Application ⇨ Phase transitions ⇨ *k*-SAT, where?



$\varphi \in M^k_{m,n}$

Pr(∃ model for φ) → *1*

Second Moment

Pr(φ has no model) → *1*

First Moment

Pr(∃ model for φ) → *1*

#occur. alg. & Friedgut

Pr(φ has no model) → *1*

First Moment

$\frac{c_k \cdot 2^k}{k}$

$ln(2) \cdot 2^k$

1

m/n

$-ln(2)/ln(1-2^{-k})$ *with elementary use of first moment method*

Application ⇨ Phase transitions ⇨ *k*-SAT, where?

Assume $\varphi \in M^{k}_{m,n}$

Cannot use second moment method to bound *Pr*(∃ model for φ) due to high variance in # models for φ

But # models for a related problem has low variance:

Not all equal k-SAT: A *nae-model* for φ is a model for φ such that every clause has at least one satisfied literal and one falsified literal.

Pr(∃ model for φ) > Pr(∃ nae-model for φ)

Application ⇨ Phase transitions ⇨ *k*-SAT, where

Let z and w be assignments for φ agreeing in αn variables. Then

$$Pr(z \text{ satisfies } \varphi \mid w \text{ satisfies } \varphi) = \left(1 - \frac{1-\alpha^k}{2^k-1}\right)^m$$

Lopsided: maximum occurs at $\alpha \neq 1/2$, variance is out of control.

$$\sum_{0\leq\alpha\leq 1} \binom{n}{\alpha n} \left(1 - \frac{1-\alpha^k}{2^k-1}\right)^m \quad \text{gets too big}$$

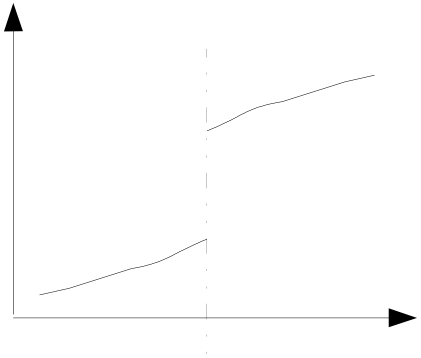
But

$$Pr(z \; nae\text{-satisfies } \varphi \mid w \; nae\text{-satisfies } \varphi) = \left(1 - \frac{1-\alpha^k-(1-\alpha)^k}{2^{k-1}-1}\right)^m$$

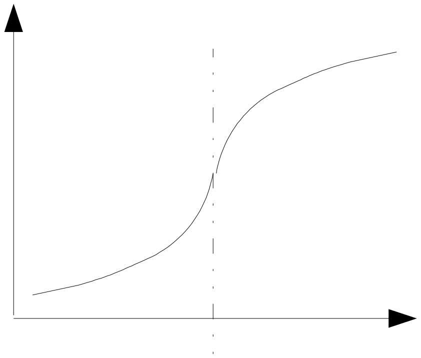
Balanced: looks the same whether α goes to 1 or 0, maximum terms at $\alpha=1/2$, variance is low.

$$\sum_{0\leq\alpha\leq 1} \binom{n}{\alpha n} \left(1 - \frac{1-\alpha^k-(1-\alpha)^k}{2^{k-1}-1}\right)^m \approx 2^n(1-2^{-k+1})^m/\sqrt{n} = o(\mu)$$

Application ⇨ Phase transitions ⇨ 1st, 2nd Order

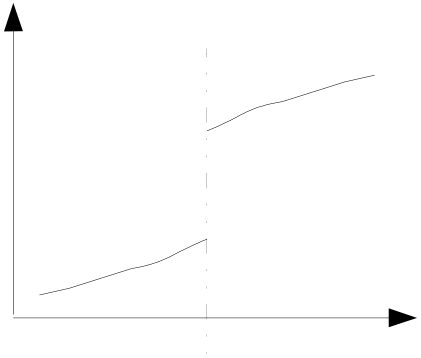


1st order phase transition

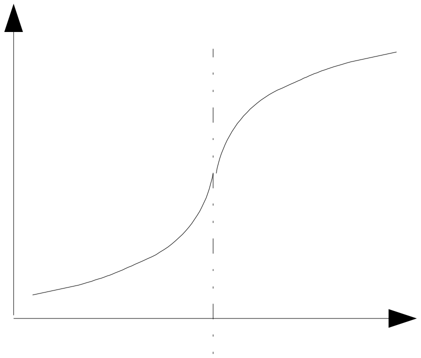


2nd order phase transition

Application ⇨ Phase transitions ⇨ 1st, 2nd Order



1st order phase transition

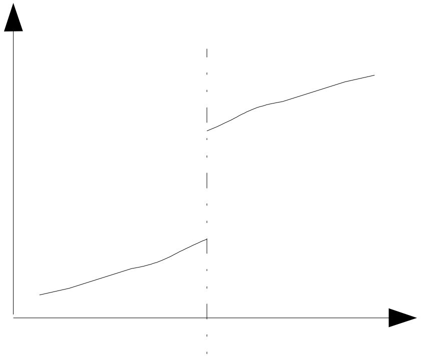


2nd order phase transition

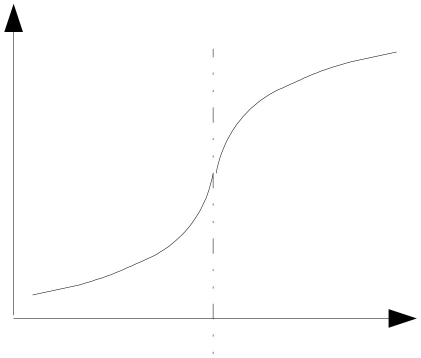
Is there some property for which 1st order transitions mean "hard" instances?
Maybe a property for which 1st order transitions mean some algorithm is 2^{cn}?

Application ⇨ Phase transitions ⇨ 1st, 2nd Order

An *order parameter:* function which is 0 on one side of the transition and non-zero on the other side.



1st order phase transition



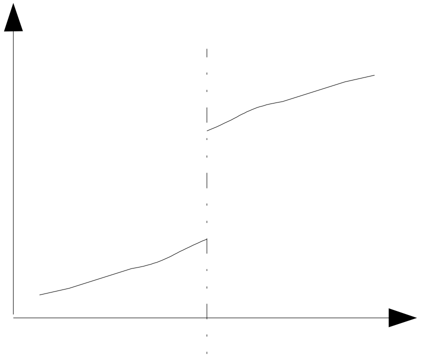
2nd order phase transition

Is there some property for which 1st order transitions mean "hard" instances?

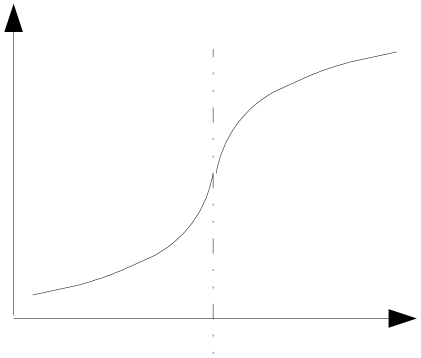
Maybe a property for which 1st order transitions mean some algorithm is 2^{cn}?

Application ⇨ Phase transitions ⇨ 1st, 2nd Order

An *order parameter:* function which is 0 on one side of the transition and non-zero on the other side.



1st order phase transition



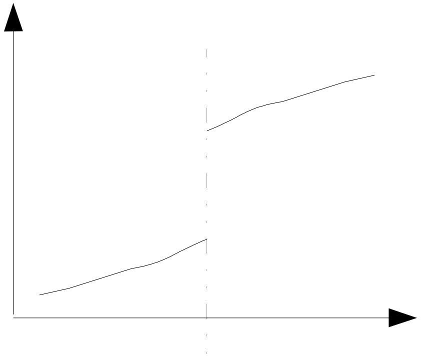
2nd order phase transition

Is there some property for which 1st order transitions mean "hard" instances?
Maybe a property for which 1st order transitions mean some algorithm is 2^{cn}?

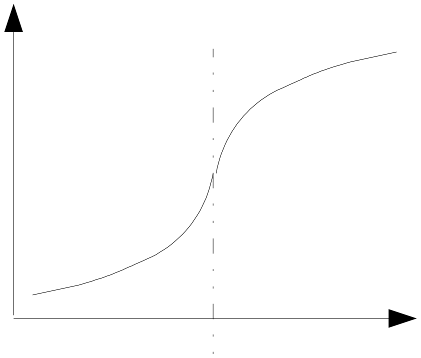
Backbone(φ): variables with same value for all solutions of φ

Application ⇨ Phase transitions ⇨ 1st, 2nd Order

An *order parameter:* function which is 0 on one side of the transition and non-zero on the other side.



1st order phase transition



2nd order phase transition

Is there some property for which 1st order transitions mean "hard" instances?

Maybe a property for which 1st order transitions mean some algorithm is 2^{cn}?

Backbone(φ): variables with same value for all solutions of φ

Backbone size stays at constant fraction of variables near transition (1st order)
DPP has trouble knowing how to set values for those variables and a mistake causes fruitless search of a tree of height *cn*

Hence 1st order transition for backbones suggests trouble for DPLL algs

Application ⇨ Phase transitions ⇨ 1st, 2nd Order

Spines: "relaxed" version of backbones

spine(φ) = $\{ x : x \in \varphi$ and $\exists \varphi' \subseteq \varphi$, φ' has a model, $\varphi' \wedge \{\neg x\}$ has no model $\}$

Application ⇨ Phase transitions ⇨ 1st, 2nd Order

Spines: "relaxed" version of backbones

spine(φ) = { x : $x \in \varphi$ and $\exists \varphi' \subseteq \varphi$, φ' has a model, $\varphi' \wedge \{\neg x\}$ has no model }

2-SAT has 2nd order (continuous) phase transition, and $r_2(n) = 1$

3-SAT has 1st order (discontinuous) phase transition, and $r_3(n) \approx 4.25$

(2+p)-SAT has 2nd order phase transition when $p < 0.413...$

has 1st order phase transition when $p > 0.413...$

2-XOR-SAT has a coarse threshold at $p_2(n) = 1/n$

k-XOR-SAT has a sharp threshold at $p_k(n) = O(n^{1-k})$

Application ⇨ Phase transitions ⇨ 1st, 2nd Order

Spines: "relaxed" version of backbones

spine(φ) = $\{\, x : x \in \varphi$ and $\exists \varphi' \subseteq \varphi$, φ' has a model, $\varphi' \wedge \{\neg x\}$ has no model $\}$

2-SAT has 2nd order (continuous) phase transition, and $r_2(n) = 1$

3-SAT has 1st order (discontinuous) phase transition, and $r_3(n) \approx 4.25$

(2+p)-SAT has 2nd order phase transition when $p < 0.413...$

has 1st order phase transition when $p > 0.413...$

2-XOR-SAT has a coarse threshold at $p_2(n) = 1/n$

k-XOR-SAT has a sharp threshold at $p_k(n) = O(n^{1-k})$

But

1-in-k-SAT (each clause with 1 *true* literal) has 2nd order phase transition

Application ⇨ Phase transitions ⇨ 1st, 2nd Order

However, there is some connection between phase transitions and complexity:

Resolution complexity for random *k*-SAT is usually exponential (and *k*-SAT has a first-order phase transition)

Application ⇨ Phase transitions ⇨ 1st, 2nd Order

However, there is some connection between phase transitions and complexity:

Resolution complexity for random *k*-SAT is usually exponential (and *k*-SAT has a first-order phase transition)

More general SAT problems:

replace *k*-literal clauses with *k*-input Boolean functions
choose the functions according to some probability distribution *D*

If a random SAT(*D*) problem has a sharp, 1st order phase transition
then it has exponential resolution complexity (w.h.p.)

Application ⇨ Phase transitions ⇨ Sharpness

Schaefer (roughly): for every definable class *A* of Satisfiability that can be recognized in *log(n)* space, *A* is either one of the following:

Horn-SAT
2-SAT
XOR-SAT

or else it is NP-complete

Application ⇨ Phase transitions ⇨ Sharpness

Schaefer (roughly): for every definable class *A* of Satisfiability that can be recognized in *log(n)* space, *A* is either one of the following:

Horn-SAT
2-SAT
XOR-SAT

or else it is NP-complete

Constraint function f:$\{0,1\}^k \rightarrow \{0,1\}$ strongly depends on one component if:

$\exists \varepsilon \in \{0,1\}, \exists i \in \{1,\ldots,k\}, \forall (a_1,\ldots,a_k) \in \{0,1\}^k, f(a_1,\ldots,a_k) = 1 \Rightarrow a_i = \varepsilon$

Constraint function f:$\{0,1\}^k \rightarrow \{0,1\}$ strongly depends on a 2-XOR relation if:

$\exists \varepsilon \in \{0,1\}, \exists i \neq j \in \{1,\ldots,k\}^2, \forall (a_1,\ldots,a_k) \in \{0,1\}^k, f(a_1,\ldots,a_k) = 1 \Rightarrow a_i \oplus a_j = \varepsilon$

Sharp phase transition iff no constraint function strongly depends on any one component or on a 2-XOR relation

Polytime solvable iff every constraint function expressible as a 2-CNF instance or every constraint function is expressible as an XOR-instance

Application ⇨ Property (polytime solvable classes)

Probabilistic analysis of polytime solvable classes can reveal:

1. What vulnerability does a class have? That is, what critically distinguishes this class from more difficult classes?

Application ⇨ Property (polytime solvable classes)

Probabilistic analysis of polytime solvable classes can reveal:

1. What vulnerability does a class have? That is, what critically distinguishes this class from more difficult classes?
2. Is one class much larger than another incomparable class in some sense?

Application ⇨ Property ⇨ Horn Set

Horn Set: CNF expression, every clause has at most one positive literal

Example: $\{v_1, \neg v_2\}, \{\neg v_2, \neg v_4, \neg v_5\}, \{v_3\}, \{\neg v_4\}$

Interesting fact: There exists a minimal model with respect to *1* and it is found as follows:

```
Repeat the following until all clauses of $\varphi$ are non-positive:
    Let $\{v\}$ be a positive unit clause in $\varphi$
    Set $v$ to $1$
    $\varphi = \{ c - \{\neg v\} : c \in \varphi, v \notin c\}$
Set all unset variables to $0$
If there are no contradictions, a model has been found
```

Hidden Horn: There exists some switch set that makes φ Horn

Application ⇨ Property ⇨ q-Horn

Write CNF expression as matrix M, rows indexed on clauses, columns indexed on variables, entries $\in\{0,1,-1\}$, and

$$M_{i,j} = \begin{cases} 1 \text{ if } i^{\text{th}} \text{ clause contains } v_j, \\ -1 \text{ if } i^{\text{th}} \text{ clause contains } \neg v_j, \\ 0 \text{ otherwise.} \end{cases}$$

Scale columns by -1, permute rows and columns to get quadrants:

$$\begin{array}{l|c|c|l}
 & -1\ \ 0\ 1\ 0\ 0\ 0 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & \\
 & 1\ {-1}\ {-1}\ \ 0\ 1\ 0 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & \\
\text{Horn} & 0\ \ 0\ {-1}\ {-1}\ 0\ 0 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & \text{All Zero} \\
 & 0\ \ 1\ {-1}\ 0\ 0\ 0 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 & \\
\hline
 & 0\ {-1}\ {-1}\ \ 0\ 0\ 0 & 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0 & \\
 & -1\ \ 0\ 0\ {-1}\ 0\ 0 & 1\ 0\ 0\ {-1}\ 0\ 0\ 0\ 0 & \\
 & 0\ \ 0\ 0\ {-1}\ 0\ 0 & 0\ 0\ 0\ 0\ 1\ 0\ 0\ {-1} & \\
\text{Non-positive} & 0\ \ 0\ {-1}\ {-1}\ 0\ 0 & 0\ {-1}\ 0\ 0\ 0\ 0\ 0\ 0 & \text{2-SAT}
\end{array}$$

Application ⇨ Property ⇨ q-Horn

Interesting property: Let $P_i = \{ j : v_j \in c_i \}, N_i = \{ j : \neg v_j \in c_i \}$

Construct $$\sum_{j \in P_i} \alpha_j + \sum_{j \in N_i} (1 - \alpha_j) \le Z, \quad i=1,2,\ldots,m$$

$$0 \le \alpha_j \le 1 \quad j=1,2,\ldots,n$$

Then if minimum Z satisfying above is no greater than 1, it's q-Horn

Class of all problems with minimum Z greater than $1+n^{-\epsilon}$ for any ϵ is NPC

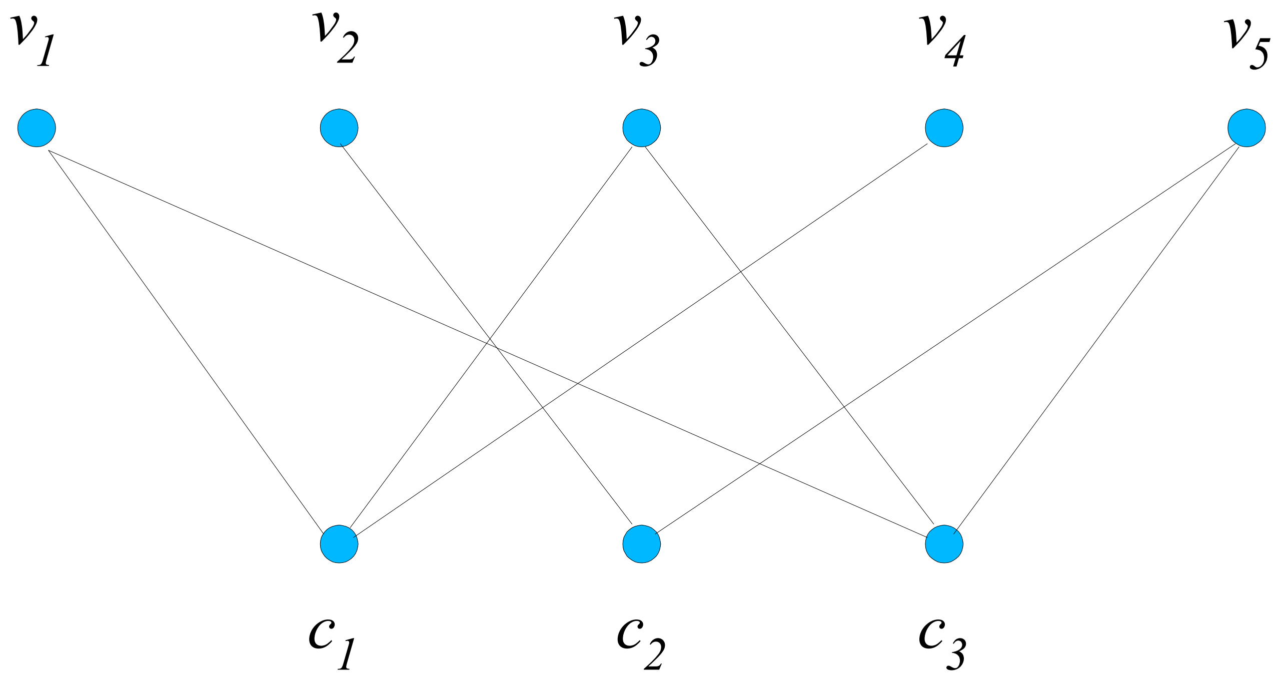
Application ⇨ Property ⇨ SLUR

SLUR instance: For all possible sequences of selected variables, SLUR algorithm below does not give up.

```
SLUR ($\varphi$):
    Repeat the following as long as $\varphi \neq \emptyset$
        Select a variable $v \in \varphi$
        Set $\varphi_1 = \{ c - \{v\} : c \in \varphi, \neg v \notin c \}$
        Set $\varphi_2 = \{ c - \{\neg v\} : c \in \varphi, v \notin c \}$
        Repeat the following as long as a unit clause $\{l\}$ exists in $\varphi_1$:
            Set $\varphi_1 = \{ c - \{\neg l\} : c \in \varphi_1, l \notin c\}$
        Repeat the following as long as a unit clause $\{l\}$ exists in $\varphi_2$:
            Set $\varphi_2 = \{ c - \{\neg l\} : c \in \varphi_2, l \notin c\}$
        If $\emptyset \in \varphi_1$, and $\emptyset \in \varphi_2$ Then give up,
        Otherwise, if $\emptyset \in \varphi_2$ , Set $\varphi = \varphi_1$
        Otherwise, if $\emptyset \in \varphi_1$ , Set $\varphi = \varphi_2$
        Otherwise set $\varphi$ arbitrarily to $\varphi_1$ or $\varphi_2$
```

Application ⇨ Property ⇨ Matched

Matched instance: Total matching in bipartite graph with vertex sets representing clauses and variables and edge $<c,v>$ exists if and only if clause c contains either literal v or $\neg v$.



$\{v_1,\ \neg v_3,\ v_4\},\ \{\neg v_2, \neg v_5\},\ \{\neg v_1, v_3, \neg v_5\}$

Application ⇨ Property

Matched, SLUR, q-Horn classes are incomparable:

Any Horn set with more clauses than distinct variables is q-Horn and SLUR but not Matched.

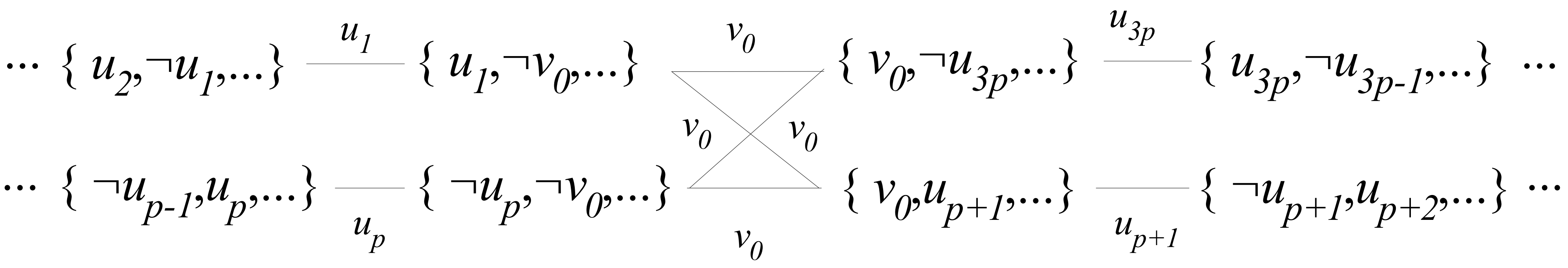
$\{ v_0, \neg v_1, v_3 \}, \{ v_0, v_1, v_4 \}, \{ \neg v_0, \neg v_2, v_5 \}, \{ \neg v_0, v_2, v_6 \}$
is Matched and q-Horn (set α_0 - α_2 = 1/2 and other α_i to *0*, then *Z* is *1*)
but not SLUR (choosing *0* for v_3 - v_6 leaves unsat)

$\{ v_1, v_2, \neg v_4 \}, \{ \neg v_0, \neg v_2, v_3 \}, \{ v_0, v_1, \neg v_3 \}$
is Matched and SLUR (no choices lead to unsat without unit clauses)
but not q-Horn (minimum *Z* is 4/3)

Above can be extended to infinite families

Application ⇨ Property

SLUR vulnerability:



u_1 v_0 u_{3p}

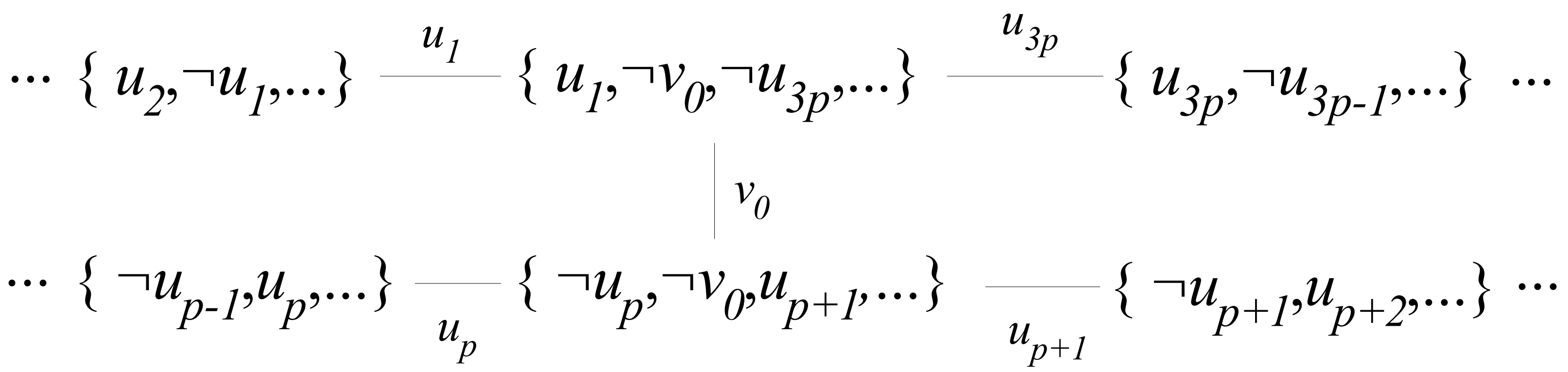
$\cdots\ \{u_2, \neg u_1, \ldots\}$ $\{u_1, \neg v_0, \ldots\}$ $\{v_0, \neg u_{3p}, \ldots\}$ $\{u_{3p}, \neg u_{3p-1}, \ldots\}\ \cdots$

v_0 v_0

$\cdots\ \{\neg u_{p-1}, u_p, \ldots\}$ $\{\neg u_p, \neg v_0, \ldots\}$ $\{v_0, u_{p+1}, \ldots\}$ $\{\neg u_{p+1}, u_{p+2}, \ldots\}\ \cdots$

u_p v_0 u_{p+1}

q-Horn vulnerability:



u_1 u_{3p}

$\cdots\ \{u_2, \neg u_1, \ldots\}$ $\{u_1, \neg v_0, \neg u_{3p}, \ldots\}$ $\{u_{3p}, \neg u_{3p-1}, \ldots\}\ \cdots$

v_0

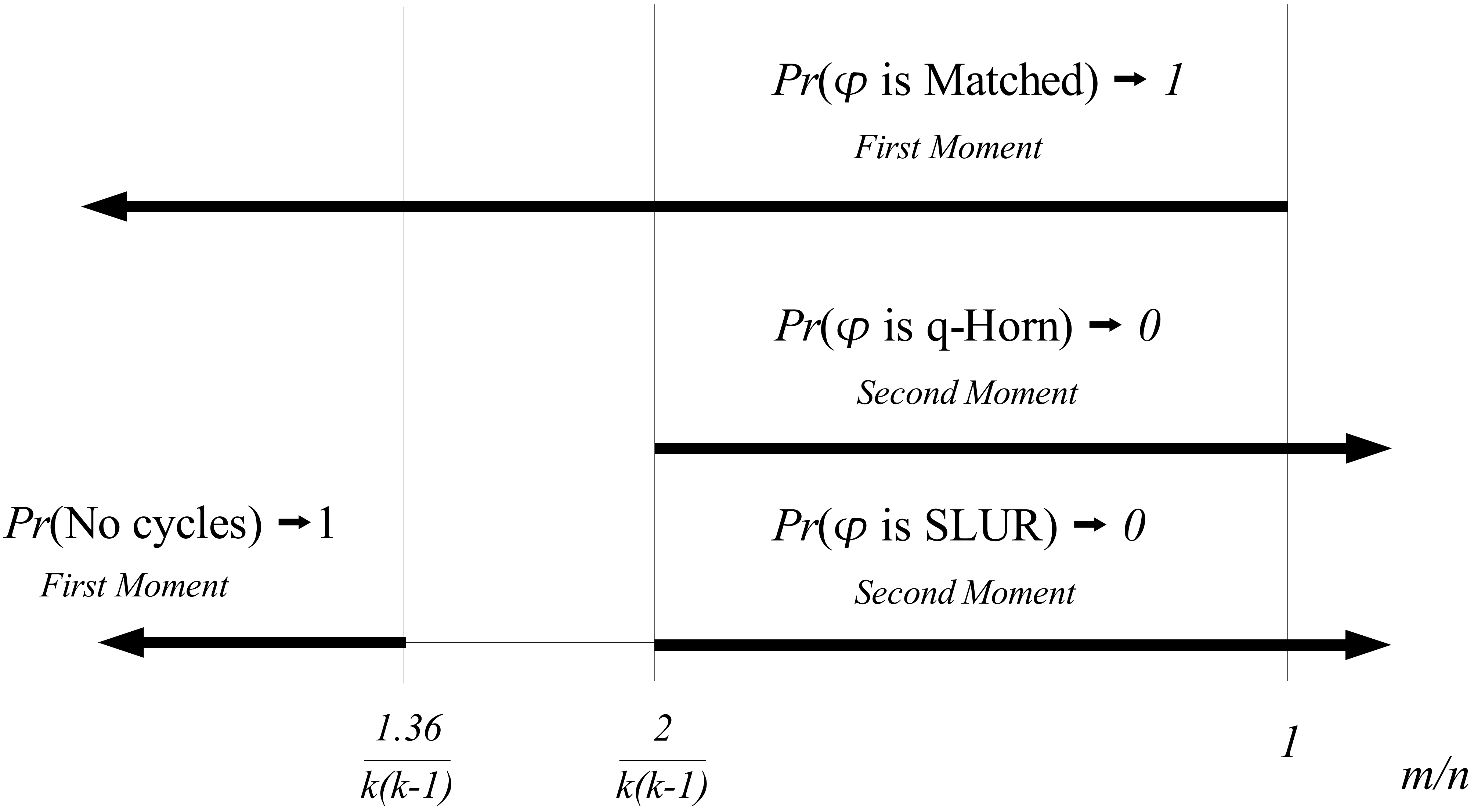
$\cdots\ \{\neg u_{p-1}, u_p, \ldots\}$ $\{\neg u_p, \neg v_0, u_{p+1}, \ldots\}$ $\{\neg u_{p+1}, u_{p+2}, \ldots\}\ \cdots$

u_p u_{p+1}

Matched vulnerability:

Not vulnerable to cycles! Only problem is #clauses < #variables

Application ⇨ Property



Pr(φ is Matched) → *1*

First Moment

Pr(φ is q-Horn) → *0*

Second Moment

Pr(No cycles) → 1

First Moment

Pr(φ is SLUR) → *0*

Second Moment

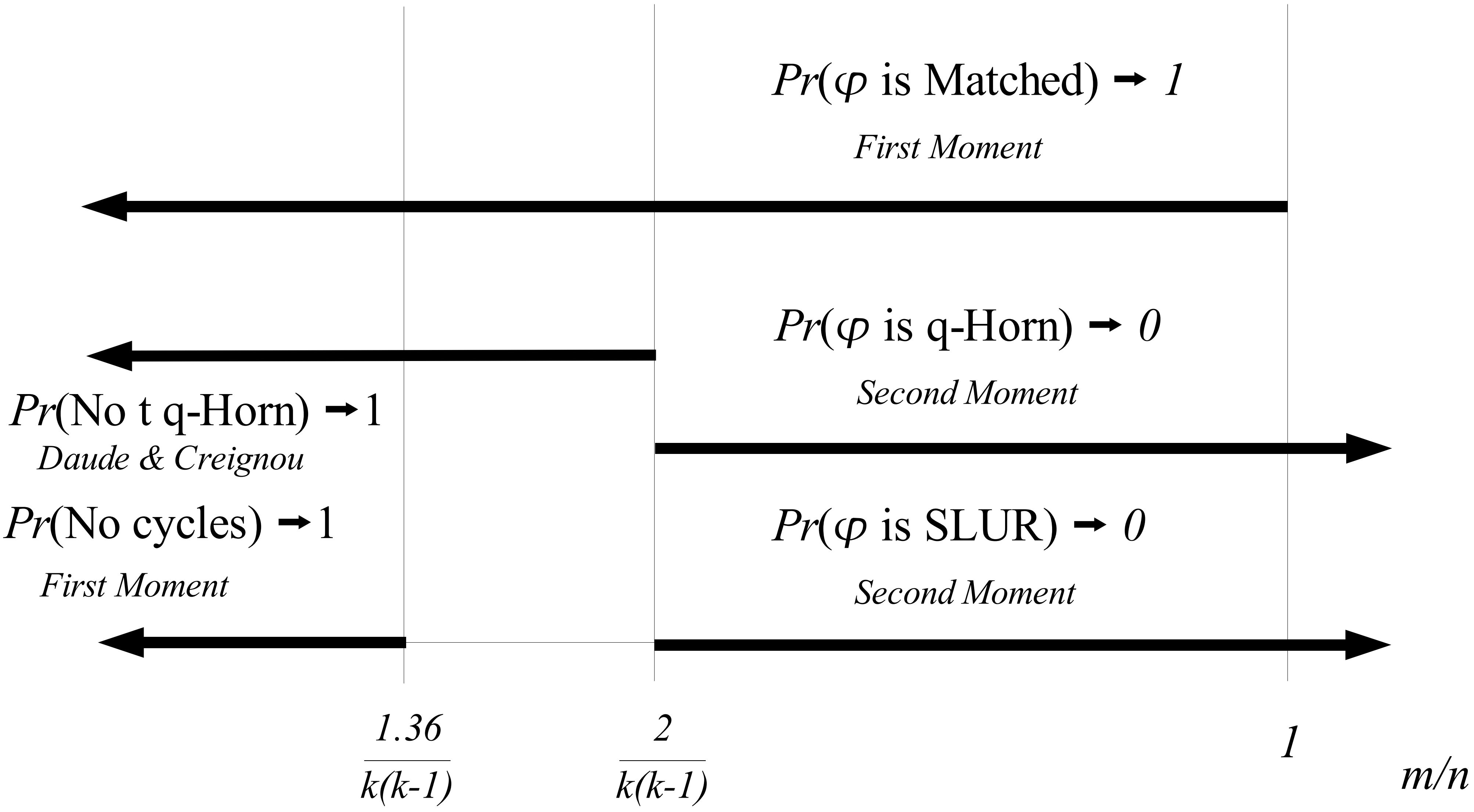
$\frac{1.36}{k(k-1)}$

$\frac{2}{k(k-1)}$

1

m/n

Application ⇨ Property



Pr(φ is Matched) → *1*

First Moment

Pr(φ is q-Horn) → *0*

Second Moment

Pr(No t q-Horn) → 1

Daude & Creignou

Pr(No cycles) → 1

First Moment

Pr(φ is SLUR) → *0*

Second Moment

$\frac{1.36}{k(k-1)}$

$\frac{2}{k(k-1)}$

1

m/n