Quantitative Reasoning on Hybrid Formulas with Dynamic Programming

Vu Phan's PhD thesis defense
Committee: Prof. Moshe Vardi, Prof. Devika Subramanian, Prof. Leonardo Duenas-Osorio

Rice University, Department of Computer Science

2022-07-14

- Motivation: probabilistic models quantify uncertainties in real-world applications
- Bridge: probabilistic models are reducible to Boolean formulas
- Statement: we can efficiently solve problems on Boolean formulas by partitioning

From Qualitative to Quantitative Reasoning

Boolean formula φ

- *SAT*: find a *satisfying assignment*, i.e., *model*, of φ [Coo71]
  - *Davis-Putnam-Logemann-Loveland (DPLL)* algorithm [DLL62]
  - *Conflict-driven clause learning (CDCL)* [MS96]

- *Weighted SAT*:
  - receive weights of assignments on vars (φ)
  - find a model of φ with the highest weight [SBK07]
- *Model counting*: find the number of satisfying assignments of φ [Val79]

Variable Eliminations

- Pseudo-Boolean function $f : \mathbb{B}^S \to \mathbb{R}$
- *Maximal projection* $\max_x f : \mathbb{B}^{S \setminus \{x\}} \to \mathbb{R}$

$$\left(\max_x f\right)(\tau) := \max\left(f\left(\tau \cup \{\langle x, 1\rangle\}\right), f\left(\tau \cup \{\langle x, 0\rangle\}\right)\right)$$

$$\left(\max_S f\right)(\varnothing) \in \mathbb{R}$$

- *Summative projection* $\sum_x f : \mathbb{B}^{S \setminus \{x\}} \to \mathbb{R}$

$$\left(\sum_x f\right)(\tau) := f\left(\tau \cup \{\langle x, 1\rangle\}\right) + f\left(\tau \cup \{\langle x, 0\rangle\}\right)$$

$$\left(\sum_S f\right)(\varnothing) \in \mathbb{R}$$

Quantitative Problems

- Boolean formula φ, where $\text{vars}(\varphi) = S$
- Boolean function $f = [\varphi] : \mathbb{B}^S \to \mathbb{B}$
- *Weight function* $W : \mathbb{B}^S \to \mathbb{R}_+$

Problem	Form	Notes	Complexity
Weighted SAT	$\max_S (f \cdot W)$	$\equiv$ *maximum SAT (MaxSAT), most probable explanation (MPE)*	NP-H
Model counting	$\sum_S f$	$\equiv$ probability of evidence, i.e., marginalization in Bayesian networks	#P-C
Projected counting	$\sum_X \max_Y f$	$\{X, Y\}$: partition of S	$\#\text{P}^{\text{NP}}$-C
Exist-random SAT (ERSAT)	$\max_X \sum_Y f$	$\equiv$ *maximum a posteriori (MAP)*	$\text{NP}^{\#\text{P}}$-H

Hybrid Constraints

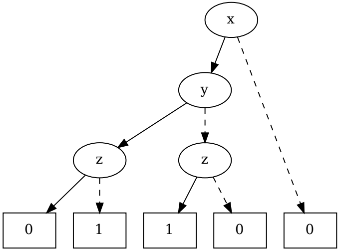
- *Conjunction normal form (CNF)* formulas are conjunctions of disjunctive clauses
- *Disjunctive clauses* (disjunctions of literals) alone can be inconvenient
  - *XOR clauses* (XORs of literals) are natural in cryptography [BKR11]
  - Performance of CNF encodings (e.g., [Tse83]) depends on solvers [Pre09]
- *XOR-CNF formulas* are conjunctions of disjunctive and XOR clauses

$$\varphi = x \wedge (x \vee \neg y) \wedge (y \oplus z) \quad \textit{factored representation}$$

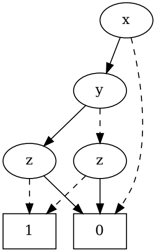
$$[\varphi] = [x] \cdot [x \vee \neg y] \cdot [y \oplus z] \quad \textit{(multiplicative) join}$$

Approaches to Solving Constraints

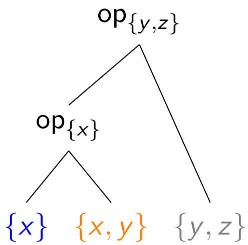
Model counting and SAT on XOR-CNF: $x \wedge (x \vee \neg y) \wedge (y \oplus z)$



Backtracking search: *binary decision tree*



Knowledge compilation: *binary decision diagrams (BDDs)* [Bry86], etc.



Dynamic programming: *project-join tree*, where

$$\mathrm{op} := \begin{cases} \sum & \text{model counting} \\ \max & \text{SAT} \end{cases}$$

Main Contribution: Versatile Framework for Quantitative Reasoning

- Dynamic programming: exploit factored representations [Pha19; DPV20a]
  - XOR-CNF formula: product of clauses
  - Assignment weight: product of literal weights
- Project-join tree T for an XOR-CNF formula φ: project out variables and conjoin clauses
  - *Planning phase*: build T from φ
  - *Execution phase*: traverse T to reason about φ

Single plan: multiple executions, one for each problem

Projection operators	Published (overlap with [Dud21, Jeff])	Archived (later submissions)
One: $\sum$, max	Model counting [DPV20b]	Weighted SAT [PV22b]
Two: $\sum$ max, max $\sum$	Projected counting [DPV21]	ERSAT [PV22a]

Progress

1. Model Counting
   - Planners
   - Executors
   - Evaluating Model Counters
2. Weighted SAT
   - Evaluating Weighted-SAT Solvers
3. Projected Counting
   - Evaluating Projected Counters
4. Exist-Random SAT (ERSAT)
   - Evaluating ERSAT Solvers

Model Counting

- Applications of model counting:
  - Analysis of information flows [KMM13]
  - Estimation of power reliability [Due+17]
- Input: a Boolean formula φ, where $S = \mathrm{vars}(\varphi)$
- Output: the number of assignments on S that satisfy φ

$$\#\varphi := \sum_{S} [\varphi] \qquad \textit{model count}$$

Early Projection: Push Variable Eliminations Inward

$$\sum_{x,y,z} f(x) \cdot g(x,y) \cdot h(y,z)$$

function over 3 variables: x, y, z

$$= \sum_{y,z} \left(\sum_{x} f(x) \cdot g(x,y) \right) \cdot h(y,z)$$

function over 2 variables: x, y

$$= \sum_{y,z} \left(\sum_{x} f(x) \cdot g(x,y) \right) \cdot h(y,z)$$

function over 1 variable: y

$$= \sum_{y,z} \left(\sum_{x} f(x) \cdot g(x,y) \right) \cdot h(y,z)$$

function over 2 variables: y, z

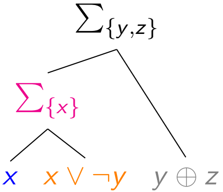
- Bayesian inference [ZP94]
- Database-query optimization [McM+04]

Project-Join Tree

XOR-CNF formula $\varphi = x \wedge (x \vee \neg y) \wedge (y \oplus z)$

Planning phase $\downarrow$

Project-join tree T



Execution phase $\downarrow$

$$f(x, y) = [x] \cdot [x \vee \neg y]$$

$$g(y) = \sum_{x} f(x, y)$$

$$h(y, z) = g(y) \cdot [y \oplus z]$$

$$\#\varphi = \sum_{y,z} h(y, z)$$

Model count $\#\varphi = 2$

Width of T $\quad \mathrm{width}(T) = 2$

Framework and Implementation

DPMC (dynamic-programming model counter) framework:

Planner: XOR-CNF formula $\mapsto$ project-join tree

Model counter (planner-executor pair)

Executor: project-join tree $\mapsto$ model count

Implementation:

Constraint-programming planner

Tree-decomposition planner

ADDMC [DPV20a]

Crossover1

Crossover2

TensorOrder [DDV19, Jeff]

Algebraic-decision-diagram executor

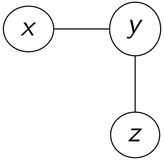
Tensor executor

Performance on single CPU cores:

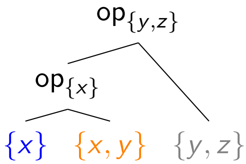
Crossover1 $>$ ADDMC $>$ TensorOrder $>$ Crossover2

Planning with Heuristics in *Constraint Programming (CP)*

XOR-CNF formula: $x \wedge (x \vee \neg y) \wedge (y \oplus z)$



Gaifman graph, i.e., *primal constraint graph*



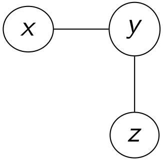
Project-join tree

CP heuristics:

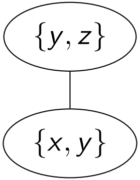
- Variable ordering:
  - *Maximum-cardinality search* [TY84]
  - *Minimum fill-in* [Dec03]
- Clause ordering:
  - *Bucket elimination* [Dec99]
  - *Bouquet's Method* [Bou99]

Planning with *Tree Decompositions (TDs)*

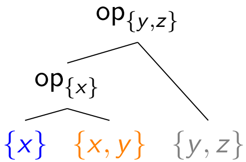
XOR-CNF formula: $x \wedge (x \vee \neg y) \wedge (y \oplus z)$



Gaifman graph



TD [RS91]



Project-join tree (Jeff)

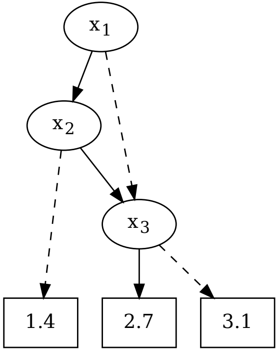
Tree decomposers from the PACE Challenge 2017 [Del+18]:

- `FlowCutter` [Str17]
- `Meiji` [Tam19]
- `htd` [AMW17]

Execution with Tensors and *Algebraic Decision Diagrams (ADDs)*

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
1	1	1	2.7
1	1	0	3.1
1	0	1	1.4
1	0	0	1.4
0	1	1	2.7
0	1	0	3.1
0	0	1	2.7
0	0	0	3.1

Tensor: *dense* representation (Jeff)

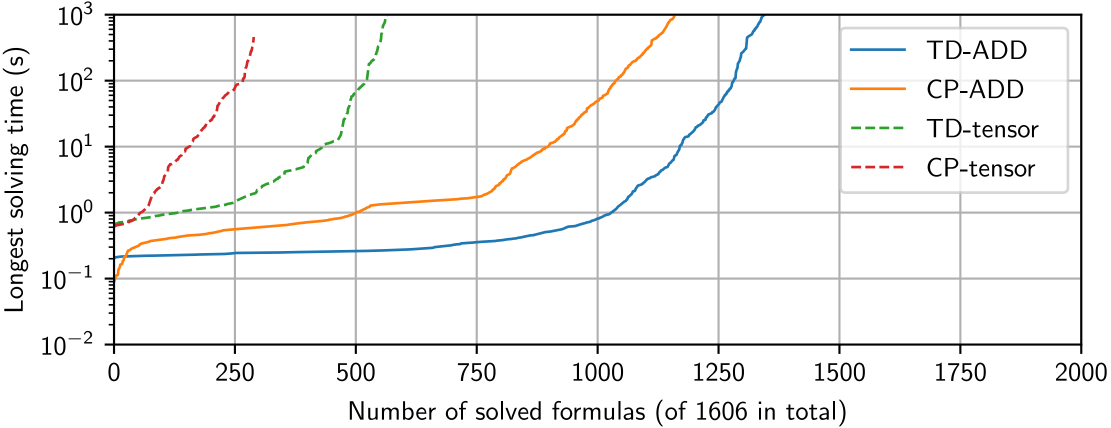


ADD: *sparse* representation

Evaluation of Model Counters

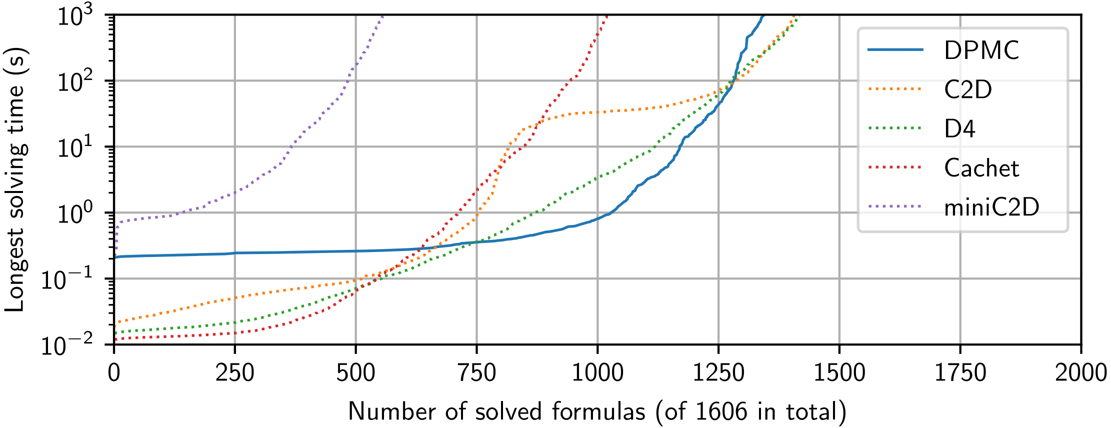
- 1606 CNF formulas:
  - 1049 benchmarks from Bayesian inference [SBK05]
  - 577 benchmarks from planning [PG09]
- State-of-the-art model counters:
  - `Cachet` [San+04]
  - `C2D` [Dar04]
  - `miniC2D` [OD15]
  - `D4` [LM17]
- NOTS cluster at Rice University:
  - CPU: single cores
  - RAM: 25 GB
  - Time: 1000 seconds per solver per benchmark

Cactus Plot: Planner-Executor Combinations



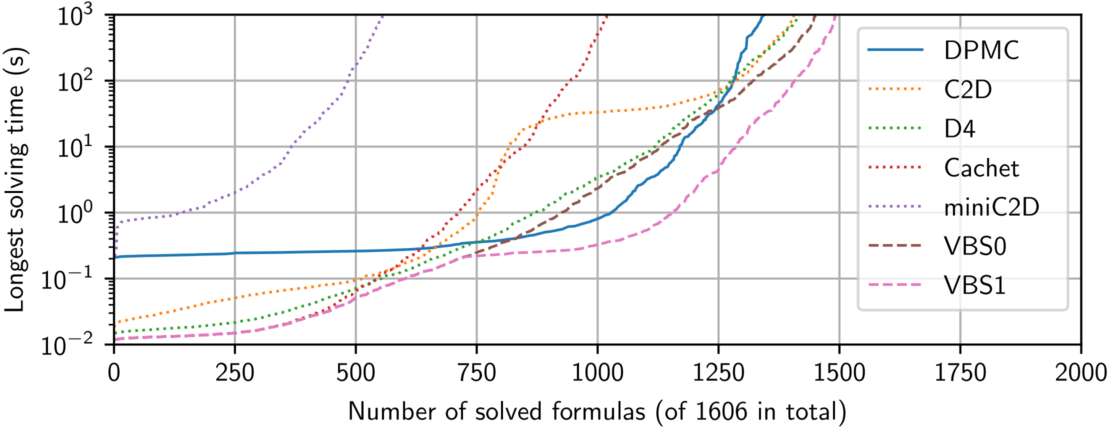
Tree-decomposition (TD) planner outperforms constraint-programming (CP) planner

Cactus Plot: Actual Solvers



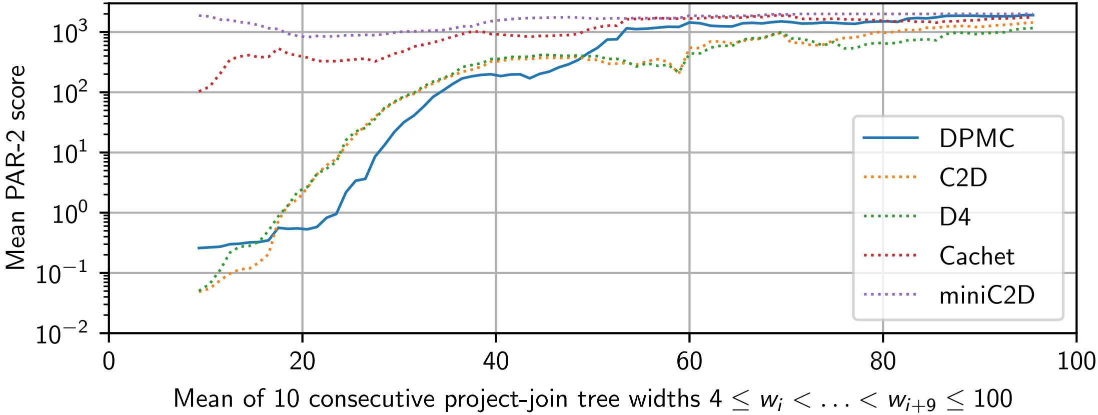
`DPMC` (TD planner and ADD executor) is competitive

Cactus Plot: Virtual Best Solvers



Virtual best solver `VBS1` (with `DPMC`) significantly outperforms `VBS0` (without `DPMC`)

Performance on 1522 Project-Join Trees for 1606 Benchmarks (94%)



$$\text{PAR2score}\,(\mathit{tool}, \varphi) := \begin{cases} \text{solving time} & \text{if } \mathit{tool} \text{ solves } \varphi \text{ within 1000 seconds} \\ 2000 & \text{otherwise} \end{cases}$$

Progress

Weighted SAT

- Applications of weighted SAT and its equivalent, MaxSAT:
  - Verification of neural networks [Sak20]
  - Synthesis of hardware exploits [Zha+20]
- Input 1: a Boolean formula φ, where $S = \text{vars}(\varphi)$
- Input 2: a weight function W mapping assignments on S to positive real-valued weights
- Output: a model of φ with the highest weight

$$\max_{S} ([\varphi] \cdot W) \qquad \text{the } \textit{maximum}$$

$$\operatorname*{argmax}_{S} ([\varphi] \cdot W) \qquad \text{a } \textit{maximizer}$$

Finding Maximizers

- Pseudo-Boolean function $f : \mathbb{B}^S \to \mathbb{R}$
- *Derivative sign* $\operatorname{dsgn}_x f : \mathbb{B}^{S \setminus \{x\}} \to \mathbb{B}^{\{x\}}$

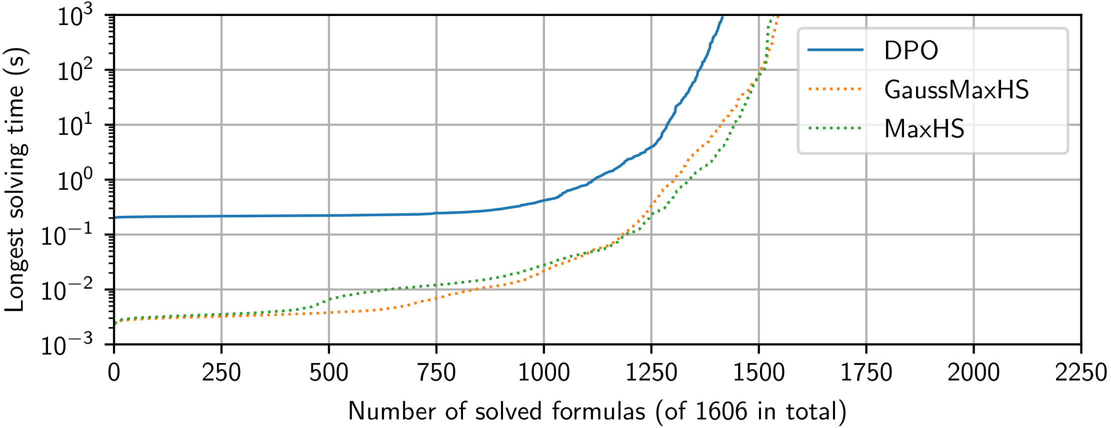
$$\left(\operatorname*{dsgn}_x f\right)(\tau) := \begin{cases} \{\langle x, 1\rangle\} & \text{if } f(\tau \cup \{\langle x, 1\rangle\}) \geq f(\tau \cup \{\langle x, 0\rangle\}) \\ \{\langle x, 0\rangle\} & \text{otherwise} \end{cases}$$

- *Iterative maximization* in pseudo-Boolean programming [CHJ90] and MaxSAT [KVZ22]:
  - Let $g = \max_x f : \mathbb{B}^{S \setminus \{x\}} \to \mathbb{R}$
  - Assume $\tau \in \mathbb{B}^{S \setminus \{x\}}$ is a maximizer of g
  - Then $\alpha = \tau \cup (\operatorname{dsgn}_x f)(\tau) \in \mathbb{B}^S$ is a maximizer of f
- `DPO` (dynamic-programming optimizer):
  - Use iterative maximization to find a maximizer
  - Adapt `DPMC` (model counting) to find the maximum

Evaluation of Weighted-SAT Solvers

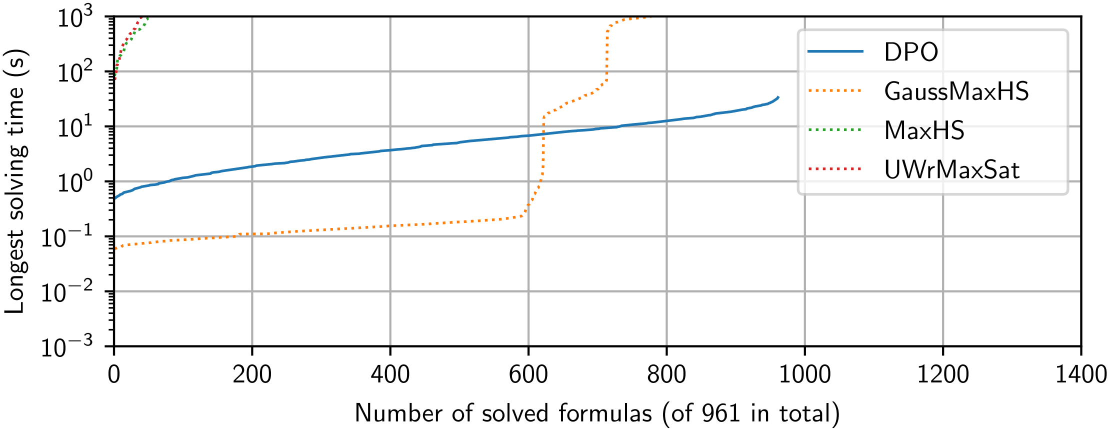
- Benchmarks:
  - 1606 application formulas in CNF from model counting
  - 961 crafted formulas in XOR-CNF from MaxSAT [KVZ22, Zhiwei]
- State-of-the-art solvers:
  - `UWrMaxSat` [Pio20]
  - `MaxHS` [DB11]
  - `GaussMaxHS` [SM21]

Application CNF Benchmarks



`UWrMaxSat` does not support floating-point weights

Crafted XOR-CNF Benchmarks



Neither `UWrMaxSat` nor `MaxHS` supports XOR

Progress

Projected Counting

- Applications of projected counting:
  - Planning [Azi+15]
  - Sampling [Gup+19]
- Input 1: a Boolean formula φ
- Input 2: a partition $\{X, Y\}$ of $\text{vars}(\varphi)$
  - X: *summative variables*
  - Y: *maximal variables*
- Output: the number of assignments τ on X such that $\varphi \mid \tau$ is satisfiable

$$\sum_X \max_Y [\varphi] \qquad \textit{projected count}$$

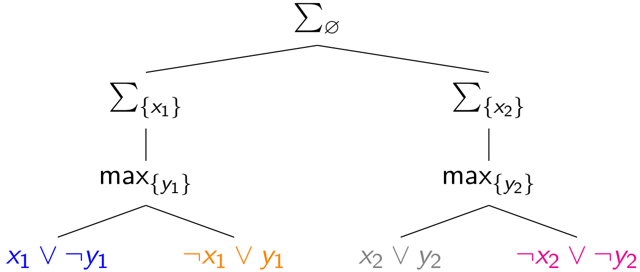
- Summative projection does not commute with maximal projection:

$$\sum_x \max_y f \neq \max_y \sum_x f \qquad \text{in general}$$

Graded Project-Join Tree

$$\sum_X \max_Y [(x_1 \vee \neg y_1) \wedge (\neg x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge (\neg x_2 \vee \neg y_2)]$$

projected count



$\langle X, Y \rangle$-*graded* project-join tree: X nodes are closer to the root than Y nodes

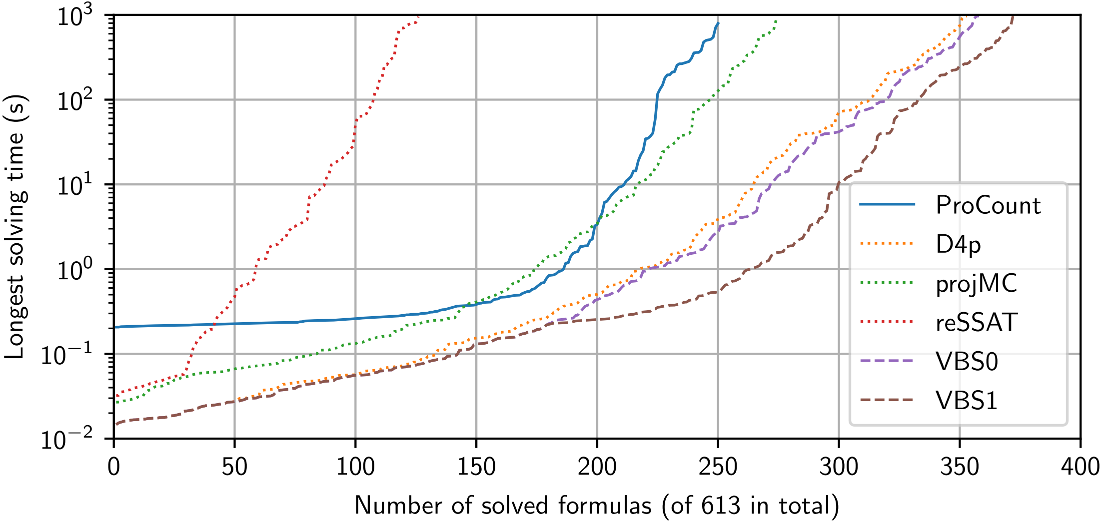
Adapting `DPMC` (Model Counting) for Projected Counting

- Reduction from graded project-join trees to ungraded project-join trees (Jeff)
- `ProCount` (projected counter):
  - Adapt planners:
    - tree decompositions (Jeff)
    - constraint programming
  - Adapt executor: ADDs

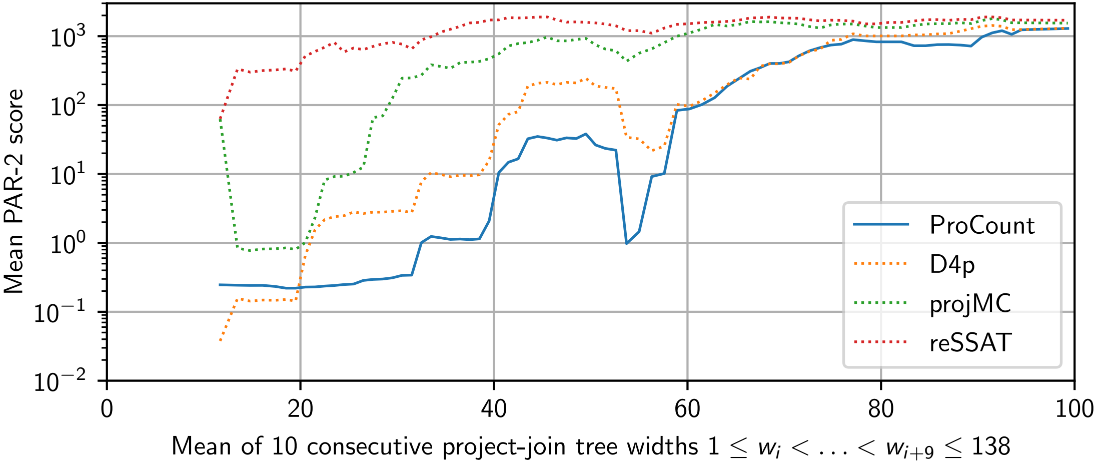
Evaluation of Projected Counters

- 613 CNF formulas:
  - 500 benchmarks from projected counting [SM19]
  - 113 benchmarks from projected sampling [Gup+19]
- State-of-the-art projected counters:
  - `reSSAT` [LWJ17]
  - `D4p` [LM17]
  - `projMC` [LM19]

Cactus Plot: Virtual Best Solvers



Performance on 291 Project-Join Trees for 613 Benchmarks (47%)



`ProCount` is fast on benchmarks whose project-join trees have low widths (below 60)

Progress

Exist-Random SAT (ERSAT)

- Applications of ERSAT:
  - Planning [ML98]
  - Fairness in machine learning [GBM21]
- Input 1: a Boolean formula φ
- Input 2: a partition $\{X, Y\}$ of $\text{vars}(\varphi)$
  - X: maximal variables
  - Y: summative variables
- Output: an assignment τ on X that maximizes the model count of $\varphi \mid \tau$

$$\max_X \sum_Y [\varphi] \qquad \text{the maximum}$$

$$\operatorname{argmax}_X \sum_Y [\varphi] \qquad \text{a maximizer}$$

- `DPER` (dynamic-programming ERSAT solver):
  - Adapt `ProCount` (projected counting) to find the maximum
  - Adapt `DPO` (weighted SAT) to find a maximizer

Evaluation of ERSAT Solvers

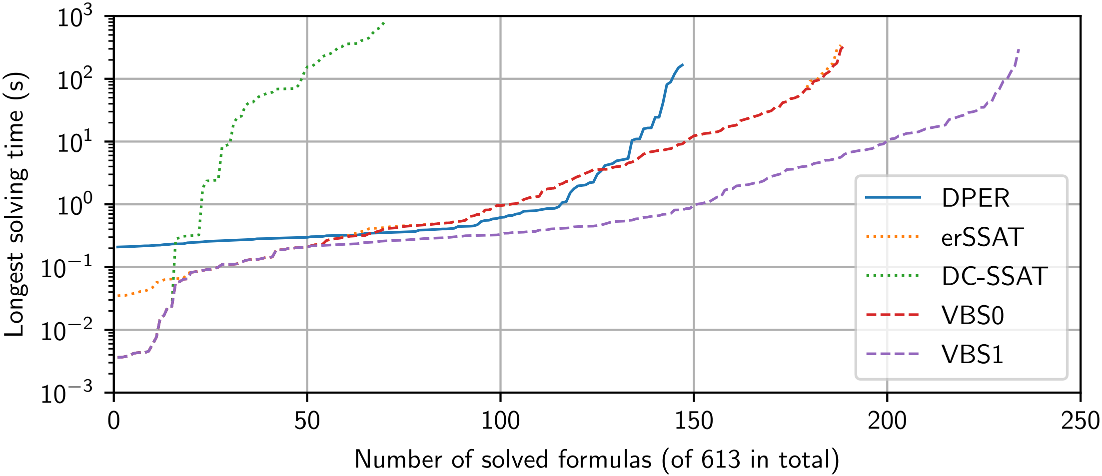
- 613 CNF formulas from projected counting:

$$\sum_{X} \max_{Y} f \qquad \text{original}$$

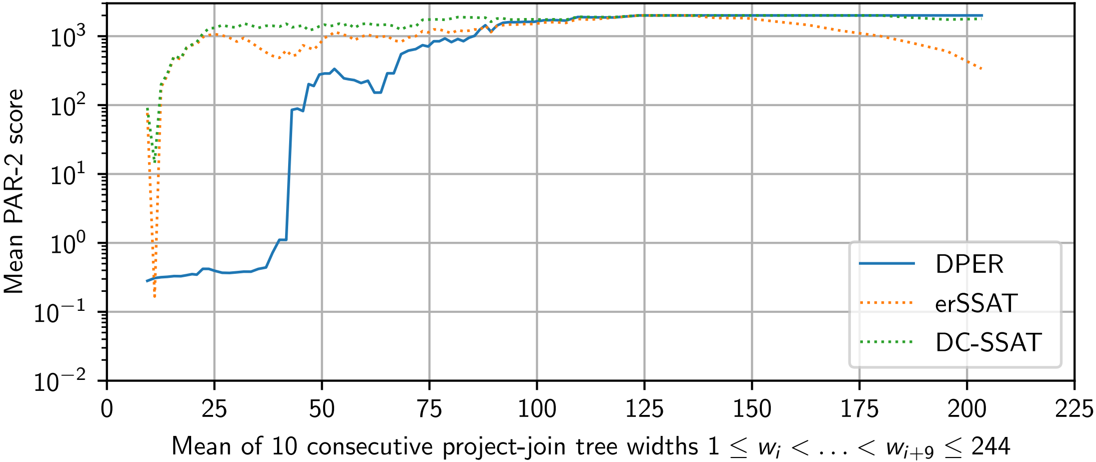
$$\max_{Y} \sum_{X} f \qquad \text{adapted for ERSAT}$$

- State-of-the-art ERSAT solvers:
  - `erSSAT` [LWJ18]
  - `DC-SSAT` [MB05]

Cactus Plot: Virtual Best Solvers

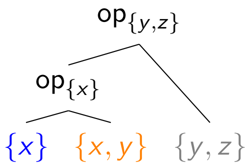


Performance on 204 Project-Join Trees for 613 Benchmarks (33%)



DPER is fast on benchmarks whose project-join trees have low widths (below 80)

Summary: Versatile Framework for Quantitative Reasoning



Project-join tree for $x \wedge (x \vee \neg y) \wedge (y \oplus z)$, where $\mathrm{op} := \begin{cases} \sum & \text{for summative variables} \\ \max & \text{for maximal variables} \end{cases}$

- Planning phase:
  - Tree decompositions (TDs)
  - Constraint programming
- Execution phase:
  - Tensors
  - Algebraic decision diagrams (ADDs)

Single plan with multiple executions:

- Model counting
- Weighted SAT
- Projected counting
- Exist-random SAT (ERSAT)

Unifying Current Work and Proposing Future Work

Current work: two projection operators and two join operators

Project-join tree	Projections	Join	Problem	Students
Ungraded	$\sum_S$	$\prod_{c \in \varphi} [c]$	Model counting [DPV20b]	Vu, Jeff
	$\max_S$	$\prod_{c \in \varphi} [c]$	Weighted SAT [PV22b]	Vu
	$\max_S$	$\sum_{c \in \varphi} [c]$	MaxSAT [KVZ22]	Zhiwei
Graded	$\sum_X \max_Y$	$\prod_{c \in \varphi} [c]$	Projected counting [DPV21]	Vu, Jeff
	$\max_X \sum_Y$	$\prod_{c \in \varphi} [c]$	ERSAT [PV22a]	Vu
	$\min_X \max_Y$	$\sum_{c \in \varphi} [c]$	*MinMaxSAT* [KVZ22]	Zhiwei

Future work:

- Hybrid inputs [KVZ22]:
  - Cardinality constraints
  - Pseudo-Boolean constraints
- Executors:
  - Multi-core ADDs [DP15]
  - Database engines [Fic+20]

References I

[AMW17] Michael Abseher, Nysret Musliu, and Stefan Woltran. "htd—a free, open-source framework for (customized) tree decompositions and beyond". In: *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. 2017, pp. 376–386.

[Azi+15] Rehan Abdul Aziz et al. "Projected model counting". In: *International Conference on Theory and Applications of Satisfiability Testing*. 2015, pp. 121–137.

[BKR11] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. "Biclique cryptanalysis of the full AES". In: *International Conference on the Theory and Application of Cryptology and Information Security*. 2011, pp. 344–371.

[Bou99] 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.

References II

[Bry86] Randal E Bryant. “Graph-based algorithms for Boolean function manipulation”. In: *IEEE Transactions on Computers* 100.8 (1986), pp. 677–691.

[CHJ90] Yves Crama, Pierre Hansen, and Brigitte Jaumard. “The basic algorithm for pseudo-Boolean programming revisited”. In: *Discrete Applied Mathematics* 29.2–3 (1990), pp. 171–185.

[Coo71] Stephen A Cook. “The complexity of theorem-proving procedures”. In: *ACM Symposium on Theory of Computing*. 1971, pp. 151–158.

[Dar04] Adnan Darwiche. “New advances in compiling CNF to decomposable negation normal form”. In: *European Conference on Artificial Intelligence*. 2004, pp. 318–322.

[DB11] Jessica Davies and Fahiem Bacchus. “Solving MaxSAT by solving a sequence of simpler SAT instances”. In: *International Conference on Principles and Practice of Constraint Programming*. 2011, pp. 225–239.

[DDV19] Jeffrey M Dudek, Leonardo Duenas-Osorio, and Moshe Y Vardi. "Efficient contraction of large tensor networks for weighted model counting through graph decompositions". In: *arXiv preprint arXiv:1908.04381* (2019).

[Dec03] Rina Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[Dec99] Rina Dechter. "Bucket elimination: a unifying framework for reasoning". In: *Artificial Intelligence* 113.1–2 (1999), pp. 41–85.

[Del+18] Holger Dell et al. "The PACE 2017 parameterized algorithms and computational experiments challenge: the second iteration". In: *International Symposium on Parameterized and Exact Computation*. 2018.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. "A machine program for theorem proving". In: *Communications of the ACM* 5.7 (1962), pp. 394–397.

[DP15] Tom van Dijk and Jaco van de Pol. "Sylvan: multi-core decision diagrams". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 2015, pp. 677–691.

[DPV20a] Jeffrey M Dudek, Vu H N Phan, and Moshe Y Vardi. "ADDMC: weighted model counting with algebraic decision diagrams". In: *AAAI Conference on Artificial Intelligence*. Vol. 34. 2020, pp. 1468–1476.

[DPV20b] Jeffrey M Dudek, Vu H N Phan, and Moshe Y Vardi. "DPMC: weighted model counting by dynamic programming on project-join trees". In: *International Conference on Principles and Practice of Constraint Programming*. 2020, pp. 211–230.

References V

[DPV21] Jeffrey M Dudek, Vu H N Phan, and Moshe Y Vardi. "ProCount: weighted projected model counting with graded project-join trees". In: *International Conference on Theory and Applications of Satisfiability Testing*. 2021, pp. 152–170.

[Dud21] Jeffrey M Dudek. "Planning and execution for discrete integration". PhD thesis. Rice University, 2021.

[Due+17] Leonardo Duenas-Osorio et al. "Counting-based reliability estimation for power-transmission grids". In: *AAAI Conference on Artificial Intelligence*. Vol. 31. 2017.

[Fic+20] Johannes K Fichte et al. "Exploiting database management systems and treewidth for counting". In: *International Symposium on Practical Aspects of Declarative Languages*. 2020, pp. 151–167.

References VI

[GBM21] Bishwamittra Ghosh, Debabrota Basu, and Kuldeep S Meel. "Justicia: a stochastic SAT approach to formally verify fairness". In: *AAAI Conference on Artificial Intelligence*. Vol. 35. 2021, pp. 7554–7563.

[Gup+19] Rahul Gupta et al. "WAPS: weighted and projected sampling". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 2019, pp. 59–76.

[KMM13] Vladimir Klebanov, Norbert Manthey, and Christian Muise. "SAT-based analysis and quantification of information flow in programs". In: *International Conference on Quantitative Evaluation of Systems*. 2013, pp. 177–192.

[KVZ22] Anastasios Kyrillidis, Moshe Y Vardi, and Zhiwei Zhang. "DPMS: an ADD-based symbolic approach for generalized MaxSAT solving". In: *arXiv preprint arXiv:2205.03747* (2022).

[LM17] Jean Marie Lagniez and Pierre Marquis. “An improved decision-DNNF compiler”. In: *IJCAI*. Vol. 17. 2017, pp. 667–673.

[LM19] Jean Marie Lagniez and Pierre Marquis. “A recursive algorithm for projected model counting”. In: *AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 1536–1543.

[LWJ17] Nian Ze Lee, Yen Shi Wang, and Jie Hong R Jiang. “Solving stochastic Boolean satisfiability under random-exist quantification”. In: *IJCAI*. 2017, pp. 688–694.

[LWJ18] Nian Ze Lee, Yen Shi Wang, and Jie Hong R Jiang. “Solving exist-random quantified stochastic Boolean satisfiability via clause selection”. In: *IJCAI*. 2018, pp. 1339–1345.

[MB05] Stephen M Majercik and Byron Boots. “DC-SSAT: a divide-and-conquer approach to solving stochastic satisfiability problems efficiently”. In: *AAAI*. 2005, pp. 416–422.

References VIII

[McM+04] Benjamin J McMahan et al. "Projection pushing revisited". In: *International Conference on Extending Database Technology*. 2004, pp. 441–458.

[ML98] Stephen M Majercik and Michael L Littman. "MAXPLAN: a new approach to probabilistic planning". In: *International Conference on Artificial Intelligence Planning Systems*. 1998, pp. 86–93.

[MS96] J P Marques Silva and K A Sakallah. "GRASP—a new search algorithm for satisfiability". In: *International Conference on Computer Aided Design*. 1996, pp. 220–227.

[OD15] Umut Oztok and Adnan Darwiche. "A top-down compiler for sentential decision diagrams". In: *International Joint Conference on Artificial Intelligence*. 2015.

[PG09] Hector Palacios and Hector Geffner. "Compiling uncertainty away in conformant planning problems with bounded width". In: *Journal of Artificial Intelligence Research* 35 (2009), pp. 623–675.

[Pha19] Vu Hoang Nguyen Phan. "Weighted model counting with algebraic decision diagrams". MA thesis. Rice University, 2019.

[Pio20] Marek Piotrow. "UwrMaxSAT: efficient solver for MaxSAT and pseudo-Boolean problems". In: *International Conference on Tools with Artificial Intelligence*. 2020, pp. 132–136.

[Pre09] Steven D Prestwich. "CNF encodings". In: *Handbook of satisfiability* 185 (2009), pp. 75–97.

[PV22a] Vu H N Phan and Moshe Y Vardi. "DPER: dynamic programming for exist-random stochastic SAT". In: *arXiv preprint arXiv:2205.09826* (2022).

[PV22b] Vu H N Phan and Moshe Y Vardi. "DPO: dynamic-programming optimization on hybrid constraints". In: *arXiv preprint arXiv:2205.08632* (2022).

[RS91] Neil Robertson and Paul D Seymour. "Graph minors. X. Obstructions to tree-decomposition". In: *Journal of Combinatorial Theory, Series B* 52.2 (1991), pp. 153–190.

[Sak20] Masahiro Sakai. "BNN verification dataset for MaxSAT Evaluation 2020". In: *MaxSAT Evaluation 2020* (2020), p. 37.

[San+04] Tian Sang et al. "Combining component caching and clause learning for effective model counting". In: *SAT* 4 (2004).

[SBK05] Tian Sang, Paul Beame, and Henry A Kautz. "Performing Bayesian inference by weighted model counting". In: *AAAI*. Vol. 5. 2005, pp. 475–481.

[SBK07] Tian Sang, Paul Beame, and Henry A Kautz. "A dynamic approach for MPE and weighted MaxSAT". In: *IJCAI*. 2007, pp. 173–179.

References XI

[SM19] Mate Soos and Kuldeep S Meel. "BIRD: engineering an efficient CNF-XOR SAT solver and its applications to approximate model counting". In: *AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 1592–1599.

[SM21] Mate Soos and Kuldeep S Meel. "Gaussian elimination meets maximum satisfiability". In: *International Conference on Principles of Knowledge Representation and Reasoning*. Vol. 18. 2021, pp. 581–587.

[Str17] Ben Strasser. "Computing tree decompositions with FlowCutter: PACE 2017 submission". In: *arXiv preprint arXiv:1709.08949* (2017).

[Tam19] Hisao Tamaki. "Positive-instance driven dynamic programming for treewidth". In: *Journal of Combinatorial Optimization* 37.4 (2019), pp. 1283–1311.

[Tse83] Grigori S Tseitin. "On the complexity of derivation in propositional calculus". In: *Automation of reasoning*. Springer, 1983, pp. 466–483.

References XII

[TY84] Robert E Tarjan and Mihalis Yannakakis. "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs". In: *SIAM Journal on computing* 13.3 (1984), pp. 566–579.

[Val79] Leslie G Valiant. "The complexity of enumeration and reliability problems". In: *SIAM Journal on Computing* 8.3 (1979), pp. 410–421.

[Zha+20] Changjian Zhang et al. "Automated synthesis of minimal hardware exploits with Checkmate and MaxSAT solver". In: *MaxSAT Evaluation 2020* (2020), p. 49.

[ZP94] Nevin L Zhang and David Poole. "A simple approach to Bayesian network computations". In: *Canadian Conference on Artificial Intelligence*. 1994.