


Reencoding Unique Literal Clauses, Abstract

Aeacus Sheng 

Department of Philosophy, Carnegie Mellon University

Joseph E. Reeves 

Computer Science Department, Carnegie Mellon University

Marijn J. H. Heule 

Computer Science Department, Carnegie Mellon University

1 Introduction

When encoding combinatorial problems into propositional logic, one commonly must encode selecting an object from a set. The *direct encoding* of EXACTLYONE (ℓ_1, \dots, ℓ_k) splits the constraint into ATLEASTONE (ℓ_1, \dots, ℓ_k) and ATMOSTONE (ℓ_1, \dots, ℓ_k) , where the former is just a clause and the latter blocks any pair of these literals from both being true.:

$$(\ell_1 \vee \dots \vee \ell_k) \wedge \bigwedge_{1 \leq i < j \leq k} (\bar{\ell}_i \vee \bar{\ell}_j)$$

On the other hand, the *sequential counter encoding* [3] introduce variables that help solvers to reason about multiple objects at the same time.

► **Example 1.** Consider the constraint EXACTLYONE (ℓ_1, ℓ_2, ℓ_3) . The sequential counter encoding uses the following clauses:

$$\underbrace{(s_1 \vee \bar{\ell}_1) \wedge (\bar{s}_1 \vee \ell_1)}_{s_1 \leftrightarrow \ell_1} \wedge \underbrace{(\bar{s}_2 \vee s_1 \vee \ell_2) \wedge (s_2 \vee \bar{s}_1) \wedge (s_2 \vee \bar{\ell}_2)}_{s_i \leftrightarrow (s_{i-1} \vee \ell_i)} \wedge \underbrace{(\bar{s}_1 \vee \bar{\ell}_2) \wedge (\bar{s}_2 \vee \bar{\ell}_3)}_{\bar{s}_{i-1} \vee \bar{\ell}_i} \wedge \underbrace{(s_2 \vee \ell_3)}_{\bar{s}_{k-1} \rightarrow \ell_k}$$

This paper presents a lightweight reencoding method that adds such variables while replacing some clauses to improve solver performance across a variety of benchmarks.

2 Reencoding Method

For a formula F , we say a clause C is a *unique literal clause* (ULC) if no literal in C appears outside of C in F . Intuitively, all literals in a ULC are unique to that clause. An important property is that if F is satisfiable, then F has a satisfying assignment where *exactly one* literal in each ULC C is true. In other words, any satisfying assignment can be adjusted (if necessary) so that no ULC has two or more true literals, since extra true literals in C can be flipped to false without affecting other clauses. So, every ULC represents an implicit exactly-one constraint. We reencode them using the sequential counter encoding. In place of a ULC C with k literals, we introduce $k - 1$ new auxiliary “order” variables and add clauses linking these variables with the original k literals to ensure at most one of the originals can be true (while preserving the at-least-one requirement). The added variables and clauses provide additional structure that could help guide the solver’s CDCL search. We also observed that the effectiveness of reencoding depends on the ordering of literals in the sequential counter. In practice, leaving the clause literals in arbitrary order (or shuffling them) can hurt performance. On the other hand, a nice ordering of literals in ULCs could allow the solver to learn short, useful clauses. We therefore apply a heuristic to sort the literals of each ULC according to a structural criterion before encoding, called *aligning* ULCs. This heuristic not only improves performance, but also helps predicting whether reencoding a formula will be beneficial. We implemented the above reencoding as a preprocessing step in CaDiCaL with support for solution reconstruction and proof logging.



© Aeacus Sheng, Joseph Reeves, and Marijn Heule;
licensed under Creative Commons License CC-BY 4.0

Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

3 Experimental Results

Within the 5,355 formulas in Anniversary track of the SAT Competition 2022 [1], our method reencoded 1,014 ($\sim 19\%$) of them, and led to substantial speedups when solving the 459 ($\sim 9\%$) alignable formulas.

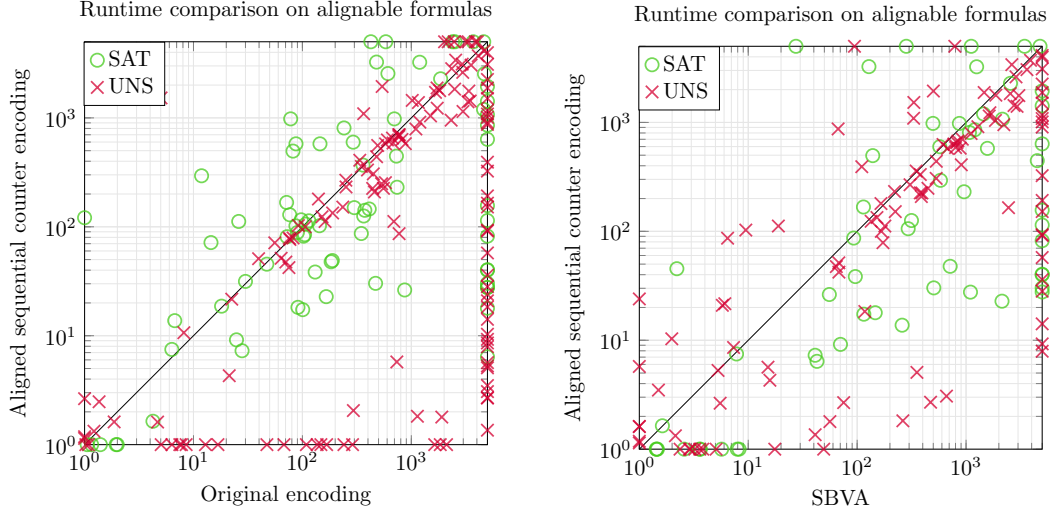


Figure 1 Sequential counter encoding on alignable instances: Aligned vs. original (left) and Aligned vs. SBVA (right).

The left plot of Figure 1 compares solver runtimes on alignable formulas, contrasting our method’s aligned sequential counter encoding against the original encoding (baseline). For alignable formulas, we observe a clear performance improvement with the aligned reencoding, particularly on UNSAT instances. Most UNSAT points lie well below the diagonal, often on the axes, indicating that adopting our method can solve more UNSAT formulas using much less time. The SAT instances exhibit mixed results, but still, our method solves more SAT formulas and has an overall advantage. We have verified all the proofs and satisfying assignments produced by our method on the original formulas in this plot to make sure these speedups are not due to conceptual or implementation errors.

We also tested if the state-of-the-art reencoding technique SBVA (structured bounded variable addition) [2] would reencode these formulas, and found that it does affect most of them. So, we compared our reencoding method with SBVA as preprocessor for CaDiCaL. The results are shown in Figure 1. Our sequential counter encoding with alignment can solve dozens of alignable formulas that cannot be solved when using SBVA, suggesting that these approaches are orthogonal, and future work could attempt to combine them.

4 Conclusion

We presented a new reencoding technique for SAT formulas. While all existing work on reencoding focuses on reducing the size of formulas, our technique increases the number of variables and may also increase the number of clauses. The experimental results show that our method increases the number of solved formulas from SAT Competitions, providing concrete evidence that our reencoding method can greatly improve solver performance and even outperform state-of-the-art reencoding techniques on the class of formulas we target.

References

- 1 Tomas Balyo, Marijn J.H. Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors. *Proceedings of SAT Competition 2022: Solver and Benchmark Descriptions*. Department of Computer Science Series of Publications B. Department of Computer Science, University of Helsinki, Finland, 2022. Accessed: 2025-03-13. URL: <http://hdl.handle.net/10138/359079>.
- 2 Andrew Haberlandt, Harrison Green, and Marijn J. H. Heule. Effective auxiliary variables via structured reencoding. In *Theory and Applications of Satisfiability Testing (SAT)*, volume 271, pages 11:1–11:19, Dagstuhl, Germany, 2023. Schloss Dagstuhl.
- 3 Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005*, pages 827–831, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.