The probabilistic analysis of a greedy satisfiability algorithm

Alexis C. Kaporis *† Lefteris M. Kirousis *† Efthimios G. Lalas *

Abstract

Consider the following simple, greedy Davis-Putnam algorithm applied to a random 3-CNF formula of fixed density (clauses to variables ratio): Arbitrarily select and set to True a literal that appears in as many clauses as possible, irrespective of their size (and irrespective of the number of occurrences of the negation of the literal). Delete these clauses from the formula, and also delete the negation of this literal from any clauses it appears. Repeat. If however unit clauses ever appear, then first repeatedly and in any order set the literals in them to True and delete and shrink clauses accordingly, until no unit clause remains. Also if at any step an empty clause appears, then do not backtrack, but just terminate the algorithm and report failure. A slight modification of this algorithm is probabilistically analyzed in this paper (rigorously). It is proved that for random formulas of n variables and density up to 3.42, it succeeds in producing a satisfying truth assignment with bounded away from zero probability, as n approaches infinity. Therefore the satisfiability threshold is at least 3.42. Previously, Davis-Putnam algorithms of increasing sophistication were analyzed to obtain a lower bound of at most 3.26. Preliminary experiments we performed suggest that the value 3.42 can be considerably increased (up to approximately 3.6) with algorithms like the above, which for the selection of the literal to be set take into account not only its number of occurrences but also the number of occurrences of its negation, but ignore the size of the clauses were the literal appears (apart of the unit-clause, "forced" steps). So, perhaps, for formulas with density up to (a number close to) the satisfiability threshold, a satisfying truth assignment can be found by simple Davis-Putnam algorithms based only on the number of occurrences of the literals.

April 25, 2002

Please do not circulate

*University of Patras, Department of Computer Engineering and Informatics, University Campus, GR-265 04 Patras, Greece. Authors' e-mail addresses: {kaporis,kirousis,lalas}@ceid.upatras.gr.

†Research supported by the University of Patras Research Committee (Project Carathéodory under contract no. 2445) and by the Computer Technology Institute.

1 Preliminaries

We consider random 3-CNF formulas in the standard $G_{n,m}$ model. This means that a random formula is constructed by choosing m clauses uniformly, independently and with replacement from the set of clauses on n variables that contain exactly three literals. The *density* c of a formula is the ratio m/n, i.e. c is the scaled, with respect to n, number of clauses. The *degree* of a literal is the number of its occurrences in the formula. If the formula contains clauses of varying size, then any occurrence of a literal, irrespective of the size of the clause where it appears, counts towards the degree.

Previously, Davis-Putnam algorithms of increasing sophistication were rigorously analyzed towards showing that random formulas up to a certain density are a.s. (as n approaches infinity) satisfiable (see [8, 9, 7, 13, 1, 5]). The best lower bound for the satisfiability threshold thus obtained is 3.26 [5]. These algorithms (with the exception of the "plain" pure literal algorithm, which succeeds for formulas of density up to 1.63 [7, 16]) take into account the size of the clauses were the literal to be set appears.

We describe here a greedy Davis-Putnam algorithm that in selecting the literal to be set takes into account only how many clauses, irrespective of their size, will be satisfied. Its probabilistic analysis is based on the method of differential equations studied by Wormald in [20]. As the algorithm is rather simple, its analysis is also relatively easy. This simplicity, contrasted with the improvement over the lower bounds of previous analyses, is an interesting, we think, aspect of this work.

The algorithm is applied to a 3-CNF formula with n variables and density c. Let h be an a priori decided integer parameter. At a first phase, the algorithm arbitrarily selects and sets to TRUE literals of degree *at least* h, 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. 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). Let $\mathcal{X}_j, j = 0, \dots, h-1$, be the current collection of literals of degree j, and let $\mathcal{X}_h$ be the current collection of literals of degree at least h (although this notation is not uniform for $j < h$ and for h, it is convenient). Literals in $\mathcal{X}_h$ are called **h***eavy*. Del&Shrink is the usual procedure of any Davis-Putnam algorithm where clauses that contain the literal that is set to TRUE are deleted, whereas clauses that contain its negation are shrunk by deleting it from them. Formally, the algorithm, is the following:

Greedy Algorithm
begin
let j=h;
while unset literals exist do
    while $\mathcal{X}_j \neq \emptyset$ do
        set an arbitrary literal in $\mathcal{X}_j$ to TRUE, its negation to FALSE and Del&Shrink;
        while unit clauses exist do
            set the literal of an arbitray unit clause to TRUE, its negation to FALSE and Del&Shrink;
            if an empty clause is generated then report failure and terminate the algorithm;
        end do
    end do
    $j = j - 1$;
end do
report success and return the truth value assigned to each literal;
end

Notice that the algorithm does not backtrack, i.e. if an empty clause is ever generated, the algorithm reports failure and terminates.

We were motivated to work with literals rather than variables from the approach in [16] (see also about the decoupling of a literal from its negation in Section 3). We were motivated to give priority to large degrees from [2, 5], where the need to capitalize on vertex-degree information was pointed out and from [4], where, in the context of the 3-coloring problem, the Brélaz heuristic [6] was analyzed. According to this heuristic, vertices of maximum degree are given priority, but only in case they can be legally colored by 2 out of the 3 colors. The maximum degree of such vertices is not necessarily decreasing, whereas in our case the degree of any literal is always decreasing, a fact that simplifies the analysis. We were motivated to put together all heavy literals from [18], where light vertices are deleted in order to find the k-core of a random graph.

To put in use the above ideas in the probabilistic analysis, we introduce in Section 3 a model for random formulas that we call the *truncated configuration model*. It is a variant of the configuration model for random graphs with a given degree sequence (see e.g. [14]). The name comes from a variant of the Poisson distribution we use, which is called the truncated Poisson distribution. A special case (for $h = 1$) of this distribution was used in [7], in the analysis of the pure literal heuristic. Section 4 outlines our basic tool: Wormald's method of analyzing random processes with differential equations. In Sections 2, 5 and 6 we elaborate on certain aspects particular to analyses of the type we make in this paper. The same or similar techniques as those in the above three sections have been used before (see, in this respect, [16, 1, 2, 5] and also, with respect to algorithms for the coloring problem, see [3, 4]). Although the basic ideas involved are essential to understand the method, the reader may gloss over the technicalities of these three sections. Section 8 contains the main result and Sections 7, 9 and Appendix 2 concern numerical computations and experiments. Finally, the derivation of the differential equations is given in Appendix 1, as their particulars do not add anything new to our approach.

2 Rounds

The algorithm proceeds in *rounds*. A round consists of one *free step*, i.e. a step where a literal in X_j is set to TRUE ($j = h, \ldots, 0$), followed by a number of *forced steps* i.e. steps where unit 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.

As in [5], 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 $c = 3.42$ and $h = 10$), the expected number of unit clauses generated at any atomic step is bounded below 1 (this, as we will show in Section 5, guarantees that the probability of success of the algorithm is bounded above 0). But then, during the course of 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 Section 5). 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 [20]) 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, Sections 4 and 6. The technique of rounds (introduced in [5]), i.e. the change of the time parameter to count the number of rounds, guarantees that Wormald's theorem on the study of the evolution of stochastic parameters by the use of differential equations is applicable.

3 The randomness model

For a 3-CNF formula with n variables, let its *truncated*, scaled (with respect to n) degree sequence be the sequence $x_j, j = 0, \ldots, h$, where for $0 \leq j < h$, x_j is the scaled number of literals of degree exactly j, whereas x_h is the scaled number of heavy literals, i.e., literals of degree at least h (a non-uniform but convenient notation again). Notice that the degree sequence refers not to variables but to literals. Also, let ℓ denote the scaled total number of literals ($\ell = 2$ in the beginning of the algorithm).

It is immediate that for a uniformly random formula with n variables and density c, each of the $3cn$ literal positions has probability to be occupied by any specified literal equal to $1/(2n)$. But then, for large n, the probability of a literal having degree j follows a Poisson distribution with parameter $3c/2$. So the expected value of the scaled number of literals of degree j is $2e^{-(3/2)c}((3/2)c)^j/j!$. From this and from a simplified version of Azuma's inequality (see for example the "Simple Concentration Bound" in [17]) it follows that almost all uniformly random formulas with n variables and density c have truncated scaled degree sequence given by $x_j = 2e^{-(3/2)c}((3/2)c)^j/j! + o(1)$, for $j = 0, \ldots h-1$ (then necessarily $x_h = 2 - \sum_{j=0}^{h-1} x_j + o(1)$). So to prove that a result holds a.s., in the large n limit, for uniformly random formulas with n variables and scaled number of clauses c, we only have to prove it for formulas that are additionally conditional on that their truncated scaled degree sequence is the one given above. In the sequel, we assume that the Davis-Putnam algorithm is initialized with such a formula.

Notice that although we start with a 3-CNF formula, at the end of each round we get a $\{3, 2\}$-CNF formula, i.e. a formula that may contain 3-clauses and 2-clauses (unit clauses appear only during the course of each round; empty clauses are the clauses we aim at showing that do not appear). We study the evolution of the following $h + 3$ scaled stochastic parameters during the course of the algorithm: ℓ, the scaled number of unset literals; c_3, the scaled number of 3-clauses; c_2, the scaled number of 2-clauses and x_j, $0 \leq j \leq h-1$. The scaling at all steps of the algorithm is with respect to n, the initial number of variables. However, we will also consider the current densities, of either 3-clauses or 2-clauses, i.e., the current number of 3-clauses or 2-clauses, respectively, divided by the *current* number of unset variables. We use the symbols ρ_2 and ρ_3, respectively, for the densities; notice that $\rho_i = 2c_i/\ell, i = 2, 3$. We denote by $\mathcal{S}$ the vector whose coordinates are the current unscaled (i.e. multiplied with n) values of the $h+3$ parameters above, i.e.

$$\mathcal{S} = n(\ell, c_3, c_2, x_0, \ldots, x_{h-1}). \tag{1}$$

As we have already indicated, initially, $\ell = 2$, $c_3 =$ scaled number c of 3-clauses of the initially given 3-CNF formula $c_2 = 0$ and $x_j = 2e^{-(3/2)c}((3/2)c)^j/j!$, $j = 0, \ldots h-1$.

It can be shown, by the technique in [15], that during the evolution of the algorithm, the formula remains random conditional on the current number of these parameters (it is essential for this result that all occurrences of a literal, irrespective of the clause size, count towards its degree).

The model we use for formulas that are random conditional on the values of the parameters in $\mathcal{S}$, which we call the *truncated configuration model*, is described as follows: think of the ℓn literals as bins and of the $(3c_3+2c_2)n$ clause positions as balls. Randomly select h subsets of the balls of cardinalities $jx_jn, j=0,\dots,h-1$, respectively. Then for each $j=0,\dots,h-1$ randomly throw the balls of the corresponding subset into the x_jn bins of literals of degree j, in a away that every bin gets exactly j balls. Then randomly throw the remaining $(\ell-\sum_{j=0}^{h-1} jx_j)n$ balls into the bins of heavy literals, but so that each heavy bin gets at least h balls. Although this model allows the same literal, or literals of the same variable, to occur in a clause, such phenomena violating the simplicity of the formula are, for the large n limit, of bounded away from zero probability (the term "simplicity" comes from simple graphs, i.e. graphs without loops or multiple edges). Therefore, if a result, like our main theorem in Section 8, is shown to hold a.s. for the possibly non-simple formulas in this model, it also holds a.s. for random simple formulas. In the sequel, we use the above model in all probabilistic considerations.

An important aspect of the algorithm 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 it is usually the case, with a 2 dimensional degree sequence that at (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.

Let now p_h denote the scaled number of literal positions that are occupied with heavy literals. Then $p_h=3c_3+2c_2-\sum_{j=0}^{h-1} jx_j$. Then the probability mass of the number of literals of degree i, for any fixed $i\geq h$, follows a truncated Poisson distribution. This means that for any $i\geq h$, the probability that a heavy literal has degree i is:

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

where μ is the solution, with respect to x, of the following equation:

$$\frac{p_h}{x_h}=\frac{x\left(e^x-\sum_{j=0}^{h-2}\frac{x^j}{j!}\right)}{e^x-\sum_{j=0}^{h-1}\frac{x^j}{j!}}. \tag{3}$$

This result about the truncated Poisson distribution is proved for $h=1$ and for 3-CNF formulas in [7]. Its generalization to arbitrary h and for $\{3,2\}$-CNF formulas in the truncated configuration model is not hard. We omit the proof.

4 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 $Y_j, j=1,\dots,a$ are stochastic parameters related to a formula, like e.g. the number of clauses with a specified size, or the number of literals with a specified degree. In our case, the Y_js 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 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. Notice that, conveniently, a Davis-Putnam algorithm is a Markov process, although Wormald's theory is applicable to more general processes.

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

$$\mathbf{E}[Y_j(t+1) - Y_j(t) \mid Y_1(t), \ldots, Y_a(t)] = f_j(t/n, Y_1(t)/n, \ldots, Y_a(t)/n) + o(1), j = 1, \ldots, a, \quad (4)$$

where the f_js are Lipschitz continuous functions. Suppose also that the change of each of the parameters Y_j from t to $t+1$ is well concentrated on its expected value (see [20] for the exact statement). Then the solution of the system of differential equations:

$$dy_j(x)/dx = f_j(x, y_1(x), \ldots, y_a(x)), j = 1, \ldots, a, \quad (5)$$

with the initial conditions $y_j(0) = Y_j(0)/n$, gives, at any time t and a.s. (in the large n limit), the scaled value of Y_j, i.e.

$$y_j(t/n) = (1/n)Y_j(t) + o(1), j = 1, \ldots, a. \quad (6)$$

In applications, an open, connected and bounded domain D that contains $(Y_1(0)/n, \ldots, Y_a(0)/n)$ and a time interval $[0, t_f)$ are 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_js 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.

5 Inside a round and probability of success

Assume, for the moment, that the density of 2-clauses remains constant during a round (we will elaborate on this point below). Then the generation of the unit clauses in the the forced steps of the round follows the pattern of a Galton-Watson branching process (see [11]). 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 (in our case the branching process starts not with one root, but with the number of unit clauses generated at the first free step of the round). The number of offspring in a Galton-Watson tree may follow an arbitrary distribution whose expected value is known as the Malthus parameter μ. It is known that 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 $1 + \mu + \mu^2 + \ldots = 1/(1 - \mu)$.

In our case now, as we have shown in Section 3 when referring to the decoupling of a literal from its negation, the Malthus parameter is equal to the density ρ_2 of 2-clauses. Suppose that for a given initial scaled number of clauses c and predetermined h, ρ_2 remains bounded below 1 a.s. during the course of the algorithm. Then, the total number of unit clauses generated during the round is, with probability 1, finite. Moreover, their expected number is $m_1 = \rho_2/(1 - \rho_2)$ (recall that we start not with one, but with expectedly ρ_2 unit clauses generated at the free step). In [11, Theorem 5.8], it is proved that $y(s) = sf(y(s))$, where $y(s)$ is the probability generating function of the size of the tree and $f(s)$ is the probability generating function of the number of offspring. From this, by differentiation, we can easily compute the second moment m_2 of the size of a subcritical Galton-Watson tree in terms of the Malthus parameter and the second moment μ_2 of the number of offspring. In our case, μ_2 is equal to the second moment of the number of occurrences in 2-clauses of a random literal. In the proof of Claim (12) in Appendix 1, this moment is computed. All we need here is that μ_2 is $O(1)$. Therefore, m_2 as well is $O(1)$, a fact we use below.

Condition now on the actual size $|T|$ of the Galton-Watson tree at a round. In order to derive the differential equations of Wormald's theorem, we need to compute the total expected change during all steps of a round of each parameter in $\mathcal{S}$. First notice for any parameter Y_j as in Wormald's theorem, if its expected change from one atomic step to the next obeys a change law like in Relation (4), then its expected change in $|T|$ consecutive steps, by the smoothness of f_j, is given by

$$\mathbf{E}[Y_j(t + |T|) - Y_j(t) \mid Y_1(t), \ldots, Y_a(t)] = f_j(t/n, Y_1(t)/n, \ldots, Y_a(t)/n)|T| + |T|^2 o(1). \quad (7)$$

Taking now the expectation of the above relation with respect to $|T|$ we get:

Claim 1. *The expected change of a parameter in $\mathcal{S}$ during a round, unconditionally on the number of unit clauses generated at the round, is equal to its expected change at the free step plus its expected change at the first forced step multiplied with $\rho_2/(1 - \rho_2)$. Moreover, to compute the expected change at the first forced step, we may assume, by continuity, that the value of the scaled parameters in $\mathcal{S}$ are the same at the beginning of the free step and at the beginning of the forced steps. The total error term is $m_2 o(1)$, i.e. $o(1)$, as required by Wormald's theorem.*

We strongly use the above claim to derive the differential equations.

By the same technique of conditioning on the size of the tree and then un-conditioning with passing to the moments, we can show that the change of a parameter from round to round is concentrated around its mean, as sharply as required by Wormald's theorem.

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 $O(m_2/n)$ (when the tree is subcritical). Therefore, for all rounds, the probability that no contradiction occurs is $(1 - O(m_2/n))^n = e^{-O(m_2)}$, so the probability of success of the algorithm, as long as the generation of unit clauses is subcritical, is bounded above zero.

It may seem that our arguments above are circular, as we use the sub-criticality of the unit-clauses to derive the differential equations, and then from them to prove this sub-criticality. The way out is to apply Wormald's theorem for a domain D which includes only values for which ρ_2

is bounded below 1. Then the conclusion of the theorem remains true as long as the solutions of the differential equations stay in D. Notice that the rôle of D is to force the parameters in $\mathcal{S}$ to a.s. remain inside a desirable path.

6 Transition between phases and termination of the algorithm

The jth ($j = h, \ldots, 0$) *phase* of the algorithm is comprised by all rounds whose free steps set to True a literal of degree j. In other words, a phase corresponds to the middle loop of the pseudo-code of the algorithm. Fix now a phase j. Can the domain of application of Wormald's theorem $[0, t_f) \times D$ be extended to include the vector $(T_0/n, \mathcal{S}(T_0)/n)$, where T_0 is the instant where the phase j terminates and $\mathcal{S}(T_0)/n$ is the vector of the scaled values of the parameters in $\mathcal{S}$ at T_0? The answer is no, because the expected change of $|\mathcal{X}_j|$ before T_0 (i.e. when $\mathcal{X}_j \neq \emptyset$) and the expected change of $|\mathcal{X}_j|$ at T_0 (i.e. when $\mathcal{X}_j = \emptyset$) cannot both be expressed by the same Lipschitz continuous function, as this expected change changes discontinuously when we pass from values before T_0 to T_0 (before T_0 we always set to True a literal from $\mathcal{X}_j$, whereas we cease doing so at T_0).

This creates a problem with respect to the computation of the scaled value of parameters in $\mathcal{S}$ at the very end of the phase j. We can overcome this difficulty by continuity arguments, i.e. by finding the limiting values of these parameters as we approach the end of the phase (see in this respect [20, Note 5, p. 1220]). However we prefer an indirect, but easy to explain, alternative solution, which moreover is better suited to the numerical calculations: we assume that the transition from phase j to $j-1$ occurs when the scaled parameter x_j becomes not greater than ϵ, where ϵ is a predetermined small number. In our numerical calculations, we take $\epsilon = (1/2)10^{-5}$. Then of course at each subsequent phase, the differential equations have to take into account the existence of leftover literals from previous phases. However, the degree of any literal is always monotonically decreasing, and therefore the total number of leftover literals is decreasing. So these leftover literals, although taken into account in the differential equations, do not substantially affect the results of the numerical calculations. On the other hand the cardinalities, for each degree, of the leftover literals remain $\Omega(n)$, so the domain restrictions of Wormald's theorem are satisfied. Of course, taking $\epsilon < (1/2)10^{-5}$ would improve the lower bound of the threshold, but it turns out that this improvement is negligible with respect to the two decimal digits of accuracy that we keep for the lower bound.

A similar as above problem arises with the termination of the algorithm, since again we cannot extend the differential equations until the scaled number of literals becomes zero (here not even continuity arguments work, because even if we find the limiting value of the parameters, the unscaled value of them is given only within an error $o(n)$, and so even at the boundary of the extension of the differential equations, we may have unset literals). The solution we give is the following: we carry on the algorithm until the size of the formula is small enough, but remains $\Omega(n)$, and then we apply a criterion for satisfiability of formulas with a given degree sequence given by Cooper et al. [10, Theorem 1(a)]. By simple calculations, it follows from this result in [10] that a random $\{3, 2\}$-CNF formula with a given degree sequence is a.s. satisfiable if the sum of the densities of the 3-clauses and 2-clauses $\rho_3 + \rho_2$ is bounded below 1. Notice that the result in [10] refers to 2-CNF formulas; to apply it to $\{3, 2\}$-CNF formulas, we randomly delete a literal from each 3-clause. Therefore, once we reach a point where $\rho_3 + \rho_2 < 1$ we stop the algorithm and refer to the criterion in [10]. We mention again that similar tricks have been used in most previous probabilistic analyses of Davis-Putnam procedures.

Finally, let us point out that the technique of rounds was introduced for similar as above reasons having to do with the fact that the rate of change of unit clauses cannot be expressed by a Lipschitz continuous function, because the class of unit clauses during the algorithm becomes empty unboundedly many times (at the end of each round).

7 Numerical integration

A word about the numerical integration of the system of differential equations: as the system of equations has the form $dy_j(x)/dx = f_j(x, y_1(x), \ldots, y_a(x)), j = 1, \ldots, a$, to numerically integrate it, we divide the interval where x (the scaled time) ranges into equal subintervals of small length Δx (in our case, we have taken $\Delta x = (1/2)10^{-5}$). We then recursively estimate the value of the functions $y_j(x)$ at the endpoints of these intervals by setting $y_j(0) = Y_j(0)/n$ and

$$y_j((s+1)\Delta x) = f_j(s\Delta x, y_1(s\Delta x), \ldots, y_a(s\Delta x))\Delta x + y_j(s\Delta x), s \geq 0.$$

Thus, in the large n limit, because of the continuity of the functions f_j, we get an estimation of the functions $y_j(x)$ at any desired degree of accuracy. This recursive integration of the system of the differential equations is much more easily handled by a numerical computation program like Maple. We take $\Delta x = (1/2)10^{-5}$. To avoid the accumulation of errors, we work, on Maple 7, with 40 digits of accuracy.

8 Putting everything together

Working with the randomness model described earlier, and using the results about the truncated Poisson distribution and Claim 1, the expected changes, from a round to the next, of the $h+3$ (13 in our case, as we take $h = 10$) parameters in $\mathcal{S}$ can be computed by elementary probabilistic calculations. From the expressions that give these expectations, we derive the system of differential equations. As these equations do not add anything to the understanding of the techniques, they are given, with short explanations, in Appendix 1; however see next section for experimental verification of the results. It turns out that for initial density less than 3.42, ρ_2 stays below 1 all the way until $\rho_3 + \rho_2$ become strictly less than 1. At this stage we apply the criterion in [10] to conclude that the remaining formula is satisfiable (see Section 5). Therefore, we proved that for $c < 3.42$, the algorithm succeeds in producing a satisfying truth assignment, with bounded above 0 probability. By a standard argument (see e.g. [2, Corollary 2]), which easily follows from the result of Friedgut that the satisfiability of k-CNF formulas has a sharp threshold [12], we get that:

Theorem. *Random 3-CNF formulas with scaled number of clauses less than 3.42 (the scaling is with respect to the number n of variables) are a.s., in the large n limit, satisfiable.*

9 Experiments

We compared the results we obtained from the numerical integration of differential equations with the results of a simulation of the algorithm.

As we already mentioned, the differential equations were worked out on Maple 7 using the recursive technique outlined at Section 7, with $\Delta x = (1/2)10^{-5}$. We used precision of 40 digits.

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 [19]. Our implementation was influenced and makes use of the code for the implementation of GSAT, also available in 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 from Table 1 given in Appendix 2. Finally, preliminary experiments we conducted suggest that by greedy algorithms that take into account not only the number of clauses satisfied by the selection of a literal but also the number of clauses that are shrunk (but ignore the size of the clause where this literal appears), the value 3.42 can be raised significantly, up to 3.6 and perhaps more. So perhaps formulas with density below the experimentally computed value 4.25 (or below a number close to it) not only are they satisfiable, but a satisfying truth assignment can be found by simple greedy heuristics. This of course does not provide a good moral, as far as satisfaction is concerned, since it is the clause-size dependent heuristics, and not the greedy ones, that care to take into account dangerous, i.e. small-size, clauses.

Acknowledgment

Discussions of the second author with D. Achlioptas were crucial in developing the ideas in 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, 2000.

[2] D. Achlioptas, "Lower bounds for random 3-SAT via differential equations," *Theoretical Computer Science* 265 (1-2), pp. 159–185, 2001.

[3] D. Achlioptas and 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, 1997.

[4] D. Achlioptas and C. Moore, "Almost all graphs with average degree 4 are 3-colorable," presented at the Workshop on Computational Complexity and Statistical Physics, Santa Fe, N.M. USA, 2001.

[5] 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, 2000.

[6] D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM* 22, pp. 251–256, 1979.

[7] 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, 1993.

[8] M. T. Chao and J. Franco, "Probabilistic analysis of two heuristics for the 3-satisfiability problem," *SIAM Journal of Computing* 15(4), pp. 1106–1118, 1986.

[9] 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, 1992.

[10] 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, 2002.

[11] L. Devroye, "Branching processes and their applications in the analysis of tree structures and tree algorithms," in: *Probabilistic Methods for Algorithmic Discrete Mathematics,* M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed (eds.), pp. 249–314, Springer, 1998.

[12] E. Friedgut, appendix by J. Bourgain, "Sharp thresholds of graph properties, and the k-SAT problem," *Journal of the AMS* 12, pp. 1017–1054, 1999.

[13] A. Frieze and S. Suen, "Analysis of two simple heuristics for random instances of k-SAT," *Journal of Algorithms* 20, pp. 312–355, 1996.

[14] S. Janson, T. Łuczak, and A. Ruciński, *Random Graphs,* Wiley, 2000.

[15] A.C. Kaporis, L.M. Kirousis, and Y.C. Stamatiou, "How to prove conditional randomness using the Principle of Deferred Decisions," Technical Report, Computer Technology Institute, Greece, 2002. Available at: www.ceid.upatras.gr/faculty/kirousis/kks-pdd02.ps

[16] M. Mitzenmacher, "Tight thresholds for the pure literal rule," Technical Report, Digital Equipment Corporation, 1997. Available at: www.research.compaq.com/SRC/

[17] M. Molloy, "The probabilistic method," in: *Probabilistic Methods for Algorithmic Discrete Mathematics,* M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed (eds.), pp. 1–35, Springer, 1998.

[18] B. Pittel, J. Spencer, and N.C. Wormald, "Sudden emergence of a giant k-core in a random graph," *J. Combinatorial Theory, Series B* 67, pp. 111–151, 1996.

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

[20] N.C. Wormald, "Differential equations for random processes and random graphs," *The annals of applied probability* 5(4), pp. 1217–1235, 1995.

Appendix 1

Fix a phase $j, j = h, \ldots, 0$ of the algorithm. We give here the differential equations for the scaled parameters:

$$(1/n)\mathcal{S} = (\ell, c_3, c_2, x_0, \ldots, x_{h-1}).$$

Recall, that ℓ is the scaled number of the currently unset literals, c_3 and c_2 are the current scaled numbers of 3-clauses and 2-clauses, respectively and for $i = 0, \ldots, h-1$, x_i is the current scaled number of literals of degree i. Also recall that by x_h we denote the current scaled number of literals of degree at least h. The above parameters and all the additional ones we will consider are functions of time t (number of rounds), but to avoid overloading the notation, we omit indicating the dependence on t. The formula we deal with is a random $\{3,2\}$-CNF formula conditional on the parameters in $\mathcal{S}$. We have to introduce some additional notation: let $p = 3c_3 + 2c_2$ be the total scaled number of literal positions in the formula. Let $p_h = p - \sum_{s=0}^{h-1} s x_s$ be the scaled number of occurrences of all heavy literals. Then, obviously, p_h/x_h is the expected degree of a random heavy literal. Let d_{fr}^j be the expected degree of the literal selected at a free step of phase j. By the above:

$$d_{fr}^j = \begin{cases} p_h/x_h & \text{if } j = h, \\ j & \text{if } 0 \le j \le h-1. \end{cases} \tag{8}$$

Observe that then the expected number of occurrences in 2-clauses of a literal selected at a free step of phase j is $2(c_2/p)d_{fr}^j$. Let μ be the solution, with respect to x, of the equation:

$$\frac{p_h}{x_h} = \frac{x\left(e^x - \sum_{s=0}^{h-2} \frac{x^s}{s!}\right)}{e^x - \sum_{s=0}^{h-1} \frac{x^s}{s!}}. \tag{9}$$

Let also

$$\lambda = e^{\mu} - \sum_{s=0}^{h-1} \frac{\mu^s}{s!}. \tag{10}$$

Then by Equation (2), in Section 3, we have that for any $i \ge h$, the probability of a heavy literal having degree i is $\mu^i/(\lambda i!)$. Therefore the scaled expected number of heavy literals of degree i (for $i \ge h$) is $(x_h \mu^i)/(\lambda i!)$. In particular, the scaled expected number of heavy literals of degree exactly h, which we denote by y_h, is given by:

$$y_h = \frac{x_h \mu^h}{\lambda h!}. \tag{11}$$

Finally, let d_{fo} be the degree of the literal of the unit clause satisfied at a forced step, decremented by one. The decrementing is made so that d_{fo} gives the expected number of clauses satisfied without counting the unit clause itself. Then the expected number of 3-clauses satisfied when setting such a literal is $3(c_3/p)d_{fo}$, while that of 2-clauses is $2(c_2/p)d_{fo}$.

We now claim that

Claim 2.

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

Proof For any $s \geq 0$, let y_s denote the expected number of literals of degree exactly s. We have seen that for $s \geq h$, $y_s = (x_h \mu^s)/(\lambda s!)$, while obviously, for $s = 0, \dots h-1$, $y_s = x_s$. It is now convenient to switch to the balls-to-bins parlance of the truncated configuration model (see Section 3). Under this view, the probability of the literal in a unit clause having degree s (the occurrence in the unit clause counted) is the probability of a random ball getting into a bin with totally s bins. This probability is easily sy_s/p. Therefore the expected number of balls in the bin where a random ball is thrown is

$$\frac{\sum_{s\geq 0} s^2 y_s}{p} = \frac{\sum_{s=0}^{h-1} s^2 y_s}{p} + \frac{x_h}{\lambda p} \sum_{s\geq h} \frac{s^2 \mu^s}{s!}.$$

Now easily, $\sum_{s\geq h} \left(s^2\mu^s/s!\right) = \mu^2 e^\mu + \mu e^\mu - \sum_{s=0}^{h-1} \left(s^2 \mu^s/s!\right)$, and the required follows. Notice, in addition, that as is well known, in any balls-to bins experiment, irrespective of the distribution of balls into bins, the second moment of the number of balls in a random bin is equal to the expected number of balls in the bin of a random ball multiplied by the expectation of the number of balls in a random bin. So we have also proved that the second moment of the occurrences in 2-clauses of a random literal is $(2(c_2/p)d_{fo} + 1)\rho_2$, therefore the second moment of the number of the offspring of the Galton-Watson tree that corresponds to the generation of unit clauses is $O(1)$ (see the argument about the moments of the size of this tree in Section 5). $\square$

We now give the differential equations that govern the evolution of the parameters in $(1/n)\mathcal{S}$. These equations are derived by computing the expected change, from a round to the next, of the unscaled versions of the parameters. The computations are carried out in the truncated configuration model, by *elementary* probabilistic arguments, using as a basic tool Claim 1. We do not give proofs. Notice however, that the remarkable affinity of the results obtained from the equations with the simulation results given in Table 1 in Appendix 2 testify to the correctness of the equations. In the rhs of the equations below, the term with the factor $\rho_2/(1-\rho_2)$ gives the change during the forced steps, whereas the other terms give the change at the free step.

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

$$(b) \qquad \frac{dc_3}{dt} = -3\left(\frac{d_{fr}^j c_3}{p} + \frac{\rho_3}{2}\right) - 3\left(\frac{d_{fo} c_3}{p} + \frac{\rho_3}{2}\right)\frac{\rho_2}{1-\rho_2},$$

$$(c) \qquad \frac{dc_2}{dt} = \frac{3\rho_3}{2} - \frac{2d_{fr}^j c_2}{p} - \rho_2 + \left(\frac{3\rho_3}{2} - \frac{2d_{fo}c_2}{p} - \rho_2\right)\frac{\rho_2}{1-\rho_2},$$

$$(d) \qquad \frac{dx_s}{dt} = (6c_3 + 2c_2)\tfrac{(s+1)x_{s+1} - sx_s}{p^2} d_{fr}^j - \tfrac{x_s}{\ell} - \delta_{s,j}$$

$$+ \left((6c_3 + 2c_2)\tfrac{(s+1)x_{s+1} - sx_s}{p^2} d_{fo} - \tfrac{x_s}{\ell} - \tfrac{sx_s}{p}\right)\tfrac{\rho_2}{1-\rho_2},$$

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

$$(e) \qquad \frac{dx_{h-1}}{dt} = (6c_3 + 2c_2)\tfrac{hy_h - (h-1)x_{h-1}}{p^2} d_{fr}^j - \tfrac{x_{h-1}}{\ell} - \delta_{h-1,j}$$

$$+ \left((6c_3 + 2c_2)\tfrac{(hy_h - (h-1)x_{h-1}}{p^2} d_{fo} - \tfrac{x_{h-1}}{\ell} - \tfrac{(h-1)x_{h-1}}{p}\right)\tfrac{\rho_2}{1-\rho_2},$$

where

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

Initially:

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

Appendix 2

In Table 1, the results obtained from the numerical computations are compared with the results of the simulation of the algorithm. Each row consists of two sub-rows: the sub-row indicated by "d.e." gives values obtained from the numerical solution of the system of differential equations on Maple 7, whereas the sub-row indicated by "sim." gives values obtained from the simulation of the algorithm with the C code. We opted to include ρ_2 and ρ_3 in the table instead of c_2 and c_3 (recall that $\rho_i = 2c_i/\ell, i = 2, 3$), because keeping ρ_2 bounded below 1 guarantees the sub-criticality of unit-clauses and hence a bounded above 0 probability of success; also when $\rho_2 + \rho_3$ becomes less than 1, we stop the algorithm and make use of the satisfiability criterion of Cooper et al. [10]. We also included $x_{10} = \ell - \sum_{j=0}^{9} x_j$. The scaled time instants in the table were chosen among the 2×10^5 values that our increment choice of $(1/2)10^{-5}$ in Maple 7 gives so that crucial instances of the algorithm (like the maximization of ρ_2) are depicted. All the zero values in the sub-rows d.e. are approximations of positive, but smaller than 10^{-6}, values.

	t	ℓ	ρ_2	ρ_3	x_0	x_1	x_2	...	x_8	x_9	x_{10}
d.e.	.000000	2.000000	.000000	3.420000	.011833	.060703	.155705	...	.140772	.080240	.073582
sim.	.000000	2.000000	.000000	3.420000	.012078	.060154	.155992	...	.141078	.079986	.073526
d.e.	.010000	1.979478	.051372	3.294281	.013127	.065913	.165471	...	.131459	.072808	.055457
sim.	.010000	1.979436	.051455	3.294243	.013392	.065444	.165650	...	.131904	.072540	.055550
d.e.	.020000	1.957801	.102948	3.165518	.014615	.071725	.176000	...	.121703	.064449	.039038
sim.	.020000	1.957622	.102767	3.165561	.014962	.071308	.175780	...	.122222	.064282	.039098
d.e.	.030000	1.934832	.154631	3.033612	.016330	.078224	.187345	...	.111393	.055419	.024384
sim.	.030000	1.934700	.154814	3.033580	.016772	.077558	.187116	...	.111658	.055340	.024436
d.e.	.040000	1.910425	.206383	2.898391	.018320	.085510	.199558	...	.100470	.045974	.011536
sim.	.040000	1.910484	.206123	2.898846	.018770	.084898	.199070	...	.100874	.045974	.011542
d.e.	.050000	1.884366	.258116	2.759672	.020639	.093699	.212686	...	.088941	.036379	.000520
sim.	.050000	1.884208	.258151	2.759493	.021228	.092942	.212402	...	.089040	.036536	.000508
d.e.	.050510	1.882988	.260752	2.752500	.020767	.094143	.213381	...	.088338	.035890	.000017
sim.	.050510	1.882800	.260922	2.752160	.021382	.093422	.213088	...	.088480	.036042	.000000
d.e.	.060000	1.856401	.310999	2.632164	.022984	.101647	.224767	...	.077376	.020061	.000015
sim.	.060000	1.856256	.311209	2.631981	.023624	.100960	.224676	...	.077812	.020238	.000000
d.e.	.070000	1.826188	.364454	2.500700	.025703	.110515	.237584	...	.064064	.006658	.000012
sim.	.070000	1.826040	.364570	2.500637	.026374	.109922	.237944	...	.064476	.006812	.000000
d.e.	.076105	1.806463	.397338	2.417882	.027596	.116482	.245831	...	.055433	.000008	.000011
sim.	.076105	1.806744	.396826	2.418985	.028210	.115870	.246294	...	.055834	.000120	.000000
d.e.	.080000	1.793301	.418885	2.367798	.028785	.120138	.250699	...	.046740	.000008	.000010
sim.	.080000	1.793068	.419080	2.367642	.029412	.119574	.251230	...	.047058	.000000	.000000
d.e.	.090000	1.757114	.474996	2.234567	.032230	.130388	.263740	...	.027271	.000007	.000008
sim.	.090000	1.756488	.475355	2.233755	.032886	.130000	.263852	...	.027436	.000000	.000000
d.e.	.104000	1.669236	.555691	2.034841	.038258	.147221	.283259	...	.006595	.000005	.000005
sim.	.104000	1.699116	.556006	2.035145	.038954	.146628	.283420	...	.006798	.000000	.000000
d.e.	.109955	1.671313	.590879	1.944219	.041385	.155450	.291943	...	.000005	.000005	.000004
sim.	.109955	1.671960	.590224	1.946347	.042202	.154384	.292072	...	.000180	.000000	.000000
d.e.	.120000	1.617999	.654258	1.790959	.047159	.169702	.305322	...	.000005	.000004	.000003
sim.	.120000	1.617592	.654565	1.790711	.047866	.169326	.305842	...	.000000	.000000	.000000
d.e.	.130000	1.553796	.721099	1.620394	.054604	.186559	.318659	...	.000004	.000003	.000002
sim.	.130000	1.554480	.720315	1.622145	.055456	.186224	.318190	...	.000000	.000000	.000000
d.e.	.140000	1.470975	.793518	1.421630	.064857	.207177	.330796	...	.000003	.000002	.000001
sim.	.140000	1.475280	.789439	1.430790	.065046	.206406	.329776	...	.000000	.000000	.000000
d.e.	.150000	1.346948	.880189	1.170968	.080346	.232620	.336419	...	.000002	.000001	.000000
sim.	.150000	1.346224	.880747	1.169965	.080994	.232604	.335538	...	.000000	.000000	.000000
d.e.	.155000	1.235824	.936604	.979773	.094423	.249925	.330070	...	.000001	.000000	.000000
sim.	.155000	1.234012	.937041	.978266	.095532	.249156	.329176	...	.000000	.000000	.000000
d.e.	.158375	.970051	.994392	.613143	.127607	.266841	.276895	...	.000000	.000000	.000000
sim.	.158375	1.081412	.982441	.753100	.114804	.263960	.304444	...	.000000	.000000	.000000
d.e.	.160000	.786125	.956421	.408063	.150848	.256640	.215237	...	.000000	.000000	.000000
sim.	.160000	.777636	.953371	.401699	.151582	.255130	.212056	...	.000000	.000000	.000000
d.e.	.170000	.600820	.803173	.222643	.179484	.225146	.137641	...	.000000	.000000	.000000
sim.	.170000	.595608	.797578	.219674	.179668	.223954	.135280	...	.000000	.000000	.000000
d.e.	.171605	.585324	.782149	.207836	.182467	.221233	.130492	...	.000000	.000000	.000000
sim.	.171605	.579056	.774584	.203503	.183088	.219382	.127930	...	.000000	.000000	.000000

Table 1: Comparison of values obtained from the numerical solution of the differential equations with the values given by the simulation.