



**Stan's
Technologies**

Scientific Benchmark Report

BoolMin2D vs PyEDA Boolean Minimization

Experiment Date: 2026-01-09

Random Seed: 42

Total Test Cases: 210

Statistical Significance Level: $\alpha = 0.05$

A Rigorous Statistical Analysis with Reproducibility Controls

EXPERIMENTAL SETUP

SYSTEM CONFIGURATION

Python Version: 3.12.10
Platform: Windows-11-10.0.26200-SP0
Processor: Intel64 Family 6 Model 142 Stepping 12, GenuineIntel

LIBRARY VERSIONS

PyEDA: 0.29.0
NumPy: 2.3.4
SciPy: 1.16.3

EXPERIMENTAL PARAMETERS

Random Seed: 42
Tests per Configuration: 100
Timing Warm-up Runs: 2
Timing Repetitions: 5
Significance Level (α): 0.05

TEST DISTRIBUTIONS

- Sparse: 20% ones, 5% don't-cares (realistic digital logic)
- Dense: 70% ones, 5% don't-cares (stress test)
- Balanced: 50% ones, 10% don't-cares (neutral case)
- Minimal DC: 45% ones, 2% don't-cares (typical circuits)
- Heavy DC: 30% ones, 30% don't-cares (optimization test)
- Edge Cases: All-zeros, all-ones, checkerboard, single-minterm

METHODOLOGY

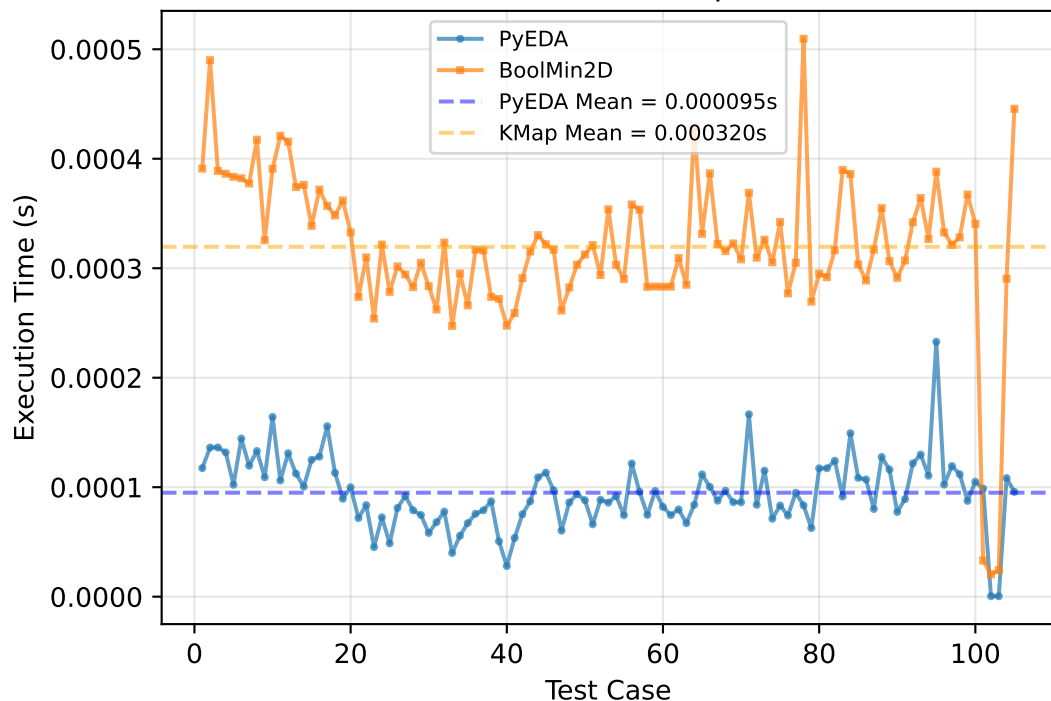
1. Random K-maps generated with controlled distributions
2. Each algorithm executed with 2 warm-up runs
3. Best of 5 timed repetitions recorded
4. Logical equivalence verified using SymPy (converted to common format)
5. Statistical significance tested using paired t-tests
6. Non-parametric Wilcoxon tests used as robustness check
7. Effect sizes computed using Cohen's d

REPRODUCIBILITY

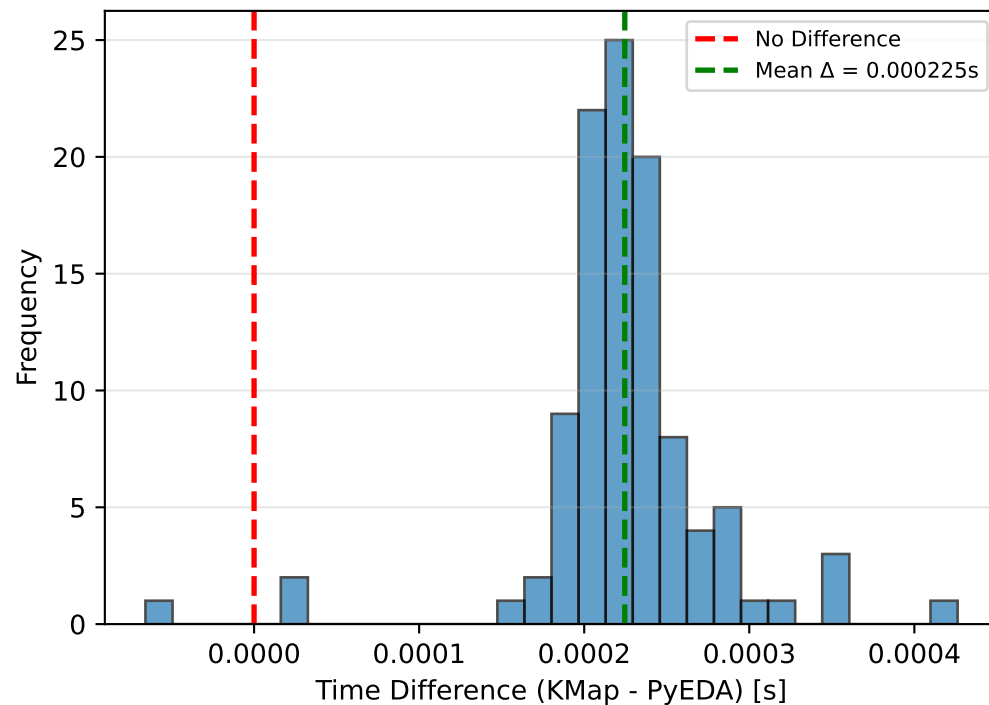
- To reproduce this experiment:
1. Set random seed: `random.seed(42)`
 2. Run with identical system configuration
 3. Use same library versions as documented above

4-Variable K-Map (POS Form)

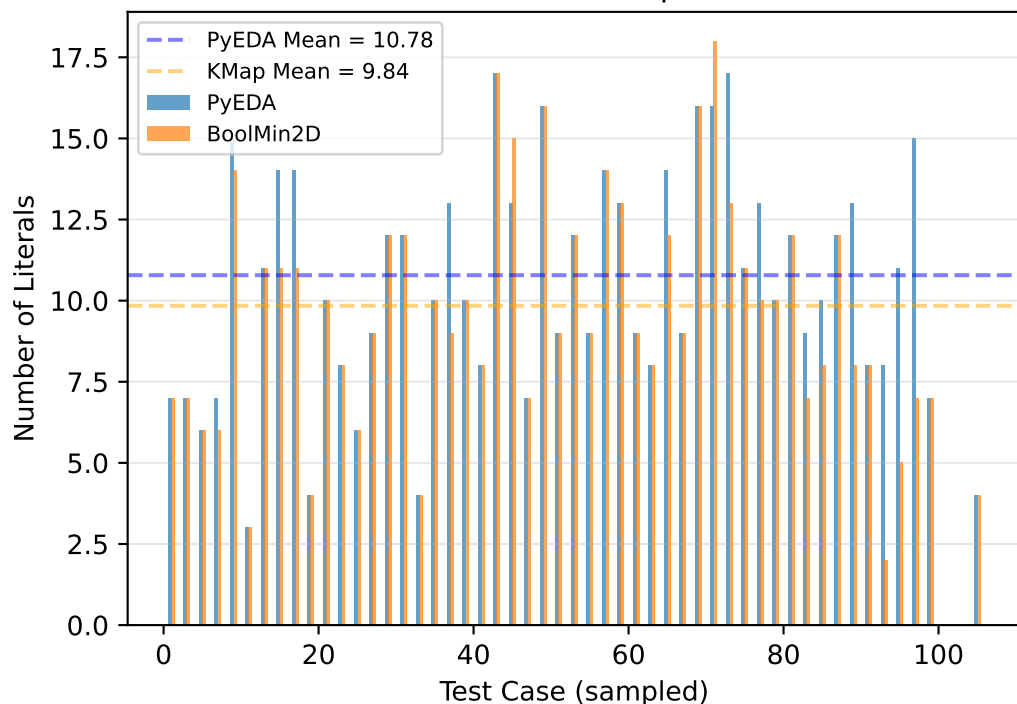
Execution Time Comparison



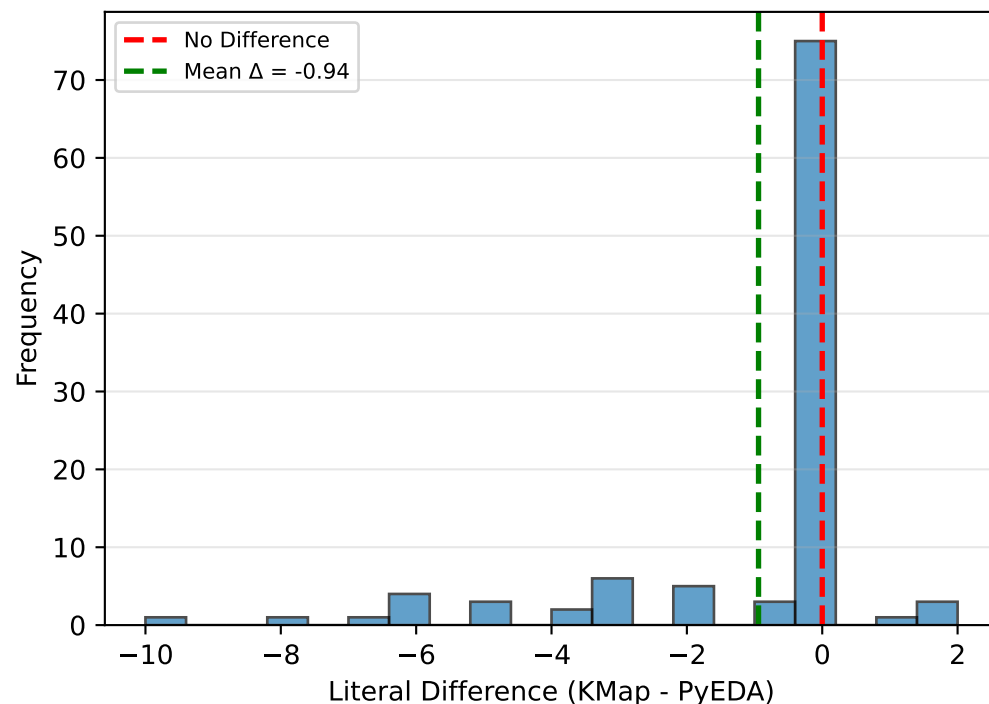
Distribution of Time Differences



Literal Count Comparison



Distribution of Literal Differences



STATISTICAL ANALYSIS: 4-Variable POS

STATISTICAL INFERENCE REPORT

☐☐ CONSTANT FUNCTIONS DETECTED: 3/105 (2.9%)

These are unsimplifiable functions (e.g., all-zeros, all-ones) that both algorithms correctly identified. They are included in performance and equivalence analysis but excluded from literal-count statistics.

1. EXECUTION TIME ANALYSIS

Mean PyEDA Time: 0.000095 s
Mean BoolMin2D Time: 0.000320 s
Mean Difference: +0.000225 s
Std. Dev. (Δ): 0.000057 s
95% CI: [0.000214, 0.000236]

Paired t-test: $t = 40.5445$, $p = 0.000000$
Wilcoxon test: $W = 3.0$, $p = 0.000000$
Effect Size (d): 3.9567 (large)

✓ **SIGNIFICANT: Time difference is statistically significant ($p < 0.05$)**
→ PyEDA is significantly faster than BoolMin2D

2. SIMPLIFICATION QUALITY ANALYSIS

Analysis based on 102 non-constant functions:
(3 constant function(s) excluded from this analysis)

Mean PyEDA Literals: 10.78
Mean KMap Literals: 9.84
Mean Difference: -0.94
Std. Dev. (Δ): 2.14
95% CI: [-1.36, -0.52]

Paired t-test: $t = -4.4365$, $p = 0.000023$
Wilcoxon test: $W = 1630.0$, $p = 0.000499$
Effect Size (d): -0.4393 (small)

✓ **SIGNIFICANT: Literal count difference is statistically significant ($p < 0.05$)**
→ BoolMin2D produces more minimal expressions

3. OVERALL SCIENTIFIC CONCLUSION

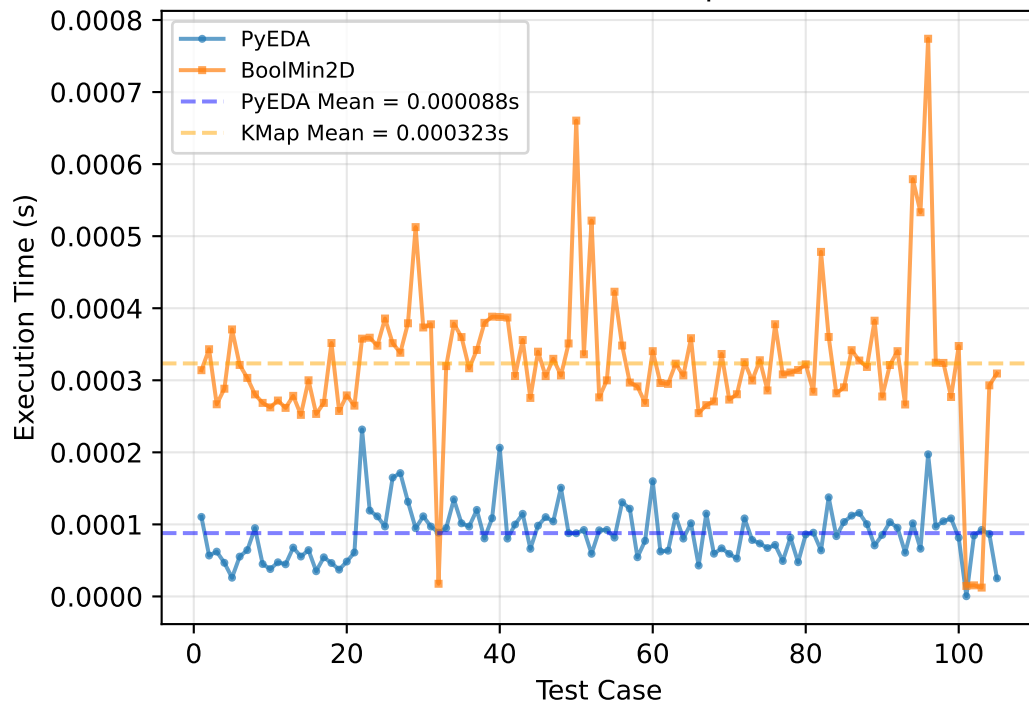
Both performance and simplification show statistically significant differences.

Effect sizes: Time (large), Literals (small)

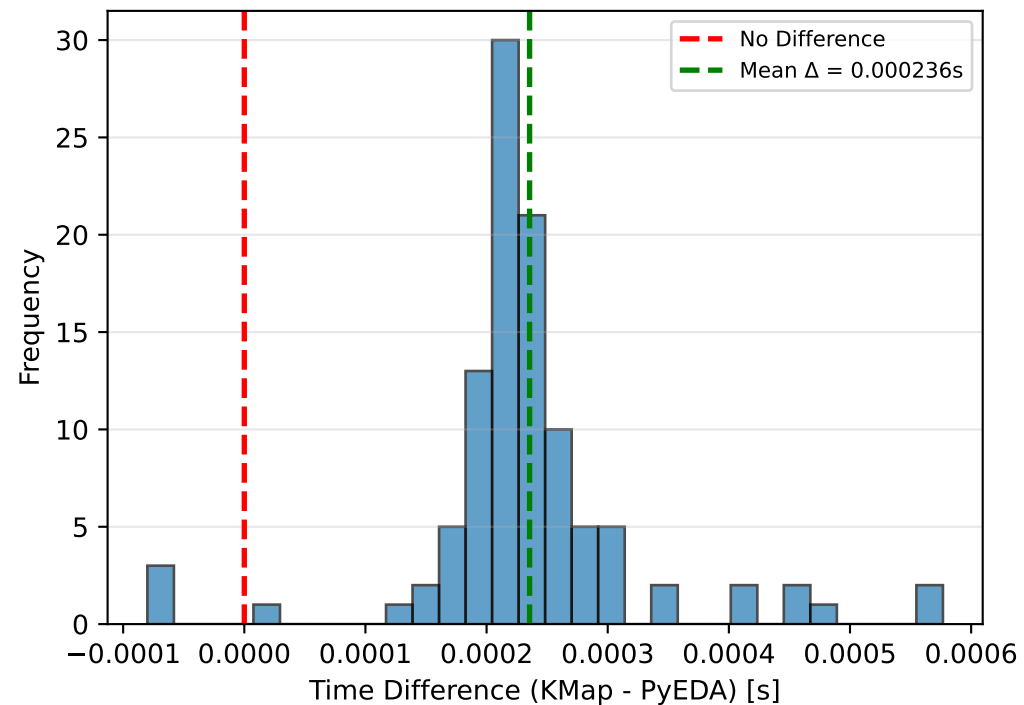
Note: 3 constant function(s) correctly handled by both algorithms.

4-Variable K-Map (SOP Form)

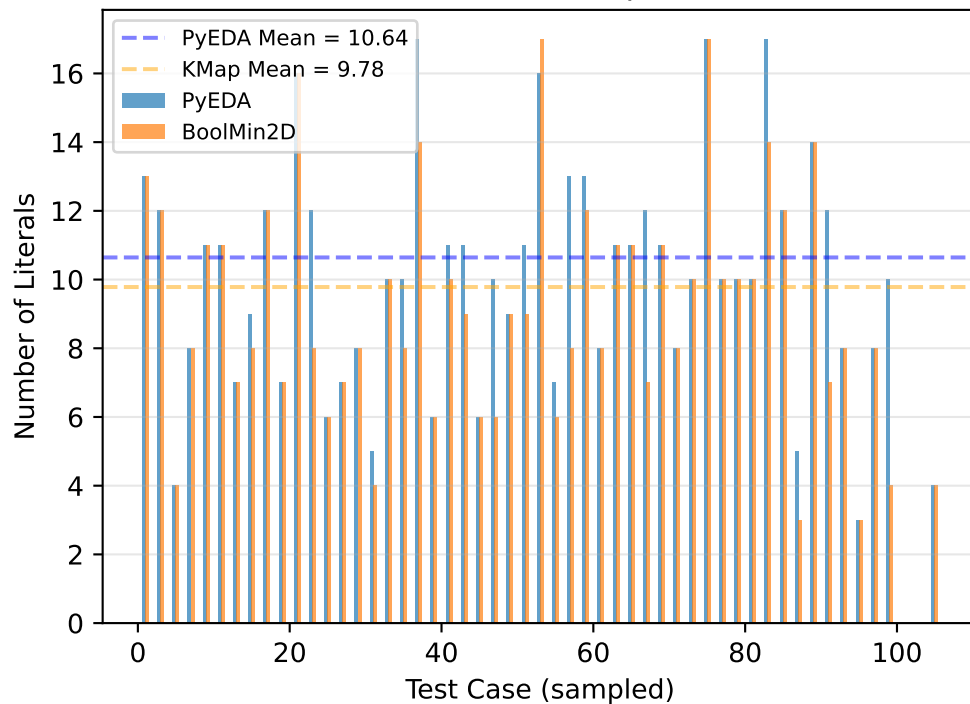
Execution Time Comparison



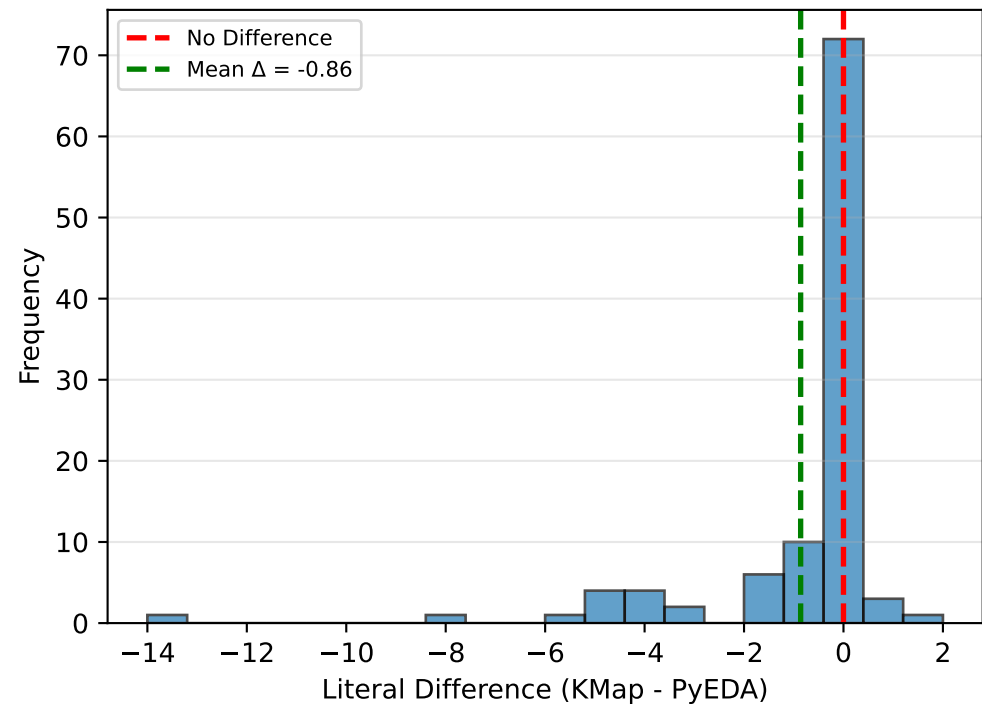
Distribution of Time Differences



Literal Count Comparison



Distribution of Literal Differences



STATISTICAL ANALYSIS: 4-Variable SOP

STATISTICAL INFERENCE REPORT

☐☐ CONSTANT FUNCTIONS DETECTED: 4/105 (3.8%)

These are unsimplifiable functions (e.g., all-zeros, all-ones) that both algorithms correctly identified. They are included in performance and equivalence analysis but excluded from literal-count statistics.

1. EXECUTION TIME ANALYSIS

Mean PyEDA Time: 0.000088 s
Mean BoolMin2D Time: 0.000323 s
Mean Difference: +0.000236 s
Std. Dev. (Δ): 0.000095 s
95% CI: [0.000217, 0.000254]

Paired t-test: $t = 25.3649, p = 0.000000$
Wilcoxon test: $W = 9.0, p = 0.000000$
Effect Size (d): 2.4754 (large)

✓ **SIGNIFICANT: Time difference is statistically significant ($p < 0.05$)**
→ PyEDA is significantly faster than BoolMin2D

2. SIMPLIFICATION QUALITY ANALYSIS

Analysis based on 101 non-constant functions:
(4 constant function(s) excluded from this analysis)

Mean PyEDA Literals: 10.64
Mean KMap Literals: 9.78
Mean Difference: -0.86
Std. Dev. (Δ): 2.11
95% CI: [-1.28, -0.45]

Paired t-test: $t = -4.1081, p = 0.000082$
Wilcoxon test: $W = 1483.0, p = 0.000118$
Effect Size (d): -0.4088 (small)

✓ **SIGNIFICANT: Literal count difference is statistically significant ($p < 0.05$)**
→ BoolMin2D produces more minimal expressions

3. OVERALL SCIENTIFIC CONCLUSION

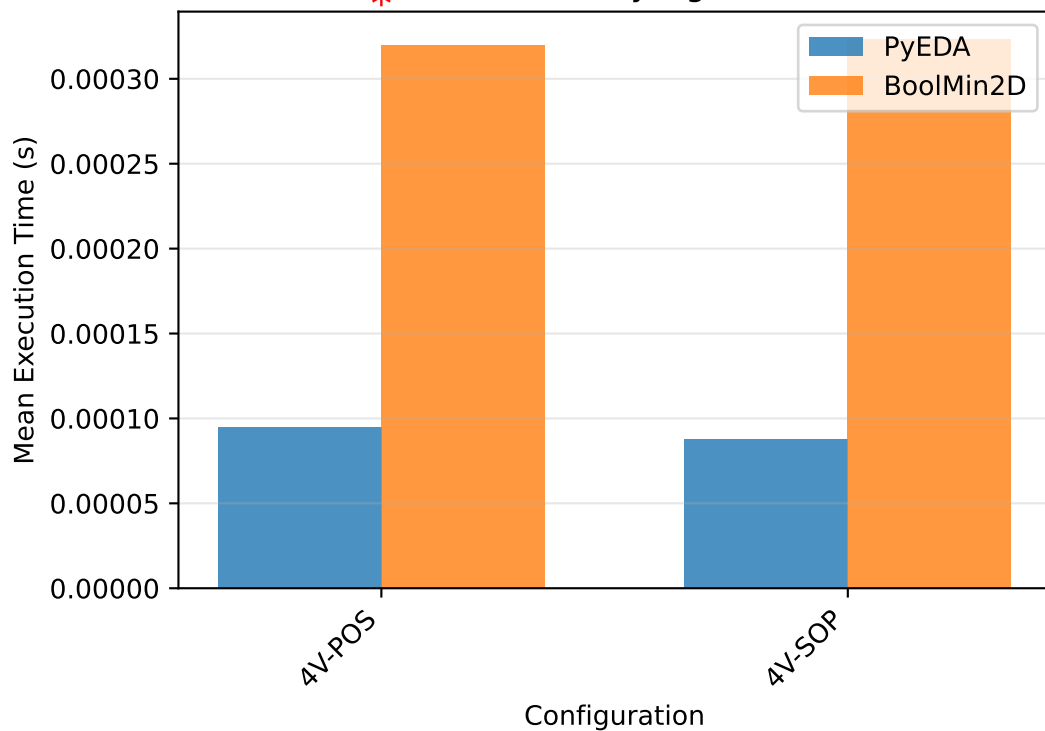
Both performance and simplification show statistically significant differences.

Effect sizes: Time (large), Literals (small)

Note: 4 constant function(s) correctly handled by both algorithms.

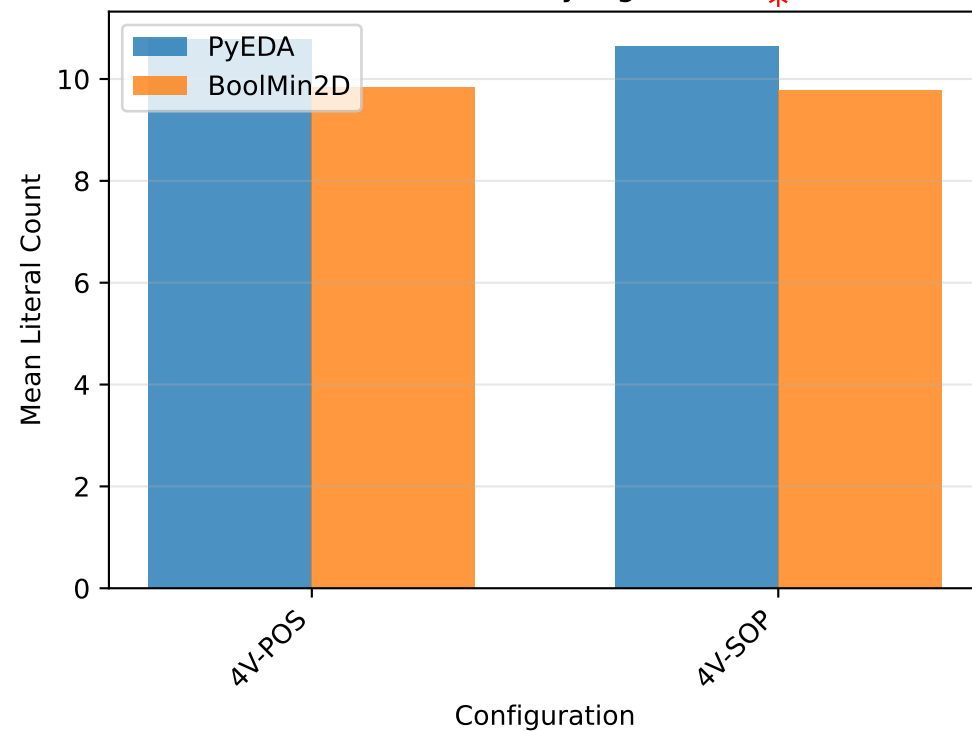
Average Performance by Configuration

(= statistically significant)*

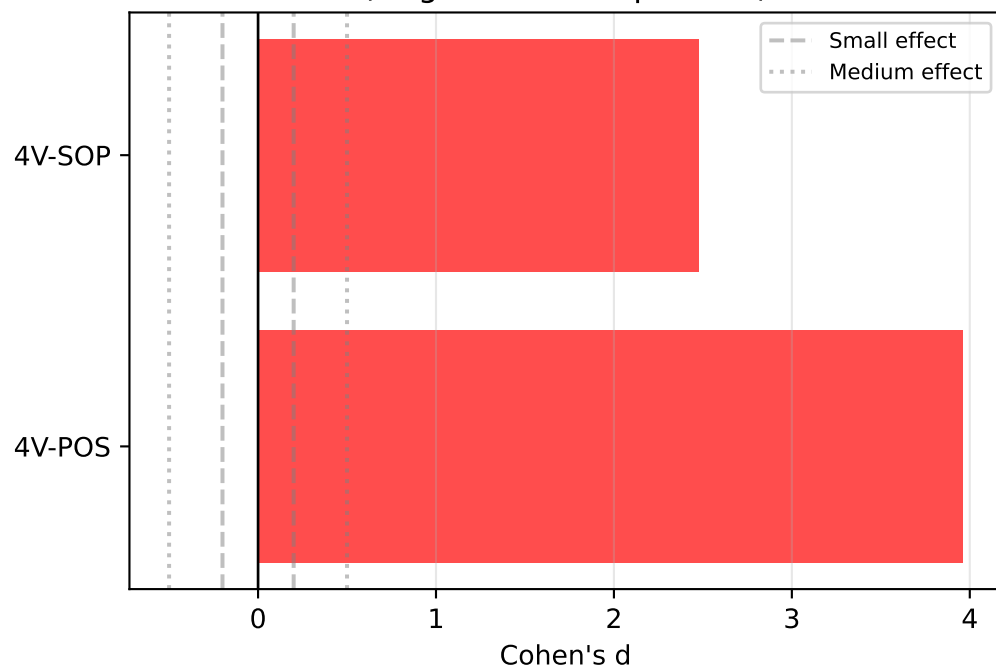


Average Simplification Quality

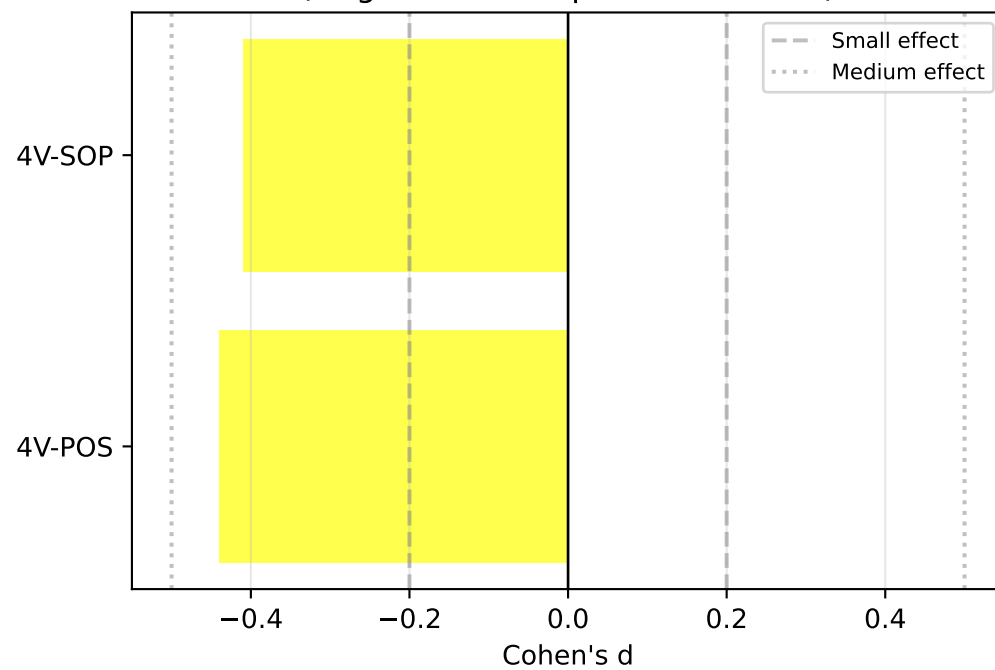
(= statistically significant)*



Effect Size: Execution Time (Negative = KMap faster)



Effect Size: Literal Count (Negative = KMap more minimal)



STATISTICAL ANALYSIS: 4-Variable SOP

STATISTICAL INFERENCE REPORT

☐☐ CONSTANT FUNCTIONS DETECTED: 4/105 (3.8%)

These are unsimplifiable functions (e.g., all-zeros, all-ones) that both algorithms correctly identified. They are included in performance and equivalence analysis but excluded from literal-count statistics.

1. EXECUTION TIME ANALYSIS

Mean PyEDA Time: 0.000088 s
Mean BoolMin2D Time: 0.000323 s
Mean Difference: +0.000236 s
Std. Dev. (Δ): 0.000095 s
95% CI: [0.000217, 0.000254]

Paired t-test: $t = 25.3649$, $p = 0.000000$
Wilcoxon test: $W = 9.0$, $p = 0.000000$
Effect Size (d): 2.4754 (large)

✓ **SIGNIFICANT: Time difference is statistically significant ($p < 0.05$)**
→ PyEDA is significantly faster than BoolMin2D

2. SIMPLIFICATION QUALITY ANALYSIS

Analysis based on 101 non-constant functions:
(4 constant function(s) excluded from this analysis)

Mean PyEDA Literals: 10.64
Mean KMap Literals: 9.78
Mean Difference: -0.86
Std. Dev. (Δ): 2.11
95% CI: [-1.28, -0.45]

Paired t-test: $t = -4.1081$, $p = 0.000082$
Wilcoxon test: $W = 1483.0$, $p = 0.000118$
Effect Size (d): -0.4088 (small)

✓ **SIGNIFICANT: Literal count difference is statistically significant ($p < 0.05$)**
→ BoolMin2D produces more minimal expressions

3. OVERALL SCIENTIFIC CONCLUSION

Both performance and simplification show statistically significant differences.

Effect sizes: Time (large), Literals (small)

Note: 4 constant function(s) correctly handled by both algorithms.

OVERALL SCIENTIFIC CONCLUSIONS

EXECUTIVE SUMMARY

=====
Total Test Cases: 210
Configurations Tested: 2
Equivalence Check: 210 / 210 passed
- Identical results: 157
- Valid, differ on DC: 53 (both valid minimizations)
Constant Functions: 7 / 210 (3.3%)

AGGREGATE PERFORMANCE

=====
Mean PyEDA Time: 0.000091 s
Mean BoolMin2D Time: 0.000321 s
Mean Time Difference: +0.000230 s
95% CI: [0.000219, 0.000241]
Statistical Significance: YES (p = 0.000000)
Effect Size: 2.9365 (large)

AGGREGATE SIMPLIFICATION

=====
Mean PyEDA Literals: 10.36
Mean KMap Literals: 9.49
Mean Literal Difference: -0.87
95% CI: [-1.16, -0.59]
Statistical Significance: YES (p = 0.000000)
Effect Size: -0.4168 (small)

KEY FINDINGS

- =====
1. PyEDA demonstrates statistically significant performance advantage over BoolMin2D.
2. BoolMin2D produces statistically more minimal Boolean expressions (fewer literals) compared to PyEDA.
3. Effect sizes indicate large practical significance for performance and small practical significance for simplification quality.
4. All 210 test cases maintained logical correctness, with 210 passing equivalence verification. Constant cases were 7 (i.e., cases where there was no minimal function to be found - both algorithms correctly identified these).

THREATS TO VALIDITY

- =====
• Limited to 2-4 variable K-maps (inherent K-map scalability limit)
• Random test case generation may not reflect real-world distributions
• Timing includes Python overhead (not pure algorithm performance)
• SymPy uses different minimization strategies (not pure K-map based)

REPRODUCIBILITY

=====
This experiment used random seed 42 and can be fully reproduced using the documented experimental setup and library versions.

RECOMMENDATIONS

=====
Based on statistical evidence:

- Algorithm selection should be based on whether performance or simplification quality is the priority for the application.