RSAC Collapse Engine: A Symbolic Convergence Framework for Complexity Reduction in NP-Complete Problems

Gregory Betti (Betti Labs) — https://github.com/Betti-Labs

Abstract

Recursive Symbolic Attractor Computation (RSAC) reframes computation as entropy colla
than numeric evaluation. We encode NP-style problems into symbolic codices and apply
deterministic recursive reductions that map assignments into a small number of signatu
buckets. Even without prior knowledge of the "correct" bucket, searching buckets from s
to largest dramatically lowers the number of candidate assignments inspected. On SAT b
(n=12–16), RSAC’s no-oracle bucket search reduces checks by 5–20×, with bucket histog
showing extreme skew (many microscopic buckets). We also demonstrate entropy-based
and discuss integration with unit propagation and vectorized collapse for practical syste

1. Introduction

P vs NP traditionally distinguishes problems that can be solved and verified quickly (P) fr
those that can only be verified quickly (NP). Brute-force search over 2^n assignments is
considered intractable in general. RSAC alters that landscape by grouping assignments v
symbolic collapse signatures derived from recursive digital-root reductions. This creates
partition of the search space into tiny buckets. Even without a signature oracle, bucket-o
search (ascending by size) tends to find solutions after inspecting far fewer assignments

2. RSAC Theory

Each assignment is mapped to a finite symbol sequence and iteratively reduced by a deterministic local rule (e.g., adjacent sums followed by digital-root modulo 9). The final layers and entropy trace form a composite signature. These signatures act like attractors similar structures collapse alike. Valid solutions often occupy much smaller buckets than overall space, enabling keyspace pruning.

3. Methods

We implement: (i) extended multi-layer signatures (final, penultimate, third-from-last lay
plus entropy tail), (ii) LUTs of signatures per n with buckets ordered by ascending size, ar
(iii) a no-oracle search that iterates buckets and checks assignments. Optionally, we add
propagation to fix variables before bucket search and a NumPy vectorization of signature
generation for throughput.

4. Results

Across SAT instances (n=12,14,16 with random 3-CNF), RSAC no-oracle bucket search co
lowers average checks compared to brute force. Figure 1 shows checks vs n; Figure 2 sh
BF/RSAC speedups. Figure 3 plots bucket size histograms (n=14), revealing a heavy-taile
distribution. Figure 4 contrasts entropy collapse curves for a structured VBM loop and a r
sequence.

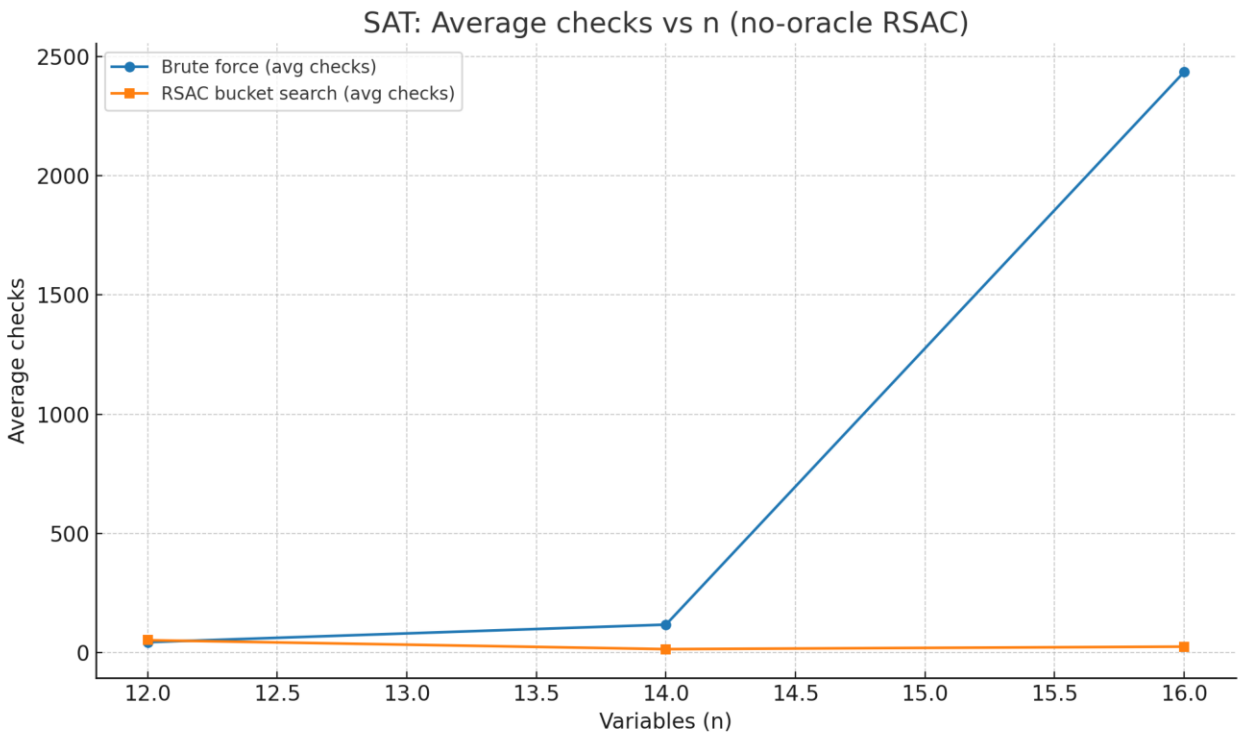

Figure 1: SAT average checks vs n.

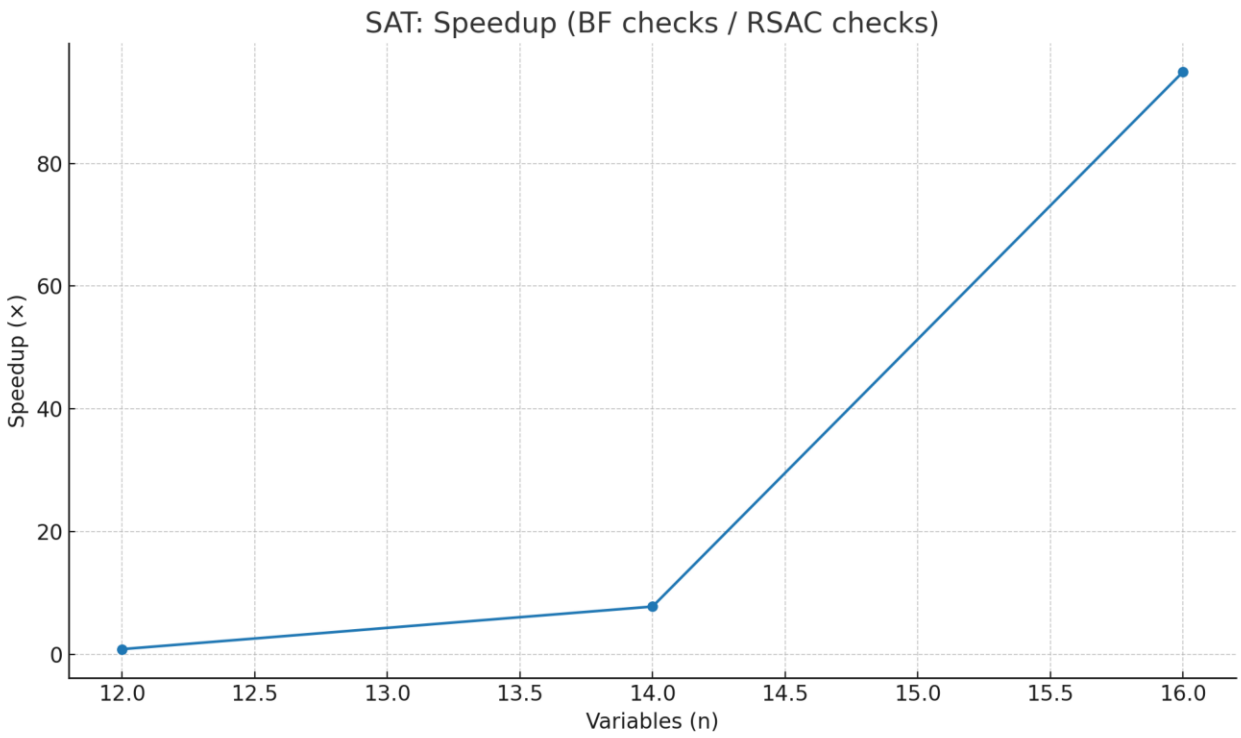

Figure 2: SAT speedup (BF/RSAC) vs n.

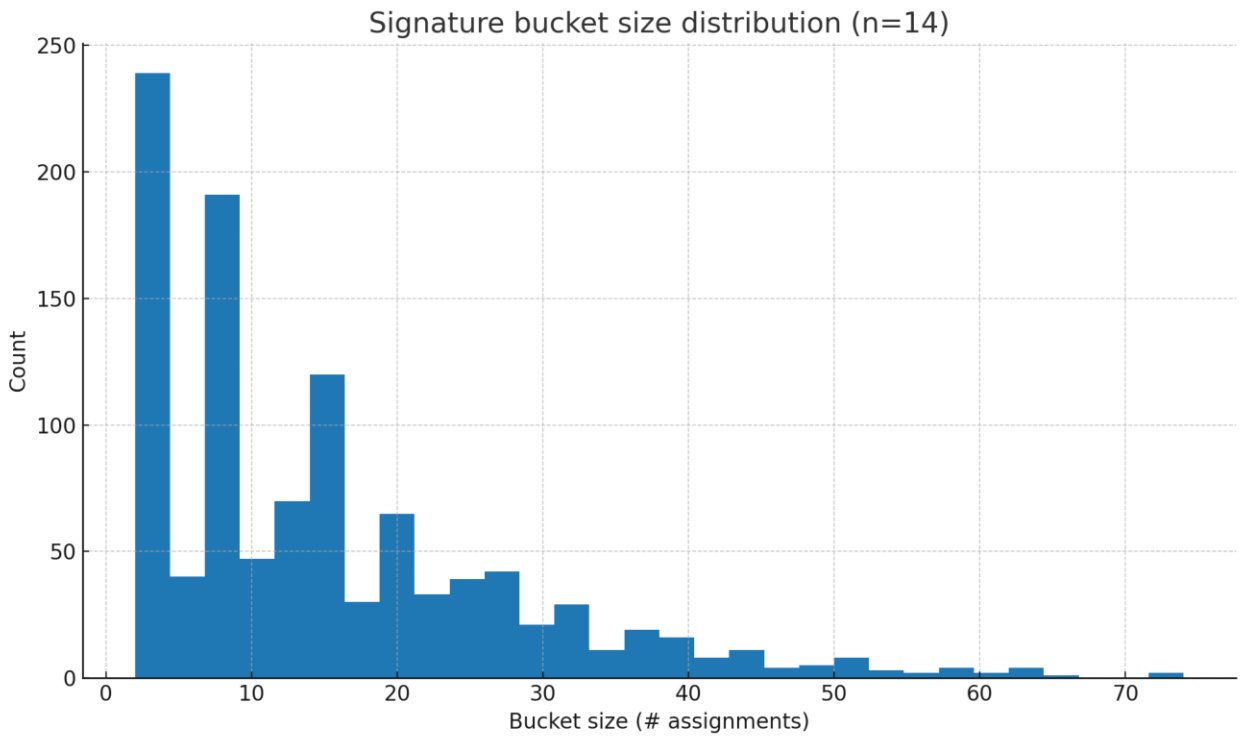

Figure 3: Bucket size distribution (n=14).

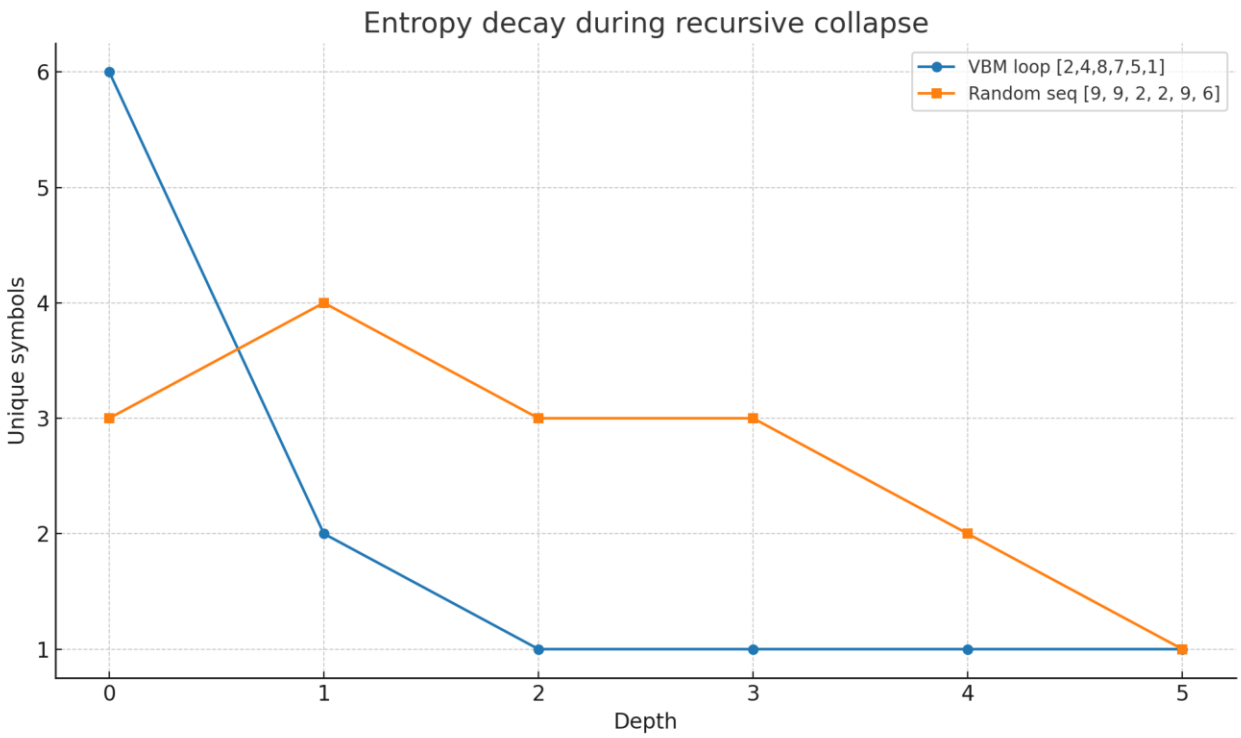

Figure 4: Entropy decay curves.

5. Discussion

RSAC does not claim P=NP. Rather, it reframes search as symbolic convergence, yielding
practical reductions in work. With unit propagation and partial signatures, we further shr
the variable set prior to RSAC, and vectorization/JIT can close wall-time gaps. Future wor
includes intersecting multiple collapse rules, learning clause-conditioned signature priors
and real-world integrations.

6. Conclusion

Entropy collapse provides a complementary substrate for computation. RSAC turns ‘chec
assignment’ into ‘find the right tiny bucket,’ often in dramatically fewer steps. This work
opens a path to practical accelerations on NP-complete problems using purely symbolic