Available online at www.sciencedirect.com

ScienceDirect

Discrete Applied Mathematics 155 (2007) 1627–1639

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

Phase transitions of PP-complete satisfiability problems☆

Delbert D. Bailey[a], Víctor Dalmau[b], Phokion G. Kolaitis[a]

[a] *Computer Science Department, University of California, Santa Cruz, Santa Cruz, CA 95064, USA*

[b] *Departament de tecnologia, Universitat Pompeu Fabra,Estació de França, Passeig de la circumval.lació 8, Barcelona 08003, Spain*

Received 12 March 2001; received in revised form 21 March 2006; accepted 26 September 2006

Available online 14 December 2006

Abstract

The complexity class PP consists of all decision problems solvable by polynomial-time probabilistic Turing machines. It is well known that PP is a highly intractable complexity class and that PP-complete problems are in all likelihood harder than NP-complete problems. We investigate the existence of phase transitions for a family of PP-complete Boolean satisfiability problems under the fixed clauses-to-variables ratio model. A typical member of this family is the decision problem # 3SAT($\geqslant 2^{n/2}$): given a 3CNF-formula, is it satisfied by at least the square-root of the total number of possible truth assignments? We provide evidence to the effect that there is a critical ratio $r_{3,2}$ at which the asymptotic probability of # 3SAT($\geqslant 2^{n/2}$) undergoes a phase transition from 1 to 0. We obtain upper and lower bounds for $r_{3,2}$ by showing that $0.9227 \leqslant r_{3,2} \leqslant 2.595$. We also carry out a set of experiments on random instances of # 3SAT($\geqslant 2^{n/2}$) using a natural modification of the Davis–Putnam–Logemann–Loveland (DPLL) procedure. Our experimental results suggest that $r_{3,2} \approx 2.5$. Moreover, the average number of recursive calls of this modified DPLL procedure reaches a peak around 2.5 as well.

Keywords: Phase transitions; Satisfiability; PP-complete

1. Introduction and summary of results

During the past several years, there has been an intensive investigation of random Boolean satisfiability in probability spaces parameterized by a fixed clauses-to-variables ratio. More precisely, if $k \geqslant 2$ is an integer, n is a positive integer and r is a positive rational such that rn is an integer, then $F_k(n, r)$ denotes the space of random kCNF-formulas with n variables $x_1, \ldots, x_n$ and rn clauses that are generated uniformly and independently by selecting k variables without replacement from the n variables and then negating each variable with probability $\frac{1}{2}$. Much of the work in this area is aimed at establishing or at least providing evidence for the conjecture, first articulated by Chvátal and Reed [10], that a phase transition occurs in the probability $p_k(n, r)$ of a random formula in $F_k(n, r)$ being satisfiable, as $n \to \infty$. Specifically, this conjecture asserts that, for every $k \geqslant 2$, there is a positive real number r_k such that if $r < r_k$, then $\lim_{n\to\infty} p_k(n, r) = 1$, whereas if $r > r_k$, then $\lim_{n\to\infty} p_k(n, r) = 0$.

☆ A preliminary version of this paper appeared in the *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (*IJCAI* 2001), pp. 183–189. Research of the authors was partially supported by NSF Grants no. CCR-9610257, CCR-9732041, and IIS-9907419.

E-mail addresses: dbailey@cs.ucsc.edu (D.D. Bailey), victor.dalmau@tecn.upf.es (V. Dalmau), kolaitis@cs.ucsc.edu (P.G. Kolaitis).

0166-218X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2006.09.014

So far, this conjecture has been established only for $k = 2$ by showing that $r_2 = 1$ [10,11,14]. For $k \geqslant 3$, upper and lower bounds for r_k have been obtained analytically and experiments have been carried out that provide evidence for the existence of r_k and estimate its actual value. For $k = 3$, in particular, it has been proved that $3.42 \leqslant r_3 \leqslant 4.571$ [15,16] and extensive experiments have suggested that $r_3 \approx 4.2$ [23]. Moreover, the experiments reveal that the median running time of the Davis–Putnam–Logemann–Loveland (DPLL) procedure for satisfiability attains a peak around 4.2. Thus, the critical ratio at which the probability of satisfiability undergoes a phase transition coincides with the ratio at which this procedure requires maximum computational effort to decide whether a random formula is satisfiable.

Boolean satisfiability is the prototypical NP-complete problem. Since many reasoning and planning problems in artificial intelligence turn out to be complete for complexity classes beyond NP, in recent years researchers have embarked on an investigation of phase transitions for such problems. For instance, it is known that STRIPS planning is complete for the class PSPACE of all polynomial-space solvable problems [6]. A probabilistic analysis of STRIPS planning and an experimental comparison of different algorithms for this problem have been carried out in [7]. In addition to STRIPS planning, researchers have also investigated phase transitions for the prototypical PSPACE-complete problem QSAT, which is the problem of evaluating a given quantified Boolean formula [8,12]. Actually, this investigation has mainly focused on the restriction of QSAT to random quantified Boolean formulas with two alternations (universal–existential) of quantifiers, a restriction which forms a complete problem for the class Π_2P at the second level of the polynomial hierarchy PH. The lowest level of PH is NP, while higher levels of this hierarchy consist of all decision problems (or of the complements of all decision problems) computable by nondeterministic polynomial-time Turing machines using oracles from lower levels (see [27] for additional information on PH and its levels). Another PSPACE-complete problem closely related to QSAT is stochastic Boolean satisfiability SSAT, which is the problem of evaluating an expression consisting of existential and randomized quantifiers applied to a Boolean formula. Experimental results on phase transitions for SSAT have been reported in [17,19].

Between NP and PSPACE lie several other important complexity classes that contain problems of significance in artificial intelligence. Two such classes, closely related to each other and of interest to us here, are #P and PP. The class #P, introduced and first studied by Valiant [26,27], consists of all functions that count the number of accepting paths of nondeterministic polynomial-time Turing machines. The prototypical #P-complete problem is #SAT, which is the problem of counting the number of truth assignments that satisfy a CNF-formula. It is well known that numerous #P-complete problems arise naturally in logic, algebra, and graph theory [26,27]. Moreover, #P-complete problems are encountered in artificial intelligence; these include the problem of computing Dempster's rule for combining evidence [20] and the problem of computing probabilities in Bayesian belief networks [22]. In recent years, researchers have initiated an experimental investigation of extensions of the DPLL procedure for solving SAT. Specifically, a procedure for solving SAT, called counting Davis–Putnam (CDP), was presented and experiments on random 3CNF formulas from the space $F_3(n, r)$ were carried out in [5]. The main experimental finding was that the median running time of CDP reaches its peak when $r \approx 1.2$. A different DPLL extension for solving SAT, called decomposing Davis–Putnam (DDP), was presented in [4]; this procedure is based on recursively identifying connected components in the constraint graph associated with a CNF-formula. Additional experiments on random 3CNF-formulas from $F_3(n, r)$ were conducted and it was found out that the median running time of DDP reaches its peak when $r \approx 1.5$.

In the case of the NP-complete problems kSAT , $k \geqslant 3$, the peak in the median running time of the DPLL procedure occurs at the critical ratio at which the probability of satisfiability appears to undergo a phase transition. Since #SAT is a counting problem (returning numbers as answers) and not a decision problem (returning "yes" or "no" as answers), it is not meaningful to associate with it a probability of getting a "yes" answer; therefore, it does not seem possible to correlate the peak in the median running times of algorithms for #SAT with a structural phase transition of #SAT . Nonetheless, there exist decision problems that in a certain sense embody the intrinsic computational complexity of #P-complete problems. These are the problems that are complete for the class PP of all decision problems solvable using a polynomial-time *probabilistic Turing machine*, that is, a polynomial-time non-deterministic Turing machine M that accepts a string x if and only if at least half of the computations of M on input x are accepting. The class PP was first studied by Simon [24] and Gill [13], where several problems were shown to be PP-complete under polynomial-time reductions. In particular, the following decision problem, also called #SAT , is PP-complete: given a CNF-formula φ and a positive integer i, does φ have at least i satisfying truth assignments? This problem constitutes the decision version of the counting problem #SAT , which justifies the innocuous overload of notation.

Another canonical PP-complete problem, which is actually a special case of #SAT , is MAJORITY SAT: given a CNF-formula, is it satisfied by at least half of the possible truth assignments to its variables? In addition, several evaluation and testing problems in probabilistic planning under various domain representations have recently been shown to be PP-complete [18].

It is known that the class PP contains both NP and coNP, and is contained in PSPACE (see [21]). Moreover, as pointed out by Angluin [3], there is a tight connection between #P and PP. Specifically, $P^{\#P} = P^{PP}$, which means that the class of decision problems computable in polynomial time using #P oracles coincides with the class of decision problems computable in polynomial time using PP oracles. This is precisely the sense in which PP-complete problems embody the same intrinsic computational complexity as #P-complete problems. Moreover, PP-complete problems (and #P-complete problems) are considered to be substantially harder than NP-complete problems, since in a technical sense they dominate all problems in the polynomial hierarchy PH. Indeed, the main result in [25] asserts that $PH \subseteq P^{PP} = P^{\#P}$. In particular, Toda's result implies that no PP-complete problem lies in PH, unless PH collapses down to one of its levels, which is considered to be a highly improbable state of affairs in complexity theory.

In [17], initial experiments were carried out to study the median running time of an extension of the DPLL procedure on instances (φ, i) of the PP-complete decision problem #SAT in which φ was a random 3CNF-formula drawn from $F_3(n, rn)$ and $i = 2^t$, for some nonnegative integer $t \leqslant n$. These experiments were also reported in [19], which additionally contains a discussion on possible phase transitions for the decision problem #SAT and preliminary results concerning coarse upper and lower bounds for the critical ratios at which phase transitions may occur (in these two papers #SAT is called MAJSAT). As noted earlier, the main emphasis of both [17] and Littman et al. [19] is not on #SAT or on PP-complete problems, but on stochastic Boolean satisfiability SSAT, which is a PSPACE-complete problem containing #SAT as a special case.

In this paper, we embark on a systematic investigation of phase transitions for a large family of PP-complete satisfiability problems. Specifically, for every integer $k \geqslant 3$ and every integer $t \geqslant 2$, let #kSAT($\geqslant 2^{n/t}$) be the following decision problem: given a kCNF-formula φ with n variables, does φ have at least $2^{n/t}$ satisfying truth assignments? In particular, for $t = 2$ and for every $k \geqslant 3$, we have the decision problem #kSAT($\geqslant 2^{n/2}$): given a kCNF-formula, is it satisfied by at least the square-root of the total number of possible truth assignments? Intuitively, each of these problems can be thought of as a question about the occurrence of 1 in a prefix part of the binary representation of the total number of satisfying assignments. For instance, #3SAT($\geqslant 2^{n/2}$)asks whether at least one of the first $n/2$ Clearly, for every integer $tgeq2$, the problem #kSAT($\geqslant 2^{n/t}$) is a restriction of the decision problem #SAT . Note that, while an instance of #SAT is a pair (φ, i), an instance of #kSAT($\geqslant 2^{n/t}$) is just a kCNF-formula φ; this makes it possible to study the behavior of random #kSAT($\geqslant 2^{n/t}$) in the same framework as the one used for random kSAT . One may also consider the behavior of random MAJORITY kSAT, $k \geqslant 3$. In Section 3, however, we observe that the asymptotic behavior of random MAJORITY kSAT is trivial and that, in particular, it does not undergo any phase transition. In contrast, the state of affairs for random #kSAT($\geqslant 2^{n/t}$) will turn out to be by far more interesting.

We first show that, for every $k \geqslant 3$ and every $t \geqslant 2$, the problem #kSAT($\geqslant 2^{n/t}$) is indeed PP-complete. We conjecture that each of these problems undergoes a phase transition at some critical ratio $r_{k,t}$ of clauses to variables: as $n \to \infty$, for ratios $r < r_{k,t}$, almost all formulas in $F_k(n, r)$ are "yes" instances of #kSAT($\geqslant 2^{n/t}$), whereas for ratios $r > r_{k,t}$, almost all formulas in $F_k(n, r)$ are "no" instances of #kSAT($\geqslant 2^{n/t}$). As a first step towards this conjecture, we establish analytically upper and lower bounds for $r_{k,t}$. A standard application of Markov's inequality easily yields that $((t-1)/t)(1/(k - \lg(2^k - 1)))$ is an upper bound for $r_{k,t}$ (this was also implicit in [19]). Using an elementary argument and the fact that the probability of satisfiability of random 2CNF-formulas undergoes a phase transition at $r_2 = 1$, we show that $(1 - 1/t)$ is a coarse lower bound for $r_{k,t}$. In particular, these results imply that the critical ratio $r_{3,2}$ of #3SAT($\geqslant 2^{n/2}$) obeys the following bounds: $0.5 \leqslant r_{3,2} \leqslant 2.595$. After this, we analyze a randomized algorithm, called extended unit clause (EUC), for #3SAT($\geqslant 2^{n/2}$) and show that with probability bounded away from zero it returns a "yes" answer when $r \leqslant 0.9227$; therefore, $r_{3,2} \geqslant 0.9227$. Although EUC is a simple heuristic, its analysis is rather complex. This analysis is carried out by adopting and extending the powerful methodology of differential equations, first used by Achlioptas [1] to derive improved lower bounds for the critical ratio r_3 of random 3CNF-formulas.

Finally, we complement these analytical results with a set of experiments for #3SAT($\geqslant 2^{n/2}$) by implementing a modification of the CDP procedure and running it on formulas drawn from $F_3(n, rn)$. Our experimental results suggest that the probability of #3SAT($\geqslant 2^{n/2}$) undergoes a phase transition when $r \approx 2.5$. Thus, the 2.595 upper bound for $r_{3,2}$

obtained using Markov's inequality turns out to be remarkably close to the value of $r_{3,2}$ suggested by the experiments. Moreover, the average number of recursive calls of the modified CDP procedure reaches a peak around the same critical ratio 2.5.

2. PP-completeness of #kSAT($\geqslant 2^{n/t}$)

In [26], the counting problem #SAT was shown to be #P-complete via *parsimonious* reductions, which means that every problem in #P can be reduced to #SAT via a polynomial-time reduction that preserves the number of solutions. Moreover, the same holds true for the counting versions of many other NP-complete problems, including #kSAT , the restriction of #SAT to kCNF-formulas. We now use this fact to identify a large family of PP-complete problems.

Proposition 1. *For every integer $k \geqslant 3$ and every integer $t \geqslant 2$, the decision problem* #kSAT($\geqslant 2^{n/t}$) is PP-complete. In particular, #3SAT($\geqslant 2^{n/2}$) is PP-complete.

Proof. For concreteness, in what follows we show that #3SAT($\geqslant 2^{n/2}$) is PP-complete. Let Q be the following decision problem: given a 3CNF-formula ψ and a positive integer i, does ψ have at least 2^i satisfying truth assignments? Since #3SAT is a #P-complete problem under parsimonious reductions, there is a polynomial-time transformation such that, given a CNF-formula φ with variables $x_1, \ldots, x_n$, it produces a 3CNF-formula ψ whose variables include $x_1, \ldots, x_n$ and has the same number of satisfying truth assignments as φ. Consequently, φ is a "yes" instance of MAJORITY SAT (that is, it has at least 2^{n-1} satisfying truth assignments) if and only if $(\psi, n-1)$ is a "yes" instance of Q. Consequently, Q is PP-complete.

We now show that there is a polynomial-time reduction of Q to #3SAT($\geqslant 2^{n/2}$). Given a 3CNF-formula ψ with variables $x_1, \ldots, x_n$ and a positive integer i, we can construct in polynomial time a 3CNF-formula χ with variables $x_1, \ldots, x_n, y_1, \ldots, y_n$ that is tautologically equivalent to the CNF-formula $\psi \wedge y_{n-i+1} \wedge \cdots \wedge y_n$. It is clear that $\#\chi = 2^{n-i}\#\psi$, where $\#\chi$ and $\#\psi$ denote the numbers of truth assignments that satisfy χ and ψ, respectively. Consequently, ψ has at least 2^i satisfying truth assignments if and only if χ has at least $2^n = 2^{2n/2}$ satisfying truth assignments. □

3. Upper and lower bounds for #kSAT($\geqslant 2^{n/t}$)

Let $X_k^{n,r}$ be the random variable on $F_k(n, r)$ such that $X_k^{n,r}(\varphi)$ is the number of truth assignments on $x_1, \ldots, x_n$ that satisfy φ, where φ is a random kCNF-formula in $F_k(n, r)$. Thus, φ is a "yes" instance of #kSAT($\geqslant 2^{n/t}$) if and only if $X_k^{n,r}(\varphi) \geqslant 2^{n/t}$. We now have all the notation in place to formulate the following conjecture for the family of problems #kSAT($\geqslant 2^{n/t}$), where $k \geqslant 3$ and $t \geqslant 2$.

Conjecture 2. *For every integer $k \geqslant 3$ and every integer $t \geqslant 2$, there is a positive real number $r_{k,t}$ such that*

- *If $r < r_{k,t}$, then* $\lim_{n\to\infty} \Pr[X_k^{n,r} \geqslant 2^{n/t}] = 1$.
- If $r > r_{k,t}$, then $\lim_{n\to\infty} \Pr[X_k^{n,r} \geqslant 2^{n/t}] = 0$.

We have not been able to settle this conjecture, which appears to be as difficult as the conjecture concerning phase transitions of random kSAT , $k \geqslant 3$. In what follows, however, we establish certain analytical results that yield upper and lower bounds for the value of $r_{k,t}$; in particular, these results demonstrate that the asymptotic behavior of random #kSAT($\geqslant 2^{n/t}$) is nontrivial.

3.1. Upper bounds for #kSAT($\geqslant 2^{n/t}$)

Let X be a random variable taking nonnegative values and having finite expectation $E(X)$. Markov's inequality is a basic result in probability theory which asserts that if s is a positive real number, then $\Pr[X \geqslant s] \leqslant (E(X)/s)$. The special case of this inequality with $s = 1$ has been used in the past to obtain a coarse upper bound for the critical ratio r_k in random k SAT . We now use the full power of Markov's inequality to obtain an upper bound for $r_{k,t}$. As usual, $\lg(x)$ denotes the logarithm of x in base 2.

Proposition 3. *Let $k \geqslant 3$ and $t \geqslant 2$ be two integers. For every positive rational number $r > ((t-1)/t)(1/(k - \lg(2^k - 1)))$,*

$$\lim_{n\to\infty} \Pr[X_k^{n,r} \geqslant 2^{n/t}] = 0.$$

It follows that if $r_{k,t}$ exists, then $r_{k,t} \leqslant ((t-1)/t)(1/(k - \lg(2^k - 1)))$. In particular, $r_{3,2} \leqslant \frac{1}{2}(1/(3 - \lg 7)) \approx 2.595$.

Proof. For every truth assignment α on the variables $x_1, \ldots, x_n$, let I_α be the random variable on $F_k(n, r)$ such that $I_\alpha(\varphi) = 1$, if α satisfies φ, and $I_\alpha(\varphi) = 0$, otherwise. Each I_α is a Bernoulli random variable with mean $(1 - 1/2^k)^{rn}$. Since $X_k^{n,r} = \Sigma_\alpha I_\alpha$, the linearity of expectation implies that $E(X_k^{n,r}) = (1 - 1/2^k)^{rn} 2^n$. By Markov's inequality, we have that

$$\Pr[X_k^{n,r} \geqslant 2^{n/t}] \leqslant (1 - 1/2^k)^{rn} 2^{(1-1/t)n}.$$

It follows that if r is such that $(1 - 1/2^k)^r 2^{(1-1/t)} < 1$, then $\lim_{n\to\infty} \Pr[X_k^{n,r} \geqslant 2^{n/t}] = 0$. The result then is obtained by taking logarithms in base 2 in both sides of the above inequality and solving for r. □

Several remarks are in order now. First, note if k is kept fixed while t is allowed to vary, then the smallest upper bound is obtained when $t = 2$. Moreover, the quantity $(1/(k - \lg(2^k - 1)))$ is the coarse upper bound for the critical ratio r_k for random kSAT obtained using Markov's inequality. In particular, for random 3SAT this bound is ≈ 5.91, which is twice the bound for $r_{3,2}$ given by Proposition 3.

Let MAJORITY kSAT be the restriction of MAJORITY SAT to kCNF-formulas, $k \geqslant 2$. Obviously, a formula φ in $F_k(n, r)$ is a "yes" instance of MAJORITY kSAT if and only if $X_k^{n,r}(\varphi) \geqslant 2^{n-1}$. Markov's inequality implies that $\Pr[X_k^{n,r} \geqslant 2^{n-1}] \leqslant 2(1 - 1/2^k)^{rn}$, from which it follows that $\lim_{n\to\infty} \Pr[X_k^{n,r} \geqslant 2^{n-1}] = 0$, for every $k \geqslant 2$. Thus, for every $k \geqslant 2$, the asymptotic behavior of random MAJORITY kSAT is trivial; in particular, MAJORITY kSAT does not undergo any phase transition. It is also worth noting that, although MAJORITY SAT is PP-complete, it is not known whether there is an integer $k \geqslant 3$, such that MAJORITY k-SAT is PP-complete.

3.2. Lower bounds for #kSAT($\geqslant 2^{n/t}$)

We say that a partial truth assignment α *covers* a clause c if it satisfies at least one of the literals of c. We also say that α *covers* a CNF-formula φ with n variables if α covers every clause of φ. Perhaps the simplest sufficient condition for φ to have at least $2^{n/t}$ satisfying truth assignments is to ensure that there is a partial assignment over $\lfloor n - n/t \rfloor$ variables covering φ. The next proposition shows that if r is small enough, then this sufficient condition is almost surely true for formulas in $F_k(n, r)$, as $n \to \infty$.

Proposition 4. *Let $k \geqslant 3$ and $t \geqslant 2$ be two integers. If $0 < r < 1 - 1/t$, then, as $n \to \infty$, almost all formulas in $F_k(n, r)$ are covered by a partial truth assignment on $\lfloor n - n/t \rfloor$ variables. Consequently, if $0 < r < 1 - 1/t$, then*

$$\lim_{n\to\infty} \Pr[X_k^{n,r} \geqslant 2^{n/t}] = 1.$$

It follows that if $r_{k,t}$ exists, then $r_{k,t} \geqslant 1 - 1/t$. In particular, $r_{3,2} \geqslant 0.5$.

Proof. In [10,11,14], it was shown that if $r < 1$, then 2CNF-formulas in $F_{2,n,r}$ are satisfiable with asymptotic probability 1. Fix a ratio $r < 1 - 1/t$ and consider a random formula φ in $F_k(n, r)$. By removing $(k - 2)$ literals at random from every clause of φ, we obtain a random 2CNF-formula φ^* which is almost surely satisfiable. Let β be a satisfying truth assignment of φ^* and let α be the partial truth assignment obtained from β by taking for each clause a literal satisfied by α. Since $r < 1 - 1/t$, we have that α is a truth assignment on $\lfloor n - n/t \rfloor$ variables that covers φ^*; hence, α covers φ as well. □

The preceding Propositions 3 and 4 imply that, unlike MAJORITY kSAT, for every $k \geqslant 3$ and every $t \geqslant 2$, the asymptotic behavior of #kSAT($\geqslant 2^{n/t}$) is nontrivial.

3.3. An improved lower bound for #3SAT($\geqslant 2^{n/2}$)

In what follows, we focus on #3SAT($\geqslant 2^{n/2}$). So far, we have established that if $r_{3,2}$ exists, then $0.5 \leqslant r_{3,2} \leqslant 2.595$. The main result is an improved lower bound for $r_{3,2}$.

Theorem 5. *For every positive real number* $r \leqslant 0.9227$,

$$\lim_{n\to\infty} \Pr[X_3^{n,r} \geqslant 2^{n/2}] > 0.$$

It follows that if $r_{3,2}$ *exists, then* $r_{3,2} \geqslant 0.9227$.

In the remainder of this section, we first discuss the methodology used and then present the Proof of Theorem 5. We adopt an algorithmic approach, which originated in [9] and has turned out to be very fruitful in establishing a lower bound for the critical ratio r_3 of random 3SAT (see [2] for an overview). We consider a particular randomized algorithm, called EUC, that takes as a input a 3CNF-formula φ on n variables and attempts to construct a *small* partial assignment α covering all clauses of φ. Algorithm EUC succeeds if a covering partial assignment α is produced which assigns values to at most $\leqslant \lfloor n/2 \rfloor$ variables, and fails otherwise. Our goal is to show that algorithm EUC succeeds with positive probability (not tending to 0 as $n \to \infty$) on formulas φ from $F_3(n, r)$ for each $r \leqslant 0.9227$. Consequently, $r_{3,2} \geqslant 0.9227$.

EUC Algorithm:
For $t := 1$ to n do
  If there are any 1-clauses, (*forced* step)
    pick a 1-clause uniformly at random and satisfy it.
  Otherwise, (*free* step)
    pick an unassigned variable uniformly at random and remove all
    literals involving that variable in all remaining clauses.
  Return *true* if no 0-clause (contradiction) has been generated and
      the number of assigned variables is $\leqslant \lfloor n/2 \rfloor$;
    otherwise, return *false*.

Note that if algorithm EUC returns *true*, then the assigned variables form a covering partial assignment. To analyze the average performance of algorithm EUC we use the *differential equations methodology* (DEM), initially introduced in [1] and described more extensively in [2].

Since DEM was developed to analyze satisfiability testing algorithms, it should not be surprising that certain modifications are needed so that it can be applied to counting algorithms, such as EUC. The main component of EUC not handled directly by DEM is the *free step*, since in a satisfiability context it is always a better strategy to assign a value to the selected variable, instead of removing all the literals involving that variable. We will describe *where* and *how* we extend DEM to handle free steps.

Let $V(t)$ be the random set of variables remaining at iteration $t\,(0 \leqslant t \leqslant n)$ and let $S_i(t)$ denote the set of random i-clauses $(0 \leqslant i \leqslant 3)$ remaining at iteration t. To trace the value of $|S_i(t)|$, we rely on the assumption that, at every iteration of the execution of the algorithm being considered, a property called *uniform randomness* is maintained. This property asserts that at every iteration $0 \leqslant t \leqslant n$, conditional on $|V(t)|=n'$ and $|S_i(t)|=m'$, $S_i(t)$ is drawn from $F_i(n', m')$ on the variables in $V(t)$. In [2], a protocol, called *card game*, is presented; this protocol restricts the possible ways in which a variable can be selected and assigned. It is shown that any algorithm obeying that protocol satisfies the uniform randomness property.

Due to the presence of the free steps, algorithm EUC does not satisfy the card game protocol. Let us first describe briefly the card protocol, as introduced in [2]. Then we will define an extension of the card game suitable for the EUC algorithm.

At each iteration of the card game protocol, a variable is selected and set to a value *true* or *false*. After the value of a variable has been fixed, the set of clauses is modified accordingly: that is, every clause satisfied by the assignment is removed and every literal falsified by the assignment is removed in the remaining clauses. The card game specifies some restrictions in the way a variable can be selected and set. More precisely, the satisfiability algorithm does not have complete access to the formula (set of clauses). Instead the algorithm can only "see" for each clause a collection

of cards, each containing a literal, turned "face-down", and placed in a column. In order to select the variable, the algorithm is then allowed to do one of the following:

1. It can point to a particular card; in this case, some hypothetical intermediary will react by revealing all the cards carrying the literal whose underlying variable coincides with the variable of the literal occurring in the selected card.
2. It can name a variable; in this case, the reaction of the intermediary will be to flip every card that carries a literal whose underlying variable coincides with the named variable.

In either cases, the algorithm is then forced to select a value for the involved variable. At this point, the algorithm can only use the information at hand, that is, the state of the cards as shown to it. After the value has been set, the intermediary removes all the cards associated with satisfied clauses and falsified literals, and a new round of the game starts. Since the card game was originally conceived for satisfiability, it is not surprising than the only reasonable action to perform after a variable has been selected is to give it a value, since any other choice could only diminish our probabilities to find a satisfying assignment. However, if our goal is to certify that a large number of solutions exist by finding a small cover, then there is a legitimate second choice: to "save" the variable as unset, thus, removing all literals associated with that variable in the set of clauses. We will call *extended card game* the generalization of the ordinary card game in which this last possibility is also allowed. It is tedious, but straightforward, to show that the extended card game also guarantees the uniform randomness property.

We analyze the algorithm EUC by studying the evolution of the random variables that count the number of clauses in $S_i(t)$, $C_i(t) = |S_i(t)|$. We also need a random variable $F(t)$ that counts the number of variables assigned up to iteration t. We trace the evolution of $C_i(t)$ and $F(t)$ by using the following result from [28].

Theorem 6 (*Wormald, [28]*). *Let $X_1(t), \ldots, X_k(t)$ be random variables evolving jointly with $t = 1, \ldots, n$ such that at each iteration t, conditional on the history of the joint process $X_1(1), \ldots, X_k(1), \ldots, X_1(t), \ldots, X_k(t)$, the following properties hold*:

1. *The random variable $X_j(t)$ is bounded by Bn, for some B.*
2. *The random variable $\Delta X_j(t) = X_j(t+1) - X_j(t)$ with high probability (w.h.p.) is very close to its conditional expectation.*
3. *$\Delta X_j(t)$ evolves smoothly with t and $X_1(t), \ldots, X_k(t)$.*

Then the entire evolution of $X_j(t)$ will remain close to its mean path, that is, the path that $X_j(t)$ would follow if $\Delta X_j(t)$ was, at each iteration, the value of its conditional expectation. Furthermore, this mean path can be expressed as the set of solutions of a system of differential equations corresponding to the scaled version of the space state of the process obtained by dividing every parameter by n.

As discussed in [2], Wormald's Theorem guarantees that the value of the random variables considered differs in $o(n)$ from its mean path.

Unfortunately, we faced several obstacles when we attempted to apply Wormald's theorem to analyze algorithm EUC. In order to overcome these obstacles, we will modify algorithm EUC, making it suitable for application of Wormald's theorem, while keeping, at the same time, the algorithmic scheme behind EUC. To understand where the problem arises, we will start by tracing the evolution of the values of $C_i(t)$ and $F(t)$ under the application of forced and free steps.

Let $\mathbf{H}(t)$ be a random variable representing the history of $C_i(t)$, $0 \leqslant i \leqslant 3$ and $F(t)$ up to iteration t, (that is, $\mathbf{H}(t) = \langle C(0), \ldots, C(t)\rangle$, where $C(t) = (C_0(t), C_1(t), C_2(t), C_3(t), F(t))$). Here, the uniform randomness comes to play: since $S_i(t)$, $1 \leqslant i \leqslant 3$ distributes as $F_i(n-t, C_i(t))$, the expected number of clauses in $C_i(t)$ containing a given literal l, chosen uniformly among the literals with underlying variable in $V(t)$, is equal to $iC_i(t)/2(n-t)$. Furthermore, given a variable chosen uniformly among the variables in $V(t)$, the expected number of clauses in $C_i(t)$ containing a v or $\bar{v}$ is equal to $iC_i(t)/(n-t)$.

Consequently, for all $0 \leqslant t \leqslant n-3$, conditional on $\mathbf{H}(t)$, if $C_1(t) = 0$ (free step):

$$\begin{aligned}
&\mathbf{E}(\Delta C_3(t)|\mathbf{H}(t)) = -\frac{3C_3(t)}{n-t},\\
&\mathbf{E}(\Delta C_2(t)|\mathbf{H}(t)) = -\frac{2C_2(t)}{n-t} + \frac{3C_3(t)}{(n-t)},\\
&\mathbf{E}(\Delta C_1(t)|\mathbf{H}(t)) = \frac{2C_2(t)}{n-t},\\
&\mathbf{E}(\Delta C_0(t)|\mathbf{H}(t)) = 0,\\
&\mathbf{E}(\Delta F(t)|\mathbf{H}(t)) = 1.
\end{aligned}$$

To prove this set of difference equations we first claim that for every $0 \leqslant t \leqslant n-3$ the variable v selected in the free step is chosen uniformly among the variables in $V(t)$, which is a direct consequence of the definition of the EUC algorithm. Thus, the set of clauses "leaving" S_3 at turn t are precisely those containing v or $\bar{v}$. By the uniform randomness the expected number of such clauses is $3C_3(t)/(n-t)$. Every such clause, after removal of v or $\bar{v}$ becomes a 2-clause and consequently, ends up in $S_2(t+1)$; hence the positive term $3C_3(t)/(n-t)$ in the right side of $\mathbf{E}(\Delta C_2(t)|\mathbf{H}(t))$. Furthermore, there is also a stream of clauses from S_2 to S_1 (after removal of v or $\bar{v}$). By the uniform randomness, the expected value of the number of such clauses is $2C_2(t)/(n-t)$. Finally, since $S_1(t) = \emptyset$, the flow of clauses from S_1 to S_0 is null.

The difference equations for the case of forced step are derived similarly, obtaining that for all $0 \leqslant t \leqslant n-3$, conditional on $\mathbf{H}(t)$, if $C_1(t) \neq 0$ (forced step)

$$\begin{aligned}
&\mathbf{E}(\Delta C_3(t)|\mathbf{H}(t)) = -\frac{3C_3(t)}{n-t},\\
&\mathbf{E}(\Delta C_2(t)|\mathbf{H}(t)) = -\frac{2C_2(t)}{n-t} + \frac{3C_3(t)}{2(n-t)},\\
&\mathbf{E}(\Delta C_1(t)|\mathbf{H}(t)) = -\frac{C_1(t)}{n-t} + \frac{C_2(t)}{n-t},\\
&\mathbf{E}(\Delta C_0(t)|\mathbf{H}(t)) = \frac{C_1(t)}{2(n-t)},\\
&\mathbf{E}(\Delta F(t)|\mathbf{H}(t)) = 0.
\end{aligned}$$

We are now in a situation manifesting the several technical difficulties that prevent us from applying directly Worlmald's theorem in the analysis of algorithm EUC. All of these difficulties have to do with the smoothness condition (3) of Wormald's theorem which states, leaving technicalities aside, that for every function f, changing any of t, $F(t)$, $C_0(t)$, $C_2(t)$, $C_3(t)$ by $\mathrm{O}(f(n))$ should affect each $\Delta F(t)$, $\Delta C_o(t)$, $\Delta C_1(t)$, $\Delta C_2(t)$, $\Delta C_3(t)$ within $\mathrm{O}(f(n)/n)$. First, it is easy to observe that, as t gets close to n, the quantity $n-t$ gets small and then the smoothness condition is easily violated. Following Achlioptas [2], this problem is fixed by determining an iteration $t^* = \lfloor (1-\varepsilon)n \rfloor$ at which our algorithm will stop the iterative process. Then the algorithm deals with the remaining formula in a deterministic fashion.

Nonetheless, there are additional and deeper difficulties. First, due to the different nature of free and forced steps, knowing whether $C_1(t) = 0$ or not may affect $\Delta C_2(t)$, $\Delta C_1(t)$, $\Delta C_0(t)$, and $\Delta F(t)$ by $\Omega(1)$. Finally, notice also that knowing the value of $C_0(t)$ with $\mathrm{o}(n)$ precision, as guaranteed by Wormald's theorem, is rather useless since we need to now *exactly* whether $C_0(t) = 0$ in order to certify that EUC algorithm succeeds.

To settle technical difficulties of this nature, Achlioptas derived an elegant solution, inspired by a result from queuing theory, called the *lazy-server* lemma. This lemma states that if, instead of handling unit clauses deterministically as they appear, our algorithm handles them in a lazy randomized fashion, by taking care of unit clauses at iteration t with probability $p(t) < 1-\delta$, for some fixed $\delta > 0$, and performing a free step with probability $1-p(t)$, then we can guarantee that with positive probability (that is not tending to 0 as $n \to \infty$) the number of unit clauses remains bounded and that 0-clauses are not generated whenever $p(t)$ is bigger than the expected number of 1-clauses added to $S_1(t)$. Furthermore, if $p(t)$ is very close to the number of 1-clauses added to $S_1(t)$, then the probability that the algorithm with lazy-server policy tries to satisfy a unit clause and finds $S_1(t)$ empty tends to 0. Thus, we have a strategy

to decide whether to perform a free or forced step that does not depend on the actual content of $S_1(t)$ and that can be made arbitrarily 'close' to the deterministic selection performed in EUC.

We now modify algorithm EUC by incorporating the features described above and obtain the following algorithm

EUC with lazy-server policy:
For $t := 1$ to t^* do
  Set $U(t) = 1$ with probability $p(t)$
  If $U(t) = 1$
    (a) If there are 1-clauses,
      pick a 1-clause uniformly at random and satisfy it.
    (b) Otherwise
      pick an unset variable uniformly at random
      and assign it uniformly at random
  Otherwise
    Pick an unset variable uniformly at random
    and remove all literals with that underlying
    variable in all remaining clauses.
Find a minimal covering for the remaining clauses.
Return *true* if no 0-clause has been generated and
      the number of assigned variables is $\lfloor n/2 \rfloor$;
    otherwise, return *false*.

Notice than when the algorithm attempts to satisfy a unit clause and finds $S_1(t)$ empty, instead of doing something 'clever', it just satisfies a randomly selected literal. The reason for this is that, in this way, the dynamics of $C_2(t)$ and $F(t)$ are completely independent of the value $C_1(t)$, depending only on the random coin flips that determine $U(t)$.

Now we can apply Wormald's theorem to analyze algorithm EUC with lazy-server policy. First, we compute the difference equations that determine the expected value for the evolution of $C_2(t)$, $C_3(t)$ and $F(t)$, conditional on the history of the random variables considered up to iteration t. It is not necessary to trace the values of $C_1(t)$ and $C_0(t)$ as the lazy-server lemma takes care of it.

$$\mathbf{E}(\Delta C_3(t)|\mathbf{H}(t)) = -\frac{3C_3(t)}{n-t},$$
$$\mathbf{E}(\Delta C_2(t)|\mathbf{H}(t)) = -\frac{2C2(t)}{n-t} + p(t)\frac{3C_3(t)}{2(n-t)} + (1-p(t))\frac{3C_3(t)}{n-t},$$
$$\mathbf{E}(\Delta F(t)|\mathbf{H}(t)) = p(t),$$

with initial conditions $C_2(0) = 0$, $C_3(0) = rn$, $F(0) = 0$.

Each difference equation is obtained by adding the right value of the corresponding difference equation for the forced step in the EUC algorithm weighted (multiplied) by $p(t)$ and the right value of the corresponding difference equation for the free step multiplied by $1 - p(t)$.

It is time to fix the value of $p(t)$. At time t the expected number of clauses added to $S_1(t)$, is $(p(t)(C_2(t)/(n-t)) + (1-p(t))(2C_2(t)/(n-t))$. Thus, according to the lazy-server lemma any value such that

$$p(t) > p(t)\frac{C_2(t)}{n-t} + (1-p(t))\frac{2C_2(t)}{n-t}$$

would suffice. This inequality is solved by setting

$$p(t) = (1+\theta)\frac{(2C_2(t)/(n-t))}{1+C_2(t)/(n-t)},$$

with $\theta > 0$.

To obtain the set of differential equations associated with the process, as discussed in [2], we consider the scaled version of the process, x, $c_2(x)$, $c_3(x)$, $f(x)$ obtained by dividing every parameter t, $C_2(t)$, $C_3(t)$, $F(t)$ by n. We obtain the following differential equations:

$$\frac{dc_3}{dx} = -\frac{3c_3(x)}{1-x},$$
$$\frac{dc_2}{dx} = -\frac{2c_2(x)}{1-x} + p(x, c_2(x))\frac{3c_3(x)}{2(1-x)} + (1 - p(x, c_2(x))\frac{3c_3(x)}{1-x},$$
$$\frac{df}{dx} = p(x, c_2(x)),$$

with $p(x, c_2(x)) = (1+\theta)(2c_2(x)/(1-x))/(1+c_2(x)/(1-x))$ and initial conditions $c_3(0) = r$, $c_2(0) = 0$ and $f(0) = 0$. We solve the system numerically using the utility `dsolve` of mapple (it is easy to get $c_3(x) = r(1-x)^3$ analytically) setting, with foresight, $r = 0.9227$, $\varepsilon = 10^{-2}$ and $\theta = 10^{-5}$.

Now we are almost done. First, we have that $S_0(t) = \emptyset$ with positive probability; this can be seen by testing numerically that $p(x, c_2(x)) < 1 - 10^{-1}$, if $x \in [0, 1 - 10^{-2}]$, and appealing to the lazy-server lemma. Moreover, we get $f(1 - 10^{-2}) < 0.49978$ and, transforming this result back to the randomized space state, we can infer that w.h.p $F(t^*) < 0.49978n + o(n)$. Similarly, the number of remaining clauses at that iteration is w.h.p $C_2(t^*) + C_3(t^*) = c_2(1 - 10^{-2})n + c_3(1 - 10^{-2})n + o(n) < 0.00021n + o(n)$. It is easy to verify that the ratio of clauses to variables at iteration t^* is smaller than 1, and we can apply again the argument used to obtain the first naive lower bound of $\frac{1}{2}$ to guarantee that there exists a covering partial assignment with size $< 0.00021n$. Thus, by adding the previous quantities, we get $0.49999n < n/2$, which means that the algorithm succeeds.

The astute reader might wonder whether algorithm EUC (and its lazy-server variant) suffers from greed. Clearly, delaying the satisfaction of a 1-clause can only hurt our probabilities of succeeding; it is still conceivable, however, that when choices are available (that is, clauses with more than one literal are present), a wiser policy that does not prioritize the retention of unset variables could do better. Our alternative algorithm could, instead, try to satisfy as soon as possible the formula by satisfying a uniformly chosen literal in each step hoping that after setting $\lfloor n/2 \rfloor$ variables all the clauses are satisfied (covered), or even more generally, it could toss a q-coin, and decide the type of operation, to satisfy a uniformly chosen literal or to mark a uniformly chosen variable as unset, depending on the outcome of the q-coin. After analyzing these possibilities, we found that algorithm EUC dominates them even in the case in which a presumably more intelligent strategy is used in the process of selecting the literal satisfied in the case of absence of 1-clauses. The new algorithm, called EUC with majority, performs the following actions: if there are 1-clauses then pick one of them uniformly at random and satisfy it. When no 1-clauses are present and the outcome of a q-coin is 0, then select a variable v uniformly at random, and remove all the occurrences of v and $\bar{v}$ in the formula. If the outcome is 1, then pick a variable v at random and set it in such a way that minimizes the number of 3-clauses that become 2-clauses. Again the best performance is obtained when $q = 0$, in which case the algorithm behaves identically to EUC.

There is still room for improvement. One possibility is to design more sophisticated algorithms to find small coverings. Nevertheless, it is not difficult to see that this approach is limited. In fact, a direct application of the Markov bound shows that if $r \geqslant 1.4343$, then almost surely there does not exist a covering of size at most $\lfloor n/2 \rfloor$ of a formula drawn according to $F_3(n, rn)$. Hence, any attempt to find a lower bound close to the observed value 2.5 of $r_{3,2}$ must necessarily be based on a completely different approach.

4. Experimental results for #3SAT($\geqslant 2^{n/2}$)

Experiments were run for random 3CNF-formulas with 10, 20, 30 and 40 variables on a SUN Ultra 5 workstation. For each space we generated 1200 random 3CNF-formulas with sizes ranging from 1 to 200 clauses in length. Each clause was generated by randomly selecting three variables without replacement and then negating each of them with probability of $\frac{1}{2}$.

Our goal was to test the formulas for being "yes" instances of #3SAT($\geqslant 2^{n/2}$), that is, for having at least as many satisfying assignments as the square root of the total number of truth assignments. For this, we implemented a threshold DPLL algorithm by modifying the basic CDP algorithm in [5] to include tracking of lower and upper bounds on the count and early termination, if the threshold is violated by the upper bound or satisfied by the lower bound.

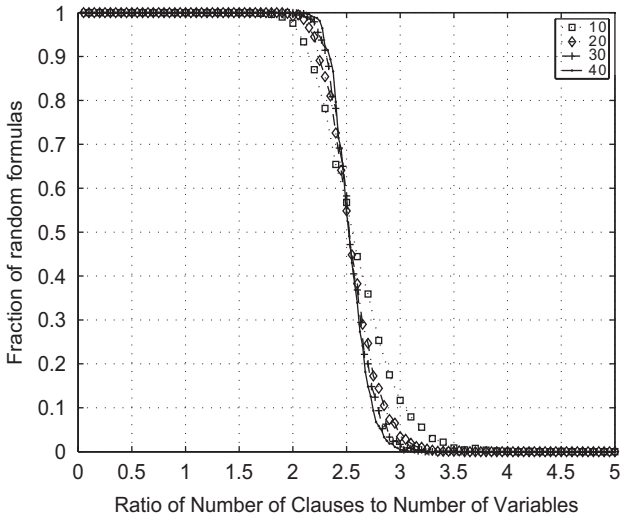

Fig. 1. Phase transition graphs.

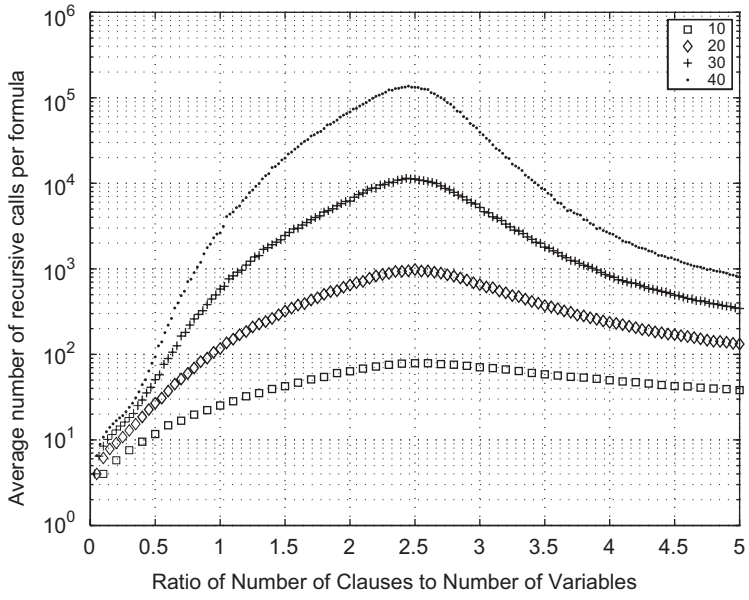

Fig. 2. Performance graphs.

The results are depicted in Figs. 1 and 2. In both figures the horizontal axis is the ratio of the number of clauses to number of variables in the space. The ranges of formula sizes represented in the graphs are 1–50, 1–100, 1–1500, and 1–200 for the 10, 20, 30 and 40 variable spaces, respectively.

The phase transition graphs show for each test point the fraction of 1200 newly generated random formulas that had a number of satisfying truth assignments greater than or equal to the square root of the total number of truth assignments. They strongly suggest that 2.5 is a critical ratio around which a phase transition occurs. The performance graphs show the average number of recursive calls required to test each formula and they exhibit a peak around the same ratio. In the test runs a range of 1–200 clauses was used for each space and the run times on the SUN Ultra 5 varied from several minutes to several hours.

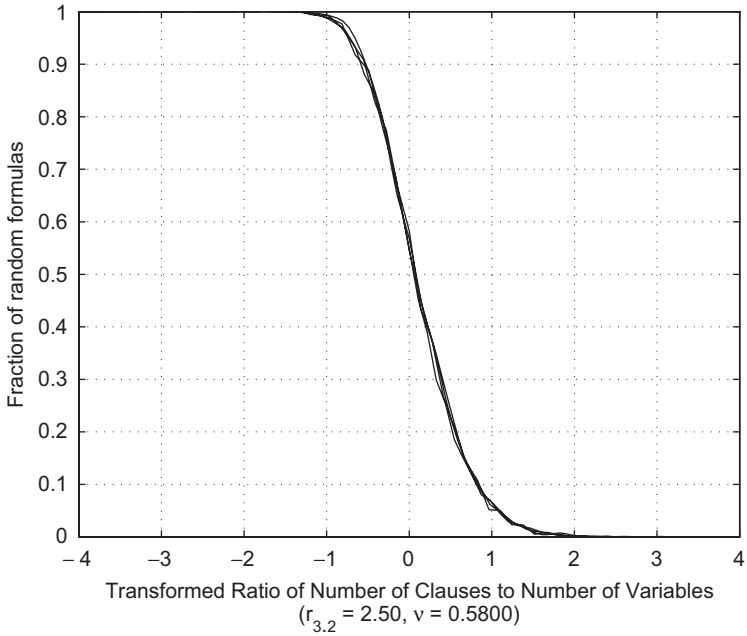

Fig. 3. Finite-size scaling results.

We also performed finite-size scaling, assuming a power law of the form $((r - r_{3,2})n^{\nu})/r_{3,2}$, to obtain estimates for the critical ratio $r_{3,2}$ and for the exponent ν. The estimates from the finite-size scaling are $r_{3,2} = 2.50$ and $\nu = 0.5800$. Fig. 3 shows evidence that the probability curves collapse to one curve when the re-scaling is applied which further supports the value of 2.5 for the critical ratio of the phase transition.

We conclude by pointing out that in Section 3.1.4 of [19] it was suggested that for every $0 \leqslant \alpha \leqslant 1$, the critical ratio of #3SAT($\geqslant 2^{\alpha n}$) is given approximately by the formula $4.2(1 - \alpha)$. By taking $\alpha = \frac{1}{2}$, this formula suggests that the critical ratio $r_{3,2}$ of #3SAT($\geqslant 2^{n/2}$) should be approximately 2.1, which is at odds with our experimental finding of 2.5 as the approximate value of $r_{3,2}$. We believe that this discrepancy is not caused by any significant difference in the outcome between the experiments carried out by Littman et al. [19] and ours, but rather is due to the way in which the above formula was extrapolated from the experiments in [19]. Specifically, in [19] experiments were carried out by varying α and the ratio r of clauses to variables, but keeping the number of variables to a fixed value $n = 30$. The above formula $4.2(1 - \alpha)$ was then derived by visual inspection of the resulting surface. We believe that, instead, the value of the critical ratio should be estimated by the crossover points of the curves obtained from experiments for different values of the number n of variables. In any case, we see no theoretical argument or experimental evidence that a linear relationship between the critical ratio and α should hold.

Acknowledgments

We are grateful to Dimitris Achlioptas for stimulating discussions and pointers to his work on lower bounds for 3SAT via differential equations. We also wish to thank Peter Young for generously sharing with us his expertise on phase transition phenomena, and Moshe Y. Vardi for listening to an informal presentation of the results reported here and offering valuable suggestions on an earlier draft of this paper.

References

[1] D. Achlioptas, 2000. Setting two variables at a time yields a new lower bound for random 3-SAT, in: 32nd Annual ACM Symposium on Theory of Computing, 2000, pp. 28–37.
[2] D. Achlioptas, Lower bounds for random 3-SAT via differential equations, Theoret. Comput. Sci. 265 (2001) 1–2.
[3] D. Angluin, On counting problems and the polynomial-time hierarchy, Theoret. Comput. Sci. 12 (1980) 161–173.

[4] R. Bayardo, J. Pehoushek, Counting models using connected components, in: seventeenth National Conference on Artificial Intelligence, 2000, pp. 157–162.
[5] E. Birnbaum, E. Lozinskii, The good old Davis–Putnam procedure helps counting models, J. Artif. Intel. Res. 10 (1999) 457–477.
[6] T. Bylander, The computational complexity of propositional STRIPS planning, Artif. Intel. 69 (1994) 161–204.
[7] T. Bylander, A probabilistic analysis of propositional STRIPS planning, Artif. Intel. 81 (1996) 241–271.
[8] M. Cadoli, A. Giovanardi, M. Schaerf, 1997. Experimental analysis of the computational cost of evaluating quantified Boolean formulas, in: Proceedings of 5th Conference of the Italian Association for Artificial Intelligence. Lecture Notes in Artificial Intelligence, vol. 218 Springer, Berlin, pp. 207–218.
[9] M.T. Chao, J. Franco, Probabilistic analysis of two heuristics for the 3-satisfiability problem, SIAM J. Comput. 15 (4) (1986) 1106–1118.
[10] V. Chvátal, B. Reed, Mick gets some (the odds are on his side), in: 33th Annual Symposium on Foundations of Computer Science, 1992, pp. 620–627.
[11] W. Fernandez de la Vega, On random 2-SAT, manuscript, 1992.
[12] I. Gent, T. Walsh, 1999. Beyond NP: the QSAT phase transition, in: Proceedings of 16th National Conference on Artificial Intelligence, pp. 653–658.
[13] J. Gill, Computational complexity of probabilistic Turing machines, SIAM J. Comput. 6 (4) (1977) 675–695.
[14] A. Goerdt, A threshold for unsatisfiability, J. Comput. System Sci. 53 (3) (1996) 469–486.
[15] A.C. Kaporis, L.M. Kirousis, E.G. Lalas, The probabilistic analysis of a greedy satisfiability algorithm, in: Proceedings of the Fifth International Symposium on the Theory and Application of Satisfiability Testing, SAT2002, 2000, pp. 362–376.
[16] A.C. Kaporis, L.M. Kirousis, Y.C. Stamatriou, M. Vamvadari, M. Zito, Coupon collectors, q-binomial coefficients and the unsatisfiability threshold, in: Proceedings of 7th Italian Conference, ICTCS 2001, Lecture Notes in Computer Science, Springer, Berlin, 2001, p. 328–338.
[17] M. Littman, Initial experiments in stochastic satisfiability, in: Proceedings of the 16th National Conference on Artificial Intelligence, 1999, pp. 667–672.
[18] M. Littman, J. Goldsmith, M. Mundhenk, The computational complexity of probabilistic planning, J. Artif. Intel. Res. 9 (1998) 1–36.
[19] M. Littman, S. Majercik, T. Pitassi, Stochastic Boolean satisfiability, J. Automat. Reason. 27 (3) (2001) 251–296.
[20] P. Orponen, Dempster's rule of combination is #-P-complete, Artificial Intelligence 44 (1990) 245–253.
[21] C.H. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.
[22] D. Roth, On the hardness of approximate reasoning, Artificial Intelligence 82 (1–2) (1996) 273–302.
[23] B. Selman, D.G. Mitchell, H.J. Levesque, Generating hard satisfiability problems, Artificial Intelligence 81 (1–2) (1996) 17–29.
[24] J. Simon, On some central problems in computational complexity Ph.D. Thesis, Cornell University, Computer Science Department, 1975.
[25] S. Toda, On the computational power of PP and $\oplus$P, in: Proceedings 30th IEEE Symposium on Foundations of Computer Science (FOCS'89), Research Triangle Park, North Carolina, USA, 1989, pp. 514–519.
[26] L.G. Valiant, The complexity of computing the permanent, Theoret. Comput. Sci. 8 (2) (1979) 189–201.
[27] L.G. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput. 8 (3) (1979) 410–421.
[28] N.C. Wormald, Differential equations for random processes and random graphs, Ann. Appl. Probab. 5 (4) (1995) 1217–1235.