Eur. Phys. J. B **22**, 505–531 (2001)

THE EUROPEAN PHYSICAL JOURNAL B

EDP Sciences

Statistical physics analysis of the computational complexity of solving random satisfiability problems using backtrack algorithms

S. Cocco[1,2] and R. Monasson[1,3,a]

[1] CNRS-Laboratoire de Physique Théorique de l'ENS, 24 rue Lhomond, 75005 Paris, France
[2] Department of Physics, The University of Illinois at Chicago, 845 W. Taylor St., Chicago IL 60607, USA
[3] The James Franck Institute, The University of Chicago, 5640 S. Ellis Av., Chicago IL 60637, USA

Received 10 March 2001

Abstract. The computational complexity of solving random 3-Satisfiability (3-SAT) problems is investigated using statistical physics concepts and techniques related to phase transitions, growth processes and (real–space) renormalization flows. 3-SAT is a representative example of hard computational tasks; it consists in knowing whether a set of αN randomly drawn logical constraints involving N Boolean variables can be satisfied altogether or not. Widely used solving procedures, as the Davis-Putnam-Loveland-Logemann (DPLL) algorithm, perform a systematic search for a solution, through a sequence of trials and errors represented by a search tree. The size of the search tree accounts for the computational complexity, *i.e.* the amount of computational efforts, required to achieve resolution. In the present study, we identify, using theory and numerical experiments, easy (size of the search tree scaling polynomially with N) and hard (exponential scaling) regimes as a function of the ratio α of constraints per variable. The typical complexity is explicitly calculated in the different regimes, in very good agreement with numerical simulations. Our theoretical approach is based on the analysis of the growth of the branches in the search tree under the operation of DPLL. On each branch, the initial 3-SAT problem is dynamically turned into a more generic 2+p-SAT problem, where p and $1-p$ are the fractions of constraints involving three and two variables respectively. The growth of each branch is monitored by the dynamical evolution of α and p and is represented by a trajectory in the static phase diagram of the random 2+p-SAT problem. Depending on whether or not the trajectories cross the boundary between satisfiable and unsatisfiable phases, single branches or full trees are generated by DPLL, resulting in easy or hard resolutions. Our picture for the origin of complexity can be applied to other computational problems solved by branch and bound algorithms.

PACS. 05.10.-a Computational methods in statistical physics and nonlinear dynamics – 05.70.-a Thermodynamics – 89.20.Ff Computer science and technology

1 Introduction

Out-of-equilibrium dynamical properties of physical systems form the subject of intense studies in modern statistical physics [1]. Over the past decades, much progress has been made in fields as various as glassy dynamics, growth processes, persistence phenomena, vortex depinning..., where dynamical aspects play a central role. Among all the questions related to these issues, the existence and characterization of stationary states reached in some asymptotic limit of large times is of central importance. In turn, the notion of asymptotic regime raises the question of relaxation, or transient, behavior: what time do we need to wait for in order to let the system relax? How does this time grow with the size of the system? Such interrogations are not limited to out-of-equilibrium dynamics but also arise in the study of critical slowing down phenomena accompanying second order phase transitions.

Computer science is another scientific discipline where dynamical issues are of central importance. There, the main question is to know the time or, more precisely, the amount of computational resources required to solve some given computational problem, and how this time increases with the size of the problem to be solved [2]. Consider for instance the sorting problem [3]. One is given a list $\mathcal{L}$ of N integer numbers to be sorted in increasing order. What is the computational complexity of this task, that is, the minimal number of operations (essentially comparisons) necessary to sort any list $\mathcal{L}$ of length N? Knuth answered this question in the early seventies: complexity scales at least as $N \log N$ and there exists a sorting algorithm, called Mergesort, achieving this lower bound [3].

To calculate computational complexity, one has to study how the configuration of data representing the

[a] e-mail: monasson@physique.ens.fr

computational problem dynamically evolves under the prescriptions encoded in the algorithm. Let us consider the sorting problem again and think of the initial list $\mathcal{L}$ as a (random) permutation of $\mathcal{I} = \{1, 2, \ldots, N\}$. Starting from $\mathcal{L}(0) = \mathcal{L}$, at each time step (operation) T, the sorting algorithm transforms the list $\mathcal{L}(T)$ into another list $\mathcal{L}(T+1)$, reaching finally the identity permutation, *i.e.* the ordered list $\mathcal{I}$. Obviously, the dynamical rules imposed by a solving algorithm are of somewhat unusual nature from a physicist's point of view. They might be highly non-local and non-Markovian. Yet, the operation of algorithms gives rise to well posed dynamical problems, to which methods and techniques of statistical physics may be applied as we argue in this paper.

Unfortunately, not all problems encountered in computer science are as simple as sorting. Many computational problems originating from industrial applications, *e.g.* scheduling, planning and more generally optimization tasks, demand computing efforts growing enormously with their size N. For such problems, called NP–complete, all known solving algorithms have execution times increasing *a priori* exponentially with N and it is a fundamental conjecture of computer science that no polynomial solving procedure exists. To be more concrete, let us focus on the 3-satisfiability (3-SAT) problem, a paradigm of the class of NP–complete computational problems [2]. A pedagogical introduction to the 3-SAT problem and some of the current open issues in theoretical computer science may be found in [4].

3-SAT is defined as follows. Consider a set of N Boolean variables and a set of $M = \alpha N$ constraints (called clauses), each of which being the logical OR of three variables or of their negation. Then, try to figure out whether there exists or not an assignment of variables satisfying all clauses. If such a solution exists, the set of clauses (called instance of the 3-SAT problem) is said satisfiable (sat); otherwise the instance is unsatisfiable (unsat). To solve a 3-SAT instance, *i.e.* to know whether it is sat or unsat, one usually resorts to search algorithms, as the ubiquitous Davis–Putnam–Loveland–Logemann (DPLL) procedure [4–6]. DPLL operates by trials and errors, the sequence of which can be graphically represented as a search tree. Computational complexity is the amount of operations performed by the solving algorithm and is conventionally measured by the size of the search tree.

Complexity may, in practice, vary enormously with the instance of the 3-SAT problem under consideration. To understand why instances are easy or hard to solve, computer scientists have focused on model classes of 3-SAT instances. Probabilistic models, that define distributions of random instances controlled by few parameters, are particularly useful. An example, that has attracted a lot of attention over the past years, is random 3-SAT: all clauses are drawn randomly and each variable negated or left unchanged with equal probabilities. Experiments [6–10] and theory [11,12] indicate that instances are almost surely always sat (respectively unsat) if α is smaller (resp. larger) than a critical threshold $\alpha_C \simeq 4.3$ as soon as M, N go to infinity at fixed ratio α. This phase transition [12,13] is accompanied by a drastic peak of computational hardness at threshold [6–8], see Figure 1. Random 3-SAT generates simplified and idealized versions of real-world instances. Yet, it reproduces essential features (sat *vs.* unsat, easy *vs.* hard) and can shed light on the onset of complexity, in the same way as models of condensed matter physics help to understand global properties of real materials.

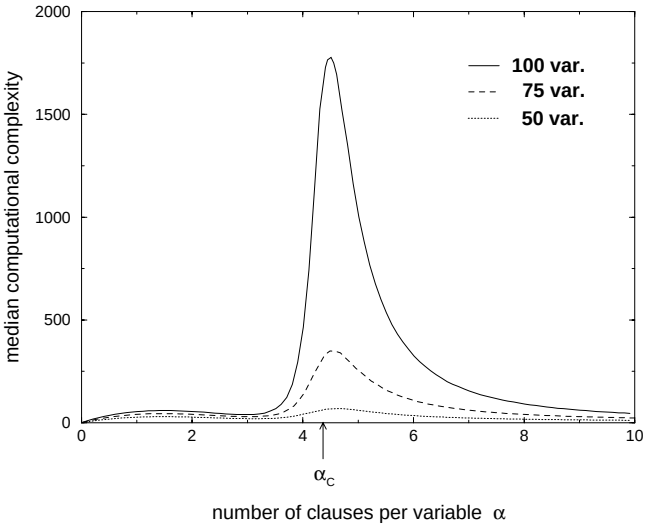

Fig. 1. Solving complexity of random 3-SAT as a function of the ratio α of clauses per variables, and for three increasing problem sizes N. Data are averaged over 10,000 randomly drawn samples. Complexity is maximal at the threshold $\alpha_C \simeq 4.3$.

Phases in random 3-SAT, or in physical systems, characterize the overall *static* behavior of a sample in the large size limit – a large instance with ratio *e.g.* $\alpha = 3$ will be almost surely sat (existence proof) – but do not convey direct information of *dynamical* aspects – how long it will take to actually find a solution (constructive proof). This situation is reminiscent of the learning problem in neural networks ("equilibrium" statistical mechanics allows to compute the maximal storage capacity, irrespective of the memorization procedure and of the learning time) [14], or liquids at low enough temperatures (that should crystallize from a thermodynamical point of view but undergo some kinetical glassy arrest) [15][1].

[1] The analogy between relaxation in physical systems and computational complexity in combinatorial problems is even clearer when the latter are solved using local search algorithms, *e.g.* simulated annealing or other solving strategies making local moves based on some energy function. Consider a random 3-SAT instance below threshold. We define the energy function (depending on the configuration of Boolean variables) as the number of unsatisfied clauses [16]. The goal of the algorithm is to find a solution, *i.e.* to minimize the energy. The configuration of Boolean variables evolve from some initial value to some solution under the action of the algorithm. During this evolution, the energy of the "system" relaxes from an initial (large) value to zero. Computational complexity is, in this case, equal to the relaxation time of the dynamics before reaching (zero temperature) equilibrium.

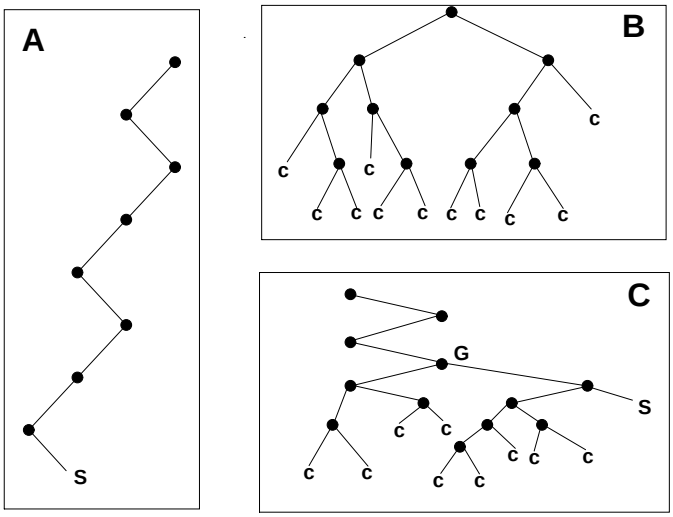

Fig. 2. Examples of search trees. A) *simple branch:* the algorithm finds easily a solution without ever (or with a negligible amount of) backtracking. B) *dense tree:* in the absence of solution, the algorithm builds a "bushy" tree, with many branches of various lengths, before stopping. C) *mixed case, branch + tree:* if many contradictions arise before reaching a solution, the resulting search tree can be decomposed in a single branch followed by a dense tree. The junction G is the highest backtracking node reached back by DPLL.

This paper is an extended version of a previous work [17], where we showed how the dynamics induced by the DPLL search algorithm could be analyzed using off-equilibrium statistical mechanics and combined to the static phase diagram of random K-SAT (with $K = 2,\ 3$) to calculate computational complexity. We start by exposing in a detailed way the definition of the random K-SAT problem and the DPLL procedure in Section 2. We then expose the experimental measures of complexity in Section 3. Our analytical approach is based on the fact that, under DPLL action, the initial instance is modified and follows some trajectory in the phase diagram. The structure of the search tree generated by DPLL procedure is closely related to the nature of the region visited by the instance trajectory. Search trees reduce to essentially one branch – sat instances at low ratio α, Section 4 – or are dense, including an exponential number of branches – unsat instances, Section 5. Mixed structures – SAT instances with ratios slightly below threshold, Section 6 – are made of a branch followed by a dense tree and reflect trajectories crossing the phase boundary between sat and unsat regimes (Fig. 2). While branch trajectories could be obtained straightforwardly from previous works by Chao and Franco [18], we develop in Section 5 a formalism to study the construction of dense trees by DPLL. We show that the latter can be reformulated in terms of a (bidimensional) growth process described by a non-linear partial differential equation. The resolution of this growth equation allows an analytical prediction of the complexity that compares very well to extensive numerical experiments. We present in Section 7 the full complexity diagram of solving random SAT models, and explain the relationship with static studies of the phase transition [13]. Last of all, we show in Section 8 how our study suggests some possible ways to improve existing algorithms.

The reader is kindly recommended to go through [17] to get an overview of our approach and results, before pursuing and reading the present article.

2 Davis-Putnam-Loveland-Logemann algorithm and random 3-SAT

In this section, the reader will be briefly recalled the main features of the random 3-Satisfiability model. We then present the Davis-Putnam-Loveland-Logemann (DPLL) solving procedure, a paradigm of branch and bound algorithm, and the notion of search tree. Finally, we introduce the idea of dynamical trajectory, followed by an instance under the action of DPLL.

2.1 A reminder on random satisfiability

Random K-SAT is defined as follows [8]. Let us consider N Boolean variables x_i that can be either true (T) or false (F) ($i = 1, \ldots, N$). We choose randomly K among the N possible indices i and then, for each of them, a literal, that is, the corresponding x_i or its negation $\bar{x}_i$ with equal probabilities one half. A clause C is the logical OR of the K previously chosen literals, that is C will be true (or satisfied) if and only if at least one literal is true. Next, we repeat this process to obtain M independently chosen clauses $\{C_\ell\}_{\ell=1,\ldots,M}$ and ask for all of them to be true at the same time, *i.e.* we take the logical AND of the M clauses. The resulting logical formula is called an instance of the K-SAT problem. A logical assignment of the x_i's satisfying all clauses, if any, is called a solution of the instance.

For large instances ($M, N \to \infty$), K-SAT exhibits a striking threshold phenomenon as a function of the ratio $\alpha = M/N$ of the number of clauses per variable. Numerical simulations indicate that the probability of finding a solution falls abruptly from one down to zero when α crosses a critical value $\alpha_{\rm C}(K)$ [6–8]. Above $\alpha_{\rm C}(K)$, all clauses cannot be satisfied any longer. This scenario is rigorously established in the $K = 2$ case, where $\alpha_{\rm C} = 1$ [19]. For $K \geq 3$, much less is known; $K(\geq 3)$-SAT belongs to the class of hard, NP-complete computational problems [2]. Studies have mainly concentrated on the $K = 3$ case, whose instances are simpler to generate than for larger values of K. Some lower [20] and upper [21] bounds on $\alpha_{\rm C}(3)$ have been derived, and numerical simulations have recently allowed to find precise estimates of $\alpha_{\rm C}$, *e.g.* $\alpha_{\rm C}(3) \simeq 4.3$ [8,10].

The phase transition taking place in random 3-SAT has attracted a large deal of interest over the past years due to its close relationship with the emergence of computational complexity. Roughly speaking, instances are much harder to solve at threshold than far from criticality [6–9].

We now expose the solving procedure used to tackle the 3-SAT problem.

2.2 The Davis-Putnam-Loveland-Logemann solving procedure

2.2.1 Main operations of the solving procedure and search trees

3-SAT is among the most difficult problems to solve as its size N becomes large. In practice, one resorts to methods that need, *a priori*, exponentially large computational resources. One of these algorithms, the Davis-Putnam-Loveland-Logemann (DPLL) solving procedure [4,5], is illustrated in Figure 3. DPLL operates by trials and errors, the sequence of which can be graphically represented as a search tree made of nodes connected through edges as follows:

1. A node corresponds to the choice of a variable. Depending on the value of the latter, DPLL takes one of the two possible edges.

2. Along an edge, all logical implications of the last choice made are extracted.

3. DPLL goes back to step 1 unless a solution is found or a contradiction arises; in the latter case, DPLL backtracks to the closest incomplete node (with a single descendent edge), inverts the attached variable and goes to step 2; if all nodes carry two descendent edges, unsatisfiability is proven.

Examples of search trees for satisfiable (sat) or unsatisfiable (unsat) instances are shown Figure 2. Computational complexity is the amount of operations performed by DPLL, and is measured by the size of the search tree, *i.e.* the number of nodes.

2.2.2 Heuristics of choice

In the above procedure, step 1 requires to choose one literal among the variables not assigned yet. The choice of the variable and of its value obeys some empirical rule called splitting heuristic. The key idea is to choose variables that will lead to the maximum number of logical implications [22]. Here are some simple heuristics:

- *"Truth table" rule:* fix unknown variables in lexicographic order, from x_1 up to x_N and assign them to *e.g.* true. This is an inefficient rule that does not follow the key principle exposed above.

- *Generalized Unit-Clause (GUC) rule:* choose randomly one literal among the shortest clauses [18]. This is an extension of unit-propagation that fixes literal in unitary clauses. GUC is based on the fact that a clause of length K needs at most $K-1$ splittings to produce a logical implication. So variables are chosen preferentially among short clauses.

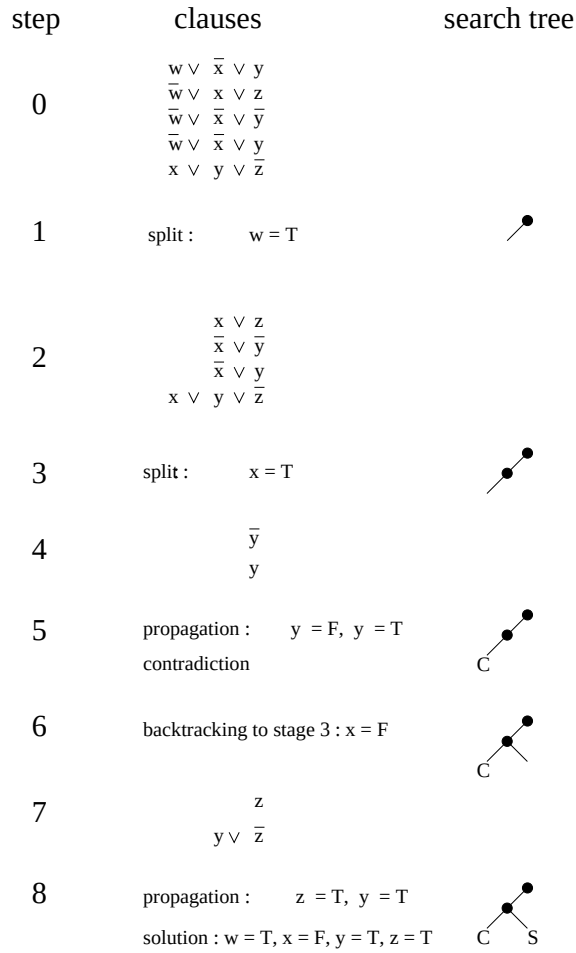

Fig. 3. Example of 3-SAT instance and Davis-Putnam-Loveland-Logemann resolution. Step 0) The instance consists of $M = 5$ clauses involving $N = 4$ variables that can be assigned to true (T) or false (F). $\bar{w}$ means (NOT w) and $\vee$ denotes the logical OR. The search tree is empty. 1) DPLL randomly selects a variable among the shortest clauses and assigns it to satisfy the clause it belongs to, *e.g.* $w = \mathrm{T}$ (splitting with the Generalized Unit Clause – GUC – heuristic). A node and an edge symbolizing respectively the variable chosen (w) and its value (T) are added to the tree. 2) The logical implications of the last choice are extracted: clauses containing w are satisfied and eliminated, clauses including $\bar{w}$ are simplified, and the remaining clauses are left unchanged. If no unitary clause (*i.e.* with a single variable) is present, a new choice of variable has to be made. 3) Splitting takes over. Another node and another edge are added to the tree. 4) Same as step 2 but now unitary clauses are present. The variables they contain have to be fixed accordingly (propagation). 5) The propagation of the unitary clauses results in a contradiction. The current branch dies out and gets marked with C. 6) DPLL backtracks to the last split variable (x), inverts it (F) and creates a new edge. 7) Same as step 4. 8) Propagation of the unitary clauses eliminates all the clauses. A solution S is found. This example show how DPLL find a solution for a satisfiable instance. For an unsatisfiable instance, unsatisfiability is proven when backtracking (see step 6) is not possible anymore since all split variables have already been inverted. In this case, all the nodes in the final search tree have two descendent edges and all branches terminate by a contradiction C.

- *Maximal occurrence in minimum size clauses (MOMS) rule:* pick up the literal appearing most often in shortest clauses. This rule is a refinement of GUC.

Global performances of DPLL depend quantitatively on the splitting rule. From a qualitative point of view, however, the easy-hard-easy picture emerging from experiments is very robust [7,8,10]. Hardest instances seem to be located at threshold. Solving them demand an exponentially large computational effort scaling as $2^{N\omega_C}$. The values of ω_C found in literature roughly range from 0.05 to 0.1, depending on the splitting rule used by DPLL [6,22].

In this paper, we shall focus on the GUC heuristic which is simple enough to allow analytical studies and, yet, is already quite efficient.

2.3 2+p-SAT and instance trajectory

We shall present in Section 3 the experimental results on solving 3-SAT instances using DPLL procedure in a detailed way. The main scope of this paper is to compute in an analytical way the computational complexity ω in the easy and hard regimes. To do so, we have made use of the precious notion of dynamical trajectory, that we now expose.

As shown in Figure 3, the action of DPLL on an instance of 3-SAT causes the reduction of 3-clauses to 2-clauses. Thus, a mixed 2+p-SAT distribution [13], where p is the fraction of 3-clauses, may be used to model what remains of the input instance at a node of the search tree. A 2+p-SAT formula of parameters p, α is the logical AND of two uncorrelated random 2-SAT and 3-SAT instances including $\alpha\,(1-p)\,N$ and $\alpha\,p\,N$ clauses respectively. Using experiments [13] and statistical mechanics calculations [12], the threshold line $\alpha_C(p)$ may be obtained with the results shown in Figure 4 (full line). Replica calculations suggest that the sat/unsat transition taking place at $\alpha_C(p)$ is continuous if $p < p_S$ and discontinuous if $p > p_S$, where $p_S \simeq 0.41$ [23]. The tricritical point T_S is shown in Figure 4. Below p_S, the threshold $\alpha_C(p)$ coincides with the upper bound $1/(1-p)$, obtained when requiring that the 2-SAT subformula only be satisfied. Rigorous studies have shown that $p_S \geq 2/5$, leaving open the possibility it is actually equal to 2/5 [24], or to some slightly larger value [25].

The phase diagram of 2+p-SAT is the natural space in which DPLL dynamic takes place [17]. An input 3-SAT instance with ratio α_0 shows up on the right vertical boundary of Figure 4 as a point of coordinates $(p = 1, \alpha_0)$. Under the action of DPLL, the representative point moves aside from the 3-SAT axis and follows a trajectory. The location of this trajectory in the phase diagram allows a precise understanding of the search tree structure and of complexity.

3 Numerical experiments

3.1 Description of the numerical implementation of the DPLL algorithm

We have implemented DPLL with the GUC rule, see Figure 3 and Section 2.2.2, to have a fast unit propagation and an inexpensive backtracking [6]. The program is divided in three parts. The first routine draws the clauses and represents the data in a convenient structure. The second, main routine updates and saves the state of the search, *i.e.* the indices and values of assigned variables, to allow an easy backtracking. Then, it checks if a solution is found; if not, a new variable is assigned. The third routine extracts the implication of the choice (propagation). If unit clauses have been generated, the corresponding literals are fixed, or a contradiction is detected.

3.1.1 Data representation

Three arrays are used to encode the data: the two first arrays are labelled by the clause number $m = 1, \ldots, M$ and the number $b = 1, \ldots, K$ of the components in the clause (with $K = 2, 3$). The entries of these arrays, initially drawn at random, are the indices $clausnum\,(m, b)$ and the values $clausval\,(m, b)$, true or false, of the variables. The indices of the third array are integers $i = 1, \ldots, N$ and $j = 1, \ldots, o_i$ where o_i is the number of occurrences of x_i in the clauses (from zero to M). The entries of the matrix, $a\,(i, j)$, are the numbers of the corresponding clauses (between 1 and M).

3.1.2 Updating of the search state

If the third routine has found a contradiction, the second routine goes back along the branch, inverts the last assigned variable and calls again the third routine. If not, the descent along the branch is pursued. A Boolean-valued vector points to the assigned variables, while the values are stored in another unidimensional array. For each clause, we check if the variables are already assigned and, if so, whether they are in agreement with the clauses. When splitting occurs, a new variable is fixed to satisfy a 2-clause, *i.e.* a clause with one false literal and two unknown variables, and the third subroutine is called. If there are only 3-clauses, a new variable is fixed to satisfy any 3-clause and the third subroutine is called. The variable chosen and its value are stored in a vector with index the length of the branch, *i.e.* the number of nodes it contains, to allow future backtracking. If there are neither 2- nor 3-clauses left, a solution is found.

3.1.3 Consequences of a choice and unit propagation

All clauses containing the last fixed variable are analyzed by taking into account all possibilities: 1. the clause is satisfied; 2. the clause is reduced to a 2- or 1-clause; 3. the

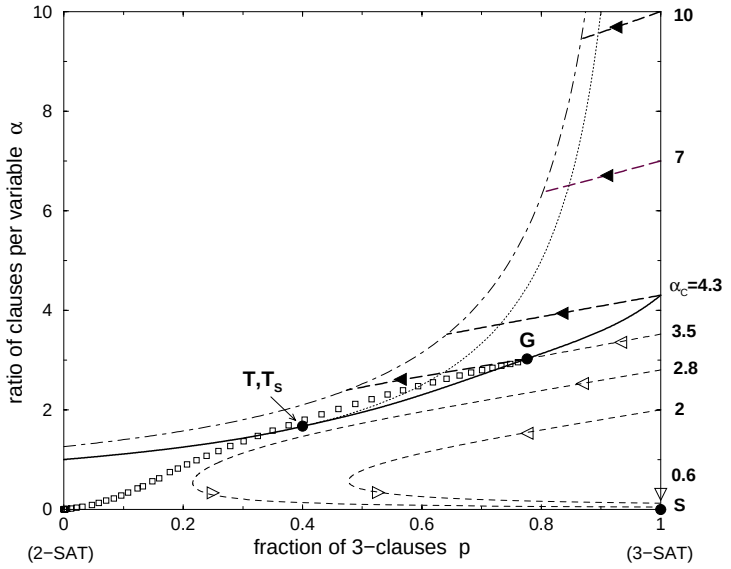

Fig. 4. Phase diagram of the 2+p-SAT model and trajectories generated by DPLL. The threshold line $\alpha_{\rm C}(p)$ (bold full line) separates sat (lower part of the plane) from unsat (upper part) phases. Extremities lie on the vertical 2-SAT (left) and 3-SAT (right) axis at coordinates $(p = 0, \alpha_{\rm C} = 1)$ and $(p = 1, \alpha_{\rm C} \simeq 4.3)$ respectively. The threshold line coincides with the $\alpha = 1/(1-p)$ curve (dotted line) when $p \leq p_{\rm S} \simeq 0.41$, that is, up to the tricritical point $\rm T_S$. Departure points for DPLL trajectories are located on the 3-SAT vertical axis and the corresponding values of α_0 are explicitly given. Dashed curves represent tree trajectories in the unsat region (thick lines, black arrows) and branch trajectories in the sat phase (thin lines, empty arrows). Arrows indicate the direction of motion along trajectories parametrized by the fraction t of variables set by DPLL. For small ratios, *e.g.* $\alpha_0 = 2 < \alpha_{\rm L} \simeq 3.003$, trajectories remain confined in the sat phase. At $\alpha_{\rm L}$, the single branch trajectory hits tangentially the threshold line in T of coordinates $(2/5, 5/3)$, very close to $\rm T_S$. In the intermediate range $\alpha_{\rm L} < \alpha_0 < \alpha_{\rm C}$, the branch trajectory intersects the threshold line at some point G that depends on α. A dense tree then grows in the unsat phase, as happens when 3-SAT departure ratios are above threshold $\alpha > \alpha_{\rm C} \simeq 4.3$. The tree trajectory halts on the dot-dashed curve $\alpha \simeq 1.259/(1-p)$ where the tree growth process stops. At this point, DPLL has reached back the highest backtracking node in the search tree, that is, the first node when $\alpha_0 > \alpha_{\rm C}$, or node G for $\alpha_{\rm L} < \alpha_0 < \alpha_{\rm C}$. In the latter case, a solution can be reached from a new descending branch (rightmost path in Figure 2C) while, in the former case, unsatisfiability is proven. Small squares show the trajectory corresponding to this successful branch for $\alpha = 3.5$, as obtained from simulations for $N = 300$. The trajectory coincides perfectly with the theoretical branch trajectory up to point G (not shown), and then reaches the $\alpha = 0$ axis when a solution is found.

clause is violated (contradiction). In the second case, the 1-clause is stored to be analyzed by unit-propagation once all clauses containing the variable have been reviewed.

3.2 Characteristic running times

We have implemented the DPLL search algorithm in Fortran 77; the algorithm runs on a Pentium II PC with a 433M Hz frequency clock. The number of nodes added per minute ranges from 300,000 (typically obtained for $\alpha = 3.5$) to 100,000 ($\alpha = 10$) since unit propagation is more and more frequent as α increases. The order of magnitude of the computational time needed to solve an instance are listed in Table 1 for ratios corresponding to hard instances. These times limit the maximal size N of the instances we have experimentally studied and the number of samples over which we have averaged. Some rare instances may be much harder than the typical times indicated in Table 1. For instance, for $\alpha = 3.1$ and $N = 500$, instances are usually solved in about 4 minutes but some samples required more than 2 hours of computation. This phenomenon will be discussed in Section 3.4.1.

Table 1. Typical computational time required to solve some hard random 3-SAT instances for different ratios α and sizes N. Slightly below threshold, some rare instances may require much longer computational time (Sect. 3.4.1).

Ratio α clause/var.	Nb. of var.	Resolution time
20	900	6 seconds
10	700	1 hour
7	400	20 minutes
4.3	350	2 days
3.5	500	20 minutes

3.3 Overview of experiments

3.3.1 Number of nodes of the search tree

We have first investigated complexity by a direct count of the number of splittings, that is the number of nodes (black points) in Figure 2, for sat (Figs. 2A, 2C) and unsat (Fig. 2B) trees.

3.3.2 Histogram of branch length

We have also experimentally investigated the structure of search trees for unsat instances (Fig. 2B). A branch is defined as a path joining the root (first node at the top of the search tree) to a leaf marked with a contradiction C (or a solution S for sat instance) in Figure 2. The length of a branch is the number of nodes it contains. For an unsat instance, the complete search tree depends on the variables chosen at each split, but not on the values they are assigned. Indeed, to prove that there is no solution, all nodes in the search tree have to carry two outgoing branches, corresponding to the two choices of the attached variables. What choice is made first does not matter. This simple remark will be of crucial importance in the theoretical analysis of Section 5.2.1.

We have derived the histogram of the branch lengths by counting the number $B(l)$ of branches having length $l\,N$ once the tree is entirely built up. The histogram is very useful to deduce the complexity in the unsat phase, since in a complete tree the total number of branches B is related to the number of nodes Q through the identity,

$$B = \sum_{l=\frac{1}{N},\ldots,1} B(l) = Q + 1 \qquad (1)$$

that can be inferred from Figure 2B.

3.3.3 Highest backtracking point

Another key property we have focused upon is the highest backtracking point in the search tree. In the unsat phase, DPLL backtracks all the nodes of the tree since no solution can be present. The highest backtracking point in the tree simply coincides with the top (root) node. In the sat phase, the situation is more involved. A solution generally requires some backtracking and the highest backtracking node may be defined as the closest node to the origin through which two branches pass, node G in Figure 2B. We experimentally keep trace of the highest backtracking point by measuring the numbers $C_2(\mathrm{G})$, $C_3(\mathrm{G})$ of 2- and 3-clauses, the number of not-yet-assigned variables $N(\mathrm{G})$, and computing the coordinates $p_\mathrm{G} = C_3(\mathrm{G})/(C_2(\mathrm{G}) + C_3(\mathrm{G})), \alpha_\mathrm{G} = (C_2(\mathrm{G}) + C_3(\mathrm{G}))/N(\mathrm{G})$ of G in the phase diagram of Figure 4.

3.4 Experimental results

3.4.1 Fluctuations of complexity

The size of the search tree built by DPLL is a random variable, due to the (quenched) randomness of the 3-SAT instance and the choices made by the splitting rule ("thermal noise"). We show on Figure 5 the distribution of the logarithms (in base 2, and divided by N) of the number of nodes for different values of N. The distributions are more and more peaked around their mean values $\omega_N(\alpha)$ as the size N increases. However, fluctuations are dramatically different at low and large ratios. For $\alpha = 10$, and more generally in the unsat phase, the distributions are roughly symmetric (Fig. 5A). Tails are small and complexity does not fluctuate too much from sample to sample [26,27]. In the vicinity of α_L, *e.g.* $\alpha = 3.1$, much bigger fluctuations are present. There are large tails on the right flanks of the distributions in Figure 5B, due to the presence of rare and very hard samples [27]. As a consequence, while the logarithm of the complexity seems to be a self-averaging quantity in the thermodynamic limit, complexity itself is not (*e.g.* at $\alpha = 3.1$). We will come back to this point in Section 6.

3.4.2 The easy-hard-easy pattern

We have averaged the logarithm of the number of nodes over 10,000 randomly drawn instances to obtain $\omega_N(\alpha)$. The typical size Q of the search tree is simply $Q = 2^{N\omega_N}$. Results are shown in Figure 1. An easy-hard-easy pattern of complexity appears clearly as the ratio α varies.

- At small ratios, complexity increases as $\gamma(\alpha)N$, that is, only linearly with the size of the instance. DPLL easily finds a solution and the search tree essentially reduces to a single branch shown in Figure 2A. For the GUC heuristic, the linear regime extends up to $\alpha_\mathrm{L} \simeq 3.003$ [18,20].
- Above threshold, complexity grows exponentially with N [28]. The logarithm $\omega(\alpha)$, limit of $\omega_N(\alpha)$ as $N \to \infty$, is maximal at criticality...
- ... and decreases at large ratios as $1/\alpha$ [29]. The "easy" region on the right hand side of Figure 1 is still exponential but with small coefficients ω.

Of particular interest is the intermediate region $\alpha_\mathrm{L} < \alpha < \alpha_\mathrm{C}$. We shall show that complexity is exponential in this range of ratios, and that the search tree is a mixture of the search trees taking place in the other ranges $\alpha < \alpha_\mathrm{L}$ and $\alpha > \alpha_\mathrm{C}$.

Let us mention that, while this paper is devoted to typical-case (happening with probability one) complexity, rigorous results have been obtained that apply to any instance. So far, using a refined version of DPLL, any instance is guaranteed to be solved in less than 1.504^N steps, *i.e.* $\omega < 0.588$. The reader is referred to reference [30] for this worst-case analysis.

3.4.3 Lower sat phase ($\alpha < \alpha_\mathrm{L}$)

The complexity data of Figure 1 obtained for different sizes $N = 50, 75, 100$ are plotted again in Figure 6 after division by N. Data collapse on a single curve, proving that complexity is linear in the regime $\alpha < \alpha_\mathrm{L}$. In the vicinity of the cross over ratio $\alpha = \alpha_\mathrm{L}$ finite-size effects became important in this region. We have thus performed extra simulations for larger sizes $N = 500, 1000$ in the range $2.5 \leq \alpha \leq 3$. that confirm the linear scaling of the complexity.

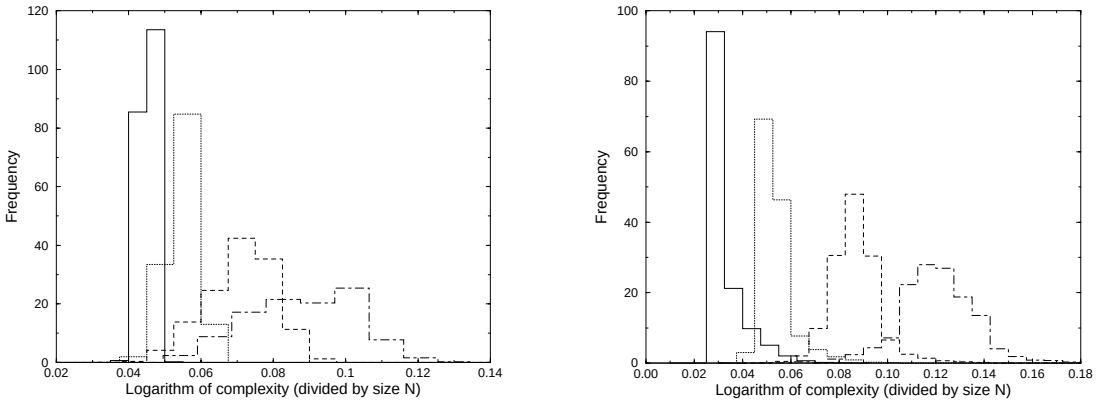

Fig. 5. Distribution of the logarithms (in base 2, and divided by N) of the complexity for four different sizes $N = 30$ (dot-dashed), 50 (dashed), 100 (dotted) and 200 (full line). The histograms are normalized to unity and represent 50,000 instances. The distribution gets more and more concentrated as the size grows. A) Ratio $\alpha = 10$. Curves are roughly symmetrical around their mean with small tails on their flanks; large fluctuations from sample to sample are absent. B) Ratio $\alpha = 3.1$. Curves have large tails on the right, reflecting the presence of rare, very hard samples.

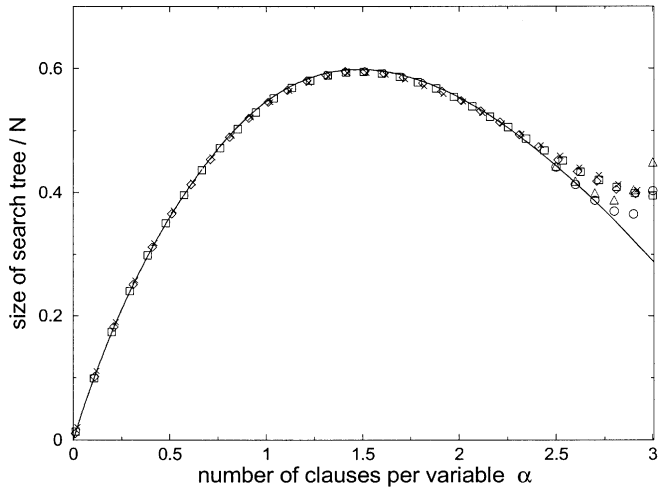

Fig. 6. Complexity of solving in the sat region for $\alpha < \alpha_{\rm L} \simeq 3.003$, divided by the size N of the instances. Numerical data are for sizes N =50 (cross), 75 (square), 100 (diamond), 500 (triangle) and 1000 (circle). For the two biggest sizes, simulations have been carried out for ratios larger than 2.5 only. Data for different N collapse onto the same curve, proving that complexity scales linearly with N. The bold continuous curve is the analytical prediction $\gamma(\alpha)$ from Section 4.3.2. Note the perfect agreement with numerics except at large ratios where finite size effects are important, due to the cross-over to the exponential regime above $\alpha_{\rm L} \simeq 3.003$.

3.4.4 Unsat phase ($\alpha > \alpha_{\rm C}$)

Results for the shape of the search trees are shown in Figure 7. We represent the logarithm $b(l)$, in base 2 and divided by N, of the number $B(l)$ of branches as a function of the branch length $l\,N$, averaged over many samples and for different sizes N and ratios α. When α increases at fixed N, branches are shorter and shorter and less and less numerous, making complexity decrease (Fig. 1).

As N gets large at fixed α, the histogram $b(l)$ becomes a smooth function of l and we can replace the discrete sum in (1) with a continuous integral on the length,

$$Q + 1 = \sum_{l=\frac{1}{N},\ldots,1} 2^{N\,b(l)} \simeq N \int_0^1 {\rm d}l\; 2^{N\,b(l)}. \qquad (2)$$

The integral is exponentially dominated by the maximal value $b_{\rm max}$ of $b(l)$. ω, the limit of the logarithm of the complexity divided by N, is therefore equal to $b_{\rm max}$. Nicely indeed, the height $b_{\rm max}$ of the histogram does not depend on N (within the statistical errors) and gives a straightforward and precise estimate of ω, not affected by finite-size effects. The values of $b_{\rm max}$ as a function of α are listed in the third column of Table 2.

The above discussion is also very useful to interpret the data on the size Q of the search trees. From the quadratic correction around the saddle-point, $b(l) \simeq b_{\rm max} - \beta(l - l_{\rm max})^2/2$, the expected finite size correction to ω_N read

$$\omega_N \simeq \omega + \frac{1}{2N}\log_2 N + \frac{1}{2N}\log_2\left[\frac{2\pi}{\beta \ln 2}\right] + o\left(\frac{1}{N}\right). \qquad (3)$$

We have checked the validity of this equation by fitting $\omega_N - \log_2 N/(2N)$ as a polynomial function of $1/N$. The constant at the origin gives values of ω in very good agreement with $b_{\rm max}$ (second column in Tab. 2) while the linear term gives access to the curvature β. We compare in Table 3 this curvature with the direct measurements of β from the histogram. The agreement is fair, showing that equation (3) is an accurate way of extrapolating data on Q to the infinite size limit.

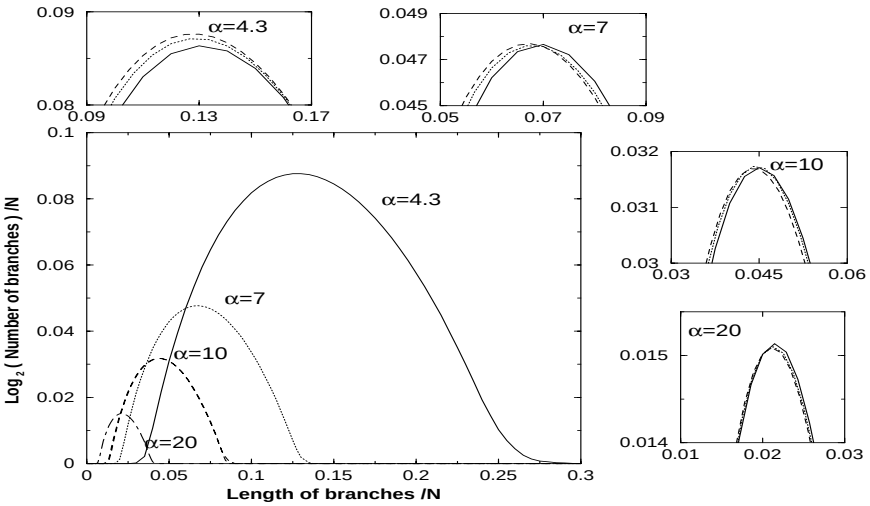

Fig. 7. Logarithm of the number of branches in unsat search trees as a function of the branch length. Main figure: the size of the search tree is a decreasing function of α at fixed N. Histograms are presented here for ratios equal to $\alpha = 4.3$ (solid line, $N = 200$), $\alpha = 7$ (dotted line, $N = 400$), $\alpha = 10$ (dashed line, $N = 600$) and $\alpha = 20$ (dotted-dashed line, $N = 900$) and have been averaged over hundreds of samples. Inset: the heights of the tops of the histograms show a very weak dependence upon N. Numerical extrapolations of ω_N to $N \to \infty$ and statistical errors are reported Table 2. Sat instances (which may be present for small sizes at $\alpha = 4.3$) have not be considered in the averaging procedure. For each inset, we give below the sizes N followed by the number of instances in parenthesis used for averaging. Ratio $\alpha = 4.3$: solid line: 100 (5000), dotted line: 150 (500), dashed line: 200 (400); $\alpha = 7$: solid line: 200 (10000), dotted line: 300 (1000), dashed line: 400 (200). $\alpha = 10$: solid line: 400 (500), dotted line: 500 (400), dashed line: 600 (100); $\alpha = 20$: solid line: 700 (600), dotted line: 800 (1000), dashed line: 900 (1000).

Table 2. Logarithm of the complexity ω from experiments and theory in the unsat phase. Values of ω from measures of search tree sizes (nodes), histograms of branch lengths (histogram) and theory within linear (lin.) and quadratic (quad.) approximations. Note the excellent agreement between theory and experiments for large α.

Ratio α	Experiments		Theory	
of clause/var.	nodes	histogram	lin.	quad.
20	0.0153 ± 0.0002	0.0151 ± 0.0001	0.0152	0.0152
15	0.0207 ± 0.0002	0.0206 ± 0.0001	0.0206	0.0206
10	0.0320 ± 0.0005	0.0317 ± 0.0002	0.0319	0.0319
7	0.0481 ± 0.0005	0.0477 ± 0.0005	0.0477	0.0476
4.3	0.089 ± 0.001	0.0895 ± 0.001	0.0875	0.0852

Table 3. Details on the fits of search tree sizes from equation (3). For different ratios α_0, the slope γ of the fit of $\omega_N - \log_2 N/N$ *vs.* $1/N$ is shown as well as the corresponding curvature β deduced from (3) (column "nodes"). The curvatures measured directly at the top of the branch lengths histograms are listed in the "histogram" column.

Ratio α_0	Slope	Curvature β	
of clause/var.	γ	nodes	histogram
15	-1.47	69.6	75.6
10	-1.32	56.5	47.8
7	-1.06	39.4	29.6
4.3	-0.58	20.2	13.6

3.4.5 Upper sat phase ($\alpha_L < \alpha < \alpha_C$)

To investigate the sat region slightly below threshold $\alpha_L < \alpha < \alpha_C$, we have carried out simulations with a starting ratio $\alpha = 3.5$. Results are shown in Figure 8A. As instances are sat with a high probability, no simple identity relates the number of nodes Q to the number of branches B, see search tree in Figure 2C and we measure the complexity through Q only. Complexity clearly scales exponentially with the size of the instance and exhibits large fluctuations from sample to sample. The annealed complexity (logarithm of the average complexity), $\omega_3^{\rm ann}$, is larger than the typical solving hardness (average of the logarithm of the complexity), $\omega_3^{\rm typ}$, see Table 4.

To reach a better understanding of the structure of the search tree, we have focused on the highest backtracking point G defined in Figure 2C and Section 3.3.3. The coordinates p_G, α_G of point G, averaged over instances are shown for increasing sizes N in Figure 9. The coordinates of G exhibit strong sample fluctuations which make the large N extrapolation, $p_G = 0.78 \pm 0.01$, $\alpha_G = 3.02 \pm 0.02$ rather imprecise.

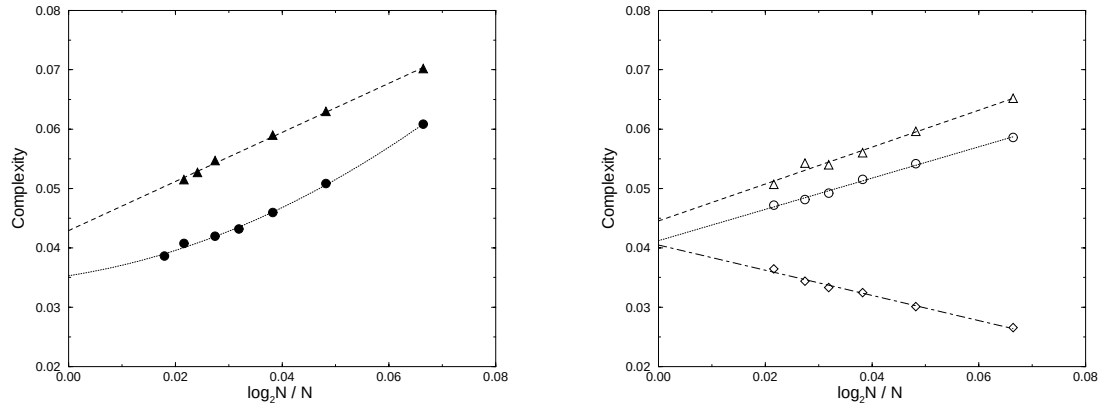

Fig. 8. Solving complexity in the upper sat phase for sizes N ranging from $N = 100$ to $N = 400$. The size of the symbols accounts for the largest statistical error bar. A) 3-SAT problem with ratio $\alpha = 3.5$: typical (average of the logarithm, full circles) and annealed (logarithm of the average, full triangles) size of the search tree. Dotted lines are the quadratic and linear fits of the typical and annealed complexities, giving $\omega_3^{\rm typ} = 0.035 \pm 0.003$ and $\omega_3^{\rm ann} = 0.043 \pm 0.002$ in the infinite size limit. B) Related 2+p-SAT problem with parameters $p_{\rm G} = 0.78, \alpha_{\rm G} = 3.02$. The typical complexity is measured from the size of the search tree (circles) and the top of the branch length distribution (diamonds), with the same large N extrapolation: $\omega_{2+p}^{\rm typ} = 0.041 \pm 0.02$. This value is slightly smaller than the annealed complexity $\omega_{2+p}^{\rm ann} = 0.044 \pm 0.02$ (triangles). All fits are linear.

Table 4. Logarithm of the complexity ω from experiments and theory in the upper sat phase. Experiments determine ω from measures of the annealed complexity (ann.), of the typical search tree sizes (nod.) and of histograms of branch lengths (his.). Data are presented for 3-SAT instances with ratio $\alpha = 3.5$, and 2+p-SAT instances with parameters $p = 0.78, \alpha = 3.02$. Theoretical predictions within linear (lin.) and quadratic (quad.) approximations are reported for the 2+p-SAT model, then for 3-SAT using equation (60).

Parameters		Experiments			Theory	
p	α	$\omega^{\rm ann}$	$\omega_{\rm nod.}^{\rm typ}$	$\omega_{\rm his.}^{\rm typ}$	lin.	quad.
1	3.5	0.043 ± 0.002	0.035 ± 0.003		0.0355	0.0329
0.78	3.02	0.044 ± 0.002	0.041 ± 0.002	0.041 ± 0.002	0.0440	0.0407

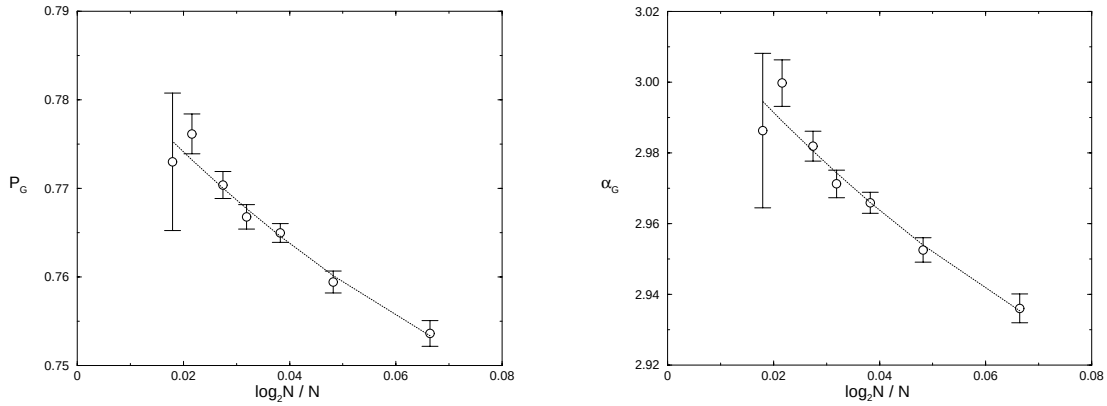

Fig. 9. Coordinates of the highest backtracking point G in the search tree with a starting ratio $\alpha = 3.5$ (the upper sat phase) for different sizes $N = 100, \ldots, 500$ and averaged over 10,000 (small sizes) to 128 ($N = 500$) instances. The fits shown are quadratic functions of the plotting coordinate $\log_2 N/N$, and give the extrapolated location of G in the large size limit: $p_{\rm G} = 0.78 \pm 0.01$, $\alpha_{\rm G} = 3.02 \pm 0.02$. Note the uncertainty on these values due to the small number of instances available at large instance sizes.

In Section 6, we shall show how the solving complexity in the upper sat phase is related to the solving complexity of corresponding 2+p-SAT problems with parameters $p_{\rm G}, \alpha_{\rm G}$.

4 Branch trajectories and the linear regime (lower sat phase)

In this section, we investigate the dynamics of DPLL in the low ratio regime, where a solution is rapidly found (in a linear time) and the search tree essentially reduces to a single branch shown Figure 2A. We start with some general comments on the dynamics induced by DPLL (Sect. 4.1), useful to understand how the trajectory followed by the instance can be computed in the p, α plane (Sect. 4.2). These two first sections merely expose some previous works by Chao and Franco, and the reader is asked to consult [18] for more details. In the last Section 4.3, our numerical and analytical results for the solving complexity are presented.

In this section, as well as in Sections 5 and 6, the ratio of the 3-SAT instance to be solved will be denoted by α_0.

4.1 Remarks on the dynamics of clauses

4.1.1 Dynamical flows of populations of clauses

As pointed out in Section 2.3, under the action of DPLL, some clauses are eliminated while other ones are reduced. Let us call $C_j(T)$ the number of clauses of length j (including j variables), once T variables have been assigned by the solving procedure. T will be called hereafter "time", not to be confused with the computational time necessary to solve a given instance. At time $T = 0$, we obviously have $C_3(0) = \alpha_0 N$, $C_2(0) = C_1(0) = 0$. As Boolean variables are assigned, T increases and clauses of length one or two are produced. A sketchy picture of DPLL dynamics at some instant T is proposed in Figure 10.

We call e_1, e_2, e_3 and w_2, w_1 the flows of clauses represented in Figure 10 when time increases from T to $T+1$, that is, when one more variable is chosen by DPLL after T have already been assigned. The evolution equations for the three populations of clauses read,

$$\begin{aligned} C_3(T+1) &= C_3(T) - e_3(T) - w_2(T) \\ C_2(T+1) &= C_2(T) - e_2(T) + w_2(T) - w_1(T) \\ C_1(T+1) &= C_1(T) - e_1(T) + w_1(T). \end{aligned} \qquad (4)$$

The flows e_j and w_j are of course random variables that depend on the instance under consideration at time T, and on the choice of the variable (label and value) done by DPLL. For a single descent, *i.e.* in the absence of backtracking, and for the GUC heuristic, the evolution process (4) is Markovian and unbiased. The distribution of instances generated by DPLL at time T is uniform over the set of all the instances having $C_j(T)$ clauses of length $j = 1, 2, 3$ and drawn from a set of $N - T$ variables [18].

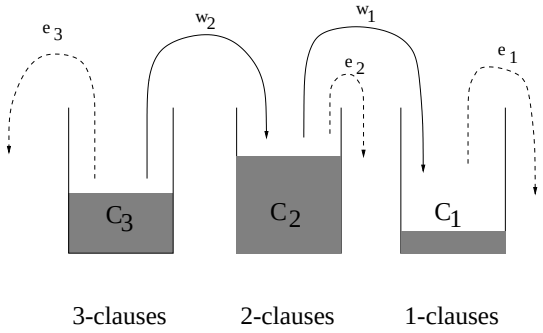

Fig. 10. Schematic view of the dynamics of clauses. Clauses are sorted into three recipients according to their lengths, *i.e.* the number of variables they include. Each time a variable is assigned by DPLL, clauses are modified, resulting in a dynamics of the recipients populations (lines with arrows). Dashed lines indicate the elimination of (satisfied) clauses of lengths 1, 2 or 3. Bold lines represent the reduction of 3-clauses into 2-clauses, or 2-clauses into 1-clauses. The flows of clauses are denoted by e_1, e_2, e_3 and w_2, w_1 respectively. A solution is found when all recipients are empty. The level of the rightmost recipient coincides with the number of unitary clauses. If this level is low (*i.e.* $O(1)$), the probability that two contradictory clauses x and $\bar{x}$ are present in the recipient is vanishingly small. When the level is high (*i.e.* $O(\sqrt{N})$), contradictions will occur with high probability.

4.1.2 Concentration of distributions of populations

As a result of the additivity of (4), some concentration phenomenon takes place in the large size limit. The numbers of clauses of lengths 2 and 3, *a priori* extensive in N, do not fluctuate too much,

$$C_j(T) = N\, c_j\left(\frac{T}{N}\right) + o(N) \qquad (j = 2, 3)\ . \qquad (5)$$

where the c_j's are the densities of clauses of length j averaged over the instance (quenched disorder) and the choices of variables ("thermal" disorder). In other words, the populations of 2- and 3-clauses are self-averaging quantities and we shall attempt at calculating their mean densities only. Note that, in order to prevent the occurrence of contradictions, the number of unitary clauses must remain small and the density c_1 of 1-clauses has to vanish.

4.1.3 Time scales separation and deterministic vs. stochastic evolutions

Formula (5) also illustrates another essential feature of the dynamics of clause populations. Two time scales are at play. The short time scale, of the order of the unity, corresponds to the fast variations of the numbers of clauses $C_j(T)$ ($j = 1, 2, 3$). When time increases from T to $T + O(1)$ (with respect to the size N), all C_j's vary by $O(1)$ amounts. Consequently, the densities of clauses c_j, that is, their numbers divide by N, are not modified. The densities c_js evolve on a long time scale of the order of N and depend on the reduced time $t = T/N$ only.

Due to the concentration phenomenon underlined above, the densities $c_j(t)$ will evolve deterministically with the reduced time t. We shall see below how Chao and Franco calculated their values. On the short time scale, the relative numbers of clauses $D_j(T) = C_j(T) - Nc_j(T/N)$ fluctuate (with amplitude $\ll N$) and are stochastic variables. As said above the evolution process for these relative numbers of clauses is Markovian and the probability rates (master equation) are functions of slow variables only, *i.e.* of the reduced time t and of the densities c_2 and c_3. As a consequence, on intermediary time scales, much larger than unity and much smaller than N, the D_js may reach some stationary distribution that depend upon the slow variables.

This situation is best exemplified in the case $j = 1$ where $c_1(t) = 0$ as long as no contradiction occurs and $D_1(T) = C_1(T)$. Consider for instance a time delay $1 \ll \Delta T \ll N$, *e.g.* $\Delta T = \sqrt{N}$. For times T lying in between $T_0 = t\,N$ and $T_1 = T_0 + \Delta T = t\,N + \sqrt{N}$, the numbers of 2- and 3-clauses fluctuate but their densities are left unchanged and equal to $c_2(t)$ and $c_3(t)$. The average number of 1-clauses fluctuates and follows some master equation whose transition rates (from $C_1' = C_1(T)$ to $C_1 = C_1(T+1)$) define a matrix $\mathcal{M}(C_1, C_1')$ and depend on t, c_2, c_3 only. $\mathcal{M}$ has generally a single eigenvector $\mu(C_1)$ with eigenvalue unity, called equilibrium distribution, and other eigenvectors with smaller eigenvalues (in modulus). Therefore, at time T_1, C_1 has forgotten the "initial condition" $C_1(T_0)$ and is distributed according to the equilibrium distribution $\mu(C_1)$ of the master equation.

To sum up, the dynamical evolution of the clause populations may be seen as a slow and deterministic evolution of the clause densities to which are superimposed fast, small fluctuations. The equilibrium distribution of the latter adiabatically follows the slow trajectory.

4.2 Mathematical analysis

In this section, we expose Chao and Franco's calculation of the densities of 2- and 3-clauses.

4.2.1 Differential equations for the densities of clauses

Consider first the evolution equation (4) for the number of 3-clauses. This can be rewritten in terms of the average density c_3 of 3-clauses and of the reduced time t,

$$\frac{\mathrm{d}c_3(t)}{\mathrm{d}t} = -z_3(t), \tag{6}$$

where $z_3 = \langle e_3 + w_2 \rangle$ denotes the averaged total outflow of 3-clauses (Sect. 4.1.1).

At some time step $T \to T+1$, 3-clauses are eliminated or reduced if and only if they contain the variable chosen by DPLL. Let us first suppose that the variable is chosen in some 1- or 2-clauses. A 3-clause will include this variable or its negation with probability $3/(N-T)$ and disappear with the same probability. Due to the uncorrelation of clauses, we obtain $z_3(t) = 3c_3(t)/(1-t)$. If the literal assigned by DPLL is chosen among some 3-clause, this result has to be increased by one (since this clause will necessarily be eliminated) in the large N limit.

Let us call $\rho_j(t)$ the probability that a literal is chosen by DPLL in a clause of length j $(= 1, 2, 3)$. The normalization of probability imposes that

$$\rho_1(t) + \rho_2(t) + \rho_3(t) = 1, \tag{7}$$

at any time t. Extending the above discussion to 2-clauses, we obtain

$$\begin{aligned} \frac{\mathrm{d}c_3(t)}{\mathrm{d}t} &= -\frac{3}{1-t}c_3(t) - \rho_3(t) \\ \frac{\mathrm{d}c_2(t)}{\mathrm{d}t} &= \frac{3}{2(1-t)}c_3(t) - \frac{2}{1-t}c_2(t) - \rho_2(t), \end{aligned} \tag{8}$$

In order to solve the above set of coupled differential equations, we need to know the probabilities ρ_j. As we shall see, the values of the ρ_j can directly be deduced from the heuristic of choice, the so-called generalized unit-clause (GUC) rule exposed in Section 2.2.2.

The solutions of the differential equations (8) will be expressed in terms of the fraction p of 3-clauses and the ratio α of clauses per variable using the identities

$$p(t) = \frac{c_3(t)}{c_2(t) + c_3(t)}, \qquad \alpha(t) = \frac{c_2(t) + c_3(t)}{1-t}\,. \tag{9}$$

4.2.2 Solution for $\alpha_0 \leq 2/3$

When DPLL is launched, 2-clauses are created with an initial flow $\langle w_2(0) \rangle = 3\,\alpha_0/2$. Let us suppose that $\alpha_0 \leq 2/3$, *i.e.* $w_2(0) \leq 1$. In other words, less than one 2-clause is created each time a variable is assigned. Since the GUC rule compels DPLL to look for literals in the smallest available clauses, 2-clauses are immediately removed just after creation and do not accumulate in their recipient. Unitary clauses are almost absent and we have

$$\rho_1(t) = 0; \quad \rho_2(t) = \frac{3c_3(t)}{2(1-t)}; \quad \rho_3(t) = 1 - \rho_2(t) \qquad (\alpha_0 < 2/3). \tag{10}$$

The solutions of (8) with the initial condition $p(0) = 1, \alpha(0) = \alpha_0$ read

$$\begin{aligned} p(t) &= 1, \\ \alpha(t) &= (\alpha_0 + 2)(1-t)^{3/2} - 2(1-t). \end{aligned} \tag{11}$$

Solution (11) confirms that the instance never contains an extensive number of 2-clauses. At some final time t_{end}, depending on the initial ratio, $\alpha(t_{\mathrm{end}})$ vanishes: no clause is left and a solution is found.

4.2.3 Solution for $2/3 < \alpha_0 < \alpha_L$

We now assume that $\alpha_0 > 2/3$, *i.e.* $\langle w_2(0)\rangle > 1$. In other words, more than one 2-clause is created each time a variable is assigned. 2-clauses now accumulate, and give rise to unitary clauses. Due to the GUC prescription, in presence of 1- or 2-clauses, a literal is never chosen in a 3-clause. Thus,

$$\rho_1(t) = \frac{c_2(t)}{1-t}\ ; \quad \rho_2(t) = 1-\rho_1(t); \quad \rho_3(t) = 0 \qquad (\alpha_0 > 2/3), \quad (12)$$

as soon as $t > 0$. The solutions of (8) now read

$$p(t) = \frac{4\alpha_0(1-t)^2}{\alpha_0(1-t)^2 + 3\alpha_0 + 4\ln(1-t)},$$
$$\alpha(t) = \frac{\alpha_0}{4}(1-t)^2 + \frac{3\alpha_0}{4} + \ln(1-t). \quad (13)$$

Solution (13) requires that the instance contains an extensive number of 2-clauses. This is true at small times since $p'(0) = 1/\alpha_0 - 3/2 < 0$. At some time $t^* > 0$, depending on the initial ratio, $p(t^*)$ reaches back unity: no 2-clause are left and hypothesis (12) breaks down. DPLL has therefore reduced the initial formula to a smaller 3-SAT instance with a ratio $\alpha^* = \alpha(t^*)$. It can be shown that $\alpha^* < 2/3$. Thus, as the dynamical process is Markovian, the further evolution of the instance can be calculated from Section 4.2.2.

4.2.4 Trajectories in the p, α plane

We show in Figure 4 the trajectories obtained for initial ratios $\alpha_0 = 0.6$, $\alpha_0 = 2$ and $\alpha_0 = 2.8$. When $\alpha_0 > 2/3$, the trajectory first heads to the left (Sect. 4.2.3) and then reverses to the right until reaching a point on the 3-SAT axis at small ratio $\alpha^*(< 2/3)$ without ever leaving the sat region. Further action of DPLL leads to a rapid elimination of the remaining clauses and the trajectory ends up at the right lower corner S, where a solution is achieved (Sect. 4.2.2).

As α_0 increases up to α_L, the trajectory gets closer and closer to the threshold line $\alpha_C(p)$. Finally, at $\alpha_L \simeq 3.003$, the trajectory touches the threshold curve tangentially at point T with coordinates $(p_T = 2/5, \alpha_T = 5/3)$. Note the identity $\alpha_T = 1/(1-p_T)$.

4.3 Complexity

In this section, we compute the computational complexity in the range $0 \le \alpha_0 \le \alpha_L$ from the previous results.

4.3.1 Absence of backtracking

The trajectories obtained in Section 4.2.4 represent the deterministic evolution of the densities of 2- and 3-clauses when more and more variables are assigned. Equilibrium fluctuations of number of 1-clauses have been calculated by Frieze and Suen [20]. The stationary distribution $\mu_t(C_1)$ of the population of 1-clauses can be exactly computed at any time t. The most important result is the probability that C_1 vanishes,

$$\mu_t(0) = 1 - \alpha(t)\big(1-p(t)\big). \quad (14)$$

$\mu_t(0)$ (respectively $1-\mu_t(0)$) may be interpreted as the probability that a variable assigned by DPLL at time t is chosen through splitting (resp. unit–propagation). When DPLL starts solving a 3-SAT instance, $\mu_{t=0}(0) = 1$ and many splits are necessary. If the initial ratio α_0 is smaller than 2/3, this statement remains true till the final time $t_{\rm end}$ and the absence of 1-clauses prevents the onset of contradiction. Conversely, if $2/3 < \alpha_0 < \alpha_L$, as t grows, $\mu_t(0)$ decreases are more and more variables are fixed through unit-propagation. The population of 1-clauses remains finite and the probability that a contradiction occurs when a new variable is assigned is $O(1/N)$ only. However small is this probability, $O(N)$ variables are fixed along a complete trajectory. The resulting probability that a contradiction never occurs is strictly smaller than unity [20],

$$\text{Proba(no contradiction)} = \exp\left(-\frac{1}{4}\int_0^{t_{\rm end}} \frac{{\rm d}t}{1-t}\frac{(1-\mu_t(0))^2}{\mu_t(0)}\right). \quad (15)$$

Frieze and Suen have shown that contradictions have no dramatic consequences. The number of backtrackings necessary to find a solution is bounded from above by a power of $\log N$. The final trajectory in the p, α plane is identical to the one shown in Section 4.2.4 and the increase of complexity is negligible with respect to $O(N)$.

When α_0 reaches α_L, the trajectory intersects the $\alpha = 1/(1-p)$ line in T. At this point, $\mu(0)$ vanishes and backtracking enters massively into play, signaling the cross-over to the exponential regime.

4.3.2 Length of trajectories

From the above discussion, it appears that a solution is found by DPLL essentially at the end of a single descent (Fig. 2A). Complexity thus scales linearly with N with a proportionality coefficient $\gamma(\alpha_0)$ smaller than unity.

For $\alpha_0 < 2/3$, clauses of length unity are never created by DPLL (Sect. 4.2.2). Thus, DPLL assigns the overwhelming majority of variables by splittings. $\gamma(\alpha_0)$ simply equals the total fraction $t_{\rm end}$ of variables chosen by DPLL. From (11), we obtain

$$\gamma(\alpha_0) = 1 - \frac{4}{(\alpha_0+2)^2}, \qquad \alpha_0 \le 2/3. \quad (16)$$

For larger ratios, *i.e.* $\alpha_0 > 2/3$, the trajectory must be decomposed into two successive portions (Sect. 4.2.3). During the first portion, for times $0 < t < t^*$, 2-clauses are present with a non vanishing density $c_2(t)$. Some of

these 2-clauses are reduced to 1-clauses that have to be eliminated next. Consequently, when DPLL assigns an infinitesimal fraction $\mathrm{d}t$ of variables, a fraction $\rho_1(t) = \alpha(t)(1-p(t))\mathrm{d}t$ are fixed by unit-propagation only. The number of nodes (divided by N) along the first part of the branch thus reads,

$$\gamma_1 = t^* - \int_0^{t^*} \mathrm{d}t\ \alpha(t)(1-p(t)). \tag{17}$$

At time t^*, the trajectory touches back the 3-SAT axis $p=1$ at ratio $\alpha^* \equiv \alpha(t^*) < 2/3$. The initial instance is then reduces to a smaller and smaller 3-SAT formula, with a ratio $\alpha(t)$ vanishing at t_{end}. According to the above discussion, the length of this second part of the trajectory equals

$$\gamma_2 = t_{\mathrm{end}} - t^*. \tag{18}$$

It results convenient to plot the total complexity $\gamma = \gamma_1 + \gamma_2$ in a parametric way. To do so, we express the initial ratio α_0 and the complexity γ in terms of the end time t^* of the first part of the branch. A simple calculation from equations (13) leads to

$$\begin{aligned}\alpha_0(t^*) =& -\frac{4\ln(1-t^*)}{3t^*(2-t^*)}\\ \gamma(t^*) =& 1 - \frac{4(1-t^*)}{(2+(1-t^*)^2\alpha_0(t^*))^2} + t^*\\ &+ (1-t^*)\ln(1-t^*) - \frac{1}{4}\alpha_0(t^*)\ (t^*)^2(3-t^*).\end{aligned} \tag{19}$$

As t^* grows from zero to $t_L^* \simeq 0.892$, the initial ratio α_0 spans the range $[2/3; \alpha_{\mathrm{L}}]$. The complexity coefficient $\gamma(\alpha_0)$ can be computed from equations (16,19) with the results shown Figure 6. The agreement with the numerical data of Section 3.4.3 is excellent.

5 Tree trajectories and the exponential regime (UNSAT phase)

To present our analytical study of the exponentially large search trees generated when solving hard instances, we consider first a simplified growth tree dynamics in which variables, on each branch, are chosen independently of the 1- or 2-clauses and all branches split at each depth T. This toy model is too simple a growth process to reproduce a search tree analogous to the ones generated by DPLL on unsat instances. In particular, it lacks two essential ingredients of the DPLL procedure: the generalized unit clause rule (literals are chosen from the shortest clauses), and the possible emergence of contradictions halting a branch growth. Yet, the study of the toy model allows us to expose and test the main analytical ideas, before turning to the full analysis of DPLL in Section 5.2.

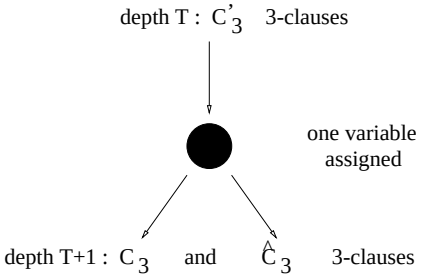

Fig. 11. Schematic representation of the search tree building up in the toy dynamical process. A branch carrying a formula with C_3' 3-clauses splits once a variable is assigned, and gives birth to two branches having C_3 and $\hat{C}_3$ 3-clauses respectively. The variable is assigned randomly, independently of the 3-clauses. Clauses of lengths one or two are not considered.

5.1 Analytical approach for exponentially large search trees: a toy case

5.1.1 The toy growth dynamics

In the toy model of search tree considered hereafter, only 3-clauses matter. Initially, the search tree is empty and the 3-SAT instance is a collection of C_3' 3-clauses drawn from N variables. Next, a variable is randomly picked up and fixed to true or false, leading to the creation of one node and two outgoing branches carrying formulae with C_3 and $\hat{C}_3$ 3-clauses respectively (Fig. 11). This elementary process is then repeated for each branch. At depth or "time" T, that is when T variables have been assigned along each branch, there are 2^T branches in the tree.

The quantity we focus on is the number of branches having a given number of 3-clauses at depth T. On each branch, after a variable has been assigned, C_3 decreases (by clause reduction and elimination) or remains unchanged (when the chosen variable does not appear in the clauses). So, the recipients of 3-clauses attached to each branch leak out with time T (Fig. 10).

We now assume that each branch in the tree evolves independently of the other ones and obeys a Markovian stochastic process. This amounts to neglect correlations between the branches, which could arise from the selection of the same variable on different branches at different times. As a consequence of this decorrelation approximation, the value of C_3 at time $T+1$ is a random variable whose distribution depends only on C_3' (and time T).

The decorrelation approximation allows to write a master equation for the average number $B(C_3;T)$ of branches carrying C_3 3-clauses at depth T in the tree,

$$B(C_3;T+1) = \sum_{C_3'=0}^{\infty} K_{\mathrm{toy}}\ (C_3, C_3';T)\ B(C_3';T). \tag{20}$$

K_{toy} is the branching matrix; the entry $K_{\mathrm{toy}}(C_3, C_3')$ is the averaged number of branches with C_3 clauses issued from a branch carrying C_3' clauses after one variable is assigned. This number lies between zero (if $C_3 > C_3'$) and two (the maximum number of branches produced by a split), and

is easily deduced from the evolution of the recipient of 3-clauses in Figure 10,

$$K_{\rm toy}\ (C_3, C_3'; T) = 2\ \chi\,(C_3' - C_3)\ \binom{C_3'}{C_3' - C_3} \times \left(\frac{3}{N-T}\right)^{C_3' - C_3}\ \left(1 - \frac{3}{N-T}\right)^{C_3}\cdot \quad (21)$$

The factor 2 comes from the two branches created at each split; $\chi(C_3' - C_3)$ equals unity if $C_3' - C_3 \geq 0$, and zero otherwise. The binomial distribution in (21) expresses the probability that the variable fixed at time T appears in $C_3' - C_3$ 3-clauses.

5.1.2 Partial differential equation for the distribution of branches

For large instances $N, M \to \infty$, this binomial distribution simplifies to a Poisson distribution, with parameter $m_3(T) = 3\ C_3'/(N-T)$. The branching matrix (21) thus reads,

$$K_{\rm toy}\ (C_3, C_3'; T) \simeq K_{\rm toy}\ (C_3' - C_3, m_3(T)) = 2\ \chi\,(C_3' - C_3)\ {\rm e}^{-m_3(T)}\ \frac{m_3(T)^{C_3' - C_3}}{(C_3' - C_3)!}\cdot \quad (22)$$

Consider now the variations of the entries of K (22) over a time interval $T_0 = t\,N \leq T \leq T_1 = (t+\epsilon)\,N$. Here, ϵ is a small parameter but of the order of one with respect to N. In other words, we shall first send N to infinity, and then ϵ to zero. $m_3(T)$ weakly varies between T_0 and T_1: $m_3(t) = 3\ c_3'/(1-t) + O(\epsilon)$ where $c_3' = C_3'/N$ is the intensive variable for the number of 3-clauses. The branching matrix (22) can thus be rewritten, for all times T ranging from T_0 to T_1, as

$$K_{\rm toy}(C_3' - C_3, m_3(t)) = 2\ \chi\,(C_3' - C_3) \times {\rm e}^{-m_3(t)}\ \frac{m_3(t)^{C_3' - C_3}}{(C_3' - C_3)!} + O(\epsilon). \quad (23)$$

We may now iterate equation (20) over the whole time interval of length $\mathcal{T} = T_1 - T_0 = \epsilon\,N$,

$$B(C_3; t\,N + \mathcal{T}) = \sum_{C_3'=0}^{\infty}\ [K_{\rm toy}]^{\mathcal{T}}\ (C_3' - C_3; t)\ B(C_3'; t\,N), \quad (24)$$

where $.^{\mathcal{T}}$ denotes the $\mathcal{T}$th power of the branching matrix. As K depends only on the difference $C_3' - C_3$, it can be diagonalized by plane waves $v(q_3, C_3) = {\rm e}^{{\rm i}\,q_3\,C_3}/\sqrt{2\pi}$ with wave numbers $0 \leq q_3 < 2\,\pi$ (because C_3 is an integer-valued number). The corresponding eigenvalues read

$$\lambda_{q_3}(t) = 2\ \exp\left[m_3(t)\ \left({\rm e}^{{\rm i}\,q_3} - 1\right)\right]. \quad (25)$$

Reexpressing matrix (23) using its eigenvectors and eigenvalues, equation (24) reads

$$B(C_3; t\,N + \mathcal{T}) = \sum_{C_3'=0}^{\infty}\ \int_0^{2\pi}\ \frac{{\rm d}q_3}{2\pi}\ \left(\lambda_{q_3}(t)\right)^{\mathcal{T}} \times {\rm e}^{{\rm i}(C_3 - C_3')\,q_3}\ B(C_3'; t\,N). \quad (26)$$

Branches duplicate at each time step and proliferate exponentially with $T = t\,N$. A sensible scaling hypothesis for their number is, at fixed fraction c_3,

$$\lim_{N\to\infty}\ \frac{1}{N}\ln B(c_3\,N; t\,N) = \omega(c_3, t). \quad (27)$$

Note that ω has to be divided by $\ln 2$ to obtain the logarithm of the number of branches in base 2 to match the definition of Section 3.4.1. Similarly, the sum over C_3' in (24, 26) will be replaced in the large N limit with an integral over the real-valued variable r_3 defined through

$$C_3' - C_3 = r_3\,\mathcal{T}. \quad (28)$$

r_3 is of the order of unity when $\mathcal{T}$ gets very large, since the number of 3-clauses that disappear after having fixed $\mathcal{T}$ variables is typically of the order of $\mathcal{T}$. We finally obtain from equation (26),

$$\exp\left(N\ \omega(c_3, t+\epsilon)\right) = \mathcal{T}\ \int_0^{c_3/\epsilon}{\rm d}r_3\ \int_0^{2\pi}\ \frac{{\rm d}q_3}{2\pi} \times \exp\left[N\ \left(\epsilon\ \ln\lambda_{q_3}(t) - {\rm i}\,\epsilon\,r_3\,q_3 + \omega(c_3 + \epsilon\,r_3, t)\right)\right]. \quad (29)$$

In the $N \to \infty$ limit, the integrals in (29) may be evaluated using the saddle point method,

$$\omega(c_3, t+\epsilon) = \max_{r_3, q_3}\left[\epsilon\ \ln\lambda_{q_3}(t) - {\rm i}\,\epsilon\,r_3\,q_3 + \omega(c_3 + \epsilon\,r_3, t)\right] + O(\epsilon^2), \quad (30)$$

due to the terms neglected in (23). Assuming that $\omega(c_3, t)$ is a partially differentiable functions of its arguments, and expanding (30) at the first order in ϵ, we obtain the partial differential equation,

$$\frac{\partial\omega}{\partial t}(c_3, t) = \max_{r_3, q_3}\left(\ln\lambda_{q_3}(t) - {\rm i}\,r_3\,q_3 + r_3\,\frac{\partial\omega}{\partial c_3}(c_3, t)\right). \quad (31)$$

The saddle point lies in $q_3 = -{\rm i}\frac{\partial\omega}{\partial c_3}$, leading to,

$$\frac{\partial\omega}{\partial t}(c_3, t) = \ln 2 - \frac{3\,c_3}{1-t} + \frac{3\,c_3}{1-t}\ \exp\left[\frac{\partial\omega}{\partial c_3}(c_3, t)\right]. \quad (32)$$

It is particularly interesting to note that a partial differential equation emerges in (32). In contradistinction with the evolution of a single branch described by a set of ordinary differential equations (Sect. 4), the analysis of a full tree requires to handle information about the whole distribution ω of branches and, thus, to a partial differential equation.

5.1.3 Legendre transform and solution of the partial differential equation

To solve this equation is convenient to define the Legendre transformation of the function $\omega(c_3, t)$,

$$\varphi(y_3, t) = \max_{c_3} \left[\omega(c_3, t) + y_3\, c_3 \right]. \qquad (33)$$

From a statistical physics point of view this is equivalent to pass, at fixed time t, from a microcanonical 'entropy' ω defined as a function of the 'internal energy' c_3, to a 'free energy' φ defined as a function of the 'temperature' y_3. More precisely, $\varphi(y_3, t)$ is the logarithm divided by N of the generating function of the number of nodes $B(C_3; t\ N)$. Equation (33) defines the Legendre relations between ω and φ,

$$\left.\frac{\partial \omega}{\partial c_3}\right|_{c_3(y_3)} = -y_3 \qquad \text{and} \qquad \left.\frac{\partial \varphi}{\partial y_3}\right|_{y_3(c_3)} = c_3. \qquad (34)$$

In terms of φ and y_3, the partial differential equation (32) reads

$$\frac{\partial \varphi}{\partial t}(y_3, t) = \ln 2 - \frac{3}{1-t}\,(1 - \mathrm{e}^{-y_3})\,\frac{\partial \varphi}{\partial y_3}(y_3, t), \qquad (35)$$

and is linear in the partial derivatives. This is a consequence of the Poissonian nature of the distribution entering K_{toy} (22). The initial condition for the function $\varphi(y_3, t)$ is smoother than for $\omega(c_3, t)$. At time $t = 0$, the search tree is empty: $\omega(c_3, t = 0)$ equals zero for $c_3 = \alpha_0$, and $-\infty$ for $c_3 \neq \alpha_0$. The Legendre transform is thus $\varphi(y_3, 0) = \alpha_0\ y_3$, a linear function of y_3. The solution of equation (35) reads

$$\varphi(y_3, t) = t\ \ln 2 + \alpha_0 \ln \left[1 - (1 - \mathrm{e}^{y_3})\ (1-t)^3 \right] \qquad (36)$$

and, through Legendre inversion,

$$\omega(c_3, t) = t\ \ln 2 + 3c_3\ \ln(1-t) + \alpha_0 \ln \alpha_0 - c_3 \ln c_3 - (\alpha_0 - c_3) \ln \left[\frac{\alpha_0 - c_3}{1 - (1-t)^3} \right], \qquad (37)$$

for $0 < c_3 < \alpha_0$, and $\omega = -\infty$ outside this range. We show in Figure 12 the behavior of $\omega(c_3, t)$ for increasing times t. The curve has a smooth and bell-like shape, with a maximum located in $\hat{c}_3(t) = \alpha_0(1-t)^3, \hat{\omega}(t) = t \ln 2$. The number of branches at depth t equals $B(t) = \mathrm{e}^{N\,\hat{\omega}(t)} = 2^{N\,t}$, and almost all branches carry $\hat{c}_3(t)\,N$ 3-clauses, in agreement with the expression for $c_3(t)$ found for the simple branch trajectories in the case $2/3 < \alpha_0 < 3.003$ (Sect. 4.2.3). The top of the curve $\omega(c_3, t)$, at fixed t, provides direct information about the dominant, most numerous branches.

For the real DPLL dynamics studied in next section, the partial differential equation for the growth process is much more complicated, and exact analytical solutions are not available. If we focus on exponentially dominant branches, we may as a first approximation follow the dynamical evolution of the points of the curve $\omega(c_3, t)$ around the top. To do so, we linearize the partial differential equation (35) for the Legendre transform φ around the origin $y_3 = 0$, see (34),

$$\frac{\partial \varphi}{\partial t}(y_3, t) \simeq \ln 2 - \frac{3}{1-t}\ y_3\ \frac{\partial \varphi}{\partial y_3}(y_3, t). \qquad (38)$$

The solution of the linearized equation, $\varphi(y_3, t) = t\ \ln 2 + \alpha_0(1-t)^3\ y_3$, is itself a linear function of y_3. Through Legendre inversion, the slope gives us the coordinate $\hat{c}_3(t)$ of the maximum of ω, and the constant term, the height $\hat{\omega}(t)$ of the top.

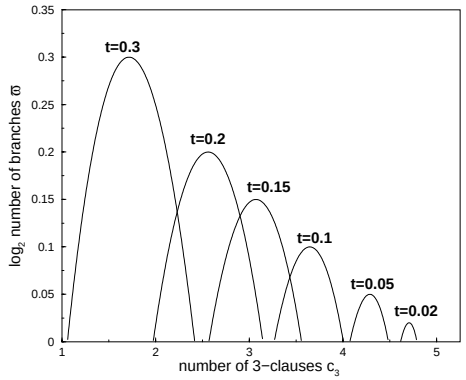

Fig. 12. Logarithm ω of the number of branches (base 2, and divided by N) as a function of the number c_3 of 3-clauses at different times t for the simplified growth process of Section 5.1. At time $t = 0$, the search tree is empty and the ratio of clauses per variable equals $\alpha_0 = 5$. Later on, the tree grows and a whole distribution of branches appears. Dominant branches correspond to the top of the distributions of coordinates $\hat{c}_3(t), \hat{\omega}(t)$. Branches become exponentially more numerous with time, while they carry less and less 3-clauses.

5.2 Analysis of the full DPLL dynamics in the unsat phase

5.2.1 Parallel versus sequential growth of the search tree

A generic refutation search tree in the unsat phase is shown in Figure 2B. It is the output of a *sequential* building process: nodes and edges are added by DPLL through successive descents and backtrackings. We have imagined a different building up, that results in the same complete tree but can be mathematically analyzed: the tree grows in *parallel*, layer after layer (Fig. 13). A new layer is added by assigning, according to DPLL heuristic, one more variable along each living branch. As a result, some branches split, others keep growing and the remaining ones carry contradictions and die out.

As in Section 5.1.1, when going from depth T to $T+1$, we neglect all the correlations between branches that are not immediately issued from a splitting node (Fig. 11). As a result, the evolution of the search tree is Markovian and can be described by a master equation for the average

$$K(C_1,C_2,C_3;C_1',C_2',C_3';T) = \binom{C_3'}{C_3'-C_3}\left(\frac{3}{N-T}\right)^{C_3'-C_3}\left(1-\frac{3}{N-T}\right)^{C_3}\sum_{w_2=0}^{C_3'-C_3}\left(\frac{1}{2}\right)^{C_3'-C_3}\binom{C_3'-C_3}{w_2}$$
$$\times\left\{(1-\delta_{C_1'})\sum_{z_2=0}^{C_2'}\binom{C_2'}{z_2}\left(\frac{2}{N-T}\right)^{z_2}\left(1-\frac{2}{N-T}\right)^{C_2'-z_2}\sum_{w_1=0}^{z_2}\left(\frac{1}{2}\right)^{z_2}\binom{z_2}{w_1}\,\delta_{C_2-C_2'-w_2+z_2}\,\delta_{C_1-C_1'-w_1+1}+\delta_{C_1'}\right.$$
$$\left.\times\sum_{z_2=0}^{C_2'-1}\binom{C_2'-1}{z_2}\left(\frac{2}{N-T}\right)^{z_2}\left(1-\frac{2}{N-T}\right)^{C_2'-1-z_2}\sum_{w_1=0}^{z_2}\left(\frac{1}{2}\right)^{z_2}\binom{z_2}{w_1}\,\delta_{C_2-C_2'-w_2+z_2+1}\,[\delta_{C_1-w_1}+\delta_{C_1-1-w_1}]\right\}, \quad (40)$$

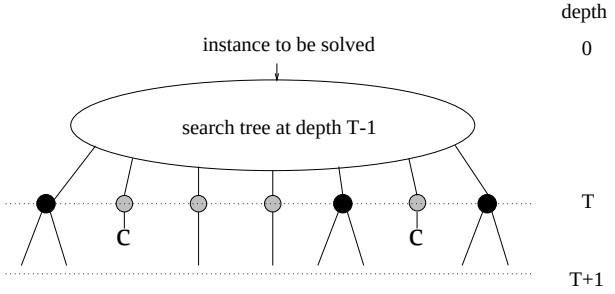

Fig. 13. Imaginary, parallel growth process of an unsat search tree used in the theoretical analysis of the computational complexity. Variables are fixed through unit-propagation, or the splitting heuristic as in the DPLL procedure, but branches evolve in parallel. T denotes the depth in the tree, that is the number of variables assigned by DPLL along each (living) branch. At depth T, one literal is chosen on each branch among 1-clauses (unit-propagation, grey circles not represented in Fig. 2), or 2, 3-clauses (split, black circles as in Fig. 1). If a contradiction occurs as a result of unit-propagation, the branch gets marked with C and dies out. The growth of the tree proceeds until all branches carry C leaves. The resulting tree is identical to the one built through the usual, sequential operation of DPLL.

number of branches B carrying instances with C_j j-clauses ($j = 1, 2, 3$),

$$B(C_1,C_2,C_3;T+1) = \sum_{C_1',C_2',C_3'=0}^{\infty} K\,(C_1,C_2,C_3,C_1',C_2',C_3';T)\;B(C_1',C_2',C_3';T). \quad (39)$$

5.2.2 Branching matrix for DPLL

The branching matrix K for DPLL with GUC heuristic equals

See equation (40) above

where δ_C denotes the Kronecker delta function: $\delta_C = 1$ if $C = 0$, $\delta_C = 0$ otherwise.

To understand formula (40), the picture of recipients in Figure 10 proves to be useful. K expresses the average flow of clauses into the sink (elimination), or to the recipient to the right (reduction), when passing from depth T to depth $T+1$ by fixing one variable. Among the $C_3' - C_3$ clauses that flow out from the leftmost 3-clauses recipient, w_2 clauses are reduced and go into the 2-clauses container, while the remaining $C_3' - C_3 - w_2$ are eliminated. w_2 is a random variable in the range $0 \le w_2 \le C_3' - C_3$ and drawn from a binomial distribution of parameter $1/2$, which represents the probability that the chosen literal is the negation of the one in the clause.

We have assumed that the algorithm never chooses the variable among 3-clauses. This hypothesis is justified *a posteriori* because in the unsat region, there is always (except at the initial time $t = 0$) an extensive number of 2-clauses. Variable are chosen among 1-clauses or, in the absence of the latters, among 2-clauses. The term on the r.h.s. of equation (40) beginning with $\delta_{C_1'}$ (respectively $1-\delta_{C_1'}$) corresponds to the latter (resp. former) case. z_2 is the number of clauses (other than the one from which the variable is chosen) flowing out from the second recipient; it obeys a binomial distribution with parameter $2/(N-T)$, equal to the probability that the chosen variable appears in a 2-clause. The 2-clauses container is, at the same time, poured with w_2 clauses. In an analogous way, the unitary clauses recipient welcomes w_1 new clauses if it was empty at the previous step. If not, a 1-clause is eliminated by fixing the corresponding literal.

The branch keeps growing as long as the level C_1 of the unit clauses recipient remains low, *i.e.* C_1 remains of the order of unity and the probability to have two, or more, 1-clauses with opposite literals can be neglected. For this reason, we do not take into account the (extremely rare) event of 1-clauses including equal or opposite literals and z_1 is always equal to zero. We shall consider later a halt criterion for the tree growth process (dot-dashed line in the phase diagram of Fig. 4), where the condition $C_1 = O(1)$ breaks down due to an avalanche of 1-clauses.

Finally, we sum over all possible flow values w_2, z_2, w_1 that satisfy the conservation laws $C_2 - C_2' = w_2 - z_2$, $C_1 - C_1' = w_1 - 1$ when $C_1' \neq 0$ or, when $C_1' = 0$, $C_2 - C_2' = w_2 - z_2 - 1$, $C_1 = w_1$ if the literal is the same as the one in the clause or $C_1 = w_1 + 1$ if the literal is the negation of the one in the clause. The presence of two δ is responsible for the growth of the number of branches. In the real sequential DPLL dynamics, the inversion of a literal at a node requires backtracking; here, the two edges grow in parallel at each node according to Section 5.2.1.

In the large N limit, the matrix (40) can be written in terms of Poisson distributions through the introduction of the parameters $m_3 = 3\,C_3/(N-T)$ and $m_2 = 2\,C_2/(N-T)$, see equation (22).

5.2.3 The ground state of the branching matrix, and its transition of delocalization

Due to the translational invariances of K in $C_3' - C_3$ and $C_2' - C_2$, the vectors

$$v_{q,q_2,q_3}(C_1', C_2', C_3') = \mathrm{e}^{\mathrm{i}(q_2 C_2' + q_3 C_3')}\, \tilde{v}_q(C_1) \qquad (41)$$

are eigenvectors of the matrix K with eigenvalues

$$\lambda_{q,q_2,q_3} = \exp\left\{ m_3 \left[\frac{\mathrm{e}^{\mathrm{i}q_3}}{2} \left(1 + \mathrm{e}^{-\mathrm{i}q_2}\right) - 1 \right] \right\} \tilde{\lambda}_{q,q_2}, \qquad (42)$$

if and only if $\tilde{v}_q(C_1)$ is an eigenvector, with eigenvalue $\tilde{\lambda}_{q,q_2}$, of the reduced matrix

$$\tilde{K}_{q_2}(C_1, C_1') = \sum_{z_2=0}^{\infty} \mathrm{e}^{-m_2} \frac{(m_2\, \mathrm{e}^{\mathrm{i}q_2})^{z_2}}{z_2!} \sum_{w_1=0}^{z_2} \binom{z_2}{w_1} \frac{1}{2^{z_2}} \times \left\{ \delta_{C_1 - C_1' - w_1 + 1}\, (1 - \delta_{C_1'}) + \mathrm{e}^{\mathrm{i}\,q_2} (\delta_{C_1 - w_1} + \delta_{C_1 - 1 - w_1})\, \delta_{C_1'} \right\}. \qquad (43)$$

Note that, while q_2, q_3 are wave numbers, q is a formal index used to label eigenvectors. The matrix $\tilde{K}_{q_2}$ (43) has been obtained by applying K onto the vector v (41), and summing over C_3', C_2', w_2 and z_3.

The diagonalization of the non hermitian matrix $\tilde{K}_{q_2}$ is exposed in Appendix A, and relies upon the introduction of the generating functions of the eigenvectors $\tilde{v}_q$,

$$\tilde{V}_q(x) = \sum_{C_1=0}^{\infty} \tilde{v_q}(C_1)\, x^{C_1}. \qquad (44)$$

The eigenvalue equation for $\tilde{K}_{q_2}$ translates into a self-consistent equation for $\tilde{V}_q(x)$, the singularities of which can be analyzed in the x plane, and permit to calculate the largest eigenvalue[2] of $\tilde{K}_{q_2}$,

$$\tilde{\lambda}_{0,q_2} = \frac{2}{-1 + \sqrt{1 + 4\,\mathrm{e}^{-\mathrm{i}q_2}}} \times \exp\left\{ -m_2 + m_2 \frac{\mathrm{e}^{\mathrm{i}\,q_2}}{4} \left(1 + \sqrt{1 + 4\,\mathrm{e}^{-\mathrm{i}q_2}}\right) \right\}. \qquad (45)$$

The properties of the corresponding, maximal eigenvector $\tilde{v}_0$ are important. Depending on parameters q_2 and m_2, $\tilde{v}_0(C_1)$ is either localized around small integer values of C_1 (the average number of 1-clauses is of the order of the unity), or extended (the average value of C_1 is of the order of N). As contradictions inevitably arise when $C_1 = O(N)$, the delocalization transition undergone by the maximal eigenvector provides a halt criterion for the growth of the tree.

[2] We show in next section that q_2 is purely imaginary at the saddle-point, and therefore the eigenvalue in equation (45) is real-valued.

5.2.4 Partial differential equation for the search tree growth

Following Section 5.1.2 and equation (26), we write the evolution of the number of branches between times $t\,N$ and $t\,N + \mathcal{T}$ using the spectral decomposition of K,

$$B(C_1, C_2, C_3; t\,N + \mathcal{T}) = \sum_{C_1', C_2', C_3'} \sum_{q} \int_0^{2\pi} \frac{\mathrm{d}q_2}{2\pi} \frac{\mathrm{d}q_3}{2\pi} \times \mathrm{e}^{\mathrm{i}[q_3\,(C_3 - C_3') + q_2\,(C_2 - C_2')]}\, \tilde{v}_q(C_1)\, \tilde{v}_q^+(C_1') \times (\lambda_{q,q_2,q_3})^{\mathcal{T}}\, B(C_1', C_2', C_3', t\,N). \qquad (46)$$

$\sum_q$ denotes the (discrete or continuous) sum on all eigenvectors and $\tilde{v}_q^+$ the left eigenvector of $\tilde{K}$. We make the adiabatic hypothesis that the probability to have C_1 unit clauses at time $t\,N + \mathcal{T}$ becomes stationary on the time scale $1 \ll \mathcal{T} \ll N$ and is independent of the number C_1' of 1-clauses at time $t\,N$ (Sect. 4). As $\mathcal{T}$ gets large, and at fixed q_2, q_3, the sum over q is more and more dominated by the largest eigenvalue $q = 0$, due to the gap between the first eigenvalue (associated to a localized eigenvector) and the continuous spectrum of delocalized eigenvectors (Appendix A). Let us call $\Lambda(q_2, q_3) \equiv \lambda_{0,q_2,q_3}$ this largest eigenvalue, obtained from equations (45, 42). Defining the average of the number of branches over the equilibrium distribution of 1-clauses,

$$B(C_2, C_3; T) = \sum_{C_1=0}^{\infty} B(C_1, C_2, C_3)\, \tilde{v}_0^{\dagger}(C_1), \qquad (47)$$

equation (46) leads to

$$B(C_2, C_3; t\,N + \mathcal{T}) = \sum_{C_2', C_3'=0}^{\infty} \int_0^{2\pi} \frac{\mathrm{d}q_2}{2\pi} \frac{\mathrm{d}q_3}{2\pi} \times \mathrm{e}^{\mathrm{i}[q_3\,(C_3 - C_3') + q_2\,(C_2 - C_2')]} \big(\Lambda(q_2, q_3)\big)^{\mathcal{T}}\, B(C_2', C_3'; t\,N). \qquad (48)$$

The calculation now follows closely the lines of Section 5.1.2. We call $\omega(c_2, c_3, t)$ the limit value of the logarithm of the number of branches carrying an instance with $c_2\,N$ 2-clauses and $c_3\,N$ 3-clauses at depth $t\,N$ as $N \to \infty$. Similarly, we rewrite the sums on C_2', C_3' on the r.h.s. of equation (48) as integrals over the reduced variable $r_2 = (C_2' - C_2)/\mathcal{T}$, $r_3 = (C_3' - C_3)/\mathcal{T}$, see equations (26, 28, 29). A saddle-point calculation of the four integrals over q_2, q_3, r_2, r_3 can be carried out, resulting in a partial differential equation for $\omega(c_2, c_3, t)$,

$$\frac{\partial \omega}{\partial t}(c_2, c_3, t) = \ln \Lambda \left(-\mathrm{i}\frac{\partial \omega}{\partial c_2}, -\mathrm{i}\frac{\partial \omega}{\partial c_3} \right), \qquad (49)$$

or, equivalently,

$$\frac{\partial\omega}{\partial t} = \frac{\partial\omega}{\partial c_2} + \ln\left[\frac{1+\sqrt{4\,\mathrm{e}^{-\frac{\partial\omega}{\partial c_2}}+1}}{2}\right] + \frac{3\,c_3}{1-t}\left[\frac{\mathrm{e}^{\frac{\partial\omega}{\partial c_3}}}{2}\,(1+\mathrm{e}^{-\frac{\partial\omega}{\partial c_2}})-1\right] + \frac{2c_2}{1-t}\left[\frac{\mathrm{e}^{\frac{\partial\omega}{\partial c_2}}}{4}\left(1+\sqrt{4\,\mathrm{e}^{-\frac{\partial\omega}{\partial c_2}}+1}\right)-1\right]. \quad (50)$$

5.2.5 Approximate solution of the partial differential equation

As in Section 5.1.3, we introduce the Legendre transformation of $\omega(c_2, c_3, t)$,

$$\varphi(y_2, y_3, t) = \max_{c_2,c_3}\left(\omega(c_2, c_3, t) + y_2\,c_2 + y_3\,c_3\right). \quad (51)$$

The resulting partial differential equation on φ is given in Appendix B, and cannot be solved analytically. We therefore limit ourselves to the neighborhood of the top of the surface $\omega(c_2, c_3, t)$ through a linearization around $y_2 = y_3 = 0$,

$$\frac{\partial\varphi}{\partial t} \simeq \ln\left(\frac{1+\sqrt{5}}{2}\right) - \left(\frac{5+\sqrt{5}}{10}\right) y_2 - \frac{3}{1-t}\,\left(y_3 - \frac{y_2}{2}\right)\frac{\partial\varphi}{\partial y_3} - \frac{1}{1-t}\left(\frac{3-\sqrt{5}}{2} + \frac{5+3\sqrt{5}}{10}y_2\right)\frac{\partial\varphi}{\partial y_2}. \quad (52)$$

The solution of equation (52) is given by the combination of a particular solution with $\frac{\partial\varphi}{\partial y_3} = 0$ and the solution of the homogeneous counterpart of equation (52). We write below the general solution for any 2+p-SAT unsat problem with parameters $p_0, \alpha_0 > \alpha_{\mathrm{C}}(p_0)$, the 3-SAT case corresponding to $p_0 = 1$. The initial condition at time $t = 0$ reads $\varphi(y_2, y_3, 0) = \alpha_0\,p_0\,y_3 + \alpha_0\,(1-p_0)\,y_2$. We obtain

$$\varphi(y_2, y_3, t) = \hat{c}_2(t)\,y_2 + \hat{c}_3(t)\,y_3 + \hat{\omega}(t) \quad (53)$$

with

$$\begin{aligned}
\hat{c}_2(t) &= \alpha_0\,p_0\,\left(\frac{9\sqrt{5}+75}{116}\right)\left[(1-t)^{\frac{5+3\sqrt{5}}{10}} - (1-t)^3\right] \\
&\quad + \left(\alpha_0(1-p_0) + 2 + \sqrt{5}\right)\,(1-t)^{\frac{5+3\sqrt{5}}{10}} \\
&\quad -(2+\sqrt{5})(1-t), \\
\hat{c}_3(t) &= \alpha_0\,p_0(1-t)^3\,, \\
\hat{\omega}(t) &= \alpha_0\,p_0\left(\frac{15-4\sqrt{5}}{58}\right)\left(1-(1-t)^3\right) \\
&\quad +t\left(\ln\left(\frac{1+\sqrt{5}}{2}\right) + \frac{1+\sqrt{5}}{2}\right) \\
&\quad +\frac{7\sqrt{5}-15}{2}\left(\alpha_0\,p_0\left(\frac{9\sqrt{5}+75}{116}\right) + \alpha_0\,(1-p_0)\right. \\
&\quad \left.+2+\sqrt{5}\right)\left((1-t)^{\frac{5+3\sqrt{5}}{10}} - 1\right).
\end{aligned} \quad (54)$$

Within the linearized approximation, the distribution $\omega(c_2, c_3, t)$ has its maximum located in $\hat{c}_2(t), \hat{c}_3(t)$ with a height $\hat{\omega}(t)$, and is equal to minus infinity elsewhere. The coordinates of the maximum as functions of the depth t defines the tree trajectory, *i.e.* the evolution, driven by the action of DPLL, of the dominant branches in the phase diagram of Figure 4. We obtain straightforwardly this trajectory from equations (54) and the transformation rules (9).

The numerical resolution of PDE (50), through the use of the Legendre transform (51), is presented in [31].

5.2.6 Interpretation of the tree trajectories and results for the complexity

In Figure 4, the tree trajectories corresponding to solving 3-SAT instances with ratios $\alpha_0 = 4.3, 7$ and 10 are shown. The trajectories start on the right vertical axis $p = 1$ and head to the left until they hit the halt line (dot-dashed curve),

$$\alpha = \frac{3+\sqrt{5}}{2}\,\ln\left(\frac{1+\sqrt{5}}{2}\right)\,\frac{1}{1-p} \simeq \frac{1.259}{1-p}, \quad (55)$$

at some time $t_{\mathrm{h}} > 0$, which depends on α_0. On the halt line, a delocalization transition for the largest eigenvector takes place and causes an avalanche of unitary clauses with the emergence of contradictions, preventing branches from further growing. The delocalization transition taking place on the halt line means that the stationary probability $\mu_t(0)$ of having no unit-clause

$$\mu_t(0) = \frac{\tilde{v}_0(0)}{\sum_{C_1=0}^{\infty}\tilde{v}_0(C_1)} = \frac{\tilde{V}_0(0)}{\tilde{V}_0(1)}, \quad (56)$$

vanishes at $t = t_{\mathrm{h}}$. From equations (42, 45, 56, A.1, A.3), the largest eigenvalue for dominant branches, $\Lambda(0,0)$

reaches its lowest value, $\Lambda(0,0) = 1$, on the halt line (55) (for parameters $y_2 = y_3 = 0$, see Appendix A and Fig. 17). As expected, the emergence of contradictions on dominant branches coincides with the halt of the tree growth, see equation (49).

The logarithm $\hat{\omega}(t)$ of the number of dominant branches increases along the tree trajectory, from zero at $t = 0$ up to some value $\hat{\omega}(t_{\rm h}) > 0$ on the halt line. This final value, divided by $\ln 2$, is our analytical prediction (within the linear approximation of Sect. 5.2.5) for the complexity ω. We describe in Appendix B, a refined, quadratic expression for the Legendre transform φ (51) that provides another estimate of ω.

The theoretical values of ω, within linear and quadratic approximations, are shown in Table 2 for $\alpha_0 = 20, 15, 10, 7, 4.3$, and compare very well with numerical results. Our calculation, which is fact an annealed estimate of ω (Sect. 3.4.1), is very accurate. The decorrelation approximation (Sect. 5.1) becomes more and more precise with larger and larger ratios α_0. Indeed, the repetition of variables in the search tree decreases with the size of the tree. For large values of α_0, we obtain

$$\omega(\alpha_0) \sim \frac{(3+\sqrt{5})\left[\ln\left(\frac{1+\sqrt{5}}{2}\right)\right]^2}{6\,\ln 2\,\alpha_0} \sim \frac{0.292}{\alpha_0}\,. \qquad (57)$$

The $1/\alpha_0$ scaling of ω has been previously proven by Beam, Karp, and Pitassi [29], independently of the particular heuristics used. Showing that there is no solution for random 3-SAT instances with large ratios α_0 is relatively easy, since assumptions on few variables generate a large number of logical consequences, and contradictions emerge quickly (Fig. 1). This result can be inferred from Figure 4. As α_0 increases, the distance between the vertical 3-SAT axis and the halt line decreases; consequently, the trajectory becomes shorter, and the size of the search tree, smaller.

5.2.7 Length of the dominant branches

To end with, we calculate the length of dominant branches. The probability that a splitting occurs at time t is $\mu_t(0)$ defined in (56). Let us define $n(L,T)$ as the number of branches having L nodes at depth T in the tree. The evolution equation for n is $n(L,T+1) = (1-\mu_t(0))\,n(L,T) + 2\,\mu_t(0)\;\; n(L-1,T)$. The average branch length at time T, $\langle L(T)\rangle = \sum_{L=1}^{\infty} L\,n(L,T)/\sum_{L=0}^{\infty} n(L,T)$, obeys the simple evolution relation $\langle L(T+1)\rangle - \langle L(T)\rangle = 2\,\mu_t(0)/(1+\mu_t(0))$. Therefore, the average number of nodes (divided by N), $l = \langle L(t\,N)\rangle/N$, present along dominant branches once the tree is complete, is equal to

$$\langle \ell \rangle = \int_0^{t_{\rm h}} {\rm d}t\, \frac{2\,\mu_t(0)}{1+\mu_t(0)} \qquad (58)$$

where $t_{\rm h}$ is the halt time. For large ratios α_0, the average length of dominant branches scales as

$$\langle \ell \rangle(\alpha_0) \sim \frac{(3+\sqrt{5})\ln\left(\frac{1+\sqrt{5}}{2}\right)+1-\sqrt{5}}{3\,\alpha_0} \sim \frac{0.428}{\alpha_0}\,. \qquad (59)$$

The good agreement between this prediction and the numerical results can be checked on the insets of Figure 7 for different values of α_0.

6 Mixed trajectories and the intermediate exponential regime (upper sat phase)

In this section, we show how the complexity of solving 3-SAT instances with ratios in the intermediate range $\alpha_{\rm L} < \alpha_0 < \alpha_{\rm C}$ can be understood by combining the previous results on branch and tree trajectories.

6.1 Branch trajectories and the critical line of 2+p-SAT

In the upper sat phase, the single branch trajectory intersects the critical line $\alpha_{\rm C}(p)$ at some point G, whose coordinates $p_{\rm G}, \alpha_{\rm G}$ depend on the initial ratio α_0. The point G corresponding to $\alpha_0 = 3.5$ is shown in Figure 4. Beyond G, the branch trajectory (which describes the first descending branch in the tree) enters the unsat phase, meaning that the solving procedure has turned a satisfiable 3-SAT instance into an unsatisfiable 2+p-SAT instance.

This situation is schematized in Figure 14. $\rm G_1$ denotes the highest node in the tree carrying an unsat instance. $\rm G_0$ is its immediate ancestor. We also denote by N_0 and N_1 the average numbers of variables not assigned by DPLL at points $\rm G_0$ and $\rm G_1$ respectively: $N_1 < N_0 < N$.

The crossing of the critical line corresponds to the move from $\rm G_0$ to $\rm G_1$ through the assignment of one variable, say x to true. Note that, while the parameters p, α of the 2+p-SAT instances corresponding to $\rm G_0$ and $\rm G_1$ are not equal for finite size instances, they both tend to $p_{\rm G}, \alpha_{\rm G}$ in the $N \to \infty$ limit. Similarly, N_0/N and N_1/N converge to $1 - t_{\rm G}$, where $t_{\rm G}$ is the fraction of variable assigned by the algorithm when reaching point G.

Once DPLL has reached node $\rm G_1$, a refutation subtree must be built by DPLL below $\rm G_1$ in the tree. The corresponding (sub)tree trajectory, penetrating the unsat phase from G up to the arrest line, is shown in Figure 4. Let us call $Q_1 \equiv 2^{N\,\omega}$ the size of this subtree. Once the subtree has been entirely backtracked, DPLL is back in $\rm G_0$, assigns x to false, starts descending on the right side of (Fig. 14). Further backtracking will be necessary but a solution will eventually be found. The branch containing the solution is the rightmost path in Figure 2C. Its trajectory is highly non typical among the branches in the tree (which almost surely follow the (sub)tree trajectory described above), and is plotted in Figure 4.

Q_1 obviously provides a lower bound to the total complexity. It is a reasonable assumption that the amount of

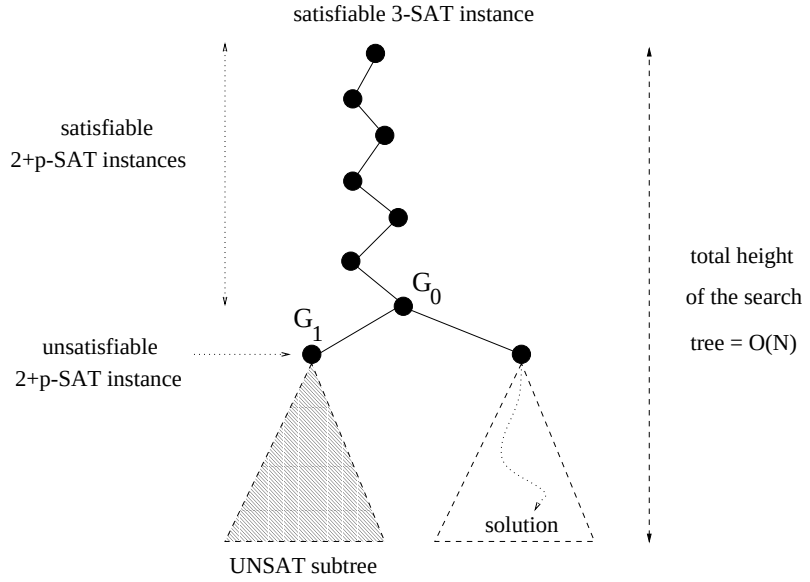

Fig. 14. Detailed structure of the search tree in the upper sat phase ($\alpha_{\rm L} < \alpha < \alpha_{\rm C}$). DPLL starts with a satisfiable 3-SAT instance and transforms it into a sequence of 2+p-SAT instances. The leftmost branch in the tree symbolizes the first descent made by DPLL. Above node $\rm G_0$, instances are satisfiable while below $\rm G_1$, instances have no solutions. A grey triangle accounts for the (exponentially) large refutation subtree that DPLL has to go through before backtracking above $\rm G_1$ and reaching $\rm G_0$. By definition, the highest node reached back by DPLL is $\rm G_0$. Further backtracking, below $\rm G_0$, will be necessary but a solution will be eventually found (right subtree), see Figure 2C.

backtracking in the right subtree (containing the solution) will be, to the exponentially dominant order, equal to Q_1. Therefore, the logarithm of the total size of the search tree (divided by N) is simply equal to ω.

6.2 Analytical calculation of the size of the refutation subtree

The coordinates $p_{\rm G}, \alpha_{\rm G} = \alpha_{\rm C}(p_{\rm G})$ of the crossing point G depend on the initial 3-SAT ratio α_0 and may be computed from the knowledge of the 2+p-SAT critical line $\alpha_{\rm C}(p)$ and the branch trajectory equations (13). For $\alpha_0 = 3.5$, we obtain $p_{\rm G} = 0.78$ and $\alpha_{\rm G} = 3.02$. Point G is reached by the branch trajectory once a fraction $t_{\rm G} \simeq 0.19$ of variables have been assigned by DPLL.

Once G is known, we consider the unsatisfiable 2+p-SAT instances with parameters $p_{\rm G}, \alpha_{\rm G}$ as a starting point for DPLL. The calculation exposed in Section 5 can be used with initial conditions $p_{\rm G}, \alpha_{\rm G}$. We show in Table 4 the results of the analytical calculation of $\omega_{\rm G}$, within linear and quadratic approximations for a starting ratio $\alpha_0 = 3.5$. Note that the discrepancy between both predictions is larger than for higher values of α_0.

The logarithm ω of the total complexity is defined through the identity $2^{N\,\omega} = 2^{N\,(1-t_{\rm G})\,\omega_{\rm G}}$, or equivalently,

$$\omega = \omega_{\rm G}\,(1 - t_{\rm G}). \qquad (60)$$

The resulting value for $\alpha_0 = 3.5$ is shown in Table 4.

6.3 Comparison with numerics for $\alpha_0 = 3.5$

We have checked numerically the scenario of Section 6.1 in two ways.

First, we have computed during the action of DPLL, the coordinates in the p, α plane of the highest backtracking point in the search tree. The agreement with the coordinates of G computed in the previous paragraph is very good (Sect. 3). However, the numerical data show large fluctuation and the experimental fits are not very accurate, leading to uncertainties on $p_{\rm G}$ and $\alpha_{\rm G}$ of the order of 0.01 and 0.02 respectively. In addition, note that the analytical values of the coordinates of G are not exact since the critical line $\alpha_{\rm C}(p)$ is not rigorously known (Sect. 2.3).

Secondly, we compare in Table 4 the experimental measures and theoretical predictions of the complexity of solving unsatisfiable 2+p-SAT instances with parameters $p_{\rm G}, \alpha_{\rm G}$. The agreement between all values is quite good and leads to $\omega_G = 0.042 \pm 0.002$. Numerics indicate that the annealed value of the complexity is equal (or slightly larger) than the typical value. Therefore the annealed calculation developed in Section 5 agrees well the data obtained for 2+p-SAT instances. Once $\omega_{\rm G}$ and $t_{\rm G}$ are known, equation (60) gives access to the theoretical value of ω.

The agreement between theory and experiment is very satisfactory (Tab. 4). Nevertheless, let us stress the existence of some uncertainty regarding the values of the highest backtracking point coordinates $p_{\rm G}, \alpha_{\rm G}$. Numerical simulations on 2+p-SAT instances and analytical checks show that ω depends strongly on the initial fraction p_0. Variations of the initial parameter $p_{\rm G}$ around 0.78 by $\Delta p_0 = 0.01$ change the final result for the complexity by $\Delta\omega = 0.003 - 0.004$, twice as large as the statistical

uncertainty at fixed $p_0 = p_G = 0.78$. Improving the accuracy of the data would require a precise determination of the coordinates of G.

We show in Figure 4 the trajectory of the atypical, rightmost branch (ending with a solution) in the tree, obtained from simulations for $N = 300$. It comes as no surprise that this trajectory, which carries a satisfiable and highly biased 2+p-SAT instance, may enter the unsat region defined for the unbiased 2+p-SAT distribution. The trajectory eventually reaches the $\alpha = 0$ axis when all clauses are eliminated. Notice that the end point is not S, but the lower left corner of the phase diagram.

As a conclusion, our work shows that, in the $\alpha_L < \alpha < \alpha_C$ range, the complexity of solving 3-SAT is related to the existence of a critical point of 2+p-SAT. The right part of the 2+p-SAT critical line, comprised between T and the threshold point of 3-SAT, can be determined experimentally as the locus of the highest backtracking points in 3-SAT solving search trees, when the starting ratio α_0 spans the interval $\alpha_L \leq \alpha_0 \leq \alpha_C$.

7 Complexity of 2+p-SAT solving and relationship with static

7.1 Complexity diagram

We have analyzed in the previous sections the computational complexity of 3-SAT solving. The analysis may extended to any 2+p-SAT instance, with the results shown in Figure 15.

In addition to the three regimes unveiled in the study of 3-SAT[3], a new complexity region appears, on the left side of the line $\alpha = 1/(1-p)$, referred to as "weak delocalization" line. To the right (respectively left) of the weak delocalization line, the second largest eigenvector of the branching matrix K is localized (resp. delocalized), see Appendix A. When solving a 3-SAT instance, or more generally a 2+p-SAT instance with parameters $p_0, \alpha_0 \leq 1/(1-p_0)$, the size of the search tree is exponential in N when the weak delocalization line is crossed by the tree trajectory (Fig. 4). Thus, the contribution to the average flow of unit-clauses coming from the second largest eigenvector is exponentially damped by the largest eigenvector contribution, and no contradiction arises until the halt, strong delocalization line is hit.

If one now desires to solve a 2+p-SAT instance whose representative point p_0, α_0 lies to the left of the weak delocalization curve, the delocalization of the second largest eigenvector stops immediately the growth of the tree, before the distribution of 1-clauses could reach equilibrium, see discussion of Section 5.2.4. Therefore, in the range of parameters $p_0, \alpha_0 \geq 1/(1-p_0)$, proving unsatisfiability does not require an exponentially large computational effort.

[3] Though differential equations (8) depend on t when written in terms of c_2, c_3, they are Markovian if rewritten in term of the variables p, α. Therefore, the locus of the 2+p-SAT instances points, p_0, α_0, giving rise to trajectories touching the threshold line in T, simply coincides with the 3-SAT trajectory starting from $\alpha_0 = \alpha_L$.

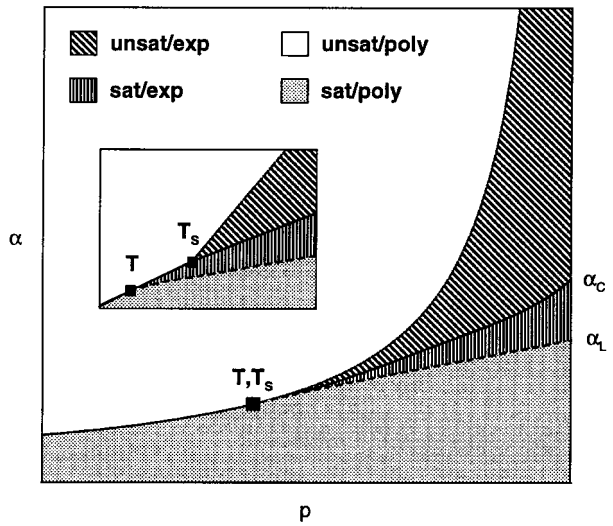

Fig. 15. Four different regions coexist in the phase diagram of the 2+p-SAT model, according to whether complexity is polynomial or exponential, and formulae are sat or unsat. Borderlines are (from top to down): $\alpha = 1/(1-p)$ (dotted line), $\alpha_C(p)$ (full line), and the branch trajectory (dashed line), starting in $(1, \alpha_L)$ and ending at point T tangentially to the threshold line. The tricritical point T_S, with coordinates $p_S \simeq 0.41, \alpha_S = 1/(1-p_S)$, separates second from first order critical point on the threshold line, and lies very close to T. Inset: schematic blow-up of the T, T_S region (same symbols as in the main picture).

7.2 Polynomial/exponential crossover and the tricritical point

The inset of Figure 15 show a schematic blow up of the neighborhood of T and T_S, where all complexity regions meet. From the above discussion, the complexity of solving critically constrained 2+p-SAT instances is polynomial up to p_S, and exponential above, in agreement with previous claims based on numerical investigations [13]. In the range $2/5 < p_0 < p_S$, computational complexity exhibits a somewhat unusual behavior as a function of α. The peak of hardness is indeed not located at criticality (where the scaling of complexity is only polynomial), but slightly below threshold, where complexity is exponential. Unfortunately, the narrowness of the region shown in the inset of Figure 15 seems to forbid to check this statement through experiments. Let us recall that, though replica calculations provide $p_S \simeq 0.41$, no rigorous proof of the inequality $p_S > 2/5$ is available, leaving open the possibility that T and T_S could actually coincide.

To end with, let us stress that T, conversely to T_S, depends *a priori* on the splitting heuristic. Nevertheless, the location of T seems to be more insensitive to the choice of the heuristics than branch trajectories. For instance, the UC and GUC heuristics both lead to the same tangential hit point T, while the starting ratios of the corresponding

trajectories, $\alpha_L = 8/3$ and $\alpha_L \simeq 3.003$, differ. Understanding this relative robustness, and the surprising closeness of T and T_S, would be interesting [31,32].

8 Conclusion and perspectives

In this paper we have analyzed the action of a search algorithm, the DPLL procedure, on random 3-SAT instances to derive the typical complexity as a function of the size (number of variables) N of the instance and the number α of clauses per variables. The easy, polynomial in N, as well as the hard, exponential in N, regimes have been investigated. We have measured, through numerical simulations, the size and the structure of the search tree by computing the number of nodes, the distribution of branch lengths, and the highest backtracking point. From a theoretical point of view, we have analyzed the dynamical evolution of a randomly drawn 3-SAT instance under the action of DPLL. The random 3-SAT statistical ensemble, described by a single parameter α, is not stable under the action of DPLL. Another variable p, the fraction of length three clauses, has to be considered to account for the later evolution of the instance[4]. Parameters p and α are the coordinates of the phase diagram of Figure 4. The dynamical evolution of the instance is itself of stochastic nature, due to the random choices made by the splitting rule. We can follow the ensemble evolution in 'time', that is, the number of variables assigned by DPLL, and represent this evolution by a trajectory in the phase diagram of Figure 4. For 3-SAT instances, located on the $p = 1$ axis, we show that there are three different behaviors, depending on the starting ratio α. In the low sat phase, $\alpha < \alpha_L \simeq 3.003$, trajectories are always confined in the sat region of the phase diagram. As a consequence, the search tree reduces essentially to a simple branch, and the complexity scales linearly with N. On the opposite, in the unsat phase, the algorithm has to build a complete search tree, with all branches ending with a contradiction, to prove unsatisfiability. We have imagined a tree growth process that reproduces faithfully the DPLL rules for assigning a new literal on a branch, but in which all branches evolve in parallel, not in the real and sequential way. We have derived the partial differential equation describing the stochastic growth of the search tree. The tree trajectory plotted on phase diagram of Figure 4 represents the evolution of the instance parameters p, α for typical, statistically dominant branches in the tree. When the trajectory hits the halt line, contradictions prevent the tree from further growing. Computational complexity is, to exponential order, equal to the number of typical branches. Last, in the upper sat phase $\alpha_L < \alpha < \alpha_c$, the trajectory intersects the critical line $\alpha_C(p)$ in some point G shown in Figure 4, and enters the unsat phase of 2+p-SAT instances. Below G, a complete refutation subtree has to be built. The full search tree turns out to be a mixture of a single branch and some (not exponentially numerous in N) complete subtrees (Fig. 14). The exponentially dominant contribution to the complexity is simply the size of the subtree that can be computed analyzing the growth process starting from G.

Statistical physics tools can be useful to study the solving complexity of branch and bound algorithms [2,4] applied to hard combinatorial optimization or decision problems. The phase diagram of Figure 4 affords an accurate understanding of the probabilistic complexity of DPLL variants on random instances. This view may reveal the nature of the complexity of search algorithms for SAT and related NP-complete problems. In the sat phase, branch trajectories are related to polynomial time computations while in the unsat region, tree trajectories lead to exponential calculations. Depending on the starting point (ratio α of the 3-SAT instance), one or a mixture of these behaviors is observed. A recent study of the random vertex cover problem [33] has shown that our approach can be successfully applied to other decision problems.

Figure 4 furthermore gives some insights to improve the search algorithm. In the unsat region, trajectories must be as horizontal as possible (to minimize their length) but resolution is necessarily exponential [28]. In the sat domain, heuristics making trajectories steeper could avoid the critical line $\alpha_C(p)$ and solve 3-SAT polynomially up to threshold.

Fluctuations of complexity are another important issue that would deserve further studies. The numerical experiments reported in Figure 5 show that the annealed complexity, that is, the average solving time required by DPLL, agrees well with the typical complexity in the unsat phase but discrepancies appear in the upper sat phase. It comes as no surprise that our analytical framework, designed to calculate the annealed complexity, provides accurate results in the unsat regime. We were also able to get rid of the fluctuations, and to calculate the typical complexity in the upper sat phase of 3-SAT from the annealed complexity of critical 2+p-SAT (Tab. 4). This suggests that fluctuations may originate from atypical points G in the mixed structure of the search tree unveiled in Section 6. Such atypical points G, coming from the $1/\sqrt{N}$ finite size fluctuations of the branch trajectory[5], lead to exponentially large fluctuations of the complexity (Fig. 5B).

It would be rewarding to achieve a better theoretical understanding of such fluctuations, and especially of fluctuations of solving times from run to run of DPLL procedure on a single instance [10]. Practitioners of hard problem solving have reached empirical evidence that exploiting in a cunning way the tails of the complexity

[4] This situation is reminiscent of what happens in real–space renormalization, *e.g.* decimation. New couplings, absent in the initial Hamiltonian are generated that must be taken into account. The renormalization flow takes place in the smallest coupling space, stable under the decimation procedure, that includes the original Hamiltonian as a point (Leo Kadanoff, *private communication*).

[5] Fluctuations also come from the finite width $W \sim N^{1-1/\nu}$ of the critical 2+p-SAT line. Recently derived lower bounds on the critical exponent ν (≥ 2 [34]) reveal that finite size effects could be larger than $O(1/\sqrt{N})$; for 2-SAT indeed, $\nu = 3$ and relative fluctuations scale as $W/N \sim N^{-2/3}$.

distribution may allow a drastic improvement of performances [35]. Suppose one is given one hour CPU time to solve one instance which, in 99% of cases, require 10 hours of calculation, and with probability 1%, ten seconds only. Then, one could proceed by running the algorithm for eleven seconds, stop it if the instance has not been solved yet, start again hundreds of time if necessary till the completion of the task. Investigating whether such a procedure could be used to treat successfully huge 3-SAT instances would be very interesting.

We are grateful to J. Franco for his precious encouragements and helpful discussions. S.C. was partly funded by a Borsa di Perfezionamento of Università di Roma "La Sapienza" and benefited from A. della Riccia grant. R.M. is supported in part by the MRSEC Program of the NSF under Award number DMR-9808595, and by the ACI Jeunes Chercheurs "Algorithmes d'optimisation et systèmes désordonnés quantiques" from the French Ministry of Research.

Appendix A: Largest eigenvalues and eigenvectors of the effective branching matrix

In this appendix, the largest eigenvalue of the effective branching matrix (43) is computed. We start by multiplying both sides of the eigenvalue equation, obtained by applying the matrix $\tilde{K}_{q_2}$ onto the eigenvector $\tilde{v}_q(C_1')$, by x^{C_1}. Then, we sum over C_1 and obtain the following equation for the eigenvectors generating functions $\tilde{V}_q(x)$,

$$\tilde{V}_q(x) = \tilde{V}_q(0)\frac{L(x)\ N(x)}{\Lambda_q - L(x)}\,. \tag{A.1}$$

where

$$N(x) = \mathrm{e}^{-y_2}\ (x + x^2) - 1, \quad L(x) = \frac{1}{x}\ \exp\left(\frac{m_2}{2}\,\mathrm{e}^{-y_2}\,x\right), \tag{A.2}$$

and

$$\Lambda_q = \tilde{\lambda}_{q,q_2} \exp\left\{m_2\ \left(1 - \frac{\mathrm{e}^{-y_2}}{2}\right)\right\}, \tag{A.3}$$

with $y_2 = -\mathrm{i}q_2$. From Section 5, q_2 is purely imaginary at saddle-point, and it is therefore convenient to manipulate the real-valued number y_2.

A.1 Zeroes and poles of $\tilde{V}_q$

$N(x)$ has two zeroes $x^\dagger < 0 < x^*$ that are functions of y_2 solely and given by

$$x^\dagger = \frac{1}{2}\left(-1 - \sqrt{1 + 4\mathrm{e}^{y_2}}\right)$$
$$x^* = \frac{1}{2}\left(-1 + \sqrt{1 + 4\mathrm{e}^{y_2}}\right). \tag{A.4}$$

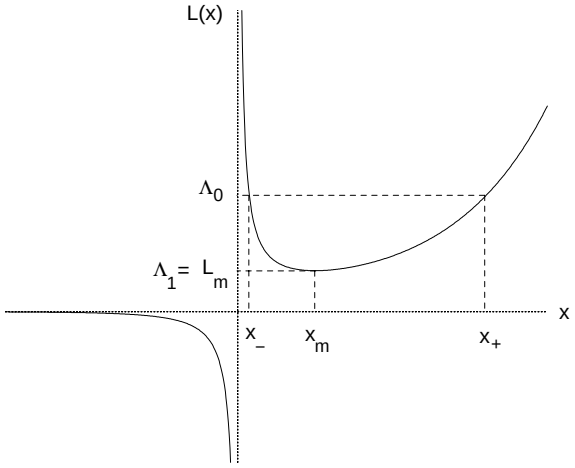

Fig. 16. Sketch of the function $L(x)$ appearing in the denominator of the eigenvector generating function $\tilde{V}_q(x)$. $L(x)$ is positive (resp. negative) for positive (resp. negative) arguments x. The local positive minimum is located at $x_\mathrm{m} = 1/\gamma_2, L_\mathrm{m} = e\,\gamma_2$. The height of the minimum, L_m, is equal to the edge Λ_1 of the (excited states) continuous spectrum. For $\Lambda > L_\mathrm{m}$, the equation $L(x) = \Lambda$ has two roots x_-, x_+ such that $x_- < x_\mathrm{m} < x_+$. When x_- coincides with the positive zero x^* of the numerator $N(x)$, the maximal eigenvalue Λ_0 is obtained (Appendix A).

$N(x)$ is negative when its argument x lies between the zeroes, positive otherwise. The function $L(x)$ is plotted Figure 16. The positive local minimum of $L(x)$ is located at $x_\mathrm{m} = 2\,\mathrm{e}^{y_2}/m_2, L_\mathrm{m} = m_2\,\mathrm{e}^{1-y_2}/2$. The number of poles of $\tilde{V}_q(x)$ can be inferred from Figure 16.

- If $\Lambda_q < 0$, there is a single negative pole.
- If $0 \leq \Lambda_q < L_\mathrm{m}$, there is no pole.
- If $\Lambda_q > L_\mathrm{m}$, there are two positive poles x_-, x_+ with $x_- < x_\mathrm{m} < x_+$ that coalesce when $\Lambda_q = L_\mathrm{m}$.

A.2 Largest eigenvalue and eigenvector

Consider the largest eigenvector $\tilde{v}_0(C_1)$ and the associated eigenvalue $\Lambda_0 > 0$. The ratios

$$\mu(C_1) = \frac{v_0(C_1)}{\sum_{C_1'=0}^{\infty} v_0(C_1')} \tag{A.5}$$

define the probability that the number of unit-clauses be equal to C_1 at a certain stage of the search. Consequently, as long as no contradiction occurs, we expect all the ratios to be positive and decrease quickly with C_1. The generating function $\tilde{V}_q(x)$ must have a finite radius of convergence R, and be positive in the range $0 \leq x < R$. Note that the radius of convergence coincides with a pole of $\tilde{V}_q$.

The asymptotic behavior of $\tilde{v}_0(C_1)$ is simply given by the radius of convergence,

$$\tilde{v}_0(C_1) \sim \left(\frac{1}{R}\right)^{C_1}, \tag{A.6}$$

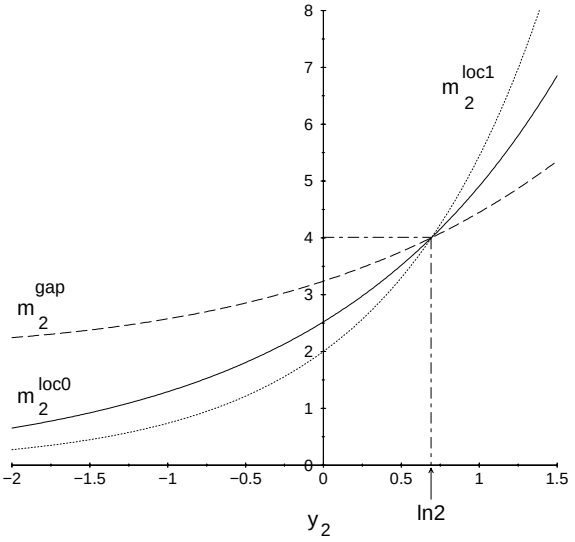

Fig. 17. Critical curves of $m_2 = 2c_2/(1-t)$ as a function of the parameter y_2. From bottom to top (left side): delocalization threshold $m_2^{\mathrm{loc.1}}$ for the second largest eigenvector (dotted line), delocalization threshold $m_2^{\mathrm{loc.0}}$ for the largest eigenvector (full line), and zero gap curve m_2^{gap} (dashed line). All curve meet in $y_2 = \ln 2, m_2 = 4$. For $y_2 < \ln 2$ and small m_2, the largest eigenvector is separated from a continuum of excited states by a finite gap, and undergoes a delocalization condition when m_2 reaches $m_2^{\mathrm{loc.0}}$. For $y_2 > \ln 2$, the largest eigenvector merges with the continuum spectrum when $m_2 \geq m_2^{\mathrm{gap}}$, and gets delocalized on the critical line m_2^{loc1}.

up to non-exponential terms. New branches are all the more frequent that many splittings are necessary and unit-clauses are not numerous, *i.e.* $\tilde{v}_0(C_1)$ decreases as fast as possible. From the discussion of Section A.1, the radius of convergence is maximal if $R = x_+$. To avoid the presence of another pole to $\tilde{V}_q(x)$ in x_-, the zero of the numerator function $N(x)$ must fulfills $x^* = x_-$. We check that $\tilde{V}_q(x)$ is positive for $0 \leq x < R$. The corresponding eigenvalue is $\Lambda_0 = L(x_+)$, see equation (45). In Section A.3, we explain that this theoretical value of Λ_0 has to be modified in a small region of the (y_2, m_2) plane.

The eigenvector $\tilde{v}_0$ undergoes a delocalization transition on the critical line $x_+(y_2, m_2) = 1$ or, equivalently, $L(x^*) = L(1)$. At fixed y_2, the eigenvector is localized provided that parameter m_2 is smaller than a critical value

$$m_2^{\mathrm{loc.0}}(y_2) = 2\ \mathrm{e}^{y_2}\ \frac{\ln x^*(y_2)}{x^*(y_2) - 1}\cdot \tag{A.7}$$

The corresponding curve is shown in Figure 17.

A.3 Excited state and spectral gap

As $\tilde{K}_{q_2}$ is not a symmetric matrix, complex eigenvalues may be found in the spectrum. We have performed numerical investigations by diagonalizing upper left corners of $\tilde{K}_{q_2}$ of larger and larger sizes. From a technical point of view, we define the $U \times U$ matrix $\tilde{K}_{q_2}^{(U)}(C_1, C_1') = \tilde{K}_{q_2}(C_1, C_1')$ for $0 \leq C_1, C_1' \leq U$. Numerics show that complex eigenvalues are of small modulus with respect to real-valued eigenvalues.

If $y_2 < \ln 2$, the largest eigenvalue $\Lambda_0^{(U)}$ of $\tilde{K}_{q_2}^{(U)}$ converges very quickly to the theoretical value $L(x^+)$ as U increases (with more than five correct digits when $U \geq 30$). At small values of m_2, the second largest eigenvalue (in absolute value) is negative. Let us call it $\Lambda_\dagger$. The associated eigenvector $\tilde{v}_\dagger(C_1)$ is localized; all components of $\tilde{v}_\dagger$ have the same sign except $\tilde{v}_\dagger(0)$. The value of $\Lambda_\dagger$ may be computed along the lines of Section A.2,

$$\Lambda_\dagger = L(x^\dagger). \tag{A.8}$$

Indeed, $x^\dagger < 0$ implies from Figure 16 that $L(x^\dagger) < 0$. As m_2 increases (at fixed y_2), $\Lambda_\dagger$ becomes smaller (in modulus) than the second largest positive eigenvalue Λ_1. Λ_1 is followed by a set of positive eigenvalues $\Lambda_2 > \Lambda_3 > \ldots$. Successive Λ_q ($q \geq 1$) gets closer and closer as U increases, to form a continuous spectrum in the $U \to \infty$ limit.

The eigenvectors $\tilde{v}_q$ ($q = 1, 2, 3, \ldots$) have real-valued components of periodically changing signs. The corresponding generating functions have therefore complex-valued radii of convergences, and $V_q(x)$ does not diverge for real positive arguments x. Consequently, the edge of the continuous spectrum Λ_1 is given by

$$\Lambda_1 = L_{\mathrm{m}}. \tag{A.9}$$

The above theoretical prediction is in excellent agreement with the large U extrapolation of the numerical values of Λ_1. Repeating the discussion of Section A.2, the first excited state $\tilde{v}_1$ becomes delocalized when $x_{\mathrm{m}} = 1$, that is, when m_2 exceeds the value

$$m_2^{\mathrm{loc.1}}(y_2) = 2\ \mathrm{e}^{y_2}. \tag{A.10}$$

The corresponding curve is shown in Figure 17.

The gap between Λ_0 and Λ_1 will be strictly positive as long as $x^* < x_{\mathrm{m}}$. This defines another upper value for m_2,

$$m_2^{\mathrm{gap}}(y_2) = 2\ \mathrm{e}^{y_2} \frac{1}{x^*(y_2)}, \tag{A.11}$$

beyond which the largest eigenvalue coincides with the top Λ_1 of the continuous spectrum. As can be seen from Figure 17, Λ_0 coincides with $L(x^+)$ in the region $m_2 < m_2^{\mathrm{loc0}}$ as long as the largest eigenvalue is separated from the continuous spectrum by a finite gap. Therefore, in the region $(y_2 > \ln 2, m_2^{\mathrm{gap}} < m_2 < m_2^{\mathrm{loc0}})$, the largest eigenvalue merges with Λ_1, and is given by equation (A.9).

When m_2 crosses the critical line m_2^{loc0}, the largest eigenvector gets delocalized and the average number of unit-clauses flows to infinity. As a result of the avalanche of unitary clauses, contradictions necessarily occur and the growth of the tree stops. Notice that, as far as the total number of branches is concerned, we shall be mostly concerned by the $y_2 = 0$ axis. The critical values of interest are in this case: $m_2^{\mathrm{loc1}} = 2$, $m_2^{\mathrm{loc0}} = (3+\sqrt{5})\ln[(1+\sqrt{5})/2] \simeq 2.5197$ and $m_2^{\mathrm{gap}} = 1+\sqrt{5} \simeq 3.2361$.

Appendix B: Quadratic approximation for the growth partial differential equation

The partial differential equation for the growth of the search tree may be written in terms of the Legendre transform $\varphi(y_2, y_3, t)$ of the logarithm of the number of branches as

$$\frac{\partial\varphi}{\partial t}(y_2,y_3,t) = g_1(y_2) - \frac{2}{1-t}g_2(y_2)\frac{\partial\varphi}{\partial y_2} - \frac{3}{1-t}g_3(y_2,y_3)\frac{\partial\varphi}{\partial y_3}, \quad \text{(B.1)}$$

with

$$\begin{aligned} g_1(y_2) &= -y_2 + \ln\left[\frac{1}{2}\left(1+\sqrt{1+4\mathrm{e}^{2y_2}}\right)\right] \\ g_2(y_2) &= 1 - \frac{1}{2}\left[1+\frac{1}{2}\left(-\mathrm{e}^{-y_2}+\sqrt{4+\mathrm{e}^{-2y_2}}\right)\right] \\ g_3(y_2,y_3) &= 1 - \frac{1}{2}e^{-y_3}\left(1+\mathrm{e}^{y_2}\right). \end{aligned} \quad \text{(B.2)}$$

B.1 Linear approximation

At the first order in y_2, y_3, we replace the functions g appearing in (B.1) with their linearized counterparts,

$$\begin{aligned} g_1^{(1)}(y_2) &= \ln\left(\frac{1+\sqrt{5}}{2}\right) - \left(\frac{5+\sqrt{5}}{10}\right)\, y_2 \\ g_2^{(1)}(y_2) &= \frac{3-\sqrt{5}}{4} + \left(\frac{5+3\sqrt{5}}{4}\right) y_2 \\ g_3^{(1)}(y_2,y_3) &= y_3 - \frac{1}{2}y_2, \end{aligned} \quad \text{(B.3)}$$

and solve the corresponding partial differential equation. The solution, called $\varphi^{(1)}(y_2, y_3, t)$ is given in equations (53) and (54).

B.2 Quadratic approximation

At the second order in y_2, y_3, we consider the quadratic corrections to the g functions,

$$\begin{aligned} g_1^{(2)}(y_2) &= \frac{\sqrt{5}}{50}(y_2)^2 \\ g_2^{(2)}(y_2) &= -\left(\frac{25+11\sqrt{5}}{200}\right)(y_2)^2 \\ g_3^{(2)}(y_2,y_3) &= -\frac{1}{2}(y_3)^2 + \frac{1}{2}y_3y_2 - \frac{1}{4}(y_2)^2, \end{aligned} \quad \text{(B.4)}$$

and look for a solution of (B.1) of the form $\varphi = \varphi^{(1)} + \varphi^{(2)}$. Neglecting higher order terms in $g^{(2)}\varphi^{(2)}$, we end with

$$\frac{\partial\varphi^{(2)}}{\partial t}(y_2,y_3,t) = G^{(2)}(y_2,y_3) - \frac{2}{1-t}g_2^{(1)}(y_2)\frac{\partial\varphi^{(2)}}{\partial y_2} - \frac{3}{1-t}g_3^{(1)}(y_2,y_3)\frac{\partial\varphi^{(2)}}{\partial y_3}, \quad \text{(B.5)}$$

where

$$G^{(2)}(y_2,y_3) = g_1^{(2)}(y_2) - \frac{3}{1-t}g_2^{(2)}(y_2)\frac{\partial\varphi^{(1)}}{\partial y_2} - \frac{2}{1-t}g_3^{(2)}(y_2,y_3)\frac{\partial\varphi^{(1)}}{\partial y_3}\cdot \quad \text{(B.6)}$$

A particular solution of (B.5) may be found under the form

$$\varphi^{(2)}_{\text{part.}}(z_2,z_3,t) = a_0(t) + a_{21}(t)y_2 + a_{22}(t)(y_2)^2 + a_{33}(t)(y_3)^2, \quad \text{(B.7)}$$

where the a's are linear combinations of $1-t$, $(1-t)^3$ and $(1-t)^\mu$ with $\mu = (5+3\sqrt{5})/10$.

The general solution of the homogeneous version of (B.5) reads

$$\varphi^{(2)}_{\text{hom.}}(z_2,z_3,t) = \Phi\left[(1-t)^3\left(y_3 - \frac{75+9\sqrt{5}}{116}y_2 - \frac{15-4\sqrt{5}}{58}\right)\times(1-t)^\mu\left(y_2+\frac{7\sqrt{5}-15}{2}\right)\right], \quad \text{(B.8)}$$

where Φ is a differentiable function of two arguments u, v. Assuming that $\Phi[u;v]$ is a quadratic form in u and v, we fix its six coefficients through the initial condition (at time $t = 0$),

$$\varphi^{(2)}(y_2,y_3,0) = \varphi^{(2)}_{\text{part.}}(y_2,y_3,0) + \varphi^{(2)}_{\text{hom.}}(y_2,y_3,0) = 0\ . \quad \text{(B.9)}$$

The resulting expression for $\varphi^{(2)}$ is too long to be given here but can be obtained safely by using an algebraic computation software, *e.g.* Mathematica.

References

1. *Scale invariance, interfaces, and non–equilibrium dynamics* edited by A. McKane, M. Droz, J. Vannimenus, D. Wolf, Nato Asi Series B: Physics, Vol. 344, (Plenum Press, New-York, 1995).
2. M. Garey, D.S. Johnson, *Computers and Intractability; A guide to the theory of NP–completeness* (W.H. Freeman and Co., San Francisco, 1979).
3. D.E. Knuth, *The art of computer programming*, Vol. 3, *Sorting and Searching* (Addison-Wesley, Reading, Massachusetts, 1998).
4. B. Hayes, Ame. Sci. **85**, 108 (1996), `http://www.amsci.org/amsci/issues/Comsci97/compsci9703.html`.
5. M. Davis, H. Putnam, J. Assoc. Comput. Mach. **7**, 201 (1960); M. Davis, G. Logemann, D. Loveland, *A machine program for theorem proving, Communications of the ACM* **5**, 394 (1962).

6. J. Crawford, L. Auton, *Proc. 11th Natl. Conference on Artificial Intelligence AAAI-93,* (The AAAI Press/MIT Press, Cambridge, MA, 1993), p. 21.
7. P. Cheeseman, B. Kanefsky, W.M. Taylor, in *Proc. of IJCAI-91,*, edited by J. Mylopoulos, R. Reiter (Morgan Kaufmann, San Mateo, CA, 1991), p. 331.
8. D. Mitchell, B. Selman, H. Levesque, *Proc. of the Tenth Natl. Conf. on Artificial Intelligence AAAI-92* (The AAAI Press/MIT Press, Cambridge, MA, 1992), p. 440.
9. *Frontiers in problem solving: phase transitions and complexity,* in *Artificial Intelligence*, Vol. 81 (I & II), edited by T. Hogg, B.A. Huberman, C. Williams (1996).
10. S. Kirkpatrick, B. Selman, Science **264**, 12971 (1994).
11. E. Friedgut, J. Am. Math. Soc. **12**, 1017 (1999).
12. R. Monasson, R. Zecchina, Phys. Rev. E **56**, 1357 (1997).
13. R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, L. Troyansky, Nature **400**, 133 (1999).
14. J.A. Hertz, A.S. Krogh, R.G. Palmer, *Introduction to the theory of neural computation*, *SFI Studies in the Sciences of Complexity* (Addison-Wesley, 1991).
15. *Dynamics of Structural Change in Liquids*, edited by C.A. Angell, M. Goldstein (N.Y. Academy of Sciences, 1986).
16. B. Selman, H.J. Levesque, D. Mitchell, in *Proc. AAAI-92, San Jose, California, 1992*, p. 440.
17. S. Cocco, R. Monasson, Phys. Rev. Lett. **86**, 1654 (2001).
18. M.T. Chao, J. Franco, Inf. Sci. **51**, 289 (1990).
19. A. Goerdt, J. Comput. System Sci. **53**, 469 (1996); V. Chvàtal, B. Reed, in *Proc. 33rd IEEE Symp. on Foundations of Computer Science* (IEEE Computer Soc. Press, New York, 1992), p. 620.
20. A. Frieze, S. Suen, J. Algorithms **20**, 312 (1996).
21. A. Kamath, R. Motwani, K. Palem, P. Spirakis, Random Structures and Algorithms **7**, 59 (1995); O. Dubois, Theor. Comp. Sci. **81**, 49 (1991).
22. S.A. Cook, D.G. Mitchell, in *Satisfiability Problem: Theory and Applications*, in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 35, edited by Du, Gu, Pardalos (1997).
23. R. Monasson, R. Zecchina, J. Phys. A **31**, 9209 (1998).
24. G. Biroli, R. Monasson, M. Weigt, Eur. Phys. J. B **14**, 551 (2000).
25. D. Achlioptas, L. Kirousis, E. Kranakis, D. Krizanc, *Rigorous results for random (2+p)-SAT*, Special issue of Theor. Comp. Sci. on *NP-hardness and Phase transitions* (to be published).
26. B. Selman, S. Kirkpatrick, Artificial Intelligence **81**, 273 (1996).
27. I.P. Gent, T. Walsh, Artificial Intelligence **70**, 335 (1994).
28. V. Chvàtal, E. Szmeredi, J. Assoc. Comput. Mach. **35**, 759 (1988).
29. P. Beame, R. Karp, T. Pitassi, M. Saks, *ACM Symp. on Theory of Computing STOC98* (Assoc. Comput. Mach., New York, 1998), p. 561.
30. O. Kullmann, Theor. Comp. Sci. **223**, 1 (1999).
31. S. Cocco, R. Monasson, *Statistical physics analysis of the backtrack resolution of random 3-SAT instances*, in *Proceedings of LICS 2001 Workshop on Theory and Applications of Satisfiability Testing (SAT 2001), Boston, MA, June 2001.* Available at: `http://www.elsevier.nl/locate/endm/`.
32. D. Achlioptas, *Lower bounds for random 3-SAT via differential equations*, Special issue of Theor. Comp. Sci. on *NP-hardness and Phase transitions* (to be published).
33. A. Hartmann, M. Weigt, Phys. Rev. Lett. **86**, 1658 (2001).
34. D.B. Wilson, *The empirical values of the critical k-SAT exponents are wrong*, `arXiv:math/0005136`.
35. C.P. Gomes, B. Selman, N. Crato, H. Kautz, J. Automated Reasoning **24**, 67 (2000).