By Esther Ng Xin Yue

FEATURE ATTRIBUTION FOR MACHINE LEARNING MODELS

EXACT SHAP COMPUTATION ON TRACTABLE BOOLEAN CIRCUITS

03 Feb 2026

WHAT IS SHAP?

SHAP (SHapley Additive exPlanations) is a feature attribution method for explaining individual model predictions.

Cooperative game theory

- **Players** → input features x
- **Game payout** → model prediction f(x)
- **Goal** → fair distribution of the payout amongst the players

$$f(x) = \Phi_0 + \sum_{i=1}^{d} \Phi_i$$

Why is SHAP so popular?

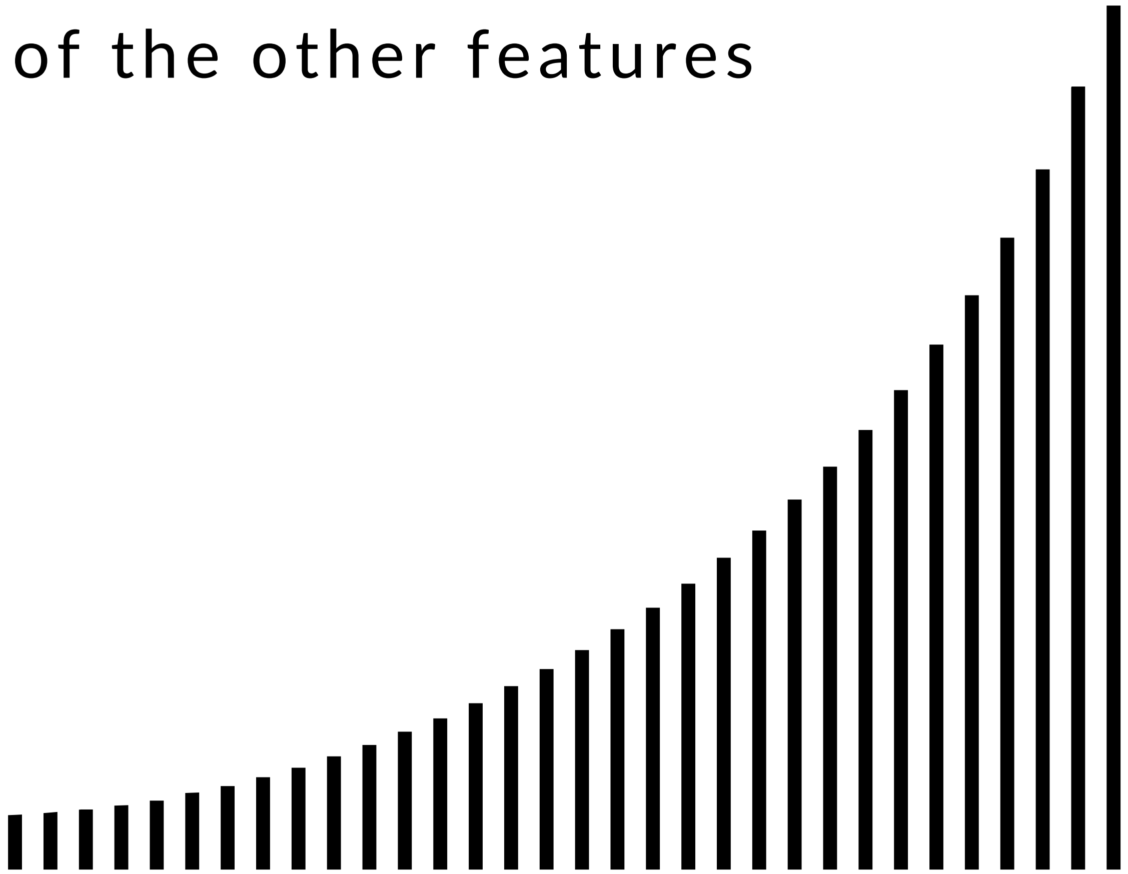
- Model-agnostic
- Strong theoretical guarantees
- widely used in model debugging, fairness audits, and compliance testing

WHY IS SHAP COMPUTATIONALLY HARD?

01 EXPONENTIAL NUMBER OF FEATURE COALITIONS

- SHAP is based on Shapley values
- For each feature, SHAP considers all subsets of the other features

Number of coalitions per feature: 2^{n-1}



WHY IS SHAP COMPUTATIONALLY HARD?

02 EACH COALITION REQUIRES A MODEL EVALUATION

For each subset S, SHAP evaluates a value function v(S):

- This fixes features in S
- Integrates out all remaining features
- Computes an expected model output

$$v(S) = E[f(X) | X_s = x_s]$$

WHY IS SHAP COMPUTATIONALLY HARD?

03 #P-HARDNESS OF EXACT SHAP

- Computing v(S) reduces to weighted model counting
- Weighted model counting is #P-hard
  - Exact SHAP is intractable in general

CAN WE RESTRICT THE MODEL CLASS SO THAT SHAP BECOMES TRACTABLE?

A problem is tractable if it can be solved efficiently,
even as the problem size grows.

YES! with a tractable circuit representation

D-DNNFS

SENTENTIAL DECISION DIAGRAMS (SDDS)

STRUCTURAL CONSTRAINTS

Decomposability

- Subcircuits depend on disjoint sets of variables
- Independent parts can be evaluated separately

Determinism

- At most one branch is true for any input
- Prevents double-counting of probability mass or explanations

Structured Decomposability

- All decisions follow a fixed variable structure
- Ensures global consistency across the circuit

These properties allow:

- Efficient model counting
- Efficient conditional expectations
- Efficient SHAP aggregation

Key Insight for SHAP

- SHAP requires many conditional expectations
- In general models: #P-hard
- In tractable circuits:
  - Each conditional expectation reduces to a single linear-time circuit traversal

WHAT ARE SDDS?

An SDD expresses a Boolean function as a disjunction of mutually exclusive cases:

$$(p_1 \cap s_1) \cup (p_2 \cap s_2) \cup \ldots$$

Key Structural Properties

Decomposability

- primes and subs depend on disjoint variable sets

Determinism

- no two cases can be true at the same time

Structured Decomposability

- all decisions follow a fixed variable tree (vtree)

- Instead of checking all combinations of features,
- an SDD organises them so we never double-count or overlap work.
- supports:
  - Linear-time model counting
  - Efficient conditioning
  - Exact probability computation
  - Exact SHAP computation

SDDs avoid redundant computation by structuring decisions

Computing SHAP Scores on SDDs

Structure → Tractable Inference → Tractable SHAP

SHAP Component	SDD Intepretation
FEATURE SUBSET S	PARTIAL ASSIGNMENT OF VARIABLES $X_s = x_s$
VALUE FUNCTION OF V(S)	EXPECTED OUTPUT UNDER CONDITIONING ON S $E\left[f\left(X\right)\left\|X_s = x_s\right.\right]$
MARGINAL CONTRIBUTION	DIFFERENCE OF TWO CONDITIONAL EXPECTATIONS $E\left[f\left(X\right)\left\|x_s = 1\right.\right] - E\left[f\left(X\right)\left\|x_s = 0\right.\right]$
SUMMATION OVER COALITION	AGGREGATION VIA DYNAMIC PROGRAMMING

SYSTEM ARCHITECTURE

BOOLEAN MODEL (CNF) → **SENTENTIAL DECISION DIAGRAM (SDD)** → **EXACT SHAP COMPUTATION** → **FEATURE ATTRIBUTION OUTPUT**

- Input: Boolean classifier expressed as a CNF formula
- Compiled into an SDD using PySDD

- Run a bottom-up dynamic program over the SDD
- For each feature:
  - Compute γ and δ arrays per gate
  - Aggregate contributions using Shapley weights
- No sampling, no coalition enumeration
- Exact SHAP via circuit traversal

- Output: SHAP value per feature

BENCHMARKING

BENCHMARK 1: Compilation vs SHAP runtime (means, ms)

Test Case	Compile(ms)	SHAP(ms)	Total(ms)
Small (2 vars, 4 clauses) – Symmetric	9.39	0.14	9.53
Medium (4 vars, 2 clauses) – Independent OR	6.92	2.18	9.11
Large (5 vars, 10 clauses) – Constraint SAT	9.49	3.33	12.86

BENCHMARK 2: Runtime vs circuit size (SDD nodes) [mean_shap]

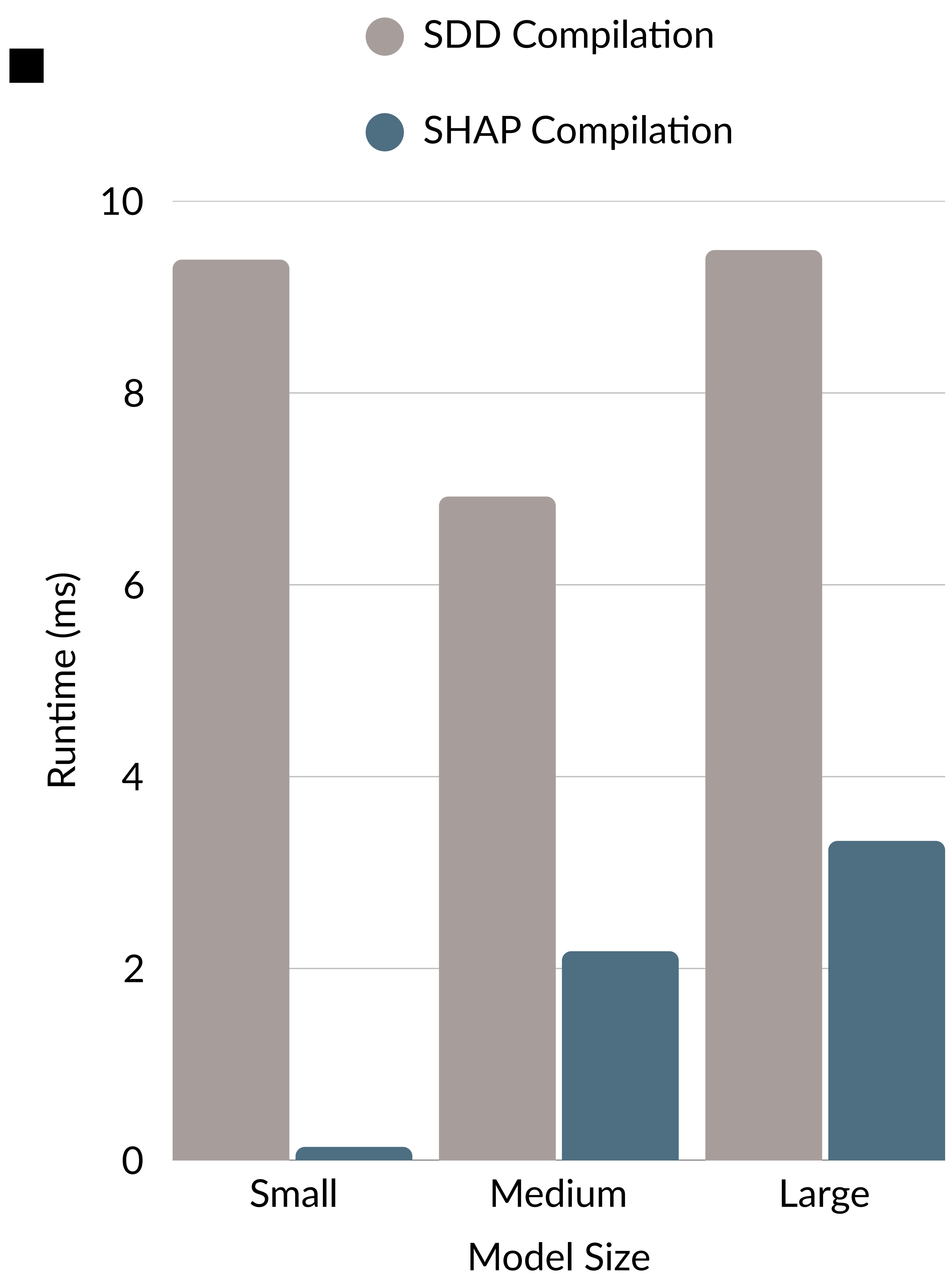
Test Case	SDD(nodes)	Runtime(ms)
Small (2 vars, 4 clauses) – Symmetric	5	0.14
Medium (4 vars, 2 clauses) – Independent OR	12	2.18
Large (5 vars, 10 clauses) – Constraint SAT	28	3.33

BENCHMARK 3: Compilation vs SHAP cost breakdown (mean % of total)

Test Case	Compile%	SHAP%	(rest)
Small (2 vars, 4 clauses) – Symmetric	98.5	1.5	0.1
Medium (4 vars, 2 clauses) – Independent OR	76.0	23.9	0.1
Large (5 vars, 10 clauses) – Constraint SAT	73.9	25.9	0.2

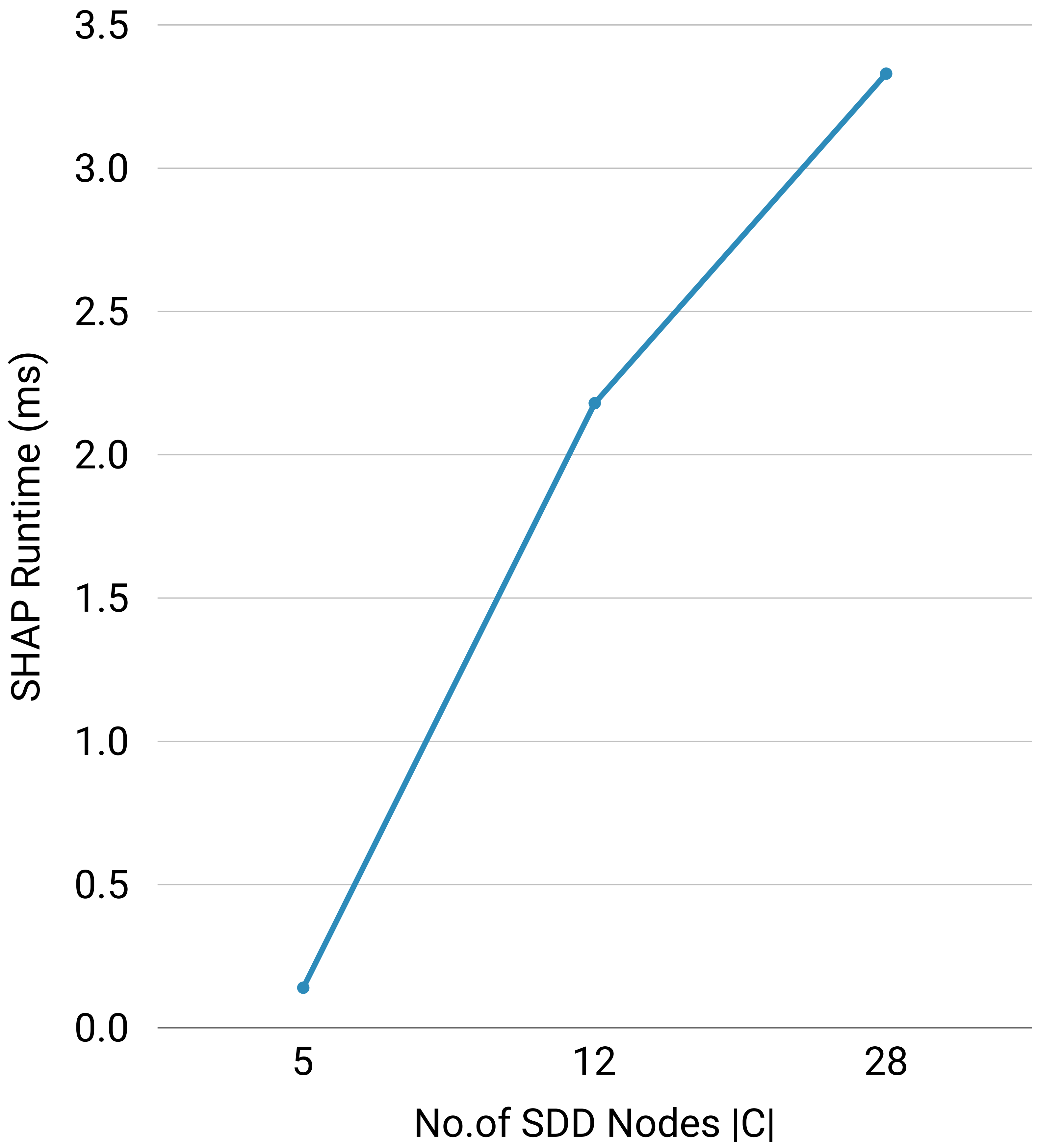
BENCHMARK 4: SHAP runtime vs number of variables (means)

Test Case	Vars	SHAP(ms)
Small (2 vars, 4 clauses) – Symmetric	2	0.14
Medium (4 vars, 2 clauses) – Independent OR	4	2.18
Large (5 vars, 10 clauses) – Constraint SAT	6	3.33



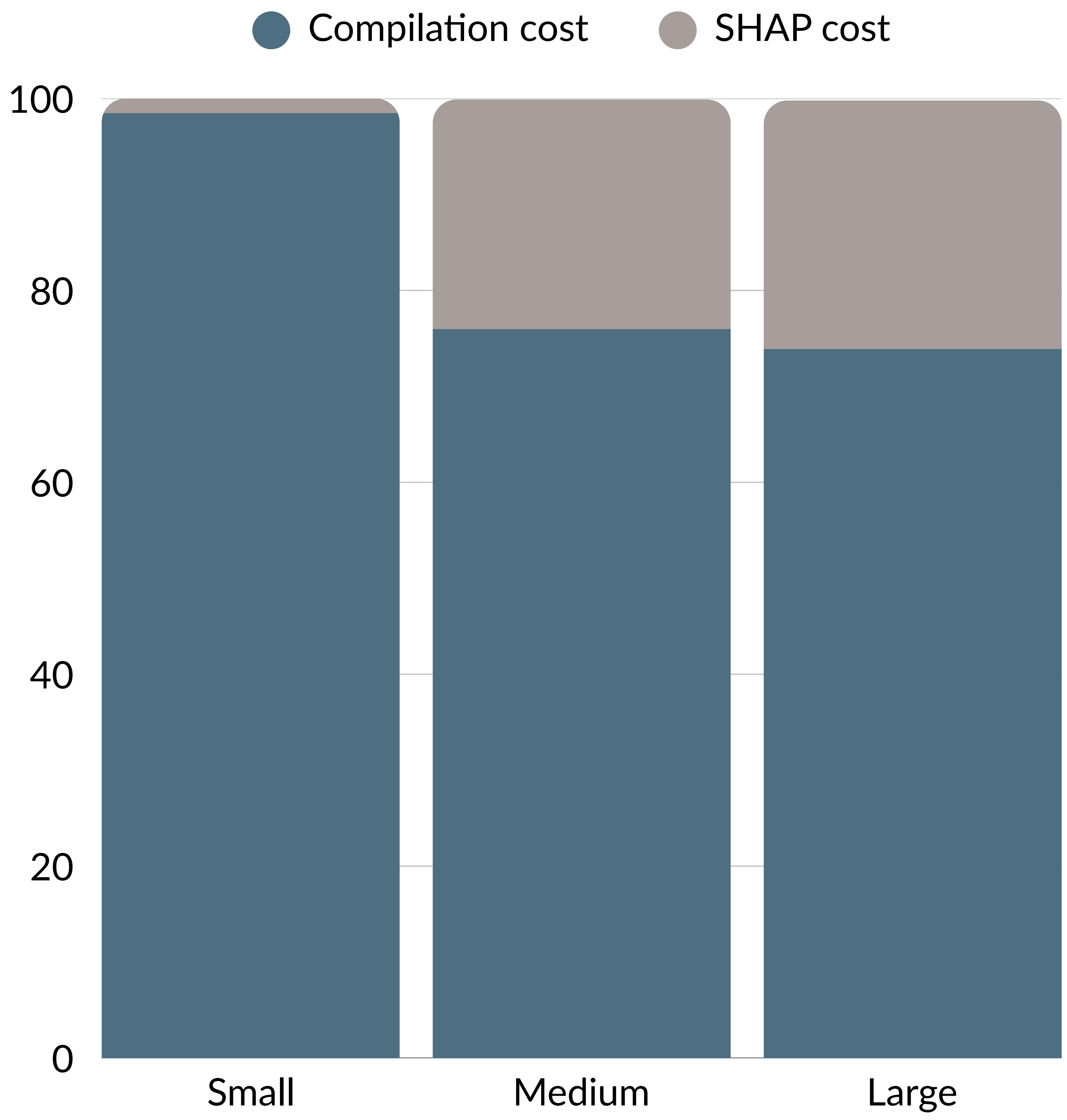
Compilation vs SHAP Runtime

- Compilation dominates total runtime
- SHAP computation is consistently small
- Once compiled, explanations are cheap



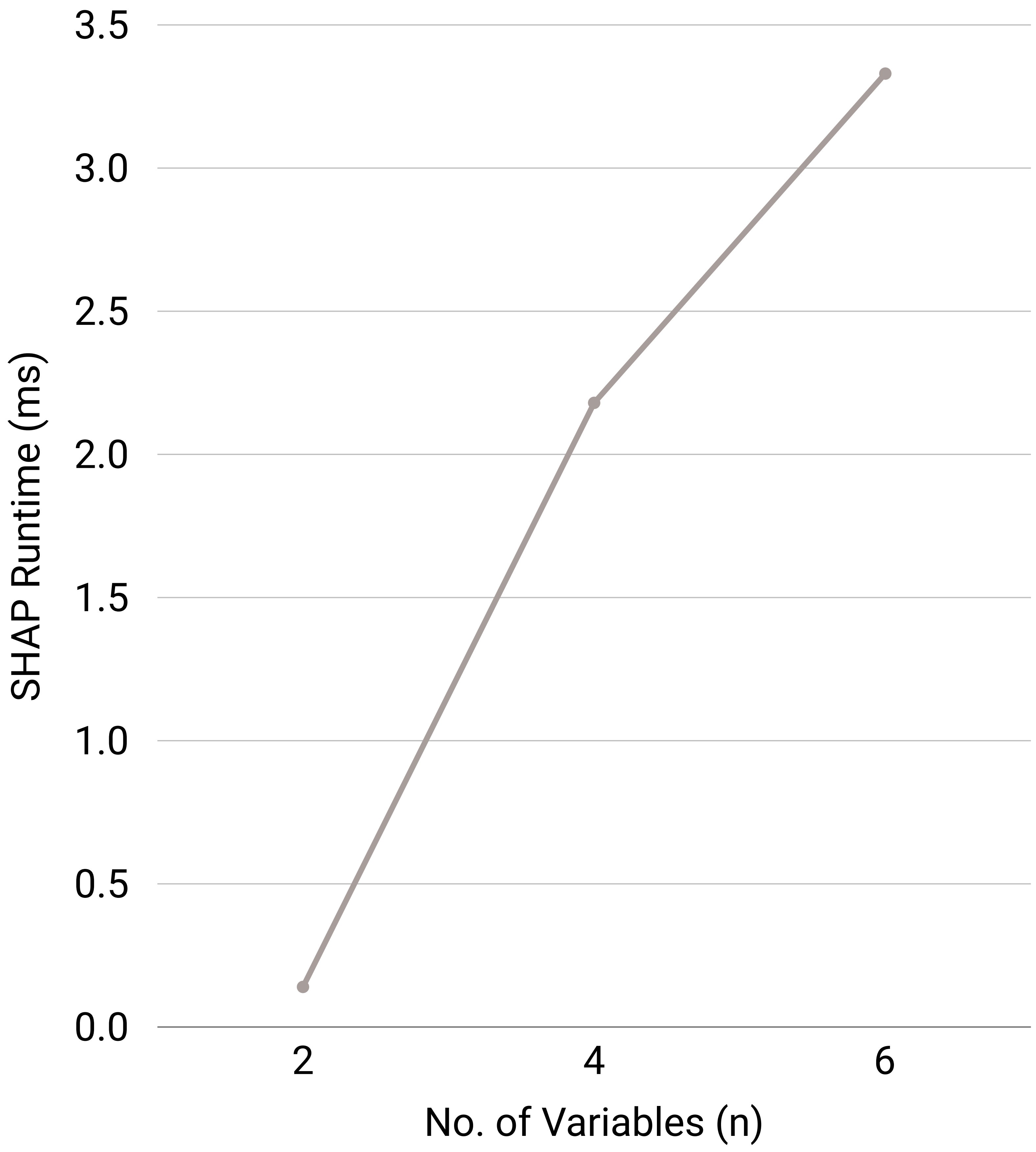
Runtime vs Circuit Size

- Smooth, near-linear scaling
- No exponential blow-up
- Matches theoretical complexity



Compilation vs SHAP Cost Breakdown

- SHAP is a minor fraction of total cost
- Confirms tractability in practice



SHAP Runtime vs Number of Variables

- Polynomial growth
- No combinatorial explosion
- Practical even as n increases

APPLICATIONS OF TRACTABLE SHAP

SHAP (SHapley Additive exPlanations) is a feature attribution method for explaining individual model predictions.

Content Moderation Rules

- Exact attribution of which rule conditions triggered a decision
- Clear explanation for:
  - Appeals
  - Policy audits
  - Internal debugging
- "This post was flagged primarily due to X, not Y."

Policy Based Classifiers

- Transparent justification for accept/reject outcomes
- Stable explanations across runs
- No sampling noise
- Especially important in regulated environments.

Hybrid AI Systems

- Feature-level explanations of the policy layer
- Clear separation between:
  - Learned behaviour
  - Enforced constraints

01

Main Bottleneck : SDD Compilation

In practice: explainability is cheap once the model is compiled

02

Model Class Restrictions

Only applies for models that can be expressed as boolean logic, and not directly applicable to large neural networks, or unstructured continuous models

03

Scalability depends on the Structure

Worst-case SDD size can be exponential

04

Assumptions

We are assuming fully factorised input distributions, more complex dependencies would require richer circuit representations, such as PSDDs.

Future Research Directions

From SDDs to Probabilistic SDDs (PSDDs)

Many real systems are inherently probabilistic

Extend exact SHAP computation from:
Boolean expectations→ probabilistic expectations

SHAP for Bayesian Network Classifiers via SDDs

Can exact SHAP be computed efficiently for Bayesian classifiers through circuit compilation?