The Probabilistic Analysis of a Greedy Satisfiability Algorithm

Alexis C. Kaporis, Lefteris M. Kirousis, Efthimios G. Lalas
Department of Computer Engineering and Informatics University Campus, University of Patras, GR-265 04 Patras, Greece; e-mail: {kaporis,kirousis,lalas}@ceid.upatras.gr

Received 7 March 2003; accepted 24 November 2004; received in final form 22 July 2005
Published online 8 November 2005 in Wiley InterScience (www.interscience.wiley.com).
DOI 10.1002/rsa.20104

ABSTRACT: On input a random 3-CNF formula of clauses-to-variables ratio r_3 applies repeatedly the following simple heuristic: Set to TRUE a literal that appears in the maximum number of clauses, irrespective of their size and the number of occurrences of the negation of the literal (ties are broken randomly; 1-clauses when they appear get priority). We prove that for $r_3 < 3.42$ this heuristic succeeds with probability asymptotically bounded away from zero. Previously, heuristics of increasing sophistication were shown to succeed for $r_3 < 3.26$. We improve up to $r_3 < 3.52$ by further exploiting the degree of the negation of the evaluated to TRUE literal. © 2005 Wiley Periodicals, Inc. Random Struct. Alg., 28, 444–480, 2006

1. INTRODUCTION

Consider n Boolean variables $V = \{x_1, \ldots, x_n\}$ and the corresponding set of $2n$ literals $L = \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$. Each $x_i \in V$ may appear as positive literal x_i or as negative $\bar{x}_i$ in a k-clause, $k \geq 2$. A k-clause is a disjunction of k literals of distinct underlying variables. A random formula $\phi_{n,m}$ in k conjunctive normal form (k-CNF) is the conjunction of m

Contract grant sponsor: University of Patras Research Committee under Project *Carathéodory*.
Contract grant number: 2445.
Contract grant sponsor: *Research Academic Computer Technology Institute (RACTI)*.
Contract grant sponsor: European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II), and *PYTHAGORAS I*.
Contract grant sponsor: Future and Emerging Technologies programme of the EU.
Contract grant number: EU 001907, "*Dynamically Evolving, Large Scale Information Systems (DELIS)*".

clauses, each selected uniformly and independently among $2^k\binom{n}{k}$ possible clauses on n variables in V. The density r_k of a k-CNF formula $\phi_{n,m}$ is the clauses-to-variables ratio m/n. A k-CNF formula $\phi_{n,\lfloor r_k n\rfloor}$ is satisfiable if there exists an assignment of truth values to the variables such that $\phi_{n,\lfloor r_k n\rfloor}$ evaluates to 1. We say that for a given density r_k almost all formulas $\phi_{n,\lfloor r_k n\rfloor}$ are (un)-satisfiable iff the the ratio of (un)-satisfiable to all possible formulas approaches 1, as $n \to \infty$.

It is conjectured that for each $k \geq 2$ there exists a critical clauses-to-variables ratio r_k^* such that almost all k-CNF formulas $\phi_{n,\lfloor rn\rfloor}$ with ratio $(r > r_k^*)r < r_k^*$ are (un)-satisfiable, as $n \to \infty$. Friedgut [32] proved that for each $k \geq 2$ there exists a sequence of threshold values $r_k^*(n)$, depending on the number n of variables, such that for any $\epsilon > 0$ almost all k-CNF formulas $(\phi_{n,\lfloor(r_k^*(n)+\epsilon)n\rfloor})\phi_{n,\lfloor(r_k^*(n)-\epsilon)n\rfloor}$ are (un)-satisfiable, as $n \to \infty$. However, the convergence $\lim_{n\to\infty} r_k^*(n) = r_k^*$ for each $k \geq 3$ still remains open. Let

$$r_k^{*-} = \underline{\lim}_{n\to\infty} r_k^*(n) = \sup\{r_k : \Pr[\phi_{n,\lfloor r_k n\rfloor}\text{is satisfiable} \to 1]\}$$

and

$$r_k^{*+} = \overline{\lim}_{n\to\infty} r_k^*(n) = \inf\{r_k : \Pr[\phi_{n,\lfloor r_k n\rfloor}\text{is satisfiable} \to 0]\}.$$

Therefore, $r_k^{*-} \leq r_k^* \leq r_k^{*+}$, if r_k^* exists.

Franco and Paull pioneered the study of random k-CNF formulas and proved the general upper bound $r_k^{*+} < 2^k \ln 2$ in [31]. Then Chao and Franco established the general lower bound $\frac{1}{2}\left(\frac{k-1}{k-2}\right)^{k-2} 2^k/k < r_k^{*-}$ in [16]. These results suggested the simple law $r_k^{*-} = r_k^* = r_k^{*+} \sim 2^k \ln 2$. A series of experimental results comes up in favor of the threshold conjecture, see [20, 25, 58]. Monasson and Zecchina, using the non-rigorous replica method from statistical mechanics, predicted $r_k^* \sim 2^k \ln 2$ in [61].

Chvátal and Reed in [17] proved the simple law $\frac{1}{4}2^k/k < r_k^{*-}$ and further proved that $r_2^{*-} = r_2^* = r_2^{*+} = 1$; also see [12, 17, 37, 67, 68]. Frieze and Suen improved to $z_k 2^k/k < r_k^{*-}$, where $z_k = O(1)$ depending on k, as the best algorithmic lower bound for general k-CNF formulas in [33].

Wilson in [70] proved that for each $k \geq 2$ the characteristic width of the phase transition is at least $\Theta(n^{1/2})$, contradicting a number of empirical results in [35, 48, 49, 62, 63]. The width denotes the amount of extra clauses needed to be added in the random formula for the probability of satisfiability to drop from $1 - \epsilon$ to ϵ.

In a recent advance, Frieze and Wormald [34] proved that $2^k \ln 2 < r_k^{*-}$ as $k - \log_2 n \to \infty$, employing a second moment argument. Independently, Achlioptas and Moore [6] also applied the second moment method to prove that $\frac{2^k}{2} \ln 2 - z_k < r_k^{*-}$ for any *fixed* value of $k \geq 2$, where $z_k > 0$ is constant and depends on k. Recently, Achlioptas and Peres refined this method, proving $2^k(\ln 2 + o(1)) < r_k^{*-}$ in [8].

An important question concerns the complexity to compute a satisfying assignment, or on the contrary, to prove that none exists near the conjectured threshold value. To this end, Haken, Urquhardt, and Chvátal and Szemerédi in [18, 39, 66] were led to the conclusion that for k-CNF formulas of density $r_k > 2^k \ln 2$ any resolution proof of unsatisfiability contains at least $(1 + \epsilon)^n$ clauses. Monasson et al, using statistical mechanics, showed that the first-order phase transition correlates to the running time until a satisfying truth assignment is returned, by heuristics that are based on the Davis–Putnam simplification rule [62, 63]. Furthermore, Mézard and colleagues [55, 56] suggest a linear time algorithmic criterion that may improve the lower bound on r_k^{*-}. Achlioptas, Beame, and Molloy in [3] proved a $2^{\Omega(n)}$ lower bound for the running time for the DPLL (for Davis, Putnam, Logemann,

and Loveland) procedures GUC, UC, and ORDERED-DLL; see [15, 16] and [21, 22]. Informally, a DPLL procedure spits a formula into two sub-formulas by setting a variable to a fixed value and recursively invokes itself on each sub-formula.

For the particular case $k = 3$, upper bounds to r_3^{*+} have been proven using probabilistic counting arguments [26, 27, 42, 44, 47, 50, 53]; see the surveys [24, 51] about the techniques employed. Dubois, Boufkhad, and Mandler proved $r_3^{*+} < 4.506$ as the current best upper bound in [27]. As a corollary of [32], to prove that $c < r_3^{*-}$ it suffices to prove that a random 3-CNF formula of density c has a satisfying truth assignment with the probability of at least a positive constant. In this vein, Davis–Putnam algorithms of increasing sophistication were rigorously analyzed [1, 9, 14, 15, 17, 33]; see the surveys [2, 30] describing in detail various heuristics and techniques on their analysis. The best previous lower bound for the satisfiability threshold thus obtained is $3.26 < r_3^{*-}$ by Achlioptas and Sorkin in [9].

2. CONTRIBUTION

Almost all of the above algorithms (with the exception of the `Pure Literal` algorithm [14, 29, 59]) take into account *only* the clause size where the selected literal appears. Due to this limited information exploited on selecting the next variable, the simplified formula in each algorithmic step remains random conditional only on the current numbers of 3;2-clauses and variables. However, selecting the next variable only on the basis of the current numbers of 3;2-clauses led to algorithms of increasing sophistication that gave the lower bound $3.26 \leq r_3^{*-}$.

The first part of this paper concerns the analysis of a greedy Davis–Putnam algorithm that exploits *degree* information (number of literal occurrences) to select and set to TRUE a literal per free step (i.e., while there exist no 1-clauses); see Section 4.4. The algorithm is simple: in each round evaluate to TRUE a literal τ so as to satisfy the *maximum* number of clauses, irrespective of the occurrences of $\overline{\tau}$. It succeeds for densities $r_3 \leq 3.42$ establishing that $r_3^{*-} \geq 3.42$. Its simplicity, contrasted with the improvement over the previously obtained lower bounds, suggests the importance of analyzing heuristics that take into account degree information of the reduced formula. A preliminary version of this paper appeared in [45].

In the second part of this paper we exploit the number of occurrences of the negation $\overline{\tau}$ of the high degree literal τ selected per free step; see Section 5. Consider literals $\tau_1, \ldots, \tau_s \in L$, all of the highest degree in the current formula. Then it seems natural to give priority for satisfaction to literal $\tau_i, i \in \{1, \ldots, s\}$, whose negation $\overline{\tau}_i$ occurs in the fewest clauses. Intuitively, obtaining control on *complementary* literals $\tau, \overline{\tau} \in L$ we maximize the number of satisfied clauses and minimize the generation of new 1-clauses, increasing the probability of success of the algorithm. Our heuristic succeeds for densities $r_3 \leq 3.52$ establishing that $3.52 < r_3^{*-}$. However, the ordering of selection of such pairs of literals is not trivial. Assume that τ_1 with $\deg(\tau_1) = \Delta$ is the unique literal of currently maximum degree and $\deg(\overline{\tau}_1) = 3$. Should we better select literal τ_2 with $\deg(\tau_2) = \Delta - 1$ and $\deg(\overline{\tau}_2) = 2$? Here, each pair of complementary literals has *discrepancy* $\Delta - 3$. We provide a framework for analyzing algorithms that under an *arbitrary rule* Select & Set a pair of complementary literals per free step, irrespective of the clause sizes. By standard techniques, our algorithm can be easily modified to run in linear time. Thus, not only the satisfiability threshold, but also the threshold (experimental again) where the complexity of searching for satisfying truth assignments jumps from polynomial to exponential is at least 3.52. This should be contrasted with the value 3.9 for the complexity threshold given by theoretical (but not

mathematically rigorous) techniques of Statistical Physics [55, 56]. Hajiaghayi and Sorkin independently analyzed heuristics similar to those of our paper and claim to have obtained the same lower bound [38].

3. PLAN OF THE PAPER

The first part of this paper is Section 4. It concerns the probabilistic analysis of the algorithm `Greedy` described in sub-Section 4A. We define the notion of a *round* of algorithm's operation in sub-Section 4B. We also point out the reason we use the current number of rounds as the "time" parameter. Then we describe the model of generating a random formula obtained at the end of each round of the algorithm in sub-Section 4C. Connections of this model to existing ones are presented in sub-Section 4D. Furthermore, we initialize the degree sequence of the formula and we prove useful statistical properties per round in sub-Section 4E. A subcritical Galton Watson process of polylogarithmic total size establishes that the sequence of forced steps does not dominate any round in sub-Section 4F. We also prove a sufficient condition for positive probability of success for the algorithm. We compute the expected change of each of $(h + 3)$ parameters given which the reduced formula retains randomness in sub-Section 4G. Then we show that a theorem of Wormald applies in order to approximate within $o(1)$ and probability $1 - o(1)$ each of these $(h + 3)$ parameters in sub-Section 4H. We write down the system of $(h+3)$ differential equations, the solution of which approximates within $o(1)$ the dynamics of the algorithm per round in sub-Section 4I. We employ a theorem proved by Wormald [72], which helps us to approximate the dynamics of the algorithm with the solution of the system of differential equations, with high probability; see sub-Section 4H. We implement the d.e. until we reach a round where we can apply a theorem by Cooper Frieze and Sorkin [19] and safely terminate the algorithm; see sub-Section 4J. Finally, numerical computations and experiments are presented in sub-Section 4K.

The second part of this paper is devoted to the study of the dynamics of the algorithm `CL`; see Section 5. The randomness invariance of the reduced formula in each round of the algorithm is retained by keeping track of an appropriate degree sequence of $(h + 1)^2 + 3$ parameters; see the details in sub-Section 5B. The statistical properties of the reduced formula are presented in sub-Section 5C. We introduce the corresponding system of differential equations to keep track of the parameters given which the formula retains randomness in sub-Section 5D. Finally, we apply a criterion for the termination of the algorithm `CL` in sub-Section 5F.

4. NEGATION BLIND DEGREE SEQUENCE

A. Algorithm

The algorithm is applied to a random 3-CNF formula with n variables and density r_3. Let h be an a priori decided integer parameter, say $h = 10$. At a first phase, the algorithm arbitrarily selects and sets to TRUE literals of degree *at least h* (during *free* steps), until they are exhausted. At subsequent phases, it continues with literals of degree *exactly* $h - 1$ etc, in decreasing order of the degree. Unit clauses, whenever they appear, are given priority (*forced* step). In the numerical computations, we take $h = 10$ (a larger h gives a larger lower bound, but only with respect to its second decimal digit). The degree or number of occurrences of a literal τ in the formula is denoted $\deg(\tau)$ in the definition below.

Definition 4.1.

$$\mathcal{X}_j = \{\tau \in L \mid \deg(\tau) = j\},\ j = 0, \ldots, h-1, \text{ and } \mathcal{X}_h = \{\tau \in L \mid \deg(\tau) \geq h\}.$$

If $\tau \in \mathcal{X}_h$ then we call τ heavy; otherwise we call it light.

`Del&Shrink(`τ`)` is the Davis–Putnam simplification rule: delete all clauses in the current formula that contain literal τ, and delete $\overline{\tau}$ from all clauses in which it appears.

```
Algorithm: Greedy
begin:
j ← h;
while unset literals exist do:
  while X_j ≠ ∅ do:
    Select τ ∈ X_j & Set τ = 1;
    Del&Shrink(τ);
    while 1-clauses exist do:
      Select τ in a 1-clause & Set τ = 1;
    Del&Shrink(τ);
    end do;
  end do;
  j ← j − 1;
end do;
if a 0-clause is generated then report failure;
else report success;
end;
```

In all `Select` commands above, the selection can be based on a deterministic but otherwise arbitrary rule, e.g., always select the object with the least index that satisfies the corresponding requirements.

We prove that for $h = 10$ algorithm `Greedy` on input a random 3-CNF formula of initial density $r_3 \leq 3.42$ computes a satisfying truth assignment with probability at least a positive constant. Therefore:

Theorem 4.2. *The lower bound on r_3^{*-} is at least 3.42.*

The crucial property, proved in [59], that the negation of a 0-degree literal is a random literal, motivated us to work with arbitrary j-degree literals. Hence, in each free step, we set to TRUE a j-degree literal, with j maximum, satisfying in this way the maximum number of clauses while the number of shrinked clauses has the same expectation as if we had to set to true a random literal.

We were motivated to give priority to large degrees from [2, 9], where the need to capitalize on variable-degree information was pointed out and from [5], where, in the context of the 3-coloring problem, the Brélaz heuristic [13] was analyzed. According to [5], vertices of maximum degree are given priority, but only in case they can be legally colored by 2 of 3 possible colors.

Also in [36] Johnson's heuristic [43] is evaluated experimentally. This heuristic selects at each free step both a literal τ and its negation $\overline{\tau}$ on the basis of their corresponding degrees among 3,2-clauses. Algorithm `Greedy` is a simplification of this heuristic, since in a free step it selects a literal τ of the biggest degree in the formula (irrespective of the clause sizes) while $\overline{\tau}$ is random.

Finally, we were motivated to put together all heavy literals from [14], ([64]), where pure literals (light vertices) are set to TRUE (deleted) in order to find a satisfying truth assignment of a random formula (the k core of a graph), respectively.

B. Rounds

The algorithm proceeds in *rounds*. A round consists of one *free step*, i.e., a step where a literal in $\mathcal{X}_j$ is set to TRUE ($j = h, \ldots, 0$), followed by a number of *forced steps*, i.e., steps where 1-clauses are satisfied (the steps of the inner loop in the pseudo-code above). Of course, each of these steps is followed by the call of a Del&Shrink procedure. At the end of the sequence of forced steps only 3,2-clauses exist, and we reach the same reduced formula irrespective of the ordering that the algorithm satisfies the 1-clauses.

As in [9], in the analysis of the evolution of the algorithm, we consider as discrete time the number of rounds rather than the number of individual steps, which, for distinction, are to be called *atomic steps*. To explain why this choice of time is made, take into account that as the solution to the differential equations will show (for $r_3 = 3.42$ and $h = 10$), the expected number of unit clauses generated at any atomic step is bounded below 1; see sub-Section 4F.1. But then, during the course of the algorithm the number of unit clauses is equal to 0 unboundedly many times. This happens at the end of each round; all rounds have $O(1)$ atomic steps, so there are $O(n)$ of them, assuming no contradiction appears; see sub-Section 4F. As a consequence, if time corresponds to atomic steps, the evolution of the number of unit clauses cannot be analyzed by the method of differential equations. This is so because to apply this method, the rate of change, from a current step to the next, of the parameter under examination should be given by a smooth function (Lipschitz continuous function, see [72]) of the current scaled value of the parameter. This is not possible for the number of unit clauses, as its rate of change when there is at least one unit clause is discontinuously different from its rate of change when there is none (in the former case we deterministically delete one unit clause). See, for more details, sub-Section 4H. The technique of rounds, i.e., the change of the time parameter to count the number of rounds, guarantees that Wormald's theorem is applicable for the study of the evolution of stochastic parameters by the use of differential equations.

C. Randomness Invariance of the Formula in Each Round

We show that at the end of each round of the algorithm, the reduced formula is uniformly at random distributed over the space of all formulas with a given degree sequence, as the one in Definition 4.1 and with given number $|C_i|$ of i-clauses, $i = 2, 3$.

Algorithm `Greedy` selects randomly a literal τ that has specified degree (or that appears in a 1-clause), during each free (or forced) step. It transforms the current formula ϕ into the reduced ϕ', by deleting all clauses where τ appears and deleting all occurrences of $\bar{\tau}$, schematically: $\phi' \leftarrow \text{Del\&Shrink}(\phi, \tau)$.

Consider *Procedures A* and *B* below (and *C* in Section 5.B), where *Procedure A* corresponds to an atomic step of algorithm *Greedy*. We will show that *B* preserves conditional randomness (as defined below). From this we will deduce that the same is true for *A*. We refer by *Model A* (resp., *Model B* or *C*) to the processes corresponding to *Procedure A* (or to *Procedure B* or *C*).

Procedure A.

1. Select a literal occurrence τ in a clause of length one, if any (forced atomic step),
2. or select a literal τ of specified degree (free atomic step),
3. $\phi' \leftarrow$ Del&Shrink(ϕ, τ).

In addition, consider the simpler *Model B*, under which atomic steps such as

Procedure B.

1. select a literal occurrence τ in a clause of length one, if any (forced atomic step),
2. or select a literal τ (free atomic step),
3. $\phi' \leftarrow$ Del&Shrink(ϕ, τ),

can be expressed. Notice that it differs from *Model A* in that it is not possible to select a literal of specified degree. Observe that *Model B* cannot express each free step of `Greedy`, while it expresses each forced one. However, studying its limitations and modifying it accordingly, we finally construct *Model A*, which is adequate to express any atomic step of `Greedy`.

Lemma 4.3. *At the end of each algorithmic operation according to* Model B*, the reduced formula remains random conditional on its current number $|C_i|$ of i-clauses and its current number $|L|$ of literals, $i = 2, 3$.*

Proof. A random formula conditional on the number $|C_i|$ of i-clauses and the number $|L|$ of literals is constructed by selecting uniformly at random i-clauses from the space of all possible clauses on these literals, $i = 2, 3$. Each of the i literal occurrences in an i-clause is selected uniformly at random over $|L|$ possible literals. This makes a total of $3|C_3| + 2|C_2|$ clause occurrences that their underlying literal is *unexposed* or *secret*. The fact that these literal occurrences are unexposed means that the corresponding clause places can be filled uniformly at random over the $|L|$ possible literals. We can interpret these $3|C_3| + 2|C_2|$ literal occurrences as *cards* facing down, or *registers* with unexposed content. Also, let $|L|$ be unexposed registers containing each of the literals available. Working analogously as in [46], we can view each i-clause as an i-tuple of unexposed *clause-registers*, each register containing a secret pointer to one of the $|L|$ *literal-registers* of the literals available. Similarly, each of the $|L|$ literals can be seen as an unexposed literal-register with secret pointers to all the i-tuples that contain registers pointing to this literal. Also, each literal-register points to the unexposed literal-register of the negation of its underlying literal.

All in all, the fact that the pointer of a clause-register is secret means that its content can be specified uniformly at random among the $|L|$ possible literals. (Note that all these $|L|$ literals need not appear in the formula.) In a symmetric fashion, the content of each of the secret pointers in a literal-register can be specified uniformly at random among the $3|C_3| + 2|C_2|$ possible literal occurrences. This is done in an analogous manner as the content of a card is revealed in the expository card game presented in [52].

Using *Model B*, we can Select & Set to TRUE a random literal (or a random literal occurrence in a clause of specified length), amounting to model's B first (or second) kind of permissible atomic steps. These are performed by the following operations:

1. Select uniformly at random a literal-register among the $|L|$ possible (or select a literal occurrence from a random i-tuple of clause-registers). Let τ its underlying literal.

2. Delete the content of the literal-register of τ; delete the content of all the i-tuples of clause-registers that contain a clause-register pointed by the literal-register of $\tau, i = 2, 3$; update the content of all the remaining registers. This amounts to deleting τ and deleting all clauses in which it appears. Also delete the content of the literal-register of $\bar{\tau}$; delete the content of all the clause-registers it points to; update all the remaining registers. This amounts to deleting $\bar{\tau}$ and deleting all occurrences of $\bar{\tau}$.

Observe that we cannot infer information about the current content of any register that remains undeleted and unexposed. We should stress here that we cannot infer information combining the knowledge exposed from the currently exposed registers and the ones that were exposed during previous algorithmic steps. Therefore, the reduced formula remains random conditional on the new numbers $|C_i'|$ and $|L'|$ of i-tuples of clause-registers and literal-registers $i = 2, 3$, respectively. ∎

Lemma 4.4. *At the end of each algorithmic operation according to* Model A*, the reduced formula remains random conditional on its current number $|C_i|$ of i-clauses, $i = 2, 3$, its number $|L|$ of literals, and the number of literals of degree $j = 0, \ldots, 3|C_3| + 2|C_2|$.*

Proof. *Model A* is now easily constructed by assuming that each literal-register described in *Model B* is adjacent to an *exposed* degree-register that contains an integer equal to the degree of its underlying literal. In this way, an algorithm may Select & Set to TRUE a random literal of specified degree, during each free step. Once more we cannot infer information about the content of unexposed registers, as soon as the update of all the registers that remain undeleted is completed. Therefore, the reduced formula remains random conditional on its current number of unexposed registers. ∎

Lemma 4.5. *At the end of each algorithmic step of* `Greedy`*, the reduced formula remains random conditional on its current number $|C_i|$ of i-clauses, $i = 2, 3$, its number $|L|$ of literals, and the number $|\mathcal{X}_j|, j = 0, \ldots, h$, of literals (see Definition 4.1), where h is a sufficiently high integer, say $h = 10$. More precisely, the formula is random given the vector*

$$\mathcal{S} = \langle \ell, c_3, c_2, x_0, \ldots, x_{h-1} \rangle, \tag{4.1}$$

where $\ell = |L|/n; c_i = |C_i|/n; x_j = |\mathcal{X}_j|/n, j = 0, \ldots, h-1$, and n is the number of variables of the initial random formula.

Proof. The result follows easily by modifying slightly *Model A* described in the proof of Lemma 4.4. In this case, each literal-register is adjacent to an exposed degree-register that either contains the integer $j = 0, \ldots, h-1$, which equals the exact degree of its underlying literal, or contains integer h if the corresponding degree is at least h. That is, we have no information about the exact degree of any literal with degree-register equal to h. During each algorithmic step, the deletion of some clauses may cause some literals of current degree $j \geq h$ to finally get degree $j < h$. Although the degree content of such high degree-registers is secret, to perform the corresponding updates we need to know their exact degree during the simplification step. However, as soon as all updates are completed, it is not possible to infer the content of any unexposed register from the combined knowledge of current and previous information about the registers. ∎

D. Connection to Other Models of Random Formulas

In the previous literature concerning algorithms for the k-SAT [1, 2, 9, 15, 16, 17, 33], excluding `Pure Literal` [14, 29, 59], the models studied for generating random formulas give only information about the total number of clauses and the set of variables that a random formula can be constructed from.

The model that seems to capture all the computationally interesting aspects of k-SAT is the following: Let $V = \{x_1, \ldots, x_n\}$ the set of variables and their literals $L = \{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$. A k-clause is a disjunction of k literals of distinct underlying variables. A random k-SAT formula $\phi_{n,m}$ is the conjunction of a random m-subset of distinct clauses, selected uniformly from the set of all $2^k\binom{n}{k}$ possible clauses.

According to this model, no repetition of clauses is allowed to appear in the random instance and no clause may contain multiple or complementary literals. However, to simplify the probabilistic analysis, many papers have adopted slight modifications of this model, which may allow repeated or complementary literals in a clause and repetitions of clauses.

A popular model [2, 7, 14, 27, 34, 59], not restricted to the study of algorithmic issues concerning k-SAT, is the following: We construct a random $\phi_{n,m}$ by selecting, for each of the km total clause positions in it, a literal in L uniformly at random with replacement. Observe that multiplicities of clauses and literals may occur. An interested reader may find in [7, 33] (Sections 4.1 and 8, respectively) explanatory details why multiplicities of clauses or literals are irrelevant.

We adopt this model to construct the initial random formula, as seen in the proof of Lemma 4.3. Then we modify it accordingly in Lemmata 4.4, 4.5, 5.3, and 5.4, in order to handle degree information per step. We use the *Principle of Deferred Decisions* [52] and give a simple proof of randomness of Lemma 4.3, working as in [46, 52]. An interested reader may find early applications of this method in the context of myopic algorithms for k-SAT in [2, 33] Sections 2.1 and 2, respectively.

Furthermore, a random formula as described in Lemmata 4.4, 4.5, 5.3, and 5.4, where we need to handle degree information per algorithmic step, can be constructed using the *Configuration Model*. For example, a random formula in view of Lemma 4.4 can be constructed as follows: create j copies of each literal of degree $j = 0, \ldots, 3|C_3| + 2|C_2|$. Fill each of the $3|C_3|+2|C_2|$ available clause positions of the formula by selecting a literal copy uniformly with no replacement. Multiplicities of literals in clauses and multiple clauses in the formula are insignificant; see also the discussion below.

The Configuration Model was introduced by Bender and Canfield in [10] and refined in [12, 73]. The problem of handling degree information of a random structure has attracted a lot of interest lately. Of particular interest is the issue of generating random *r-regular* graphs [71, 74]. In such a graph all n vertices have degree r and is constructed by creating r copies of each of the n vertices (or hanging semi-edges) and choosing a random matching on these semi-edges. As long as no side effects such as multiple edges or self-loops occur, the resulting graph is distributed uniformly at random. These side effects are of similar nature as repetitions of literals or clauses and are discussed in detail in the Introduction of [74]. Recently, the Configuration Model was used for analyzing an algorithm in the context of coloring a random graph [5].

We should stress here that Lemma 4.3, or even 4.4 and 4.5, might be possible to prove via counting arguments as in [64]. However, enumerating all formulas with an unbounded or even bounded *negation dependent degree sequence*, as Lemmata 5.3 and 5.4 require, would be quite complicated, we believe.

E. Statistics of the Literals

Algorithm `Greedy` is initialized with a formula having the degree sequence defined below:

Proposition 4.6. *A random 3-SAT formula of density c on ℓn literals has w.h.p. the typical single degree sequence:*

$$x_j = \ell e^{-\lambda}\lambda^j/j! + o(1), j = 0, \ldots, h-1, \text{ and } x_h = \ell - \sum_{j=0}^{h-1} x_j + o(1),$$

where $\lambda = 3c/\ell$ is the expected degree of a random literal.

Proof. Concerning 3-SAT random formulas, the basic idea for the proof of this proposition can be found in [14], Lemma 4.3. In particular, Theorem 4.2 establishes that the scaled number x_0 of 0-degree literals is sharply concentrated to its expected value. In our paper we simply generalize this argument, from 0-degree to arbitrary j-degree literals, $0 \le j < h$, where h is a given integer. Also, it is helpful to see [54] where the analogous case of the degree sequence of the vertices of a random graph is studied. Finally, in papers [27, 28], a similar argument was applied to prove concentration results for the corresponding double degree sequence of the complementary literals of a random formula and the vertices of a random graph; see Proposition 5.5 in subsection 5C.

We sketch here the basic lines of the proof. A random 3-SAT formula consisting of cn 3-clauses over ℓn literals can be constructed by a random *balls into bins* game. We represent each of the $3cn$ clause positions of the formula as a distinct *ball* and each of the ℓn literals as a distinct *bin*. Each ball (clause position) independently lands in a bin (literal). The degree d_i of an arbitrary literal corresponds to the load of the underlying bin, $i = 1, \ldots, \ell n$. It follows that the joint distribution of the d_i's is $Multinomial(3cn; \frac{1}{\ell n}, \ldots, \frac{1}{\ell n})$.

Unfortunately the random variables d_i's are not independent, since the knowledge that a particular $d_i = k$ (that is, the load of a specific bin is k) affects the load distribution of any other $d_j, j \neq i$ (since now there remain $3cn - k$ balls to be distributed to the other bins).

However, consider the *independent* $Poisson(\lambda)$ random variables d_i''s with mean $\lambda = 3c/\ell$ that equals the expected load of a random bin in the above process. Let the random variable $\sum_{i=1}^{\ell n} id_i' = M$. Then M is a $Poisson(3cn)$ random variable with the nice property of concentration of its probability mass to its expected value $3cn$, that is,

$$\Pr[M = 3cn] = \text{poly}(n)^{-1}.$$

Given that $M = 3cn$ (which corresponds to the total number of balls in a random formula), then the d_i''s are distributed as d_i's,

$$\begin{aligned}\Pr[d_1 = k_1, \ldots, d_{\ell n} = k_{\ell n}] &= \Pr[d_1' = k_1, \ldots, d_{\ell n}' = k_{\ell n} \mid M = 3cn] \\ &= \Pr[d_1' = k_1, \ldots, d_{\ell n}' = k_{\ell n}] \cdot \text{poly}(n)^{-1},\end{aligned}$$

by deconditioning on $M = 3cn$. Using the independence of d_i''s, the number $|\mathcal{X}_j|$ of bins (literals) with load j is a $Binomial(\ell n, Poisson(\lambda; j))$ random variable, where

$$Poisson(\lambda; j) = e^{-\lambda}(\lambda)^j/j!, \ \ j = 0, \ldots, 3cn,$$

which is the probability that a particular bin receives j balls. Applying a Binomial large-deviation inequality, we obtain that $|\mathcal{X}_j|$ deviates by a constant factor from its

expectation $E[|\mathcal{X}_j|]$ with exponentially small probability. Then we get that the *scaled*, i.e., divided by n, number of literals of degree j is with high probability equal to

$$x_j = \frac{E[|\mathcal{X}_j|]}{n} = \ell e^{-\lambda}(\lambda)^j/j!,\ \ j = 0, \ldots, h-1.$$

■

Heuristics that Select & Set a literal without exploiting degree information enjoy the property that the remaining unset literals obey the Poisson distribution and the reduced formula is random given $\mathcal{S} = \langle \ell, c_3, c_2 \rangle$. Any degree guided heuristic, for example `Pure Literal` [14, 29, 59], violates this nice randomness property. In this simple example, the formula is random given the vector $\mathcal{S}_0 = \langle \ell, c_3, x_0 \rangle$, i.e., we also need to keep track of the scaled number x_0 of pure (light) literals per step. The following theorem that describes the distribution of literals in $\mathcal{X}_1$ (the set of literals with degree ≥ 1), will be generalized in Theorem 4.8, part 1, which describes the distribution of literals in $\mathcal{X}_h$ (the set of literals with degree $\geq h$), where h is an appropriate constant, say 10.

Theorem 4.7. [Broder, Frieze, and Upfal [14], Mitzenmacher [59]] *Let $\mathcal{X}_1$ be the set of literals with degree $k \geq 1$ at the end of each step of the Pure Literal algorithm. Each literal $\tau \in \mathcal{X}_1$ follows a truncated at 0 Poisson probability distribution:*

$$P_1(\mu; k) = \Pr[\deg(\tau) = k \mid \tau \in \mathcal{X}_1] = \frac{\mu^k}{(e^\mu - 1)k!}, k \geq 1,$$

where μ is the solution of the equation,

$$\lambda_1 = \tfrac{\mu e^\mu}{e^\mu - 1}, \textit{and}$$

$$\lambda_1 = \tfrac{3c_3}{x_1} \textit{ is the average load of a heavy bin.}$$

Proof. The reduced formula at the end of each step of the algorithm `Pure Literal` can be generated uniformly at random by using the model described in Lemma 4.5 and setting $h = 1$. According to this model, each of the $3c_3n$ clause-registers points to one of the x_1n literal-registers uniformly at random, such that each literal-register is pointed by at least one clause-register. This is equivalent to throwing randomly $3c_3n$ distinct *balls* into x_1n distinct *bins* such that no bin remains empty. Then it is well known that the probability that a bin (literal) $\tau \in \mathcal{X}_1$ has load (degree) $j \geq 1$ is a *truncated* at 0 Poisson distribution: $\frac{\mu^j}{(e^\mu - 1)j!}$. Parameter μ is the solution of the equation $\lambda_1 = \frac{\mu e^\mu}{e^\mu - 1}$, where $\lambda_1 = 3c_3/x_1$ is the expected load of a heavy bin. ■

A crucial observation, see also [59], is that the expected load of a *random* bin (expected degree of a random literal) equals the current density of the formula $\lambda_1 \frac{x_1}{\ell} = \frac{3c_3}{\ell} = \rho_3$. An important aspect of the algorithm `Greedy` is that at any atomic step, the literal to be set to TRUE is selected on the basis of information about itself and irrespective of properties of its negation. To describe this situation, we say that literals are decoupled from their negation. As a consequence, the literal set to FALSE at any atomic step is always uniformly random over all literals (the restriction that it has to be different from the literal set to TRUE introduces an $o(1)$ discrepancy, which is neglected). It is because of this that we can work with a degree sequence based on literals and not, as is usually the case, with a 2-dimensional

degree sequence that at each (i,j) gives the number of variables that have i positive and j negative occurrences. From the fact that the literals that are set to FALSE are uniformly random literals, it immediately follows that the expected number of unit clauses generated at any atomic step is the expected number of occurrences in 2-clauses of a random literal. This number is trivially the current density ρ_2 of 2-clauses at that atomic step.

Theorem 4.8. *Any literal $\tau \in L$ and any literal occurrence b in a formula that is random given the vector $\mathcal{S}$ in (4.1) has the following properties:*

$$P_h(\mu;k) = \Pr[\deg(\tau) = k \mid \tau \in \mathcal{X}_h] = \frac{\mu^k}{\left(e^\mu - \sum_{s=0}^{h-1} \frac{\mu^s}{s!}\right)k!}, k \geq h,$$

where μ is the solution of the equation:

1. $$\lambda_h = \frac{\mu\left(e^\mu - \sum_{j=0}^{h-2} \frac{\mu^j}{j!}\right)}{e^\mu - \sum_{j=0}^{h-1} \frac{\mu^j}{j!}},$$ *and λ_h equals*

$$\lambda_h = \frac{3c_3 + 2c_2 - \sum_{j=0}^{h-1} jx_j}{\ell - \sum_{j=0}^{h-1} x_j},$$ *i.e., it is the average load of a heavy bin.*

2. $\Pr[\exists \tau : \deg(\tau) > \ln n \mid \tau \in \mathcal{X}_h] \leq e^{-(1-o(1))\ln n \ln\ln n}$.

3. $\Pr[\tau \in \mathcal{X}_i] = \frac{x_i}{\ell}, i \leq h.$

$\Pr[$*Literal occurrence $b \in \mathcal{X}_i$*$] = \frac{\zeta_i^h x_i}{p}, i \leq h,$

4. *we define:* $$\zeta_i^h = \begin{cases} i, & i < h, \\ \lambda_h, & i = h. \end{cases}$$

5. $m = \mathbf{E}[\deg(\bar{b}) \mid b$ *is a literal occurrence*$] = \frac{3}{2}\rho_3 + \rho_2.$

$\varepsilon_1 = \mathbf{E}[\deg(b)$ *in 2, 3-clauses*$\mid b$ *appears in a 1-clause*$]$

6. $$= \frac{\sum_{s=0}^{h-1} s^2 x_s}{p} + \frac{x_h\left(\mu^2 e^\mu + \mu e^\mu - \sum_{s=0}^{h-1} \frac{s^2\mu^s}{s!}\right)}{p\left(e^\mu - \sum_{j=0}^{h-1} \frac{\mu^j}{j!}\right)} - 1$$

Proof. 1. The proof is generalization from $h = 1$ to an arbitrary integer h of the one given in Theorem 4.7. Now we have $p_h n = (3c_3 + 2c_2 - \sum_{j=0}^{h-1} jx_j)n$ distinct balls that are thrown uniformly at random into $x_h n = (\ell - \sum_{j=0}^{h-1} x_j)n$ distinct bins, in a way that all bins receive at least h balls. Then the probability mass of the number of literals of degree k, for any fixed integer $k \geq h$, follows a truncated at $(h-1)$ Poisson distribution. This means that for any $k \geq h$, the probability that a heavy literal has degree k is

$$\frac{\mu^k}{\left(e^\mu - \sum_{j=0}^{h-1} \frac{\mu^j}{j!}\right)k!},$$

where μ is the solution the equation

$$\lambda_h = \frac{\mu\left(e^\mu - \sum_{j=0}^{h-2} \frac{\mu^j}{j!}\right)}{e^\mu - \sum_{j=0}^{h-1} \frac{\mu^j}{j!}},$$

and $\lambda_h = p_h/x_h$ is the expected load of a heavy bin.

2. Inequality (4.2) of Theorem 4.2 in [14] applies verbatim for each heavy literal in $\mathcal{X}_h$; therefore,

$$\Pr[\exists \text{ Literal with degree} > \ln n] \leq e^{-(1-o(1))\ln n \ln\ln n},$$

i.e., we have sharp concentration to the expected load.

3. According to the model in Lemma (4.5), there are $x_i n$ literal-registers with underlying literal into the set $\mathcal{X}_i, i = 0, \ldots, h$. The desired probability follows by selecting uniformly at random one literal-register among the ℓn possible literal-registers.

4. Also, there are $pn = (3c_3 + 2c_2)n$ possible clause-registers, i.e., literal occurrences. In case $i < h$, among these clause-registers there are $ix_i n$ ones pointing to literal-registers with underlying literals in $\mathcal{X}_i$. Selecting uniformly at random a clause-register (literal occurrence) b we obtain

$$\Pr[b \in \mathcal{X}_i] = \frac{ix_i}{p}.$$

In case $i = h$, consider a heavy literal $\tau \in \mathcal{X}_h$. From part 1 above, we have that $\deg(\tau) = k \geq h$ with probability $P_h(\mu; k)$. Therefore, there are $x_h P_h(\mu; k)n$ literals in $\mathcal{X}_h$ each of degree $k \geq h$. Since each such literal is pointed by k clause-registers (literal occurrences), there are $\left(\sum_{k=h}^{\infty} kP_h(\mu; k)\right)x_h n = \lambda_h x_h n$ clause-registers, among pn possible, that point literal-registers with underlying literals in $\mathcal{X}_h$. Notice here that by the definition of the expectation it holds that $\sum_{k=h}^{\infty} kP_h(\mu; k) = \lambda_h$. Selecting uniformly at random a clause-register (literal occurrence) b we obtain

$$\Pr[b \in \mathcal{X}_h] = \frac{\lambda x_h}{p}.$$

5. Selecting at random a literal occurrence b amounts to selecting at random a clause-register. In turn, this points to the corresponding literal-register. This literal-register is adjacent to an *unexposed* literal-register with underlying literal $\bar{b}$. Since the register is unexposed, this means that $\bar{b}$ is selected uniformly at random among all literals. Then it has degree j with probability $\frac{x_j}{\ell}$, according to part 3 above, $j \leq h$. As in [59], we obtain

$$\mathbf{E}[\deg(\bar{b})|b \text{ is literal occurrence}] = m = \sum_{j=0}^{h-1} j\frac{x_j}{\ell} + \lambda_h \frac{x_h}{\ell} = \frac{3c_3 + 2c_2}{\ell} = \frac{3}{2}\rho_3 + \rho_2.$$

6. According to parts 1 and 4 above we have

$$\Pr[\deg(b) = s \mid b \in \text{1-clause}] = \begin{cases} \frac{sx_s}{p}, & s < h, \\ \frac{sx_h P_h(\mu;s)}{p}, & s \geq h. \end{cases}$$

Then the expectation equals

$$\begin{aligned}
\varepsilon_1 &= \sum_{s=1}^{h-1}(s-1)\frac{sx_s}{p} + \sum_{s\geq h}(s-1)\frac{sx_h P_h(\mu;s)}{p} \\
&= \sum_{s=1}^{h-1} s^2\frac{x_s}{p} + \frac{x_h}{p\left(e^{\mu} - \sum_{j=0}^{h-1}\frac{\mu^j}{j!}\right)}\sum_{s\geq h} s^2\frac{\mu^s}{s!} - 1 \\
&= \sum_{s=1}^{h-1} s^2\frac{x_s}{p} + \frac{x_h\left(\mu^2 e^{\mu} + \mu e^{\mu} - \sum_{s=0}^{h-1}\frac{s^2\mu^s}{s!}\right)}{p\left(e^{\mu} - \sum_{j=0}^{h-1}\frac{\mu^j}{j!}\right)} - 1.
\end{aligned}$$

■

F. Inside a Round

F.1. A Galton–Watson process. Assume, for the moment, that the density ρ_2 of 2-clauses remains constant during a round (we will elaborate on this point below). Then the generation of the 1-clauses during the forced steps of the round follows the pattern of a Galton–Watson branching process (see [23]). Such a process starts with a *pater familias* or root (or *alma mater*) and then at every step all individuals born at the previous step generate a number of offspring. The number of offspring in a Galton–Watson tree may follow an arbitrary fixed distribution whose mean is known as the Malthus parameter μ. If the Malthus parameter is <1, then, irrespective of other characteristics of the distribution of the offspring, the population certainly becomes extinct, eventually. Formally this means that with probability 1, the size of the Galton–Watson tree, i.e., the total number of individuals of all generations, is finite. Such processes are called subcritical. For a subcritical process, the expected size of the tree is equal to $1 + \mu + \mu^2 + \cdots = 1/(1 - \mu)$.

In our case, each atomic step amounts to selecting a literal-register (or clause-register) containing literal τ. According to Lemma 4.5, each such literal or clause-register is adjacent to an unexposed literal-register, which points to $\bar{\tau}$. Since this literal-register is unexposed, $\bar{\tau}$ corresponds to a random literal. This means that at each of these steps we choose a random literal. Therefore, the expected number of new 1-clauses during the jth *atomic* step may fluctuate since it equals the *current* density $\rho_2(j)$ of 2-clauses. That is, if the total number $|T|$ of 1-clauses generated in this round is large, this may affect the expected number of new 1-clauses, mainly due to repetitions of some already selected literals into the same clause.

Consider the G–W tree with starting node a random literal τ (bin). Its number of offsprings is Poisson-like distributed with mean ρ_2. Each child node corresponds to a random literal τ' whose number of offsprings has the same distribution. For each node-literal of the tree of the above G–W process, mark with symbol $\times(\rightarrow)$ its neighboring occurrences among 2-clauses (3-clauses). Perform no deletion or shrinking of clauses of the current formula. That is, symbol $\times(\rightarrow)$ denotes that the marked clause position corresponds to 1-clauses (2-clauses). If the literal of a new node has occurrence on a previously marked as $\rightarrow$ position then mark its neighboring position with $\times$. It is possible to mark multiply some clause positions or for some clauses to receive more than one mark. However, if we condition on $|T|$, such bad events occur with probability $O(|T|^2/n)$, where $|T|$ is the total number of nodes of the G–W tree. Taking the expectation (removing the condition on $|T|$), the expected length of the round is $\frac{1}{1-\rho_2} + \mathbf{E}[|T|^2]/n = \frac{1}{1-\rho_2} + o(1)$, as follows by Proposition 4.9.

Proposition 4.9. *Consider the G–W precess that corresponds to the generation of 1-clauses. If the first moment* $\mathbf{E}[\xi]$ *of the number of 1-clauses offsprings equals* $\rho_2 < 1$ *and the second moment* $\mathbf{E}[\xi^2]$ *is* $O(1)$ *then the first and second moment* $\mathbf{E}[|T|], \mathbf{E}[|T|^2]$, *respectively, of the size* $|T|$ *of the 1-clauses G–W tree, are both* $O(1)$.

Proof. Let $f(s)$ be the generating function of the number of offspring ξ of the above fictitious G–W process representing the evolution of 1-clauses. Also let $y(s)$ be the generating function of the size (total number of offspring) $|T|$ of the 1-clauses of this process. In Theorem 5.8 in [23] it is proved that $y(s) = sf(y(s))$. From this, by differentiation we can easily show that the second moment $\mathbf{E}[|T|^2] = O(1)$ in terms of the first $\mathbf{E}[\xi] = \rho_2 < 1$ and the second moment $\mathbf{E}[\xi^2]$ of the number of offspring. ∎

Having in mind the above fictitious process, observe that on each forced step during $t+1$ round the conditional on $\mathcal{S}(t)$ and $|T|$ expected change of each parameter $Y_j, j = 1, \ldots, h$ in vector (4.1) is given by

$$\mathbf{E}[Y_j(\text{per forced step}) \mid \mathcal{S}(t), |T|] = f_j(t/n, Y_1(t)/n, \ldots, Y_h(t)/n) + \frac{|T|}{n},$$

where each f_j is a Lipschitz continuous functions. Then during all $|T|$ forced steps during $t+1$ round we get

$$\mathbf{E}[Y_j(t+1) - Y_j(t) \mid \mathcal{S}(t), |T|] = f_j(t/n, Y_1(t)/n, \ldots, Y_h(t)/n)|T| + \frac{|T|^2}{n},$$

and averaging over $|T|$ we get

$$\mathbf{E}[Y_j(t+1) - Y_j(t) \mid \mathcal{S}(t)] = f_j(t/n, Y_1(t)/n, \ldots, Y_h(t)/n)\frac{\rho_2}{1-\rho_2} + o(1),$$

since $\mathbf{E}[|T|^2]/n = o(1)$, by Proposition 4.9.

F.2. Positive probability of success. At a fixed round again, conditional on the size $|T|$ of the Galton–Watson tree, it is easy to see that the probability that both a literal and its negation appear in the unit clauses is $O(|T|^2/n)$ (the decoupling of a literal from its negation is needed here). Therefore, we immediately conclude that the unconditional probability of contradiction during the round is $O(\mathbf{E}[|T|^2]/n)$ (when the tree is subcritical). Therefore, for all rounds, the probability that no contradiction occurs is $(1 - O(\mathbf{E}[|T|^2]/n))^n = e^{-O(\mathbf{E}[|T|^2])} = \Omega(1) > 0$, according to Proposition 4.9. Therefore, the probability of success of the algorithm, as long as the generation of unit clauses is subcritical, is bounded below by a positive constant.

Improper events are for example, multiple occurrences of a literal in the same clause or the simultaneous occurrence of pairs of literals $l, \bar{l}$ in the same clause. Given $|T|$ the probability for an improper event to occur is $O(|T|^2/n)$; see sub-Section 4F.1. Then averaging over all possible $|T|$'s during a round, we get that the probability of at least one improper event is $O(\mathbf{E}[|T|^2]/n) = o(1)$. Therefore, improper events introduce vanishing terms in each differential equation described in sub-Section 4I. In this way, we can safely discard such events, as $n \to \infty$.

G. Expected Changes per Round

Let $t \in [0, 1)$ denote the scaled number of rounds performed by the algorithm. We partition "time" interval $[0, 1)$ into subintervals $[0, T_h] \cup (T_h, T_{h-1}] \cup (T_{h-1}, T_{h-2}] \cup \ldots \cup (T_2, T_1]$, each sub-interval corresponding to a j-phase of the algorithm, $j = h, \ldots, 1$. Initially the algorithm is in the h-phase while the current scaled number of rounds $t \in [0, T_h]$. During this phase the algorithm Selects&Sets to TRUE literals from the set $\mathcal{X}_h$. Let $T_h \in [0, 1)$ be the scaled number of rounds such that $x_h(T_h) = 0.000005$. Here T_h is the time instance that the scale number x_h of literals with degree $\geq h$ has become insignificant. In the sequel, the algorithm enters $(h-1)$-phase and the current scaled number of rounds is $t \in (T_h, T_{h-1}]$. In this phase, it Selects&Sets to TRUE literals from the set $\mathcal{X}_{h-1}$ until it reaches a round T_{h-1} such that $x_{h-1}(T_{h-1}) = 0.000005$. Similarly, it enters $(h-2)$-phase and so on.

Lemma 4.10. *Suppose that during round $t \in [0,1)$ the algorithm* `Greedy` *has entered the jth phase, $j = h, \ldots, 1$. Then the expected change of each parameter conditional on the current vector,*

$$\mathcal{S} = \langle \ell, c_3, c_2, x_0, \ldots, x_{h-1} \rangle,$$

of the $(h+3)$ scaled parameters such that $\rho_2 < 1$, are within $o(1)$ equal to

$$(a) \quad \mathbf{E}[\Delta[|L|]|\ \mathcal{S}] = -2 - 2\frac{\rho_2}{1-\rho_2},$$

$$(b) \quad \mathbf{E}[\Delta[|C_3|]|\ \mathcal{S}] = -3\left(\frac{\varepsilon^j c_3}{p} + \frac{\rho_3}{2}\right) - 3\left(\frac{\varepsilon_1 c_3}{p} + \frac{\rho_3}{2}\right)\frac{\rho_2}{1-\rho_2},$$

$$(c) \quad \mathbf{E}[\Delta[|C_2|]|\ \mathcal{S}] = \frac{3\rho_3}{2} - \frac{2\varepsilon^j c_2}{p} - \rho_2 + \left(\frac{3\rho_3}{2} - \frac{2\varepsilon_1 c_2}{p} - \rho_2\right)\frac{\rho_2}{1-\rho_2},$$

$$(d) \quad \mathbf{E}[\Delta[|X_s|]|\ \mathcal{S}] = (6c_3 + 2c_2)\tfrac{(s+1)x_{s+1} - sx_s}{p^2}\varepsilon^j - \tfrac{x_s}{\ell} - \delta_{s,j} + \left((6c_3 + 2c_2)\tfrac{(s+1)x_{s+1} - sx_s}{p^2}\varepsilon_1 - \tfrac{x_s}{\ell} - \tfrac{sx_s}{p}\right)\tfrac{\rho_2}{1-\rho_2},$$

for $s = 0, \ldots, h-2$,

$$(e) \quad \mathbf{E}[\Delta[|X_{h-1}|]|\ \mathcal{S}] = (6c_3 + 2c_2)\tfrac{hy_h - (h-1)x_{h-1}}{p^2}\varepsilon^j - \tfrac{x_{h-1}}{\ell} - \delta_{h-1,j} + \left((6c_3 + 2c_2)\tfrac{hy_h - (h-1)x_{h-1}}{p^2}\varepsilon_1 - \tfrac{p + \ell(h-1)}{\ell p}x_{h-1}\right)\tfrac{\rho_2}{1-\rho_2},$$

where

$$\delta_{s,j} = \begin{cases} 1 & \text{if } j = s, \\ 0 & \text{otherwise,} \end{cases} \quad s = 0, \ldots, h-1,$$

$$\varepsilon^j = \begin{cases} \lambda_h & \text{if } j = h, \\ j & \text{if } j < h, \end{cases}$$

$$y_h = \frac{x_h \mu^h}{\left(e^\mu - \sum_{s=0}^{h-1} \frac{\mu^s}{s!}\right) h!}.$$

(a) In paragraph 4F.1 we prove that the expected number of forced steps per round is $\frac{\rho_2}{1-\rho_2}$. Therefore, $(1 + \frac{\rho_2}{1-\rho_2})$ steps are expected per round, where in each 2 literals are set.

(b) and (c) During each free (forced) step, a literal(clause)-register is selected that is adjacent to an unexposed literal-register that contains the negation of this selected literal. Since this literal-register is unexposed, its negation is a random literal. Therefore, it is expected in $\frac{k}{\ell}c_k = \frac{k}{2}\rho_k$ k-clauses, $k = 3, 2$; see also Lemma 4.8, part 4. In a free (forced) step of the round, the evaluated to TRUE literal τ has expected degree $\varepsilon^j(\varepsilon_1)$; see Lemma 4.8, part 6. Then, in the free step, each occurrence of τ (a ball) is expected in $\varepsilon^j \frac{kc_k}{p}$ k-clauses, while in each forced step it is expected in $\varepsilon_1 \frac{kc_k}{p}$ k-clauses, $k = 3, 2$, since the total of balls is pn and $kc_k n$ of them belong in k-clauses. Expected changes (b) and (c) are obtained by the expected change of the free step plus the expected change of a single forced step multiplied by the expected number of forced steps $\frac{\rho_2}{1-\rho_2}$.

(d) As above, in the free step, the evaluated to TRUE literal is expected to occur in $\varepsilon^j \frac{kc_k}{p}$ k-clauses deleting $\varepsilon^j \frac{kc_k}{p}(k-1)$ neighboring literal occurrences. In each forced step,

the evaluated to TRUE literal is expected in $\varepsilon_1 \frac{kc_k}{p}$ k-clauses deleting $\varepsilon_1 \frac{kc_k}{p}(k-1)$ neighboring occurrences, $k = 3, 2$. Now, each of these occurrences has degree s with probability $\frac{sx_s}{p}$ introducing a flow-out from the set $\mathcal{X}_s$ and has degree $s+1$ with probability $\frac{(s+1)x_{s+1}}{p}$ introducing a flow-in to $\mathcal{X}_s, s = 0, \ldots, h-2$. This gives the expected change due to the deletion of the neighboring occurrences in the satisfied and deleted clauses that the evaluated to TRUE literal appears per step. It remains to compute the expected change in $\mathcal{X}_s$ due to the deletion of the evaluated to TRUE literal and its negation per step. In each free/forced step the negation of the selected literal is a random literal and is removed from $\mathcal{X}_s$ with probability $\frac{x_s}{\ell}$. In each forced step the 1-clause literal is a literal occurrence and is removed from $\mathcal{X}_s$ with probability $\frac{sx_s}{p}$. Finally, in the free step we deterministically remove the selected literal from $\mathcal{X}_s$ iff $s = j$. This is why we introduce the indicator variable $\delta_{s,j}$.

(*e*) Here the expected changes per step go verbatim as in (*d*). A subtle difference is that x_h denotes the scaled number of literals of degree $\geq h$. However, to compute the expected flow of literals into set $\mathcal{X}_{h-1}$ we need the scaled expected number of literals of degree exactly h. This number equals $y_h = x_h \Pr_h(\mu; h)$; see Theorem 4.8, part 1.

H. Wormald's Theorem

As we already pointed out, our analysis is based on the method of differential equations. For an exposition of how the relevant Wormald's theorem is applied to the satisfiability problem, see [2]. Roughly, the situation is as follows: suppose that $Y_j, j = 1, \ldots, a$ are stochastic parameters related to a formula, e.g., the number of clauses with a specified size or the number of literals with a specified degree. In our case, the Y_j's are the $h+3$ parameters in $\mathcal{S}$. We want to estimate the evolution of the parameters Y_j during the course of a Davis–Putnam algorithm. The formula initially is a 3-CNF formula with n variables and is uniformly random conditional on given initial values $Y_j(0), j = 1, \ldots, a$. These initial values in our case are constant multiples of n (in general, they may be random numbers). As the formula is random with respect to the names (labels) of the literals, we assume that the Davis–Putnam algorithm selects at any atomic step the first literal (in some arbitrary ordering of the labels) that is subject to the restrictions of the algorithm. In other words, the algorithm is assumed to be deterministic and the sample space is determined by the initial formula only.

Suppose that the expected change of each $Y_j, j = 1, \ldots, a$, from time step t to $t+1$, conditional on the values $Y_j(t), j = 1, \ldots, a$ of the parameters at t, for all possible values of the random parameters $Y_j(t)$, is given by

$$\mathbf{E}[Y_j(t+1) - Y_j(t) \mid Y_1(t), \ldots, Y_a(t)] = f_j(t/n, Y_1(t)/n, \ldots, Y_a(t)/n) + o(1),$$

and each $f_j : \Re^{a+1} \to \Re$ is a Lipschitz continuous function, according to conditions (ii) and (iii) of Theorem 2 in [72]. Suppose also that the probability that $|Y_j(t+1) - Y_j(t)| > n^{1/5}$ is at most $o(n^{-3})$, i.e., the change of each parameter is concentrated to its expected change per step. Then the solution of the system of differential equations,

$$dy_j(x)/dx = f_j(x, y_1(x), \ldots, y_a(x)), j = 1, \ldots, a,$$

with the initial point $y_j(0) = Y_j(0)/n$, satisfies for all t with probability $1 - o(1)$ as n tends to infinity:

$$y_j(t/n) = (1/n)Y_j(t) + o(1), j = 1, \ldots, a.$$

In applications, an open, connected, and bounded domain D that contains the initial point $(Y_1(0)/n, \ldots, Y_a(0)/n)$ and a time interval $[0, t_f)$ is considered, and it is assumed that the hypotheses of the theorem hold up to the last time instant $T < t_f$ such that for all $t \in [0, T]$, $(Y_1(t)/n, \ldots, Y_a(t)/n) \in D$ (T is a random variable). Then the conclusion of the theorem holds, for large enough n, up to any $t < t_f$ such that for all $x \in [0, t/n]$, $(y_1(x), \ldots, y_a(x)) \in D$. In this context, it is sufficient that the Lipschitz continuity of the f_j's holds over $[0, t_f/n) \times D$. The above is only a rough outline of Wormald's theorem, not in its full generality, but restricted to the purposes of our particular problem.

In our case, we can see that Wormald's conditions hold for each expected change (a)–(e) described in Lemma 4.10, as we demonstrate below.

- Each atomic step in a round is equivalent to the deletion of the content of balls (literal occurrences) of a pair of bins (literals). Part 2 of Theorem 4.8 establishes the Poisson-like tail bounds for the probability of the load of any bin exceeding $\ln n$. Then during the fictitious G–W process, each unscaled parameter in vector (4.1) is concentrated to its conditional expected change described in Eqs. (a)–(e) of Lemma 4.10, as required from condition (i′) of Theorem 2 in [72].
- Observe that Eqs. (a)–(e) of Lemma 4.10 give the expected change of the corresponding $h + 3$ unscaled parameters in vector (4.1) within an $o(1)$ error, as required from condition (ii) of Theorem 2 in [72]. This is due to the fact that any unscaled parameter in vector (4.1) may change by at most $O(\ln^2 n)$ during an arbitrary round, with high probability. This may introduce at most $o(1)$ fluctuation per round from the corresponding expected change described in Eqs. (a)–(e) of Lemma 4.10.
- Finally, according to condition (iii) of Theorem 2 in [72] the right-hand side of each differential equation (a)–(e) in Lemma 4.10 is Lipschitz continuous. Recall that each parameter in vector (4.1) is strictly positive. Therefore, each fractional term appearing in the free and forced part of each equation (a)–(e) is bounded since it has denominator > 0; see Remark 4.11. Furthermore, during round t we condition upon a subcritical degree sequence $\mathcal{S}$ such that $\rho_2 < 1$ and the term $\frac{\rho_2}{1-\rho_2}$ is bounded too. We conclude that for each equation (a)–(e) there exists an absolute Lipschitz constant $L_j(\rho_2)$, while ρ_2 remains <1.

  Notice that a given subcritical degree sequence $\mathcal{S}(t)$ may yield a supercritical $\mathcal{S}(t+1)$ one with $\rho_2 \geq 1$. Clearly, we cannot apply Wormald's theorem on $\mathcal{S}(t+1)$ to compute $\mathcal{S}(t+2)$. However, the theorem remains true if we restrict its application to a domain $D = \{\langle \mathcal{S}(t), t\rangle |$ such that $\rho_2(t) < 1\} \subseteq \Re^{(h+3)+1}$ consisting only of these $[(h+3)+1]$-dimensional vectors that have the property $\rho_2 < 1$.

I. Differential Equations

Wormald's Theorem is described in sub-Section 4H. Here we apply this theorem to map the expected changes of Lemma 4.10 into a system of differential equations. As a consequence, we obtain Lemma 4.12, which approximates within $o(1)$ and with probability $1 - o(1)$ the values of each scaled parameter of vector $\mathcal{S}$.

Remark 4.11. *Each differential equation* (a)–(e) *consists of two parts. The first part is the expected change of the parameter under consideration of the single free step. The second part is the expected change in a single forced step multiplied by the factor* $\frac{\rho_2}{1-\rho_2}$, *which is the expected number of forced steps.*

Lemma 4.12. *Suppose that the algorithm* `Greedy` *has entered the jth phase, $j = h, \ldots, 1$. Then, in each round the $h+3$ parameters in the vector*

$$\mathcal{S} = \langle \ell, c_3, c_2, x_0, \ldots, x_{h-1} \rangle,$$

such that $\rho_2 < 1$, are approximated within $o(1)$ and with probability $1-o(1)$ by the solution of the following system of differential equations:

$$(a) \quad \frac{d\ell}{dt} = -2 - 2\frac{\rho_2}{1-\rho_2},$$

$$(b) \quad \frac{dc_3}{dt} = -3\left(\frac{\varepsilon^j c_3}{p} + \frac{\rho_3}{2}\right) - 3\left(\frac{\varepsilon_1 c_3}{p} + \frac{\rho_3}{2}\right)\frac{\rho_2}{1-\rho_2},$$

$$(c) \quad \frac{dc_2}{dt} = \frac{3\rho_3}{2} - \frac{2\varepsilon^j c_2}{p} - \rho_2 + \left(\frac{3\rho_3}{2} - \frac{2\varepsilon_1 c_2}{p} - \rho_2\right)\frac{\rho_2}{1-\rho_2},$$

$$(d) \quad \frac{dx_s}{dt} = (6c_3 + 2c_2)\tfrac{(s+1)x_{s+1} - sx_s}{p^2}\varepsilon^j - \tfrac{x_s}{\ell} - \delta_{s,j} + \left((6c_3 + 2c_2)\tfrac{(s+1)x_{s+1} - sx_s}{p^2}\varepsilon_1 - \tfrac{x_s}{\ell} - \tfrac{sx_s}{p}\right)\tfrac{\rho_2}{1-\rho_2},$$

for $s = 0, \ldots, h-2$,

$$(e) \quad \frac{dx_{h-1}}{dt} = (6c_3 + 2c_2)\tfrac{hy_h - (h-1)x_{h-1}}{p^2}\varepsilon^j - \tfrac{x_{h-1}}{\ell} - \delta_{h-1,j} + \left((6c_3 + 2c_2)\tfrac{(hy_h - (h-1)x_{h-1}}{p^2}\varepsilon_1 - \tfrac{x_{h-1}}{\ell} - \tfrac{(h-1)x_{h-1}}{p}\right)\tfrac{\rho_2}{1-\rho_2},$$

where

$$\delta_{s,j} = \begin{cases} 1 & \text{if } j = s, \\ 0 & \text{otherwise,} \end{cases} \qquad s = 0, \ldots, h-1,$$

$$\varepsilon^j = \begin{cases} \lambda_h & \text{if } j = h, \\ j & \text{if } j < h, \end{cases}$$

$$y_h = \frac{x_h \mu^h}{\left(e^\mu - \sum_{s=0}^{h-1} \frac{\mu^s}{s!}\right) h!}.$$

Initial conditions:

$$\ell = 2, c_3 = c, c_2 = 0, x_s = \frac{2e^{-3c/2}(3c/2)^s}{s!}, \text{ for } s = 0, \ldots, h-1.$$

Proof. The fact that conditions (i′)–(iii) of Theorem 2 in [72] hold for the expected changes described in Lemma 4.10 is proved in sub-Section 4H. ∎

J. Implementation and Termination of the Algorithm

Recall the definition of the jth *phase* of the algorithm, $j = h, \ldots, 0$ that is given in the beginning of sub-Section 4I. A j-phase corresponds to the middle loop with the constraint $|\mathcal{X}_j| > 0.00005n$ of the pseudo-code of the algorithm. Each j-phase has length of $\Omega(n)$ rounds and in each round exactly one literal is selected from $\mathcal{X}_j$. The j-phase ends as soon

as $|\mathcal{X}_j| = x_j n = 0.00005n = \Omega(n)$, i.e., when the scaled number of literals with degree j becomes insignificant. In this way, each transition from a j-phase to a $(j-1)$-phase, $j = h, \ldots, 1$, satisfies condition (iii) of Theorem 1 in [72]. As soon as the j-phase ends, the leftover quantity of j-degree literals is insignificant. Furthermore, it introduces an expected change to the system of d.e. that always diminishes as the process evolutes.

Observe that the system of d.e.'s is a non-stiff one. Each parameter in vector (4.1) is a smooth function of time t. Matlab [57] employing a second-order Runge–Kutta method can solve this system with arbitrary precision.

On input a random 3-CNF formula of initial density $c = 3.42$, the Malthus parameter ρ_2 remained <1 during all rounds of `Greedy`. We simulated the system of d.e.'s until we reached a round t^* such that we could apply Lemma 4.14 and safely terminate the algorithm. That is, the reduced random formula at the end of round t is almost surely satisfiable and it is easy to compute a satisfying truth assignment. This lemma is a consequence of Theorem 1 in [19], which we describe in detail below (Theorem 4.13).

Consider the set of 2-CNF formulas,

$$\Omega_d = \{\phi : \text{degree}(x_i) = d_i \wedge \text{degree}(\bar{x}_i) = \bar{d}_i, i = 1, \ldots, n\},$$

such that the degree of each literal $x_i, \bar{x}_i$ corresponds to a *fixed* degree sequence $d = d_1, \bar{d}_1, \ldots, d_n, \bar{d}_n$ with $d_i + \bar{d}_i \geq 1$ for each i. Let Δ_d the maximum degree with respect to d and

$$D_1 = \sum_{i=1}^{n} (d_i + \bar{d}_i) = 2M \geq n, D_2 = \sum_{i=1}^{n} d_i \bar{d}_i,$$

where M is the number of the 2-clauses in ϕ. A degree sequence d is Δ-*proper* if (i) $\Delta_d \leq \Delta$ and (ii) $\Delta_1 = 2M$.

Theorem 4.13. **[Cooper, Frieze, and Sorkin [17]]** *Let $0 < \varepsilon < 1$ be constant and $n \to \infty$. Let d be any* Δ-proper *literal-degree sequence over n variables, with $\Delta = n^{1/11}$, and let ϕ be a uniform random simple formula with degree sequence d.*

- *If $2D_2 < (1-\varepsilon)D_1$ then $P[\phi$ is satisfiable$] \to 1$.*
- *If $2D_2 > (1+\varepsilon)D_1$ then $P[\phi$ is satisfiable$] \to 0$.*

Both limits are uniform in n (independent of d).

As a consequence we get the following lemma.

Lemma 4.14. *A random formula given the degree sequence $\mathcal{S}$ in (4.1) is almost surely satisfiable if there exists $\varepsilon > 0$ such that $\rho_2 + \rho_3 < 1 - \varepsilon$.*

Proof. Consider a random formula ϕ given the current degree sequence $\mathcal{S}$. From part 2 of Theorem 4.8 it holds w.h.p. that the maximum degree Δ_d of any literal in $\mathcal{S}$ is at most $\ln n < n^{\alpha}, \alpha < 1/11$. Since the current formula ϕ consists of 3;2-clause-registers, delete exactly one random clause-register (literal occurrence) from each 3-tuple of clause-registers. Such deletions are feasible, since the 3-tuples of clause-registers are exposed; see Lemma 4.5. This results in a formula ϕ' consisting of $(c_3 + c_2)n$ 2-tuples of clause-registers (2-clauses)

and, therefore, $D_1 = 2(c_3+c_2)n$. Clearly, almost sure satisfiability of ϕ' implies almost sure satisfiability of ϕ. However, ϕ' is random given a new degree sequence $\mathcal{S}'$. Furthermore, to compute D_2 we denote as $n_{\kappa\lambda}(t^*)$ the number of unset variables with κ positive occurrences and λ negative occurrences, while $n(t^*) = \ell(t^*)/2$ is the total number of currently unset variables. In this way,

$$D_2 = \sum_{i=1}^{n} d'_i \overline{d'}_i = \sum_{\kappa,\lambda} \kappa\lambda\, n_{\kappa\lambda}(t^*). \tag{4.2}$$

Also, an arbitrary variable x has κ positive occurrences and λ negative occurrences with probability

$$\begin{aligned} p_{\kappa\lambda}(t^*) &= \mathbf{Pr}[x \in \mathcal{X}_\kappa(t^*) \wedge \overline{x} \in \mathcal{X}_\lambda(t^*)] = \frac{x_\kappa(t^*)x_\lambda(t^*)}{\ell^2(t^*)} = \frac{n_{\kappa\lambda}(t^*)}{n(t^*)} \\ &\Leftrightarrow n_{\kappa\lambda}(t^*) = \frac{x_\kappa(t^*)x_\lambda(t^*)}{\ell^2(t^*)}\frac{\ell(t^*)}{2}. \end{aligned} \tag{4.3}$$

Here, by abuse of the *truncated* on h notation, $\mathcal{X}_\kappa(t^*)$ and $x_\kappa(t^*)$ denote the set and the number of literals with arbitrary degree κ at round t^*, respectively. The independence among complementary literals is crucial in establishing Eq. (4.3) above. From (4.3), Eq. (4.2) becomes

$$\begin{aligned} D_2 &= \sum_{\kappa,\lambda} \kappa\lambda \frac{x_\kappa(t^*)x_\lambda(t^*)}{\ell^2(t^*)}\frac{\ell(t^*)}{2} = \left(\frac{0x_0(t^*)}{\ell(t^*)} + \cdots + \frac{\kappa x_\kappa(t^*)}{\ell(t^*)} + \cdots\right)^2 \frac{\ell(t^*)}{2} \\ &= \left(\frac{2c_2(t^*) + 2c_3(t^*)}{\ell(t^*)}\right)^2 \frac{\ell(t^*)}{2}. \end{aligned}$$

That is, according to Theorem 1 in [19], the resulting formula at the end of round t^* is satisfiable with high probability if it holds

$$\begin{aligned} 2D_2 \le (1-\varepsilon)D_1 &\Leftrightarrow 2\left(\frac{2c_2(t^*) + 2c_3(t^*)}{\ell(t^*)}\right)^2 \frac{\ell(t^*)}{2} \le (1-\varepsilon)(2c_2(t^*) + 2c_3(t^*)) \\ &\Leftrightarrow r_2(t^*) + r_3(t^*) \le (1-\varepsilon), \varepsilon > 0. \end{aligned}$$

■

K. Numerical Results

The simulation of the algorithm was implemented on C. For the generation of random 3-CNF formulas, we made use of the code freely distributed at SAT–The Satisfiability Library [65]. Our implementation was influenced and makes use of the code for the implementation of GSAT, also available at the above site. The simulation was implemented for 5×10^5 variables.

The simulation results for the parameters in $\mathcal{S}$ are very close to the corresponding values obtained from the numerical solution of the differential equations as can be seen from Table 1 in the Appendix. In this table, each line initiated with "d.e." contains the vector solution of the system of differential equations while each following "sim." line contains the corresponding experimental values.

5. NEGATION DEPENDENT DEGREE SEQUENCE

The remainder of the paper is devoted to the analysis of the algorithm CL. Algorithm CL is a modification of the algorithm Greedy presented in sub-Section 4A. Recall that Greedy sets to TRUE a literal of maximum degree per free step, irrespective of its negation. Therefore, it obtains no control on the number of new 2;1-clauses generated per free step. This is a serious limitation since the probability of an 0-clause generation (contradiction) increases significantly as 2;1-clauses accumulate.

The main contribution of CL is that it sets to TRUE a literal on the basis of its degree and the degree of its negation per free step. Therefore, CL improves significantly over algorithm Greedy on handling both deleted and shrunk clauses per free step by setting TRUE a literal τ of high degree while $\bar{\tau}$ has low degree.

Notice that at the end of each round performed by Greedy, the simplified formula remained random given the current number $|\mathcal{X}_j|$ of literals with degree $j = 0, \ldots, h$ (see Lemma 4.5) and the number $|C_i|$ of i-clauses, $i = 2, 3$. Now, for the probabilistic analysis of CL, we additionally have to keep track of the current number $|\mathcal{X}_{i,j}|$ of literals with degree $i = 0, \ldots, h$ *whose negation* has degree $j = 0, \ldots, h$. More formally, we introduce the following *negation-dependent* degree sequence; see also the corresponding definition of the *negation-blind* degree sequence for Greedy in Definition 4.1 in sub-Section 4A.

Definition 5.1.

$$\mathcal{X}_{i,j} = \{\tau \in L \mid \deg(\tau) \succeq_h i \text{ and } \deg(\overline{\tau}) \succeq_h j\}, (i,j) \in A = \{0, \ldots, h\}^2,$$

where we define the relation $\succeq_h$ *as*

$$\deg(\tau) \succeq_h i \Leftrightarrow \begin{cases} \deg(\tau) = i, & i < h, \\ \deg(\tau) \geq h, & i = h. \end{cases}$$

If a literal $\tau \in L$ *belongs in* $\mathcal{X}_{h,i}, i = 0, \ldots, h$ *is called heavy; otherwise it is called light.*

Remark 5.2. *If* $\tau \in \mathcal{X}_{i,j}$ *then* $\overline{\tau} \in \mathcal{X}_{j,i}$ *and* $|\mathcal{X}_{i,j}| = |\mathcal{X}_{j,i}|$, $\forall (i,j) \in A$. *Also, if* $\tau \in \mathcal{X}_{i,i}$ *then* $\overline{\tau} \in \mathcal{X}_{i,i}, 0 \leq i \leq h$.

A. Algorithm

In this section we describe algorithm CL:

```
Algorithm: CL
begin:
  while unset literals exist do:
    (s,t)← Choose-Bucket;
        Select τ ∈ X_{s,t} & Set τ = 1;
        Del&Shrink(τ);
        while 1-clauses exist do:
          Select τ in a 1-clause & Set τ = 1;
          Del&Shrink(τ);
        end do;
  end do;
end;
```

In all `Select` commands above, the selection can be based on a deterministic but otherwise arbitrary rule, e.g., always select the object with the least index that satisfies the corresponding requirements.

Let $m_2(t)$ be the rate of generation of new 1-clauses during the round of forced steps that follow a free step t (t will denote both a step and the content of the step counter before this step is taken) performed by algorithm `CL`. This rate remains constant—a.a.s. and within $o(1)$—during the round. In other words, $m_2(t)$ is the expected flow of shrunk 2-clauses into 1-clauses during the round of forced steps.

We will see that $m_2(t)$ is the expected number of occurrences in 2-clauses of the negation of a random literal chosen among the literal-occurrences in 2-clauses, just before the step t is taken. This will become clear in Theorem 5.6, part 5, in sub-Section 5C. So $m_2(t)$ does not depend on which particular literal is chosen to be set true at free step t; it only depends on the distribution of literals in the clauses just before step t.

It is worth reminding the reader at this point that choosing randomly a literal is a different random process from choosing randomly a literal-occurence. If we think of the literal-occurrences as balls thrown into bins that correspond to literals, then choosing a random literal-occurrence corresponds to choosing a ball, while choosing a literal corresponds to choosing a bin.

Of course, once a literal-occurrence is randomly chosen, then we can consider the corresponding literal. So by the preceding paragraph, to compute $m_2(t)$, we choose a random literal-occurrence among those in 2-clauses, we then consider the corresponding literal; then we take its negation and finally count the expected number of occurrences in 2-clauses of the latter.

Let also $m_3(t)$ be the rate of generation of new 2-clauses during the round of forced steps that follow t. This rate remains constant—a.a.s. and within o(1)—during the round. In other words, $m_3(t)$ is the expected flow of shrunk 3-clauses into 2-clauses during the round of forced steps.

Similarly, we will show that $m_3(t)$ is the expected number of occurrences in 3-clauses of the negation of a random literal chosen among the literal-occurences in 2-clauses just before step t is taken. This will become clear in Theorem 5.6, part 5, of sub-Section 5C. Again, $m_3(t)$ does not depend on which particular literal is chosen to be set true at step t; it only depends on the distribution of literals into clauses just before step t.

Also if t is a free step, let t' be the step counter at the beginning of the next round of forced steps, i.e., just before the next free step is taken (in other words, $t' - t$ is the number of forced steps that follow t). Notice again that $m_2(t')$ and $m_3(t')$ do not depend on which literal is selected to be made true at the free step t', but certainly depend on which literal was selected to be made true at step t.

During a free step t, suppose that we set TRUE a literal $\tau \in \mathcal{X}_{i,j}$. Then we define the ratio

$$R(i,j) = \frac{m_2(t') - m_2(t)}{m_3(t') - m_3(t)}.$$

Notice that this ratio counts the marginal increase in the flow of 2-clauses into 1-clauses between two consecutive rounds of forced steps per unit of the marginal decrease in the flow of 3-clauses into 2-clauses between the same two consecutive rounds.

In the description of `Algorithm CL` below, at every free step the procedure `Choose-Bucket` selects the next literal to be set to TRUE so that this ratio is maximized. For more details about the implementation of this procedure see sub-Section 5F.

Notice that the dependence of $m_2(t')$ and $m_3(t')$ on the literal selected at t but not on the literal selected at t' renders the above criterion a well-defined one.

Because $m_3(t') - m_3(t)$ is negative, this criterion minimizes the increase between two consecutive rounds of forced steps of the flow of 2-clauses to 1-clauses per unit of decrease of the flow of 3-clauses to 2-clauses. Intuitively, it makes good sense to increase as little as possible the flow from 2-clauses to 1-clauses, while decreasing as much as possible the rate of flow from 3-clauses to 2-clauses.

It is worth mentioning here that in [9] it is proved that this criterion of selecting literals at the free steps of a DPLL heuristic is optimal among the ones that take into account only the number of 2-clauses and 3-clauses present before a free step and not the degree distribution of the literals (such heuristics were called "myopic" in [9]).

B. Randomness Invariance of the Formula in Each Round

Each atomic step of CL can be expressed as follows.

Procedure C.

1. Either select a literal occurrence τ in a clause of length one (forced atomic step)
2. or select a literal τ of degree i whose complement has degree j (free atomic step);
3. $\phi' \leftarrow \text{Del\&Shrink}(\phi, \tau)$.

Lemma 5.3. *At the end of each algorithmic operation according to Model C, the reduced formula remains random conditional on the number $|C_i|$ of i-clauses, $i = 2, 3$ and the number of literals of degree i whose negation has degree j, with $i, j \in \{0, 1, \ldots, 3|C_3| + 2|C_2|\}$.*

Proof. Model C is easily constructed by assuming that each literal-register described in *Model A* (see Lemma 4.4) is adjacent to an *exposed* degree-register that contains an integer i equal to the degree of its underlying literal and also contains an integer j equal to the degree of the negation of it, with $i, j \in \{0, 1, \ldots, 3|C_3| + 2|C_2|\}$.

In this way, an algorithm may select and set to TRUE a random literal of specified degree i whose negation has degree j during each free step. Once more we cannot infer information about the content of unexposed registers as soon as we complete the update of all the registers that remain undeleted. Therefore, the reduced formula remains random conditional on its current number of unexposed registers. ∎

Now, we easily truncate all the high degree literals in the above model as follows.

Lemma 5.4. *At the end of each algorithmic step of CL, the reduced formula remains random conditional on the number $|C_i|$ of i-clauses, $i = 2, 3$; the number $|\mathcal{X}_{i,j}|$ of complementary literals (see Definition 5.1), where h is a sufficiently high integer, say $h = 10$. More precisely, the formula is random given the vector*

$$\mathcal{S} = \langle \ell, c_3, c_2, x_{0,0}, x_{0,1}, \ldots x_{0,h}, \ldots, x_{h,0}, x_{h,1}, \ldots, x_{h,h} \rangle, \tag{5.1}$$

where $\ell = |L|/n; c_i = |C_i|/n; x_{i,j} = |\mathcal{X}_{i,j}|/n, i, j = 0, \ldots, h-1$, and n is the number of variables of the initial random formula.

Proof. The result follows easily by modifying slightly model C described in the proof of Lemma 5.3. Here, each literal-register is adjacent to an exposed degree-register, which

contains a pair of integers $(i,j) \in \{0,\dots,h\}^2$. In each such pair, integer i gives information about the degree of the underlying literal of this literal-register, while j gives information about the degree of the negation of it. If at least one integer of (i,j) equals h then the corresponding literal has degree $\geq h$ (no information is given about its exact degree). On the contrary, each integer $<h$ in (i,j) denotes the exact degree of the corresponding literal.

During each algorithmic step, the deletion of some clauses may cause some literals of initial degree $\geq h$ to finally get degree $j < h$. Although the degree content of such high degree-registers is secret, to perform the corresponding updates we need to know their exact degree during the simplification step. However, as soon as all updates are completed, it is not possible to infer the content of any unexposed register from the combined knowledge of current and previous information about the registers. ∎

C. Statistics of the Literals

Algorithm CL is initialized with a formula having the degree-sequence defined below.

Proposition 5.5. *A random 3-SAT formula of density c has w.h.p. the typical double degree sequence:*

$$x_{i,j} = \ell e^{-2(3c/\ell)} \frac{(3c/\ell)^{i+j}}{i!j!} + o(1), (i,j) \in \{0,\dots,h-1\}^2, i,j < h,$$

$$x_{i,h} = x_{h,i} = \ell e^{-(3c/\ell)} \frac{(3c/\ell)^{i}}{i!} \left(1 - \sum_{s=0}^{h-1} e^{-(3c/\ell)} \frac{(3c/\ell)^{s}}{s!}\right) + o(1), i < h,$$

$$x_{h,h} = \ell - \sum_{s,t\geq 0}^{h-1} x_{s,t} - 2\sum_{s=0}^{h-1} x_{s,h} + o(1).$$

Proof. The proof is analogous to the one given in Proposition 4.6. ∎

Theorem 5.6. *Any literal $\tau \in L$ and any literal occurrence b in a formula that is random given vector $\mathcal{S}$ in Definition 5.1 has the following properties:*

1. $P_h(\mu;k) = \Pr[\deg(\tau) = k |\ \tau \in \mathcal{X}_{h,j}] = \frac{\mu^k}{\left(e^{\mu} - \sum_{s=0}^{h-1} \frac{\mu^s}{s!}\right)k!}, k \geq h, j \leq h,$

   $\lambda_h = \frac{\mu\left(e^{\mu} - \sum_{j=0}^{h-2} \frac{\mu^j}{j!}\right)}{e^{\mu} - \sum_{j=0}^{h-1} \frac{\mu^j}{j!}},$

   where μ is the solution of the equation,

   $\lambda_h = \frac{p - \left(\sum_{s=0,t>s}^{h-1} (s+t)x_{s,t} + \sum_{s=0}^{h-1}(sx_{s,h} + sx_{s,s})\right)}{\sum_{s=0}^{h} x_{s,h}}$ *and*

   λ_h is the average load of a heavy bin.
2. $\Pr[\exists \tau : \deg(\tau) > \ln n |\ \tau \in \mathcal{X}_{h,j}] \leq e^{-(1-o(1))\ln n \ln\ln n}, j \leq h.$
3. $\Pr[\tau \in \mathcal{X}_{i,j}] = \frac{x_{i,j}}{\ell}, (i,j) \in A^2.$
4. $\Pr[\textit{Literal occurrence } b \in \mathcal{X}_{i,j}] = \frac{\zeta_i^h x_{i,j}}{p}, (i,j) \in A^2,$

   $$\textit{we define: } \zeta_i^h = \begin{cases} i, & i < h, \\ \lambda_h, & i = h. \end{cases}$$

5. $m = \mathbf{E}[\deg(\bar{b})|\ b \text{ is a literal occurrence}] = \frac{1}{p}\sum_{s,t=0}^{h} \zeta_s^h \zeta_t^h x_{s,t}$, *and*,
$m_i = m\frac{ic_i}{p}, i = 2, 3.$

6. $\varepsilon_1 = \mathbf{E}[\deg(b)$ *in 2, 3-clauses*$|\ b$ *is a literal occurrence in a 1-clause*$]$
$= \frac{1}{p}\left(\sum_{s=2}^{h-1}\sum_{t=0}^{h}(s-1)tx_{s,t} + \sum_{s=0}^{h} x_{h,s}\frac{\mu^2 e^{\mu} - \sum_{s=2}^{h-1}\frac{s(s-1)\mu^s}{s!}}{e^{\mu} - \sum_{s=0}^{h-1}\frac{\mu^s}{s!}}\right).$

Proof. 1. The proof is generalization to an arbitrary integer h of the analogous ones given in Theorems 4.7 and 4.8. We have

$$p_h n = \left(p - \left(\sum_{s=0,t>s}^{h-1}(s+t)x_{s,t} + \sum_{s=0}^{h-1}(sx_{s,h} + sx_{s,s})\right)\right)n$$

distinct balls (representing clause-registers with the degree-register of their underlying literal in the form $(h,j), j = 0,\ldots,h$) thrown uniformly at random into $x_h n = \sum_{s=0}^{h} x_{h,s} n$ distinct bins (representing literal-registers with their degree-register in the form $(h,j), j = 0,\ldots,h$) such that each bin gets load at least h. Then the probability mass of the number of literals of degree i, for any fixed $i \geq h$, follows a truncated at $h-1$ Poisson distribution:

$$P_h(\mu;k) = \frac{\mu^k}{\left(e^{\mu} - \sum_{s=0}^{h-1}\frac{\mu^s}{s!}\right)k!}, k \geq h, \text{ where } \mu \text{ is the solution the equation}$$

$$\lambda_h = \frac{\mu\left(e^{\mu} - \sum_{s=0}^{h-2}\frac{\mu^s}{s!}\right)}{e^{\mu} - \sum_{s=0}^{h-1}\frac{\mu^s}{s!}}, \text{ and } \lambda_h = p_h/x_h \text{ is the expected load of a heavy bin.}$$

2. Inequality (4.2) of Theorem 4.2 in [14] applies verbatim in the balls into bins game; therefore,

$$\Pr[\exists \text{ Heavy bin with load} > \ln n] \leq e^{-(1-o(1))\ln n \ln\ln n},$$

i.e., we have sharp concentration to the expected load.

3. Literal $\tau \in L$ corresponds to a random bin of ℓn possible and there are exactly $x_{i,j}n$ bins with underlying literals in the set $\mathcal{X}_{i,j}, (i,j) \in A^2$.

4. Each literal occurrence b corresponds to a ball, from $pn = (3c_3 + 2c_2)n$ possible, that u.a.r. lands to a bin. If $i < h$, the bins with underlying literals in $\mathcal{X}_{i,j}$ receive randomly $ix_{i,j}n$ balls, among pn possible, $j \in \{0,\ldots,h\}$. Therefore, the probability that a random literal occurrence b belongs into $\mathcal{X}_{i,j}$ is

$$\frac{i\mathcal{X}_{i,j}}{p}.$$

If $i = h$, consider a heavy literal $\tau \in \mathcal{X}_{h,j}$. From part 1 above, $\deg(\tau) = k \geq h$ with probability $P_h(\mu;k)$. Therefore, there are $P_h(\mu;k)x_{h,j}n$ literals (bins) in $\mathcal{X}_{h,j}$ each having exact degree $k \geq h$. Their corresponding bins contain exactly $kP_h(\mu;k)x_{h,j}n$ balls. Then in the bins of $\mathcal{X}_{h,j}, j \in \{0,\ldots,h\}$, there are exactly $\left(\sum_{k=h}^{\infty} kP_h(\mu;k)\right)x_{h,j}n = \lambda_h x_{i,j} n$ balls among pn possible balls. In this case the probability that a random literal occurrence b belongs into $\mathcal{X}_{h,j}$ is

$$\frac{\lambda_h \mathcal{X}_{h,j}}{p}.$$

5. Suppose that during a forced step a random literal occurrence b is selected. Observe that the degree k of $\bar{b}$ is dictated by the corresponding set $\mathcal{X}_{j,k}$ that the random literal

occurrence b may belong, $j = 1, \ldots, h$. In this way, if $k < h$ then $\Pr[\deg(\bar{b}) = k] = \Pr[b \in \mathcal{X}_{1,k} \cup \cdots \cup \mathcal{X}_{h,k}]$. Since the sets $\mathcal{X}_{j,k}, j = 1, \ldots, h$, are disjoint, by part 4 above we get

$$\Pr[\deg(\bar{b}) = k] = \frac{1x_{1,k} + 2x_{2,k} + \ldots + \lambda_h x_{h,k}}{p}, k < h. \tag{5.2}$$

If $k \geq h$, then

$$\begin{aligned}\Pr[\deg(\bar{b}) = k] &= \Pr[\deg(\bar{b}) = k \wedge (b \in \mathcal{X}_{1,h} \cup \ldots \cup \mathcal{X}_{h,h})] \\ &= \sum_{j=1}^{h} \Pr[\deg(\bar{b}) = k \wedge\ b \in \mathcal{X}_{j,h}],\end{aligned} \tag{5.3}$$

since the sets $\mathcal{X}_{j,h}, j = 1, \ldots, h$ are disjoint. From part 4 we obtain

$$\Pr[\deg(\bar{b}) = k \wedge\ b \in \mathcal{X}_{j,h}] = \frac{\zeta_j^h x_{j,h}}{p} P_h(\mu; k). \tag{5.4}$$

Summing (5.4) over $j = 1, \ldots, h$ and plugging in (5.2) we obtain for all k

$$\mathbf{Pr}[\deg(\bar{b}) = k] = \begin{cases} \frac{1x_{1,k} + 2x_{2,k} + \ldots + \lambda_h x_{h,k}}{p}, & k < h, \\ \frac{1x_{1,h} + 2x_{2,h} + \ldots + \lambda_h x_{h,h}}{p} P_h(\mu; k), & k \geq h. \end{cases} \tag{5.5}$$

Using (5.5) we obtain

$$\begin{aligned}\mathbf{E}[\deg(\bar{b})] &= \sum_{k=0}^{h-1} k \Pr[\deg(\bar{b}) = k] + \sum_{k=h}^{\infty} k \Pr[\deg(\bar{b}) = k] \\ &= \frac{1}{p} \sum_{k=0}^{h-1} \sum_{i=1}^{h} k \zeta_i^h x_{i,k} + \frac{1}{p} \sum_{k=h}^{\infty} k P_h(\mu; k) \sum_{i=1}^{h} \zeta_i^h x_{i,h} \\ &= \frac{1}{p} \sum_{k=0}^{h-1} \sum_{i=1}^{h} k \zeta_i^h x_{i,k} + \frac{1}{p} \zeta_h^h \sum_{i=1}^{h} \zeta_i^h x_{i,h} = m.\end{aligned}$$

6. First consider the case that the 1-clause literal occurrence b of total degree s appears in the 3, 2-clauses exactly $s - 1$ times, with $s < h$ (its occurrence in the 1-clause is subtracted). This happens iff $b \in \mathcal{X}_{s,1} \cup \cdots \cup \mathcal{X}_{s,h}$. These events are disjoint. That means that, if $s < h$ then b appears in $s - 1$ other clauses, excluding its 1-clause, with probability $\Pr[b \in \mathcal{X}_{s,1}] + \cdots + \Pr[b \in \mathcal{X}_{s,h}] = \frac{1}{p} \sum_{t=0}^{h} s x_{s,t}$, since each literal in $\mathcal{X}_{s,t}, 0 \leq t \leq h$, corresponds to a bin with exactly s balls. In this case we obtain the term

$$\frac{1}{p} \sum_{s=2}^{h-1} \sum_{t=0}^{h} (s-1) t x_{s,t}. \tag{5.6}$$

It remains to compute the expected occurrences of b in 3, 2-clauses excluding its 1-clause, that is to compute the expected $\deg(b) - 1$, when $b \in \mathcal{X}_{h,0} \cup \cdots \cup \mathcal{X}_{h,h}$. Since the above events are disjoint, we obtain

$$\begin{aligned}pr(k-1) &= \Pr[\deg(b) = k \wedge (b \in \mathcal{X}_{h,0} \cup \cdots \cup \mathcal{X}_{h,h})] \\ &= \Pr[\deg(b) = k \wedge\ b \in \mathcal{X}_{h,0}] + \cdots + \Pr[\deg(b) = k \wedge\ b \in \mathcal{X}_{h,h}].\end{aligned}$$

Consider the event: $\{\deg(b) = k \wedge\ b \in \mathcal{X}_{h,j}\}, 0 \le j \le h$. There are $x_{h,j}P_h(\mu;k)n$ literals (bins) in $\mathcal{X}_{h,j}$ containing $k \ge h$ balls each. Then b appears in these $x_{h,j}P_h(\mu;k)n$ bins of $\mathcal{X}_{h,j}$ with probability $\frac{1}{p}x_{h,j}P_h(\mu;k)k$, which equals the probability $\Pr[\deg(b) = k \wedge\ b \in \mathcal{X}_{h,j}], 0 \le j \le h$. Therefore, b appears in $k-1$ other 3, 2-clauses with probability $pr(k-1) = \frac{1}{p}P_h(\mu;k)k\sum_{j=0}^{h} x_{h,j}$. Summing over $k \ge h$ we get

$$\sum_{k=h}^{\infty}(k-1)pr(k-1) = \frac{1}{p}\sum_{j=0}^{h}x_{h,j}\sum_{k=h}^{\infty}(k-1)kP_h(\mu;k)$$
$$= \frac{1}{p}\sum_{j=0}^{h}x_{h,j}\frac{\mu^2e^{\mu} - \sum_{s=2}^{h-1}\frac{s(s-1)\mu^s}{s!}}{e^{\mu} - \sum_{s=0}^{h-1}\frac{\mu^s}{s!}}. \tag{5.7}$$

Summing (5.6) and (5.7) we get ε_1. ∎

D. Expected Changes per Round

Lemma 5.7. *Suppose that during round $t \in [0,1)$ the algorithm* CL *selects a pair of complementary literals $(\tau,\overline{\tau}) \in \mathcal{X}_{i,j}$ and sets τ to* TRUE*, with arbitrary fixed indices $(i,j) \in A^2$. Then the expected change of each parameter conditional on the current vector*

$$\mathcal{S} = \langle \ell, c_3, c_2, x_{0,0}, x_{0,1}, \ldots x_{0,h}, \ldots, x_{h,0}, x_{h,1}, \ldots, x_{h,h}\rangle$$

of the $(h+1)^2+3$ scaled parameters such that $\varepsilon_{\overline{\tau}} < 1$, are within $o(1)$ equal to

(a) $\mathbf{E}[\Delta[|L(t)|] \mid \mathcal{S}] = -2(1+\varepsilon_{\overline{\tau}})$,

(b) $\mathbf{E}[\Delta[|C_3(t)|] \mid \mathcal{S}] = -(\deg(\tau)+\deg(\overline{\tau}))\dfrac{3c_3}{p} - (\varepsilon_1+m)\dfrac{3c_3}{p}\varepsilon_{\overline{\tau}}$,

(c) $\mathbf{E}[\Delta[|C_2(t)|] \mid \mathcal{S}] = \deg(\overline{\tau})\frac{3c_3}{p} - (\deg(\tau)+\deg(\overline{\tau}))\frac{2c_2}{p}$
$+\left(m_3 - (\varepsilon_1+m)\frac{2c_2}{p}\right)\varepsilon_{\overline{\tau}}$,

(d) $\mathbf{E}[\Delta[|X_{i,j}(t)|] \mid \mathcal{S}] = \deg(\tau)\frac{6c_3+2c_2}{p^2}f(i,j) - \delta_{i,j}^{\tau,\overline{\tau}}$
$+\left(\varepsilon_1\frac{6c_3+2c_2}{p^2}f(i,j) - g(i,j)\frac{x_{ij}}{p}\right)\varepsilon_{\overline{\tau}}$,
$\forall (i,j) \in \{0,\ldots,h\}^2 \setminus (h,h)$,

where

$$\varepsilon_{\overline{\tau}} = \deg(\overline{\tau})\frac{2c_2}{p(1-m_2)},\ \textit{and}\ p = 3c_3+2c_2.$$

$$m_k = m\frac{kc_k}{p}, k = 2,3,\ \textit{also:}$$

$$\delta_{i,j}^{\tau,\overline{\tau}} = \begin{cases} 1, & i \ne j \textit{ and } i = \deg(\tau), j = \deg(\overline{\tau}) \textit{ or } i = \deg(\overline{\tau}), j = \deg(\tau), \\ 2, & i = j = \deg(\tau) = \deg(\overline{\tau}), \\ 0, & \textit{otherwise.} \end{cases}$$

$$f(i,j) = \begin{cases} (i+1)x_{i+1,j}\theta_i^h + (j+1)x_{i,j+1}\theta_j^h - (i+j)x_{i,j}, & i,j < h, \\ (k+1)x_{k+1,h} - kx_{k,h} - hx_{k,h}\theta_{h-1}^h, & (i,j) \in B, \\ hx_{h,h}\theta_{h-1}^h - (h-1)x_{h-1,h} - hx_{h-1,h}\theta_{h-1}^h, & (i,j) \in G, \end{cases}$$

where: $B = \{(h,k),(k,h)\}, k \le h-2$,

and: $G = \{(h-1,h),(h,h-1)\}$,

also: $\theta_s^h = \begin{cases} P_h(\mu;h), & s = h-1, \\ 1, & \textit{otherwise.} \end{cases}$

$$g(i,j) = \begin{cases} i+j, & 0 \le i,j \le h-1, \\ k+\lambda_h, & (i,j) \in B, \\ h-1+\lambda_h, & (i,j) \in G. \end{cases}$$

Initial conditions:

$$\ell = 2, c_3 = c, c_2 = 0.00005, \textit{ and each } x_{i,j}, (i,j) \in A^2 \textit{ is as in Proposition 5.5.}$$

Proof. (*a*) During the free step $\deg(\overline{\tau})$ clauses are shrunk. Among them, $\deg(\overline{\tau})\frac{2c_2}{p}$ many correspond to 2-clauses that are shrunk to 1-clauses. Then, each such 1-clause corresponds to the root of a Galton–Watson process that gives rise to the subsequent offspring of 1-clauses, according to sub-Section 5A. Each such process has a Malthus parameter equal to m_2, where this parameter is defined in Theorem 5.6, part 5. Therefore, $(1+\varepsilon_{\overline{\tau}})$ steps are expected in the round, and two literals are set and thus deleted per step.

(*b*) and (*c*) In the free step, the satisfied i-clauses are $\deg(\tau)ic_i/p$ while the unsatisfied are $\deg(\overline{\tau})ic_i/p$, in expectation, $i = 2,3$. This follows from Lemma 5.4, since each literal occurrence (ball) appears in an i-clause (deleting or shrinking it) with probability $ic_i/p, i = 2,3$. In each of the $\varepsilon_{\overline{\tau}}$ expected forced steps, we select a literal occurrence b that corresponds to a 1-clause. According to part 6 of Theorem 5.6, ball b is expected to appear in $\varepsilon_1\frac{ic_i}{p}$ other balls corresponding to i-clauses. From part 5 of Theorem 5.6, its negation $\overline{b}$ is expected to appear in $m\frac{ic_i}{p} = m_i$ other balls in i-clauses, $i = 2,3$. So in the forced steps of the round we expect to lose $(\varepsilon_1+m)\frac{3c_3}{p}\varepsilon_{\overline{\tau}}$ 3-clauses and $\left(m_3 - (\varepsilon_1+m)\frac{2c_2}{p}\right)\varepsilon_{\overline{\tau}}$ 2-clauses.

(*d*) First we compute the expected change of $\mathcal{X}_{i,j}$ with indices $i,j < h$. Consider the deletion of the neighboring literal occurrences (balls) to the evaluated to TRUE literal per step. In the free step, the evaluated to TRUE literal (a ball) is expected in $\deg(\tau)\frac{kc_k}{p}$ k-clauses deleting $\deg(\tau)\frac{kc_k}{p}(k-1)$ neighboring balls, $k = 3,2$. In each forced step, the selected literal is expected in $\varepsilon_1\frac{kc_k}{p}$ k-clauses deleting $\varepsilon_1\frac{kc_k}{p}(k-1)$ neighboring balls, $k = 3,2$. Flow into $\mathcal{X}_{i,j}$ is created by the deletion of balls that belong into $\mathcal{X}_{i+1,j}$ and $\mathcal{X}_{i,j+1}$ with corresponding probabilities $(i+1)x_{i+1,j}\theta_i^h\frac{1}{p}$ and $(j+1)x_{i,j+1}\theta_j^h\frac{1}{p}$. Flow out from $\mathcal{X}_{i,j}$ is created by the deletion of balls that belong to $\mathcal{X}_{i,j}$ and $\mathcal{X}_{j,i}$ with corresponding probabilities $ix_{i,j}\frac{1}{p}$ and $jx_{i,j}\frac{1}{p}$. Now, consider the deletion of the evaluated to TRUE literal τ and its negation $\overline{\tau}$ per step. This creates a flow out of $\mathcal{X}_{i,j}$ with probabilities $(i+j)x_{i,j}\frac{1}{p} = \Pr[\tau \in \mathcal{X}_{i,j} \cup \tau \in \mathcal{X}_{j,i}]$.

Next we compute the expected change of $\mathcal{X}_{i,j}$ with $(i,j) \in B$ where we define $B = \{(h,k),(k,h)\}, k < h-1$. Consider the deletion of the neighboring literal occurrences (balls) to the evaluated to TRUE literal per step. For example, flow into $\mathcal{X}_{k,h}, k < h-1$ is created from $\mathcal{X}_{k+1,h}$ with probability $(k+1)x_{k+1,h}\frac{1}{p}$. Flow out from $\mathcal{X}_{k,h}$ is created

with probability $(kx_{k,h} + hx_{k,h}\theta^h_{h-1})\frac{1}{p}$. Now, consider the deletion of the evaluated to TRUE literal τ and its negation per step. This creates a flow out from $\mathcal{X}_{k,h}$ with probabilities $(k+\lambda_h)x_{k,h}\frac{1}{p} = \Pr[\tau \in \mathcal{X}_{k,h} \cup \tau \in \mathcal{X}_{h,k}]$.

Finally, we compute the expected change of $\mathcal{X}_{i,j}$ with $(i,j) \in G$ where we define $G = \{(h-1,h),(h,h-1)\}$. First consider the deletion of the neighboring literal occurrences (balls). For example, flow into $\mathcal{X}_{h-1,h}$ is created from $\mathcal{X}_{h,h}$ with probability $hx_{h,h}\theta^h_{h-1}\frac{1}{p}$. Flow out from $\mathcal{X}_{h-1,h}$ is created with probability $((h-1)x_{h-1,h} - hx_{h-1,h}\theta^h_{h-1})\frac{1}{p} = \Pr[\text{neighboring ball} \in \mathcal{X}_{h-1,h}] + \Pr[\text{neighboring ball} \in \mathcal{X}^{\sharp}_{h,h-1}]$. Now, consider the deletion of the evaluated to TRUE literal τ and its negation per step. This creates a flow out from $\mathcal{X}_{h-1,h}$ with probabilities $(h-1+\lambda_h)x_{h-1,h}\frac{1}{p} = \Pr[\tau \in \mathcal{X}_{k,h} \cup \tau \in \mathcal{X}_{h,k}]$. ∎

E. Wormald's Theorem and Differential Equations

We can show that conditions (i′)–(iii) of Theorem 2 in [72] hold working analogously as in sub-Section 4H. The proof is omitted. This allows us to approximate within $o(1)$ error and probability $1 - o(1)$ the trajectories of the expected changes described in Lemma 5.7 by the solution of the following system of differential equations.

Lemma 5.8. *Suppose that for $O(n)$ rounds the algorithm* CL *selects pairs of complementary literals $(\tau,\overline{\tau}) \in \mathcal{X}_{i,j}$ and sets τ to* TRUE*, with arbitrary fixed indices $(i,j) \in A^2$.*

Then the $(h+1)^2+3$ parameters in the vector

$$\mathcal{S} = \langle \ell, c_3, c_2, x_{0,0}, x_{0,1}, \ldots x_{0,h}, \ldots, x_{h,0}, x_{h,1}, \ldots, x_{h,h}\rangle$$

are approximated within $o(1)$ and with probability $1-o(1)$ by the solution of the following system of differential equations:

$$(a)\quad \frac{d\ell}{dt} = -2(1+\varepsilon_{\overline{\tau}}),$$

$$(b)\quad \frac{dc_3}{dt} = -(\deg(\tau)+\deg(\overline{\tau}))\frac{3c_3}{p} - (\varepsilon_1+m)\frac{3c_3}{p}\varepsilon_{\overline{\tau}},$$

$$(c)\quad \frac{dc_2}{dt} = \deg(\overline{\tau})\frac{3c_3}{p} - (\deg(\tau)+\deg(\overline{\tau}))\frac{2c_2}{p} + \left(m_3 - (\varepsilon_1+m)\frac{2c_2}{p}\right)\varepsilon_{\overline{\tau}},$$

$$(d)\quad \frac{dx_{i,j}}{dt} = \deg(\tau)\tfrac{6c_3+2c_2}{p^2}f(i,j) - \delta^{\tau,\overline{\tau}}_{i,j} + \left(\varepsilon_1\tfrac{6c_3+2c_2}{p^2}f(i,j) - g(i,j)\tfrac{x_{i,j}}{p}\right)\varepsilon_{\overline{\tau}},$$
$$\forall (i,j) \in \{0,\ldots,h\}^2 \setminus (h,h),$$

where

$$\varepsilon_{\overline{\tau}} = \deg(\overline{\tau})\frac{2c_2}{p(1-m_2)},\ \textit{and}\ p = 3c_3+2c_2.$$

$$m_k = m\frac{kc_k}{p}, k = 2,3,\ \textit{also:}$$

$$\delta^{\tau,\overline{\tau}}_{i,j} = \begin{cases} 1, & i \neq j \textit{ and } i = \deg(\tau), j = \deg(\overline{\tau}) \textit{ or } i = \deg(\overline{\tau}), j = \deg(\tau), \\ 2, & i = j = \deg(\tau) = \deg(\overline{\tau}), \\ 0, & \textit{otherwise.} \end{cases}$$

$$f(i,j) = \begin{cases} (i+1)x_{i+1,j}\theta_i^h + (j+1)x_{i,j+1}\theta_j^h - (i+j)x_{i,j}, & i,j < h, \\ (k+1)x_{k+1,h} - kx_{k,h} - hx_{k,h}\theta_{h-1}^h, & (i,j) \in B, \\ hx_{h,h}\theta_{h-1}^h - (h-1)x_{h-1,h} - hx_{h-1,h}\theta_{h-1}^h, & (i,j) \in G, \end{cases}$$

where: $B = \{(h,k),(k,h)\}, k \leq h-2$,

and: $G = \{(h-1,h),(h,h-1)\}$,

also: $\theta_s^h = \begin{cases} P_h(\mu;h), & s = h-1, \\ 1, & \textit{otherwise.} \end{cases}$

$$g(i,j) = \begin{cases} i+j, & 0 \leq i,j \leq h-1, \\ k+\lambda_h, & (i,j) \in B, \\ h-1+\lambda_h, & (i,j) \in G. \end{cases}$$

Initial conditions:

$$\ell = 2, c_3 = c, c_2 = 0.00005, \textit{ and each } x_{i,j}, (i,j) \in A^2 \textit{ is as in Proposition 5.5.}$$

Proof. We show that conditions (i′)–(iii) of Theorem 2 in [72] hold working analogously as in sub-Section 4H. ∎

F. Implementation and Termination of the Algorithm

We plugged into the system of d.e. of Lemma 5.8 initial conditions corresponding to $r_3 = 3.52$. At round 0 we computed $m_2(0)$ and $m_3(0)$. Recall from part 5 of Theorem 5.6 that $m_i(t)$ equals the expected number of unsatisfied i-clauses in each *forced* step during round t, $i = 2,3$. For each $(i,j) \in A^2$, we performed $\epsilon = 1/100000$ rounds (restarting from round 0 each time) and we computed the corresponding $R_1(i,j) = \frac{m_2(\epsilon)-m_2(0)}{m_3(\epsilon)-m_3(0)}$. Let $R_1(i_1,j_1)$ be the maximum value. This preprocessing step of computing $R_1(i_1,j_1)$ corresponds to the procedure `Choose-Bucket` of algorithm `CL` and the pair of indices (i_1,j_1) is returned. Then we restarted from round 0 and for $T_1 = O(n)$ rounds we always set literals from $\mathcal{X}_{i_1,j_1}$. Similarly, we performed ϵ rounds (this time starting from round T_1 each time) and we computed the corresponding $R_2(i,j) = \frac{m_2(T_1+\epsilon)-m_2(T_1)}{m_3(T_1+\epsilon)-m_3(T_1)}, \forall (i,j) \in A^2$. Let $R_2(i_2,j_2)$ be the new maximum value (see procedure `Choose-Bucket`). Now, we restarted from round T_1 and for $T_2 = O(n)$ rounds we always set literals from $\mathcal{X}_{i_2,j_2}$, etc., taking always into account that each scaled parameter in $\mathcal{S}$ remains >0. For initial density $r_3 = 3.52$ the Malthus parameter m_2 remained always <1. We simulated the differential equations until we reached a round t^* such that we could apply Lemma 5.9 and safely terminate the algorithm. Applying Theorem 4.13 proved by Cooper, Frieze, and Sorkin in [19] (which is included in sub-Section 4J), we end up with a proof for Lemma 5.9. This lemma is a generalization of Lemma 4.14.

Lemma 5.9. *A random formula given $\mathcal{S}$ in (4.1) is almost surely satisfiable if it holds:* $2D_2 \leq (1-\epsilon)D_1'$, *where* $D_1' = 2(c_2+c_3)n$ *and*

$$D_2 = \frac{1}{2}\left(\sum_{s,t\geq 0}^{h-1} stx_{s,t} + 2\lambda_h \sum_{s=0}^{h-1} sx_{h,s} + \lambda_h^2 x_{h,h}\right) n.$$

Recall that λ_h, defined in part 2 of Theorem 4.8, is the expected load of a heavy bin.

Proof. Consider a random formula ϕ given the current degree sequence $\mathcal{S}$. From part 2 of Theorem 4.8 it holds w.h.p. that the maximum degree Δ_d of any literal in $\mathcal{S}$ is at most $\ln n < n^{\alpha}, \alpha < 1/13$. Since the current formula ϕ consists of 3, 2-clauses, delete exactly one random clause-register (literal occurrence) from each 3-tuple clause-register. Such deletions are feasible since clause-registers are exposed; see Lemma 5.4. This results in a 2-SAT formula ϕ' with $(c_3+c_2)n$ 2-tuples of clause-registers. Almost sure satisfiability of ϕ' implies almost sure satisfiability of ϕ. ϕ' is random given a new degree sequence $\mathcal{S}'$. Since we delete literal occurrences, then for each variable x_i its *new* numbers of positive d_i' and negative $\overline{d}_i'$ occurrences are at most equal to its old $d_i, \overline{d}_i$. That is, $D_2 = \sum_{i=1}^{n} d_i\overline{d}_i \geq D_2' = \sum_{i=1}^{n} d_i'\overline{d}'_i$. The new total number of literal occurrences is $D_1' = 2(c_3 + c_2)n$. Since $D_2 \geq D_2'$ we obtain the following useful inequality,

$$2\frac{D_2}{D_1'} \geq 2\frac{D_2'}{D_1'}, \tag{5.8}$$

which allows us to compute easily D_2 instead of D_2'.

To that end, we work as follows.

For $t, s \in N$ define $\mathrm{Var}_{t,s} = |\{x_i \in V \mid x_i \in \mathcal{X}_{t,s}^{\sharp}\}|$ where $\mathcal{X}_{t,s}^{\sharp}$ consists of literals with degree exactly t while their complements have degree s.

$$\begin{aligned} D_2 = \sum_{i=1}^{n} d_i\overline{d}_i = \sum_{s,t} st\,\mathrm{Var}_{s,t} = \sum_{s=0}^{h-1}\sum_{t=0}^{h-1} st\,\mathrm{Var}_{s,t} + \sum_{s=h}^{\infty}\sum_{t=0}^{h-1} st\,\mathrm{Var}_{s,t} \\ + \sum_{s=0}^{h-1}\sum_{t=h}^{\infty} st\,\mathrm{Var}_{s,t} + \sum_{s=h}^{\infty}\sum_{t=h}^{\infty} st\,\mathrm{Var}_{s,t}. \end{aligned} \tag{5.9}$$

If $s, t < h$ and $x_i \in V$, then

$$\Pr\{x_i \in \mathcal{X}_{t,s}^{\sharp}\} = \frac{x_{s,t}}{\ell} = \frac{\mathrm{Var}_{s,t}}{(\ell/2)n} \Leftrightarrow \mathrm{Var}_{s,t} = \frac{x_{s,t}}{2}n. \tag{5.10}$$

If $s \geq h, t < h$ (by symmetry of indices the case $s < h, t \geq h$ is similar) then

$$\Pr\{x_i \in \mathcal{X}_{t,s}^{\sharp}\} = \frac{x_{h,t}P_h(\mu;s)}{\ell} = \frac{\mathrm{Var}_{s,t}}{(\ell/2)n} \Leftrightarrow \mathrm{Var}_{s,t} = \frac{x_{h,t}P_h(\mu;s)}{2}n. \tag{5.11}$$

It remains the case $s \geq h, t \geq h$:

$$\begin{aligned} \Pr\{x_i \in \mathcal{X}_{t,s}^{\sharp}\} = \frac{x_{h,h}P_h(\mu;s)P_h(\mu;t)}{\ell} = \frac{\mathrm{Var}_{s,t}}{(\ell/2)n} \\ \Leftrightarrow \mathrm{Var}_{s,t} = \frac{x_{h,h}P_h(\mu;s)P_h(\mu;t)}{2}n \end{aligned} \tag{5.12}$$

We plug (5.10), (5.11), and (5.12) into the corresponding sums of (5.9). Notice that the expectation $\sum_{s=h}^{\infty} sP_h(\mu;s)$ of the truncated Poisson distribution equals λ_h. Then we easily obtain D_2 that appears in Lemma 5.9. ∎

APPENDIX

TABLE 1. Comparison of Values Obtained from the Numerical Solution of the Differential Equations with the Values Given by the Simulation

	t	ℓ	ρ_2	ρ_3	x_0	x_1	x_2	x_8
d.e.	0.000000	2.000000	0.000000	3.420000	0.011833	0.060703	0.155705	0.140772
sim.	0.000000	2.000000	0.000000	3.420000	0.012078	0.060154	0.155992	0.141078
d.e.	0.010000	1.979478	0.051372	3.294281	0.013127	0.065913	0.165471	0.131459
sim.	0.010000	1.979436	0.051455	3.294243	0.013392	0.065444	0.165650	0.131904
d.e.	0.020000	1.957801	0.102948	3.165518	0.014615	0.071725	0.176000	0.121703
sim.	0.020000	1.957622	0.102767	3.165561	0.014962	0.071308	0.175780	0.122222
d.e.	0.030000	1.934832	0.154631	3.033612	0.016330	0.078224	0.187345	0.111393
sim.	0.030000	1.934700	0.154814	3.033580	0.016772	0.077558	0.187116	0.111658
d.e.	0.040000	1.910425	0.206383	2.898391	0.018320	0.085510	0.199558	0.100470
sim.	0.040000	1.910484	0.206123	2.898846	0.018770	0.084898	0.199070	0.100874
d.e.	0.050000	1.884366	0.258116	2.759672	0.020639	0.093699	0.212686	0.088941
sim.	0.050000	1.884208	0.258151	2.759493	0.021228	0.092942	0.212402	0.089040
d.e.	0.050510	1.882988	0.260752	2.752500	0.020767	0.094143	0.213381	0.088338
sim.	0.050510	1.882800	0.260922	2.752160	0.021382	0.093422	0.213088	0.088480
d.e.	0.060000	1.856401	0.310999	2.632164	0.022984	0.101647	0.224767	0.077376
sim.	0.060000	1.856256	0.311209	2.631981	0.023624	0.100960	0.224676	0.077812
d.e.	0.070000	1.826188	0.364454	2.500700	0.025703	0.110515	0.237584	0.064064
sim.	0.070000	1.826040	0.364570	2.500637	0.026374	0.109922	0.237944	0.064476
d.e.	0.076105	1.806463	0.397338	2.417882	0.027596	0.116482	0.245831	0.055433
sim.	0.076105	1.806744	0.396826	2.418985	0.028210	0.115870	0.246294	0.055834
d.e.	0.080000	1.793301	0.418885	2.367798	0.028785	0.120138	0.250699	0.046740
sim.	0.080000	1.793068	0.419080	2.367642	0.029412	0.119574	0.251230	0.047058
d.e.	0.090000	1.757114	0.474996	2.234567	0.032230	0.130388	0.263740	0.027271
sim.	0.090000	1.756488	0.475355	2.233755	0.032886	0.130000	0.263852	0.027436
d.e.	0.104000	1.669236	0.555691	2.034841	0.038258	0.147221	0.283259	0.006595
sim.	0.104000	1.699116	0.556006	2.035145	0.038954	0.146628	0.283420	0.006798
d.e.	0.109955	1.671313	0.590879	1.944219	0.041385	0.155450	0.291943	0.000005
sim.	0.109955	1.671960	0.590224	1.946347	0.042202	0.154384	0.292072	0.000180
d.e.	0.120000	1.617999	0.654258	1.790959	0.047159	0.169702	0.305322	0.000005
sim.	0.120000	1.617592	0.654565	1.790711	0.047866	0.169326	0.305842	0.000000
d.e.	0.130000	1.553796	0.721099	1.620394	0.054604	0.186559	0.318659	0.000004
sim.	0.130000	1.554480	0.720315	1.622145	0.055456	0.186224	0.318190	0.000000
d.e.	0.140000	1.470975	0.793518	1.421630	0.064857	0.207177	0.330796	0.000003
sim.	0.140000	1.475280	0.789439	1.430790	0.065046	0.206406	0.329776	0.000000
d.e.	0.150000	1.346948	0.880189	1.170968	0.080346	0.232620	0.336419	0.000002
sim.	0.150000	1.346224	0.880747	1.169965	0.080994	0.232604	0.335538	0.000000
d.e.	0.155000	1.235824	0.936604	0.979773	0.094423	0.249925	0.330070	0.000001
sim.	0.155000	1.234012	0.937041	0.978266	0.095532	0.249156	0.329176	0.000000
d.e.	0.170000	0.600820	0.803173	0.222643	0.179484	0.225146	0.137641	0.000000
sim.	0.170000	0.595608	0.797578	0.219674	0.179668	0.223954	0.135280	0.000000
d.e.	0.171605	0.585324	0.782149	0.207836	0.182467	0.221233	0.130492	0.000000
sim.	0.171605	0.579056	0.774584	0.203503	0.183088	0.219382	0.127930	0.000000

ACKNOWLEDGMENTS

Discussions of the second author with D. Achlioptas were crucial in developing the ideas in this work. Also we thank two anonymous referees for their comments that helped us to improve the presentation of this work.

REFERENCES

[1] D. Achlioptas, Setting two variables at a time yields a new lower bound for random 3-SAT, Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC '00), pp. 28–37.

[2] D. Achlioptas, Lower bounds for random 3-SAT via differential equations. Theor Comput Sci 265 (2001), 159–185.

[3] D. Achlioptas, P. Beame, and M. Molloy, A sharp threshold in proof complexity. Proc. 31st Annual ACM Symposium on Theory of Computing (STOC '01), pp. 337–346.

[4] D. Achlioptas and M. Molloy, The analysis of a list-coloring algorithm on a random graph. Proc. 38th Annual Symposium on Foundations of Computer Science (FOCS '97), pp. 204–212.

[5] D. Achlioptas and C. Moore, Almost all graphs with average degree 4 are 3-colorable. Proc. 34th Annual ACM Symposium on Theory of Computing (STOC '02), pp. 199–208.

[6] D. Achlioptas and C. Moore, The asymptotic order of the random k-SAT threshold. Proc. 43rd Annual Symposium on Foundations of Computer Science (FOCS '02), pp. 779–788.

[7] D. Achlioptas and C. Moore, Random k-SAT: Two moments suffice to cross a sharp threshold, SIAM Journal of Computing, to appear.

[8] D. Achlioptas and Y. Peres, The threshold for random k-SAT is $2^k(\ln 2 + o(1))$. Proc. 35th Annual ACM Symposium on Theory of Computing (STOC '03).

[9] D. Achlioptas and G. B. Sorkin, Optimal myopic algorithms for random 3-SAT. Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS '00), pp. 590–600.

[10] E. A. Bender and E. R. Canfield, The asynptotic number of labelled graphs with given degree sequences. J Comb Theory A 24 (1978), 296–307.

[11] B. Bollobás, Random Graphs, Academic Press, London, 1985.

[12] B. Bollobás, C. Borgs, J. T. Chayes, J. H. Kim, and D. B. Wilson, The scaling window of the 2-SAT transition. Random Struct Algor 18 (2001), 201–256.

[13] D. Brélaz, New methods to color the vertices of a graph. Commun ACM 22 (1979), 251–256.

[14] A. Broder, A. Frieze, and E. Upfal, On the satisfiability and maximum satisfiability of random 3-CNF formulas. Proc. 4th ACM-SIAM Symposium on Discrete Algorithms (SODA '93), pp. 322–330.

[15] M. T. Chao and J. Franco, Probabilistic analysis of two heuristics for the 3-satisfiability problem. SIAM J Comput 15 (1986), 1106–1118.

[16] M. T. Chao and J. Franco, Probabilistic analysis of a generalization of the unit-clause litteral selection heuristics for the k-satisfiability problem. Inform Sci 51 (1990), 289–314.

[17] V. Chvátal and B. Reed, Mick gets some (the odds are on his side). Proc. 33rd Annual Symposium on the Foundation of Computer Science (FOCS '92), pp. 620–627.

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

[19] C. Cooper, A. Frieze, and G. B. Sorkin, A note on random 2-SAT with prescribed literal degrees. Proc. 13th ACM-SIAM Symposium on Discrete Algorithms (SODA '02), pp. 316–320.

[20] M. J. Crawford and L. D. Auton, Experimental results on the crossover point in random 3-SAT. Artifi Intelli 81 (1996), 31–57.

[21] M. Davis, G. Logemann, and D. Loveland, A machine program for theorem-proving. Commun ACM, 5 (1962), 394–397.

[22] M. Davis and H. Putnam, A computing procedure for quantification theory. J Assoc Comput Mach 7 (1960), 201–215.

[23] L. Devroye, Branching processes and their applications in the analysis of tree structures and tree algorithms. In: M. Habib, C. McDiarmid, R. Alfonsin, and B. Reed (eds.): Probabilistic Methods for Algorithmic Discrete Mathematics. Lecture Notes in Computer Science, Springer-Verlag, Berlin (1998), pp. 249–314.

[24] O. Dubois, Upper bounds on the satisfiability threshold. Theor Comput Sci 265 (2001), 187–197.

[25] O. Dubois, P. André, Y. Boufkhad, and J. Carlier, SAT versus UNSAT. In: D. S. Johnson and M. A. Trick (eds.): Second DIMACS Challenge. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. AMS (1993), 415–436.

[26] O. Dubois and Y. Boufkhad, A general upper bound for the satisfiability threshold of random r-SAT. J Algor 24 (1997), 395–420.

[27] O. Dubois, Y. Boufkhad, and J. Mandler, Typical random 3-SAT formulae and the satisfiability threshold. Proc. 11th Symposium on Discrete Algorithms (SODA '00), pp. 126–127. Available at: `http://www.eccc.uni-trier.de/eccc-local/Lists/TR-2003.html`

[28] O. Dubois and J. Mandler, On the 3-colourability of random graphs. Available at: `http://arxiv.org/abs/math.CO/0209087`

[29] J. Franco, Probabilistic analysis of the pure literal heuristic for the satisfiability problem. Ann Oper Res 1 (1984), 273–289.

[30] J. Franco, Results related to threshold phenomena research in satisfiability: Lower Bounds. Theor Comput Sci 265 (2001), 147–157.

[31] J. Franco and M. Paull, Probabilistic analysis of the Davis–Putnam procedure for solving the satisfiability problem. Discrete Appl Math 5 (1983), 77–87.

[32] E. Friedgut (Appendix by J. Bourgain), Sharp thresholds of graph properties, and the k-SAT problem. J AMS 12 (1997), 1017–1054.

[33] A. Frieze and S. Suen, Analysis of two simple heuristics for random instances of k-SAT. J Algor 20 (1996), 312–355.

[34] A. Frieze and N. Wormald, k-SAT: A tight threshold for moderately growing k. Proc. 5th International Symposium on the Theory and Applications of Satisfiability Testing (2002), pp. 1–6.

[35] I. Gent and T. Walsh, The SAT phase transition. In A. Cohn (ed.): 11th European Conference on Artificial Intelligence, Wiley, New York, 1994.

[36] C. Giannella, On extending two threshold algorithms to non-threshold algorithms by attaching the unit clause rule, Master Thesis, University of Indiana, (1999). Available at: `http://www.cs.indiana.edu/~cgiannel/work.html`

[37] A. Goerdt, A threshold for unsatisfiability. J Comput Syst Sci 33 (1996), 469–486.

[38] M. T. Hajiaghayi and G. Sorkin, personal communication, 2002.

[39] A. Haken, The intractability of resolution. Theor Comp Sci 39 (1985), 297–308.

[40] G. Istrate, Phase transitions and all that. Submitted. CoRRcs.CC/0211012 (2002).

[41] S. Janson, T. Łuczak, and A. Ruciński, Random graphs. Wiley, New York, 2000.

[42] S. Janson, Y. C. Stamatiou, and M. Vamvakari, Bounding the unsatisfiability threshold of random 3-SAT. Random Struct Algor 17 (2000), 103–116.

[43] D. S. Johnson, Approximation algorithms for combinatorial problems. J Comp Syst Sci 9 (1974), 256–278.

[44] A. Kamath, R. Motwani, K. Palem, and P. Spirakis, Tail bounds for occupancy and the satisfiability threshold conjecture. Random Struct Algor 7 (1995), 59–80.

[45] A. C. Kaporis, L. M. Kirousis, and E. G. Lalas, The probabilistic analysis of a greedy satisfiability algorithm. In: Proc. 10th Annual European Symposium on Algorithms (ESA 2002), Track A, pp. 574–585.

[46] A. C. Kaporis, L. M. Kirousis, and Y. C. Stamatiou, How to prove conditional randomness using the Principle of Deferred Decisions. Special Volume on Computational Complexity and Statistical Physics. Santa Fe Institute, Studies in the Sciences of Complexity. Oxford University Press. To appear. Available at: `www.ceid.upatras.gr/faculty/kirousis/kks-pdd02.ps`

[47] A. C. Kaporis, L. M. Kirousis, Y. C. Stamatiou, M. Vamvakari, and M. Zito, The unsatisfiability threshold revisited. Discreto Mathematics, Elsevier, to appear.

[48] S. Kirkpatrick, G. Györgyi, N. Tishby, and L. Troyansky, In: J. Cowan, G. Tesauro, and J. Alspector (eds.): The statistical mechanics of k-satisfaction. Adv Neur Inform Proc Syst 6 (1993), 439–446.

[49] S. Kirkpatrick and B. Selman, Critical behavior in the satisfiability of random Boolean expressions. Science 264 (1994), 1297–1301.

[50] L. M. Kirousis, E. Kranakis, D. Krizanc, and Y. C. Stamatiou, Approximating the unsatisfiability threshold of random formulas. Random Struct Algor 12 (1998), 253–269.

[51] L. M. Kirousis, Y. C. Stamatiou, and M. Zito, The unsatisfiability threshold conjecture: The techniques behind upper bound improvements. Special Volume on Computational Complexity and Statistical Physics. Santa Fe Institute, Studies in the Sciences of Complexity. Oxford University Press. To appear.

[52] D. E. Knuth, Stable marriage and its relation to other combinatorial problems: an introduction to the mathematical analysis of algorithms (English edition: CRM Proceedings & Lecture Notes 10, American Mathematical Society, 1997; first French edition: Les Presses de l'Université de Montréal, 1976). Also see: D. Knuth, R. Motwani, and B. Pittel, Stable husbands, Random Struct Algor 1 (1990), 1–14.

[53] M. Maftouhi and W. Fernandez de la Vega, On random 3-SAT. Combin Probabil Comp 4 (1995), 190–195.

[54] B. McKay and N. C. Wormald, The degree sequence of a random graph. I. The models. Random Struct Algor 11 (1997), 97–117.

[55] G. Parisi, M. Mézard and R. Zecchina, Analytic and algorithmic solution of random satisfiability problems. Science 297 (2002), 812.

[56] M. Mézard and R. Zecchina, The random k-Satisfiability problem: From an analytic solution to an efficient algorithm. Physical Review E, Vol. 66, (2002).

[57] `http://www.mathworks.com/`

[58] D. Mitchell, B. Selman, and H. Levesque, Hard and easy distribution of SAT problems. Proc. 10th National Conference on Artificial Intelligence (AAAI '92), pp. 459–465.

[59] M. Mitzenmacher, Tight thresholds for the pure literal rule. TR, Digital Equipment Corporation (1997). Available at: www.research.compaq.com/SRC/

[60] M. Molloy, The probabilistic method. In: M. Habib, C. McDiarmid, R. Alfonsin, and B. Reed (eds.): Probabilistic Methods for Algorithmic Discrete Mathematics. Lecture Notes in Computer Science, Springer-Verlag, Berlin (1998), pp. 1–35.

[61] R. Monasson and R. Zecchina, Statistical mechanics of the random k-Sat problem. Phys Rev E 56 (1997), 1357–1361.

[62] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, Determining computational complexity from characteristic phase transitions. Nature 400 (1999), 133–137.

[63] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, $2+p$-SAT Relation of typical-case complexity to the nature of the phase transition. Random Struct Algor 15 (1999), 414–435.

[64] B. Pittel, J. Spencer, and N. C. Wormald, Sudden emergence of a giant k-core in a random graph. J Combin Theor B 67 (1996), 111–151.

[65] SAT–The Satisfiability Library, www.intellektik.informatik.tu-darmstadt.de/SATLIB/

[66] A. Urquhart, Hard examples for resolution. J Assoc Comput Mach 34 (1987), 209–219.

[67] W. Fernandez de la Vega, On random 2-SAT. Manuscript (1992).

[68] Y. Verhoeven, Random 2-SAT and unsatisfiability. Inform Proc Lett 72 (1999), 119–123.

[69] J. S. Vitter and P. Flajolet, Average-case analysis of algorithms and data structures. In: J. van Leeuwen (ed.): Algorithms and Complexity. Handbook of Theoretical Computer Science **A**, MIT Press, Amsterdam (1990), pp. 431–524.

[70] D. Wilson, On the critical exponents of random k-SAT. Random Structures & Algorithms, 21(2), 182–195 (2002).

[71] N. C. Wormald, Models of random regular graphs. In: J. D. Lamb and D. A. Preece (eds.): Surveys in Combinatorics London Mathematical Society Lecture Note Series. Cambridge University Press Cambridge, Vol. 276 (1999), pp. 239–298.

[72] N. C. Wormald, Differential equations for random processes and random graphs. Ann Appl Probabil 5 (1995), 1217–1235.

[73] N. C. Wormald, Some problems in the enumeration of labelled graphs, Ph.D. Thesis, Newcastle University (1978).

[74] A. Steger and N. C. Wormald, Generating random regular graphs quickly. Combin Probabil Comput 8 (1999), 377–396.