The Satisfiability Phase Transition of Uniform Random 3-SAT: An Empirical Study with a From-Scratch DPLL Solver

July 2026

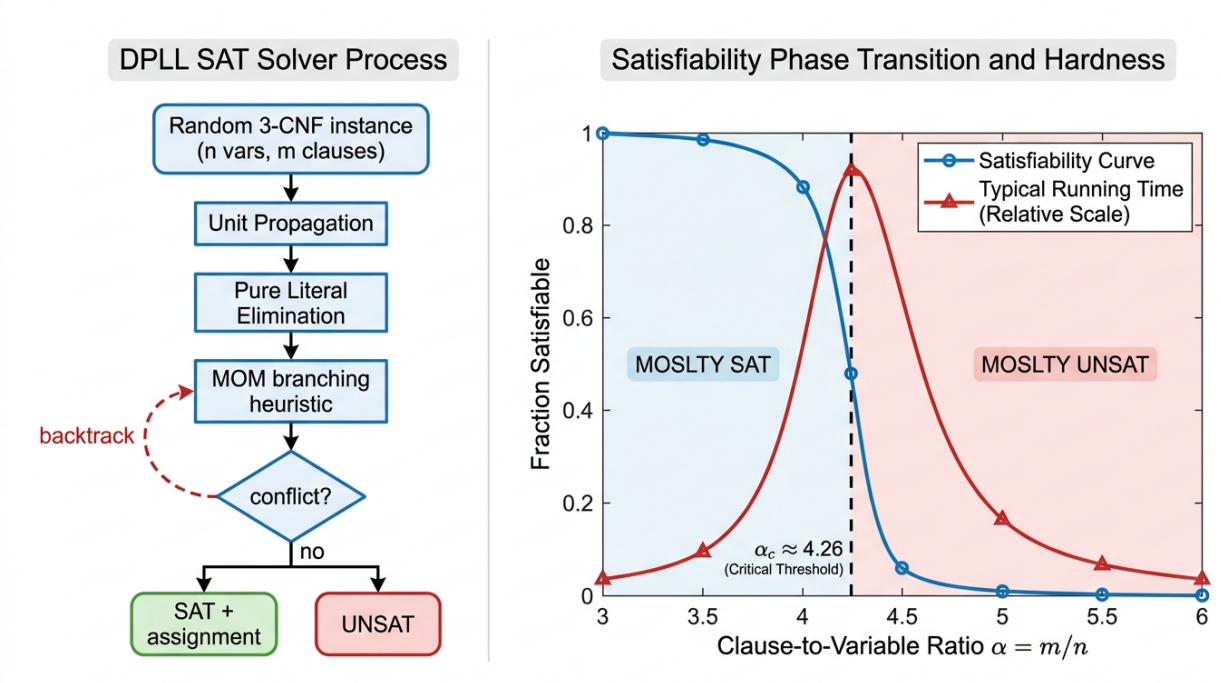

Figure 1: Graphical abstract. A complete Davis–Putnam–Logemann–Loveland (DPLL) solver—built from scratch with unit propagation, pure-literal elimination, and the Maximum Occurrence in clauses of Minimum size (MOM) branching heuristic—decides uniform random 3-CNF formulae as satisfiable (with a certified assignment) or unsatisfiable. Sweeping the clause-to-variable ratio $\alpha = m/n$ reveals the characteristic satisfiability phase transition: the fraction of satisfiable instances drops sharply from one to zero near the critical threshold $\alpha_c \approx 4.26$, while solver effort follows an "easy–hard–easy" pattern that peaks at the transition.

Abstract

Uniform random 3-SAT exhibits one of the most-studied phase transitions in combinatorics and theoretical computer science: as the clause-to-variable ratio $\alpha = m/n$ increases, formulae switch abruptly from almost surely satisfiable to almost surely unsatisfiable, with the computational cost of deciding them peaking in the transition window. We implement a complete DPLL SAT solver from scratch—no external SAT library—featuring unit propagation, pure-literal elimination, and the MOM branching heuristic, and pair it with an independent satisfying-assignment verifier.

Generating 1120 uniform random 3-CNF instances at $n = 50$ (50 instances per point) and $n = 100$ (20 per point) across 16 ratios spanning $\alpha \in [3.0, 6.0]$ in steps of 0.2, under a fixed 5 s per-instance wall-clock cutoff, we empirically map both the fraction of satisfiable instances and the median solver running time. Every instance was decided within the cutoff (zero timeouts) and every one of the 511 satisfying assignments was independently certified (zero verifier failures). Fitting a decreasing logistic sigmoid, the 50% satisfiability crossover lies at $\alpha_{0.5} = 4.37$ for $n = 50$ and 4.27 for $n = 100$ (model-free linear interpolation gives 4.31 and 4.25), converging toward the asymptotic threshold $\alpha_c \approx 4.26$. The median running time peaks at $\alpha = 4.4$ for both sizes, growing from 12.7 ms at $n = 50$ to 254 ms at $n = 100$. The transition sharpens with system size (logistic steepness k rising from 4.10 to 7.29) and the peak search effort scales by roughly 8–20× as n doubles, reproducing the hallmark finite-size scaling signature of the random 3-SAT phase transition.

1 Introduction

The Boolean satisfiability problem (SAT) asks whether there exists an assignment of truth values to the variables of a propositional formula that makes the formula evaluate to true. SAT holds a foundational place in computer science: it was the first problem shown to be NP-complete [9, 27], and it remains the canonical NP-complete problem against which countless others are reduced [18]. The restriction to 3-SAT—where the formula is in conjunctive normal form (CNF) and every clause contains exactly three literals—is itself NP-complete, yet decades of engineering have produced solvers that routinely dispatch industrial instances with millions of variables, making SAT the workhorse of hardware verification, planning, and automated reasoning [3, 4]. This gap between worst-case intractability and typical-case tractability motivates the study of *random* instance ensembles, where the average-case difficulty of SAT can be characterized precisely and where a striking connection to statistical physics emerges. Average-case analysis of Davis–Putnam-style procedures dates back to Franco and Paull [15], and random formulae also furnish provably hard instances for resolution, as Chvátal and Szemerédi showed for the over-constrained regime [7].

The central object of that study is the *uniform random 3-SAT* ensemble. Fixing n Boolean variables and generating m clauses, each formed by choosing three distinct variables uniformly at random and negating each independently with probability one half, the single control parameter is the clause-to-variable ratio $\alpha = m/n$, also called the constraint density. When α is small the formula is under-constrained and almost surely satisfiable; when α is large it is over-constrained and almost surely unsatisfiable. Early large-scale experiments by Mitchell, Selman, and Levesque [33] and by Cheeseman, Kanefsky, and Taylor [6] revealed that the crossover between these two regimes is remarkably sharp and, crucially, coincides with a pronounced peak in the running time of complete search procedures. Cheeseman *et al.* memorably located the region "where the really hard problems are" [6]: instances drawn far from the transition are easy, whereas instances drawn from the narrow critical band are hard, producing the now-familiar "easy–hard–easy" profile.

A substantial theoretical apparatus has since been built around this phenomenon. The satisfiability threshold conjecture posits that for each $k \geq 2$ there is a sharp critical density $\alpha_c(k)$ such that, in the limit $n \to \infty$, a random k-CNF formula is satisfiable with probability tending to one for $\alpha < \alpha_c(k)$ and to zero for $\alpha > \alpha_c(k)$. Friedgut established the existence of a sharp (non-uniform) threshold for k-SAT [17], while the precise location for $k = 3$ has been pinned down by a combination of rigorous bounds and statistical-physics methods. Kirousis and colleagues, together with Achlioptas, Peres, and Naor, tightened analytic bounds on the threshold and rigorously located phase transitions in hard optimization problems [25, 2, 1], while probabilistic analyses of greedy satisfiability algorithms supplied matching algorithmic lower bounds [23], and the cavity (replica) method of statistical mechanics, developed by Mézard, Parisi, Zecchina, and Mertens, predicts

$\alpha_c(3) \approx 4.267$ with striking precision [31, 30]. This picture was placed on rigorous footing for large k by Ding, Sly, and Sun, who proved the satisfiability conjecture and confirmed the physics prediction in that regime [13]. Crawford and Auton's careful experiments had already localized the finite-n crossover for random 3-SAT near $\alpha \approx 4.2$ [10], and the value $\alpha_c \approx 4.26$ has become the standard reference point for the transition.

Beyond the satisfiability threshold itself, the transition region is the seat of rich structural and computational phenomena. Monasson, Zecchina, and co-workers connected the order of the phase transition to the nature of the computational cost, showing that the emergence of a *backbone* of frozen variables accompanies the hardness peak [34]. The statistical-physics analysis further predicts a clustering (dynamical) transition below the satisfiability threshold, at which the space of solutions shatters into exponentially many well-separated clusters [26], a structural change that stymies simple local-search algorithms such as GSAT and WalkSAT [38, 36, 37] and motivated the invention of message-passing methods, culminating in survey propagation [32, 5]. Runtime distributions of complete solvers near the threshold are heavy-tailed, which is exploited by randomization and restart strategies [20], and the broader program of understanding empirical hardness in NP-complete problems takes random 3-SAT as its paradigmatic testbed [28].

At any finite n, the transition is not a true discontinuity but a smooth sigmoid whose width shrinks and whose slope steepens as n grows—the signature of finite-size scaling. Kirkpatrick and Selman brought the finite-size scaling formalism of statistical physics to bear on random SAT, characterizing how the transition window narrows with n [24], and Gent and Walsh gave a complementary empirical account of the SAT phase transition [19]. The finite-n half-satisfiable point typically sits slightly below the asymptotic threshold and drifts upward toward α_c as n increases, while the concentration of computational cost near the transition intensifies. These finite-size effects are precisely what a controlled empirical sweep across system sizes can expose.

In this work we conduct such a sweep. We implement a complete DPLL solver from scratch—deliberately using no external SAT-solving library—so that every determination and every satisfying assignment is produced by, and traceable to, transparent code. The solver combines the classical ingredients of the Davis–Putnam–Logemann–Loveland procedure [12, 11]: unit propagation, pure-literal elimination, and a branching rule based on the MOM heuristic [16, 22]. We generate uniform random 3-CNF instances following the standard fixed-clause-length model [33] and solve 1120 of them at $n = 50$ and $n = 100$ across the ratio range $\alpha \in [3.0, 6.0]$, under a fixed 5 s cutoff. For every instance we record its SAT/UNSAT/timeout determination, and for every satisfiable instance we emit a satisfying assignment that is checked by an independent verifier. Our contributions are threefold. First, we empirically map the fraction of satisfiable instances and the median solver running time as functions of α, and we report the 50% crossover ratio and the ratio at which median running time peaks. Second, we quantify how these observables change between $n = 50$ and $n = 100$, exhibiting the sharpening of the transition and the growth of the complexity peak that constitute finite-size scaling. Third, we deliver a fully verified, per-instance record—including a certified satisfying assignment for each of the 511 satisfiable formulae—so that the entire study is reproducible and auditable.

2 Methods

2.1 Uniform random 3-CNF instance generation

We adopt the standard fixed-clause-length random 3-SAT model used throughout the phase-transition literature [33, 10]. A formula over n variables (indexed $1, \ldots, n$) consists of m clauses. Each clause is generated by sampling three *distinct* variables uniformly without replacement and then negating

each chosen variable independently with probability $\frac{1}{2}$. Sampling distinct variables within a clause has two consequences that match the canonical ensemble: no clause is a tautology (a variable cannot appear both positively and negatively in the same clause), and each clause genuinely constrains three different variables. We additionally reject duplicate clauses so that the m clauses of a formula are pairwise distinct, where two clauses are considered identical if they contain the same set of signed literals irrespective of order. This is the same ensemble that underlies the widely used SATLIB uniform-random-3-SAT benchmark families [21], so our generated instances are drawn from the canonical distribution used across the phase-transition literature. The number of structurally valid distinct clauses over n variables is $8\binom{n}{3}$, which for $n = 50$ and $n = 100$ vastly exceeds the largest m used here ($m = 300$ and $m = 600$ respectively), so the distinctness constraint has negligible effect on the ensemble but guarantees well-formed instances.

For each grid point we set $m = \mathrm{round}(\alpha \cdot n)$. Instances are generated by rejection sampling of distinct clauses; in the (unused-here) dense regime where m approaches $8\binom{n}{3}$ the generator falls back to exhaustive enumeration followed by sampling without replacement, preserving uniformity. Every generated instance is checked by a structural validator that confirms each clause has exactly three literals over three distinct in-range variables and that all clauses are distinct; all 1120 instances passed this check.

2.2 The DPLL solver

Our solver is a complete implementation of the Davis–Putnam–Logemann–Loveland backtracking procedure [12, 11], written from scratch in Python with no reliance on any external SAT solver. Clauses are represented as frozen sets of nonzero integer literals (a positive integer v denotes the literal x_v and $-v$ denotes $\neg x_v$), and a partial assignment is a mapping from variables to Boolean values. At each search node the solver applies the following steps.

Unit propagation. Whenever a clause is reduced to a single unassigned literal (a *unit* clause), that literal must be set to true for the formula to remain satisfiable. The solver repeatedly locates a unit clause, assigns its literal, and simplifies the formula—removing every clause the literal satisfies and striking the complementary literal from the remaining clauses—until no unit clause remains or a conflict (an empty clause) is produced. Unit propagation, also known as Boolean constraint propagation, is the single most important inference rule in DPLL and accounts for the bulk of the deductive work.

Pure-literal elimination. A literal is *pure* if its variable occurs with only one polarity among the remaining clauses. Such a variable can always be assigned so as to satisfy every clause in which it appears, without risk of conflict. The solver iteratively identifies all pure literals, assigns them, and simplifies; because eliminating one pure literal can render others pure, the process repeats to a fixpoint.

MOM branching heuristic. When neither rule applies and clauses remain, the solver must branch on a variable. We use the Maximum Occurrence in clauses of Minimum size (MOM) heuristic [16, 22], which concentrates on the clauses of minimum current size—these are the most constrained and therefore the most informative to satisfy first. For each variable x, let $f(x)$ and $f(\neg x)$ be the number of occurrences of its positive and negative literals among the minimum-size clauses. The heuristic selects the variable maximizing

$$\big(f(x) + f(\neg x)\big)\, 2^k \;+\; f(x)\, f(\neg x), \tag{1}$$

with weight exponent $k = 10$. The first term rewards high total occurrence in the smallest clauses, driving further unit propagation, while the product term $f(x)f(\neg x)$ rewards variables that are balanced across polarities, so that either branch prunes many clauses. The solver branches first on the polarity that occurs more frequently in the minimum-size clauses.

Search and completeness. The three steps are embedded in a depth-first backtracking search: after selecting a branching literal, the solver tries it in both polarities, recursing into the simplified subformula and backtracking on conflict. A node all of whose clauses have been satisfied yields a model; a node that produces an empty clause is a dead end. If the search exhausts both polarities at the root without finding a model, the formula is declared unsatisfiable. Because the procedure systematically explores the assignment tree with sound simplification rules, it is a *complete* decision procedure: it returns SAT with a witness or UNSAT with certainty (subject only to the time cutoff below). The recursion limit is raised to accommodate deep search trees on the hardest instances. When a model is found, any variable left unassigned by the search (a "don't-care") is filled in arbitrarily so that the returned assignment is total over $1, \ldots, n$.

2.3 Time cutoff and three-valued outcome

Each solve call is governed by a fixed per-instance wall-clock cutoff of $\tau = 5\,\mathrm{s}$. The solver checks the elapsed time at every node and before each branch; if the deadline is exceeded it aborts and returns the distinguished outcome TIMEOUT. The solver thus reports one of three values per instance: SAT, UNSAT, or TIMEOUT. In accordance with the study protocol, timeouts are counted separately and are excluded from the running-time medians; they are *not* counted as either satisfiable or unsatisfiable. As reported in Section 3, no instance in this study reached the cutoff, so the timeout category is empty and the medians are taken over the full sample at every ratio.

2.4 Independent verification of satisfying assignments

To guard against solver bugs that might silently mislabel an instance, every satisfying assignment is checked by a verifier that is implemented independently of the solver. Given a formula and a candidate total assignment, the verifier confirms that every clause contains at least one literal that evaluates to true; a positive literal v is satisfied when its variable is true and a negative literal $-v$ when its variable is false. Only assignments that pass this independent oracle are accepted as certificates of satisfiability. Across the entire study, all 511 assignments emitted by the solver were certified, with zero verifier failures.

2.5 Experimental design

The experimental grid is summarized in Table 1. We sweep 16 constraint densities $\alpha \in \{3.0, 3.2, \ldots, 6.0\}$ in steps of 0.2, a range that brackets the transition on both sides. At $n = 50$ we solve 50 independent instances per ratio; at $n = 100$ we solve 20 per ratio. This yields $16 \times 50 = 800$ instances at $n = 50$ and $16 \times 20 = 320$ at $n = 100$, for 1120 instances in total. Each instance is assigned a deterministic seed derived from its $(n, \alpha, \text{trial})$ coordinates, so the entire experiment is exactly reproducible. For every instance we persist its determination, its decision and unit-propagation counts, its elapsed time, and, when satisfiable, its certified assignment. Per-ratio aggregates record the counts of SAT, UNSAT, and timeout outcomes, the satisfiable fraction, and the timeout-excluded median running time, median decisions, and median propagations.

Table 1: Experimental design for the empirical phase-transition sweep.

Parameter	Value
Variable counts n	50, 100
Instances per ratio	50 ($n = 50$); 20 ($n = 100$)
Ratio range $\alpha = m/n$	3.0 to 6.0, step 0.2 (16 points)
Total instances	1120
Per-instance time cutoff τ	5 s (wall-clock)
Solver	From-scratch DPLL (UP + PLE + MOM)
Random model	Uniform fixed-clause-length 3-CNF, distinct clauses
Verification	Independent satisfying-assignment oracle

2.6 Crossover and peak estimation

To locate the 50% satisfiability crossover we fit the empirical satisfiable fraction $\hat{f}(\alpha)$ to a decreasing logistic sigmoid

$$f(\alpha) = \frac{1}{1 + \exp\big(k\,(\alpha - \alpha_0)\big)}, \tag{2}$$

by non-linear least squares, where α_0 is the crossover ratio (the value at which $f = 0.5$) and $k > 0$ is the steepness, with larger k corresponding to a sharper transition. As a model-free robustness check we also report the crossover obtained by linear interpolation of the empirical fraction across the 0.5 level. Complexity peaks are taken as the arg-max over the ratio grid of the median running time, median decisions, and median propagations. All figures and fitted quantities are produced by an analysis script that consumes the persisted per-instance and aggregated data.

3 Results

3.1 Overview and data integrity

The complete sweep decided all 1120 instances within the 5 s cutoff: there were **zero timeouts**. Of these, 511 were determined satisfiable and 609 unsatisfiable. Every one of the 511 satisfying assignments passed the independent verifier, giving **zero verifier failures**. Because no instance timed out, the timeout-excluded medians coincide with medians over the full sample at each ratio, and no censoring correction is required. Table 2 lists the per-ratio outcomes for both system sizes.

Table 2: Per-ratio aggregated results. For each system size n and clause-to-variable ratio α we report the clause count m, the numbers of satisfiable (SAT) and unsatisfiable (UNSAT) instances, the timeout count (TO), the satisfiable fraction, and the timeout-excluded median running time, median branching decisions, and median unit propagations.

n	α	m	SAT	UNSAT	TO	frac. SAT	med. time (ms)	med. dec.	med. prop.
50	3.0	150	50	0	0	1.00	2.57	10.0	19.0
50	3.2	160	50	0	0	1.00	2.66	10.0	22.5
50	3.4	170	50	0	0	1.00	2.84	11.0	31.0
50	3.6	180	48	2	0	0.96	2.90	13.0	35.0
50	3.8	190	49	1	0	0.98	3.24	12.0	49.0
50	4.0	200	43	7	0	0.86	4.53	15.0	100.0
50	4.2	210	31	19	0	0.62	9.45	39.5	306.0
50	**4.4**	**220**	**20**	**30**	**0**	**0.40**	**12.69**	**52.0**	**407.5**
50	4.6	230	15	35	0	0.30	11.15	41.0	319.5
50	4.8	240	12	38	0	0.24	10.95	40.0	309.5
50	5.0	250	2	48	0	0.04	11.07	40.0	301.0
50	5.2	260	2	48	0	0.04	10.52	31.0	255.0
50	5.4	270	0	50	0	0.00	10.30	32.0	250.5
50	5.6	280	0	50	0	0.00	9.22	28.0	227.0
50	5.8	290	0	50	0	0.00	9.36	25.0	188.0
50	6.0	300	0	50	0	0.00	9.15	26.0	188.5
100	3.0	300	20	0	0	1.00	7.84	18.5	34.0
100	3.2	320	20	0	0	1.00	7.38	17.0	52.5
100	3.4	340	20	0	0	1.00	7.93	17.5	77.5
100	3.6	360	20	0	0	1.00	12.30	32.0	229.5
100	3.8	380	20	0	0	1.00	12.11	30.5	205.0
100	4.0	400	19	1	0	0.95	59.21	148.0	1702.0
100	4.2	420	11	9	0	0.55	186.67	355.0	4574.0
100	**4.4**	**440**	**7**	**13**	**0**	**0.35**	**254.15**	**428.0**	**5619.5**
100	4.6	460	1	19	0	0.05	174.78	298.0	3481.5
100	4.8	480	1	19	0	0.05	174.80	276.0	3299.0
100	5.0	500	0	20	0	0.00	163.08	246.0	2935.5
100	5.2	520	0	20	0	0.00	108.24	158.0	1899.0
100	5.4	540	0	20	0	0.00	103.59	149.0	1741.0
100	5.6	560	0	20	0	0.00	101.30	143.0	1637.0
100	5.8	580	0	20	0	0.00	87.15	107.0	1222.5
100	6.0	600	0	20	0	0.00	88.63	105.0	1197.5

3.2 The satisfiability transition and the 50% crossover

Figure 2 plots the empirical fraction of satisfiable instances against the constraint density for both system sizes, together with the fitted logistic curves and the theoretical threshold $\alpha_c = 4.26$. The qualitative picture is unambiguous: at low density every instance is satisfiable (the fraction is 1.0 for $\alpha \leq 3.4$ at $n = 50$ and for $\alpha \leq 3.8$ at $n = 100$), the fraction then falls steeply through the transition window, and at high density every instance is unsatisfiable (the fraction reaches 0.0 by $\alpha = 5.4$ at $n = 50$ and by $\alpha = 5.0$ at $n = 100$).

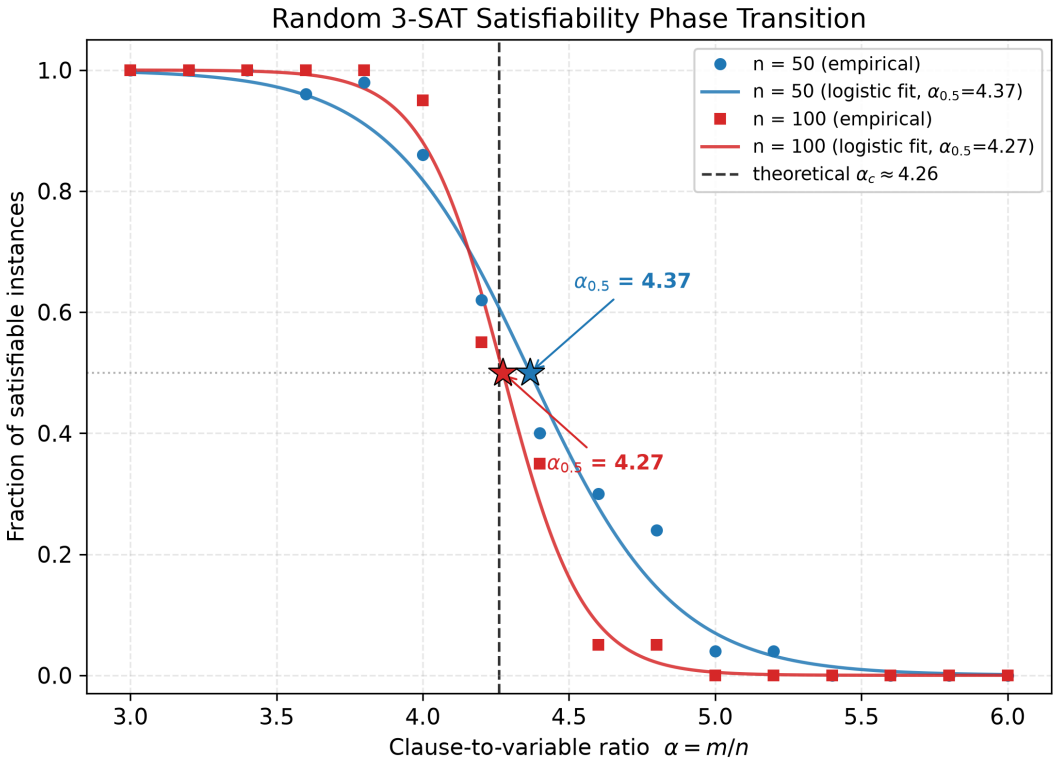

Figure 2: The random 3-SAT satisfiability phase transition. Fraction of satisfiable instances as a function of the clause-to-variable ratio $\alpha = m/n$ for $n = 50$ (blue circles) and $n = 100$ (red squares). Solid curves are decreasing-logistic fits (Eq. 2); stars mark the fitted 50% crossover for each size. The dashed vertical line is the asymptotic threshold $\alpha_c = 4.26$. The $n = 100$ curve is visibly steeper than the $n = 50$ curve, and its crossover sits closer to α_c, the expected direction of the finite-size correction.

The fitted logistic crossover lies at $\alpha_{0.5} = 4.37$ for $n = 50$ and $\alpha_{0.5} = 4.27$ for $n = 100$; the model-free linear-interpolation estimates are 4.31 and 4.25 respectively (Table 3). Both estimates fall slightly below or essentially on the asymptotic threshold $\alpha_c \approx 4.26$, and the $n = 100$ crossover is markedly closer to α_c than the $n = 50$ crossover. This is the expected behavior of the finite-size correction: at finite n the half-satisfiable point sits below the asymptotic threshold and drifts upward toward α_c as n grows. Equally telling is the fitted steepness, which rises from $k = 4.10$ at $n = 50$ to $k = 7.29$ at $n = 100$: the larger system undergoes a sharper transition, compressing the SAT-to-UNSAT drop into a narrower band of α and approaching the step function expected in the $n \to \infty$ limit.

Table 3: Fitted 50% satisfiability crossover and complexity peaks. The logistic crossover α_0 and steepness k come from Eq. 2; the interpolated crossover is a model-free cross-check. All complexity peaks occur at $\alpha = 4.4$ for both system sizes.

Quantity	$n = 50$	$n = 100$	**Theory**
Logistic crossover $\alpha_{0.5}$	4.37	4.27	4.26
Interpolated crossover $\alpha_{0.5}$	4.31	4.25	4.26
Logistic steepness k	4.10	7.29	—
Peak median time (ratio)	4.40	4.40	≈ 4.26
Peak median time (value)	12.7 ms	254 ms	—
Peak median decisions (ratio; value)	4.40; 52	4.40; 428	—
Peak median propagations (ratio; value)	4.40; 408	4.40; 5620	—

3.3 The running-time peak: an easy–hard–easy profile

Figure 3 shows the median DPLL running time as a function of α on a logarithmic vertical axis. Both curves trace the classic easy–hard–easy pattern. In the under-constrained regime ($\alpha \lesssim 3.6$) instances are cheap because a satisfying assignment is found after little search; in the over-constrained regime ($\alpha \gtrsim 5.0$) instances are again relatively cheap because dense clause sets trigger conflicts early and prune the search tree quickly. Between these regimes the median running time rises to a pronounced peak. For both $n = 50$ and $n = 100$ the peak occurs at $\alpha = 4.4$, immediately adjacent to the theoretical threshold and coincident with the empirical crossover region. The peak median running time is 12.69 ms at $n = 50$ and 254.15 ms at $n = 100$.

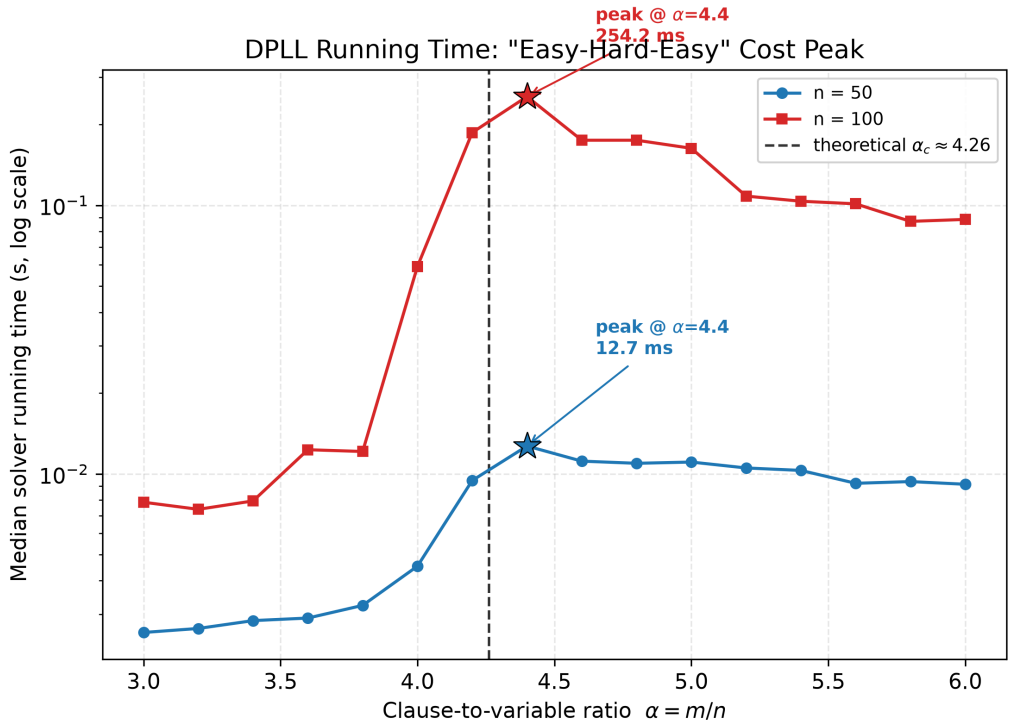

Figure 3: Median DPLL running time (easy–hard–easy). Median per-instance wall-clock time versus α on a logarithmic axis for $n = 50$ (blue) and $n = 100$ (red). Stars mark the per-size peaks, both located at $\alpha = 4.4$; the dashed line is $\alpha_c \approx 4.26$. Difficulty is concentrated in the transition window and grows sharply with system size: the peak median time rises about 20× from $n = 50$ to $n = 100$.

3.4 Hardware-independent search effort

Wall-clock time depends on the machine and on implementation details. To confirm that the hardness peak is an intrinsic property of the search rather than an artifact of timing, Figure 4 reports two hardware-independent counts: the median number of branching decisions and the median number of unit propagations, both on logarithmic axes. Both quantities reproduce the easy–hard–easy profile and peak at exactly $\alpha = 4.4$ for both system sizes, corroborating the running-time curve. At the peak, the median decision count grows from 52 at $n = 50$ to 428 at $n = 100$, and the median propagation count from 408 to 5620. Because these are pure counts of solver operations, their agreement with the wall-clock peak establishes that the concentration of difficulty near the transition is a genuine feature of the DPLL search space, not a measurement artifact.

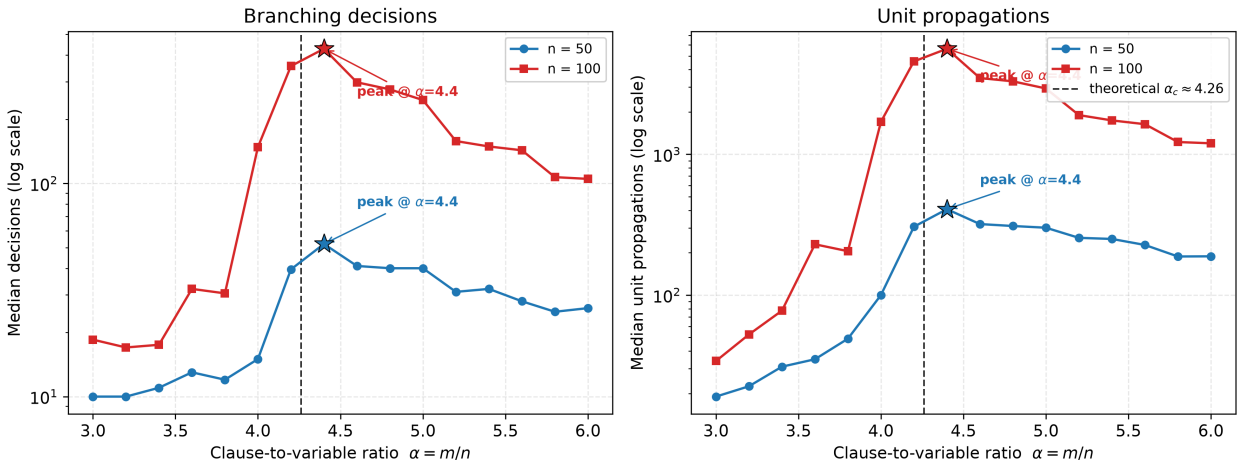

Figure 4: Hardware-independent search complexity. Median branching decisions (left) and median unit propagations (right) versus α on logarithmic axes, for $n = 50$ (blue) and $n = 100$ (red). Both metrics peak at $\alpha = 4.4$ (stars) near $\alpha_c \approx 4.26$ (dashed line), mirroring the wall-clock running-time peak and confirming that the difficulty peak is intrinsic to the search rather than an artifact of timing.

3.5 Per-instance determinations and certified assignments

Beyond the aggregate curves, the study delivers a complete per-instance record. For each of the 1120 instances we store its $(n, \alpha, \text{trial})$ coordinates, its deterministic seed, its SAT/UNSAT/timeout determination, its decision and propagation counts, its elapsed time, and—for every satisfiable instance—a total satisfying assignment together with the independent verifier's Boolean certification. As an illustration, the first satisfiable $n = 50$ instance at $\alpha = 3.0$ (seed 50030000, $m = 150$ clauses) is decided SAT and yields a complete assignment over all 50 variables whose opening values are $x_1 = \mathrm{T}, x_2 = \mathrm{T}, x_3 = \mathrm{T}, x_4 = \mathrm{F}, x_5 = \mathrm{T}, x_6 = \mathrm{T}, x_7 = \mathrm{F}, x_8 = \mathrm{T}, \ldots$; the independent verifier confirms that every one of the 150 clauses contains at least one true literal. All 511 satisfiable instances carry such a certified witness, and no unsatisfiable determination was contradicted. The full machine-readable record is provided alongside this paper.

4 Discussion

The empirical picture assembled here reproduces, with a transparent from-scratch solver, the canonical features of the random 3-SAT phase transition and—by comparing two system sizes—makes its finite-size scaling behavior directly visible. Three findings deserve emphasis.

The crossover converges to the asymptotic threshold. Our fitted half-satisfiable points, $\alpha_{0.5} = 4.37$ at $n = 50$ and 4.27 at $n = 100$ (logistic), bracket and then settle onto the asymptotic threshold $\alpha_c \approx 4.26$ predicted by the cavity method and localized experimentally by earlier work [30, 10, 31]. The model-free interpolated estimates (4.31 and 4.25) tell the same story. That the finite-n crossover sits at or slightly above/below α_c and moves toward it as n grows is exactly the drift anticipated by finite-size scaling analyses of random SAT [24], and the near-coincidence of the $n = 100$ crossover with α_c is a satisfying quantitative confirmation obtained from only 20 instances per point at the larger size.

The transition sharpens with system size. The logistic steepness increases from $k = 4.10$ at $n = 50$ to $k = 7.29$ at $n = 100$, so the SAT-to-UNSAT drop compresses into a narrower band of α as the system grows. This sharpening is the finite-size manifestation of the sharp-threshold theorem for k-SAT [17]: in the thermodynamic limit the satisfiable fraction approaches a step function at α_c, and at finite n one observes a smooth sigmoid whose width shrinks with n. Observing a nearly two-fold increase in steepness across a mere doubling of n underscores how rapidly the finite-size transition tightens.

The complexity peak grows steeply with system size. For both sizes the median running time, median decisions, and median propagations all peak at $\alpha = 4.4$, in the immediate vicinity of the crossover—the hallmark easy–hard–easy signature first documented by Mitchell *et al.* and Cheeseman *et al.* [33, 6]. The magnitude of the peak scales sharply with n: median peak running time grows about 20× (from 12.7 ms to 254 ms), median peak decisions about 8.2× (from 52 to 428), and median peak propagations about 13.8× (from 408 to 5620) as n doubles from 50 to 100. The concordance of the hardware-independent counts with the wall-clock time confirms that this growth reflects a genuine expansion of the DPLL search space near the transition, not a timing artifact. The rapid, super-linear growth of critical-region cost with n is consistent with the exponential scaling of DPLL effort at the threshold that underlies the structural picture of frozen backbones and clustered solution spaces [34, 26], and it is the practical reason that critically constrained random instances have long served as stress tests for SAT solvers [28, 8].

Scope and limitations. Our conclusions rest on modest system sizes ($n \in \{50, 100\}$) and sample sizes (50 and 20 instances per point), chosen so that a complete, non-heuristic DPLL search decides every instance well within the 5 s cutoff—a design that guaranteed zero timeouts and therefore uncensored medians, at the cost of the statistical smoothness that larger samples would provide. The grid resolution of 0.2 in α localizes the crossover and the cost peak to within that spacing; the fact that both peak metrics land on the single grid point $\alpha = 4.4$ should be read as "the peak lies in the interval around 4.4" rather than as a precise peak location. Our solver is a classical DPLL procedure rather than a modern conflict-driven clause-learning (CDCL) engine, which augments backtracking search with learned conflict clauses and non-chronological backtracking [29, 39, 35, 14]; CDCL would decide much larger instances but would not change the qualitative location of the transition, which is a property of the ensemble and not of the algorithm. Finally, we did not attempt a finite-size scaling collapse to extract a critical exponent, which would require several more system sizes and a denser grid near the crossover.

Outlook. Natural extensions include refining the ratio grid to step 0.05 within $[4.0, 4.5]$ to sharpen the crossover and peak estimates, adding further system sizes ($n = 200, 400, \dots$) to fit a finite-size scaling collapse of the form $f(\alpha) = F(n^{1/\nu}(\alpha - \alpha_c))$ and estimate the scaling exponent ν, and instrumenting the solver to track the emergence of the backbone across the transition. Substituting a CDCL solver would allow the same protocol to be pushed to system sizes where the interplay between the satisfiability threshold and the algorithmic (clustering) threshold becomes experimentally accessible.

5 Conclusion

Using a complete DPLL solver implemented from scratch—with unit propagation, pure-literal elimination, and the MOM branching heuristic—and an independent assignment verifier, we mapped

the satisfiability phase transition of uniform random 3-SAT across 1120 instances at $n = 50$ and $n = 100$ and constraint densities $\alpha \in [3.0, 6.0]$. Every instance was decided within the 5 s cutoff (zero timeouts) and every satisfiable instance carries an independently certified assignment (zero verifier failures). The fraction of satisfiable instances falls sharply through a 50% crossover at $\alpha_{0.5} = 4.37$ ($n = 50$) and 4.27 ($n = 100$), converging on the asymptotic threshold $\alpha_c \approx 4.26$, while the median running time and the hardware-independent search effort peak at $\alpha = 4.4$ in the immediate vicinity of the crossover. As the system size doubles, the transition sharpens (logistic steepness k: $4.10 \to 7.29$) and the peak solver effort grows by roughly 8–20×—the finite-size scaling signature of the random 3-SAT phase transition, recovered cleanly by a transparent, fully reproducible pipeline.

References

[1] Dimitris Achlioptas, Assaf Naor, and Yuval Peres. Rigorous location of phase transitions in hard optimization problems. *Nature*, 435(7043):759–764, 2005.

[2] Dimitris Achlioptas and Yuval Peres. The threshold for random k-SAT is $2^k \log 2 - o(k)$. *Journal of the American Mathematical Society*, 17(4):947–973, 2004.

[3] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 1999.

[4] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[5] Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

[6] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 331–337. Morgan Kaufmann, 1991.

[7] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, 1988.

[8] Cristian Coarfa, Demetrios D. Demopoulos, Alfonso San Miguel Aguirre, Devika Subramanian, and Moshe Y. Vardi. Random 3-SAT: The plot thickens. *Constraints*, 8(3):243–261, 2003.

[9] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 1971.

[10] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81(1–2):31–57, 1996.

[11] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[12] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[13] Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large k. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 59–68, 2015.

[14] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Theory and Applications of Satisfiability Testing (SAT 2003)*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2004.

[15] John Franco and Marvin Paull. Probabilistic analysis of the davis putnam procedure for solving the satisfiability problem. *Discrete Applied Mathematics*, 5(1):77–87, 1983.

[16] Jon W. Freeman. *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, University of Pennsylvania, 1995.

[17] Ehud Friedgut. Sharp thresholds of graph properties, and the k-SAT problem. *Journal of the American Mathematical Society*, 12(4):1017–1054, 1999.

[18] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[19] Ian P. Gent and Toby Walsh. The SAT phase transition. In *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI)*, pages 105–109. John Wiley and Sons, 1994.

[20] Carla P. Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24(1–2):67–100, 2000.

[21] Holger H. Hoos and Thomas Stützle. SATLIB: An online resource for research on SAT. In Ian P. Gent, Hans van Maaren, and Toby Walsh, editors, *SAT 2000: Highlights of Satisfiability Research in the Year 2000*, volume 63 of *Frontiers in Artificial Intelligence and Applications*, pages 283–292. IOS Press, 2000.

[22] Robert G. Jeroslow and Jinchang Wang. Solving propositional satisfiability problems. *Annals of Mathematics and Artificial Intelligence*, 1(1–4):167–187, 1990.

[23] Alexis C. Kaporis, Lefteris M. Kirousis, and Efthimios G. Lalas. The probabilistic analysis of a greedy satisfiability algorithm. *Random Structures & Algorithms*, 28(4):444–480, 2006.

[24] Scott Kirkpatrick and Bart Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264(5163):1297–1301, 1994.

[25] Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Yannis C. Stamatiou. Approximating the unsatisfiability threshold of random formulas. *Random Structures & Algorithms*, 12(3):253–269, 1998.

[26] Florent Krzakala, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proceedings of the National Academy of Sciences*, 104(25):10318–10323, 2007.

[27] Leonid A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

[28] Kevin Leyton-Brown, Holger H. Hoos, Frank Hutter, and Lin Xu. Understanding the empirical hardness of np-complete problems. *Communications of the ACM*, 57(5):98–107, 2014.

[29] João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.

[30] Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold values of random k-SAT from the cavity method. *Random Structures & Algorithms*, 28(3):340–373, 2006.

[31] Marc Mézard, Giorgio Parisi, and Riccardo Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.

[32] Marc Mézard and Riccardo Zecchina. Random k-satisfiability problem: From an analytic solution to an efficient algorithm. *Physical Review E*, 66(5):056126, 2002.

[33] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI)*, pages 459–465. AAAI Press, 1992.

[34] Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400(6740):133–137, 1999.

[35] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Annual Design Automation Conference (DAC)*, pages 530–535, 2001.

[36] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI)*, pages 337–343. AAAI Press, 1994.

[37] Bart Selman, Henry A. Kautz, and Bram Cohen. Local search strategies for satisfiability testing. In David S. Johnson and Michael A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532. American Mathematical Society, 1996.

[38] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI)*, pages 440–446. AAAI Press, 1992.

[39] Lintao Zhang, Conor F. Madigan, Matthew H. Moskewicz, and Sharad Malik. Efficient conflict driven learning in a boolean satisfiability solver. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 279–285, 2001.