Algorithms for Random 3-SAT

Abraham D. Flaxman, Microsoft Research

INDEX TERMS: Satisfiability, Random structures.

1 PROBLEM DEFINITION

This classic problem in complexity theory is concerned with efficiently finding a satisfying assignment to a propositional formula. The input is a formula with n Boolean variables which is expressed as an AND of ORs with 3 variables in each OR clause (a 3-*CNF formula*). The goal is to (1) find an assignment of variables to TRUE and FALSE so that the formula has value TRUE, or (2) prove that no such assignment exists. Historically, recognizing satisfiable 3-CNF formulas was the first "natural" example of an NP-complete problem, and, because it is NP-complete, no polynomial-time algorithm can succeed on all 3-CNF formulas unless P = NP [14, 31]. Because of the numerous practical applications of 3-SAT, and also due to its position as the canonical NP-complete problem, many heuristic algorithms have been developed for solving 3-SAT, and some of these algorithms have been analyzed rigorously on random instances.

Notation A 3-CNF formula over variables $x_1, x_2, \ldots, x_n$ is the conjunction of m clauses $C_1 \wedge C_2 \wedge \ldots \wedge C_m$, where each clause is the disjunction of 3 literals, $C_i = \ell_{i_1} \vee \ell_{i_2} \vee \ell_{i_3}$, and each literal ℓ_{i_j} is either a variable or the negation of a variable (the negation of the variable x is denoted by $\overline{x}$). A 3-CNF formula is *satisfiable* if and only if there is an assignment of variables to truth values so that every clause contains at least one true literal. Here, all asymptotic analysis is in terms of n, the number of variables in the 3-CNF formula, and a sequence of events $\{\mathcal{E}_n\}$ is said to hold *with high probability* (abbreviated **whp**) if $\lim_{n\to\infty} \Pr[\mathcal{E}_n] = 1$.

Distributions There are many distributions over 3-CNF formulas which are interesting to consider, but research to date has focused primarily on (1) sparse satisfiable instances, (2) dense satisfiable instances, and (3) dense unsatisfiable instances.

This attention can be attributed to the behavior of an instance with n variables, consisting of m random clauses selected independently and uniformly from all triples of literals (this distribution of instances will be denoted $\mathbb{I}_{n,m}$). The probability of satisfiability for instances drawn from this distribution has an interesting and mysterious relationship with the ratio of clauses to variables (the *clause density*). When the ratio is low, the instances are quite likely to be satisfiable (an example of (1), the sparse satisfiable case), while when the ratio is high, the instances are unsatisfiable with overwhelming probability (an example of (3), the dense unsatisfiable case). This phenomenon is treated in detail in Chapter **??** of this volume.

A very similar distribution which sometime leads to more convenient computation is the n variable instances where each triple of literals appears as a clause independently with probability p. This distribution will be denoted here by $\mathbb{I}_{n,p}$.

Dense satisfiable instances can be formed by conditioning on the event $\{I_{n,m}$ is satisfiable$\}$, but this conditional distribution is difficult to sample from and to analyze. This has led to research in "planted" random instances of 3-SAT, which are formed by first choosing a truth assignment ϕ uniformly at random, and then selecting each clause independently from the triples of literals where at least one literal is set to TRUE by the assignment ϕ. The clauses can be included with equal probabilities in analogy to the $\mathbb{I}_{n,p}$ or $\mathbb{I}_{n,m}$ distributions above [29, 30], or different probabilities can be assigned to the clauses with one, two, or three literals set to TRUE by ϕ, in an effort to better hide the satisfying assignment [3, 21].

A particularly appealing modification of the planted random distribution is to the semirandom instance, where a planted random instance is perturbed by an adversary who is constrained to add only clauses which are consistent with the planted satisfying assignment and not to delete anything [20].

Problem 1 (3-SAT)**.**
INPUT*: 3-CNF Boolean formula* $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$*, where each clause* C_i *is of the form* $C_i = \ell_{i_1} \vee \ell_{i_2} \vee \ell_{i_3}$*, and each literal* ℓ_{i_j} *is either a variable or the negation of a variable.*
OUTPUT*: A truth assignment of variables to Boolean values which makes at least one literal in each clause TRUE, or a certificate that no such assignment exists.*

2 KEY RESULTS

It is convenient to consider three categories of heuristics for 3-SAT, based on the type of input distribution for which the heuristic is likely to be successful.

Algorithms for sparse satisfiable instances In the analysis of heuristics, it is often difficult to cope with the conditioning introduced by backtracking algorithms. However, there are some simple heuristics which make no use of backtracking that are likely to succeed on sparse random instances, that is, instances of 3-SAT where the ratio of clauses to variables is sufficiently small.

The *Pure Literals Heuristic (PL)* is one approach which has no backtracking. It functions as follows: Repeat the following: if the formula contains a literal and not the negation of that literal, set this literal to TRUE and remove all clauses containing the literal. Otherwise set a randomly chosen literal to TRUE, remove all the clauses it satisfies, and remove its negation from all the clauses in which it appears. If a clause become empty, halt and declare failure. Otherwise, when no clauses are left, return the satisfying assignment generated. The PL heuristic has been analyzed by many researchers, and the following result appears in [32] and is proved rigorously in [28, 33]:

Theorem 1. *There exists a constant* $c_{PL} \approx 1.637$*, such that for any constant* $c < c_{PL}$*,* ***whp*** *the PL heuristic succeeds on* $I_{n,cn}$*, while for any constant* $c > c_{PL}$*,* ***whp*** *the PL heuristic fails on* $I_{n,cn}$*.*

The *Generalized Unit Clause Heuristic (GUC)* is an alternative approach which does not employ backtracking, and works in the following manner: Repeat the following: choose a clause C uniformly at random from all clauses of shortest length, and choose a literal ℓ uniformly at random from C. Set the variable corresponding to ℓ so that C is satisfied, remove all clauses in the instance in which ℓ appears, and remove $\overline{\ell}$ from all the clauses in which it appears. If a clause become empty, halt and declare failure. Otherwise, when no clauses are left, return the satisfying assignment generated. The GUC heuristic has been analyzed by many researchers, eventually leading to the following result from [24]:

Theorem 2. *There exists a constant* $c_{GUC} \approx 3.003$ *such for any* $c < c_3$*, the GUC heuristic succeeds on* $I_{n,cn}$ *with asymptotically positive probability, while for any* $c \geq c_3$*, the GUC heuristic fails* ***whp****.*

Myopic algorithms are a class of heuristics which are defined formally in [1]. They generalize this backtracking-free approach. These algorithms and their close relatives in [26, 27] provide a popular approach to rigorously proving lower bounds on the satisfiability threshold (which is currently known to be at least 3.52). They are treated in detail in Chapter **??** of this volume.

Random Walk SAT (RWalkSAT): Unlike the previous algorithms, RWalkSAT is a local search heuristic which does not commit to any of its decisions on how to set a variable. The algorithm begins with an arbitrary truth assignment, and, while the assignment does not satisfy all clauses, the algorithm chooses an unsatisfied clause uniformly at random, then chooses a variable in that clause uniformly at random, and then flips the value of the variable in the truth assignment. Local search heuristics are usually difficult to analyze on random instances, and RWalkSAT is no exception. The following theorem, from [2], works by relating RWalkSAT to the Pure Literal Heuristic:

Theorem 3. *For any constant $c < c_{PL} \approx 1.637$, **whp** heuristic RWalkSAT succeeds on $I_{n,cn}$.*

Experimental results indicate that RWalkSAT also succeeds for larger values of c, up to $c \approx 2.65$ [34]. Experimental success has also been reported in combining RWalkSAT with the hill climbing algorithm *GSAT*, which starts with a random assignment and repeatedly flips the value of the variable which results in the greatest increase in the number of clauses satisfied [35] (The combination of RWalkSAT and GSAT is called *WalkSAT* in the literature, and a careful combination has shown experimental success for values of c to just below 4.2).

Survey Propagation is a very exciting new approach for random instances of 3-SAT. Algorithms which use survey propagation have not been proven to succeed on random instances drawn from $\mathbb{I}_{n,cn}$, but experimental results suggest that some will succeed for values of c very close to the conjectured satisfiability threshold [7]. The survey-propagation-based algorithm *Survey Inspired Decimation* is described in the EXPERIMENTAL RESULTS section below.

Algorithms for dense unsatisfiable instances A simple calculation shows that **whp** $I_{n,cn}$ is unsatisfiable for $c > 1/\log_2(8/7) \approx 5.191$ (and better bounds are known, see Chapter **??** of this volume). However, no algorithm is known which will certify the unsatisfiability of such instances in polynomial time for any $c \ll \sqrt{n}$. The hypothesis that no such algorithm exists (or a related, weaker hypothesis) forms the basis for worst-case hardness-of-approximation results in [16].

One approach to refuting unsatisfiable instances of 3-SAT is *resolution*, which carefully simplifies the instance by combining clauses according to the rule $(x \vee C_1) \wedge (\bar{x} \vee C_2) \implies (C_1 \vee C_2)$. This is not an entire specification of an algorithm, and there is a lot of flexibility in how to choose the clauses to combine. However, no polynomial-time algorithm will be successful **whp**, because **whp** the shortest resolution proof has length exponential in the number of variables [4–6, 10].

Theorem 4. *For any $0 \leq \epsilon \leq 1/2$, there exists a constant C_ϵ such that for any $c = c(n) \geq 1$, any resolution proof that $I_{n,cn}$ is unsatisfiable has size at least $\exp\left\{C_\epsilon n/c^{2-\epsilon}\right\}$.*

When the clause density is tending to ∞ sufficiently rapidly, two approaches have been shown to succeed **whp**.

The first is *the Davis-Putnam-Logemann-Loveland procedure (DPLL)*, which is closely related to the resolution proof system above, and performs a recursive search for a satisfying assignment by considering separately the effects of setting a variable to TRUE and to FALSE. When the recursive application of the search results in a clause with all literals set to FALSE, the branch terminates. In the version of DPLL analyzed in [4], (called *ordered DLL* there), the next variable to set is selected uniformly from the variables in unit clauses if such a clause exists, and according to a fixed order on all variables otherwise. In [4], it is proved that

Theorem 5. *Ordered DLL certifies the unsatisfiability of $I_{n,cn}$ in time $2^{O(n/c)}n^{O(1)}$ **whp**. Thus ordered DLL runs in polynomial time **whp** for $c = \Omega(n/\log n)$.*

Spectral refutation algorithms provide an alternative approach which have been proven to work **whp** for lower clause density. These algorithms work by constructing certain graphs based on the 3-CNF formulas and then calculating the eigenvalues of these graphs, which must satisfy certain inequalities if the formula is satisfiable. The approach which currently is proven to work for the widest range of instances is from [19], which works for instances $I_{n,cn^{3/2}}$ whenever c exceeds a sufficiently large constant. For simplicity, however, the algorithm which follows is a simpler spectral refutation approach from [22].

Construct a graph $G = (V, E)$ from the instance as follows. The vertex set V consists of ordered pairs of variables, $V = \{(x_i, x_j) : i, j \in [n]\}$. The edge set E consists of all edges $\{(a_1, b_1), (a_2, b_2)\}$ such that there exists a variable z so that $a_1 \vee a_2 \vee z$ is a clause in I and $b_1 \vee b_2 \vee \overline{z}$ is also a clause in I. Similarly construct a graph $H = (V, F)$ from the instance, with the same vertex set, but with edge set F consisting of all edges $\{(a_1, b_1), (a_2, b_2)\}$ such that there exists a variable z so that that $\overline{a_1} \vee \overline{a_2} \vee z$ is a clause in I and $\overline{b_1} \vee \overline{b_2} \vee \overline{z}$ is also a clause in I. Then construct the matrix M_G defined by $M_G(i, j) = 1$ if $\{i, j\} \notin E$ and $M_G(i, j) = -\frac{1-p}{p}$ otherwise. Construct the matrix M_H similarly. Calculate the largest eigenvalue of M_G and M_H, and if both values are less than $n^2/4$, this constitutes a certificate that the instance is not satisfiable (see [23] or the subsequent work in [12, 19, 25] for additional details).

Theorem 6. *For $0 < \gamma < 1/2$, for $I_{n,p}$ with $p = n^{-(1+\gamma)}$, **whp** $\lambda_1(M_G)$ and $\lambda_1(M_H) \le n^2/4$, and hence a spectral refutation heuristic certifies that $I_{n,p}$ is unsatisfiable **whp**.*

Whp random instances $I_{n,m}$ with $m > cn^{7/5}$ contain a polynomial length certificate of unsatisfiability, however it is currently not known how to find such a certificate in time less than $2^{O(n^{0.2} \log n)}$ [17].

Algorithms for dense satisfiable instances Dense satisfiable instances can be formed by conditioning on the event $\{I_{n,m}$ is satisfiable$\}$, and results for this distribution are reported in [9] and in the conclusion of [13]. Besides these works, the conditionally satisfiable distribution has resisted analysis. This has led to research in "planted" random instances of 3-SAT, which are formed by first choosing a truth assignment ϕ uniformly at random, and then selecting each clause independently from the triples of literals where at least one literal is set to TRUE by the assignment ϕ. When random instances are generated with a planted satisfying assignment, the planted assignment together with the instance constitutes a candidate for a one-way function. This potential application to cryptography, as well as the basic interest in identifying hard problems, has motivated the development of algorithms for 3-SAT which are known to work on planted random instances.

Majority Vote Heuristic: If every clause consistent with the planted assignment is included with the same probability, then there is a bias towards including the literal satisfied by the planted assignment more frequently than its negation. This is the motivation behind the Majority Vote Heuristic, which assigns each variable to the truth value which will satisfy the majority of the clauses in which it appears. Despite its simplicity, this heuristic has been proven successful **whp** for sufficiently dense planted instances [29].

Theorem 7. *When c is a sufficiently large constant and $I \sim \mathbb{I}^{\phi}_{n,cn\log n}$, **whp** the majority vote heuristic finds the planted assignment ϕ.*

When the density of the planted random instance is lower than $c \log n$, then the majority vote heuristic will fail, and if the relative probability of the clauses satisfied by one, two, and three literals are adjusted appropriately then it will fail miserably. But there are alternative approaches.

For planted instances where the density is a sufficiently large constant, the majority vote heuristic provides a good starting assignment, and then the *k-OPT heuristic* can finish the job. The k-OPT heuristic of [20] is defined as follows: Initialize the assignment by majority vote. Initialize k to 1. While there exists a set of k variables for which flipping the values of the assignment will (1) make false clauses true and (2) will not make true clauses false, flip the values of the assignment on these variables. If this reaches a local optimum that is not a satisfying assignment, increase k and continue.

Theorem 8. *When c is a sufficiently large constant and $I \sim I^{\phi}_{n,cn}$ the k-OPT heuristic finds a satisfying assignment in polynomial time* **whp***. The same is true even in the semirandom case, where an adversary is allowed to add clauses to I that have all three literals set to TRUE by ϕ before giving the instance to the k-OPT heuristic.*

A related algorithm has been shown to run in expected polynomial time in [30], and a rigorous analysis of *Warning Propagation (WP)*, a message passing algorithm related to Survey Propagation, has shown that WP is successful **whp** on planted satisfying assignments, provided that the clause density exceeds a sufficiently large constant [18]. When the relative probabilities of clauses containing one, two, and three literals are adjusted carefully, it is possible to make the majority vote assignment very different from the planted assignment. A way of setting these relative probabilities that is predicted to be difficult is discussed in [3]. If the density of these instances is high enough (and the relative probabilities are anything besides the case of "Gaussian elimination with noise"), then a spectral heuristic provides a starting assignment close to the planted assignment and local reassignment operations are sufficient to recover a satisfying assignment [21].

3 APPLICATIONS

Some applications of algorithms for random 3-SAT have already been mentioned, including proving lower bounds on the satisfiability threshold (a major open problem in probabilistic combinatorics, treated in Chapter **??** of this volume), understanding the security of using planted instances as one-way functions, and developing further conditional hardness-of-approximation results in worst-case complexity theory.

Another intriguing application is based on the experimental observation that, for a fixed number of variables, DPLL algorithms take the most time to solve instances for which the clause density appears to be equal to the satisfiability threshold [8, 15, 36]. This has led to speculation that there is a connection between phase transitions and computational difficulty. However, evidence of this relationship is inconclusive (see, for example, [11]).

Additionally, 3-SAT is a universal problem, and due to its simplicity, it has potential applications in many other areas, including proof theory and program checking, planning, cryptanalysis, machine learning, and modeling biological networks.

4 OPEN PROBLEMS

Several open problems in algorithms for random 3-SAT have already been mentioned; find better bounds on the satisfiability threshold, determine whether refuting random instances of constant density is computationally intractable, developing distributions over instances-and-satisfying-assignments so that the instance is hard to satisfy if the assignment is kept secret, and analyzing RWalkSAT and other backtracking algorithms for clause density closer to the satisfiability threshold.

Another important direction is to develop alternative models of random distributions which more accurately reflect the type of instances that occur in the real world.

Finally, the rigorous theoretical analysis of the survey-propagation-based heuristics presents an important new challenge.

5 EXPERIMENTAL RESULTS

This section describes the survey-propagation-based algorithm, *Survey Inspired Decimation (SID)* which is based on considerations from statistical mechanics and works in experiments for sparse random instances with clause density close to the satisfiability threshold. In many ways, SID is similar to the other algorithms for sparse satisfiable instances which do not use any backtracking. It works by selecting a variable to fix to a truth value and then reducing the formula to reflect this decision. The decision of which variable to fix is made after finding a stable solution to a system of nonlinear dynamics described below. In experiments, it has been helpful to use SID to fix a certain fraction of the variables and then use the local search algorithm WalkSAT to find a satisfying assignment for the remaining formula.

To determine which variable to fix, SID runs a iterative procedure (survey propagation) to calculate values $\eta_{a\to i}$ for every clause a and every variable i which appears in a. For brevity, below is an equivalent formulation of the update rule, in terms of $\gamma_{C\to\ell}$, where C is a clause and ℓ is a literal. This rearranging of terms may remove the physical intuition which motivates the algorithm. For details on the intuition, see [7].

To determine which variable to fix, do the following: for each literal occurrence ℓ in each clause C, initialize $\gamma_{C\to\ell}$ to a random value, independently and uniform in $[0,1]$. Then iterate through the literal occurrences according to a random permutation, updating the value of $\gamma_{C\to\ell}$, for $C = j \vee k \vee \ell$ according to the rule

$$\gamma_{C\to\ell} = 1 - \left(1 - \frac{Z_{\bar{j}}}{Z_{\bar{j}} + Z_j^C - Z_{\bar{j}} Z_j^C}\right)\left(1 - \frac{Z_{\overline{k}}}{Z_{\overline{k}} + Z_k^C - Z_{\overline{k}} Z_k^C}\right),$$

where

$$Z_\ell = \prod_{C' : \ell \in C'} \gamma_{C'\to\ell}, \qquad \text{and} \qquad Z_\ell^C = \prod_{\substack{C' : \ell \in C', \\ C' \neq C}} \gamma_{C'\to\ell},$$

with the empty product is defined to be 1. (When dealing with a two-literal clause $C = k \vee \ell$, use the rule $\gamma_{C\to\ell} = \frac{Z_{\overline{k}}}{Z_{\overline{k}} + Z_k^C + Z_{\overline{k}} Z_k^C}$.) Repeat this until the largest change in any $\gamma_{C\to\ell}$ is less than some threshold value ϵ, and then find the variable x for which the magnitude of $\frac{-Z_x + Z_{\overline{x}}}{Z_x + Z_{\overline{x}} + Z_x Z_{\overline{x}}}$ is maximized, and set this variable to TRUE if the sign is positive and FALSE if the sign is negative. (If the absolute value is less than ϵ for all variables, run WalkSAT.) Clean the formula by removing the variable just set, and then iteratively removing any unit clauses generated, and then return to iterating the $\gamma_{C\to\ell}$ values (the sample code available does not reseed the $\gamma_{C\to\ell}$ values for the remaining formula).

6 DATA SETS

Sample instances of satisfiability and 3-SAT are available on the web at `http://www.satlib.org/`.

7 URL to CODE

Solvers and information on the annual satisfiability solving competition are available on the web at `http://www.satlive.org/`.

Source code for Survey Propagation is at `http://www.ictp.trieste.it/~zecchina/SP/`

8 CROSS REFERENCES

satisfiability threshold chapter

9 RECOMMENDED READING

References

[1] Achlioptas, D., and Sorkin, G. B. Optimal myopic algorithms for random 3-SAT. In *41st Annual Symposium on Foundations of Computer Science* (2000), IEEE Comput. Soc. Press, Los Alamitos, CA, pp. 590–600.

[2] Alekhnovich, M., and Ben-Sasson, E. Linear upper bounds for random walk on small density random 3-cnf. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science* (Washington, DC, USA, 2003), IEEE Computer Society, p. 352.

[3] Barthel, W., Hartmann, A. K., Leone, M., Ricci-Tersenghi, F., Weigt, M., and Zecchina, R. Hiding solutions in random satisfiability problems: A statistical mechanics approach. *Phys. Rev. Lett. 88* (2002), 188701.

[4] Beame, P., Karp, R., Pitassi, T., and Saks, M. On the complexity of unsatisfiability proofs for random k-CNF formulas. In *Proceedings of the 30th annual ACM symposium on Theory of computing* (1998), ACM Press, pp. 561–571.

[5] Ben-Sasson, E. *Expansion in Proof Complexity.* PhD thesis, Computer Science Department, Hebrew University, Jerusalem, 2001.

[6] Ben-Sasson, E., and Wigderson, A. Short proofs are narrow — resolution made simple. *J. ACM 48*, 2 (2001), 149–169.

[7] Braunstein, A., Mézard, M., and Zecchina, R. Survey propagation: an algorithm for satisfiability. *Random Structures Algorithms 27*, 2 (2005), 201–226.

[8] Cheeseman, P., Kanefsky, B., and Taylor, W. M. Where the Really Hard Problems Are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, IJCAI-91, Sidney, Australia* (1991), pp. 331–337.

[9] Chen, H. An algorithm for SAT above the threshold. In *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers* (2003), pp. 14–24.

[10] Chvátal, V., and Szemerédi, E. Many hard examples for resolution. *J. Assoc. Comput. Mach. 35*, 4 (1988), 759–768.

[11] Coarfa, C., Demopoulos, D. D., San Miguel Aguirre, A., Subramanian, D., and Vardi, M. Y. Random 3-SAT: the plot thickens. *Constraints 8*, 3 (2003), 243–261. Special issue of the Sixth International Conference on Principles and Practice of Constraint Programming (Singapore, 2000).

[12] Coja-Oghlan, A., Goerdt, A., and Lanka, A. Strong refutation heuristics for random k-SAT. In *APPROX-RANDOM* (2004), K. Jansen, S. Khanna, J. D. P. Rolim, and D. Ron, Eds., vol. 3122 of *Lecture Notes in Computer Science*, Springer, pp. 310–321.

[13] Coja-Oghlan, A., Krivelevich, M., and Vilenchik, D. Almost all k-colorable graphs are easy. unpublished manuscript, 2006.

[14] Cook, S. The complexity of theorem-proving procedures. In *Proc. 3rd FOCS* (1971), pp. 151–158.

[15] Crawford, J. M., and Auton, L. D. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence 81*, 1-2 (1996), 31–57. Frontiers in problem solving: phase transitions and complexity.

[16] Feige, U. Relations between average case complexity and approximation complexity. In *Proc. 34th ACM STOC* (2002).

[17] Feige, U., Kim, J. H., and Ofek, E. Witnesses for non-satisfiability of dense random 3CNF formulas. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science* (Washington, DC, USA, 2006), IEEE Computer Society.

[18] Feige, U., Mossel, E., and Vilenchik, D. Complete convergence of message passing algorithms for some satisfiability problems. In *Proceedings of APPROX-RANDOM* (2006), pp. 339–350.

[19] Feige, U., and Ofek, E. Easily refutable subformulas of large random 3CNF formulas. In *Automata, languages and programming*, vol. 3142 of *Lecture Notes in Comput. Sci.* Springer, Berlin, 2004, pp. 519–530.

[20] Feige, U., and Vilenchik, D. A local search algorithm for 3-SAT. Tech. rep., The Weizmann Institute, 2004.

[21] Flaxman, A. D. A spectral technique for random satisfiable 3CNF formulas. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (Baltimore, MD, 2003)* (New York, 2003), ACM, pp. 357–363.

[22] Friedman, J., and Goerdt, A. Recognizing more unsatisfiable random 3-SAT instances efficiently. In *Automata, languages and programming*, vol. 2076 of *Lecture Notes in Comput. Sci.* Springer, Berlin, 2001, pp. 310–321.

[23] Friedman, J., Goerdt, A., and Krivelevich, M. Recognizing more unsatisfiable random k-SAT instances efficiently. *SIAM J. Comput. 35*, 2 (2005), 408–430.

[24] Frieze, A. M., and Suen, S. Analysis of two simple heuristics on a random instance of k-SAT. *J. Algorithms 20*, 2 (1996), 312–355.

[25] Goerdt, A., and Lanka, A. Recognizing more random unsatisfiable 3-SAT instances efficiently. In *Typical case complexity and phase transitions*, vol. 16 of *Electron. Notes Discrete Math.* Elsevier, Amsterdam, 2003, p. 26 pp. (electronic).

[26] Hajiaghayi, M., and Sorkin, G. B. The satisfiability threshold of random 3-SAT is at least 3.52. Tech. Rep. RC22942, I.B.M., 2003.

[27] Kaporis, A. C., Kirousis, L. M., and Lalas, E. G. The probabilistic analysis of a greedy satisfiability algorithm. *Random Structures Algorithms 28*, 4 (2006), 444–480.

[28] Kim, J. H. Poisson cloning model for random graph. unpublished manuscript.

[29] Koutsoupias, E., and Papadimitriou, C. H. On the greedy algorithm for satisfiability. *Inform. Process. Lett. 43*, 1 (1992), 53–55.

[30] Krivelevich, M., and Vilenchik, D. Solving random satisfiable 3cnf formulas in expected polynomial time. In *SODA* (2006).

[31] Levin, L. A. Universal enumeration problems. *Problemy Peredači Informacii 9*, 3 (1973), 115–116.

[32] Mitzenmacher, M. Tight thresholds for the pure literal rule. Tech. Rep. 1997-011, Systems Research Center, 1997.

[33] Molloy, M. Cores in random hypergraphs and Boolean formulas. *Random Structures Algorithms 27*, 1 (2005), 124–135.

[34] Parkes, A. J. Scaling properties of pure random walk on random 3-SAT. In *CP '02: Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming* (London, UK, 2002), Springer-Verlag, pp. 708–713.

[35] Selman, B., Kautz, H. A., and Cohen, B. Local search strategies for satisfiability testing. In *Proceedings of the Second DIMACS Challange on Cliques, Coloring, and Satisfiability* (Providence RI, 1993), M. Trick and D. S. Johnson, Eds.

[36] Selman, B., Mitchell, D. G., and Levesque, H. J. Generating hard satisfiability problems. *Artificial Intelligence 81*, 1-2 (1996), 17–29. Frontiers in problem solving: phase transitions and complexity.