Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 320 (2004) 345–372
Theoretical Computer Science
www.elsevier.com/locate/tcs

Heuristic average-case analysis of the backtrack resolution of random 3-satisfiability instances

Simona Cocco[a], Rémi Monasson[b,c,*]

[a] *CNRS-Laboratoire de Dynamique des Fluides Complexes, 3 rue de l'université, 67000 Strasbourg, France*
[b] *CNRS-Laboratoire de Physique Théorique de l'ENS, 24 rue Lhomond, 75005 Paris, France*
[c] *CNRS-Laboratoire de Physique Théorique, 3 rue de l'université, 67000 Strasbourg, France*

Received 12 January 2003; received in revised form 22 March 2003; accepted 25 February 2004
Communicated by G. Ausiello

Abstract

An analysis of the average-case complexity of solving random 3-Satisfiability (SAT) instances with backtrack algorithms is presented. We first interpret previous rigorous works in a unifying framework based on the statistical physics notions of dynamical trajectories, phase diagram and growth process. It is argued that, under the action of the Davis–Putnam–Loveland–Logemann (DPLL) algorithm, 3-SAT instances are turned into $2+p$-SAT instances whose characteristic parameters (ratio α of clauses per variable, fraction p of 3-clauses) can be followed during the operation, and define resolution trajectories. Depending on the location of trajectories in the phase diagram of the 2+p-SAT model, easy (polynomial) or hard (exponential) resolutions are generated. Three regimes are identified, depending on the ratio α of the 3-SAT instance to be solved. Lower satisfiable (sat) phase: for small ratios, DPLL almost surely finds a solution in a time growing linearly with the number N of variables. Upper sat phase: for intermediate ratios, instances are almost surely satisfiable but finding a solution requires exponential time ($\sim 2^{N\omega}$ with $\omega > 0$) with high probability. Unsat phase: for large ratios, there is almost always no solution and proofs of refutation are exponential. An analysis of the growth of the search tree in both upper sat and unsat regimes is presented, and allows us to estimate ω as a function of α. This analysis is based on an exact relationship between the average size of the search tree and the powers of the evolution operator encoding the elementary steps of the search heuristic.

Keywords: Satisfiability; Analysis of algorithms; Backtrack

* Corresponding author. CNRS-Laboratoire de Physique Théorique de l'ENS, 24 rue Lhomond, 75005 Paris, France.
E-mail addresses: cocco@ldfc.u-strasbg.fr (S. Cocco), monasson@lpt.ens.fr (R. Monasson).
URLs: http://ludfc39.u-strasbg.fr/cocco/index.html, http://www.lptens.fr/∼monasson

0304-3975/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.02.034

(1) Choose a variable and its value (T,F) according to some heuristic rule **(Split);**

(2) Analyze the implications of the choice on all the clauses :

a: If all clauses are satisfied, then stop: a solution is found,
b: If a contradiction appears, negate the last chosen variable and go to 2 **(Backtracking),**
If all previously chosen variables have already been negated once, then stop: unsatisfiability is proven,
c: if there is at least one clause with one variable, fix the variable to satisfy the clause and go to 2 **(Unit Propagation),**
d: Else go to 1.

Fig. 1. DPLL algorithm. When a variable has been chosen at step (1) e.g. $x = T$, at step (2) some clauses are satisfied e.g. $C = (x \text{ OR } y \text{ OR } z)$ and eliminated, other are reduced e.g. $C = (\text{not } x \text{ OR } y \text{ OR } z) \to C = (y \text{ OR } z)$. If some clauses include one variable only e.g. $C = y$, the corresponding variable is automatically fixed to satisfy the clause ($y = T$). This propagation (2c) is repeated up to the exhaustion of all unit clauses. Contradictions result from the presence of two opposite unit clauses e.g. $C = (y), C' = (\text{not } y)$. A solution is found when no clauses are left. The search process of DPLL is represented by a tree (Fig. 2) whose nodes correspond to (1), and edges to (2). Branch extremities are marked with contradictions C (2B, 2C), or by a solution S (2A, 2C).

1. Introduction

This paper focuses on the average complexity of solving random 3-SAT instances using backtrack algorithms. Being an NP-complete problem, 3-SAT is not thought to be solvable in an efficient way, i.e. in time growing at most polynomially with N. In practice, one therefore resorts to methods that need, a priori, exponentially large computational resources. One of these algorithms is the ubiquitous Davis–Putnam–Loveland–Logemann (DPLL) solving procedure [12,22]. DPLL is a complete search algorithm based on backtracking; its operation is briefly recalled in Fig. 1. The sequence of assignments of variables made by DPLL in the course of instance solving can be represented as a search tree, whose size Q (number of nodes) is a convenient measure of the hardness of resolution. Some examples of search trees are presented in Fig. 2.

In the past few years, much experimental and theoretical progress has been made on the probabilistic analysis of 3-SAT [14,19,24]. Distributions of random instances controlled by few parameters are particularly useful in shedding light on the onset of complexity. 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 [11,19,29,33] and theory [13,14,17] indicate that clauses can almost surely always (respectively never) be simultaneously satisfied if α is smaller (resp. larger) than a critical threshold $\alpha_C \simeq 4.3$ as soon as the numbers M of clauses and N of variables go to infinity at a fixed ratio α. This phase transition [31] is accompanied by a drastic peak in hardness at threshold [11,19,29]. The emerging pattern of complexity is as follows. At small ratios $\alpha < \alpha_L$, where α_L depends on the heuristic used by DPLL, instances are almost surely satisfiable (sat), see Franco [16] and

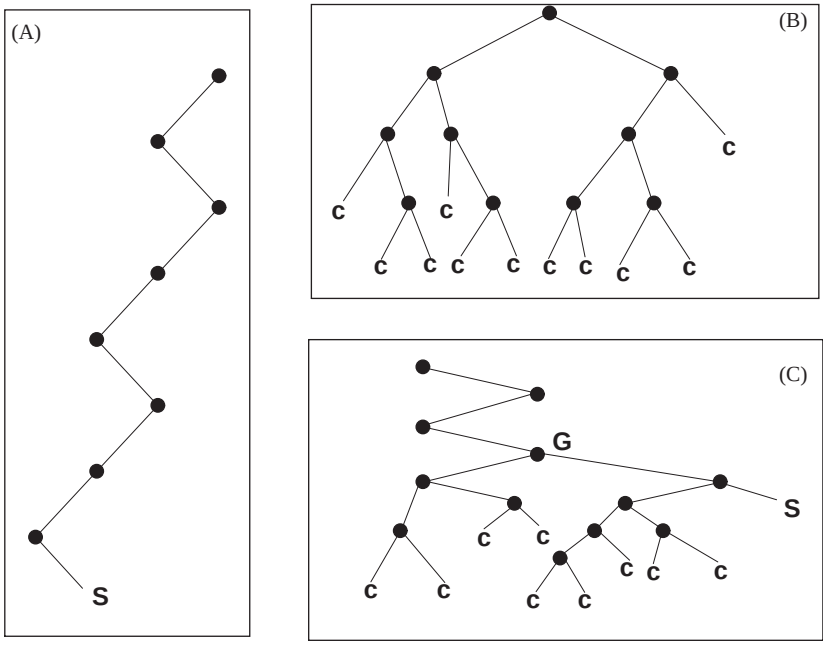

Fig. 2. Types of search trees generated by the DPLL solving procedure on random 3-SAT. **A.** *simple branch*: the algorithm finds easily a solution without ever backtracking. **B.** *dense tree:* in the absence of solution, DPLL builds a tree, including many branches ending with contradictory leaves, before stopping. **C.** *mixed case, branch+tree*: if many contradictions arise before reaching a solution, the resulting search tree can be decomposed into a single branch followed by a dense tree. G is the highest node in the tree reached by DPLL through backtracking.

Achlioptas [1] for recent reviews. The size Q of the associated search tree scales, with high probability, linearly with the number N of variables, and almost no backtracking is present [18] (Fig. 2A). Above the critical ratio, that is when $\alpha > \alpha_C$, instances are a.s. unsatisfiable (unsat) and proofs of refutation are obtained through massive backtracking (Fig. 2B), leading to an exponential hardness: $Q = 2^{N\omega}$ with $\omega > 0$ [7]. In the intermediate range, $\alpha_L < \alpha < \alpha_C$, finding a solution a.s. requires exponential effort ($\omega > 0$) [2,8,9].

The aim of this article is two-fold. Firstly, we propose a simple and intuitive framework to unify the above findings. This framework is presented in Section 2. It is based on the statistical physics notions of dynamical trajectories and phase diagram, and was, to some extent, implicitly contained in the pioneering analysis of search heuristics by Chao and Franco [5,6]. Secondly, we present in Section 3 a quantitative study of the growth of the search tree in the unsat regime. Such a study has been lacking so far due to the formidable difficulty in taking into account the effect of massive backtracking on the operation of DPLL. We first establish an exact relationship between the average size of the search tree and the powers of the evolution operator encoding the elementary steps of the search heuristic. This equivalence is then used (in a non-rigorous

way) to accurately estimate the logarithm ω of the average complexity Q as a function of α,

$$\omega(\alpha) = \lim_{N\to\infty} \frac{1}{N} \log_2 E_{(\alpha,N)}[Q], \tag{1}$$

where $E_{(N,\alpha)}$ denotes the expectation value for given N and α. The approach emphasizes the relevance of partial differential equations to analyse algorithms in presence of massive backtracking, as opposed to ordinary differential equations in the absence of the latter [1,34]. In Section 4, we focus upon the upper sat regime i.e. upon ratios $\alpha_L < \alpha < \alpha_C$. Combining the framework of Section 2 and the analysis of Section 3 we unveil the structure of the search tree (Fig. 2C) and calculate ω as a function of the ratio α of the 3-SAT instance to be solved.

For the sake of clarity and since the style of our approach may look unusual to the computer scientist reader, the status of the different calculations and results (experimental, exact, conjectured, approximate, etc.) are made explicit throughout the article.

2. Phase diagram and trajectories

2.1. The 2+p-SAT distribution and split heuristics

The action of DPLL on an instance of 3-SAT causes changes to the overall numbers of variables and clauses, and thus of the ratio α. Furthermore, DPLL reduces some 3- to 2-clauses. A mixed 2+p-SAT distribution, where p is the fraction of 3-clauses, can be used to model what remains of the input instance at a node of the search tree. Using experiments and methods from statistical mechanics [31], the threshold line $\alpha_C(p)$, separating sat from unsat phases, may be estimated with the results shown in Fig. 3. For $p \leqslant p_0 = 2/5$, i.e. to the left of point T, the threshold line is given by $\alpha_C(p) = 1/(1-p)$, as rigorously confirmed by Achlioptas et al. [3], and saturates the upper bound for the satisfaction of 2-clauses. Above p_0, no exact value for $\alpha_C(p)$ is known. Note that $\alpha_C \simeq 4.3$ corresponds to $p = 1$.

The phase diagram of 2+p-SAT is the natural space in which DPLL dynamic takes place. An input 3-SAT instance with ratio α shows up on the right vertical boundary of Fig. 3 as a point of coordinates $(p = 1, \alpha)$. Under the action of DPLL, the representative point moves aside from the 3-SAT axis and follows a trajectory. This trajectory obviously depends on the heuristic of split followed by DPLL (Fig. 1). Possible simple heuristics are [5,6],

- *Unit-clause* (*UC*): Randomly pick up a literal among a unit clause if any, or any unset variable otherwise.
- *Generalized unit-clause* (*GUC*): Randomly pick up a literal among the shortest available clauses.
- *Short clause with majority* (SC_1): Randomly pick up a literal among unit clauses if any; otherwise randomly pick up an unset variable v, count the numbers of occurrences $\ell, \bar{\ell}$ of v, $\bar{v}$ in 3-clauses, and choose v (resp. $\bar{v}$) if $\ell > \bar{\ell}$ (resp. $\ell < \bar{\ell}$). When $\ell = \bar{\ell}$, v and $\bar{v}$ are equally likely to be chosen.

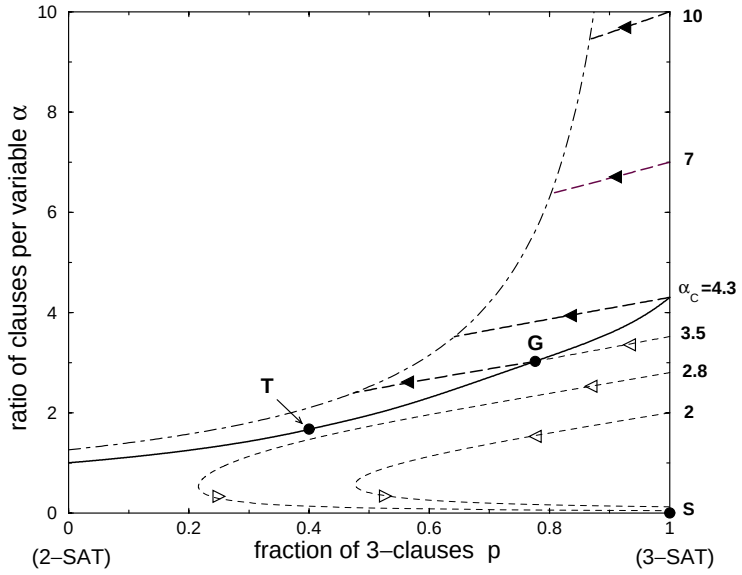

Fig. 3. Phase diagram of 2+p-SAT and dynamical trajectories of DPLL. The threshold line $\alpha_C(p)$ (bold full line) separates sat (lower part of the plane) from unsat (upper part) phases. Extremities lie on the vertical 2-SAT (left) and 3-SAT (right) axis at coordinates ($p=0, \alpha_C=1$) and ($p=1, \alpha_C \simeq 4.3$), respectively. Departure points for DPLL trajectories are located on the 3-SAT vertical axis and the corresponding values of α 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 $\alpha<\alpha_L$, branch trajectories remain confined in the sat phase, end in S of coordinates $(1,0)$, where a solution is found. At α_L ($\simeq 3.003$ for the GUC heuristic), the single branch trajectory hits tangentially the threshold line in T of coordinates $(2/5, 5/3)$. In the intermediate range $\alpha_L<\alpha<\alpha_C$, the branch trajectory intersects the threshold line at some point G (which depends on α). A dense tree then grows in the unsat phase, as happens when 3-SAT departure ratios are above threshold $\alpha>\alpha_C \simeq 4.3$. 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 $\alpha>\alpha_C$, or node G for $\alpha_L<\alpha<\alpha_C$. In the latter case, a solution can be reached from a new descending branch while, in the former case, unsatisfiability is proven, see Fig. 2.

Rigorous mathematical analysis, undertaken to provide rigorous bounds to the critical threshold α_C, have so far been restricted to the action of DPLL prior to any backtracking, that is, to the first descent of the algorithm in the search tree.[1] The corresponding search branch is drawn in Fig. 2A. These studies rely on the two following facts:

Firstly, the representative point of the instance treated by DPLL does not "leave" the 2+p-SAT phase diagram. In other words, the instance is, at any stage of the search process, uniformly distributed from the 2+p-SAT distribution conditioned to its clause-per-variable ratio α and fraction of 3-clauses p. This assumption is not true for

[1] The analysis of Frieze and Suen [18] however includes a very limited version of backtracking, see Section 2.2.

all heuristics of split, but holds for the above examples (UC, GUC, SC_1) [5]. Analysis of more sophisticated heuristics require to handle more complex instance distributions [26].

Secondly, the trajectory followed by an instance in the course of resolution is a stochastic object, due to the randomness of the instance and of the assignments done by DPLL. In the large size limit ($N \to \infty$), this trajectory gets concentrated around its average locus in the 2+p-SAT phase diagram. This concentration phenomenon results from general properties of Markov chains [1,34].

2.2. Trajectories associated to search branches

Let us briefly recall Chao and Franco [5] analysis of the average trajectory corresponding to the action of DPLL prior to backtracking. The ratio of clauses per variable of the 3-SAT instance to be solved will be denoted by α_0. The numbers of 2- and 3-clauses are initially equal to $C_2 = 0, C_3 = \alpha_0 N$ respectively. Under the action of DPLL, C_2 and C_3 follow a Markovian stochastic evolution process, as the depth T along the branch (number of assigned variables) increases. Both C_2 and C_3 are concentrated around their expectation values, the densities $c_j(t) = E[C_j(T = tN)]$ ($j = 2, 3$) of which obey a set of coupled ordinary differential equations (ODE) [1,5,6],

$$\frac{\mathrm{d}c_3}{\mathrm{d}t} = -\frac{3c_3}{1-t}, \quad \frac{\mathrm{d}c_2}{\mathrm{d}t} = \frac{3c_3}{2(1-t)} - \frac{2c_2}{1-t} - \rho_1(t)h(t), \tag{2}$$

where $\rho_1(t) = 1 - c_2(t)/(1-t)$ is the probability that DPLL fixes a variable at depth t (fraction of assigned variables) through unit-propagation. Function h depends upon the heuristic: $h_{\mathrm{UC}}(t) = 0$, $h_{\mathrm{GUC}}(t) = 1$ (if $\alpha_0 > 2/3$; for $\alpha_0 < 2/3$, see [6]), $h_{\mathrm{SC}_1}(t) = a\mathrm{e}^{-a}(I_0(a) + I_1(a))/2$ where $a \equiv 3c_3(t)/(1-t)$ and I_ℓ is the ℓth modified Bessel function. To obtain the single branch trajectory in the phase diagram of Fig. 3, we solve ODEs (2) with initial conditions $c_2(0) = 0, c_3(0) = \alpha_0$, and perform the change of variables

$$p(t) = \frac{c_3(t)}{c_2(t) + c_3(t)}, \quad \alpha(t) = \frac{c_2(t) + c_3(t)}{1-t}. \tag{3}$$

Results are shown for the GUC heuristics and starting ratios $\alpha_0 = 2$ and 2.8 in Fig. 3. The trajectory, indicated by a light dashed line, first heads to the left and then reverses to the right until reaching a point on the 3-SAT axis at a small ratio. 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 found.

Frieze and Suen [18] have shown that, for ratios $\alpha_0 < \alpha_{\mathrm{L}} \simeq 3.003$ (for the GUC heuristics), the full search tree essentially reduces to a single branch, and is thus entirely described by the ODEs (2). The amount of backtracking necessary to reach a solution is bounded from above by a power of $\log N$. The average size of the branch, Q, scales linearly with N with a multiplicative factor $\gamma(\alpha_0) = Q/N$ that can be calculated [9]. The boundary α_{L} of this easy sat region can be defined as the largest initial ratio α_0 such that the branch trajectory $(p(t), \alpha(t))$ issued from $(1, \alpha_0)$ never leaves the sat phase during DPLL action. In other words, the instance essentially keeps being sat

throughout the resolution process. We shall see in Section 4 this does not hold for sat instances with ratios $\alpha_L < \alpha_0 < \alpha_C$.

3. Analysis of the search tree growth in the unsat phase

In this section, we present an analysis of search trees corresponding to unsat instances, that is, in presence of massive backtracking. We first report results from numerical experiments, then expose our analytical approach to compute the complexity of resolution (size of search tree).

3.1. Numerical experiments

For ratios above threshold ($\alpha_0 > \alpha_C \simeq 4.3$), instances almost never have a solution but a considerable amount of backtracking is necessary before proving that clauses are incompatible. Fig. 2B shows a generic unsat, or refutation, tree. In contrast to the previous section, the sequence of points (p, α) attached to the nodes of the search tree do not arrange along a line any longer, but rather form a cloud with a finite extension in the phase diagram of Fig. 3. Examples of clouds are provided in Fig. 4.

The number of points in a cloud i.e. the size Q of its associated search tree grows exponentially with N [7]. It is thus convenient to define its logarithm ω through $Q = 2^{N\omega}$.

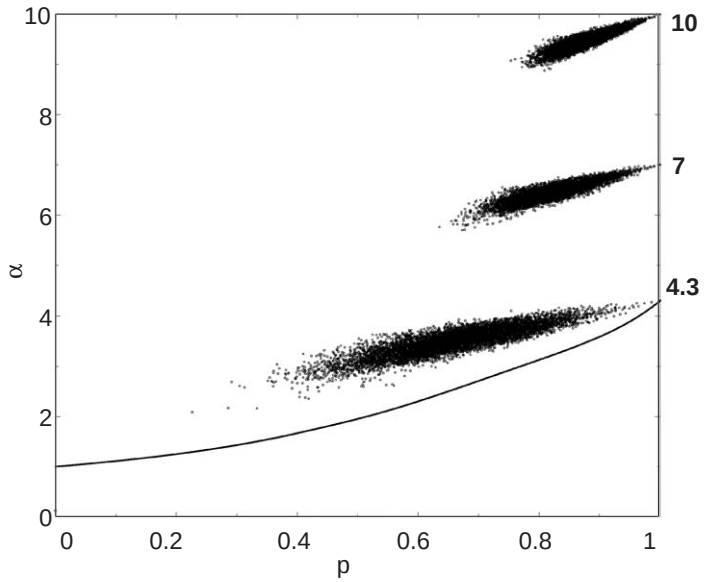

Fig. 4. Clouds associated to search trees obtained from the resolution of three unsat instances with initial ratios $\alpha_0 = 4.3, 7$ and 10, respectively. Each point in the cloud corresponds to a splitting node in the search tree. Sizes of instances and search trees are $N = 120, Q = 7597$ for $\alpha_0 = 4.3$, $N = 200, Q = 6335$ for $\alpha_0 = 7$, and $N = 300, Q = 6610$ for $\alpha_0 = 10$.

Table 1
Logarithm of the complexity ω from experiments (EXP) and theory (THE) as a function of the ratio α_0 of clauses per variable of the 3-SAT instance

α_0	4.3	7	10	15	20	3.5
ω_{EXP}	0.089	0.0477	0.0320	0.0207	0.0153	0.034
	± 0.001	± 0.0005	± 0.0005	± 0.0002	± 0.0002	± 0.003
ω_{THE}	0.0916	0.0486	0.0323	0.0207	0.0153	0.035

Ratios above 4.3 correspond to unsat instances; the rightmost ratio lies in the upper sat phase.

We experimentally measured Q, and averaged its logarithm ω over a large number of instances. Results have then be extrapolated to the $N \to \infty$ limit [9] and are reported in Table 1. ω is a decreasing function of α_0 [4]: the larger α_0, the larger the number of clauses affected by a split, and the earlier a contradiction is detected. We will use the vocable "branch" to denote a path in the refutation tree which joins the top node (root) to a contradiction (leaf). The number of branches, B, is related to the number of nodes, Q, through the relation $Q = B - 1$ valid for any complete binary tree. As far as exponential (in N) scalings are concerned, the logarithm of B (divided by N) equals ω. In the following paragraph, we show how B can be estimated through the use of a matrix formalism.

3.2. Parallel growth process and Markovian evolution matrix

The probabilistic analysis of DPLL in the unsat regime appears to be a formidable task since the search tree of Fig. 2B is the output of a complex, sequential process: nodes and edges are added by DPLL through successive descents and backtrackings (depth-first search). We have imagined a different building up of the refutation tree, which results in the same complete tree but can be mathematically analyzed. In our imaginary process (Fig. 5), the tree grows in parallel, layer after layer (breadth-first search). At time $T = 0$, the tree reduces to a root node, to which is attached the 3-SAT instance to be solved, and an attached outgoing edge. At time T, that is, after having assigned T variables in the instance attached to each branch, the tree is made of $B(T)$ ($\leqslant 2^T$) branches, each one carrying a partial assignment of variables. At next time step $T \to T + 1$, a new layer is added by assigning, according to DPLL heuristic, one more variable along every branch. As a result, a branch may keep growing through unitary propagation, get hit by a contradiction and die out, or split if the partial assignment does not induce unit clauses. This parallel growth process is Markovian, and can be encoded in an instance-dependent matrix we now construct.

To do so, we need some preliminary definitions:

Definition 1 (Partial state of variables). The partial state s of a Boolean variable x is one of the three following possibilities: undetermined (u) if the variable has not been assigned by the search heuristic yet, true (t) if the variable is assigned to true, false

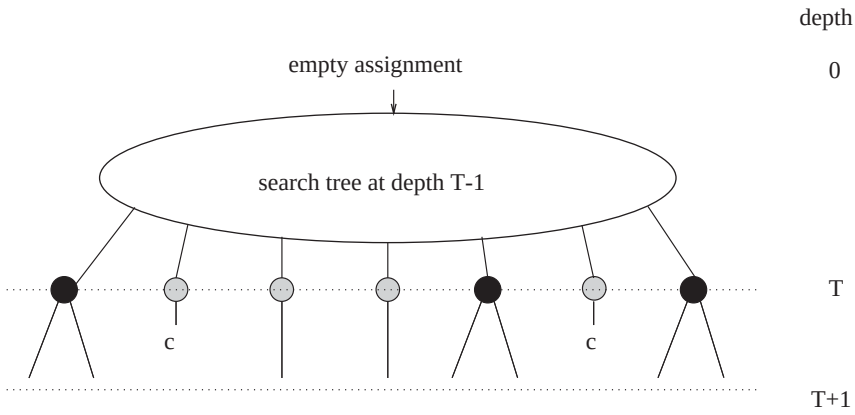

Fig. 5. Imaginary, parallel growth process of an unsat search tree used in the theoretical analysis. Variables are fixed through unit-propagation, or by the splitting heuristic as in the DPLL procedure, but branches evolve in parallel. T denotes the depth in the tree, that is the number of variables assigned by DPLL along each branch. At depth T, one literal is chosen on each branch among 1-clauses (unit-propagation, gray circles not represented in Fig. 2), or 2,3-clauses (splitting, black circles as in Fig. 2). 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.

(f) if the variable is assigned to false. The partial state S of a set of Boolean variables $X = \{x_1, x_2, \ldots, x_N\}$ is the collection of the states of its elements, $S = \{s_1, s_2, \ldots, s_N\}$.

Let **I** be an instance of the SAT problem, defined over a set of Boolean variables X with partial state S. A clause of **I** is said to be

- satisfied if at least one of its literals is true according to S;
- unsatisfied, or violated if all its literals are false according to S;
- undetermined otherwise; then its 'type' is the number ($= 1, 2, 3$) of undetermined variables it includes.

The instance **I** is said to be satisfied if all its clauses are satisfied, unsatisfied if one (at least) of its clauses is violated, undetermined otherwise. The set of partial states that violate **I** is denoted by W.

Definition 2 (Vector space attached to a variable). To each Boolean variable x is associated a three-dimensional vector space **v** with spanning basis $|u\rangle$, $|t\rangle$, $|f\rangle$, orthonormal with respect to the dot (inner) product denoted by $\langle .|.\rangle$,

$$\langle u|u\rangle = \langle t|t\rangle = \langle f|f\rangle = 1, \quad \langle u|t\rangle = \langle u|f\rangle = \langle t|f\rangle = 0. \tag{4}$$

The partial state attached to a basis vector $|s\rangle$ is s ($= u, t, f$).

Letters u, t, f stand for the different partial states the variable may acquire in the course of the search process. Note that the coefficients of the decomposition of any

vector $|x\rangle \in \mathbf{v}$ over the spanning basis,

$$|x\rangle = x^{(u)}|u\rangle + x^{(t)}|t\rangle + x^{(f)}|f\rangle, \tag{5}$$

can be obtained through use of the dot product: $x^{(s)} = \langle s|x\rangle$ with $s = u, t, f$. By extension, $\langle S|$ denotes the transposed of vector $|S\rangle$.

Definition 3 (Vector space attached to a set of variables). We associate to the set $X = \{x_1, x_2, \ldots, x_N\}$ of N Boolean variables the 3^N-dimensional vector space $\mathbf{V} = \mathbf{v}_1 \otimes \mathbf{v}_2 \otimes \cdots \otimes \mathbf{v}_N$. The spanning basis of $\mathbf{V}$ is the tensor product of the spanning basis of the $\mathbf{v}_i$'s. To lighten notations, we shall write $|s_1, s_2, \ldots, s_N\rangle$ for $|s_1\rangle \otimes |s_2\rangle \otimes \cdots \otimes |s_N\rangle$. The partial state attached to a basis vector $|S\rangle = |s_1, s_2, \ldots, s_N\rangle$ is $S = (s_1, s_2, \ldots, s_N)$. The dot product naturally extends over $\mathbf{V}$: $\langle s'_1, s'_2, \ldots, s'_N | s_1, s_2, \ldots, s_N\rangle = 1$ if $s_i = s'_i$ $\forall i$, 0 otherwise.

Any element $|X\rangle \in \mathbf{V}$ can be uniquely decomposed as a linear combination of vectors from the spanning basis. Two examples of vectors are $|\Sigma\rangle$ and $|U\rangle$, respectively, the sum of all vectors in the spanning basis and the fully undetermined vector,

$$|\Sigma\rangle = (|u\rangle + |t\rangle + |f\rangle) \otimes (|u\rangle + |t\rangle + |f\rangle) \otimes \cdots \otimes (|u\rangle + |t\rangle + |f\rangle), \tag{6}$$

$$|U\rangle = |u, u, \ldots, u\rangle. \tag{7}$$

Basis vectors fulfill the closure identity

$$\sum_S |S\rangle \ \langle S| = \mathbf{1}, \tag{8}$$

where $\mathbf{1}$ is the identity operator on $\mathbf{V}$. To establish identity (8), apply the left-hand side operator to any vector $|S'\rangle$ and take advantage of the orthonormality of the spanning basis.

Definition 4 (Heuristic-induced; Transition probabilities). Let $S = (s_1, s_2, \ldots, s_N)$ be a partial state which does not violate instance $\mathbf{I}$. Call $S^{(j,x)}$, with $j = 1, \ldots, N$ and $x = t, f$, the partial state obtained from S by replacing s_j with x. The probability that the heuristic under consideration (UC, GUC, ...) chooses to assign variable x_j when presented partial state S is denoted by $h(j|S)$. The probability that the heuristic under consideration then fixes variable x_j to x $(= t, f)$ is denoted by $g(x|S, j)$.

A few elementary facts about transition probabilities are:

(1) $h(j|S) = 0$ if $s_j \neq u$.
(2) $g(x|S, j) + g(\bar{x}|S, j) = 1$.
(3) Assume that the number $C_1(S)$ of undetermined clauses of type 1 (unit clauses) is larger or equal to unity. Call $C_1(j|S)$ the number of unit clauses containing variable x_j, and $C_1(x|S, j)$ the number of unit clauses satisfied if x_j equals x $(= t, f)$. Clearly $C_1(j|S) = C_1(t|S, j) + C_1(f|S, j)$. Then, as a result of unit-propagation

$$h(j|S) = \frac{C_1(j|S)}{C_1(S)},$$

$$g(x|S,j) = \frac{C_1(x|S,j)}{C_1(j|S)} \quad \text{for } x = t, f \text{ and } C_1(j|S) \geqslant 1. \tag{9}$$

(4) In the absence of unitary clause ($C_1(S) = 0$), transition probabilities depend on the details of the heuristic. For instance, in the case of the UC heuristic
   (a) if $s_j = u$, $h(j|S) = \frac{1}{u(S)}$ and $g(x|S,j) = \frac{1}{2}$,
   (b) if $s_j \neq u$, $h(j|S) = 0$,
   where $u(S)$ is the number of undetermined variables in partial state S.

(5) The sum of transition probabilities from a partial state S is equal to unity,

$$\sum_{j=1}^{N} h(j|S)[g(t|S,j) + g(f|S,j)] = 1. \tag{10}$$

It is important to stress that the definition of the transition probabilities does not make any reference to any type of backtracking. It relies on the notion of variable assignment through the heuristic of search only.

Let us now introduce the

Definition 5. The evolution operator is a linear operator **H** acting on **V** encoding the action of DPLL for a given unsatisfiable instance **I**. Its matrix elements in the spanning basis are

(1) if S violates **I**,

$$\langle S'|\mathbf{H}|S\rangle = \begin{cases} 1 & \text{if } S' = S \\ 0 & \text{if } S' \neq S, \end{cases} \tag{11}$$

(2) if S does not violate **I**,

$$\langle S'|\mathbf{H}|S\rangle = \begin{cases} h(j|S) \times g(x|S,j) & \text{if } C_1(S) \geqslant 1 \text{ and } S' = S^{(j,x)}, \\ h(j|S) & \text{if } C_1(S) = 0 \text{ and} \\ & (S' = S^{(j,x)} \text{ or } S' = S^{(j,\bar{x})}), \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where S, S' are the attached partial states to $|S\rangle, |S'\rangle$, and $C_1(S)$ is the number of undetermined clauses of type 1 (unitary clauses) for partial state S.

Notice that we use the same notation, **H**, for the operator and its matrix in the spanning basis. The different cases encountered in the above definition of **H** are symbolized in Fig. 6. We may now conclude:

Theorem 6 (Branch function and average size of refutation tree). *Call branch function the function B with integer-valued argument T,*

$$B(T) = \langle \Sigma|\mathbf{H}^T|U\rangle, \tag{13}$$

where **H** *is the evolution operator associated to the unsatisfiable instance* **I**, $\mathbf{H}^T$ *denotes the Tth (matricial) power of* **H**, *and vectors $|\Sigma\rangle, |U\rangle$ are defined in* (6,7).

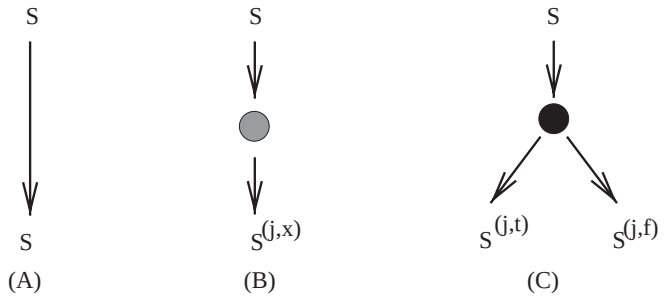

Fig. 6. Transitions allowed by the heuristic-induced evolution operator. Gray and black nodes correspond to variables assigned through unit-propagation and splitting respectively, as in Fig. 5. **A**. If the partial state S already violates the instance **I**, it is left unchanged. **B**. If the partial state does not violate **I** and there is at least one unitary clause, a variable is fixed through unit propagation (gray node) e.g. $x_j = x$. The output partial state is $S^{j,x}$. **C**. If the partial state does not violate **I** and there is no unitary clause, a variable x_j is fixed through splitting (black node). Two partial states are generated, $S^{j,t}$ and $S^{j,f}$.

Then, there exist two instance-dependent integers T^* $(\leqslant N)$ *and* B^* $(\leqslant 2^N)$ *such that*,

$$B(T) = B^*, \quad \forall T \geqslant T^*. \tag{14}$$

Furthermore, B^* *is the expectation value over the random assignments of variables of the size* (*number of leaves*) *of the search tree produced by DPLL to refute* **I**. *The smallest non-zero* T^* *for which* (14) *holds is the largest number of variables that the heuristic needs to assign to reach a contradiction.*

Proof. Let S be a partial state. We call refutation tree built from S a complete search tree that proves the unsatisfiability of **I** conditioned to the fact that DPLL is allowed to assign only variables which are undetermined in S. The height of the search tree is the maximal number of assignments leading from the root node (attached to partial state S) to a contradictory leaf.

Let T be a positive integer. We call $b_T(S)$ the average size (number of leaves) of refutation trees of height $\leqslant T$ that can be built from partial state S. Clearly, $b_T(S) = 1$ for all $S \in W$, and $b_T(S) \geqslant 2$ if $S \notin W$. Recall W is the set of violating partial states from Definition 1.

Assume now T is an integer larger or equal to 1, S a partial state with $C_1(S)$ unitary clauses. Our parallel representation of DPLL allows us to write simple recursion relations:

(1) if $S \in W$, $b_T(S) = 1 = b_{T-1}(S)$.
(2) if $S \notin W$ and $C_1(S) \geqslant 1$,

$$b_T(S) = \sum_{j=1}^{N} \sum_{x=t,f} h(j|S) g(x|S,j)\ b_{T-1}(S^{(j,x)}). \tag{15}$$

(3) if $S \notin W$ and $C_1(S) = 0$,

$$b_T(S) = \sum_{j=1}^{N} h(j|S)\ [b_{T-1}(S^{(j,t)}) + b_{T-1}(S^{(j,f)})]. \tag{16}$$

These three different cases are symbolized in Figs. 6A, B and C, respectively. From definitions (11,12), these recursion relations are equivalent to

$$b_T(S) = \sum_{S'} \langle S'|\mathbf{H}|S\rangle\ b_{T-1}(S'), \tag{17}$$

for any partial state S. Let $|b_T\rangle$ be the vector of $\mathbf{V}$ whose coefficients on the spanning basis $\{|S\rangle\}$ are the $b_T(S)$'s. In particular,

$$|b_0\rangle = \sum_{S_0 \in W} |S_0\rangle. \tag{18}$$

Then identity (17) can be written as $|b_T\rangle = \mathbf{H}^\dagger\ |b_{T-1}\rangle$, where $\mathbf{H}^\dagger$ is the transposed of the evolution operator. Note that the branch function (13) is simply $B(T) = \langle U|b_T\rangle$. We deduce

$$|b_T\rangle = (\mathbf{H}^\dagger)^T\ |b_0\rangle = \sum_{S_0 \in W} \sum_{\sigma_T} p(\sigma_T; S_0)|S_0\rangle, \tag{19}$$

where the second sum runs over all $3^{N\times T}$ sequences $\sigma_T = (S_1, S_2, \ldots, S_{T-1}, S_T)$ of T partial states with associated weight

$$\begin{aligned} p(\sigma_T; S_0) &= \langle S_T|\mathbf{H}^\dagger|S_{T-1}\rangle \times \cdots \times \langle S_2|\mathbf{H}^\dagger|S_1\rangle \times \langle S_1|\mathbf{H}^\dagger|S_0\rangle \\ &= \langle S_0|\mathbf{H}|S_1\rangle \times \langle S_1|\mathbf{H}|S_2\rangle \times \cdots \times \langle S_{T-1}|\mathbf{H}|S_T\rangle. \end{aligned} \tag{20}$$

The length of a sequence is the number of partial states it includes. We call S_0-genuine a sequence of partial states σ_T with non-zero weight (20). The second sum on the right-hand side of Eq. (19) may be rewritten as a sum over all S_0-genuine sequences σ_T of length T only.

Lemma 7. *Take $S_0 \in W$. Any S_0-genuine sequence σ_{N+1} of length $N+1$ includes at least one partial state belonging to W.*

Suppose this is not true. There exists a genuine sequence σ_{N+1} with $S_T \notin W$, $\forall 1 \leqslant T \leqslant N+1$. Call u_T the number of undetermined variables in partial state S_T. Since the sequence is genuine, $\langle S_{T-1}|\mathbf{H}|S_T\rangle \neq 0$ for every T comprised between 1 and $N+1$. From the evolution operator definition (12), S_T contains exactly one more undetermined variable than S_{T-1}, and $u_T = u_{T-1} + 1$ for all $1 \leqslant T \leqslant N+1$. Hence $u_{N+1} - u_0 = N+1$. But u_0 and u_{N+1} are, by definition, integer numbers comprised between 0 and N.

From Lemma 7, the index v of a S_0-genuine sequence σ_{N+1} of length $N+1$,

$$v = \sup\{T : 1 \leqslant T \leqslant N+1 \text{ and } S_T \in \sigma_{N+1} \text{ and } S_T \in W\}, \tag{21}$$

exists and is larger, or equal, to 1. Let us define

$$\hat{\sigma}_{N+1} = (S_{v+1}, S_{v+2}, \ldots, S_N, S_{N+1}). \tag{22}$$

From definition (11), σ_{N+1} is simply S_0 repeated v times followed by $\hat{\sigma}_{N+1}$, and $p(\sigma_{N+1}) = p(\hat{\sigma}_{N+1})$. Call $v^*(S_0)$ the smallest index of all S_0-genuine sequences of length $N+1$, and v^* the minimum of $v^*(S_0)$ over $S_0 \in W$. Then, from Eq. (19), $|b_{N+1}\rangle = |b_N\rangle = \cdots = |b_{T^*}\rangle$ where $T^* = N + 1 - v^* \leqslant N$. Thus $|b_{T^*}\rangle$ is a right eigenvector of $\mathbf{H}^\dagger$ with eigenvalue unity, and $|b_T\rangle = |b_{T^*}\rangle$ for all $T \geqslant T^*$. T^*, which depends upon instance $\mathbf{I}$, is the length of the longest genuine sequence without repetition. It is the maximal number of (undetermined) variables to be fixed before a contradiction is found.

Lemma 8. *Take $S \notin W$. Then there is no S-genuine sequence of length T^*.*

Suppose this is not true. There exist $S \notin W$ and a S-genuine sequence σ_{T^*} of length T^*. As S does not violate $\mathbf{I}$, and $\mathbf{I}$ is not satisfiable, there are still some undetermined variables in partial state S. A certain number of them, say $T' \geqslant 1$, must be assigned to some t, f values to reach a contradiction, that is, a partial state $S_0 \in W$. Therefore there exists a S_0-genuine sequence, $\tilde{\sigma}$, of length $T' \geqslant 1$ ending with S and with no repeated partial state. Concatenating $\tilde{\sigma}$ and σ_{T^*}, we obtain a S_0-genuine sequence of length $T^* + T' > T^*$ and without repetition, in contradiction with the above result. □

Using Lemma 8, we may replace $|b_0\rangle$ in Eq. (19) with $|\Sigma\rangle$, and find

$$B(T) \equiv \langle \Sigma | \mathbf{H}^T | U \rangle = \langle U | (\mathbf{H}^\dagger)^T | \Sigma \rangle = \langle U | b_{T^*} \rangle = b_{T^*}(U), \tag{23}$$

for all $T \geqslant T^*$. Hence, $B^* = b_{T^*}(U)$ is the average size (over the random assignments made by the heuristic) of the refutation tree to instance $\mathbf{I}$ generated from the fully undetermined partial state. □

3.3. Some examples of short instances and associated matrices

We illustrate the above definitions and results with three explicit examples of instances involving few variables:

Example 9 (Instance over $N = 1$ variable). Consider the following unsat instance built from a single variable:

$$\mathbf{I}_1 = x_1 \wedge \bar{x}_1. \tag{24}$$

The three-dimensional vector space $\mathbf{v}_1$ is spanned by vectors $|u\rangle, |t\rangle, |f\rangle$. The evolution matrix reads

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{pmatrix} \quad \text{with } |u\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad |t\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad |f\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{25}$$

Entries can be interpreted as follows. Starting from the u state, variable x_1 will be set through unit-propagation to t or f with equal probabilities: $\langle t|\mathbf{H}|u\rangle = \langle f|\mathbf{H}|u\rangle = 1/2$. Once the variable has reached this state, the instance is violated: $\langle t|\mathbf{H}|t\rangle = \langle f|\mathbf{H}|f\rangle = 1$. All other entries are null. In particular, state u can never be reached from any state,

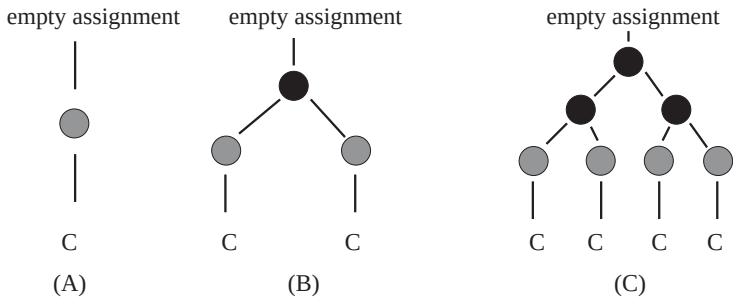

Fig. 7. Refutation search trees associated to instances $\mathbf{I}_1$, $\mathbf{I}_2$ and $\mathbf{I}_3$. Gray and black nodes correspond to variables assigned through unit-propagation and split, respectively, as in Fig. 5. **A**. Example 9: Refutation of instance $\mathbf{I}_1$ is obtained as a result of unit-propagation. The size (number of leaves) of the search tree is $B=1$. **B**. Example 10: search tree generated by DPLL on instance $\mathbf{I}_2$. The black and gray node correspond to the split of x_1 and unit-propagation over x_2, or vice versa. The size of the tree is $B=2$. **C**. Example 11: Search tree corresponding to the instance $\mathbf{I}_3$ when DPLL first splits variable x_3. The size of the tree is $B=4$. If the first split variable is x_1 or x_2, the refutation search tree of instance $\mathbf{I}_3$ corresponds to case **B**.

so the first line of the matrix is filled in with zeroes: $\langle u|\mathbf{H}|s\rangle = 0, \forall s$. Function (13) is easily calculated

$$B(T) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}^{\dagger} . \mathbf{H}^T . \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = 1, \quad \forall T \geqslant 0. \tag{26}$$

Therefore, $T^* = B^* = 1$. Indeed, refutation is obtained without any split, and the search tree involves a unique branch of length 1 (Fig. 7A).

Our next example is a 2-SAT instance whose refutation requires to split one variable.

Example 10 (Instance over $N=2$ variables, with a unique refutation tree).

$$\mathbf{I}_2 = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2). \tag{27}$$

The evolution matrix $\mathbf{H}$ is a 9×9 matrix with 16 non-zero entries,

$$\langle s,u|\mathbf{H}|u,u\rangle = \langle u,s|\mathbf{H}|u,u\rangle = \tfrac{1}{2}, \quad \forall s = t,f, \tag{28}$$

$$\langle s,s'|\mathbf{H}|s,u\rangle = \langle s,s'|\mathbf{H}|u,s'\rangle = \tfrac{1}{2}, \quad \forall s,s' = t,f, \tag{29}$$

$$\langle s',s|\mathbf{H}|s',s\rangle = 1, \quad \forall s,s' = t,f. \tag{30}$$

We now explain how these matrix elements were obtained. From the undetermined state $|u,u\rangle$, any of the four clause can be chosen by the heuristic. Thus, any of the two literals x_1, x_2 has a probability $1/2$ to be chosen: $h(1|u,u) = h(2|u,u) = \frac{1}{2}$. Next, unit-propagation will set the unassigned variable to true, or false with equal probabilities $1/2$ (29). Finally, entries corresponding to violating states in Eq. (30) are calculated according to rule (11).

The branch function $B(T)$ equals 1 for $T=0$, 2 for any $T\geqslant 1$; thus, $T^*=1$ and $B^*=2$, in agreement with the associated search tree symbolized in Fig. 7B.

We now introduce an instance with a non-unique refutation tree.

Example 11 (Instance with $N=3$ variables, and two refutation trees).

$$\mathbf{I}_3=(x_1\vee x_2)\wedge(\bar{x}_1\vee x_2)\wedge(x_1\vee\bar{x}_2)\wedge(\bar{x}_1\vee\bar{x}_2)\wedge(x_3\vee\bar{x}_3). \tag{31}$$

Notice the presence of a (trivial) clause containing opposite literals, which allows us to obtain a variety in the search trees without considering more than three variables. The evolution matrix $\mathbf{H}$ is a 27×27 matrix with 56 non-zero entries (for the GUC heuristic),

$$\langle s,u,u|\mathbf{H}|u,u,u\rangle=\langle u,s,u|\mathbf{H}|u,u,u\rangle=\tfrac{2}{5},\quad \forall s=t,f, \tag{32}$$

$$\langle u,u,s|\mathbf{H}|u,u,u\rangle=\tfrac{1}{5},\quad \forall s=t,f, \tag{33}$$

$$\langle s,s',s''|\mathbf{H}|s,u,s''\rangle=\langle s,s',s''|\mathbf{H}|u,s',s''\rangle=\tfrac{1}{2},\quad \forall s,s'=t,f;\ s''=u,t,f,$$

$$\langle s',u,s|\mathbf{H}|u,u,s\rangle=\langle u,s',s|\mathbf{H}|u,u,s\rangle=\tfrac{1}{2},\quad \forall s,s'=t,f,$$

$$\langle s,s',s''|\mathbf{H}|s,s',s''\rangle=1,\quad \forall s,s'=t,f;\ s''=u,t,f.$$

The first split variable is x_3 if the last clause is chosen (probability $1/5$), or x_1 or x_2 otherwise (with probability $2/5$ each), leading to expressions (32) and (33). The remaining entries of $\mathbf{H}$ are obtained in the same way as explained in Example 10.

We obtain $B(0)=1$, $B(1)=2$ and $B(T\geqslant 2)=12/5$. Therefore, $T^*=2$ and

$$B^*=\tfrac{12}{5}=\tfrac{4}{5}\times 2+\tfrac{1}{5}\times 4, \tag{34}$$

where the different contributions to B^* and their probabilities are explicitly written down, see Figs. 7B and C.

3.4. Dynamical annealing approximation

Let us denote by $\overline{q}$ the expectation value of a function q of the instance $\mathbf{I}$ over the random 3-SAT distribution, at given numbers of variable, N, and clauses, αN. From Theorem 6, the expectation value of the size of the refutation tree is

$$B^*(\alpha,N)\equiv\overline{B^*}=\langle\Sigma|\overline{\mathbf{H}^N}|U\rangle. \tag{35}$$

Calculation of the expectation value of the Nth power of $\mathbf{H}$ is a hard task that we were unable to perform for large sizes N. We therefore turned to a simplifying approximation, hereafter called dynamical annealing. This approximation is not thought to be justified in general, but may be asymptotically exact in some limiting cases we will expose later on.

A first temptation is to approximate the expectation of the Nth power of $\mathbf{H}$ with the Nth power of the expectation of $\mathbf{H}$. This is however too a brutal approximation to be meaningful, and a more refined scheme is needed.

Definition 12 (Clause projection operator). Consider an instance **I** of the 3-SAT problem. The clause vector $\vec{C}(S)$ of a partial state S is a three dimensional vector $\vec{C} = (C_1, C_2, C_3)$, where C_j is the number of undetermined clauses of **I** of type j. The clause projection operator, $\mathbf{P}(\vec{C})$, is the operator acting on **V** and projecting onto the subspace of partial state vectors with clause vectors $\vec{C}$

$$\mathbf{P}(\vec{C})\ |S\rangle = \left[\prod_{j=1}^{3} \delta_{C_j - C_j(S)}\right]\ |S\rangle, \tag{36}$$

where δ is the Kronecker function. The sum of all state vectors in the spanning basis with clause vector $\vec{C}$ is denoted by $|\Sigma(\vec{C})\rangle = \mathbf{P}(\vec{C})\ |\Sigma\rangle$. The sum of all state vectors in the spanning basis with clause vector $\vec{C}$ and U undetermined variables is denoted by $|\Sigma_U(\vec{C})\rangle$.

It is an easy check that **P** is indeed a projection operator: $\mathbf{P}^2(\vec{C}) = \mathbf{P}(\vec{C})$. As the set of partial states can be partitioned according to their clause vectors

$$\sum_{\vec{C}} \mathbf{P}(\vec{C}) = \sum_{\vec{C}} \mathbf{P}^2(\vec{C}) = \mathbf{1}. \tag{37}$$

We now introduce the clause vector-dependent branch function

$$B(\vec{C}, T) = \langle \Sigma(\vec{C})|\mathbf{H}^T|U\rangle. \tag{38}$$

Summation of the B's over all $\vec{C}$ gives back function (13) from identity (37). The evolution equation for $B(\vec{C}, T)$ is

$$\begin{aligned} B(\vec{C}, T+1) &= \langle \Sigma(\vec{C})|\mathbf{H} \times \mathbf{H}^T|U\rangle \\ &= \langle \Sigma(\vec{C})|\mathbf{H} \times \left(\sum_{\vec{C}'} \mathbf{P}^2(\vec{C}')\right) \times \mathbf{H}^T|U\rangle \\ &= \sum_{\vec{C}'} \langle \Sigma(\vec{C})|\mathbf{H} \times \mathbf{P}(\vec{C}') \times \left(\sum_{S} |S\rangle\langle S|\right) \times \mathbf{P}(\vec{C}') \times \mathbf{H}^T|U\rangle \\ &= \sum_{\vec{C}'} \sum_{S} \langle \Sigma(\vec{C})|\mathbf{H} \times \mathbf{P}(\vec{C}')|S\rangle\ \langle S|\mathbf{P}(\vec{C}') \times \mathbf{H}^T|U\rangle, \end{aligned} \tag{39}$$

where we have made use of identities (8) and (37). We are now ready to do the two following approximation steps:

Approximation 13 (Dynamical annealing, step A). *Substitute in Eq.* (39) *the partial state vector*

$$\mathbf{P}(\vec{C}')|S\rangle \quad \textit{with} \quad \frac{1}{\langle \Sigma|\Sigma_{N-T}(\vec{C}')\rangle}\ |\Sigma_{N-T}(\vec{C}')\rangle, \tag{40}$$

that is, with its average over the set of basis vectors with clause vector $\vec{C}'$ *and* $N-T$ *undetermined variables.*

Following step A, Eq. (39) becomes an approximated evolution equation for B,

$$B(\vec{C}, T+1) = \sum_{\vec{C}'} \hat{\mathbf{H}}[\vec{C}, \vec{C}'; T]\ B(\vec{C}, T), \tag{41}$$

where the new evolution matrix $\hat{\mathbf{H}}$, not to be confused with $\mathbf{H}$, is

$$\hat{\mathbf{H}}[\vec{C}, \vec{C}'; T] = \frac{\langle \Sigma(\vec{C})|\mathbf{H}|\Sigma_{N-T}(\vec{C}')\rangle}{\langle \Sigma|\Sigma_{N-T}(\vec{C}')\rangle}. \tag{42}$$

Then,

Approximation 14 (Dynamical annealing, step B). *Substitute in Eq.* (41) *the evolution matrix* $\hat{\mathbf{H}}$ *with*

$$\bar{\mathbf{H}}[\vec{C}, \vec{C}'; T] = \frac{\langle \Sigma(\vec{C})|\mathbf{H}|\Sigma_{N-T}(\vec{C}')\rangle}{\overline{\langle \Sigma|\Sigma_{N-T}(\vec{C}')\rangle}} \tag{43}$$

that is, consider the instance $\mathbf{I}$ *is redrawn at each time step* $T \to T+1$, *keeping information about clause vectors at time* T *only*.

Let us interpret what we have done so far. The quantity we focus on is $\bar{B}(\vec{C}; T+1)$, the expectation number of branches at depth T in the search tree (Fig. 5) carrying partial states with clause vector $\vec{C} = (C_1, C_2, C_3)$. Within the dynamical annealing approximation, the evolution of the $\bar{B}$'s is Markovian,

$$\bar{B}(\vec{C}; T+1) = \sum_{\vec{C}'} \bar{\mathbf{H}}\ [\vec{C}, \vec{C}'; T]\ \bar{B}(\vec{C}'; T). \tag{44}$$

The entries of the evolution matrix $\bar{\mathbf{H}}[\vec{C}, \vec{C}'; T]$ can be interpreted as the average number of branches with clause vector $\vec{C}$ that DPLL will generate through the assignment of one variable from a partial assignment (partial state) of variables with clause vector $\vec{C}'$.

For the GUC heuristic, we find [9],

$$\bar{\mathbf{H}}[\vec{C}, \vec{C}'; T] = \binom{C_3'}{C_3' - C_3} \left(\frac{3}{N-T}\right)^{C_3' - C_3} \left(1 - \frac{3}{N-T}\right)^{C_3} \times \sum_{w_2=0}^{C_3' - C_3} \left(\frac{1}{2}\right)^{C_3' - C_3} \binom{C_3' - C_3}{w_2}$$

$$\times \left\{ (1-\delta_{C_1'}) \left(1-\frac{1}{2(N-T)}\right)^{C_1'-1} \sum_{z_2=0}^{C_2'} \binom{C_2'}{z_2} \left(\frac{2}{N-T}\right)^{z_2} \right.$$

$$\times \left(1-\frac{2}{N-T}\right)^{C_2'-z_2} \sum_{w_1=0}^{z_2} \left(\frac{1}{2}\right)^{z_2} \binom{z_2}{w_1} \delta_{C_2-C_2'-w_2+z_2}\, \delta_{C_1-C_1'-w_1+1}$$

$$+ \delta_{C_1'} \sum_{z_2=0}^{C_2'-1} \binom{C_2'-1}{z_2} \left(\frac{2}{N-T}\right)^{z_2} \left(1-\frac{2}{N-T}\right)^{C_2'-1-z_2}$$

$$\left. \times \sum_{w_1=0}^{z_2} \left(\frac{1}{2}\right)^{z_2} \binom{z_2}{w_1} \delta_{C_2-C_2'-w_2+z_2+1} [\delta_{C_1-w_1} + \delta_{C_1-1-w_1}] \right\}, \quad (45)$$

where δ_X denotes the Kronecker delta function over integers X: $\delta_X = 1$ if $X = 0$, $\delta_X = 0$ otherwise. Expression (45) is easy to obtain from the interpretation following Eq. (44).

3.5. Generating functions and asymptotic scalings at large N

Let us introduce the generating function $G(\vec{y}; T)$ of the average number of branches $\bar{B}(\vec{C}; T)$ where $\vec{y} \equiv (y_1, y_2, y_3)$, through

$$G(\vec{y}; T) = \sum_{\vec{C}} \mathrm{e}^{\vec{y}\cdot\vec{C}} \bar{B}(\vec{C}T), \quad \vec{y}\cdot\vec{C} \equiv \sum_{j=1}^{3} y_j C_j. \quad (46)$$

Evolution Eq. (41) for the $\bar{B}$'s can be rewritten in term of the generating function G,

$$G(\vec{y}; T+1) = \mathrm{e}^{-\gamma_1(\vec{y})} G(\vec{\gamma}(\vec{y}); T) + (\mathrm{e}^{-\gamma_2(\vec{y})}(\mathrm{e}^{y_1}+1) - \mathrm{e}^{-\gamma_1(\vec{y})}) G(-\infty, \gamma_2(\vec{y}), \gamma_3(\vec{y}); T), \quad (47)$$

where $\vec{\gamma}$ is a vectorial function of argument $\vec{y}$ whose components read

$$\gamma_1(\vec{y}) = y_1 + \ln\left[1 - \frac{1}{2(N-T)}\right],$$

$$\gamma_2(\vec{y}) = y_2 + \ln\left[1 + \frac{2}{N-T}\left(\frac{\mathrm{e}^{-y_2}}{2}(1+\mathrm{e}^{y_1}) - 1\right)\right],$$

$$\gamma_3(\vec{y}) = y_3 + \ln\left[1 + \frac{3}{N-T}\left(\frac{\mathrm{e}^{-y_3}}{2}(1+\mathrm{e}^{y_2}) - 1\right)\right]. \quad (48)$$

To solve Eq. (47), we infer the large N behavior of G from the following remarks:

(1) Each time DPLL assigns variables through splitting or unit-propagation, the numbers C_j of clauses of length j undergo O(1) changes. It is thus sensible to assume that, when the number of assigned variables increases from $T_1 = tN$ to

$T_2 = tN + \Delta T$ with ΔT very large but o(N) e.g. $\Delta T = \sqrt{N}$, the densities $c_2 = C_2/N$ and $c_3 = C_3/N$ of 2-and 3-clauses have been modified by o(1).

(2) On the same time interval $T_1 < T < T_2$, we expect the number of unit-clauses C_1 to vary at each time step. But its distribution $\rho(C_1|c_2, c_3; t)$, conditioned to the densities c_2, c_3 and the reduced time t, should reach some well-defined limit distribution. This claim is a generalization of the result obtained by Friez and Suen [18] for the analysis of the GUC heuristic in the absence of backtracking.

(3) As long as a partial state does not violate the instance, very few unit-clauses are generated, and splitting frequently occurs. In other words, the probability that $C_1 = 0$ is strictly positive as N gets large.

The above arguments entice us to make the following

Claim 15 (Asymptotic expression for the generating function G). *For large N, T at fixed ratio $t = T/N$, the generating function* (46) *of the average numbers $\bar{B}$ of branches is expected to behave as*

$$G(y_1, y_2, y_3; tN) = \exp[N\varphi(y_2, y_3; t) + \psi(y_1, y_2, y_3; t) + \mathrm{o}(1)]. \tag{49}$$

Hypothesis (49) expresses in a concise way some important information on the distribution of clause populations during the search process that we now extract. Call ω the Legendre transform of φ,

$$\omega(c_2, c_3; t) = \min_{y_2, y_3} [\varphi(y_2, y_3; t) - y_2 c_2 - y_3 c_3]. \tag{50}$$

Then, combining Eqs. (46), (49) and (50), we obtain

$$\sum_{C_1 \geqslant 0} \rho(C_1|c_2, c_3; t) \bar{B}(C_1, c_2 N, c_3 N; tN) \asymp \exp[N\omega(c_2, c_3; t)], \tag{51}$$

up to non-exponential in N corrections. In other words, the expectation value of the number of branches carrying partial states with $(1 - t)N$ undetermined variables and $c_j N$ j-clauses ($j = 2, 3$) scales exponentially with N, with a growth function $\omega(c_2, c_3; t)$ related to $\varphi(y_2, y_3; t)$ through identity (50). Moreover, $\varphi(0, 0; t)$ is the logarithm of the number of branches (divided by N) after a fraction t of variables have been assigned. The most probable values of the densities $c_j(t)$ of j-clauses are then obtained from the partial derivatives of $\varphi : c_j(t) = \partial\varphi/\partial y_j(0, 0)$ for $j = 2, 3$.

Let us emphasize that φ in Eq. (49) does not depend on y_1. This hypothesis simply expresses that, as far as non-violating partial states are concerned, both terms on the right-hand side of (47) are of the same order, and that the density of unit-clauses, $c_1 = \partial\varphi/\partial y_1$, identically vanishes.

Similarly, function $\psi(y_1, y_2, y_3; t)$ is related to the generating function of distribution $\rho(C_1|c_2, c_3; t)$,

$$\sum_{C_1 \geqslant 0} \rho(C_1|c_2, c_3; t) \mathrm{e}^{y_1 C_1} = \mathrm{e}^{\psi(y_1, y_2, y_3; t) - \psi(0, y_2, y_3; t)}, \tag{52}$$

where $c_j = \partial\varphi/\partial y_j(y_2, y_3; t)$ ($j = 2, 3$) on the left-hand side of the above formula.

Inserting expression (49) into the evolution equation (47), we find

$$\frac{\partial\varphi}{\partial t}(y_2,y_3;t) = -y_1 + \frac{2}{1-t}\left[e^{-y_2}\left(\frac{1+e^{y_1}}{2}\right)-1\right]\frac{\partial\varphi}{\partial y_2}(y_2,y_3;t) + \frac{3}{1-t}\left[e^{-y_3}\left(\frac{1+e^{y_2}}{2}\right)-1\right]\frac{\partial\varphi}{\partial y_3}(y_2,y_3;t) + \ln[1+K(y_1,y_2)e^{\psi(-\infty,y_2,y_3;t)-\psi(y_1,y_2,y_3;t)}] \quad (53)$$

where $K(y_1,y_2)=e^{-y_2}(e^{2y_1}+e^{y_1})-1$. As φ does not depend upon y_1, the latter may be chosen at our convenience e.g. to cancel K and the contribution from the last term in Eq. (53),

$$y_1 = Y_1(y_2) \equiv y_2 - \ln\left(\frac{1+\sqrt{1+4e^{y_2}}}{2}\right). \quad (54)$$

Such a procedure, sometimes called kernel method and, to our knowledge, first proposed by Knuth [27], is correct in the major part of the y_2, y_3 space and, in particular, in the vicinity of $(0,0)$ we focus on in this paper.[2] We end up with the following partial differential equation (PDE) for φ:

$$\frac{\partial\varphi}{\partial t}(y_2,y_3;t) = H\left[\frac{\partial\varphi}{\partial y_2},\frac{\partial\varphi}{\partial y_3},y_2,y_3,t\right], \quad (55)$$

where H incorporates the details of the splitting heuristic,[3]

$$H_{\mathrm{GUC}}[c_2,c_3,y_2,y_3,t] = -Y_1(y_2) + \frac{3c_3}{1-t}\left[e^{-y_3}\left(\frac{1+e^{y_2}}{2}\right)-1\right] + \frac{c_2}{1-t}(e^{-Y_1(y_2)}-2). \quad (57)$$

We must therefore solve the partial differential equation (PDE) (55) with the initial condition

$$\varphi(y_2,y_3,t=0) = \alpha_0 y_3, \quad (58)$$

obtained through inverse Legendre transform (50) of the initial condition over $\bar{B}$, or equivalently over ω,

$$\omega(c_2,c_3;t=0) = \begin{cases} 0 & \text{if } c_3=\alpha_0, \\ -\infty & \text{if } c_3\neq\alpha_0. \end{cases} \quad (59)$$

[2] It has however to be to modified in a small region of the y_2, y_3 space; a complete analysis of this case was carried out by Cocco and Monasson [9].

[3] For the UC heuristic,

$$H_{\mathrm{UC}} = \ln 2 + \frac{3c_3}{1-t}\left[e^{-y_3}\left(\frac{1+e^{y_2}}{2}\right)-1\right] + \frac{c_2}{1-t}\left(\frac{3}{2}e^{-y_2}-2\right). \quad (56)$$

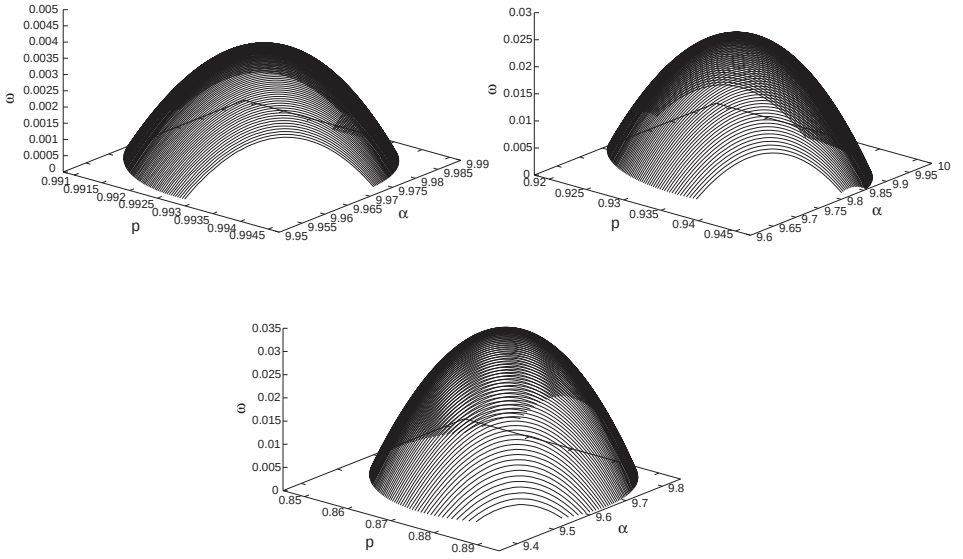

Fig. 8. Snapshots of the surface $\omega(p,\alpha;t)$ for $\alpha_0 = 10$ at three different times i.e. depths in the tree, $t = 0.01$, 0.05 and 0.09 (from left to right, top to down). The height $\omega^*(t)$ of the top of the surface, with coordinates $p^*(t), \alpha^*(t)$, is the logarithm (divided by N) of the number of branches. The coordinates $(p^*(t), \alpha^*(t))$ define the tree trajectory shown in Fig. 3. The halt line is hit at $t_h \simeq 0.094$. Note the overall growth of the surface $\omega(p,\alpha;t)$ with time (beware of the change of scales between figures).

3.6. Interpretation in terms of growth process

We can interpret the dynamical annealing approximation made in the previous paragraphs, and the resulting PDE (55) as a description of the growth process of the search tree resulting from DPLL operation. Using Legendre transform (50), PDE (55) can be written as an evolution equation for the logarithm $\omega(c_2, c_3, t)$ of the average number of branches with parameters c_2, c_3 as the depth $t = T/N$ increases,

$$\frac{\partial \omega}{\partial t}(c_2, c_3, t) = H\left[c_2, c_3, -\frac{\partial \omega}{\partial c_2}, -\frac{\partial \omega}{\partial c_3}, t\right]. \tag{60}$$

Partial differential equation (PDE) (60) is analogous to growth processes encountered in statistical physics [28]. The surface ω, growing with "time" t above the plane c_2, c_3, or equivalently from (3), above the plane p, α (Fig. 8), describes the whole distribution of branches. The average number of branches at depth t in the tree equals

$$B(t) = \int_0^1 \mathrm{d}p \int_0 \mathrm{d}\alpha \mathrm{e}^{N\omega(p,\alpha;t)} \asymp \mathrm{e}^{N\omega^*(t)}, \tag{61}$$

where $\omega^*(t)$ is the maximum over p, α of $\omega(p,\alpha;t)$ reached in $p^*(t), \alpha^*(t)$. In other words, the exponentially dominant contribution to $B(t)$ comes from branches carrying 2+p-SAT instances with parameters $p^*(t), \alpha^*(t)$, that is clause densities $c_2^*(t) = \alpha^*(t)$

$(1-p^*(t))$, $c_3^*(t)=\alpha^*(t)p^*(t)$. Parametric plot of $p^*(t),\alpha^*(t)$ as a function of t defines the tree trajectories in Fig. 3.

The hyperbolic line in Fig. 3 indicates the halt points, where contradictions prevent dominant branches from further growing. Each time DPLL assigns a variable through unit-propagation, an average number $u(p,\alpha)$ of new 1-clauses is produced, resulting in a net rate of $u-1$ additional 1-clauses. As long as $u<1$, 1-clauses are quickly eliminated and do not accumulate. Conversely, if $u>1$, 1-clauses tend to accumulate. Opposite 1-clauses x and $\bar{x}$ are likely to appear, leading to a contradiction [6,18]. The halt line is defined through $u(p,\alpha)=1$, and reads [9]

$$\alpha=\left(\frac{3+\sqrt{5}}{2}\right)\ln\left[\frac{1+\sqrt{5}}{2}\right]\frac{1}{1-p}. \tag{62}$$

It differs from the halt line $\alpha=1/(1-p)$ corresponding to a single branch [18]. As far as dominant branches are concerned, an alternative and simpler way of obtaining the halt criterion is through calculation of the probability $\rho_S^*(t)\equiv\rho(C_1=0|c_2^*(t),c_3^*(t);t)$ that a split occurs when a variable is assigned by DPLL,

$$\rho_S^*(t)=\exp\left(\frac{\partial\varphi}{\partial t}(0,0;t)\right)-1, \tag{63}$$

from Eqs. (52,53). The probability of split vanishes, and unit-clauses accumulate till a contradiction is obtained, when the tree stops growing. Along the tree trajectory, $\omega^*(t)$ grows thus from 0, on the right vertical axis, up to some final positive value, ω_{THE}, on the halt line. ω_{THE} is our theoretical prediction for the logarithm of the complexity (divided by N).[4]

Eq. (60) was solved using the method of characteristics. Using Eq. (3), we have plotted the surface ω at different times, with the results shown in Fig. 8 for $\alpha_0=10$. Values of ω_{THE}, obtained for $4.3<\alpha<20$ by solving Eq. (60) compare very well with numerical results (Table 1). We stress that, though our calculation is not rigorous, it provides a very good quantitative estimate of the complexity. It is therefore expected that our dynamical annealing approximation be qualitatively accurate. It is a reasonable conjecture that it becomes exact at large ratios α_0, where PDE (55) can be exactly solved:

Conjecture 16 (Asymptotic equivalent of ω for large ratios). *Resolution of PDE* (60) *in the large ratio* α_0 *limit gives* (*for the GUC heuristic*),

$$\omega_{\mathrm{THE}}(\alpha_0)\asymp\frac{3+\sqrt{5}}{6\ln 2}\left[\ln\left(\frac{1+\sqrt{5}}{2}\right)\right]^2\frac{1}{\alpha_0}. \tag{64}$$

This result exhibits the $1/\alpha_0$ *scaling proven by Beame et al.* [4], *and is conjectured to be exact.*

[4] Notice that we have to divide the theoretical value by ln 2 to match the definition used for numerical experiments; this is done in Table 1.

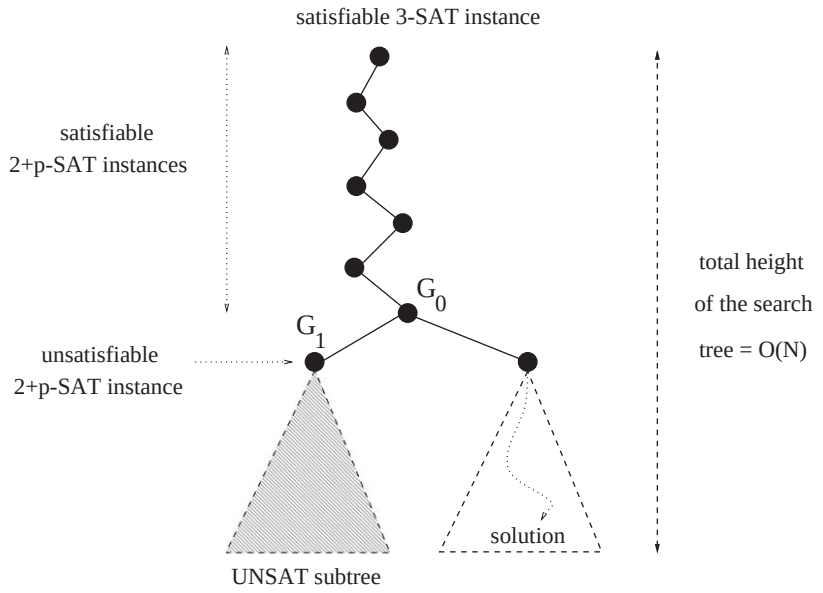

Fig. 9. Detailed structure of the search tree in the upper sat phase ($\alpha_L < \alpha < \alpha_C$). DPLL starts with a satisfiable 3-SAT instance and transforms it into a sequence of 2+p-SAT instances. The leftmost branch in the tree symbolizes the first descent made by DPLL. Above node G_0, instances are satisfiable while below G_1, instances have no solutions. A gray triangle accounts for the (exponentially) large refutation subtree that DPLL has to go through before backtracking above G_1 and reaching G_0. By definition, the highest node reached back by DPLL is G_0. Further backtracking, below G_0, will be necessary but a solution will be eventually found (right subtree), see Fig. 2C.

As α_0 increases, search trees become smaller and smaller, and correlations between branches, weaker and weaker, making dynamical annealing more and more accurate.

4. Upper phase and mixed branch-tree trajectories

The interest of the trajectory framework proposed in this paper is best seen in the upper sat phase, that is, for ratios α_0 ranging from α_L to α_C. This intermediate region juxtaposes branch and tree behaviors, see search tree in Figs. 2C and 9.

The branch trajectory, started from the point $(p = 1, \alpha_0)$ corresponding to the initial 3-SAT instance, hits the critical line $\alpha_C(p)$ at some point G with coordinates (p_G, α_G) after Nt_G variables have been assigned by DPLL, see Fig. 3. The algorithm then enters the unsat phase and, with high probability, generates a 2+p-SAT instance with no solution. A dense subtree that DPLL has to go through entirely, forms beyond G till the halt line (left subtree in Fig. 9). The size of this subtree can be analytically predicted from the theory exposed in Section 3. All calculations are identical, except initial condition (58) which has to be changed into

$$\varphi(y_2, y_3, t = 0) = \alpha_G(1 - p_G)y_2 + \alpha_G p_G y_3. \tag{65}$$

As a result we obtain the size $2^{N_G\omega_G}$ of the unsatisfiable subtree to be backtracked (leftmost subtree in Fig. 9). $N_G = N(1 - t_G)$ denotes the number of undetermined variables at point G.

G is the highest backtracking node in the tree (Figs. 2C and 9) reached back by DPLL, since nodes above G are located in the sat phase and carry 2+p-SAT instances with solutions. DPLL will eventually reach a solution. The corresponding branch (rightmost path in Fig. 2C) is highly non-typical and does not contribute to the complexity, since almost all branches in the search tree are described by the tree trajectory issued from G (Fig. 3). We expect that the computational effort DPLL requires to find a solution will, to exponential order in N, be given by the size of the left unsatisfiable subtree of Fig. 9. In other words, massive backtracking will certainly be present in the right subtree (the one leading to the solution), and no significant statistical difference is expected between both subtrees.

We have experimentally checked this scenario for $\alpha_0 = 3.5$. The average coordinates of the highest backtracking node, ($p_G \simeq 0.78, \alpha_G \simeq 3.02$), coincide with the computed intersection of the single branch trajectory (Section 2.2) and the estimated critical line $\alpha_C(p)$ [9]. As for complexity, experimental measures of ω from 3-SAT instances at $\alpha_0 = 3.5$, and of ω_G from 2+0.78-SAT instances at $\alpha_G = 3.02$, obey the expected identity

$$\omega_{\mathrm{THE}} = \omega_G \times (1 - t_G), \tag{66}$$

and are in very good agreement with theory (Table 1). Therefore, the structure of search trees corresponding to instances of 3-SAT in the upper sat regime reflects the existence of a critical line for 2+p-SAT instances.

5. Conclusions

In this paper, we have exposed a procedure to understand the complexity pattern of the backtrack resolution of the random Satisfiability problem (Fig. 10). Main steps are:

(1) Identify the space of parameters in which the dynamical evolution takes place; this space will be generally larger than the initial parameter space since the algorithm modifies the instance structure. While the distribution of 3-SAT instances is characterized by the clause per variable ratio α only, another parameter p accounting for the emergence of 2-clauses has to be considered.
(2) Divide the parameter space into different regions (phases) depending on the output of the resolution e.g. sat/unsat phases for 2+p-SAT.
(3) Represent the action of the algorithm as trajectories in this phase diagram. Intersection of trajectories with the phase boundaries allow to distinguish hard from easy regimes (Fig. 10).

In addition, we have also presented a non-rigorous study of the search tree growth, which allows us to accurately estimate the complexity of resolution in presence of massive backtracking. From a mathematical point of view, it is worth noticing that monitoring the growth of the search tree requires a PDE, while ODEs are sufficient to account for the evolution of a single branch [1].

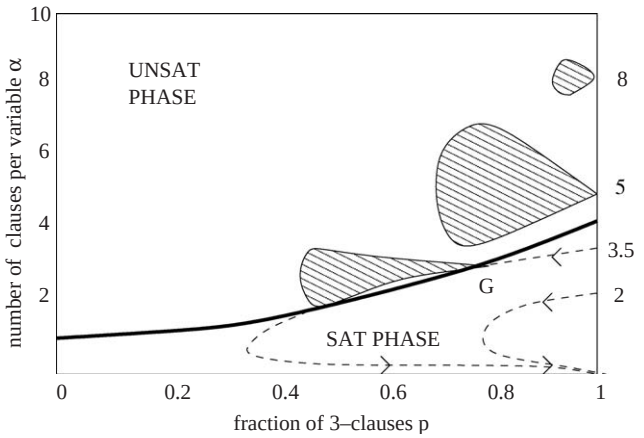

Fig. 10. Schematic representation of the resolution trajectories in the sat (branch trajectories symbolized with dashed line) and unsat (tree trajectories represented by hatched regions) phases. DPLL goes along branch trajectories in a linear time, but takes an exponential time to go through tree trajectories. The mixed case of hard sat instances correspond to the crossing of the boundary separating the two phases (bold line), which leads to the exploration of unsat subtrees before a solution is finally found.

An interesting question raised by this picture is the robustness of the polynomial/exponential crossover point T (Fig. 3). While the ratio α_L separating easy (polynomial) from hard (exponential) resolutions depends on the heuristics used by DPLL ($\alpha_L^{GUC} \simeq 3.003$, $\alpha_L^{UC} = 8/3$), T appears to be located at the same coordinates ($p_T = 2/5$, $\alpha_T = 5/3$) for all three UC, GUC, and SC_1 heuristics. From a technical point of view, the robustness of T comes from the structure of the ODEs (2). The coordinates of T, and the time t_T at which the branch trajectory issued from ($p = 1, \alpha_0 = \alpha_L$) hits the critical line $\alpha_C(p)$ tangentially, obey the equations $\rho_1 = \partial\rho_1/\partial t = 0$ with $\rho_1 = 1 - \alpha(t)(1 - p(t))$. The set of ODEs (2), combined with the previous conditions, gives $p_T = 2/5$ [1].

This robustness explains why the polynomial/exponential crossover location of critically constrained 2+p-SAT instances, which should a priori depend on the algorithm used, was found by Monasson et al. [31] to coincide roughly with the algorithm–independent, tricritical point on the $\alpha_C(p)$ line.

Our approach has already been extended to other decision problems, e.g. the vertex covering of random graphs [23] or the coloring of random graphs [15] (see [25] for recent rigorous results on backtracking in this case). It is important to stress that it is not limited to the determination of the average solving time, but may also be used to capture its distribution [10,20,32] and to understand the efficiency of restarts techniques [21]. Finally, we emphasize that Theorem 6 relates the computational effort to the evolution operator representing the elementary steps of the search heuristic for a *given* instance. It is expected that this approach will be useful to obtain results on the average-case complexity of DPLL at fixed instance, where the average is performed over the random choices done by the algorithm only [30].

Acknowledgements

We thank J. Franco for his constant support during the completion of this work. R. Monasson was in part supported by the ACI Jeunes Chercheurs "Algorithmes d'optimisation et systèmes désordonnés quantiques" from the French Ministry of Research.

References

[1] D. Achlioptas, Lower bounds for random 3-SAT via differential equations, Theoret. Comput. Sci. 265 (2001) 159–185.
[2] D. Achlioptas, P. Beame, M. Molloy, A sharp threshold in proof complexity, in: Proc. STOC 01, 2001, pp. 337–346.
[3] D. Achlioptas, L. Kirousis, E. Kranakis, D. Krizanc, Rigorous results for random (2+p)-SAT, Theoret. Comput. Sci. 265 (2001) 109–129.
[4] P. Beame, R. Karp, T. Pitassi, M. Saks, ACM Symp. on Theory of Computing (STOC98) Assoc. Comput. Mach., New York, 1998, pp. 561–571.
[5] M.T. Chao, J. Franco, Probabilistic analysis of two heuristics for the 3-satisfiability problem, SIAM J. Comput. 15 (1986) 1106–1118.
[6] M.T. Chao, J. Franco, Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k-satisfiability problem, Inform. Sci. 51 (1990) 289–314.
[7] V. Chvàtal, E. Szmeredi, Many hard examples for resolution, J. ACM 35 (1988) 759–768.
[8] C. Coarfa, D.D. Dernopoulos, A. San Miguel Aguirre, D. Subramanian, M.Y. Vardi, Random 3-SAT: The plot thickens, in: R. Dechter (Ed.), Proc. Principles and Practice of Constraint Programming (CP'2000), Lecture Notes in Computer Science, Vol. 1894, 2000, pp. 143–159.
[9] S. Cocco, R. Monasson, Trajectories in phase diagrams, growth processes and computational complexity: how search algorithms solve the 3-Satisfiability problem, Phys. Rev. Lett. 86 (2001) 1654;
S. Cocco, R. Monasson, Analysis of the computational complexity of solving random satisfiability problems using branch and bound search algorithms, Eur. Phys. J. B 22 (2001) 505.
[10] S. Cocco, R. Monasson, Exponentially hard problems are sometimes polynomial, a large deviation analysis of search algorithms for the random satisfiability problem, and its application to stop-and-restart resolutions, Phys. Rev. E 66 (2002) 037101.
[11] J. Crawford, L. Auton, Experimental results on the cross-over point in satisfiability problems, Proc. 11th National Conf. on Artificial Intelligence (AAAI-93), The AAAI Press/MIT Press, Cambridge, MA, 1993, pp. 21–27; Artif. Intell. 81 (1996).
[12] M. Davis, G. Logemann, D. Loveland, A machine program for theorem proving, Commun. ACM 5 (1962) 394–397.
[13] O. Dubois, Y. Boufkhad, J. Mandler, Typical random 3-SAT formulae and the satisfiability threshold, Proc. of the 11th annual ACM-SIAM Symposium on Discrete Algorithms (SODA). San Francisco, California, USA, January 9–11, 2000, pp. 126–127.
[14] O. Dubois, R. Monasson, B. Selman, R. Zecchina (Eds.), Phase transitions in combinatorial problems, Theoret. Comput. Sci. 265 (2001).
[15] L. Ein-Dor, R. Monasson, The dynamics of proving uncolorability of large random graphs, J. Phys. A 36 (2003) 11055.
[16] J. Franco, Results related to thresholds phenomena research in satisfiability: lower bounds, Theoret. Comput. Sci. 265 (2001) 147–157.
[17] E. Friedgut, Sharp thresholds of graph properties, and the k-sat problem, J. Amer. Math. Soc. 12 (1999) 1017.
[18] A. Frieze, S. Suen, Analysis of two simple heuristics on a random instance of k-SAT, J. Algorithms 20 (1996) 312–335.
[19] I. Gent, H. van Maaren, T. Walsh (Eds.), SAT2000: Highlights of satisfiability research in the year 2000, Frontiers in Artificial Intelligence and Applications, Vol. 63, IOS Press, Amsterdam, 2000.

[20] I.P. Gent, T. Walsh, Easy problems are sometimes hard, Artif. Intell. 70 (1994) 335–345.
[21] C.P. Gomes, B. Selman, N. Crato, H. Kautz, J. Automat. Reasoning 24 (2000) 67.
[22] J. Gu, P.W. Purdom, J. Franco, B.W. Wah, Algorithms for satisfiability (SAT) problem: a survey, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, Vol. 35, American Mathematical Society, Providence, RI, 1997, pp. 19–151.
[23] A. Hartmann, M. Weigt, Typical solution time for a vertex-covering algorithm on finite-connectivity random graphs, Phys. Rev. Lett. 86 (2001) 1658.
[24] T. Hogg, B.A. Huberman, C. Williams (Eds.), Frontiers in problem solving: phase transitions and complexity, Artificial Intelligence, Vols. 81, I and II, Elsevier Science, Essex, UK, 1996.
[25] H. Jia, C. Moore, How much backtracking does it take to 3-color a random graph? preprint (2003).
[26] A.C. Kaporis, L.M. Kirousis, E.G. Lalas, The probabilistic analysis of a greedy satisfiability algorithm, ESA (2002) 574–585.
[27] D.E. Knuth, The art of computer programming, Vol. 1: Fundamental Algorithms, Section 2.2.1, Addison-Wesley, New York, 1968.
[28] A. McKane, M. Droz, J. Vannimenus, D. Wolf (Eds.), Scale invariance, interfaces, and non–equilibrium dynamics, NATO ASI Ser. B: Physics, Vol. 344, Plenum Press, New-York, 1995.
[29] D. Mitchell, B. Selman, H. Levesque, Hard and easy distributions of SAT problems, Proc. Tenth National Conf. on Artificial Intelligence (AAAI-92), The AAAI Press/MIT Press, Cambridge, MA, 1992, pp. 440–446.
[30] R. Monasson, On the analysis of backtrack procedures for the coloring of random graphs, in: E. Ben-Naim, H. Frauenfelder, Z. Torczkai (Eds.), Complex Networks, Springer, Berlin, 2004.
[31] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, L. Troyansky, Determining computational complexity from characteristic 'phase transitions', Nature 400 (1999) 133–137;
R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, L. Troyansky, 2+p-SAT: relation of typical-case complexity to the nature of the phase transition, Random Structure and Algorithms 15 (1999) 414.
[32] A. Montanari, R. Zecchina, Optimizing searches via rare events, Phys. Rev. Lett. 88 (2002) 178701.
[33] B. Selman, S. Kirkpatrick, Critical behavior in the satisfiability of random Boolean expressions, Science 264 (1994) 1297–1301.
[34] N. Wormald, Differential equations for random processes and random graphs, Ann. Appl. Probab. 5 (1995) 1217–1235.