8 Analysis of Backtracking Procedures for Random Decision Problems

Simona Cocco, Liat Ein-Dor, Rémi Monasson

Complete search algorithms are procedures capable of deciding whether or not a decision problem has a solution. Among these are the ubiquitous backtracking-like algorithms, where a decision is reached through a sequence of trials and errors. Analysis of the performances of these procedures is difficult but can be done, to some extent, using statistical physics ideas and techniques. Here, this approach is presented and illustrated on the random Satisfiability (SAT) and Graph Coloring (COL) problems.

8.1 Introduction

The wide variety of practical problems that can be mapped onto NP-complete problems, together with the challenge in finding an answer to one of the most important open questions in theoretical computer science, 'Does $\mathrm{NP} = \mathrm{P}$?', have led to intensive studies over the past decades. Despite intense efforts, the worst-case running times of all currently known algorithms grow exponentially with the size of the inputs to these problems. However, NP-complete problems are not always hard. They might even be easy to solve on average [10, 25, 44], i.e., when their resolution complexity is measured with respect to some underlying probability distribution of instances. This 'average-case' behavior depends, of course, on the input distribution.

The average-case analysis of algorithms is a well defined branch of theoretical computer science, with close connections to probability theory and combinatorics [33, 42]. It was recently suggested that these connections could extend up to out-of-equilibrium statistical physics [15]. Indeed, scientists working in the fields of the analysis of algorithms and of statistical physics have common goals. They all aim to understand to the properties of dynamical processes involving an exponentially large (in the size of the input) set of configurations. The differences between the two disciplines mainly lie in the methods of investigation. In contrast to theoretical computer scientists, physicists rarely provide exact results. But concepts and tools developed over the past decades may prove useful in tackling the study of complex algorithms that mathematical approaches have so far failed to solve.

There is a huge variety of algorithms designed to cope with combinatorial problems [30]. Briefly speaking, one can distinguish between complete and incomplete search procedures. The latter are capable of finding solutions quickly but are unable to prove their absence[1] while

[1] This statement is true from a deterministic point of view, but incomplete procedures may be able to disprove the existence of the solution in a probabilistic manner, see [40].

New Optimization Algorithms in Physics. Edited by Alexander K. Hartmann, Heiko Rieger

ISBN: 3-527-40406-6

the former will always output the right answer (existence or absence of solution) however long it takes to find it. While incomplete search procedures are sometimes related to dynamical processes inspired from physics, e.g., simulated annealing, the operation of complete procedures rely on very different principles, with no *a priori* physical motivations. In addition, their study has a long and rich history in theoretical computer science [34]. This makes the analysis of complete algorithms with theoretical physics tools all the more fascinating. In this chapter, we shall focus upon an ubiquitous complete procedure, the so-called Davis–Putnam–Logemann–Loveland (DPLL) algorithm [19,30], at the root of branch-and-bound procedures widely used in 'practical' optimization. The reader is referred to [15] and references therein for a review on recent progress in the analysis of incomplete procedures from the physics point of view.

Virtually all decision problems can be solved with DPLL. Two widely known examples of such problems which we shall focus on throughout this chapter are:

- *Satisfiability of Boolean constraints (SAT).* In K-SAT one is given an instance, that is, a set of M logical constraints (clauses) among N boolean variables, and one wants to know if there exists a truth assignment for the variables which fulfill all the constraints. Each clause is the logical OR of K literals, a literal being one of the N variables or its negation, e.g., $(x_1 \vee x_4 \vee \overline{x_5})$ for 3-SAT.

- *Coloring of graphs (COL).* An input instance of the K-COL decision problem consists of a graph G. The problem involves finding a mapping from the set of vertices to the set of K colors such that no edge links vertices with the same color, or else proving there are none.

We now illustrate the operation of DPLL on an input F of the K-SAT problem defined over a set V of variables. Call Γ_j the set of clauses including j variables, L the set of literals, U the set of unassigned variables, and $depth$ the number of further backtrackings available to DPLL. Initially, $\Gamma_j = \emptyset$ for all $j < K$, $\Gamma_K = F$, $L = \emptyset$, $U = V$, and $depth = 0$. The procedure is defined as follows:

algorithm DPLL$[\Gamma_0, \Gamma_1, \Gamma_2, \ldots, \Gamma_K; L; U; depth]$
begin
  for $j = 1$ to K {updating of clause sets};
  begin
    for all $c = \ell_1 \vee \ell_2 \vee \ldots \vee \ell_j \in \Gamma_j$;
    begin
      if $\exists i \in [1; j]$ such that $\ell_i \in L$ **then** {clause to be removed?}
      begin
        $\Gamma_j := \Gamma_j \setminus \{c\}$;
      end;
      if $\exists i \in [1; j]$ such that $\bar{\ell}_i \in L$ **then** {clause to be reduced?}
      begin
        $\Gamma_j := \Gamma_j \setminus \{c\}$;
        $\Gamma_{j-1} := \Gamma_{j-1} \cup \{c \setminus \ell_i\}$;
      end;
    end;
  end;

```
if Γ0 = Γ1 = Γ2 = ... = ΓK = ∅ then {all clauses are satisfied?}
begin
    print 'SATISFIABLE (by literals in L)' ;
    stop
end;
if Γ0 = ∅ then
begin {there is no violated clause}
    if Γ1 ≠ ∅ then
    begin {there is some unitary clause i.e. with a unique literal}
        ℓ := a literal (unitary clause) randomly chosen in Γ1;
        x := the variable associated to literal ℓ;
        DPLL[Γ0, Γ1, Γ2, ..., ΓK; L ∪ {ℓ}; U \ {x}; depth];
    end;
    else {there is no unitary clause};
        x := a variable chosen in U according to some heuristic rule;
        ℓ := a literal equal to x, or to x̄ depending on some heuristic rule;
        DPLL[Γ0, Γ1, Γ2, ..., ΓK; L ∪ {ℓ}; U \ {x}; depth + 1];
        DPLL[Γ0, Γ1, Γ2, ..., ΓK; L ∪ {ℓ̄}; U \ {x}; depth];
    end;
end
else {there is some violated clause};
    if depth = 0 then
    begin {further backtracking is impossible}
        print 'UNSATISFIABLE';
        stop;
    end;
end;
end
```

The first part of the algorithm consists of updating the sets of clauses after a variable has been assigned at a previous step, e.g., $x = T$. Some clauses are satisfied, e.g., $c = x \vee y \vee z$ and eliminated, other are reduced, e.g., $c = \bar{x} \vee y \vee z \rightarrow c = y \vee z$. The procedure is such that, if some clauses include one variable only, e.g., $c = y$, the corresponding variable is automatically fixed to satisfy the clause ($y = T$). This unitary propagation is repeated up to the exhaustion of all unit clauses. If there is no unit clause ($\Gamma_1 = \emptyset$), a heuristic rule indicates which variable should be selected and which value it should be assigned to. In the presence of a contradiction, that is, two opposite unitary clauses, e.g., $c = y$, $c' = \bar{y}$, DPLL backtracks to the last assignment of literal and tries the opposite value for the attached variable. At the end of the process, a solution is found if no clauses are left, or no backtracking is possible and a proof of unsatisfiability is obtained.

It is convenient to represent the history of the search process, that is, the sequence of trials and errors generated by DPLL by a search tree. Examples of search trees are given in Figure 8.1. Nodes in the tree are attached to the assignment of variables, while edges represent the logical consequences (elimination of satisfied constraints, simplification of other constraints) resulting from these assignments. Branch extremities are marked with contradictions C, or by

a solution S. A good computer-independent measure of the complexity of resolution is the size of the search tree generated by DPLL. This search tree varies with the input of the problem under consideration and the sequence of assignments made by the search procedure.

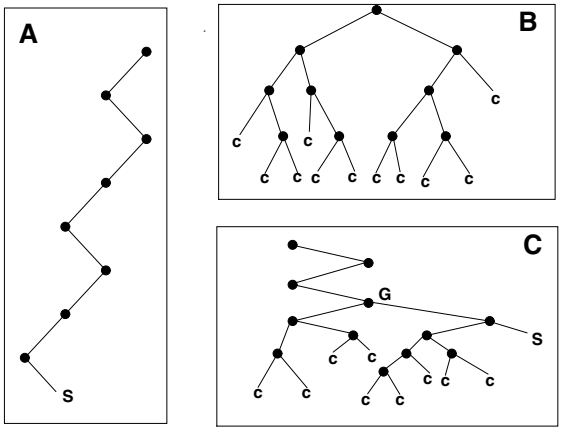

Figure 8.1: Types of search trees generated by the DPLL solving procedure on K-SAT. (A) *Simple branch:* the algorithm easily finds a solution without ever backtracking. (B) *Dense tree:* in the absence of solution, DPLL builds a tree, including many branches ending with contradictory leaves, before 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.

Analysis of the average-case performance of DPLL, that is, of the average of the search tree size for a given decision problem, e.g., SAT, requires a definition of its input distribution. Such distributions are usually unrealistic compared to structured instances from the real world, but are simple enough to allow for some analytical treatment. Current popular input distributions are:

- *Random K-SAT* is the K-SAT problem supplied with a distribution of inputs uniform over all instances having fixed values of N and M. The limit of interest is $N, M \to \infty$ at fixed ratio $\alpha = M/N$ of clauses per variable [17, 27, 37].
- *Random K-COL* inputs are random graphs G after Erdős-Rényi, i.e., drawn with uniform probability among all the graphs having N vertices and E edges. The limit of interest is $N, E \to \infty$ at fixed ratio $c = 2E/N$ of edges per vertex [4, 5, 18].

Random SAT and COL both exhibit a phase transition phenomenon [23]. For small values of their control parameter π ($= \alpha$ or c), and for large input sizes, the answer to the decision problem (existence of an assignment satisfying the constraints, or of a proper coloring) is almost definitely yes. This holds as long as π remains smaller than a (K dependent) critical value π_C called threshold. Above threshold, the answer is no with high probability. The behavior of DPLL is, to some extent, related to this phase transition as shown in Figure 8.2.

Three regimes can be identified:

1. For low control parameter $\pi < \pi_L (< \pi_C)$, the answer is almost definitely yes. There is a finite probability that a solution is found by DPLL with essentially no backtracking [8, 9, 24]. Search trees look like Figure 8.1(A). Their size grows polynomially (linearly) with the input size N (number of variables for SAT, of vertices for COL). In the following we will refer to this regime as low SAT phase.

2. For $\pi > \pi_C$, that is, when the answer is no with high probability, proving the absence of a solution requires the building up of a search tree like the one in Figure 8.1(B), whose size is exponentially large (in N) [11]. This regime will be called the UNSAT phase.

3. In the intermediate regime i.e. $\pi_L < \pi < \pi_C$, there are solutions solutions but finding them is hard (Figure 8.1(C)), and requires exponential effort [3, 12, 13]. This regime will be referred to as the upper SAT phase.

Notice that the location π_L of the easy/hard crossover depends upon the algorithm, in contrast to π_C. The purpose of this chapter is the presentation of techniques allowing the reader to reach a quantitative understanding of Figure 8.2.

This text is organized as follows. For the sake of simplicity, we focus in the first sections on the SAT problem. We start by introducing the useful notions of phase diagram, search trajectories, etc., in Section 8.2, and use these to analyze the search process in the low SAT phase where backtracking is essentially irrelevant. We turn to the opposite case of the UNSAT phase in Section 8.3, and develop tools necessary for the study of the action of DPLL in the presence of massive backtracking. Section 8.4 is devoted to the analysis of the upper SAT phase, that is, of the average-case complexity and large deviations from the latter. We show how all techniques presented in those Sections for SAT can be applied to COL in Section 8.5. Conclusions are presented in Section 8.6. Notice that, though most of the material presented in this chapter was previously published by the authors in various articles [15, 16, 20], Section 8.3.4 contains a new (and exact!) solution of the partial differential equation modeling the search tree growth.

8.2 Phase Diagram, Search Trajectories and the Easy SAT Phase

In this section, we present some useful concepts for the understanding of DPLL dynamics of search, and we apply them to investigate the low-ratio α regime, where a solution is rapidly found and the search tree essentially reduces to a single branch as shown in Figure 8.1(A). We start with some general comments on the dynamics induced by DPLL, and introduce the notion of mixed 2+p-SAT instance distribution. These concepts are made more precise and illustrated through the analysis of the single-branch trajectory, strongly inspired from some previous works by Chao and Franco [9] which the reader is referred to for more details. In the last Section 8.2.4, our numerical and analytical results for the solving complexity in the polynomial regime are presented.

Hereafter, the ratio of the 3-SAT instance to be solved will be denoted by α_0.

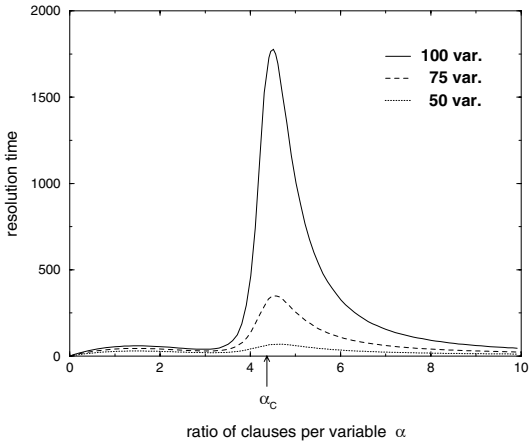

Figure 8.2: Resolution time of 3-SAT random instances by DPLL as a function of the ratio of clauses per variable α and for three different input sizes. Data correspond to the median resolution time of 10 000 instances; the average time may be somewhat larger due to the presence of rare, exceptionally hard instances. The computational complexity is maximal at α_c. It is exponential in the vicinity of the threshold and in the unsatisfiable phase, but less and less as α increases. A similar curve is obtained for random COL with α substituted with the ratio c of edges per vertex.

8.2.1 Overview of Concepts Useful to DPLL Analysis

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-clauses 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 [38], the threshold line $\alpha_C(p)$, separating SAT from UNSAT phases, may be estimated with the results shown in Figure 8.3. For $p \leq p_0 = 2/5$, i.e., to the left of point T, the threshold line is given by $\alpha_C(p) = 1/(1-p)$, as rigorously confirmed by [1], and coincides with the upper bound for the satisfaction of 2-clauses. Above p_0, no exact value for $\alpha_C(p)$ is known.

The phase diagram of 2+p-SAT is the natural space in which DPLL dynamics takes place. An input 3-SAT instance with ratio α shows up on the right vertical boundary of Figure 8.3 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, very much like real-space renormalization. This trajectory obviously depends on the heuristic of the split followed by DPLL. Possible simple heuristics are [8, 9],

- *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 (SCWM):* 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 (respectively $\bar{v}$) if $\ell > \bar{\ell}$ (resp. $\ell < \bar{\ell}$). When $\ell = \bar{\ell}$, v and $\bar{v}$ are equally likely to be chosen.

Rigorous mathematical analysis, undertaken to provide 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[2]. The corresponding search branch is drawn on Figure 8.1(A). These studies rely on the two following facts.

First, 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 all heuristics of split, but holds for the above examples (*UC*, *GUC*, *SCWM*) [8]. Analysis of more sophisticated heuristics requires the handling of more complex instance distributions [32].

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 [2, 46].

8.2.2 Clause Populations: Flows, Averages and Fluctuations

As pointed out above, 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 effort 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 8.4.

We call e_1, e_2, e_3 and w_2, w_1 the flows of clauses represented in Figure 8.4 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,

$$\begin{aligned} C_3(T+1) &= C_3(T) - e_3(T) - w_2(T) \\ C_2(T+1) &= C_2(T) - e_2(T) + w_2(T) - w_1(T) \\ C_1(T+1) &= C_1(T) - e_1(T) + w_1(T)\ . \end{aligned} \tag{8.1}$$

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 (8.1) 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 [9].

[2] The analysis of [24] however includes a very limited version of backtracking, see Section 8.2.2.

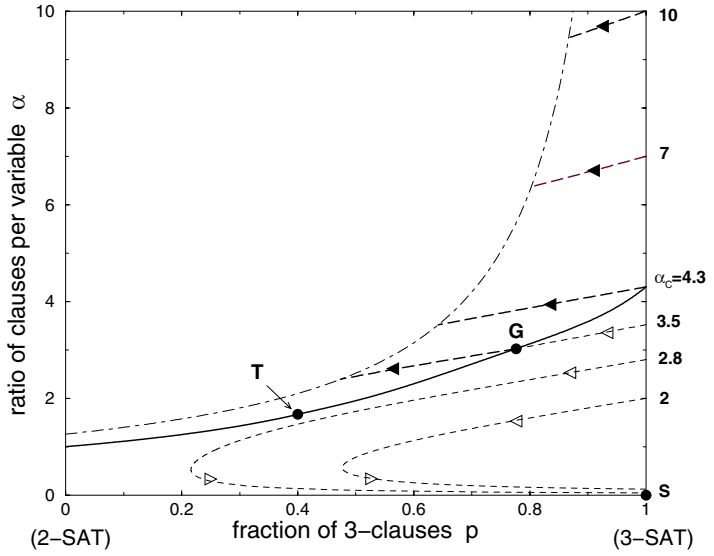

Figure 8.3: Phase diagram of 2+p-SAT and dynamical trajectories of DPLL. The threshold line $\alpha_C(p)$ (bold full line) separates SAT (lower part of the plane) from UNSAT (upper part) phases. Extremities lie on the vertical 2-SAT (left) and 3-SAT (right) axis at coordinates ($p = 0, \alpha_C = 1$) 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 parameterized 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. Once the dense tree is built, DPLL reaches 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 Figure 8.1.

As a result of the additivity of (8.1), 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,

$$C_j(T) = N\, c_j\left(\frac{T}{N}\right) + o(N) \qquad (j = 2, 3)\,. \tag{8.2}$$

where the c_j 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 densities of 2- and 3-clauses are self-averaging quantities and we shall attempt to calculate their mean values 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 unitary clauses has to vanish.

Formula (8.2) 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 unity, corresponds to the fast variations in 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 c_j of clauses, that is, their numbers divided by N, are changed by $O(1/N)$ only. The densities c_j 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 above noted, 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_j 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, called unitary clauses above, 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 $\mathbf{H}(C_1, C_1')$ and depend on t, c_2, c_3 only. $\mathbf{H}$ has a single eigenvector $\bar{\mu}(C_1)$ 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 $\bar{\mu}(C_1)$ of the master equation. Calculation of the equilibrium distribution $\bar{\mu}(C_1)$ of unit clauses will be sketched in Section 8.2.4.

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

8.2.3 Average-case Analysis in the Absence of Backtracking

In this section, we explain Chao and Franco's calculation of the densities of 2- and 3-clauses. Consider first the evolution equation (8.1) 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,

$$\frac{dc_3(t)}{dt} = -z_3(t)\,, \tag{8.3}$$

where $z_3 = \langle e_3 + w_2 \rangle$ denotes the averaged total outflow of 3-clauses (Section 8.2.2).

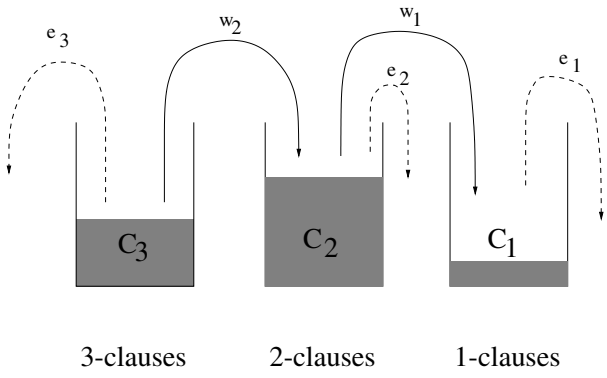

Figure 8.4: Schematic view of the dynamics of clauses. Clauses are sorted into three containers according to their lengths, i.e., the number of variables they include. Each time a variable is assigned by DPLL, clauses are modified, resulting in a dynamics of the container populations (lines with arrows). 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 containers are empty. The level of the rightmost container 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 container is vanishingly small. When the level is high (i.e., $O(\sqrt{N})$), contradictions will occur with high probability.

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 expression for z_3 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 $j\ (=1,2,3)$. Note that the sum of these probabilities is smaller than or equal to one, since we are free to choose the literal irrespective of the clause content (see UC case below). Extending the above discussion to 2-clauses, we obtain

$$\begin{aligned} \frac{dc_3(t)}{dt} &= -\frac{3}{1-t}c_3(t) - \rho_3(t) \\ \frac{dc_2(t)}{dt} &= \frac{3}{2(1-t)}c_3(t) - \frac{2}{1-t}c_2(t) - \rho_2(t)\ . \end{aligned} \tag{8.4}$$

In order to solve the above set of coupled differential equations, we need to know the probabilities ρ_j. As we shall see below, the values of the ρ_j depend on the heuristic of choice followed by DPLL and explained in Section 8.2.1. The solutions of the differential Equations (8.4) will then be expressed in terms of the fraction p of 3-clauses and the ratio α of

clauses per variable using the identities

$$p(t) = \frac{c_3(t)}{c_2(t) + c_3(t)} , \qquad \alpha(t) = \frac{c_2(t) + c_3(t)}{1 - t} . \tag{8.5}$$

8.2.3.1 Case of the GUC Heuristic

When DPLL is launched, 2-clauses are created with an initial flow $\langle w_2(0) \rangle = 3\,\alpha_0/2$. Let us first 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 container ($c_2 = 0$). Unitary clauses are almost absent and we have

$$\rho_1(t) = 0 \; ; \quad \rho_2(t) = \frac{3c_3(t)}{2(1-t)} \; ; \quad \rho_3(t) = 1 - \rho_2(t) \qquad (\alpha_0 < 2/3) . \tag{8.6}$$

The solutions of (8.4) with the initial condition $p(0) = 1, \alpha(0) = \alpha_0$ read

$$\begin{aligned} p(t) &= 1 , \\ \alpha(t) &= (\alpha_0 + 2)(1-t)^{1/2} - 2 . \end{aligned} \tag{8.7}$$

Solution (8.7) confirms that the instance never contains an extensive number of 2-clauses. At some final time t_{end}, depending on the initial ratio, $\alpha(t_{end})$ vanishes: no clause is left and a solution is found.

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 the presence of 1- or 2-clauses, a literal is never chosen in a 3-clause. We show in Section 8.2.4 that the probability that there is no 1-clause at some stage of the procedure equals $1 - c_2(t)/(1-t)$. This probability coincides with $\rho_2(t)$ and, thus, we have

$$\rho_1(t) = \frac{c_2(t)}{1-t} \; ; \quad \rho_2(t) = 1 - \rho_1(t) \; ; \quad \rho_3(t) = 0 \qquad (\alpha_0 > 2/3) , \tag{8.8}$$

as soon as $t > 0$. The solutions of (8.4) now read

$$\begin{aligned} p(t) &= \frac{4\alpha_0(1-t)^2}{\alpha_0(1-t)^2 + 3\alpha_0 + 4\ln(1-t)} , \\ \alpha(t) &= \frac{\alpha_0}{4}(1-t)^2 + \frac{3\alpha_0}{4} + \ln(1-t) . \end{aligned} \tag{8.9}$$

Solution (8.9) 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 (8.8) 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 reduces to the $\alpha_0 < 2/3$ case.

We show in Figure 8.3 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 (creation of 2-clauses), and

then reverses to the right (2-clause destruction results from splits) 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 (Section 8.2.3.1). 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 $\alpha_T = 1/(1-p_T)$.

8.2.3.2 Case of UC and SCWM Heuristics

The above study can be extended to the other heuristics presented in Section 8.2.1. For UC and SCWM, the probability $\rho_3(t)$ that a variable is chosen from a 3-clause vanishes for all positive times. The set of ODEs (8.4) is thus entirely defined from the expression of the probability ρ_2 that a variable is set through splitting of a clause,

$$\rho_2(t) = \left[1 - \frac{c_2(t)}{1-t}\right] \; h(t) \; . \tag{8.10}$$

Function h depends upon the heuristic:

- $h_{UC}(t) = 0$;
- $h_{SCWM}(t) = 3a_3\, e^{-3a_3}\, (I_0(3a_3) + I_1(3a_3))/2$ where $a_3 \equiv c_3(t)/(1-t)$ and I_ℓ is the ℓ^{th} modified Bessel function.

The reader is referred to Ref. [2, 22] for additional information.

8.2.4 Occurrence of Contradictions and Polynomial SAT Phase

In this section, we compute the computational complexity in the range $0 \leq \alpha_0 \leq \alpha_L$ from the previous results. To avoid unnecessary repetitions, we specialize to the case of the GUC heuristic.

The trajectories obtained in Section 8.2.3 represent the deterministic evolution of the densities of 2- and 3-clauses when more and more variables are assigned. Below, we briefly present the calculation[3] of the distribution $\bar{\mu}(C_1, t)$ of the number C_1 of 1-clauses at reduced time t done by Frieze and Suen [24]. Call $\mathbf{H}(C_1, C_1')$ the probability that the number of unit-clauses goes from C_1' to C_1 once a variable is fixed by DPLL, see Section 8.2.2. In the limit of large size N, the entries of matrix $\mathbf{H}$ depend on the reduced time t and the average density c_2 of 2-clauses only,

$$\mathbf{H}(C_1, C_1') = \sum_{w_1 \geq 0} e^{-a_2}\, \frac{{a_2}^{w_1}}{w_1!}\, \delta_{C_1 - C_1' + w_1 - \sigma(C_1')} \tag{8.11}$$

where $a_2 \equiv c_2/(1-t)$, w_1 is the creation flow of unit-clauses represented in Figure 8.4, and $\sigma(C_1') = 1$ if $C_1' \geq 1$, 0 if $C_1' = 0$. The evolution matrix $\mathbf{H}$ ensures probability conservation

[3] This calculation, combined with the study of the large deviations of the densities of 2- and 3-clauses, is closely related to the more complex analysis of the UNSAT phase presented in Section 8.3.

since entries along any column sum up to one. $\bar{\mu}(C_1)$, the eigenvector associated to the largest eigenvalue (equal to unity), represents the stationary distribution of the number C_1 of unit-clauses. In terms of the generating function μ of $\bar{\mu}$ [21],

$$\mu(y_1) = \sum_{C_1 \geq 0} \bar{\mu}(C_1)\, e^{y_1\, C_1}\ , \tag{8.12}$$

the eigenvalue equation reads

$$\mu(y_1) = (1 - a_2)\, \frac{e^{\nu(y_1)}\, (1 - e^{y_1})}{e^{\nu(y_1)} - 1}\ , \quad \text{where } \nu(y_1) \equiv -y_1 - a_2 + a_2\, e^{y_1}\ . \tag{8.13}$$

Sending $y_1 \to -\infty$ in (8.13) permits us to find the probability that there is no unitary clause, $\bar{\mu}(C_1 = 0) = 1 - a_2$. This probability gives the value of $\rho_2(t)$ used in the derivation of the branch trajectory of Section 8.2.3 in the $\alpha_0 > 2/3$ case.

The pole Y_1 of μ, that is, the non-vanishing zero of ν, controls the asymptotic behavior of the probability of the number of unit-clauses: $\bar{\mu}(C_1) \asymp e^{-Y_1\, C_1}$ when $C_1 \to \infty$. As long as $a_2 < 1$, Y_1 is positive, and $\bar{\mu}$ is localized. The average number of unit-clauses,

$$\langle C_1 \rangle = \sum_{C_1 \geq 0} \bar{\mu}(C_1)\, C_1 = \frac{d\mu}{dy_1}(y_1 = 0) = a_2\, \frac{2 - a_2}{2(1 - a_2)} \tag{8.14}$$

is finite. As a result, unit-clauses do not accumulate too much, and the probability that a contradiction occurs when a new variable is assigned is $O(1/N)$ only.

To calculate this probability, we consider step number T of DPLL. There are $V = N - T = N(1 - t)$ not-yet-assigned variables, and C_1 unit-clauses, a stochastic number drawn from distribution $\bar{\mu}$ with $a_2 = c_2(t)/(1 - t)$. In the absence of a unit-clause, the next variable is assigned through splitting of a clause and no immediate contradiction is possible. Otherwise, DPLL picks up a unit-clause and satisfies it. The probability that another given unit-clause is contradictory is $p = 1/(2V)$. Since clauses are independent, the probability that no contradiction emerges during step T is,

$$\mathrm{Prob}\,(\mathrm{T\ to\ T+1}) = \left(1 - \frac{1}{2(N - T)}\right)^{C_1 - 1}\ , \tag{8.15}$$

The probability that a contradiction never occurs till step $T = t\,N$ is therefore [24]:

$$\mathrm{Prob}\,(0\ \mathrm{to\ T = t\,N}) = \exp\left(-\int_0^t dt\, \frac{\langle \max(C_1 - 1, 0)\rangle(t)}{2\,(1 - t)}\right)\ . \tag{8.16}$$

This expression, combined with (8.14) gives the probability that a contradiction arises along the branch trajectory calculated in Section 8.2.3. Two cases can be distinguished:

- If the ratio α_0 of clauses per variable is smaller than $\alpha_L \simeq 3.003$, $a_2(t)$ remains strictly smaller than unity, and the probability of success (8.16) is positive. Frieze and Suen have shown that contradictions have no dramatic consequences. The number of total 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 8.2.3, and the increase in complexity is negligible with respect to $O(N)$.

- When $\alpha_0 > \alpha_L$, the trajectory intersects the $\alpha = 1/(1-p)$ line at some time t. At this point, $a_2(t) = \alpha(1-p) = 1$: the average number of unit-clauses, $\langle C_1 \rangle$, diverges. Unit-clauses accumulate, and contradictions unavoidably arise. Backtracking enters massively into play, signaling the crossover to the exponential regime.

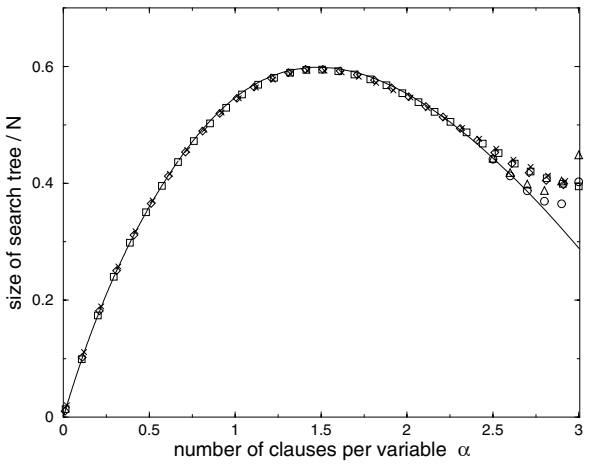

Figure 8.5: Complexity of solution in the SAT region for $\alpha < \alpha_L \simeq 3.003$, divided by the size N of the instances. Numerical data are for sizes $N = 50$ (crosses), 75 (squares), 100 (diamonds), 500 (triangles) and 1000 (circles). For the two biggest sizes, simulations have been carried 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 8.2.4. Note the perfect agreement with numerics except at large ratios where finite size effects are important, due to the crossover to the exponential regime above $\alpha_L \simeq 3.003$.

From the above discussion, it appears that a solution is found by DPLL essentially at the end of a single descent (Figure 8.1(A)) when $\alpha_0 < \alpha_L$ (lower SAT phase). 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. Thus, DPLL assigns the overwhelming majority of variables through splittings. $\gamma(\alpha_0)$ simply equals the total fraction t_{end} of variables chosen by DPLL. From (8.7), we obtain

$$\gamma(\alpha_0) = 1 - \frac{4}{(\alpha_0+2)^2} \qquad (\alpha_0 \leq 2/3)\ . \tag{8.17}$$

For larger ratios, i.e., $\alpha_0 > 2/3$, the trajectory must be decomposed into two successive portions. 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 dt of variables, a fraction $\rho_1(t) = \alpha(t)(1-p(t))dt$ are fixed by unit-propagation only. The number of nodes

(divided by N) along the first part of the branch thus reads,

$$\gamma_1 = t^* - \int_0^{t^*} dt\ \alpha(t)(1-p(t))\ . \tag{8.18}$$

At time t^*, the trajectory touches the 3-SAT axis $p = 1$ at ratio $\alpha^* \equiv \alpha(t^*) < 2/3$. The initial instance is then reduced 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

$$\gamma_2 = t_{end} - t^*\ . \tag{8.19}$$

It proves 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 (8.9) leads to

$$\begin{aligned}
\alpha(t^*) &= -\frac{4\ln(1-t^*)}{3t^*(2-t^*)} \\
\gamma(t^*) &= 1 - \frac{4(1-t^*)}{(2+(1-t^*)^2\alpha_0(t^*))^2} + t^* + (1-t^*)\ln(1-t^*) \\
&\quad -\frac{1}{4}\,\alpha(t^*)\,(t^*)^2\,(3-t^*)\ .
\end{aligned} \tag{8.20}$$

As t^* grows from zero to $t_L^* \simeq 0.892$, the initial ratio α_0 spans the range $[2/3; \alpha_L]$. The complexity coefficient $\gamma(\alpha_0)$ can be computed from (8.17) and (8.20) with the results shown in Figure 8.5. The agreement with numerical data is excellent.

8.3 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 the presence of massive backtracking. We first report results from numerical experiments, then explain our analytical approach for computing the complexity of resolution (size of search tree).

8.3.1 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. Figure 8.1(B) 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 are not arranged along a line any longer, but rather form a cloud with a finite extension in the phase diagram of Figure 8.3. Examples of clouds are provided in Figure 8.6.

The number of points in a cloud, i.e., the size Q of its associated search tree, grows exponentially with N [11]. It is thus convenient to define its logarithm ω through $Q = 2^{N\omega}$. We directly counted Q experimentally, and averaged the corresponding logarithm ω over a

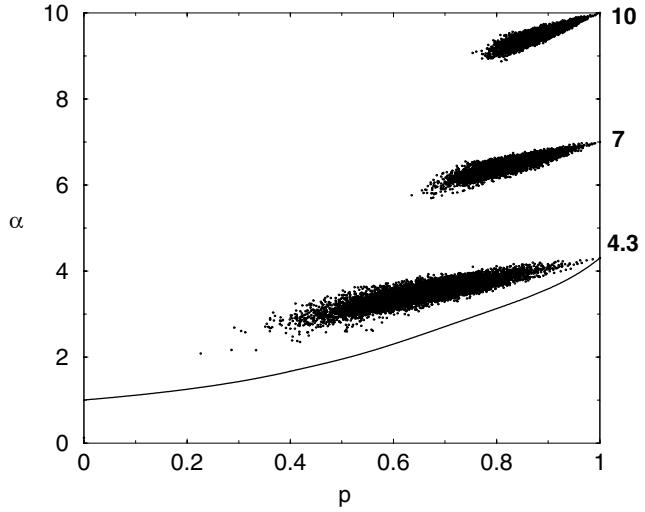

Figure 8.6: Clouds associated to search trees obtained from the resolution of three UNSAT instances with initial ratios $\alpha_0 = 4.3, 7$ and 10 respectively. Each point in the cloud corresponds to a splitting node in the search tree. Sizes of instances and search trees are $N = 120, Q = 7597$ for $\alpha_0 = 4.3$, $N = 200, Q = 6335$ for $\alpha_0 = 7$, and $N = 300, Q = 6610$ for $\alpha_0 = 10$.

Table 8.1: Logarithm of the complexity ω from experiments (EXP), theory (THE) from Section 8.3.4 and former linearization approximation (LIN) [13], as a function of the ratio α_0 of clauses per variable of the 3-SAT instance. Ratios above 4.3 correspond to UNSAT instances; the rightmost ratio lies in the upper SAT phase.

α_0	4.3	7	10	15	20	3.5
ω_{EXP}	0.089 ± 0.001	0.0477 ± 0.0005	0.0320 ± 0.0005	0.0207 ± 0.0002	0.0153 ± 0.0002	0.034 ± 0.003
ω_{THE}	0.0916	0.0486	0.0323	0.0207	0.0153	0.035
ω_{LIN}	0.0875	0.0477	0.0319	0.0206	0.0152	0.035

large number of instances. Results have then been extrapolated to the $N \to \infty$ limit [13] and are reported in Table 8.1. ω is a decreasing function of α_0 [7]: 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 word "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.

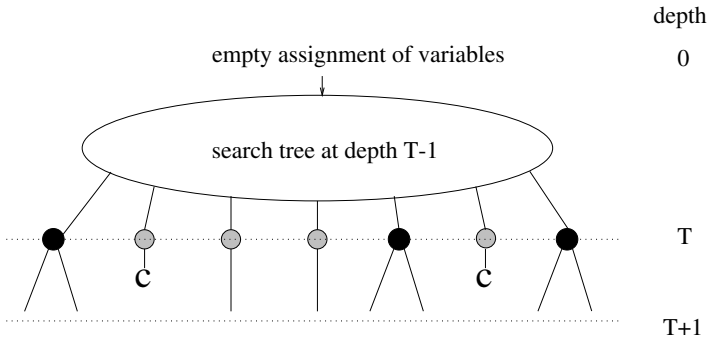

Figure 8.7: Imaginary, parallel growth process of an UNSAT search tree used in the theoretical analysis. Variables are fixed through unit propagation, or by the splitting heuristic as in the DPLL procedure, but branches evolve in parallel. T denotes the depth in the tree, that is the number of variables assigned by DPLL along each branch. At depth T, one literal is chosen on each branch among 1-clauses (unit propagation, grey circles not represented on Figure 8.1), or 2,3-clauses (splitting, black circles as in Figure 8.1). 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.

8.3.2 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 Figure 8.1(B) 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, breadth-first building up of the refutation tree, which results in the same complete tree but can be mathematically analyzed. In our imaginary process, the tree grows in parallel, layer after layer (Figure 8.7). At time $T = 0$, the tree reduces to a root node, to which is attached the empty assignment of variables (nothing is known at the beginning of the search process), 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 $B(T)$ ($\leq 2^T$) branches, each one carrying a partial assignment of variables. At next time step $T \rightarrow T+1$, 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 evolution operator $\mathbf{H}$. A detailed definition and construction of $\mathbf{H}$ is presented in [16]. We limit ourselves to explaining hereafter the main steps:

- A 3^N dimensional-vector space is introduced. Each vector $|S\rangle$ in the spanning basis is in one-to-one correspondence with a partial assignment $S = (s_1, s_2, \ldots, s_N)$ of the N variables ($s_i = t$, f, u if variable x_i is, respectively, True, False, or Undetermined, i.e., not-yet-assigned).

- Let S be a partial assignment which does not violate the (unsatisfiable) instance $\mathbf{I}$ under consideration, and $S^{(j,x)}$, with $j = 1, \ldots, N$ and $x = t, f$, the partial assignment obtained from S by replacing s_j with x. Call $h_n(j|S)$ and $h_v(x|S, j)$ the probabilities that the heuristic (UC, GUC, ...) respectively chooses to assign variable x_j, and to fix it to $x\,(= t, f)$.

- The evolution operator $\mathbf{H}$ encodes the action of DPLL on $\mathbf{I}$. Its matrix elements in the spanning basis are, see Figure 8.8,

  1. If S violates $\mathbf{I}$, $\langle S'|\mathbf{H}|S\rangle = 1$ if $S' = S$, 0 otherwise.
  2. If S does not violate $\mathbf{I}$, $\langle S'|\mathbf{H}|S\rangle = h_n(j|S) \times h_v(x|S, j)$ if $C_1(S) \geq 1$ and $S' = S^{(j,x)}$, $h_n(j|S)$ if $C_1(S) = 0$ and ($S' = S^{(j,x)}$ or $S' = S^{(j,\bar{x})}$), 0 otherwise. Here S, S' are the partial assignments corresponding to $|S\rangle, |S'\rangle$, and $C_1(S)$ the number of undetermined clauses of type 1 (unitary clauses) for partial assignment S.

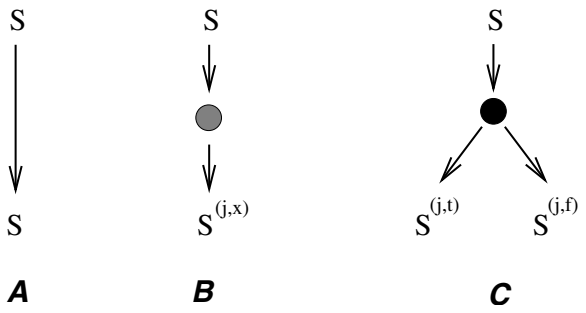

Figure 8.8: Transitions allowed by the heuristic-induced evolution operator. Grey and black nodes correspond to variables assigned through unit-propagation and split respectively, as in Figure 8.7. A. If partial assignment S already violates the instance $\mathbf{I}$, it is left unchanged. B. If the partial assignment does not violate $\mathbf{I}$ and there is at least one unitary clause, a variable is fixed through unit propagation (grey node) e.g. $x_j = x$. The output partial assignment is $S^{j,x}$. C. If the partial assignment does not violate $\mathbf{I}$ and there is no unitary clause, a variable x_j is fixed through splitting (black node). Two partial assignments are generated, $S^{j,t}$ and $S^{j,f}$.

Then, the expectation value over the random assignments of variables of the size (number of leaves) of the search tree produced by DPLL to refute $\mathbf{I}$, is equal to

$$B = \sum_S \langle S|\mathbf{H}^N|u, u, \ldots, u\rangle \;, \tag{8.21}$$

where $\mathbf{H}^N$ denotes the N^{th} (matrical) power of $\mathbf{H}$, the sum runs over all 3^N partial assignments S, and the rightmost vector corresponds to the initial, fully undetermined assignment of variables [16].

Calculation of the expectation value of the N^{th} power of $\mathbf{H}$, and of its average over the instance distribution is a hard task. We therefore turned to a simplifying approximation, called dynamical annealing. Call clause vector $\vec{C}(S)$ of a partial assignment S the three-dimensional

vector $\vec{C} = (C_1, C_2, C_3)$ where C_j is the number of undetermined clauses of length j. The quantity we focus on is $\bar{B}(\vec{C}; T+1)$, the expectation number of branches at depth T in the search tree (Figure 8.7) carrying partial assignments 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,

$$\bar{B}(\vec{C}; T+1) = \sum_{\vec{C}'} \bar{\mathbf{H}}\,[\vec{C}, \vec{C}'; T]\; \bar{B}(\vec{C}'; T)\;. \tag{8.22}$$

The entries of the evolution matrix $\bar{\mathbf{H}}[\vec{C}, \vec{C}'; T]$ can be calculated from the definition of the evolution matrix $\mathbf{H}$ [16]. They 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 of variables with clause vector $\vec{C}'$.

For the GUC heuristic, we find [13],

$$\begin{aligned}
\bar{\mathbf{H}}[\vec{C}, \vec{C}'; T] = {} & \binom{C_3'}{C_3' - C_3} \left(\frac{3}{N-T}\right)^{C_3' - C_3} \left(1 - \frac{3}{N-T}\right)^{C_3} \times \\
& \sum_{w_2=0}^{C_3' - C_3} \left(\frac{1}{2}\right)^{C_3' - C_3} \binom{C_3' - C_3}{w_2} \times \\
& \Bigg\{ (1 - \delta_{C_1'}) \left(1 - \frac{1}{2(N-T)}\right)^{C_1' - 1} \sum_{z_2=0}^{C_2'} \binom{C_2'}{z_2} \left(\frac{2}{N-T}\right)^{z_2} \times \\
& \left(1 - \frac{2}{N-T}\right)^{C_2' - z_2} \sum_{w_1=0}^{z_2} \left(\frac{1}{2}\right)^{z_2} \binom{z_2}{w_1} \delta_{C_2 - C_2' - w_2 + z_2}\, \delta_{C_1 - C_1' - w_1 + 1} + \\
& \delta_{C_1'} \sum_{z_2=0}^{C_2' - 1} \binom{C_2' - 1}{z_2} \left(\frac{2}{N-T}\right)^{z_2} \left(1 - \frac{2}{N-T}\right)^{C_2' - 1 - z_2} \times \\
& \sum_{w_1=0}^{z_2} \left(\frac{1}{2}\right)^{z_2} \binom{z_2}{w_1} \delta_{C_2 - C_2' - w_2 + z_2 + 1}\, [\delta_{C_1 - w_1} + \delta_{C_1 - 1 - w_1}] \Bigg\}\;,
\end{aligned} \tag{8.23}$$

where δ_X denotes the Kronecker delta function over integers X: $\delta_X = 1$ if $X = 0$, $\delta_X = 0$ otherwise. Expression (8.23) is easy to obtain from the interpretation following Eq. (8.22), and the picture of containers in Figure 8.4 [13]. Among the $C_3' - C_3$ clauses that flow out from the leftmost 3-clauses container, 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. It is 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. Variables are chosen among 1-clauses or, in the absence of the latter, among 2-clauses. The term on the r.h.s. of Eqn. (8.23) 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 container; 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-clause container is, at the same time, poured with w_2 clauses. In an analogous way, the unitary clause container 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 container 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. This probability enters as a multiplicative factor in the third line of (8.23). 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 in 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 8.3.2.

8.3.3 Generating Function and Large-size Scaling

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

$$G(\,\vec{y}\,;T\,) = \sum_{\vec{C}} e^{\,\vec{y}\cdot\vec{C}}\; \bar{B}(\,\vec{C}\,,T\,)\;, \quad \vec{y}\cdot\vec{C} \equiv \sum_{j=1}^{3} y_j\, C_j\;, \tag{8.24}$$

where the first sum runs over all triplets of positive clause numbers. The evolution equation (8.22) for the $\bar{B}$s can be rewritten in term of the generating function G,

$$\begin{aligned} G(\,\vec{y}\,;T+1\,) \quad &= e^{-\gamma_1(\vec{y})}\; G\big(\,\vec{\gamma}(\vec{y})\,;T\,\big) + \\ &\Big(e^{-\gamma_2(\vec{y})}(e^{y_1}+1) - e^{-\gamma_1(\vec{y})}\Big)\; G\big(-\infty,\,\gamma_2(\vec{y}),\,\gamma_3(\vec{y})\,;T\,\big) \end{aligned} \tag{8.25}$$

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

$$\begin{aligned} \gamma_1(\vec{y}) &= y_1 + \ln\left[1 - \frac{1}{2(N-T)}\right]\;, \\ \gamma_2(\vec{y}) &= y_2 + \ln\left[1 + \frac{2}{N-T}\left(\frac{e^{-y_2}}{2}\,(1+e^{y_1}) - 1\right)\right]\;, \\ \gamma_3(\vec{y}) &= y_3 + \ln\left[1 + \frac{3}{N-T}\left(\frac{e^{-y_3}}{2}\,(1+e^{y_2}) - 1\right)\right]\;. \end{aligned} \tag{8.26}$$

To solve (8.25), 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 $T_1 = t\,N$ to $T_2 = t\,N + \Delta T$ 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 [24] for the analysis of the GUC heuristic in the absence of backtracking.

3. As long as a partial assignment 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 becomes large.

The above arguments entice us to make the following claim. For large N, T at fixed ratio $t = T/N$, the generating function (8.24) of the average numbers $\bar{B}$ of branches is expected[4] to behave as

$$G(y_1, y_2, y_3; t\,N) = \exp\Big[\, N\,\varphi(y_2, y_3; t) + \psi(y_1, y_2, y_3; t) + o(1)\Big]\,. \tag{8.27}$$

Hypothesis (8.27) 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,

$$\omega(\,c_2, c_3\,;t\,) = \min_{y_2,y_3}\Big[\varphi(\,y_2, y_3\,;t\,) - y_2\,c_2 - y_3\,c_3\Big]\,. \tag{8.28}$$

Then, combining equations (8.24), (8.27) and (8.28), we obtain

$$\lim_{N\to\infty}\frac{1}{N}\ln\,\bar{B}(C_1, c_2\,N, c_3\,N; t\,N) = \omega(c_2, c_3; t\,)\,, \tag{8.29}$$

independently of the (finite) number C_1 of unit clauses. In other words, the expectation value of the number of branches carrying partial assignments 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 (8.28). 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 φ: $c_j(t) = \partial\varphi/\partial y_j(0, 0)$ for $j = 2, 3$. Let us emphasize that φ in (8.27) does not depend on y_1. This hypothesis simply expresses that, as far as non violating partial assignments are concerned, both terms on the right-hand side of (8.25) 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 the equilibrium distribution $\bar{\mu}(C_1|c_2, c_3, t)$ of unit-clause at fixed c_2, c_3, t, extending the definition and calculation of Section 8.2.4 valid in the absence of backtracking to the UNSAT regime,

$$e^{\psi(y_1,y_2,y_3;t)-\psi(0,y_2,y_3;t)} = \sum_{C_1\geq 0}\bar{\mu}(C_1|c_2, c_3, t)\; e^{\,y_1\,C_1}\,, \tag{8.30}$$

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

[4] See [29] for a similar large deviation ansatz in the context of the relaxation dynamics of the mean-field Ising model.

Inserting expression (8.27) into the evolution equation, (8.25), we find

$$\begin{aligned}\frac{\partial\varphi}{\partial t}(y_2,y_3;t) &= -y_1+\frac{2}{1-t}\left[e^{-y_2}\left(\frac{1+e^{y_1}}{2}\right)-1\right]\frac{\partial\varphi}{\partial y_2}(y_2,y_3;t)\\ &+ \frac{3}{1-t}\left[e^{-y_3}\left(\frac{1+e^{y_2}}{2}\right)-1\right]\frac{\partial\varphi}{\partial y_3}(y_2,y_3;t)\\ &+ \ln\left[1+\mathcal{K}(y_1,y_2)\,e^{\psi(-\infty,y_2,y_3;t)-\psi(y_1,y_2,y_3;t)}\right] \qquad (8.31)\end{aligned}$$

where $\mathcal{K}(y_1,y_2)=e^{-y_2}(e^{2\,y_1}+e^{y_1})-1$. As φ does not depend upon y_1, the latter may be chosen at our convenience, e.g., to cancel $\mathcal{K}$ and the contribution from the last term in (8.31),

$$y_1=Y_1(y_2)\equiv y_2-\ln\left(\frac{1+\sqrt{1+4\,e^{y_2}}}{2}\right)\ . \qquad (8.32)$$

Such a procedure, similar to the kernel method [33], is correct in the major part of the y_2,y_3 space and, in particular, in the vicinity of $(0,0)$ which we focus on in this paper[5]. We end up with the following partial differential equation (PDE) for φ,

$$\frac{\partial\varphi}{\partial t}(y_2,y_3;t)=H\left[\frac{\partial\varphi}{\partial y_2},\frac{\partial\varphi}{\partial y_3},y_2,y_3,t\right]\ , \qquad (8.33)$$

where H incorporates the details of the splitting heuristic[6],

$$\begin{aligned}H_{GUC}[c_2,c_3,y_2,y_3,t] &= -Y_1(y_2)+\frac{3\,c_3}{1-t}\left[e^{-y_3}\left(\frac{1+e^{y_2}}{2}\right)-1\right]\\ &+ \frac{c_2}{1-t}\left(e^{-Y_1(y_2)}-2\right)\ . \qquad (8.35)\end{aligned}$$

We must therefore solve the partial differential equation (PDE) (8.33) with the initial condition,

$$\varphi(y_2,y_3,t=0)=\alpha_0\ y_3\ , \qquad (8.36)$$

obtained through inverse Legendre transform (8.28) of the initial condition over $\bar{B}$, or equivalently over ω,

$$\omega(c_2,c_3;t=0)=\left\{\begin{array}{ccc}0 & \text{if} & c_3=\alpha_0\ ,\\ -\infty & \text{if} & c_3\neq\alpha_0\ .\end{array}\right.$$

[5] 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 in [13].

[6] For the UC heuristic,

$$H_{UC}=\ln 2+\frac{3\,c_3}{1-t}\left[e^{-y_3}\left(\frac{1+e^{y_2}}{2}\right)-1\right]+\frac{c_2}{1-t}\left(\frac{3}{2}e^{-y_2}-2\right)\ . \qquad (8.34)$$

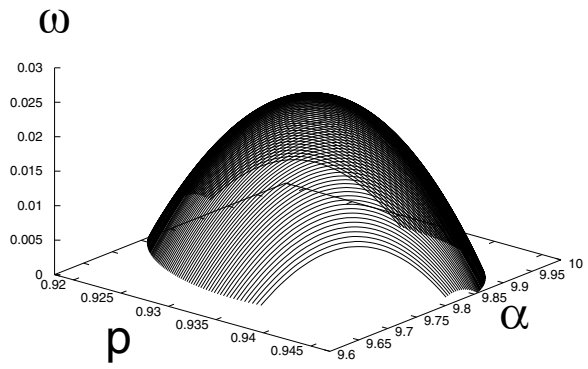

Figure 8.9: Snapshot of the surface $\omega(p, \alpha; t)$ for $\alpha_0 = 10$ at time (depth in the tree) $t = 0.05$. The height $\omega^*(t)$ of the top of the surface, with coordinates $p^*(t), \alpha^*(t)$, is the logarithm (divided by N) of the number of branches. The coordinates $(p^*(t), \alpha^*(t))$ define the tree trajectory shown in Figure 8.3. The halt line is hit at $t_h \simeq 0.094$.

8.3.4 Interpretation in Terms of Growth Process

We can interpret the dynamical annealing approximation made in the previous paragraphs, and the resulting PDE (8.33) as a description of the growth process of the search tree resulting from DPLL operation. Using Legendre transform (8.28), PDE (8.33) 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,

$$\frac{\partial \omega}{\partial t}(c_2, c_3, t) = H\left[c_2, c_3, -\frac{\partial \omega}{\partial c_2}, -\frac{\partial \omega}{\partial c_3}, t\right] . \tag{8.37}$$

Partial differential equation (PDE) (8.37) is analogous to growth processes encountered in statistical physics [36]. The surface ω, growing with "time" t above the plane c_2, c_3, or equivalently from (8.5), above the plane p, α (Figure 8.9), describes the whole distribution of branches. The average number of branches at depth t in the tree equals

$$B(t) = \int_0^1 dp \int_0 d\alpha \; e^{N\,\omega(p,\alpha;t)} \asymp e^{N\,\omega^*(t)} , \tag{8.38}$$

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 $c_2^*(t) = \alpha^*(t)(1 - p^*(t))$, $c_3^*(t) = \alpha^*(t)p^*(t)$. Along the tree trajectory, $\omega^*(t)$ grows thus from 0, on the right vertical axis, up to some halting time t_h, at which dominant branches almost surely get hit by con-

tradictions. $\omega_{THE} = \omega^*(t_h)$ is our theoretical prediction for the logarithm of the complexity (divided by N)[7].

The hyperbolic line in Figure 8.3 indicates the halt points, where contradictions prevent dominant branches from further growing [13]. To obtain this curve, we calculate the probability $\bar{\mu}^*(t) \equiv \bar{\mu}(C_1 = 0 | c_2^*(t), c_3^*(t), t)$ that a split occurs when a variable is assigned by DPLL,

$$\bar{\mu}^*(t) = \exp\left(\frac{\partial\varphi}{\partial t}(0,0;t)\right) - 1 \,, \tag{8.39}$$

from (8.30) with $y_1 = -\infty, y_2 = y_3 = 0$ and (8.31) with $y_1 = y_2 = y_3 = 0$, respectively. The probability of split vanishes, and unit-clauses accumulate until a contradiction is obtained, when the tree stops growing,

$$\bar{\mu}^*(t) = 0 \rightarrow \frac{\partial\varphi}{\partial t}(0,0;t) = \frac{\partial\omega}{\partial t}\big(c_2^*(t), c_3^*(t); t\big) = 0 \,. \tag{8.40}$$

From PDEs (8.33) or (8.37), this halting condition corresponds to crossing of

$$\alpha = \left(\frac{3+\sqrt{5}}{2}\right) \ln\left[\frac{1+\sqrt{5}}{2}\right] \; \frac{1}{1-p} \,. \tag{8.41}$$

Notice that the halt line in the UNSAT phase differs from the halt line $\alpha = 1/(1-p)$ calculated in Section 8.2.4 in the absence of backtracking.

Equation (8.37) is a first-order PDE[8] and can be solved using the characteristics method [35]. The idea is to describe the surface $\omega(c_2, c_3; t)$ as the union of curves, representing the evolution of a 'particle' in a 3-dimensional space with coordinates $c_2(t), c_3(t), \omega(t)$. Denoting by $p_j(t)$ the partial derivative $\partial\omega/\partial c_j(c_2(t), c_3(t); t)$ ($j = 2, 3$), we write the conditions fulfilled by the particle to sit on the surface at any time as a set of five first-order ordinary coupled differential equations,

$$\begin{aligned}
\frac{dp_j}{dt}(t) &= -\frac{\partial H}{\partial c_j}\Big[c_2(t), c_3(t), -p_2(t), -p_3(t), t\Big] \,, \quad (j = 2,3) \\
\frac{dc_j}{dt}(t) &= \frac{\partial H}{\partial p_j}\Big[c_2(t), c_3(t), -p_2(t), -p_3(t), t\Big] \,, \quad (j = 2,3) \\
\frac{d\omega}{dt}(t) &= H\Big[c_2(t), c_3(t), -p_2(t), -p_3(t), t\Big] + \sum_{j=2,3} p_j(t)\; \frac{dc_j}{dt}(t) \,.
\end{aligned} \tag{8.42}$$

Assume now that we focus on dominant branches only, and want to calculate the coordinates c_2^*, c_3^*, ω^* of the top of the surface as a function of time, say t', positive and smaller than the halt time. To do this, we need to solve (8.42) for $0 < t < t'$ with boundary conditions,

$$c_2(0) = 0 \,, \quad c_3(0) = \alpha_0 \,, \quad \omega(0) = 0 \,, \tag{8.43}$$

[7] 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 8.1.

[8] This statement is correct in the large-size limit only. Finite-size corrections would introduce second-derivative terms with $1/N$ multiplicative coefficients. See [29] for a similar situation.

which expresses that all trajectories describe resolution of a 3-SAT instance with ratio α_0, and

$$p_2(t') = p_3(t') = 0 \ , \tag{8.44}$$

to match the end of the trajectory with the top of the surface ω at time t'. Numerical resolution of equations (8.42) with boundary conditions (8.43,8.44) is shown in Figure 8.10. Clause densities c_2, c_3 keep positive at any time as expected. The parametric plot of the final coordinates, $p^*(t'), \alpha^*(t')$, as a function of t' defines the tree trajectories on Figure 8.3. Values of ω obtained for various initial ratios α_0 are listed in Table 8.1, and compared to the linearization approximation developed in [13].

We have plotted the surface ω at different times, with the results shown in Figure 8.9 for $\alpha_0 = 10$. Values of ω_{THE}, obtained for $4.3 < \alpha < 20$ by solving (8.37) compare very well with the numerical results (Table 8.1). Although our calculation is not rigorous, it provides a very good quantitative estimate of the complexity. It is therefore expected that our dynamical annealing approximation is quantitatively accurate. It is a reasonable conjecture that it becomes exact at large ratios α_0, where PDE (8.33) can be exactly solved.

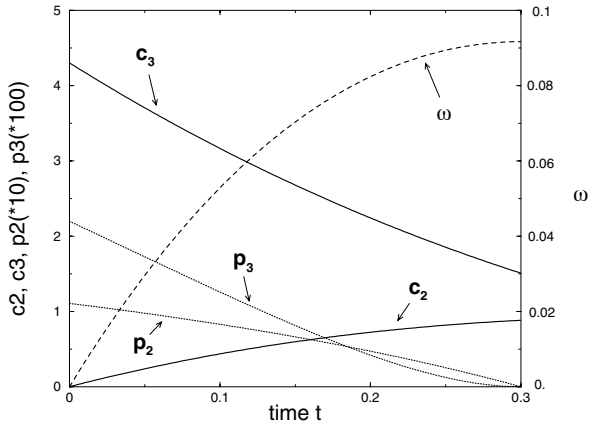

Figure 8.10: Clause densities $c_2(t), c_3(t)$, logarithm $\omega(t)$ of the number of branches, local derivatives $p_2(t), p_3(t)$ as a function of time t for the characteristic curve reaching the top surface at halt time $t_h \simeq 0.299$ corresponding to ratio $\alpha_0 = 4.3$. Left axis scale corresponds to c_2, c_3 and $p_2 \times 10, p_3 \times 100$; right axis scale shows values of ω. Boundary conditions (8.43,8.44) can be seen from the curves. The vanishing of the derivative of ω at $t = t_h$ stems from the halt condition (8.40).

8.3.4.1 Asymptotic Equivalent of ω for Large Ratios

Resolution of PDE (8.37) in the large ratio α_0 limit gives (for the GUC heuristic),

$$\omega_{THE}(\alpha_0) \asymp \frac{3+\sqrt{5}}{6\,\ln 2} \left[\ln\left(\frac{1+\sqrt{5}}{2}\right)\right]^2 \frac{1}{\alpha_0} \ . \tag{8.45}$$

This result exhibits the $1/\alpha_0$ scaling proven by [7], and is conjectured to be exact. As α_0 increases, search trees become smaller and smaller, and correlations between branches, weaker and weaker, making dynamical annealing increasingly accurate.

8.4 Hard SAT Phase: Average Case and Fluctuations

8.4.1 Mixed Branch and Tree Trajectories

The main 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 [14], see search tree in Figures 8.1(C) and 8.11.

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 $N\ t_G$ variables have been assigned by DPLL, see Figure 8.12. 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 up until the halt line (left subtree in Figure 8.11). The size of this subtree can be analytically predicted from the theory exposed in Section 8.3. All calculations are identical, except initial condition (8.36) which has to be changed into

$$\varphi(y_2, y_3, t = 0) = \alpha_G\ (1 - p_G)\ y_2 + \alpha_G\ p_G\ y_3\ . \tag{8.46}$$

As a result we obtain the size $2^{N_G\ \omega_G}$ of the unsatisfiable subtree to be backtracked (leftmost subtree in Figure 8.11). $N_G = N\,(1 - t_G)$ denotes the number of undetermined variables at point G.

G is the highest backtracking node in the tree (Figures 8.1(C) and 8.11) reached 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 Figure 8.1(C)) 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 (Figures 8.3,8.12). 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 Figure 8.11. 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 8.2.2) and the estimated critical line $\alpha_c(p)$ [13]. 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

$$\omega_{THE} = \omega_G \times (1 - t_G)\ , \tag{8.47}$$

and are in very good agreement with theory (Table 8.1). 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. The exponential scaling of the complexity ($\omega > 0$) in this upper SAT regime was recently rigorously established [3].

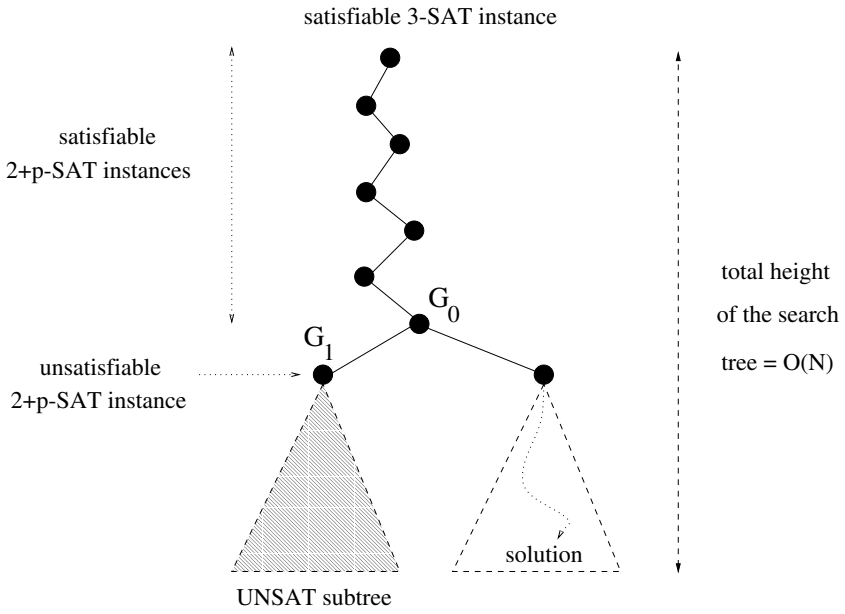

Figure 8.11: Detailed structure of the search tree in the upper SAT phase ($\alpha_L < \alpha < \alpha_C$). DPLL starts with a satisfiable 3-SAT instance and transforms it into a sequence of 2+p-SAT instances. The leftmost branch in the tree symbolizes the first descent made by DPLL. Above node G_0, instances are satisfiable while below G_1, instances have no solutions. A grey 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 Figure 8.1(C).

8.4.2 Distribution of Running Times

While in the upper SAT phase, search trees almost always look like Figure 8.1(B), they may sometimes consist of a single branch (Figure 8.1(A)). Figure 8.13 shows the normalized histogram of the logarithm ω of the solving times of instances with ratio $\alpha_0 = 3.5$ and various sizes N [14, 31, 43]. The histogram is made of a narrow peak (left side) followed by a wider bump (right side). As N grows, the right peak acquires more and more weight, while the left peak progressively disappears. The abscissa of the center of the right peak reaches a finite value $\omega^* \simeq 0.035$ as $N \to \infty$. This right peaks thus corresponds to the core of exponentially hard resolutions: with high probability resolutions of instances requiring a time scaling as $2^{N\omega^*}$ as the size of the instance gets larger and larger, in agreement with Section 8.4.1.

On the contrary, the location of the maximum of the left peak vanishes as $\log_2(N)/N$ when the size N increases, indicating that the left peak accounts for polynomial (linear) resolutions. We have thus replotted the data shown in Figure 8.13, changing the scale of the horizontal axis $\omega = \log_2(Q)/N$ into Q/N. Results are shown in Figure 8.14. We have lim-

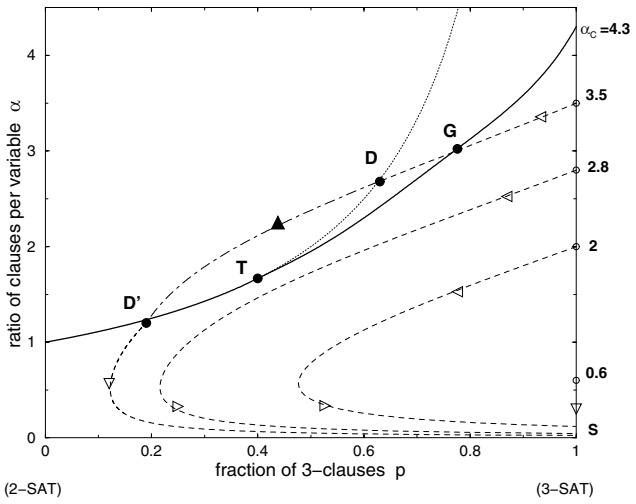

Figure 8.12: Phase diagram of 2+p-SAT and dynamical trajectories of DPLL for satisfiable instances. See caption of Figure 8.3 for definitions of critical lines and and trajectories. When the initial ratio lies in the $\alpha_L < \alpha_0 < \alpha_C$ range, with high probability, a contradiction arises before the trajectory crosses the dotted curve $\alpha = 1/(1-p)$ (point D). Through extensive backtracking, DPLL later reaches back to the highest backtracking node in the search tree (G) and finds a solution at the end of a new descending branch, see Figure 8.1(B). With exponentially small probability, the trajectory (dot-dashed curve, full arrow) is able to cross the "dangerous" region where contradictions are likely to occur; it then exits from this contradictory region (point D') and ends up with a solution (lowest dashed curve, open arrow).

ited ourselves to $Q/N < 1$, as the range of interest for analyzing the left peak of Figure 8.13. The maximum of the distribution is located at $Q/N \simeq 0.2 - 0.25$, with weak dependence upon N. The cumulative probability P_{lin} to have a complexity Q less than, or equal to N, i.e., the integral of Figure 8.14 over $0 < Q/N < 1$, decreases very quickly with N. We find an exponential decrease, $P_{lin} = 2^{-N\zeta}$, see inset of Figure 8.14. The rate $\zeta \simeq 0.011 \pm 0.001$ is determined from the slope of the logarithm of the probability shown in the inset.

The existence of rare but fast resolutions suggests the use of a systematic restart heuristic to speed up resolution [28]: if a solution is not found before N splits, DPLL is stopped and launched again after some random permutations of the variables and clauses. Intuitively, the expected number of restarts necessary to find a solution should indeed be equal to the inverse of the weight of the linear complexity peak in Figure 8.13, with a resulting total complexity scaling as $N\ 2^{0.011\,N}$, and much smaller than the one-run complexity $2^{0.035\,N}$ of DPLL. We check the above reasoning by measuring the number $N_{\rm rest}$ of restarts performed before a solution is finally reached with the restart heuristic, and averaging $\log_2(N_{res})$ over a large number of random instances. Results are reports in the inset of Figure 8.14. The typical number $N_{\rm rest} = 2^{N\bar{\zeta}}$ of required restarts clearly grows exponentially as a function of the size N with a rate $\bar{\zeta} = 0.012 \pm 0.001$. Within the accuracy of the experiments, ζ and $\bar{\zeta}$ coincide as expected.

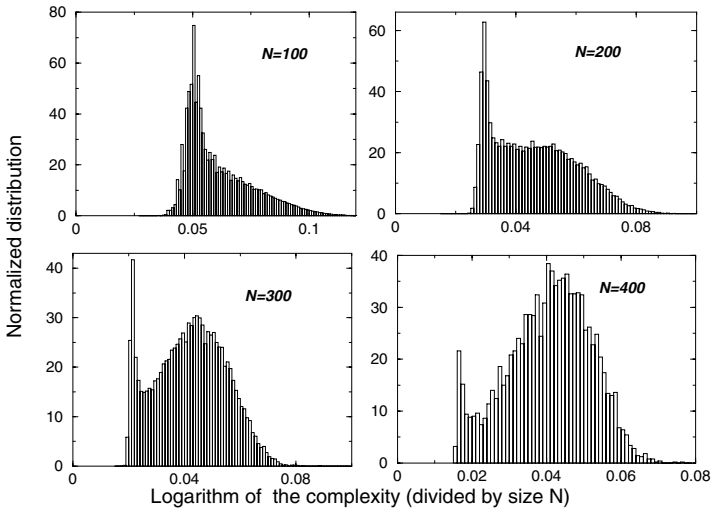

Figure 8.13: Probability distribution of the logarithm ω of the complexity (base 2, and divided by N) for $\alpha = 3.5$ and for different sizes N. Histograms are normalized to unity and obtained from 400 000 ($N = 100$), 50 000 ($N = 200$), 20 000 ($N = 300$), and 5 000 ($N = 400$) samples.

Experiments provide intuition about runs that are able to find a solution without backtracking. These are typically runs in which, although the 2-SAT subformula is supercritical ($\alpha > 1/(1-p)$) and many ($O(N)$) unitary clauses are present, no pair of these unitary clauses is contradictory (Figure 8.12). Such a "miracle" occurs with an exponentially small probability as calculated in Section 8.4.3.

This statement is supported by the analysis of the number of unit-clauses generated during easy resolutions. We have measured the maximal number $(C_1)_{\max}$ of unit-clauses generated along the last branch in the tree, leading to the solution S (Figure 8.1(B)). We found that $(C_1)_{\max}$ scales linearly with N with an extrapolated ratio $(C_1)_{\max}/N \simeq 0.022$ for $\alpha = 3.5$. This linear scaling of the number of unit-clauses is an additional proof of the trajectory entering the "dangerous" region $\alpha > 1/(1-p)$ of the phase diagram where unit-clauses accumulate. In the presence of a $O(N)$ number of 1-clauses, the probability of survival of the branch (absence of contradictory literals among the unit-clauses) will be exponentially small in N, in agreement with the scaling of the left peak weight in Figure 8.13.

8.4.3 Large Deviation Analysis of the First Branch in the Tree

We give, in the following, a lower bound to the probability that DPLL finds a solution without ever backtracking. Due to the absence of backtracking, the same probabilistic setting as in Section 8.2 may be applied: the search heuristic defines a Markov chain for the evolution of subformulas as more and more variables are set. The major difference is that we are now considering initial densities above α_L, which means that the probability of the events we

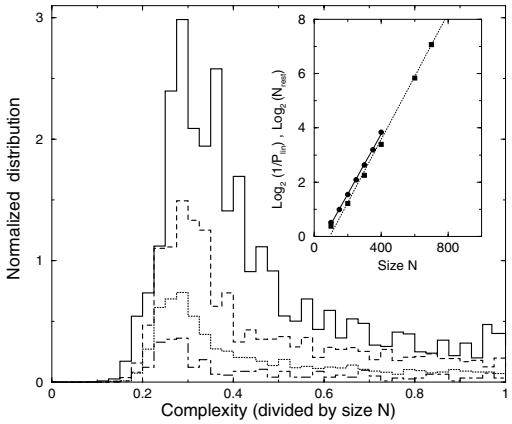

Figure 8.14: Probability distributions of the complexity Q (divided by the size N) for sizes $N = 100$ (full line), $N = 200$ (dashed line), $N = 300$ (dotted line), $N = 400$ (dashed-dotted line). Distributions are not shown for complexities larger than N. Inset: Minus logarithm of the cumulative probability of complexities smaller or equal to N as a function of N, for sizes ranging from 100 to 400 (full line); logarithm of the number of restarts necessary to find a solution for sizes ranging from 100 to 1000 (dotted line). Slopes are equal to $\zeta = 0.0011$ and $\bar{\zeta} = 0.00115$ respectively.

look for are exponentially small (in the number N of variables). In other words, rather than considering the mean resolution trajectory (which unavoidably leads to a contradiction and backtracking), we need to look at large deviations from this trajectory. Notice that, though we focus on a specific algorithm, namely GUC, our approach and the spirit of our results should hold for other heuristics.

The probability $\bar{B}(\vec{C}; T)$ that the first branch of the tree carries an instance with C_j j-clauses ($j = 1, 2, 3$) after T variables have been assigned (and no contradiction has occurred) obeys the Markovian evolution equation (8.22). The entries of the transition matrix $\bar{H}$ are given (for the GUC heuristic) by (8.23) where $\delta_{C_1 - w_1} + \delta_{C_1 - w_1 + 1}$ replaced with $\delta_{C_1 - w_1 + 1}$ in the last line.

The generating function G associated to probability $\bar{B}$ (8.24) obeys equation (8.25) with $(e^{y_1} + 1)$ replaced with 1, and $\gamma_1(\vec{y})$ in Eqn. (8.26) changed into

$$\gamma_1(\mathbf{y}) = y_1 + \ln\left[1 + \frac{1}{N-T}\left(\frac{e^{-y_1}}{2} - 1\right)\right] . \tag{8.48}$$

We now present the partial differential equations (PDE) obeyed by φ. Two cases must be distinguished: the number C_1 of unit-clauses may be bounded ($C_1 = O(1), c_1 = o(1)$), or of the order of the instance size ($C_1 = \Theta(N), c_1 = \Theta(1)$).

8.4.3.1 $C_1 = O(1)$: A Large Deviation Analysis Around Frieze and Suen's Result

When DPLL starts running on a 3-SAT instance, very few unit-clauses are generated and splittings occur frequently. In other words, the probability that $C_1 = 0$ is strictly positive when N becomes large. Consequently, both terms on the right-hand side of (8.25) are of the same order, and we make the hypothesis that φ does not depend on y_1: $\varphi(y_1, y_2, y_3; t) = \varphi(y_2, y_3; t)$. This hypothesis simply expresses that $c_1 = \partial\varphi/\partial y_1$ identically vanishes. Inserting expression (8.27) into the evolution equation (8.25), we find[9]

$$\frac{\partial \varphi}{\partial t} = -y_2 + 2\, g(y_2, y_2; t)\, \frac{\partial \varphi}{\partial y_2} + 3\, g(y_2, y_3; t)\, \frac{\partial \varphi}{\partial y_3}\ , \tag{8.49}$$

where function g is defined

$$g(u, v; t) = \frac{1}{1-t}\left(\frac{e^{-v}}{2}\,(1 + e^u) - 1\right)\ . \tag{8.50}$$

PDE (8.49) together with initial condition $\varphi(\mathbf{y}; t = 0) = \alpha_0\, y_3$ (where α_0 is the ratio of clauses per variable of the 3-SAT instance) can be solved exactly with the resulting expression,

$$\begin{aligned} \varphi(y_2, y_3; t) &= \alpha_0 \ln\left[1 + (1-t)^3\left(e^{y_3} - \frac{3}{4}e^{y_2} - \frac{1}{4}\right) + \frac{3(1-t)}{4}(e^{y_2} - 1)\right] \\ &+ (1-t)\, y_2\, e^{y_2} + (1-t)(e^{y_2} - 1)\ln(1-t) \\ &- (e^{y_2} + t - t\, e^{y_2})\ln\left(e^{y_2} + t - t\, e^{y_2}\right)\ . \end{aligned} \tag{8.51}$$

Chao and Franco, and Frieze and Suen's analysis of the GUC heuristic may be recovered when $y_2 = y_3 = 0$ as expected. It is an easy check that $\varphi(y_2 = 0, y_3 = 0; t) = 0$, i.e., the probability of survival of the branch is not exponentially small in N [24], and that the derivatives $c_2(t), c_3(t)$ of $\varphi(y_2, y_3; t)$ with respect to y_2 and y_3 coincide with the solutions of (8.4).

In addition, (8.51) also provides a complete description of rare deviations of the resolution trajectory from its highly probable locus shown in Figure 8.3. As a simple numerical example, consider DPLL acting on a 3-SAT instance of ratio $\alpha_0 = 3.5$. Once, e.g., $t = 20\%$ of variables have been assigned, the densities of 2- and 3-clauses are with high probability equal to $c_2 \simeq 0.577$ and $c_3 \simeq 1.792$ respectively. Expression (8.51) gives access to the exponentially small probabilities that c_2 and c_3 differ from their most probable values. For instance, choosing $y_2 = -0.1, y_3 = 0.05$, we find from (8.51) and (8.28) that there is a probability $e^{-0.00567N}$ that $c_2 = 0.504$ and $c_3 = 1.873$ for the same fraction $t = 0.2$ of eliminated variables. By scanning all the values of y_2, y_3 we can obtain a complete description of large deviations from Frieze and Suen's result[10].

The assumption $C_1 = O(1)$ breaks down for the most probable trajectory at some fraction t_D, e.g., $t_D \simeq 0.308$ for $\alpha_0 = 3.5$ at which the trajectory hits point D on Figure 8.12. Beyond

[9] PDE (8.49) is correct in the major part of the y_1, y_2, y_3 space and, in particular, in the vicinity of $\mathbf{y} = \mathbf{0}$ which we focus on in this paper. It has, however, to be to modified in a small region of the y_1, y_2, y_3 space; a complete analysis of this case is not reported here but may be easily reconstructed along the lines of Appendix A in [13].

[10] Though we are not concerned here with sub-exponential (in N) corrections to probabilities, we mention that it is possible to calculate the probability of split, $\bar{\mu}(C_1 = 0)$, extending the calculation of Section 8.2.4 to $\mathbf{y} \neq \mathbf{0}$.

D, 1-clauses accumulate and the probability of survival of the first branch is exponentially small in N.

8.4.3.2 Case $C_1 = O(N)$: Passing Through the "Dangerous" Region

When the number of unit-clauses becomes of the order of N, variables are almost surely assigned through unit-propagation. The first term on the right-hand side of equation (8.25) is now exponentially dominant with respect to the second one. The density of 1-clauses is strictly positive, and φ depends on y_1. We then obtain the following PDE,

$$\frac{\partial \varphi}{\partial t} = -y_1 + g(-\infty, y_1; t)\, \frac{\partial \varphi}{\partial y_1} + 2\, g(y_1, y_2; t)\, \frac{\partial \varphi}{\partial y_2} + 3\, g(y_2, y_3; t)\, \frac{\partial \varphi}{\partial y_3}\,, \qquad (8.52)$$

with $g(u, v; t)$ given by (8.50). When $y_1 = y_2 = y_3 = 0$, (8.52) simplifies to

$$\frac{dz}{dt}(t) = -\frac{c_1(t)}{2(1-t)}\,, \qquad (8.53)$$

where $c_1(t)$ is the most probable value of the density of unit-clauses, and $z(t)$ is the logarithm of the probability that the branch has not encountered any contradiction (divided by N). The interpretation of (8.53) is transparent. Each time a literal is assigned through unit-propagation, there is a probability $(1 - 1/2/(N-T))^{C_1 - 1} \simeq e^{-c_1/2/(1-t)}$ that no contradiction occurs. The right-hand side of (8.53) thus corresponds to the rate of decay of z with "time" t.

PDE (8.52) can be solved numerically [14], with results as shown in Figure 8.15. The calculated values of $\zeta \simeq 0.01, (c_1)_{\max} \simeq 0.022$ and $\gamma \simeq 0.21$ are in very good agreement with numerical experiments (Section 8.4.2). This agreement extends over the whole range $\alpha_L \leq \alpha_0 \leq \alpha_C$ [14].

8.4.3.3 More on Restarts and Cut-off

This study suggests that the cut-off time, at which the search is halted and restarted, need not be precisely tuned but is simply given by the size of the instance. This conclusion could be generic and apply to other combinatorial decision problems and other heuristics. More precisely, if a combinatorial problem admits some efficient (polynomial) search heuristic for some values of control parameter (e.g., the ratio α here, or the average adjacency degree for the coloring problem of random graphs), there might be an exponentially small probability that the heuristic is still successful (in polynomial time) in the range of parameters where resolution almost surely requires massive backtracking and exponential effort. When the decay rate of the polynomial time resolution probability ζ is smaller than the growth rate ω of the typical exponential resolution time, restart procedures with a cut-off in the search equal to a polynomial of the instance size will lead to an exponential speed-up of resolutions.

In principle, one could not rule out the existence of even luckier runs than linear ones. For instance, there could exist exponentially long (complexity $2^{\omega' N}$ with $0 < \omega' < \omega$) and rare (probability $2^{-\zeta' N}$ with $0 < \zeta' < \zeta$) runs with $\omega' + \zeta' < \zeta$. If so, $2^{\omega' N}$ would be a better cut-off for restart than N. A recent analysis of the distribution of exponentially long resolutions indicates this is not so for the problem of the vertex covering of random graphs, and that the optimal cut-off for restarts is indeed the instance size itself [39].

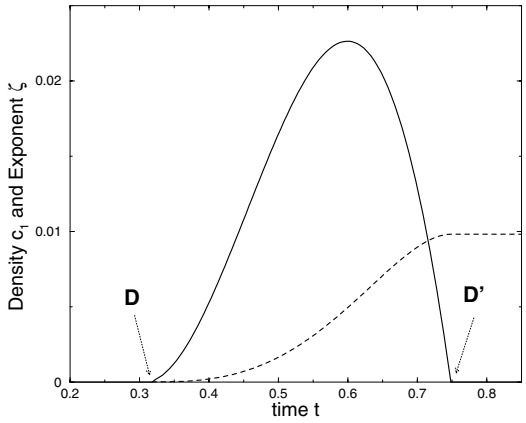

Figure 8.15: Density c_1 of unitary clauses (full line) and logarithm z (8.53) of the probability of the absence of contradiction (dashed line) along the first search branch as a function of time t (fraction of assigned variables) for an initial ratio $\alpha = 3.5$. The density of unit clauses is positive between points D and D$'$ along the branch trajectory of Figure 8.12; z is null before the trajectory reaches D, and constant and equal to the exponent ζ beyond D$'$.

8.5 The Random Graph Coloring Problem

In this section we apply the approach described above for random SAT to random COL. More precisely, we analyze the performances of a complete DPLL algorithm capable of determining whether a given graph is 3-colorable or not [20]. The algorithm is based on a combination of a coloring heuristic, 3-GREEDY-LIST (3-GL), and of backtracking steps. We first present the algorithm and then analyze the dynamics of its resolution time.

8.5.1 Description of DPLL Algorithm for Coloring

The action of the coloring procedure is described as follows:

- Necessary Information: while running, the algorithm maintains for each uncolored vertices, a list of available colors, which consists of all the colors that can be assigned to this vertex, given the colors already assigned to surrounding vertices.
- Coloring Order: the order in which the vertices are colored, is such that the most constrained vertices, i.e., with the least number of available colors, are colored first. At each time step, a vertex is chosen among the most constrained vertices, and its color is selected from the list of its available colors. Both choices are done according to some heuristic rule, which can be unbiased (no preference is made between colors), or biased (following a hierarchy between colors), see next section.
- List Updating: to ensure that no adjacent vertices have the same color, whenever a vertex is assigned a color, this color is removed from the lists (if present) which are attached to each of the uncolored neighbors.

- Contradictions and Backtracking: a contradiction occurs as soon as one of the lists becomes empty. Then, the algorithm backtracks to the most recently chosen vertex, which has more than one available color (the closest node in the search tree – see definition below).
- Termination Condition: the algorithm stops when all vertices are colored, or when all coloring possibilities have been tried.

A search tree can describe the action of the algorithm as for the SAT problem. A node in the tree represents a vertex chosen by the algorithm, which has more than one color in its available-colors list. An edge which comes out of a node, corresponds to a possible color of the chosen vertex. A leaf is either a solution (S) or a contradiction (denoted by C), see Figure 8.1.

8.5.2 Coloring in the Absence of Backtracking

Let us call the 3-GL heuristic the incomplete version of the above algorithm, obtained when the algorithm stops if a coloring is found (and outputs "Colorable"), or just after the first contradiction, instead of backtracking (and outputs "Don't know if colorable or not"). In contrast to the 3-GL algorithm with backtracking, the 3-GL heuristic is not able to prove the absence of a solution, and is amenable to rigorous analysis [5, 6].

In the simplest case, vertices and colors are chosen purely randomly without any bias between colors (Coloring Order step described above). This "symmetric" 3-GL heuristic verifies two key properties on which our analysis relies. The first one is a statistical invariance called the R-property. Throughout the execution of the algorithm, the uncolored part of the graph is distributed as $G((1-t)N, p)$ where t is the number of colored vertices divided by N. The second property is color symmetry. The search heuristic is symmetric with respect to the different colors, and the initial conditions are symmetric as well. Hence, the evolution of the algorithm can be exactly monitored by tracking of the three numbers $N_j(T)$ of j-color nodes ($j = 1, 2, 3$) only, without distinction between the colors available to each of these nodes.

The analysis of the evolution of these numbers in the course of the coloring was carried out by Achlioptas and Molloy [6]. In a way very similar to Figure 8.4 and due to the R-property, the average flows of vertices, $w_2(T)$ from $N_3(T)$ to $N_2(T)$, and $w_1(T)$ from $N_2(T)$ to $N_1(T)$ are $c\,N_3(T)/N$ and $2\,c\,N_2(T)/(3\,N)$, respectively. Note that the last factor is due to the fact that $2/3$ of the 2-color nodes adjacent to the vertex just colored have the used color as one of their two available colors. Hence, the evolution equations for the three populations of vertices read,

$$\begin{aligned} N_3(T+1) &= N_3(T) - w_2(T)\ , \\ N_2(T+1) &= N_2(T) + w_2(T) - w_1(T) - \delta N_1(T)\ , \\ N_1(T+1) &= N_1(T) + w_1(T) - (1 - \delta N_1(T))\ . \end{aligned} \tag{8.54}$$

where $\delta N_1(T) = 1$ if $N_1(T) = 0$ (a 2-color vertex is colored) and $\delta N_1(T) = 0$ if $N_1(T) \neq 0$ (a 1-color vertex is colored). For $c > 1$, both $N_2(T)$ and $N_3(T)$ are extensive in N, and can be written as

$$N_i(T) = n_i(T/N)\ N + o(N)\ . \tag{8.55}$$

The appearance of the reduced time, $t = T/N$, means that population densities $n_i(T/N)$ change by $O(1)$ over $O(N)$ time intervals. To avoid the appearance of contradictions, the number of 1-color vertices must remain of $O(1)$ throughout the execution of the algorithm. From queuing theory, this requires $w_1(t) < 1$, that is

$$\frac{2}{3}\, c\, n_2(t) < 1 \tag{8.56}$$

which means that 1-color nodes are created slowly enough to color them and do not accumulate. Thus, in the absence of backtracking, the evolution equations for the densities are

$$\frac{dn_3(t)}{dt} = -c\, n_3(t)\ , \qquad \frac{dn_2(t)}{dt} = c\, n_3(t) - 1\ . \tag{8.57}$$

The solution of these differential equations, with initial conditions $n_3(0) = 1$, $n_2(0) = 0$, is $n_3(t) = e^{-c\,t}$, $\quad n_2(t) = 1 - t - e^{-c\,t}$. Equations (8.57) were obtained under the assumption that $n_2(t) > 0$ and hold until $t = t_2$ at which the density n_2 of 2-color nodes vanishes. For $t > t_2$, 2-color vertices no longer accumulate. They are colored as soon as they are created. 1-color vertices are almost never created, and the vertices colored by the algorithm are either 2-, or 3-color vertices. Thus, when $t_2 < t < 1$, $n_2(t) = 0$, and $n_3(t) = 1 - t$ decreases to zero. A proper coloring is found at $t = 1$, i.e., when all nodes have been colored.

These equations define the trajectory of the algorithm in phase space in the absence of contradictions, i.e., as long as condition (8.56) is fulfilled. The trajectory corresponding to $c = 3$ is plotted on Figure 8.16. For $c < c_L \approx 3.847$, condition (8.56) is never violated, and the probability that the algorithm succeeds in finding an appropriate coloring without backtracking is positive. The complexity $\gamma(c)\, N$ of the algorithm in the absence of backtracking is linear with N, and equals the number of nodes in the single branch of the search tree.

$$\gamma(c) = 1 - \frac{2}{3}\, c \int_0^{t_2} dt\ n_2(t)\ , \tag{8.58}$$

where $t_2 > 0$ is the first time (after $t = 0$) that $n_2(t)$ becomes 0.

For $c > c_L$ condition (8.56) is violated at $t = t_d(c)$ which depends on c, and 1-color vertices start to accumulate. As a result, the probability for contradictions becomes large, and backtracking enters into play.

8.5.3 Coloring in the Presence of Massive Backtracking

The analytical study of the complexity in the presence of backtracking is inspired by the analysis of the DPLL algorithm acting onto random 3-SAT (see Section 8.3). In the absence of solution, DPLL builds up a complete search tree before stopping. Obviously, the order in which the available colors of a vertex are tried does not affect the final shape of the tree. This allows us to study the evolution of the parallel (instead of sequential) growth process of the search tree (see Section 8.3.2 for detailed explanations).

As was pointed out before, due to color symmetry, the three-dimensional vector $\vec{N} = (N_1, N_2, N_3)$ describes the state of a graph under the action of the algorithm. Denoting by

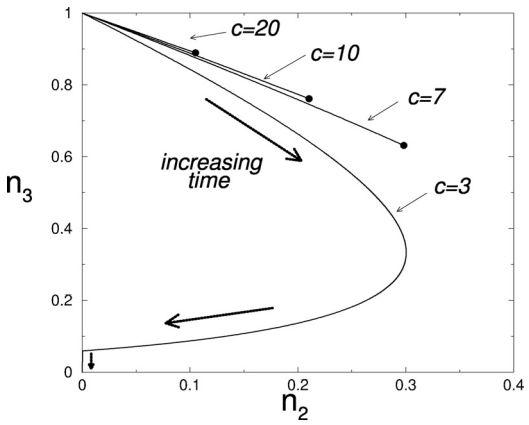

Figure 8.16: Trajectories of dominant search branches generated by DPLL in the uncolorable phase ($c > c_3 \simeq 4.7$ [18, 41]) compared to a search trajectory in the easy colorable phase ($c < c_L \simeq 3.85$). Horizontal and vertical axes represent the densities n_2 and n_3 of 2- and 3-color nodes respectively. Trajectories are depicted by solid curves, and the arrows indicate the direction of motion (increasing depth of the search tree); they originate from the left top corner, with coordinates $(n_2 = 0, n_3 = 1)$, since all nodes in the initial graph are 3-color nodes. Dots at the end of the uncol trajectories ($c = 7, 10, 20$) symbolize the halt point at which condition $n_2 < 3 \ln 2/c$ ceases to be fulfilled, and the search tree stops growing. Note that as the initial connectivity increases, the trajectories halt at an earlier stage, implying the early appearance of contradictions as the problem becomes overconstrained (large connectivity values). The col trajectory (shown here for $c = 3$) represents the under-constrained region of the problem, where the very first search branch is able to find a proper coloring (bottom left corner with coordinates $(n_2 = 0, n_3 = 0)$).

$\bar{B}(\vec{N}; T)$ the number of branches at time T with N_i ($i = 1, 2, 3$) i-color vertices (the components of $\vec{N}$), the growth process of the search tree can be described by the evolution of $\tilde{B}(\vec{N}; T)$ with time. Following the procedure exhibited in Section 8.3.2, we consider the evolution matrix

$$\bar{\mathbf{H}}(\vec{N}, \vec{N}'; T) = \sum_{w_2=0}^{N_3'} \binom{N_3'}{w_2} (\frac{c}{N})^{w_2} (1 - \frac{c}{N})^{N_3} \delta_{N_3' - N_3 - w_2} \left\{ \{(1 - \delta_{N_1'}) \right.$$

$$\sum_{w_1=0}^{N_2'} \binom{N_2'}{w_1} (\frac{2c}{3N})^{w_1} (1 - \frac{2c}{3N})^{N_2' - w_1} \delta_{N_2 - N_2' - (w_2 - w_1)} \delta_{N_1 - N_1' - w_1 + 1} +$$

$$2\delta_{N_1'} \sum_{w_1=0}^{N_2'-1} \binom{N_2' - 1}{w_1} (\frac{2c}{3N})^{w_1} (1 - \frac{2c}{3N})^{N_2' - w_1 - 1} \delta_{N_2 - N_2' - (w_2 - w_1 - 1)} \times$$

$$\left. \delta_{N_1 - N_1' - w_1} \right\} \tag{8.59}$$

where δ_N is the Kronecker delta function. The matrix describes the average number of branches with color vector $\vec{N}$ coming out from one branch with color vector $\vec{N}'$, as a result of the coloring of one vertex at step T. Note that (8.59) is written under the assumption that no 3-color nodes are chosen by the algorithm throughout the growth process. This assumption is consistent with the resultant solution which shows that in the uncolorable (uncol) region, $n_2(t)$, namely the number of 2-color vertices divided by N, keeps positive for all $t > 0$.

The generating function $G(\vec{y}; T)$ of the number $\bar{B}(\vec{N}; T)$ of branches satisfies an evolution equation similar to (8.22),

$$G(\vec{y}; T+1) = e^{-y_1}\; G\big(\vec{\gamma}(\vec{y}); T\big) + \big(2\; e^{-y_2} - e^{-y_1}\big)\; G\big(-\infty, \gamma_2(\vec{y}), \gamma_3(\vec{y}); T\big) \quad (8.60)$$

where

$$\begin{aligned} \gamma_1(\vec{y}) &= y_1\,, \\ \gamma_2(\vec{y}) &= y_2 + \frac{2c}{3N}\big(e^{y_1 - y_2} - 1\big)\,, \\ \gamma_3(\vec{y}) &= y_3 + \frac{c}{N}\big(e^{y_2 - y_3} - 1\big)\,. \end{aligned} \quad (8.61)$$

To solve (8.60), we make scaling hypotheses for $\bar{B}$ and G, similar to those made in Section 8.3.3. Namely,

$$\bar{B}(\vec{N}; T) = e^{N\,\omega(\vec{n}; t) + o(N)}, \qquad G(\vec{y}; T) = e^{N\,\varphi(\vec{y}; t) + o(N)}, \quad (8.62)$$

where $\omega(\vec{n}; t)$ is the logarithm of the number of branches $\bar{B}(\vec{N}; T)$ divided by N and $\vec{n} = (n_1, n_2, n_3)$. As in Section 8.3.3, φ is the Legendre transform of ω. At the initial stage of the tree building up, there is a single outgoing branch from the root node, carrying a fully uncolored graph. Thus, $\bar{B}(\vec{N}; T = 0) = 1$ if $\vec{N} = (0, 0, N)$, 0 otherwise, and $G(\vec{y}, T = 0) = e^{N\,y_3}$. The initial condition for function φ is simply, $\varphi(\vec{y}; t = 0) = y_3$. According to (8.55) both $N_2(T)$ and $N_3(T)$ are extensive in N; hence $n_2 > 0$ and $n_3 > 0$. Conversely, as soon as $N_1(T)$ becomes very large, contradictions are very likely to occur, and the growth process stops. Throughout the growth process, $N_1 = O(1)$ almost surely. Thus $n_1 = 0$ with high probability, and φ does not depend upon y_1. Independence of φ from y_1 allows us to choose the latter at our convenience, that is, as a function of y_2, y_3, t. Following the so-called kernel method [33], we see that equation (8.60) simplifies if $y_1 = y_2 - \ln 2$. Then, from ansatz (8.62), we obtain the following partial differential equation (PDE),

$$\frac{\partial \varphi}{\partial t}(y_2, y_3; t) = -y_2 + \ln 2 - \frac{c}{3}\frac{\partial \varphi}{\partial y_2}(y_2, y_3; t) + c\;(e^{y_2 - y_3} - 1)\frac{\partial \varphi}{\partial y_3}(y_2, y_3; t)\,. \quad (8.63)$$

This PDE can be interpreted as a description of the growth process of the search tree resulting from the algorithm operation. Through Legendre transformation, PDE (8.63) can be written as an evolution equation for the logarithm $\omega(n_2, n_3; t)$ of the average number of branches with densities n_2, n_3 of 2-, 3-colors nodes as the depth $t = T/N$ increases,

$$\frac{\partial \omega}{\partial t} = \frac{\partial \omega}{\partial n_2} + \ln 2 - \frac{c}{3}\; n_2 + c\; n_3 \left[\exp\left(\frac{\partial \omega}{\partial n_3} - \frac{\partial \omega}{\partial n_2}\right) - 1\right]\,. \quad (8.64)$$

The surface ω, growing with "time" t above the plane n_2, n_3 describes the whole distribution of branches. Here, this distribution simplifies due to node conservation. The sum $n_2 + n_3$ of 2- and 3-color node densities necessarily equals the fraction $1 - t$ of not-yet colored nodes. Therefore, ω is a function of n_3 and t only, whose expression is obtained through exact resolution of PDE (8.63) with the above initial condition,

$$\begin{aligned}\omega(n_3;t) &= \frac{c}{6}\, t\,(1-2\,t-4\,n_3) - n_3\,\ln\,n_3 - (1-n_3)\,\ln(1-n_3) - \\ &\quad (1-t-n_3)\ln 2 + (1-n_3)\,\ln\left[3\,\left(1-e^{-\,2\,t\,c/3}\right)\right]. \end{aligned} \tag{8.65}$$

Figure 8.17 exhibits $\omega(n_3, t)$ for $c = 10$.

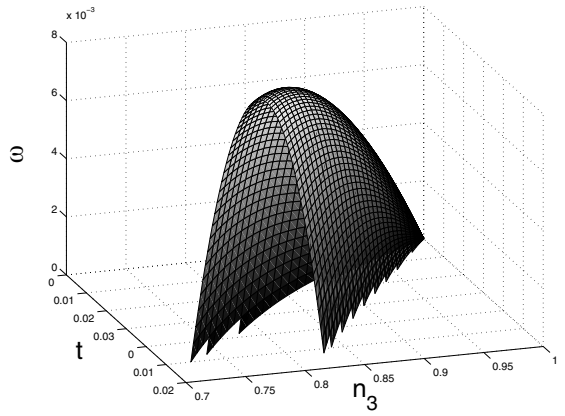

Figure 8.17: Function ω (log of number of branches with densities $n_2 = 1 - t - n_3$, n_3 of 2- and 3-color nodes at depth t in the search tree) as a function of n_3 and t for $c = 10$. The top of the curve at given time t, $\omega^*(t)$, is reached for the dominant branch 3-color density $n_3^*(t)$. The evolution of ω is shown until $t = t_h$ at which dominant branches in the search tree stop growing (die from the onset of contradictions). The maximal ω at t_h, $\omega^*(t_h)$, is our theoretical prediction for the complexity.

The maximum over n_2, n_3 of $\omega(n_2, n_3; t)$ at depth t in the tree

$$\omega^*(t) = \frac{c}{6}t^2 - \frac{c}{3}t - (1-t)\ln 2 + \ln\left[3 - \; e^{-2c\,t/3}\right] \tag{8.66}$$

is reached at $n_3^*(t) = 2/(3\; e^{\,2\,c\,t/3} - 1)$, $n_2^*(t) = 1 - t - n_3^*(t)$, and gives the logarithm of the average number of branches at depth t divided by N (see Section 8.3.4 and explanations there). Under the action of the 3-GL algorithm, initially random 3-coloring instances become random mixed 2 and 3-coloring instances, where nodes can have either 2 or 3 colors at their disposal. This phenomenon indicates that the action of the 3-GL algorithm on random 3-coloring instances can be seen as an evolution in the n_2, n_3 phase-space (Figure 8.16). Each point (n_2, n_3) in this space, represents a random mixed 2 and 3-coloring instance, with an

Table 8.2: Analytical results and simulation results of the complexity ω for different connectivities c in the uncol phase. The analytical values of ω_{THE} are derived from theory; ω_{NOD} is obtained by measuring the average value of the search tree size.

c	ω_{THE}	ω_{NOD}
20	2.886×10^{-3}	$3 \times 10^{-3} \pm 3 \times 10^{-4}$
15	5.255×10^{-3}	$5.8 \times 10^{-3} \pm 5 \times 10^{-4}$
10	1.311×10^{-2}	$1.5 \times 10^{-2} \pm 1 \times 10^{-3}$
7	2.135×10^{-2}	$3. \times 10^{-2} \pm 3.6 \times 10^{-3}$

average number $(n_2 + n_3)N$ of nodes, and a fraction $n_3/(n_2 + n_3)$ of 3-color nodes. Parametric plot of $n_2^*(t), n_3^*(t)$ as a function of t represents the trajectories of dominant branches in Figure 8.16.

The halt condition, analogous to (8.41) for the DPLL algorithm, is $n_2^*(t) = 3 \ln 2/c$. It defines the endpoints of the dominant branch trajectories in the n_2, n_3 dynamical phase diagram of Figure 8.16. Call t_h the halt time at which the halt condition is fulfilled. The logarithm $\omega^*(t_h)$ of the number of dominant branches at $t = t_h$, when divided by $\ln 2$, yields our analytical estimate for the complexity of resolution, $\ln Q/N$.

To check our theory, we have run numerical experiments to estimate ω, the logarithm of the median solving time, as a function of the initial graph degree c. Table 8.2 presents results for ω as a function of the connectivity c in the uncol phase as found from numerical experiments and from the above theory. Note the significant decrease in the complexity as the initial connectivity increases. Agreement between theory and numerics is good but deteriorates at small c. However, the high computational complexity of the algorithm for small c values, does not allow us to obtain numerical results for large sizes N, and affects the quality of the large N extrapolation of ω.

In the uncol region, as c increases, contradictions emerge in an earlier stage of the algorithm, the probability that the same vertex appears in different branches reduces, and the analytical prediction becomes exact. As a consequence of the early appearance of contradictions, the complexity ω decreases with c. At very large c, we find

$$\omega(c) \asymp \frac{3 \ln 2}{2} \frac{1}{c^2} \simeq \frac{1.040}{c^2} , \tag{8.67}$$

and therefore that the (logarithm of the) complexity exhibits a power law decay with exponent 2 as a function of connectivity c.

8.6 Conclusions

In this chapter, we have explained a procedure for understanding and quantifying the complexity pattern of the backtrack resolution of the random decision problem, for which input distributions depend on a few control parameters. Under the action of the backtracking algorithm, the inputs are modified and additional control parameters must be introduced to model their distribution. The main steps in our approach 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 us to distinguish hard from easy regimes.

In addition, we have presented a quantitative study of the search tree growth, which allows us to accurately estimate the complexity of resolution in the 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 [2]. As shown in Section 8.4, the analysis of backtracking algorithms is not limited to the average-case complexity, but may also capture the distribution of resolution times [14,26].

Although the approach has been illustrated on the SAT and COL problems, it has already been applied to other decision problems, e.g., the vertex covering (VC) of random graphs [45]. In the VC problem, the parameter space is composed of the relative fraction of vertices which are allowed to be covered, x, and the average connectivity c of the graph. Following the three aforementioned steps, a complexity pattern of a branch-and-bound algorithm was obtained, yielding a distinction between exponential and linear regimes of the algorithm. The emerging pattern of complexity is similar to those of the DPLL algorithm for SAT and COL. The bound introduced in [45], was proved to significantly reduce the time consumption of the algorithm in the exponential regime, underlying the possibility of analyzing not only pure backtracking algorithms but also their improved bound-including versions.

In the light of the success of our method in investigating the performance of the DPLL algorithm, other not-yet studied backtracking algorithms, as well as more complicated heuristics, are future possibilities for effective analysis using this method. However, from a mathematical point of view, it would be very interesting to have better control of the dynamical annealing approximation underlying the analysis of the search tree in the presence of massive backtracking. The relative technical simplicity of the analysis of DPLL for the random 3-COL problem with respect to random 3-SAT, makes 3-COL a promising candidate for future rigorous studies [16]. The growth partial differential equation, monitoring the evolution of the search tree, is simpler than its 3-SAT counterpart, a consequence of the conservation law expressing that the sum of the numbers of colored and uncolored nodes remains constant throughout the search. We hope that progress towards greater rigor will be made in the near future.

Acknowledgments

Partial support from the ACI Jeunes Chercheurs “Algorithmes d’optimisation et systèmes désordonnés quantiques” is acknowledged. L.E. benefited from the post-doctoral funding program of the French Ministry of Research during year 2001/2002.

Notes added in Proofs:

- Section 8.2.4: the critical regime $\alpha \simeq \alpha_L$ has been recently investigated (C. Deroulers, R. Monasson, *Critical scaling of search heuristics and the unit-clause universality class*, preprint (2004)). The probability that UC or GUC succeeds in finding a solution (without ever backtracking) scaled as $\exp[-N^{1/6}\,\phi((\alpha-\alpha_L)\,N^{1/3})]$ where ϕ can be explicitly expressed in terms of the Airy function.

- Section 8.4.3 : the power law behaviour of the complexity ω at large connectivities c depends on the number of colors ($Q = 3$ throughout Section 8.5). It is conjectured that ω decreases as $c^{-(Q-1)/(Q-2)}$ (R. Monasson, *Towards an analytical understanding of backtracking procedures?*, preprint (2004)).

References

[1] D. Achlioptas, L. Kirousis, E. Kranakis, and D. Krizanc, *Rigorous results for random (2+p)-SAT*, Theor. Comp. Sci. **265**, 109 (2001).

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

[3] D. Achlioptas, P. Beame, and M. Molloy, *A sharp threshold in proof complexity*, in *Proceedings of STOC 01*, 337 (2001).

[4] D. Achlioptas and E. Friedgut, *A sharp threshold for k-colorability*, Random Structures and Algorithms **14(1)**, 63 (1999).

[5] D. Achlioptas and C. Moore, *Almost all graphs with average degree 4 are 3-colorable*, Proc. on 34th Annual ACM Symposium on Theory of Computing, May 19-21, Montreal, Quebec, Canada, ACM, Montreal, 199 (2002)

[6] D. Achlioptas and M. Molloy, *Analysis of a list-colouring algorithm on a random graph*, Proc. of FOCS 97, 204 (1997).

[7] P. Beame, R. Karp, T. Pitassi, and M. Saks, *ACM symp. on theory of computing (STOC98)*, 561–571 Assoc. Comput. Mach., New York (1998).

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

[9] M.T. Chao and J. Franco, *Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k-satisfiability problem*, Information Science **51**, 289 (1990).

[10] P. Cheeseman, B. Kanefsky, and M.W. Taylor, *Where the really hard problems are*, in J. Mylopoulos and R. Reiter, editors, *Proc. of the 12th IJCAI*, 331, (Morgan Kaufmann Publishers, Inc., 1991).

[11] V. Chvàtal and E. Szmeredi, *Many hard examples for resolution*, Journal of the ACM **35**, 759 (1988).

[12] C. Coarfa, D.D. Dernopoulos, A. San Miguel Aguirre, D. Subramanian, and M.Y. Vardi, *Random 3-SAT: the plot thickens*, in R. Dechter, editor, *Proc. Principles and Practice of Constraint Programming (CP'2000)*, Lecture Notes in Computer Science **1894**, 143 (2000).

[13] S. Cocco and R. Monasson, *Trajectories in phase diagrams, growth processes and computational complexity: how search algorithms solve the 3-Satisfiability problem*, Phys. Rev. Lett. **86**, 1654 (2001); *Analysis of the computational complexity of solving random satisfiability problems using branch and bound search algorithms*, Eur. Phys. J. B **22**, 505 (2001).

[14] S. Cocco and 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**, 037101 (2002); *Restarts and exponential acceleration of the Davis–Putnam–Loveland–Logemann algorithm: a large deviation analysis of the generalized unit clause heuristic for random 3-SAT*, to appear in Ann. Math. Artificial Intelligence (2003).

[15] S. Cocco, A. Montanari, R. Monasson, and G. Semerjian, *Approximate analysis of algorithms with physical methods*, preprint (2003).

[16] S. Cocco and R. Monasson, *Heuristic average-case analysis of backtrack resolution of random 3-Satisfiability instances*, to appear in Theor. Com. Sci. A (2004).

[17] J. Crawford and L. Auton, *Experimental results on the cross-over point in satisfiability problems*, *Proc. 11th Natl. Conference on Artificial Intelligence (AAAI-93),* 21–27, (The AAAI Press / MIT Press, Cambridge, MA, 1993); Artificial Intelligence **81** (1996).

[18] J.C. Culbersome and I.P. Gent, *Frozen development in graph coloring*, Theor. Comp. Sci. **265(1-2)**, 227 (2001).

[19] M. Davis, G. Logemann, and D. Loveland, *A machine program for theorem proving.* Communications of the ACM **5**, 394 (1962).

[20] L. Ein-Dor and R. Monasson, *The dynamics of proving uncolorability of large random graphs. I. symmetric colouring heuristic*, to appear in J. Phys. A (2003).

[21] R. Segdewick and P. Flajolet, *An introduction to the analysis of algorithms*, Chapter 3, (Addison-Wesley, Boston, 1995).

[22] J. Franco, *Results related to thresholds phenomena research in satisfiability: lower bounds*, Theor. Comp. Sci. **265**, 147 (2001).

[23] E. Friedgut, *Sharp thresholds of graph properties, and the k-sat problem*, Journal of the A.M.S. **12**, 1017 (1999).

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

[25] M.R. Garey and D.S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-Completeness*, (W.H. Freeman and Company, San Fransico, 1979).

[26] I.P. Gent and T. Walsh, *Easy problems are sometimes hard*, Artificial Intelligence **70**, 335 (1994).

[27] I. Gent, H. van Maaren, and T. Walsh, (eds). *SAT2000: Highlights of satisfiability research in the year 2000*, Frontiers in Artificial Intelligence and Applications, vol. **63**, (IOS Press, Amsterdam, 2000).

[28] C.P. Gomes, B. Selman, N. Crato, and H. Kautz, J. Automated Reasoning **24**, 67 (2000).

[29] R.B. Griffiths, C.-H. Weng, and J.S. Langer, *Relaxation times for metastable states in the mean-field model of a ferromagnet*, Phys. Rev. **149**, 301 (1966).

[30] J. Gu, P.W. Purdom, J. Franco, and B.W. Wah, *Algorithms for satisfiability (SAT) problem: a survey*. DIMACS Series on Discrete Mathematics and Theoretical Computer Science **35**, 19 (American Mathematical Society, 1997).

[31] T. Hogg and C.P. Williams, *The hardest constraint problems: a double phase transition*, Artificial Intelligence **69**, 359 (1994).

[32] A.C. Kaporis, L.M. Kirousis, and E.G. Lalas, *The probabilistic analysis of a greedy satisfiability algorithm*, Lecture Notes in Computer Science **2461**, 574 (2002).

[33] D.E. Knuth, *The Art of Computer Programming, vol. 1: Fundamental Algorithms*, (Addison-Wesley, New York, 1968).

[34] D.E. Knuth, *Selected Papers on Analysis of Algorithms*, Center for the Study of Language and Information, Lecture Notes **102**, Stanford CA (2000).

[35] G. Lopez, *Partial Differential Equations of First Order and Their Applications to Physics*, (World Scientific, Singapore, 1999).

[36] A. McKane, M. Droz, J. Vannimenus, and D. Wolf (eds), *Scale Invariance, Interfaces, and Non-equilibrium Dynamics*, Nato Asi Series B: Physics **344**, (Plenum Press, New York, 1995).

[37] D. Mitchell, B. Selman, and H. Levesque, *Hard and easy distributions of SAT problems*, *Proc. of the Tenth Natl. Conf. on Artificial Intelligence (AAAI-92)*, 440, (The AAAI Press / MIT Press, Cambridge, MA, 1992).

[38] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, *Determining computational complexity from characteristic 'phase transitions'*, Nature **400**, 133 (1999); *2+p-SAT: relation of typical-case complexity to the nature of the phase transition*, Random Structure and Algorithms **15**, 414 (1999).

[39] A. Montanari and R. Zecchina, *Boosting search by rare events*, Phys. Rev. Lett. **88**, 178701 (2002).

[40] R. Motwani and P. Raghavan, *Randomized Algorithms*, (Cambridge University Press, Cambridge, 1995).

[41] R. Mulet, A. Pagnani, M. Weigt, and R. Zecchina, *Coloring random graphs*, Phys. Rev. Lett. **89**, 268701 (2002).

[42] R. Sedgewick and P. Flajolet, *An Introduction to the Analysis of Algorithms*, (Addison-Wesley, New York, 1996).

[43] B. Selman and S. Kirkpatrick, *Critical behavior in the computational cost of satisfiability testing*, Artificial Intelligence **81**, 273 (1996).

[44] J.S. Turner, *Almost all k-colorable graphs are easy to color*, Journal of Algorithms **9**, 63 (1988).

[45] M. Weigt and A.K. Hartmann, *Typical solution time for a vertex-covering algorithm on finite-connectivity random graphs*, Phys. Rev. Lett. **86**, 1658 (2001).

[46] N. Wormald, *Differential equations for random processes and random graphs*, Ann. Appl. Probab. **5**, 1217 (1995).