Proceedings of Machine Learning Research 343:140–152, 2026 Probabilistic Graphical Models (PGM)

Selective Indicator Elimination in Arithmetic Circuits for Bipartite Noisy-OR Bayesian Networks

Cory Butz CORY.BUTZ@UREGINA.CA
Department of Computer Science
University of Regina
Regina, Canada

Alejandro Santoscoy-Rivero ASJ799@UREGINA.CA
Department of Computer Science
University of Regina
Regina, Canada

Jiří Vomlel VOMLEL@UTIA.CAS.CZ
Institute of Information Theory and Automation of the ASCR
Academy of Sciences of the Czech Republic
Prague, Czech Republic

Anders Madsen ANDERS@HUGIN.COM
HUGIN EXPERT A/S
Department of Computer Science, Aalborg University
Aalborg, Denmark

Editors: Gustau Camps-Valls, Manuele Leonelli and Gherardo Varando

Abstract

Bipartite noisy-OR Bayesian networks (BN2Os) have been compiled into arithmetic circuits (ACs) for exact inference, where evaluation traditionally consists of assigning indicator values at the circuit leaves and propagating values upward to the root. We introduce selective indicator elimination (SIE), a transformation for compiled BN2O ACs that removes indicator-controlled single-variable subcircuits and replaces them with cached values. The resulting method has two phases: an offline transformation phase that prunes such subcircuits and precomputes their replacement values, and an online evaluation phase that parses a query, assigns the appropriate cached values, and propagates over the transformed AC. Unlike prior AC compression methods that simplify static numeric computation by compressing arithmetic subcircuits consisting of multiplication and addition nodes, our approach is the first to eliminate indicator leaves together with their associated indicator-controlled computation while preserving support for general exact inference queries. We formally prove that SIE preserves exact inference correctness. Statistically significant improvements in both circuit size and exact inference runtime are empirically observed for 58% of the tested synthetic BN2Os.

Keywords: Bipartite noisy-OR Bayesian networks, Arithmetic circuits, Exact inference

© 2026 C. Butz, A. Santoscoy-Rivero, J. Vomlel & A. Madsen.

1. Introduction

Bipartite noisy-OR Bayesian networks (BN2Os) (Heckerman, 1989; Pearl, 1988) are an extensively studied subclass of Bayesian networks with applications including medical diagnosis (Richens et al., 2020) and causal machine learning (Dedieu et al., 2023). A BN2O consists of Boolean root and leaf variables, with directed edges exclusively from roots to leaves, and compact conditional probability specifications in which each leaf is parameterized linearly in the number of its parents. Despite this structural simplicity, exact inference in BN2Os remains computationally challenging (Heckerman, 1989; Ng and Jordan, 1999).

Compilation-based exact inference has been studied for BN2Os by transforming the network into an arithmetic circuit (AC), enabling repeated inference through numerical circuit evaluation. The ACE system (UCLA Automated Reasoning Group, 2026) compiles BN2Os using parent divorcing and elimination orders determined by the min-fill cost heuristic (Darwiche, 2009), while Vomlel and Savický (2008) showed that smaller compiled circuits can often be obtained using a tensor rank-one construction and HUGIN Expert A/S (2026) elimination orders. In standard AC exact inference, indicator values are assigned at the circuit leaves according to the query, and values are then propagated upward through the entire circuit to the root. Although prior work has reduced compiled circuit size, this evaluation model still requires numerical propagation from the indicator leaves for every query.

In BN2O compiled circuits, some internal computations depend entirely on the indicator configuration of a single Boolean variable. Since BN2O evidence is observed only at leaf variables, each leaf variable has only three admissible exact inference indicator configurations: positive evidence, negative evidence, or unobserved. Similarly, each root variable has only three admissible query configurations: queried true, queried false, or absent from the query. These computations can therefore assume only a small finite set of values, making their repeated recomputation during every query unnecessary.

In this paper, we introduce *selective indicator elimination*, a transformation for BN2O compiled circuits that removes indicator-controlled single-variable subcircuits and replaces them with cached replacement values. The method has two phases. In an offline transformation phase, eliminable indicator leaves and their associated computation are removed, and all admissible replacement values are precomputed and cached. In an online evaluation phase, a query is parsed, the appropriate cached values are assigned directly, and standard upward propagation is performed over the pruned circuit. Unlike prior AC compression methods that simplify static circuit structure Butz et al. (2025); Darwiche (2009), our approach selectively removes indicator leaves together with their associated computation while preserving support for general exact inference queries.

The proposed method exploits the Boolean structure of BN2Os, where the number of admissible replacement values remains small. We establish the correctness of the proposed transformation and evaluate it on the synthetic BN2O benchmarks of Vomlel and Savický (2008). Statistically significant improvements in both circuit size and exact inference runtime are empirically observed for 58% of the tested synthetic BN2Os. Extending this approach to higher-cardinality Bayesian networks is not immediate, as the number of cached replacement values would grow substantially. The interaction between selective indicator elimination and arithmetic-circuit parameter differentiation warrants further investigation.

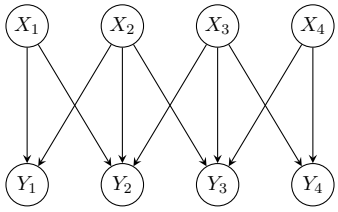


Figure 1: A BN2O with four roots and four leaves (Vomlel and Savický, 2008).

2. Background Knowledge

Here, we review bipartite noisy-OR Bayesian networks, arithmetic circuits, and exact inference therein.

We consider Boolean random variables. Variables are denoted by capital letters (e.g., X_i, Y_j, W), while their instantiated values are denoted either symbolically (e.g., x_i, y_j, w) or explicitly as Boolean values 0 and 1, as appropriate. Boldface letters denote vectors of variables or their corresponding values.

A *Bayesian network* (BN) is defined by a pair $\mathcal{B} = (G, \mathcal{P})$. $G = (\mathcal{N}, E)$ is a directed acyclic graph (DAG), whose nodes $\mathcal{N}$ are the variables in $\mathcal{B}$ and whose edges E encode dependencies. $\mathcal{P}$ is a collection of conditional probability tables (CPTs) $\{P(W \mid \mathrm{pa}(W)) \mid W \in \mathcal{N}\}$, where $\mathrm{pa}(W)$ denotes the set of parents of W in G. The joint distribution induced by the BN, which satisfies the conditional independence relations encoded by G, is uniquely given by

$$P(\mathbf{W}) = \prod_{W \in \mathcal{N}} P(W \mid \mathrm{pa}(W)).$$

The noisy-OR gate is central to this paper.

Definition 1 *(Vomlel and Savický, 2008) A* noisy-OR gate *consists of $n+1$ Boolean variables $X_1, \ldots, X_n$ and Y, where Y is a noisy-OR of its parents $X_1, \ldots, X_n$. The associated Bayesian network has bipartite structure with directed edges $X_i \to Y$ for $i = 1, \ldots, n$.*

The conditional probability table of Y is defined as follows: for $(x_1, \ldots, x_n) \in \{0,1\}^n$,

$$P(Y = 0 \mid X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} p_i^{x_i},$$

where each parameter $p_i \in [0,1]$ is the inhibition probability associated with parent X_i.

Definition 2 (BN2O) *A* bipartite noisy-OR Bayesian network *(BN2O) consists of Boolean root variables $X_1, \ldots, X_r$ and Boolean leaf variables $Y_1, \ldots, Y_s$, with directed edges exclusively from root variables to leaf variables. Each root variable X_i has a prior distribution $P(X_i)$, and each leaf variable Y_j has a noisy-OR conditional probability table $P(Y_j \mid \mathrm{pa}(Y_j))$, where $\mathrm{pa}(Y_j) \subseteq \{X_1, \ldots, X_r\}$ denotes the parents of Y_j.*

Example 1 *Figure 1 shows a representative BN2O, where CPTs are not depicted.*

Arithmetic circuits (ACs) can be compiled from Bayesian networks (Darwiche, 2003). For the subclass of BN2Os, ACE employs parent divorcing during compilation, while Vomlel and Savický (2008) showed that tensor rank-one decomposition can often produce smaller compiled circuits. Both methods use variable elimination orderings, and those for tensor rank-one are computed by HUGIN Expert A/S (2026). An arithmetic circuit (AC) is a rooted directed acyclic graph whose internal nodes are summation or multiplication operations and whose leaf nodes are probability parameters or indicator variables. Sum nodes satisfy completeness, and multiplication nodes satisfy decomposability. We use $\mathrm{ch}(n)$, $\mathrm{de}(n)$, and $\mathrm{sc}(n)$ to denote the children, descendants, and scope of a node n, respectively.

Example 2 *The tensor rank-one construction using a HUGIN elimination order as described in (Vomlel and Savický, 2008) can compile the BN2O in Figure 1 into the AC depicted in Figure 2.*

The value of an AC is computed by assigning values to all indicator leaves according to the query and propagating values upward to the root. For a Boolean variable X, the indicators I_x and $I_{\bar{x}}$ correspond to the two possible states of X. Given a query, indicators inconsistent with the query are assigned value 0, while compatible indicators are assigned value 1.

Exact inference in BN2Os involves computing posterior probabilities of root variables given observations on leaf variables, where each leaf may be observed as a positive finding, a negative finding, or remain unobserved. A marginal probability such as $P(X = x)$ can be computed with one upward propagation through the AC. A conditional probability such as

$$P(X = x \mid Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)}$$

can be computed by evaluating the AC twice, followed by a single external division. Standard AC inference propagates values from the indicator leaves through the entire compiled circuit to the root for every query.

Example 3 *Let us focus on the four sum nodes $S_1, \ldots, S_4$ shown from left to right in bottom right-side of Figure 2, redrawn as Figure 3(a). Since this portion of the AC depends only on the indicators of X_1, the values computed at these four sum nodes depend only on the indicator assignment for X_1.*

If the query sets $X_1 = 1$, then $I_{X_1=0} = 0$ and $I_{X_1=1} = 1$, and upward propagation through the original subcircuits gives

$$(S_1, S_2, S_3, S_4) = (0.95,\ 0.1334,\ 0.6506,\ 0.0913).$$

If the query sets $X_1 = 0$, then $I_{X_1=0} = 1$ and $I_{X_1=1} = 0$, and the same four sum nodes evaluate to

$$(S_1, S_2, S_3, S_4) = (0.05,\ 0.05,\ 0.05,\ 0.05).$$

Finally, if X_1 is not mentioned in the query, then $I_{X_1=0} = 1$ and $I_{X_1=1} = 1$, and the four sum nodes evaluate to

$$(S_1, S_2, S_3, S_4) = (1.0,\ 0.1834,\ 0.7006,\ 0.1413).$$

Example 3 shows that each of the four non-leaf sum nodes has only three possible values, corresponding to the three admissible indicator configurations for variable X_1.

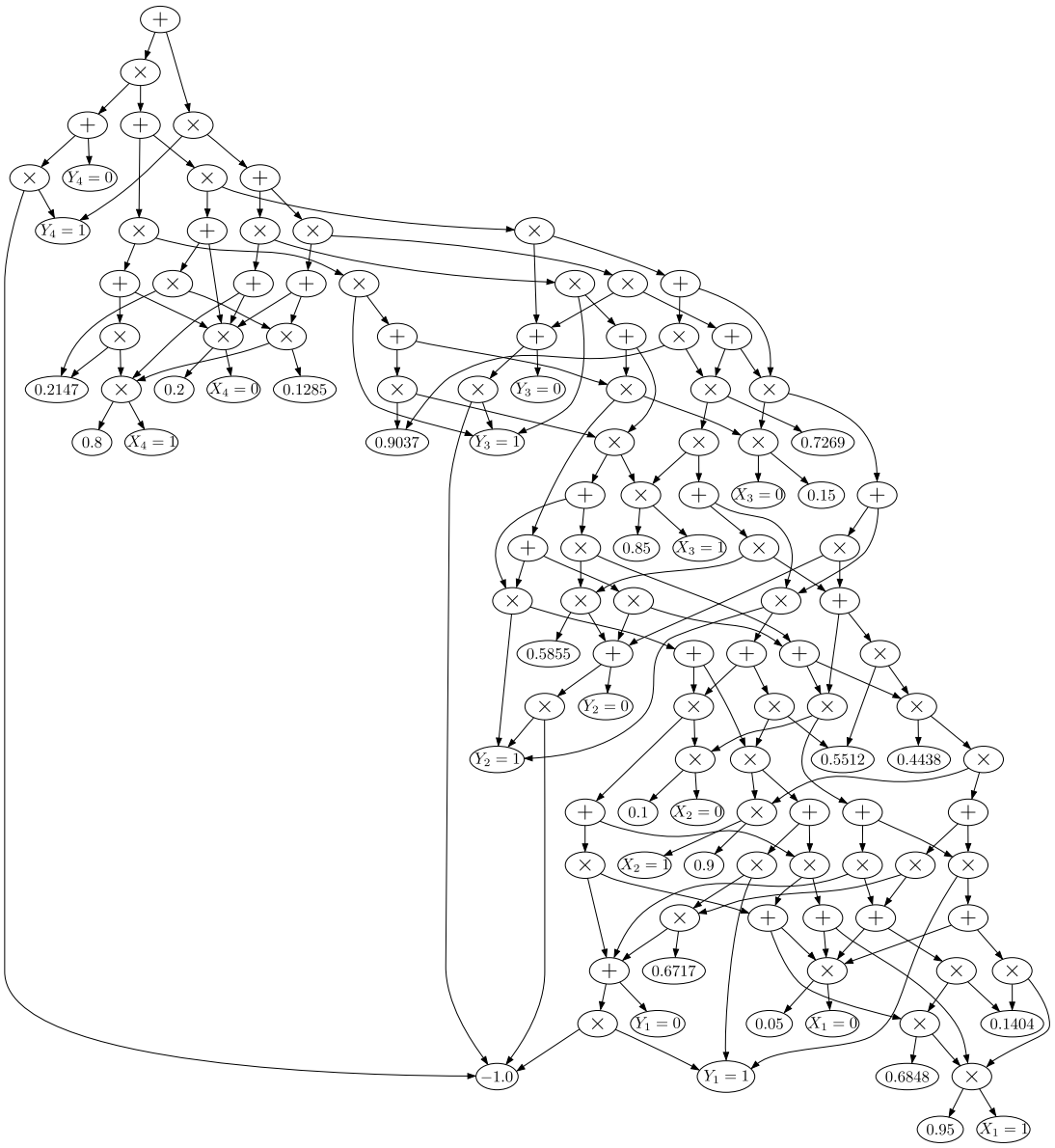


Figure 2: The tensor rank-one method using a HUGIN elimination order can compile the BN2O in Figure 1 into this AC.

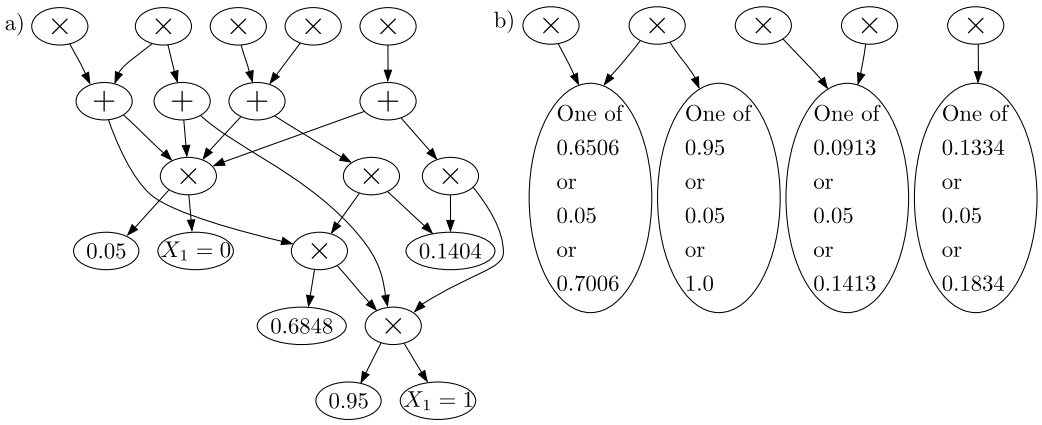


Figure 3: A portion of the AC compiled for the BN2O in Figure 1 before and after selective indicator elimination. Since each subcircuit in a) depends only on the indicators of X_1, it can be cached and replaced by a corresponding leaf in b).

3. Selective Indicator Elimination

In standard arithmetic-circuit exact inference for BN2Os, indicator values are assigned at the circuit leaves according to the query, and values are propagated upward through the entire circuit to the root. Since BN2Os consist exclusively of Boolean variables, each variable, whether root or leaf, has exactly three admissible indicator configurations during exact inference: $(1, 0)$, $(0, 1)$, and $(1, 1)$, corresponding to the variable being constrained to one state, the other state, or left unconstrained by the query.

We exploit this observation through *selective indicator elimination*. If the value of an internal arithmetic-circuit subcircuit depends exclusively on the indicators of a single Boolean variable, then that subcircuit can assume only three possible values. Rather than recomputing such subcircuits during every query, we apply an offline transformation that identifies maximal single-variable indicator-controlled subcircuits, removes their indicator leaves together with their associated computation, and precomputes the values those subcircuits can assume. During online inference, the query determines which cached replacement values are required, these values are assigned directly, and standard upward propagation proceeds over the transformed circuit to the root. This preserves support for general exact inference queries while avoiding repeated numerical computation over eliminated subcircuits.

Algorithm 1 summarizes the proposed offline transformation. The procedure identifies maximal internal subcircuits whose values depend on the indicators of exactly one Boolean variable, removes those subcircuits from the compiled arithmetic circuit, and replaces them with cached inputs corresponding to the three admissible indicator configurations of that variable.

Example 4 *The AC fragment in Figure 3(a) is a maximal single-variable subcircuit identified by Algorithm 1 with indicator scope is* $\{X_1\}$*. As shown in Example 3, the four sum nodes* $S_1, \ldots, S_4$ *(left to right) can assume only three possible value configurations, corresponding to the three admissible indicator assignments for the Boolean variable* X_1*.*

Algorithm 1: Selective indicator elimination

Input: Arithmetic circuit $\mathcal{C} = (V, E)$
Output: Pruned circuit $\mathcal{C}'$ with cached numeric values for the new leaves in $\mathcal{C}'$

`// Compute indicator scopes`
for *each leaf* λ **do**
  $\mathrm{sc}(\lambda) \leftarrow \{Z\}$ if λ is an indicator of variable Z, else $\emptyset$
end
for *each internal node* v *in reverse topological order* **do**
  $\mathrm{sc}(v) \leftarrow \bigcup_{u \in \mathrm{ch}(v)} \mathrm{sc}(u)$
end
`// Select maximal single-variable subcircuits`
$R \leftarrow \{v \in V : |\mathrm{sc}(v)| = 1$ **and** $\forall p \in \mathrm{pa}(v),\ |\mathrm{sc}(p)| > 1\}$
`// Build pruned circuit`
$\mathcal{C}' \leftarrow$ subgraph induced by $V \setminus \bigcup_{v \in R} \mathrm{de}(v)$
`// Cache replacement values`
for *each* $v \in R$ **do**
  let $\mathrm{sc}(v) = \{Z\}$
  evaluate subcircuit rooted at v under $(I_z, I_{\bar{z}}) = (1, 0)$ and store $\mathrm{cache}(v, z)$
  evaluate subcircuit rooted at v under $(I_z, I_{\bar{z}}) = (0, 1)$ and store $\mathrm{cache}(v, \bar{z})$
  evaluate subcircuit rooted at v under $(I_z, I_{\bar{z}}) = (1, 1)$ and store $\mathrm{cache}(v, *)$
end
return $\mathcal{C}'$, $\{\mathrm{cache}(v, \cdot) : v \in R\}$

Applying Algorithm 1 removes the indicator leaves for X_1 together with the associated internal computation beneath $S_1, \ldots, S_4$, replacing the eliminated subcircuits by cached leaves as shown in Figure 3(b).

If the query includes $X_1 = 1$, then the cached values assigned to the four newly created leaves $S_1, \ldots, S_4$ are

$$(0.95,\ 0.1334,\ 0.6506,\ 0.0913).$$

Otherwise, if the query includes $X_1 = 0$, then the cached values are

$$(0.05,\ 0.05,\ 0.05,\ 0.05).$$

Consequently, if X_1 does not appear in the query, then the cached values are

$$(1.00,\ 0.1834,\ 0.7006,\ 0.1413).$$

The same transformation is applied independently to every maximal single-variable subcircuit. Thereby, Algorithm 1 transforms the tensor rank-one AC in Figure 2 into the compressed AC in Figure 4.

Example 4 illustrates that online inference no longer evaluates over the eliminated arithmetic subcircuits in the original circuit. Instead, the SIE circuit directly assigns the appropriate cached values to the newly created leaves before standard upward propagation proceeds.

Theorem 3 *Let $\mathcal{C}$ be an arithmetic circuit compiled from a BN2O using the tensor rank-one construction, and let $\mathcal{C}'$ be the output of Algorithm 1 applied to $\mathcal{C}$. Then, for every exact inference query Q, the root value computed by $\mathcal{C}'$ equals the root value computed by $\mathcal{C}$ under the indicator assignment induced by Q.*

Proof Let Q be an arbitrary exact inference query, and let λ_Q denote the indicator assignment induced by Q. Consider any maximal subcircuit eliminated by Algorithm 1, and let v be its output node. By construction, the indicator scope of this subcircuit is a singleton set $\{X\}$ for some Boolean variable X. Hence, under exact AC inference, the value computed at v can depend on Q only through the three admissible indicator assignments for X:

$$(I_{X=1}, I_{X=0}) \in \{(1,0), (0,1), (1,1)\}.$$

Algorithm 1 evaluates the eliminated subcircuit offline under each of these three assignments and stores the corresponding values.

The eliminated subcircuit of $\mathcal{C}$ is replaced by a cached leaf at the same output location in the SIE circuit $\mathcal{C}'$. During online evaluation under λ_Q, this cached leaf is assigned exactly the value precomputed for the indicator configuration of X induced by Q. Therefore, for every query Q, the replacement leaf in $\mathcal{C}'$ has the same value as the output node v of the corresponding eliminated subcircuit in $\mathcal{C}$.

The maximal eliminated subcircuits are replaced independently, and every edge leaving such a subcircuit is reconnected to the corresponding cached leaf. All nodes and edges outside the eliminated subcircuits are unchanged. Thus, after the cached leaves are initialized, every remaining arithmetic operation in $\mathcal{C}'$ receives the same input values as the

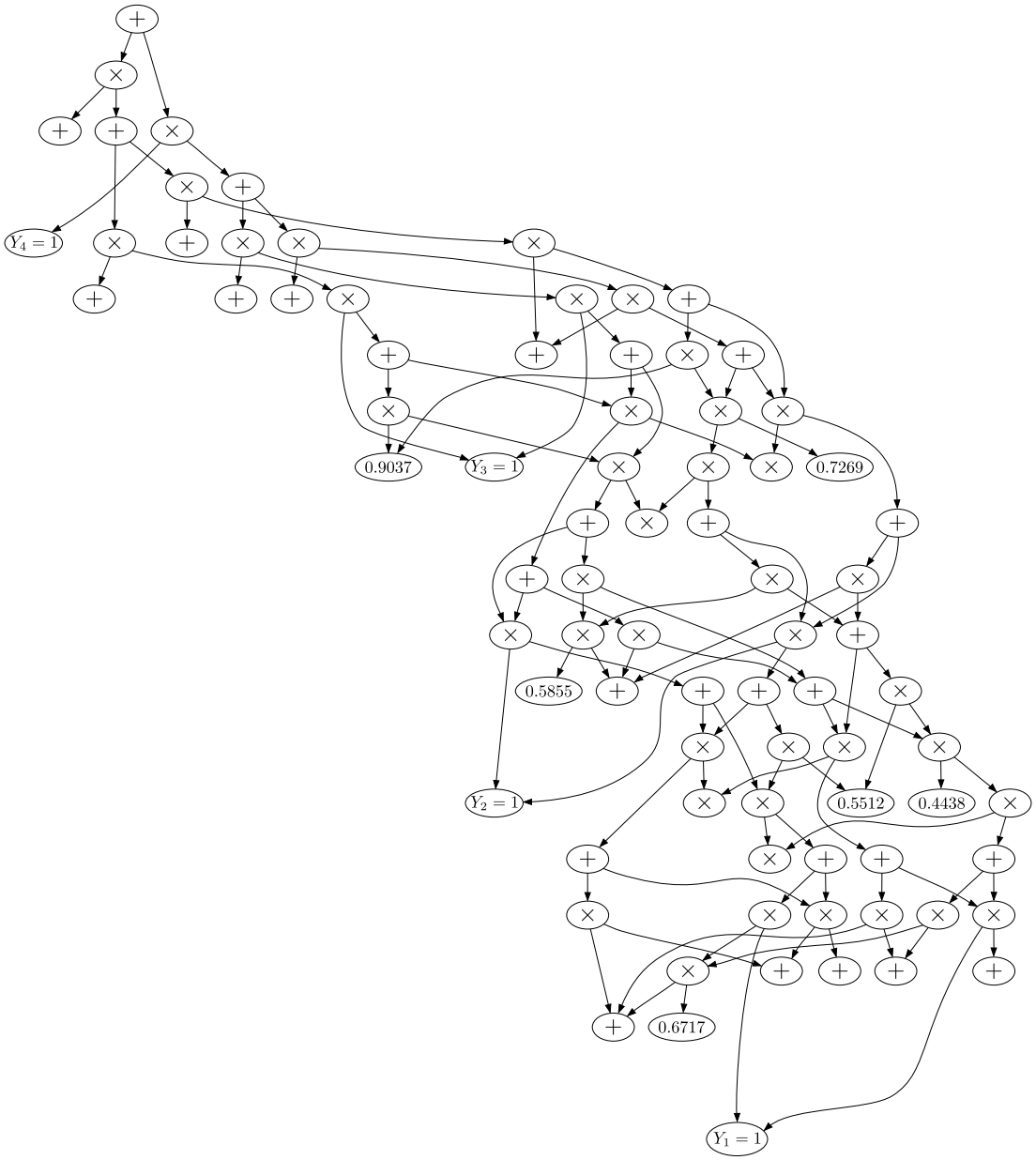


Figure 4: The SIE algorithm (Algorithm 1) transforms the tensor rank-one AC in Figure 2 into this compressed AC.

corresponding operation in $\mathcal{C}$ under λ_Q. By induction over a topological ordering of the remaining AC nodes, each corresponding node in $\mathcal{C}'$ and $\mathcal{C}$ computes the same value. In particular, the root values are equal. Since Q was arbitrary, the theorem holds for every exact inference query. ■

4. Experimental Results

The experiments evaluate SIE and the tensor rank-one construction of Vomlel and Savický (2008) with respect to both circuit size and exact inference runtimes.

The experiments were benchmarked on a workstation configured with Ubuntu 24.04.3 LTS, 64 GB RAM, and a 3.3 GHz AMD Ryzen 5 5600H CPU (6 cores/12 threads). Unlike Vomlel and Savický (2008), synthetic BN2Os were generated that include prior distributions for the root nodes.

We evaluated SIE on fixed fan-in BN2Os (four parents per leaf) using a grid search over network configurations, varying the number of Boolean root variables ($r \in \{10, 20, 30\}$), the number of Boolean leaf variables ($s \in \{10, 20, 30\}$), and the fixed number of parents per leaf ($p \in \{2, 4, 5, 10, 20\}$). This evaluation range was chosen in an attempt to ensure that tensor rank-one circuit compilation completed successfully for every tested configuration, enabling direct comparison of compiled circuit sizes before and after SIE. For each resulting r-s-p configuration with $r \geq p$, arithmetic circuits were compiled using the tensor rank-one construction of Vomlel and Savický (2008) with elimination orders generated by HUGIN Expert A/S (2026). There are 42 possible network configurations with $r \geq p$, of which the tensor rank-one method encounters a memory error when compiling 4 of them. SIE was then applied to the 38 rank-one compiled circuits, and the resulting transformed circuit sizes were recorded. Of these 38 network configurations, 22 are reported in Table 1. Configurations yielding insignificant size reductions were omitted for space considerations.

Table 1 reports the mean circuit sizes and standard deviations from 10 random seed networks of each r-s-p configuration, together with the percentage reduction relative to the original rank-one compiled circuits. According to the Wilcoxon signed-rank test with 95% confidence (p-value < 0.05), the circuit sizes between the baseline and SIE are significantly smaller in all 22 network configurations reported in Table 1, constituting 58% of the 38 available network configurations.

Table 1 shows that SIE can substantially reduce compiled circuit size, although the magnitude of the reduction depends strongly on network structure. The largest reductions occur in sparse, low fan-in networks, where savings exceed 50% for configuration 30-10-2, and remain above 25% for several additional fan-in-2 cases. Moderate reductions are also observed in some denser settings, such as 30-10-4 (46.9%) and 10-10-10 (47.7%).

We also compare exact inference running times. For each of the 10 random seeds of each network configuration, we perform inference of 10 random queries using random negative and positive observations (100 runs per network size). Table 2 reports the mean runtimes with standard deviations of inference performed on the original rank-one circuit and the SIE circuit, measured in milliseconds (ms), along with the savings percentage relative to inference in the original rank-one compiled circuits. For all configurations in Table 2 the inference times performed in the SIE AC are significantly faster than the original rank-

Table 1: Comparison of rank-one and SIE circuit sizes across fixed fan-in BN2O configurations. The savings column reports the percentage reduction achieved by selective indicator elimination relative to the rank-one circuit size. Values are reported as mean ± standard deviation. For each configuration, all pairwise comparisons yielded Wilcoxon statistic $W = 55$ and $p = 0.00097$.

Config	Rank-one	SIE	Savings (%)
10-10-2	236 ± 29	164 ± 33	31.3 ± 6.3
10-10-4	$1{,}480 \pm 300$	$1{,}390 \pm 300$	6.39 ± 1.55
10-10-10	$43{,}044 \pm 0$	$22{,}524 \pm 0$	47.67 ± 0
10-20-2	632 ± 83	551 ± 83	13.1 ± 1.9
10-20-4	$6{,}400 \pm 1{,}100$	$6{,}300 \pm 1{,}100$	1.41 ± 0.25
10-30-2	$1{,}100 \pm 130$	997 ± 129	9.26 ± 1.04
20-10-2	246 ± 16	130 ± 13	47.4 ± 2.3
20-10-4	734 ± 92	551 ± 86	25.2 ± 2.9
20-10-5	$2{,}170 \pm 360$	$1{,}870 \pm 360$	14.5 ± 2.7
20-10-10	$19{,}300 \pm 2{,}000$	$16{,}900 \pm 1{,}800$	12.5 ± 2.1
20-10-20	$84{,}024 \pm 0$	$43{,}004 \pm 0$	48.82 ± 0
20-20-2	520 ± 32	378 ± 31	27.4 ± 2.1
20-20-20	—	NA	NA
20-30-2	$1{,}050 \pm 160$	897 ± 165	14.7 ± 2.7
20-30-20	—	NA	NA
30-10-2	299.0 ± 3.2	134.8 ± 5.4	54.9 ± 1.9
30-10-4	426 ± 20	227 ± 18	46.9 ± 2.1
30-10-5	$1{,}160 \pm 250$	865 ± 240	26.2 ± 4.9
30-10-10	$14{,}500 \pm 3{,}100$	$13{,}400 \pm 3{,}100$	8.08 ± 1.68
30-10-20	$52{,}500 \pm 1{,}700$	$43{,}100 \pm 1{,}300$	17.9 ± 1.6
30-20-2	476 ± 70	284 ± 62	40.8 ± 4.1
30-20-4	$9{,}600 \pm 2{,}000$	$9{,}300 \pm 2{,}000$	3.68 ± 0.87
30-20-20	$42{,}900{,}000 \pm 1{,}700{,}000$	$41{,}400{,}000 \pm 1{,}700{,}000$	3.44 ± 0.67
30-30-2	796 ± 62	589 ± 66	26.1 ± 2.6
30-30-10	—	NA	NA
30-30-20	—	NA	NA

one AC according to the Wilcoxon signed-rank test with 95% confidence (p-value < 0.05), constituting 58% of of the 38 available network configurations.

The results in Tables 1 and 2 suggest that SIE is most effective on sparsely connected BN2Os and less sensitive to overall network size. As connectivity increases, opportunities for elimination diminish, as reflected by the near-zero savings observed in many fan-in-10 and fan-in-20 configurations.

5. Conclusion

We introduced *selective indicator elimination* (SIE), an offline transformation for arithmetic circuits compiled from bipartite noisy-OR Bayesian networks (BN2Os). SIE removes max-

Table 2: Comparison of rank-one and SIE circuit inference runtime across fixed fan-in BN2O configurations. The savings column reports the percentage reduction achieved by selective indicator elimination relative to the rank-one circuit size.

Config	Rank-one	SIE	Savings (%)
10-10-2	0.175 ± 0.014	0.160 ± 0.020	8.7 ± 6.4
10-10-4	0.69 ± 0.11	0.67 ± 0.11	3.9 ± 3.8
10-10-10	18.78 ± 0.24	11.12 ± 0.13	40.7 ± 0.9
10-20-2	0.351 ± 0.029	0.330 ± 0.031	6.0 ± 3.8
10-20-4	2.48 ± 0.39	2.42 ± 0.35	2.4 ± 3.0
10-30-2	0.561 ± 0.043	0.513 ± 0.041	8.4 ± 3.6
20-10-2	0.187 ± 0.011	0.144 ± 0.009	22.9 ± 4.8
20-10-4	0.409 ± 0.047	0.327 ± 0.046	20.1 ± 4.1
20-10-5	1.01 ± 0.13	0.88 ± 0.14	13.0 ± 3.4
20-10-10	7.0 ± 0.7	6.0 ± 0.6	14.3 ± 3.1
20-10-20	36.89 ± 0.33	19.93 ± 0.23	46.0 ± 0.7
20-20-2	0.330 ± 0.019	0.274 ± 0.015	16.7 ± 4.9
20-20-20	—	NA	NA
20-30-2	0.571 ± 0.062	0.501 ± 0.060	12.2 ± 3.1
20-30-20	—	NA	NA
30-10-2	0.234 ± 0.015	0.147 ± 0.009	37.0 ± 4.6
30-10-4	0.291 ± 0.014	0.186 ± 0.012	36.0 ± 3.3
30-10-5	0.596 ± 0.093	0.450 ± 0.090	24.8 ± 5.6
30-10-10	5.1 ± 0.9	4.6 ± 0.9	9.6 ± 3.5
30-10-20	22 ± 23	18 ± 16	19.0 ± 5.5
30-20-2	0.319 ± 0.030	0.234 ± 0.035	26.8 ± 9.2
30-20-4	4.2 ± 0.7	4.1 ± 0.8	4.1 ± 2.8
30-20-20	$13{,}600 \pm 700$	$13{,}100 \pm 800$	4.0 ± 3.1
30-30-2	0.484 ± 0.030	0.384 ± 0.051	20.7 ± 8.4
30-30-10	—	NA	NA
30-30-20	—	NA	NA

imal single-variable indicator-controlled subcircuits, replaces them with cached values, and initializes the pruned circuit using those values before standard propagation.

We established the correctness of the proposed transformation and evaluated its effect on compiled circuit size across a range of synthetic fixed fan-in BN2Os. Statistically significant improvements in both circuit size and exact inference runtime are empirically observed for 58% of the tested synthetic BN2Os, as seen in Tables 1 and 2. SIE exploits the fact that BN2Os consist entirely of Boolean variables, meaning that each indicator-controlled subcircuit admits only three possible configurations. Future work includes comparing SIE with ACE, extending the evaluation to general Bayesian networks, and studying its interaction with parameter differentiation.

Acknowledgments

This research was partially supported by the NSERC Discovery Grant Program (Grant No. 238880).

References

C. Butz, C. Lewis, A. Santoscoy-Rivero, and A. Madsen. Arithmetic circuit compilation using symbolic probabilistic inference and indicator-determined buckets. In *Proceedings of the Eighteenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 191–204, 2025.

A. Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003.

A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

A. Dedieu, G. Zhou, D. George, and M. Lázaro-Gredilla. Learning noisy-OR bayesian networks with max-product belief propagation. In *Proceedings of the International Conference on Machine Learning*, pages 7426–7448, 2023.

D. Heckerman. A tractable inference algorithm for diagnosing multiple diseases. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 163–171, 1989.

HUGIN Expert A/S. Hugin decision engine. https://www.hugin.com/, 2026.

A. Y. Ng and M. I. Jordan. Approximate inference algorithms for two-layer bayesian networks. *Advances in Neural Information Processing Systems*, 12:537–543, 1999.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

J. G. Richens, C. M. Lee, and S. Johri. Improving the accuracy of medical diagnosis with causal machine learning. *Nature Communications*, 11(1):3923, 2020.

UCLA Automated Reasoning Group. Ace: Arithmetic circuit evaluator. http://reasoning.cs.ucla.edu/ace, 2026.

J. Vomlel and P. Savický. Arithmetic circuits of the noisy-or models. In *Proceedings of the European Workshop on Probabilistic Graphical Models*, pages 297–304, 2008.