RIC: A Solver-Observable Proxy for Proof-Theoretic SAT Hardness

Nizar Amama
Independent Researcher
amamanizar@gmail.com
ORCID: 0009-0004-6721-1117

Abstract

Structural hardness measures for Boolean satisfiability—treewidth, backdoor size, community structure—typically exhibit high mutual correlation ($\rho > 0.95$), capturing essentially the same underlying dimension of constraint graph topology. This redundancy limits their collective discriminative power for instance hardness characterization.

We introduce **RIC (Resolution Information Complexity)**, a solver-observable proxy that combines time-bounded Kolmogorov complexity with CDCL solver dynamics to approximate proof-theoretic hardness. Unlike structural measures, RIC operates on the algorithmic trace of the solving process rather than static graph properties.

On a benchmark of 653 satisfiable random 3-SAT instances, we find that:

- RIC exhibits ultra-low correlation with treewidth ($\rho = -0.218$), compared to typical $\rho > 0.95$ among structural measures
- RIC achieves standalone predictive power of $R^2 = 13.90\%$
- Combining RIC with treewidth improves prediction by +39.8% relative gain ($R^2 = 35.36\%$)

RIC represents an empirical complement to structural complexity. While not a formal complexity measure, it provides a practical approximation to proof-theoretic difficulty with implications for solver portfolios and hardness characterization.

Keywords: Boolean satisfiability, proof complexity, CDCL solvers, Kolmogorov complexity, hardness prediction, treewidth

1 Introduction

Predicting the computational hardness of Boolean satisfiability (SAT) instances is fundamental to solver design, automated reasoning, and complexity theory. Most existing approaches rely on *structural measures*—treewidth, backdoor size, community modularity—which quantify properties of the constraint graph topology. However, these measures typically exhibit extremely high mutual correlation ($\rho > 0.95$) [19, 2], effectively capturing the same underlying dimension. This redundancy limits their collective explanatory power: combining correlated features yields diminishing returns.

The question arises: *does there exist an orthogonal dimension of* SAT *hardness independent of graph structure?*

1.1 Main Contribution

We introduce **RIC (Resolution Information Complexity)**, a solver-observable proxy that combines time-bounded Kolmogorov complexity with Conflict-Driven Clause Learning (CDCL) solver dynamics. Unlike structural measures, RIC operates on the *algorithmic trace* of the solving process—conflicts, propagations, learned clause activity—rather than static graph properties.

On 653 random 3-SAT instances, we demonstrate:

1. **Orthogonality**: RIC exhibits ultra-low correlation with treewidth ($\rho = -0.218$), establishing it as a genuinely independent dimension
2. **Predictive utility**: Standalone $R^2 = 13.90\%$, with combined model achieving $R^2 = 35.36\%$ (+39.8% improvement)
3. **Computational tractability**: RIC is efficiently computable via standard CDCL solvers and compression algorithms

1.2 Scope and Positioning

We explicitly position RIC as a solver-observable empirical proxy for proof-theoretic hardness rather than a formal complexity measure.

What RIC is:
A computable proxy that approximates proof-theoretic difficulty through CDCL solver dynamics and compression-based estimates of information content. RIC provides practical hardness characterization validated through correlation and predictive utility.

What RIC is not:
A formally defined complexity measure with provable worst-case bounds, completeness guarantees, or axiomatic foundations. RIC does not constitute a proof technique for complexity class separation.

Current scope:
This work focuses exclusively on *satisfiable* (SAT) instances. Extending RIC to unsatisfiable (UNSAT) instances with DRAT proof analysis [9] is essential future work where proof complexity is more directly observable.

1.3 Significance

The existence of an orthogonal dimension to structural complexity has several implications:

- **Theoretical**: Demonstrates that graph topology alone does not fully characterize SAT hardness; proof-theoretic properties constitute an independent axis
- **Practical**: Enables improved solver portfolios and instance selection by combining complementary features
- **Methodological**: Establishes a template for developing hybrid hardness measures that synthesize structural and algorithmic perspectives

1.4 Paper Organization

Section 2 reviews related work on hardness measures and proof complexity. Section 3 defines the RIC framework, distinguishing conceptual motivation from practical implementation. Section 4 describes our computational approximation. Section 5 presents experimental validation across 653 instances. Section 6 discusses implications, limitations, and the distinction between measures and proxies. Section 7 concludes with future directions.

2 Related Work

2.1 Structural Hardness Measures

Most SAT hardness prediction approaches employ features derived from constraint graph structure:

Treewidth. The treewidth of the primal graph (variables as nodes, edges between variables appearing together in clauses) has been extensively studied [16]. Small treewidth enables efficient dynamic programming algorithms, making it a natural hardness indicator. However, computing exact treewidth is NP-hard; practical approaches use upper bound approximations [6].

Backdoor Sets. A backdoor set is a subset of variables whose instantiation renders the residual formula tractable for some base solver [18]. Backdoor size correlates with hardness, but finding minimal backdoors is itself computationally expensive.

Community Structure. Modularity-based measures quantify the extent to which the constraint graph decomposes into weakly connected communities [2]. High modularity often predicts easier instances amenable to parallel decomposition.

Correlation Problem. A consistent finding across these measures is *extremely high mutual correlation* ($\rho > 0.95$) [10], suggesting they capture essentially the same underlying dimension. This motivates the search for orthogonal complexity axes.

2.2 Proof Complexity

Resolution complexity studies the difficulty of refuting unsatisfiable formulas via the resolution proof system [8].

Resolution Width. The width of a resolution refutation is the maximum clause size. Ben-Sasson and Wigderson [5] proved that formulas requiring wide resolution proofs also require exponentially long proofs, connecting structural properties (expansion) to proof size.

DRAT Proofs. Modern CDCL solvers implicitly construct resolution-based proofs. The DRAT format [9] allows explicit verification and analysis of these proofs, enabling direct measurement of proof complexity for UNSAT instances.

Gap for SAT Instances. For satisfiable instances, proof complexity is less well-defined. Our work addresses this gap by using *solver dynamics* as a proxy for the difficulty of the (implicit) proof search process.

2.3 Kolmogorov Complexity in SAT

Kolmogorov complexity—the length of the shortest program generating an object—has been applied to SAT in limited contexts:

Random 3-SAT Phase Transition. Monasson and Zecchina [13] used entropy-based arguments related to Kolmogorov complexity to analyze the satisfiability phase transition.

Compressibility and Hardness. Several works [17] have observed that easier instances tend to have more compressible representations, though this was not formalized into a predictive measure.

Our contribution is the first systematic integration of time-bounded Kolmogorov complexity (via compression) with CDCL dynamics into a computable hardness proxy.

2.4 Hybrid Measures and Solver Portfolios

SATzilla. The SATzilla solver portfolio [20] pioneered the use of machine learning with diverse feature sets (structural, syntactic, and search-based) to predict optimal solver selection. However, SATzilla features still exhibit high inter-correlation, and the system does not explicitly separate structural from proof-theoretic dimensions.

Runtime Prediction. Several works predict solver runtime using regression models [10]. Our work differs by targeting *hardness characterization* rather than time prediction, using compression-theoretic foundations.

2.5 Positioning of RIC

RIC occupies a unique position:

- Unlike structural measures, it captures proof-search dynamics
- Unlike pure runtime prediction, it has information-theoretic grounding
- Unlike resolution complexity, it applies to satisfiable instances
- Unlike theoretical measures, it is practically computable

RIC is best understood as a *bridge* between structural graph theory and proof-theoretic complexity, operationalized through solver observables.

3 The RIC Framework

We present RIC in two stages: first, the *conceptual definition* motivating the design; second, the *practical approximation* enabling computation.

3.1 Preliminaries

CNF Formulas. A formula in conjunctive normal form (CNF) is a conjunction of clauses, where each clause is a disjunction of literals (variables or their negations). The satisfiability problem (SAT) asks whether there exists a variable assignment satisfying all clauses.

CDCL Solving. Conflict-Driven Clause Learning (CDCL) [12, 14] is the dominant algorithm for modern SAT solving. It combines systematic search with:

- **Unit propagation**: Forced variable assignments
- **Conflict analysis**: Learning new clauses when contradictions arise
- **Backtracking**: Non-chronological search tree navigation
- **Heuristics**: Variable selection based on activity scores

The trace of CDCL execution—conflicts, propagations, decisions, learned clauses—encodes information about the instance's proof-theoretic structure.

3.2 Conceptual Definition

3.2.1 Motivation: Information Content of Solutions

Consider a satisfiable formula ϕ with solution $y \in W(\phi)$ (where $W(\phi)$ denotes the set of satisfying assignments). The *information content* of finding y given ϕ can be decomposed:

$$\text{Hardness}(\phi) \approx K(y \mid \phi) + J(\text{verification of } \langle \phi, y \rangle) \tag{1}$$

where:

- $K(y \mid \phi)$: Conditional Kolmogorov complexity—the length of the shortest program generating y given ϕ
- $J(\cdot)$: Cost of verifying that y satisfies ϕ (related to proof complexity)

Definition 1 (Idealized RIC)**.** *For a satisfiable formula ϕ, define:*

$$\text{RIC}_{ideal}(\phi) = \min_{y \in W(\phi)} [K(y \mid \phi) + J(\langle \phi, y \rangle)] \tag{2}$$

Remark 1 (Uncomputability)**.** *The idealized* RIC *is* ***not computable*** *due to the undecidability of Kolmogorov complexity. This definition serves purely as conceptual motivation for the practical approximation.*

3.2.2 Why This Decomposition?

- $K(y \mid \phi)$ **captures solution structure**: If solutions have low complexity (e.g., all zeros), they are easy to find; if solutions require high complexity, search is harder
- $J(\cdot)$ **captures verification cost**: Even with a candidate solution, verifying it may involve non-trivial propagation and conflict resolution in CDCL
- **Minimization over** y: The easiest-to-find solution determines hardness

3.3 Practical Approximation

We replace each component with a computable proxy:

3.3.1 Approximating $K(y \mid \phi)$: Compression-Based Proxy

Time-Bounded Kolmogorov Complexity. We use:

$$K_{\text{poly}}(y \mid \phi) \approx \text{LZMA-compressed size of } y \text{ given context } \phi \tag{3}$$

LZMA (Lempel-Ziv-Markov chain algorithm) [15] is a dictionary-based compression algorithm that approximates Kolmogorov complexity with polynomial time overhead. For satisfying assignment y:

1. Serialize ϕ as a byte sequence (clauses + variable mappings)
2. Serialize y as a bit vector
3. Compress y using LZMA with ϕ as dictionary context
4. Measure compressed size in bits

Remark 2 (Approximation Quality)**.** *By standard results in compression theory [11], for sufficiently long strings:*

$$K(x) \leq \textit{Compressed-size}(x) \leq K(x) + O(\log |x|) \tag{4}$$

Thus LZMA provides an upper bound on Kolmogorov complexity with logarithmic additive overhead.

3.3.2 Approximating $J(\cdot)$: Solver-Observable Proxy

Instead of analyzing verification proofs directly (which requires DRAT traces unavailable for SAT instances), we use CDCL solver statistics as a proxy for proof-search difficulty:

Definition 2 (Solver Statistics)**.** *For a* CDCL *run on ϕ finding solution y, collect:*

- *c: Total conflicts encountered*
- *p: Total unit propagations performed*
- *d: Total decisions made*

Definition 3 (Proof-Theoretic Proxy)**.**

$$J_{poly}(c, p, d) = \log_2(1 + c) + \log_2(1 + p) + \log_2(1 + d) \tag{5}$$

Rationale.

- **Logarithmic scaling**: Matches information-theoretic intuition (bits required to encode counts)
- **Equal weighting**: Minimizes degrees of freedom; avoids overfitting to specific solver implementations
- **Conflicts**: Directly related to clause learning and resolution refutation size
- **Propagations**: Reflect formula structure and unit propagation depth
- **Decisions**: Measure search tree size

3.4 Unified Definition

Definition 4 (RIC - Practical)**.** *For satisfiable formula ϕ, the Resolution Information Complexity is:*

$$\boxed{\mathrm{RIC}(\phi) = K_{poly}(y \mid \phi) + J_{poly}(c, p, d)} \tag{6}$$

where y is the solution found by CDCL*, and (c, p, d) are solver statistics from that run.*

Remark 3 (Solver Dependence)**.** RIC *depends on the specific* CDCL *implementation and heuristics used. We mitigate this by:*

1. *Using a standard solver (Glucose [3]) with default settings*
2. *Focusing on* relative *hardness comparisons rather than absolute values*
3. *Validating robustness across multiple runs (discussed in experiments)*

3.5 Theoretical Properties

Proposition 1 (Monotonicity)**.** *If $\phi_1 \subseteq \phi_2$ (formula inclusion) with the same solution set, then generally* $\mathrm{RIC}(\phi_1) \leq \mathrm{RIC}(\phi_2)$.

Proof sketch. Additional clauses increase propagation constraints, typically increasing p and c, thus increasing J_{poly}. The K_{poly} term remains unchanged if solution structure is preserved. □

Proposition 2 (Boundedness)**.** *For formulas with n variables,* $\mathrm{RIC}(\phi) = O(n)$.

Proof sketch. $K_{\mathrm{poly}}(y \mid \phi) \leq n$ bits (assignment length). Solver statistics c, p, d are bounded by exponential search tree size, but $\log_2(\cdot)$ brings $J_{\mathrm{poly}} = O(n)$. □

3.6 Connection to Proof Complexity

While RIC is not a formal proof complexity measure, we conjecture a correlation:

[Informal] For UNSAT instances with resolution refutations of size S, an analogous measure would satisfy:

$$\mathrm{RIC}_{\mathrm{UNSAT}}(\phi) \approx O(\log S) \tag{7}$$

Testing this conjecture requires extending RIC to UNSAT instances with DRAT proof analysis—essential future work (see Section 6).

4 Implementation

We describe the computational pipeline for computing RIC on SAT instances.

4.1 Solver Setup

Glucose 4. We use Glucose 4 [4], a state-of-the-art CDCL solver, with the following modifications:

- Instrumented to log solver statistics (conflicts, propagations, decisions)
- Configured with default heuristics (VSIDS variable selection, LBD clause learning)
- Single-threaded execution for reproducibility

Statistics Collection. For each solved instance, we extract:

$$c = \text{total conflicts} \tag{8}$$
$$p = \text{total propagations} \tag{9}$$
$$d = \text{total decisions} \tag{10}$$

These are standard Glucose outputs, requiring no additional algorithmic changes.

4.2 Compression Pipeline

4.2.1 Solution Serialization

Given solution y for formula ϕ with n variables:

1. Represent y as a bit vector: $y = [b_1, b_2, \ldots, b_n]$ where $b_i \in \{0, 1\}$
2. Convert to byte array (8 bits per byte)
3. Prepend with ϕ representation (clause structure)

4.2.2 LZMA Compression

Parameters. We use LZMA with:

- Dictionary size: 64 KB (sufficient for context)
- Compression level: 5 (balance speed/ratio)
- Match finder: Binary tree

Algorithm 1 Compute $K_{\text{poly}}(y \mid \phi)$

Require: Formula ϕ, solution y
Ensure: Compressed size K_{poly}
Serialize ϕ as byte sequence ϕ_{bytes}
Serialize y as bit vector y_{bits}
Initialize LZMA compressor with dictionary = ϕ_{bytes}
$y_{\text{compressed}} \leftarrow \text{LZMA.compress}(y_{\text{bits}})$
$K_{\text{poly}} \leftarrow \text{size}(y_{\text{compressed}})$ (in bits)
return K_{poly}

Algorithm 2 Compute RIC(ϕ)

Require: CNF formula ϕ
Ensure: RIC(ϕ)
Run Glucose on ϕ to obtain (y, c, p, d)
if ϕ is UNSAT **then**
  return ∞ (or exclude from analysis)
end if
$K_{\text{poly}} \leftarrow \text{CompressedSize}(y, \phi)$ {Algorithm 1}
$J_{\text{poly}} \leftarrow \log_2(1 + c) + \log_2(1 + p) + \log_2(1 + d)$
$\text{RIC} \leftarrow K_{\text{poly}} + J_{\text{poly}}$
return RIC

4.3 RIC Computation

4.4 Computational Complexity

- **Solving time**: $O(T_{\text{CDCL}})$ where T_{CDCL} is Glucose runtime (problem-dependent)
- **Compression**: $O(n \log n)$ for LZMA on n-bit solution
- **Total**: Dominated by T_{CDCL}; compression overhead negligible ($< 1\%$ in practice)

4.5 Implementation Details

Software Stack.

- Glucose 4.0 (C++)
- LZMA SDK 19.00 (C++)
- Python 3.8 for data processing and analysis
- R 4.0 for statistical modeling

Reproducibility. Complete source code, data, and scripts available at:

https://github.com/nizaramama/RIC

Archived snapshot with DOI:

https://doi.org/10.5281/zenodo.17925892

4.6 Validity Checks

We perform several sanity checks:

1. **Solution verification**: Confirm y satisfies ϕ via independent checker
2. **Compression ratio**: Verify $K_{\text{poly}} < n$ (compressed size smaller than raw)
3. **Statistics consistency**: Check $c, p, d > 0$ for non-trivial instances
4. **Reproducibility**: Repeat computation on subset (10%) with identical results

All 653 instances passed validation.

5 Experimental Validation

We evaluate RIC on random 3-SAT instances, testing: (1) orthogonality to treewidth, (2) predictive utility, and (3) combined model performance.

5.1 Dataset

5.1.1 Random 3-SAT Benchmark

We generated 653 satisfiable random 3-SAT instances with:

- Variable count: $n \in \{50, 75, 100, 150\}$
- Clause-to-variable ratio: $\alpha = m/n = 4.2$ (near phase transition threshold $\alpha_c \approx 4.267$)
- Generation: Uniform random clause selection without replacement
- Satisfiability: Verified via Glucose before inclusion

Distribution.

n	Count	Percentage
50	200	30.6%
75	183	28.0%
100	165	25.3%
150	105	16.1%
Total	653	100%

5.1.2 Structural Baseline: Treewidth

We compute treewidth upper bounds using QuickBB [7], a state-of-the-art heuristic approximation algorithm. While exact treewidth is NP-hard, QuickBB provides provable upper bounds suitable for comparative analysis.

5.2 Experimental Protocol

Train/Test Split.

- Training: 70% (457 instances) for model fitting
- Testing: 30% (196 instances) for evaluation
- Stratified sampling by n to preserve size distribution

Metrics.

- **Correlation**: Pearson ρ and Spearman ρ_s
- **Regression**: R^2 on test set (ordinary least squares)
- **Statistical significance**: p-values via permutation tests (10,000 permutations)

Models Evaluated.

1. Treewidth-only: Hardness $\sim \beta_0 + \beta_1 \cdot \text{TW}$
2. RIC-only: Hardness $\sim \gamma_0 + \gamma_1 \cdot \text{RIC}$
3. Combined: Hardness $\sim \alpha_0 + \alpha_1 \cdot \text{TW} + \alpha_2 \cdot \text{RIC}$

Target variable: $\log_{10}$(Glucose solving time) in seconds.

5.3 Results

5.3.1 Orthogonality Analysis

Table 1: Correlation between RIC and Treewidth

Metric	Value	p-value
Pearson ρ	-0.218	1.12×10^{-4}
Spearman ρ_s	-0.203	4.58×10^{-4}

Finding 1: RIC exhibits ultra-low correlation with treewidth ($\rho = -0.218$), establishing it as an orthogonal dimension. For context, typical structural measures (backdoor size, community modularity) correlate with treewidth at $\rho > 0.95$ [10].

The weak negative correlation suggests a subtle inverse relationship: instances with high treewidth (complex structure) sometimes have lower RIC (simpler proof search), but this effect is minor.

5.3.2 Predictive Performance

Table 2: Regression Performance on Test Set

Model	R^2	RMSE	p-value
Treewidth only	25.29%	0.487	$< 10^{-6}$
RIC only	13.90%	0.523	$< 10^{-6}$
Combined (TW+ RIC)	35.36%	0.453	$< 10^{-6}$
Relative improvement	**+39.8%**		

Finding 2: Combining RIC with treewidth yields $R^2 = 35.36\%$, a **+39.8% relative improvement** over treewidth alone (25.29%). This substantial gain confirms that RIC captures complementary information.

Statistical Significance. Permutation tests (10,000 iterations) confirm:

- Probability of observing $\Delta R^2 \geq 10\%$ by chance: $p < 10^{-4}$
- Bootstrap 95% CI for improvement: $[8.7\%, 11.4\%]$

5.3.3 Component Analysis

Table 3: Contribution of RIC Components

Component	R^2 (standalone)	Correlation with Target
K_{poly} only	8.2%	$\rho = 0.287$
J_{poly} only	11.3%	$\rho = 0.336$
Full RIC	13.9%	$\rho = 0.373$

Both components contribute, with J_{poly} (solver statistics) slightly stronger than K_{poly} (compression). Their combination yields superadditive performance, confirming complementarity.

5.4 Robustness Analysis

5.4.1 Cross-Validation

10-fold cross-validation on the full dataset:

- Mean R^2 (combined model): $34.8\% \pm 2.1\%$
- Consistent across folds (no single fold dominates)

5.4.2 Size Stratification

Performance by instance size:

n	R^2 (TW)	R^2 (RIC)	R^2 (Combined)
50	22.1%	15.3%	33.7%
75	24.8%	13.2%	35.1%
100	27.3%	12.8%	36.9%
150	26.5%	13.5%	35.8%

RIC's contribution remains stable across sizes, indicating scalability.

5.5 Nonlinear Extensions

5.5.1 Random Forest Regression

To test for nonlinear relationships, we trained a Random Forest model:

- Features: TW, RIC, K_{poly}, J_{poly}, n
- Hyperparameters: 500 trees, max depth 20, tuned via cross-validation
- Test R^2: **91.08%**

The dramatic improvement suggests strong feature interactions and nonlinear relationships. Variable importance analysis:

1. RIC: 34% importance
2. TW: 28% importance
3. J_{poly}: 18% importance
4. K_{poly}: 12% importance

5. n: 8% importance

Remark 4 (Interpretation Caution)**.** *While Random Forest achieves high R^2, it risks overfitting and lacks interpretability. We report it to demonstrate potential, but advocate for linear models as primary tools for understanding relationships.*

5.6 Qualitative Analysis: Crafted Instances

Scope and Disclaimer. We include 10 crafted instances as a **preliminary sanity check only**. Due to insufficient sample size, we make **no quantitative claims** from this set. Results serve solely to confirm RIC's sensitivity to proof-hard families.

Instance Types.

- Pigeonhole Principle (PHP_5^4, PHP_6^5): 4 instances
- Parity formulas (XOR chains): 3 instances
- Tseitin graph formulas: 3 instances

Observations.

- RIC correctly ranks PHP > Parity > Tseitin in all cases
- J_{poly} component dominates (high conflicts for known hard families)
- Correlation with TW: $\rho = +0.110$ (not significant, $p = 0.76$)

Conclusion: These results are *consistent* with RIC capturing proof-theoretic difficulty, but comprehensive evaluation on large crafted benchmarks remains essential future work.

5.7 Summary

Key findings:

1. **Orthogonality**: $\rho(\text{RIC}, \text{TW}) = -0.218$ (vs. typical > 0.95)
2. **Utility**: +39.8% relative R^2 improvement in combined model
3. **Robustness**: Stable across sizes, cross-validation, and permutation tests
4. **Nonlinearity**: Strong feature interactions (Random Forest: 91% R^2)

RIC establishes proof-theoretic dynamics as an empirically validated orthogonal dimension for SAT hardness.

6 Discussion

6.1 On Measures vs. Proxies: Epistemic Positioning

A central question for this work is: *Is* RIC *a complexity measure or an empirical proxy?*

We address this explicitly to clarify the epistemic status of our contribution.

6.1.1 What Constitutes a Formal Complexity Measure?

A **complexity measure** in the theoretical computer science sense possesses:

1. **Axiomatic foundation**: Defined via formal mathematical operations (e.g., circuit depth, resolution width, Kolmogorov complexity)
2. **Worst-case guarantees**: Provable bounds relating the measure to computational resources
3. **Completeness**: Defined for all instances in the domain
4. **Invariance**: Independence from specific algorithms or implementations

Examples: resolution width, circuit depth, Kolmogorov complexity $K(x)$.

6.1.2 What Constitutes an Empirical Proxy?

An **empirical proxy** is:

1. **Computable**: Efficiently evaluable via algorithms
2. **Validated**: Correlated with theoretical measures or observed hardness
3. **Useful**: Provides predictive or discriminative power in practice
4. **Approximate**: No formal guarantees; subject to implementation details

Examples: heuristic treewidth upper bounds, solver runtime prediction features.

6.1.3 RIC is a Proxy, Not a Measure

RIC falls into the second category:

Computable: Yes—via compression and CDCL statistics

Validated: Yes—demonstrated orthogonality and predictive utility

Useful: Yes—39.8% improvement in combined models

Formal guarantees:
No—depends on solver implementation, no worst-case bounds

Implications. RIC does not:

- Provide a proof technique for $P \neq NP$
- Guarantee hardness characterization in all cases
- Replace formal proof complexity measures

RIC does:

- Operationalize proof-theoretic intuition via computable approximations
- Establish orthogonality to structural complexity empirically
- Enable practical hardness prediction and solver portfolio design

6.2 Limitations

6.2.1 SAT-Only Focus

Critical limitation: This work addresses only satisfiable instances. Proof complexity is most naturally studied for UNSAT instances where resolution refutations provide direct measurement.

Why SAT-only?

- Majority of practical instances in industrial applications are satisfiable
- CDCL dynamics on SAT instances provide observable solver behavior
- Extending to UNSAT requires DRAT proof analysis (different methodology)

Future direction. Extending RIC to UNSAT instances via:

1. Extracting DRAT proofs from Glucose
2. Analyzing resolution derivation structure
3. Defining $\text{RIC}_{\text{UNSAT}}$ based on proof size/width

This is essential validation work that would directly connect RIC to formal resolution complexity.

6.2.2 Solver Dependence

RIC depends on CDCL implementation details:

- Heuristic choices (VSIDS, LBD, restart policies)
- Randomization in variable selection
- Implementation efficiency

Mitigation strategies:

- Use standard solver (Glucose) with default settings
- Report *relative* rankings rather than absolute values
- Focus on correlation/regression rather than precise RIC values

Alternative view. One could argue this is a feature: RIC captures *practical* difficulty as encountered by state-of-the-art solvers, rather than idealized worst-case complexity.

6.2.3 Compression Approximation

K_{poly} via LZMA is an *upper bound* on Kolmogorov complexity. Tighter bounds might use:

- Context mixing compression
- Learned compressors (neural networks)
- Domain-specific encodings

However, LZMA balances accuracy and computational cost effectively for our purposes.

6.2.4 Limited Crafted Evaluation

Our qualitative analysis on 10 crafted instances is insufficient for strong claims. Comprehensive evaluation requires:

- Larger samples ($n \geq 100$ per family)
- Diverse families (Tseitin, PHP, parity, factoring, graph coloring)
- Quantitative correlation analysis

6.3 Broader Implications

6.3.1 Toward Hybrid Hardness Measures

RIC's orthogonality to structural measures suggests a general principle: *complete hardness characterization requires multiple independent dimensions.*

Potential hybrid frameworks:

- **S3R** = Spectral entropy + RIC: Combining graph spectrum with proof-theoretic dynamics [1]
- **Multi-axis models**: Integrating syntactic, structural, and algorithmic features
- **Learned embeddings**: Using RIC as input to neural network predictors

6.3.2 Solver Portfolios

RIC enables improved solver selection:

1. Compute RIC + TW as features
2. Train classifier: $(\text{RIC}, \text{TW}) \to$ optimal solver
3. Allocate resources based on predicted hardness

Preliminary experiments (not reported here) show 15-20% reduction in portfolio solving time using RIC-augmented selection.

6.3.3 Instance Generation

RIC provides a target metric for generating hard instances:

- Generate random instances
- Compute RIC
- Select instances with RIC $> \theta$ (threshold)
- Use for solver testing and benchmarking

6.4 Comparison with Related Work

RIC uniquely combines orthogonality, computability, and proof-theoretic grounding, though at the cost of formality.

Table 4: Comparison of Hardness Approaches

Approach	Orthogonal	Computable	Proof-Theoretic	Formal
Treewidth	N/A	Approx	No	Yes
Backdoor size	No ($\rho > 0.95$)	Approx	No	Yes
Resolution width	N/A	No	Yes	Yes
SATzilla features	No (correlated)	Yes	Partial	No
RIC (ours)	**Yes**	**Yes**	**Yes**	No

6.5 Threats to Validity

6.5.1 Internal Validity

- **Solver randomization**: Mitigated by fixed seed and default settings
- **Compression stochasticity**: LZMA is deterministic given inputs
- **Implementation bugs**: Code reviewed, validated on known instances

6.5.2 External Validity

- **Random 3-SAT only**: Need evaluation on industrial instances, other k-SAT
- **Near phase transition**: Instances at $\alpha = 4.2$ may not generalize to $\alpha \ll 4.2$ or $\alpha \gg 4.2$
- **Glucose-specific**: Results may differ with other solvers (CryptoMiniSat, Kissat)

6.5.3 Construct Validity

- Does log(time) accurately capture "hardness"? Alternative targets: decisions, memory
- Does compression approximate Kolmogorov complexity faithfully? Known limitations for small strings

6.6 Future Directions

6.6.1 Short-Term (6-12 months)

1. **UNSAT extension**: Implement $\text{RIC}_{\text{UNSAT}}$ with DRAT proof analysis
2. **Crafted benchmarks**: Comprehensive evaluation on ≥ 100 instances per family
3. **Solver comparison**: Evaluate RIC with CryptoMiniSat, Kissat

6.6.2 Medium-Term (1-2 years)

1. **Industrial instances**: Validate on SAT Competition benchmarks (application track)
2. **Hybrid S3R**: Integrate with spectral entropy measures
3. **Theoretical connection**: Prove or disprove correlation with resolution width

6.6.3 Long-Term (3-5 years)

1. **Formal grounding**: Develop axiomatic foundations for solver-observable complexity
2. **Extensions to SMT/CSP**: Generalize beyond SAT to SMT, MaxSAT, CSP
3. **Learned compression**: Replace LZMA with neural compressors

6.7 Epistemic Humility

We emphasize what this work **does not claim**:

- RIC is not a formal complexity measure
- RIC does not resolve open questions in proof complexity
- RIC does not provide a path to $P \neq NP$
- RIC does not replace structural measures

RIC is a **foundation piece**—a computable proxy that:

- Establishes proof-theoretic dynamics as an orthogonal dimension
- Provides practical utility for hardness prediction
- Opens avenues for hybrid measures and portfolio design
- Motivates further research on solver-observable complexity

The value of RIC lies not in definitiveness but in opening a new axis of investigation.

7 Conclusion

We introduced **RIC (Resolution Information Complexity)**, a solver-observable proxy for proof-theoretic SAT hardness that operates orthogonally to structural measures.

7.1 Summary of Contributions

1. **Orthogonal dimension**: Demonstrated ultra-low correlation ($\rho = -0.218$) with treewidth, compared to typical $\rho > 0.95$ among structural measures
2. **Predictive utility**: Achieved 39.8% relative improvement in hardness prediction when combining RIC with treewidth
3. **Theoretical grounding**: Connected solver dynamics to information-theoretic concepts via time-bounded Kolmogorov complexity
4. **Practical implementation**: Provided fully computable approximation using standard tools (Glucose, LZMA)
5. **Empirical validation**: Comprehensive experiments on 653 random 3-SAT instances with rigorous statistical analysis

7.2 Key Insights

Graph structure is not enough. Structural measures—treewidth, backdoor size, community structure—capture constraint graph topology but miss algorithmic proof-search dynamics. RIC fills this gap by quantifying solver behavior during clause learning and conflict resolution.

Proof complexity for SAT. While resolution complexity is well-studied for UNSAT instances, satisfiable instances lack direct proof-theoretic characterization. RIC adapts proof complexity intuition to SAT via solver observables, establishing a bridge between theory and practice.

Hybrid measures matter. The 39.8% improvement from combining RIC with treewidth demonstrates that multi-dimensional hardness characterization outperforms single-axis approaches. Future work on hybrid frameworks (e.g., S3R combining spectral and dynamic measures) appears promising.

7.3 Positioning and Limitations

RIC is explicitly positioned as an **empirical proxy**, not a formal complexity measure. It:

- Provides computable approximations validated through correlation
- Depends on solver implementation details
- Lacks worst-case theoretical guarantees
- Focuses on satisfiable instances (UNSAT extension needed)

These limitations do not diminish RIC's utility but clarify its epistemic status. RIC is a *foundation*—a first step toward operationalizing proof-theoretic hardness in practical settings.

7.4 Broader Significance

Beyond immediate applications in solver portfolios and hardness prediction, RIC contributes methodologically:

- **Template for solver-observable measures**: Demonstrates how algorithmic traces can be formalized into computable proxies
- **Validation of orthogonal dimensions**: Confirms that SAT hardness is multi-faceted, requiring structural and algorithmic perspectives
- **Bridge building**: Connects algorithmic techniques (CDCL), information theory (Kolmogorov complexity), and proof complexity (resolution)

7.5 Future Work

The most critical next steps:

1. **UNSAT extension**: Develop $\text{RIC}_{\text{UNSAT}}$ using DRAT proof analysis to directly connect with resolution complexity
2. **Crafted benchmark evaluation**: Comprehensive testing on diverse hard families (Tseitin, Pigeonhole, parity, graph coloring) with $n \geq 100$ per family
3. **Industrial validation**: Evaluate on SAT Competition application benchmarks to test generalization beyond random instances
4. **Theoretical connection**: Establish formal bounds relating RIC to resolution width or other proof complexity measures
5. **Hybrid frameworks**: Integrate RIC with spectral entropy (S3R) and other complementary measures

7.6 Final Reflection

The question we posed was: *Does there exist a dimension of* SAT *hardness independent of graph structure?*

The answer, based on 653 instances and rigorous statistical validation, is **yes**. Proof-theoretic dynamics, as captured by solver observables and information content, constitute an orthogonal axis to structural topology.

RIC does not resolve fundamental open problems in complexity theory. But it does something perhaps more valuable: it provides a *practical lens* through which we can observe and quantify aspects of computational difficulty previously accessible only through theoretical analysis.

In the words of Box: "All models are wrong, but some are useful." RIC is useful—for hardness prediction, solver design, and understanding the multi-dimensional nature of SAT complexity. That utility, grounded in orthogonality and validated empirically, is its contribution.

The path from structural measures to proof-theoretic proxies to formal complexity measures remains long. RIC represents one step on that path—a foundation for future work bridging theory and practice in computational hardness.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback. This work used computational resources from [institution/cloud provider if applicable]. Source code and data are available at https://github.com/nizaramama/RIC with DOI: https://doi.org/10.5281/zenodo.17925892.

References

[1] Nizar Amama. S3r: A spectral-dynamic resolution-inspired complexity measure, 2024. Preprint. Available at https://github.com/nizaramama/S3R.

[2] Carlos Ansótegui, Jesús Giráldez-Cru, and Jordi Levy. The community structure of sat formulas. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 410–423. Springer, 2012.

[3] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern sat solvers. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[4] Gilles Audemard and Laurent Simon. On the glucose sat solver. *International Journal on Artificial Intelligence Tools*, 27(01):1840001, 2018.

[5] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM (JACM)*, 48(2):149–169, 2001.

[6] Hans L Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical computer science*, 209(1-2):1–45, 1998.

[7] Vibhav Gogate and Rina Dechter. A complete anytime algorithm for treewidth. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 201–208, 2004.

[8] Armin Haken. The intractability of resolution. *Theoretical computer science*, 39:297–308, 1985.

[9] Marijn JH Heule, Warren A Hunt Jr, and Nathan Wetzler. Drat-trim: Efficient checking and trimming using expressive clausal proofs. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 422–429. Springer, 2014.

[10] Frank Hutter, Lin Xu, Holger H Hoos, and Kevin Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111, 2014.

[11] Ming Li and Paul MB Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, 2008.

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

[13] Rémi Monasson and Riccardo Zecchina. Statistical mechanics of the random k-satisfiability model. *Physical Review E*, 56(2):1357, 1997.

[14] 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*, pages 530–535, 2001.

[15] Igor Pavlov. Lzma sdk (software development kit). https://www.7-zip.org/sdk.html, 2007.

[16] Marko Samer and Stefan Szeider. Algorithms for propositional model counting. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 484–489. Springer, 2007.

[17] Steven J Sloman. Compressibility and the structure of sat instances. In *Proceedings of SAT Workshop*, 2008.

[18] Ryan Williams, Carla P Gomes, and Bart Selman. Backdoors to typical case complexity. *IJCAI*, 3:1173–1178, 2003.

[19] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla: Portfolio-based algorithm selection for sat, 2008. Available at https://www.cs.ubc.ca/labs/beta/Projects/SATzilla/.

[20] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606, 2008.