Weighted Model Counting with Algebraic Decision Diagrams

Vu Phan's M.S. thesis defense — Computer Science Department, Rice University

Thursday 2019-11-07

M.S. committee: Dr. Moshe Vardi (chair), Dr. Devika Subramanian, Dr. Swarat Chaudhuri

Overview: Model Counting

Satisfiability problem (SAT): whether Boolean formula has satisfying assignment

- Complexity: NP-complete [Cook, 1971]

Model counting problem (#SAT): number of satisfying assignments of Boolean formula

- Complexity: #P-complete [Valiant, 1979]
- Applications in probabilistic reasoning:
  - Power-transmission reliability estimation [Duenas-Osorio et al., 2017]
  - Medical diagnosis [Shwe et al., 1991]

Contents

Progress

Background: Boolean Logic

$\mathbb{B} = \{0, 1\}$ (**Boolean set**)

Variable $x \in \mathbb{B}$	**Negation** $\neg x$
0	1
1	0

x_1	x_2	**Disjunction** $x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	**Conjunction** $x_1 \wedge x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Problem: Unweighted Model Counting

Formula: $F = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3)$

Variable set of F: $V = \text{Vars}(F) = \{x_1, x_2, x_3\}$

Assignment set over V: $2^V = \{\varnothing, \{x_1\}, \{x_2\}, \{x_3\}, \{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\}, V\}$

Assignment $\alpha \in 2^V$			$F(\alpha) : 2^V \to \mathbb{B}$	Is α a **model** of F?
x_1	x_2	x_3		
0	0	0	0	Yes iff $F(\alpha) = 1$
0	0	1	0	
0	1	0	1	
0	1	1	0	
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	1	

Unweighted model count of F: $\#F = \sum_{\alpha \in 2^V} F(\alpha) = 5$

Problem: Weighted Model Counting

Weight function: $W : 2^V \to \mathbb{R}$ (**real-number set**)

Assignment $\alpha \in 2^V$			$W(\alpha)$
x_1	x_2	x_3	
0	0	0	2.0
0	0	1	3.0
0	1	0	2.0
0	1	1	3.0
1	0	0	3.0
1	0	1	3.0
1	1	0	4.0
1	1	1	4.0

Problem: Weighted Model Counting

Formula: $F : 2^V \to \mathbb{B}$

Weight function: $W : 2^V \to \mathbb{R}$

Formula-weight product: $F \cdot W : 2^V \to \mathbb{R}$

Assignment $\alpha \in 2^V$			$F(\alpha)$	$W(\alpha)$	$(F \cdot W)(\alpha)$
x_1	x_2	x_3			
0	0	0	0	2.0	0.0
0	0	1	0	3.0	0.0
0	1	0	1	2.0	2.0
0	1	1	0	3.0	0.0
1	0	0	1	3.0	3.0
1	0	1	1	3.0	3.0
1	1	0	1	4.0	4.0
1	1	1	1	4.0	4.0

Weighted model count of F w.r.t. W: $\#(F, W) = \sum_{\alpha \in 2^V} (F \cdot W)(\alpha) = 16.0$

Related Work: Model Counting

Unweighted model counting:

- Exact unweighted model counting:
  - `sharpSAT` [Thurley, 2006]
    Component caching and implicit Boolean constraint propagation
  - Counting knight's tours [Löbbing and Wegener, 1996]
    Binary decision diagrams (BDDs)
- Probabilistically-exact unweighted model counting:
  - `GANAK` [Sharma et al., 2019]
    Probabilistic component caching
- Approximate unweighted model counting:
  - `ApproxMC2` [Chakraborty et al., 2016]
    Universal hash functions

From weighted to unweighted exact model counting:

- Polynomial-time reduction [Chakraborty et al., 2015]
  Chain formulas

Related Work: Model Counting

Exact weighted model counting:

- **Search**: DPLL-like exploration of solution space
  - `Cachet` [Sang et al., 2004]
    Component caching and clause learning
- **Knowledge compilation**: efficient data structure for formula
  - `c2d` [Darwiche, 2004]
    Deterministic decomposable negation normal form (d-DNNF)
  - `d4` [Lagniez and Marquis, 2017]
    Decision decomposable negation normal form (Decision-DNNF)
  - `miniC2D` [Oztok and Darwiche, 2015]
    Sentential decision diagrams (SDDs)
- <u>Contribution</u>: `ADDMC` [Phan, 2019; Dudek et al., 2019b]
  - Algebraic decision diagrams (ADDs) for components of formula
  - Combining ADDs using dynamic programming

Progress

Data Structure: Binary Decision Diagrams [Bryant, 1986]

Formula $F : 2^V \to \mathbb{B}$ with variable count $n = |V|$

Exhaustive table

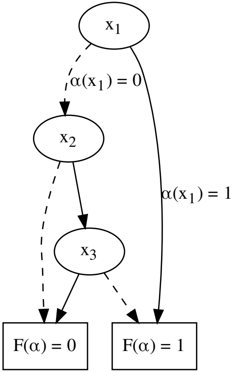
Inefficient data structure: $\Theta(2^n)$

Long construction & large storage, always

Assignment $\alpha \in 2^V$			$F(\alpha)$
x_1	x_2	x_3	
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Binary decision diagram (BDD)

Efficient data structure: $O(2^n)$



Root-terminal path $\alpha \in 2^V$

Data Structure: Algebraic Decision Diagrams [Bahar et al., 1997]

Weight function $W : 2^V \to \mathbb{R}$ with variable count $n = |V|$

Exhaustive table

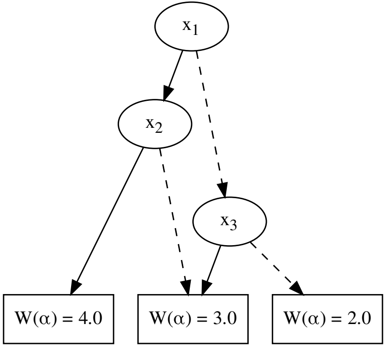
Inefficient data structure: $\Theta(2^n)$

Long construction & large storage, always

Assignment $\alpha \in 2^V$			$W(\alpha)$
x_1	x_2	x_3	
0	0	0	2.0
0	0	1	3.0
0	1	0	2.0
0	1	1	3.0
1	0	0	3.0
1	0	1	3.0
1	1	0	4.0
1	1	1	4.0

Algebraic decision diagram (ADD)

Efficient data structure: $O(2^n)$



Root-terminal path $\alpha \in 2^V$

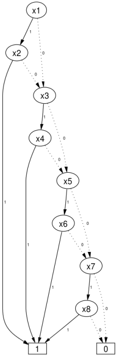
Diagram Variable Orders and BDD/ADD Sizes: Example [Beyer, 2019]

Formula $(x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee (x_5 \wedge x_6) \vee (x_7 \wedge x_8)$

Diagram variable order: injection $\{x_1, x_2, \ldots, x_8\} \to \{1, 2, \ldots, 8\}$

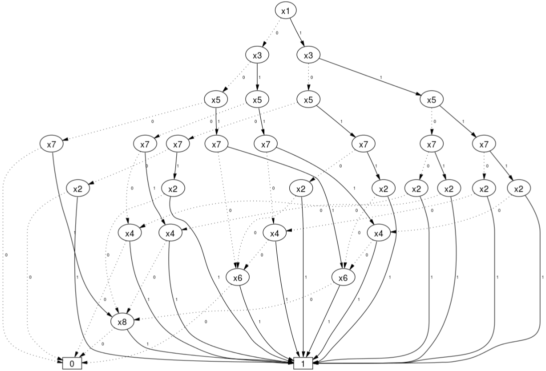
BDD with diagram variable order

$x_1 < x_2 < \ldots < x_8$



BDD with diagram variable order

$x_1 < x_3 < x_5 < x_7 < x_2 < x_4 < x_6 < x_8$



Boole (Shannon) Expansion

- Variable set $V = \{x_1, x_2, \ldots, x_n\}$
- **Boole (Shannon) expansion**:
  - $g : 2^V \to \mathbb{B}$

$$g(x_1, x_2, \ldots, x_n) = (x_1 \wedge g(1, x_2, \ldots, x_n)) \vee (\neg x_1 \wedge g(0, x_2, \ldots, x_n))$$

  - $h : 2^V \to \mathbb{R}$

$$h(x_1, x_2, \ldots, x_n) = x_1 \cdot h(1, x_2, \ldots, x_n) + (1 - x_1) \cdot h(0, x_2, \ldots, x_n)$$

Projection: Satisfiability Problem

- Formula $F : 2^V \to \mathbb{B}$
- **Projection** of F w.r.t. x_1 is $\exists_{x_1} F : 2^{V \setminus \{x_1\}} \to \mathbb{B}$

$$(\exists_{x_1} F)(x_2, \ldots, x_n) = F(0, x_2, \ldots, x_n) \vee F(1, x_2, \ldots, x_n)$$

- Exhaustive projection

$$\exists_{x_n} \ldots \exists_{x_2} \exists_{x_1} F = F(0, 0, \ldots, 0) \vee F(0, 0, \ldots, 1) \vee \ldots \vee F(1, 1, \ldots, 1)$$

Proposition 1 (Satisfiability via Projection [Pan and Vardi, 2004])

$$F \in \mathrm{SAT} \Leftrightarrow \exists_{x_n} \ldots \exists_{x_2} \exists_{x_1} F = 1$$

Projection: Unweighted Model Counting Problem

- Formula $F : 2^V \to \mathbb{B}$ as function $2^V \to \mathbb{N}$ (**natural-number set** $\{0, 1, 2, \ldots\}$)
- **Projection** of F w.r.t. x_1 is $\sum_{x_1} F : 2^{V \setminus \{x_1\}} \to \mathbb{N}$

$$\left(\sum_{x_1} F\right)(x_2, \ldots, x_n) = F(0, x_2, \ldots, x_n) + F(1, x_2, \ldots, x_n)$$

- Exhaustive projection

$$\sum_{x_n} \ldots \sum_{x_2} \sum_{x_1} F = F(0, 0, \ldots, 0) + F(0, 0, \ldots, 1) + \ldots + F(1, 1, \ldots, 1)$$

Remark 1 (Unweighted Model Counting via Projection)

$$\#F = \sum_{x_n} \ldots \sum_{x_2} \sum_{x_1} F$$

Projection: Weighted Model Counting Problem

- Formula $F : 2^V \to \mathbb{B}$, weight function $W : 2^V \to \mathbb{R}$, product $F \cdot W : 2^V \to \mathbb{R}$
- Projection of $F \cdot W$ w.r.t. x_1 is $\sum_{x_1}(F \cdot W) : 2^{V \setminus \{x_1\}} \to \mathbb{R}$

$$\left(\sum_{x_1}(F \cdot W)\right)(x_2, \ldots, x_n) = (F \cdot W)(0, x_2, \ldots, x_n) + (F \cdot W)(1, x_2, \ldots, x_n)$$

- Exhaustive projection

$$\sum_{x_n} \ldots \sum_{x_2} \sum_{x_1}(F \cdot W) = (F \cdot W)(0, 0, \ldots, 0) + \ldots + (F \cdot W)(1, 1, \ldots, 1)$$

Theorem 1 (Weighted Model Counting via Projection)

$$\#(F, W) = \sum_{x_n} \ldots \sum_{x_2} \sum_{x_1}(F \cdot W)$$

Monolithic Representation versus Factored Representation

Naive approach: using *monolithic representation* of formula F and weight function W

- Constructs big ADDs for F and W with n variables
- Scales poorly for large instances: ADD size is $\mathrm{O}\left(2^n\right)$

Contribution: algorithm that exploits *factored representation* of F and W

- Constructs small ADDs for factors of F and W
- Combines ADDs iteratively while keeping combinations small by:
  - Choosing which ADDs to combine heuristically
  - Applying early projection aggressively

Progress

Factored Representation: Conjunctive Normal Form (CNF) Formula

Formula:

$$F = (x_1 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge x_3$$

- Variables: x_1, x_2, x_3
- **Positive literals** are non-negated variables: x_1, x_2, x_3
- **Negative literals** are negated variables: $\neg x_2, \neg x_3$

Factored Representation: Conjunctive Normal Form (CNF) Formula

Formula:

$$F = (x_1 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge x_3$$

- Variables: x_1, x_2, x_3
- Positive literals are non-negated variables: x_1, x_2, x_3
- Negative literals are negated variables: $\neg x_2, \neg x_3$
- **Clauses** are disjunctions of literals: $x_1 \vee x_3, \neg x_2 \vee x_3, x_2 \vee \neg x_3, x_3$
- **Conjunctive normal form (CNF) formula** is conjunction of clauses: F

Factoring (formula and clauses as functions $2^V \to \mathbb{B}$):

$$F = (x_1 \vee x_3) \cdot (\neg x_2 \vee x_3) \cdot (x_2 \vee \neg x_3) \cdot x_3$$

Factored Representation: Literal-Weight Function

Literal weights of variable x: $\texttt{weight}(x), \texttt{weight}(\neg x) \in \mathbb{R}$
Unit-weight functions: giving pairs of literal weights

$$W_{x_1} : 2^{\{x_1\}} \to \mathbb{R} \qquad \text{where } \varnothing \mapsto \texttt{weight}(\neg x_1) \text{ and } \{x_1\} \mapsto \texttt{weight}(x_1)$$

$$W_{x_2} : 2^{\{x_2\}} \to \mathbb{R} \qquad \text{where } \varnothing \mapsto \texttt{weight}(\neg x_2) \text{ and } \{x_2\} \mapsto \texttt{weight}(x_2)$$

Literal-weight function over $V = \{x_1, x_2\}$ is $W : 2^V \to \mathbb{R}$

$$\begin{aligned}
W(\varnothing) &= W_{x_1}(\varnothing) \cdot W_{x_2}(\varnothing) \\
W(\{x_1\}) &= W_{x_1}(\{x_1\}) \cdot W_{x_2}(\varnothing) \\
W(\{x_2\}) &= W_{x_1}(\varnothing) \cdot W_{x_2}(\{x_2\}) \\
W(V) &= W_{x_1}(\{x_1\}) \cdot W_{x_2}(\{x_2\})
\end{aligned}$$

Factoring:

$$W = W_{x_1} \cdot W_{x_2}$$

Factored Representation: Literal-Weighted Model Count of CNF Formula

Construct factors of:

- Conjunctive normal form (CNF) formula F with clauses C:

$$F = \prod_{C \in F} C$$

- Literal-weight function W with variable set V:

$$W = \prod_{x \in V} W_x$$

Compute weighted model count of F w.r.t. W:

$$\#(F, W) = \sum_{x_n} \cdots \sum_{x_2} \sum_{x_1} (F \cdot W) = \sum_{x_n} \cdots \sum_{x_2} \sum_{x_1} \left(\prod_{C \in F} C \cdot \prod_{x \in V} W_x \right)$$

Avoid projecting all variables ($\sum_{x_n} \ldots \sum_{x_2} \sum_{x_1}$) after processing big product ($F \cdot W$)

- Project each variable ($\sum_x$) as early as possible while processing small products ($C \cdot W_x$)

Early Projection

Theorem 2

If we have:

- *Variable sets* Y, Z
- *Functions* $g : 2^Y \to \mathbb{R}$, $h : 2^Z \to \mathbb{R}$
- *Variable* $x \in Y \setminus Z$

Then:

$$\sum_x (g \cdot h) = \left(\sum_x g \right) \cdot h$$

Early projection can reduce sizes of intermediate computations

- Database query optimization [Kolaitis and Vardi, 2000]
- Satisfiability problem [Pan and Vardi, 2005]

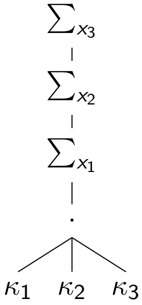
Early Projection: Unweighted Model Counting

CNF formula $F = (x_1 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge x_3$

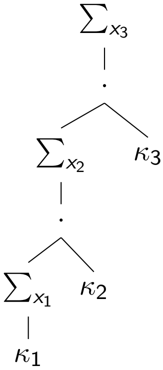
Clusters (partition of clauses)

$$\kappa_1 = \{x_1 \vee x_3\}$$
$$\kappa_2 = \{\neg x_2 \vee x_3,\ x_2 \vee \neg x_3\}$$
$$\kappa_3 = \{x_3\}$$

Late projection



Early projection



$$\#F = \sum_{x_3} \sum_{x_2} \sum_{x_1} (\kappa_1 \cdot \kappa_2 \cdot \kappa_3) = \sum_{x_3} \left(\sum_{x_2} \left(\sum_{x_1} \kappa_1 \cdot \kappa_2\right) \cdot \kappa_3\right)$$

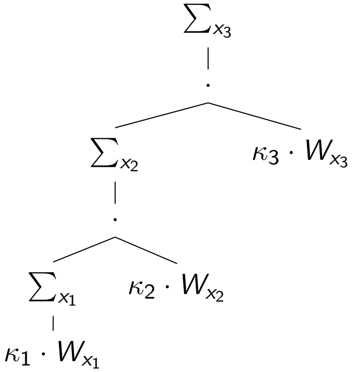
Early Projection: Weighted Model Counting

Formula $F = (x_1 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge x_3$ with weight function $W = W_{x_1} \cdot W_{x_2} \cdot W_{x_3}$

Clusters (partition of clauses)

$$\kappa_1 = \{x_1 \vee x_3\}$$
$$\kappa_2 = \{\neg x_2 \vee x_3, x_2 \vee \neg x_3\}$$
$$\kappa_3 = \{x_3\}$$

Early projection



$$\#(F, W) = \sum_{x_3} \left(\sum_{x_2} \left(\sum_{x_1} (\kappa_1 \cdot W_{x_1}) \cdot \kappa_2 \cdot W_{x_2}\right) \cdot \kappa_3 \cdot W_{x_3}\right)$$

Algorithm

Algorithm 1: Computing Literal-Weighted Model Count of CNF Formula

Input: Formula $F = \{C_1, C_2, \ldots, C_m\}$ and weight function W over set V of n variables

$\gamma \leftarrow$ **cluster-variable-order**(V) /* function $\gamma : V \to \{1, 2, \ldots, n\}$ */
$\gamma' \leftarrow$ **clause-order**(F, γ) /* function $\gamma' : F \to \{1, 2, \ldots, n\}$ */
for $i = 1, 2, \ldots, n$
    $\kappa_i \leftarrow \{C \in F : \gamma'(C) = i\}$
for $i = 1, 2, \ldots, n$
    $V_i \leftarrow \mathrm{Vars}(\kappa_i) \setminus \cup_{p>i} \mathrm{Vars}(\kappa_p)$
for $i = 1, 2, \ldots, n$
    $A_i \leftarrow \prod_{B \in \kappa_i} B$
    for $x \in V_i$
        $A_i \leftarrow \sum_x (A_i \cdot W_x)$ /* $W = W_{x_1} \cdot W_{x_2} \cdot \ldots \cdot W_{x_n}$ */
    $j \leftarrow$ **cluster-choice**(A_i, i) /* $j > i$ */
    $\kappa_j \leftarrow \kappa_j \cup \{A_i\}$
return A_n

Heuristics for Algorithm with ADDs

CNF formula $F = \{C_1, C_2, \ldots, C_m\}$ over set V of n variables

To construct ADDs:

- **Diagram variable-order heuristic**: function $\delta : V \to \{1, 2, \ldots, n\}$
  ADD size depends heavily on δ

To build clusters:

- **Cluster variable-order heuristic**: function $\gamma : V \to \{1, 2, \ldots, n\}$
- **Clause-order heuristic**: function $\gamma' : F \to \{1, 2, \ldots, n\}$

To combine clusters:

- **Cluster-choice heuristic**: how to choose which clusters to combine at each step

Heuristics: Gaifman Graphs for Variable Orders

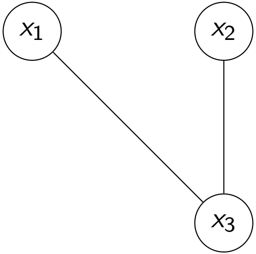
Primal constraint (Gaifman) graph of CNF formula:

- Each vertex corresponds to a variable
- Two vertices are adjacent iff both corresponding variables appear in the same clause

Formula:

$$(x_1 \lor x_3) \land (\neg x_2 \lor x_3) \land (x_2 \lor \neg x_3) \land x_3$$

Gaifman graph:



Heuristics: Diagram Variable Order and Cluster Variable Order

Heuristics to find vertex order for Gaifman graph (corresponding to variable order for formula):

- **Maximum-cardinality search (MCS)** [Tarjan and Yannakakis, 1984]
  Iteratively choose a vertex adjacent to the greatest number of previously chosen vertices
- **Inverse MCS (InvMCS)**
- **Lexicographic search for perfect order (LexP)** [Rose et al., 1976]
  1. Assign to each vertex an initially empty **label (reverse-sorted list of numbers)**
  2. For $i = n, n-1, \ldots, 1$:
     1. Choose a vertex u whose label is lexicographically largest
     2. Add i to labels of neighbors of u
- **Inverse LexP (InvLexP)**
- **Random**

Heuristics: Clause Order

Given:

- CNF formula $F = \{C_1, C_2, \ldots, C_m\}$ over set V of n variables
- Cluster variable order $\gamma : V \to \{1, 2, \ldots, n\}$

Heuristics to find clause order $\gamma' : F \to \{1, 2, \ldots, n\}$:

- **Bucket elimination (BE)** [Dechter, 1999]

$$\gamma'(C) = \min_{x \in \mathsf{Vars}(C)} \gamma(x)$$

- **Bouquet's Method (BM)** [Bouquet, 1999]

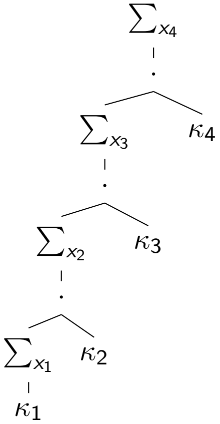
$$\gamma'(C) = \max_{x \in \mathsf{Vars}(C)} \gamma(x)$$

Heuristics: Cluster Choice

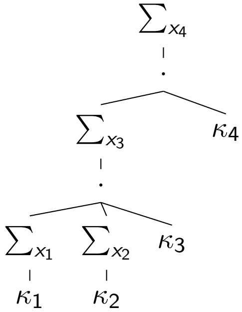
Given clusters (partition of clauses in CNF formula)

$$\kappa_1 = \{x_1 \vee x_3\}$$
$$\kappa_2 = \{\neg x_2 \vee x_3, x_2 \vee \neg x_3\}$$
$$\kappa_3 = \{x_3 \vee x_4\}$$
$$\kappa_4 = \{x_4\}$$

List: combines each projected cluster with the following cluster



Tree: combines each projected cluster with the furthest possible cluster



Contributions: Theoretical Framework and Empirical Evaluation

Contributions:

1. Algorithm for weighted model counting using algebraic decision diagrams (ADDs)
    - Exploiting factored representation of:
        - Conjunctive normal form (CNF) formula $F = \prod_{C \in F} C$
        - Literal-weighted function $W = \prod_{x \in V} W_x$
    - Constructing small ADDs for factors of F and W
    - Combining ADDs iteratively while keeping combinations small by:
        - Choosing which ADDs to combine heuristically
        - Applying early projection aggressively
2. Tool for weighted model counting: Algebraic Decision Diagram Model Counter (`ADDMC`)
    - Analysis of `ADDMC` heuristics
    - Comparison of `ADDMC` to state-of-the-art weighted model counters

    Public GitHub repository:
    `https://github.com/vardigroup/ADDMC`

Progress

Benchmarks

1914 **benchmarks**: CNF literal-weighted model counting problem instances

- 1091 benchmarks with literal weights in interval $[0, 1]$
- 823 originally unweighted benchmarks
  - Randomly generating literal weights:
    - Either $\texttt{weight}\,(x) = 0.5$ and $\texttt{weight}\,(\neg x) = 1.5$
    - Or $\texttt{weight}\,(x) = 1.5$ and $\texttt{weight}\,(\neg x) = 0.5$

  These weights reduce floating-point underflow/overflow for all model counters

Experiment 1: Comparing `ADDMC` Heuristics

Rice NOTS Linux cluster:

- Hardware: Xeon E5-2650v2 CPUs (2.60-GHz)
- Memory limit: 24 GB
- **Time limit: 10 seconds**

Experiment 1: Comparing `ADDMC` Heuristics

Setup:

- Benchmarks: 1914
- `ADDMC` heuristic configurations: 245

Table 1: Performance of best, second best, median, best **monolithic**, and worst heuristic configurations

Diagram var order	*Cluster var order*	*Clause order*	*Cluster choice*	Solved	Standing
MCS	**LexP**	**BM**	**Tree**	1202	Best
MCS	**InvLexP**	**BE**	**Tree**	1200	Best-2nd
LexP	**LexP**	**BE**	**List**	504	Median
LexP	**Mono**			188	Best-Mono
Random	**Random**	**BE**	**List**	53	Worst

Experiment 1: Comparing ADDMC Heuristics

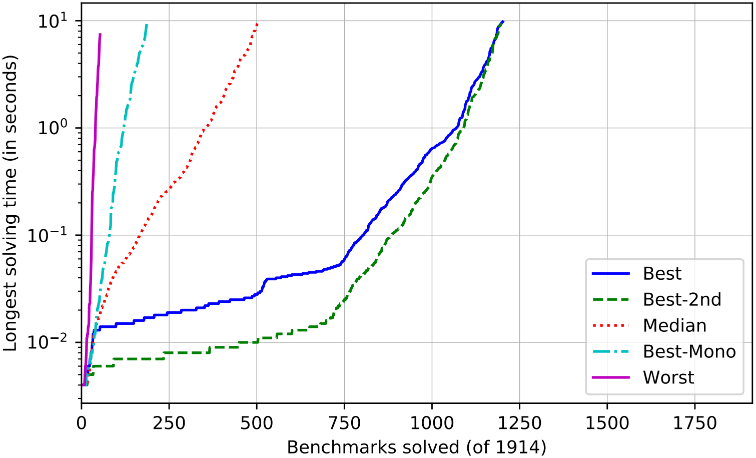


Figure 1: Runtime of best, second best, median, best monolithic, and worst heuristic configurations

Experiment 2: Comparing Weighted Model Counters

Rice NOTS Linux cluster:

- Hardware: Xeon E5-2650v2 CPUs (2.60-GHz)
- Memory limit: 24 GB
- **Time limit: 1000 seconds**

Experiment 2: Comparing Weighted Model Counters

Table 2: Performance of state-of-the-art weighted model counters

Weight model counters		Benchmarks solved (of 1914)		
		Unique solver	Fastest solver	Total
Virtual best solvers (VBS)	`VBS`	–	–	1771
	`VBS`* (no `ADDMC`)	–	–	1647
Actual solvers	`d4`	12	283	1587
	`c2d`	0	13	1417
	`miniC2D`	8	61	1407
	`ADDMC` **(our tool)**	**124**	**763**	1404
	`Cachet`	14	651	1383

Experiment 2: Comparing Weighted Model Counters

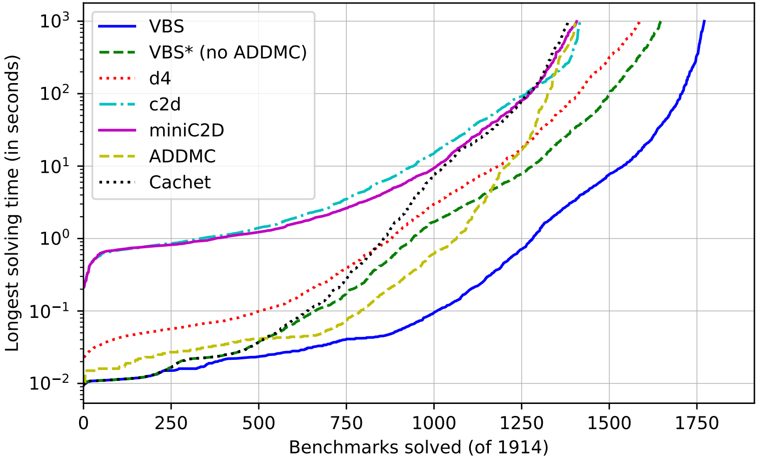


Figure 2: Runtime of weighted model counters

Summary

- Motivation: probabilistic reasoning applications
  - Power-transmission reliability estimation
  - Medical diagnosis
- Problem: model counting (#SAT)
  - Complexity: #P-complete
- Our approach:
  - Using algebraic decision diagrams (ADDs)
  - Exploiting factored representation

Empirical result: improvement for virtual best solver of weighted model counters

To Ph.D. and Beyond

Future work:

1. Increase ADDMC's accuracy and speed:
   - Arbitrary-precision model counting
   - Multi-core computing

   Neither feature supported by the currently used ADD library: CUDD [Somenzi, 2015]
   Both features supported by another library: Sylvan [van Dijk and van de Pol, 2015]
2. Build and combine clusters better for #SAT: tree decompositions of Gaifman graphs
   (Known to work for #P-hard problem of tensor-network contraction [Dudek et al., 2019a])
3. Try efficient data structures beyond ADDs:
   - Affine ADDs (AADDs) [Sanner and McAllester, 2005]
     Represent additive and multiplicative functions compactly
   - AND/OR multi-valued decision diagrams (AOMDDs) [Mateescu et al., 2008]
     Compile graphical models to answer queries in polynomial-time
4. Apply this framework (efficient data structure & dynamic programming & early projection) to probabilistic reasoning – e.g., most likely explanation – directly (no reduction to #SAT)

References I

R Iris Bahar, Erica A Frohm, Charles M Gaona, Gary D Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. *Form Method Syst Des*, 10(2-3):171–206, 1997.

Dirk Beyer. Binary Decision Diagram. `https://en.wikipedia.org/w/index.php?title=Binary_decision_diagram&oldid=915269665`, 2019.

Fabrice Bouquet. *Gestion de la dynamicité et énumération d'impliquants premiers: une approche fondée sur les diagrammes de décision binaire*. PhD thesis, Aix-Marseille 1, 1999.

Randal E Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE TC*, 35(8), 1986.

Supratik Chakraborty, Dror Fried, Kuldeep S Meel, and Moshe Y Vardi. From weighted to unweighted model counting. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

References II

Supratik Chakraborty, Kuldeep S Meel, and Moshe Y Vardi. Algorithmic improvements in approximate counting for probabilistic inference: from linear to logarithmic SAT calls. Technical report, Rice University, 2016.

Stephen A Cook. The complexity of theorem-proving procedures. In *ACM symposium on Theory of computing*, 1971.

Adnan Darwiche. New advances in compiling CNF into decomposable negation normal form. In *ECAI*, pages 328–332, 2004.

Rina Dechter. Bucket elimination: a unifying framework for reasoning. *AI*, 113(1-2):41–85, 1999.

Jeffrey M Dudek, Leonardo Dueñas-Osorio, and Moshe Y Vardi. Efficient contraction of large tensor networks for weighted model counting through graph decompositions. *arXiv preprint arXiv:1908.04381*, 2019a.

References III

Jeffrey M. Dudek, Vu H.N. Phan, and Moshe Y. Vardi. ADDMC: exact weighted model counting with algebraic decision diagrams. *arXiv preprint arXiv:1907.05000*, 2019b.

Leonardo Duenas-Osorio, Kuldeep S Meel, Roger Paredes, and Moshe Y Vardi. Counting-based reliability estimation for power-transmission grids. In *AAAI*, 2017.

Phokion G Kolaitis and Moshe Y Vardi. Conjunctive-query containment and constraint satisfaction. *JCSS*, 61(2):302–332, 2000.

Jean-Marie Lagniez and Pierre Marquis. An improved decision-DNNF compiler. In *IJCAI*, pages 667–673, 2017.

Martin Löbbing and Ingo Wegener. The number of knight's tours equals 33,439,123,484,294 – counting with binary decision diagrams. *the electronic journal of combinatorics*, 3(1):5, 1996.

Robert Mateescu, Rina Dechter, and Radu Marinescu. AND/OR multi-valued decision diagrams for graphical models. *JAIR*, 33:465–519, 2008.

Umut Oztok and Adnan Darwiche. A top-down compiler for sentential decision diagrams. In *IJCAI*, pages 3141–3148, 2015.

G. Pan and M.Y. Vardi. Symbolic techniques in satisfiability solving. *J Autom Reason*, 35 (1-3):25–50, 2005.

Guoqiang Pan and Moshe Y Vardi. Search vs. symbolic techniques in satisfiability solving. In *SAT*, pages 235–250, 2004.

Vu Phan. Weighted model counting with algebraic decision diagrams. Master's thesis, Rice University, Houston, Texas, USA, 2019.

Donald J Rose, R Endre Tarjan, and George S Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.

Tian Sang, Fahiem Bacchus, Paul Beame, Henry A Kautz, and Toniann Pitassi. Combining component caching and clause learning for effective model counting. *SAT*, pages 20–28, 2004.

References V

Scott Sanner and David McAllester. Affine algebraic decision diagrams and their application to structured probabilistic inference. In *IJCAI*, pages 1384–1390, 2005.

Shubham Sharma, Subhajit Roy, Mate Soos, and Kuldeep S Meel. GANAK: a scalable probabilistic exact model counter. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1169–1176. AAAI Press, 2019.

Michael A Shwe, Blackford Middleton, David E Heckerman, Max Henrion, Eric J Horvitz, Harold P Lehmann, and Gregory F Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. *Methods of Information in Medicine*, 1991.

Fabio Somenzi. *CUDD: CU decision diagram package - release 3.0.0*. University of Colorado at Boulder, 2015.

Robert E Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SICOMP*, 13(3):566–579, 1984.

References VI

Marc Thurley. sharpSAT–counting models with advanced component caching and implicit BCP. In *SAT*, pages 424–429, 2006.

Leslie G Valiant. The complexity of enumeration and reliability problems. *SICOMP*, 8(3): 410–421, 1979.

Tom van Dijk and Jaco van de Pol. Sylvan: multi-core decision diagrams. In *TACAS*, pages 677–691, 2015.