DPMC: Weighted Model Counting by Dynamic Programming on Project-Join Trees [1, CP 2020]

Jeffrey Dudek, **Vu Phan** (`https://VuPhan314.github.io`), Moshe Vardi

Rice University

Model Counting Workshop, 6 July 2021

Unifying dynamic-programming framework for exact literal-weighted model counting

- Faster than `cachet`, `miniC2D`, `c2d`, `d4` on 584 of 1976 benchmarks (30%)

Applications

Problems solved via reduction to model counting [2]:

- Probability of disease given symptom (e.g., COVID | fever) [3, 4]
- Reliability of electricity grid (e.g., power outage in Texas during winter storm) [5]

Propositional Model Counting

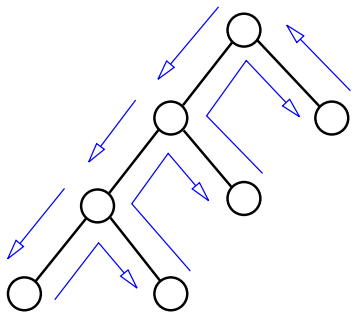
Conjunctive normal form (CNF) formula: $\varphi = x_1 \wedge (x_1 \vee \neg x_2) \wedge (\neg x_2 \vee \neg x_3)$

x_1	x_2	x_3	$\varphi(x_1, x_2, x_3)$	Model?
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	0	
1	0	0	1	Yes
1	0	1	1	Yes
1	1	0	1	Yes
1	1	1	0	

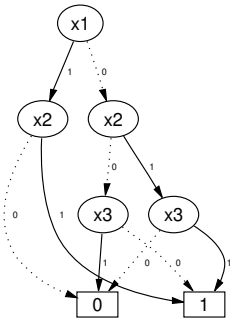
Model count: $\#\varphi = 3$ (if each literal weight is 1.0)

Related Work: Three Approaches to Model Counting

Search: explore solution space with backtracking [6] (figure from [7])



Knowledge compilation: use tractable circuit [8–10] (binary decision diagram [11], figure from [12])



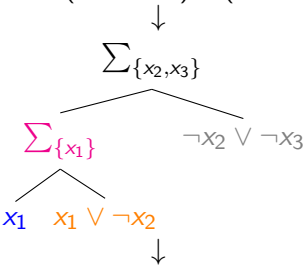
Dynamic programming: solve overlapping subproblems (e.g., `ADDMC` [13], `TensorOrder` [14])

- Contribution: unifying framework `DPMC` (dynamic-programming model counter) [1]

Model-Counting Framework: Planning Phase and Execution Phase

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

Planning phase



Project-join tree

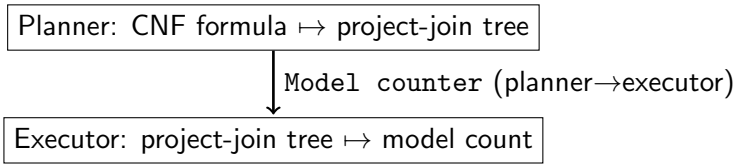
Execution phase

$$f(x_1, x_2) = x_1 \cdot (x_1 \vee \neg x_2)$$
$$g(x_2) = \sum_{x_1} f(x_1, x_2)$$
$$= f(0, x_2) + f(1, x_2)$$
$$r(x_2, x_3) = g(x_2) \cdot (\neg x_2 \vee \neg x_3)$$
$$\#\varphi = \sum_{x_2, x_3} r(x_2, x_3)$$

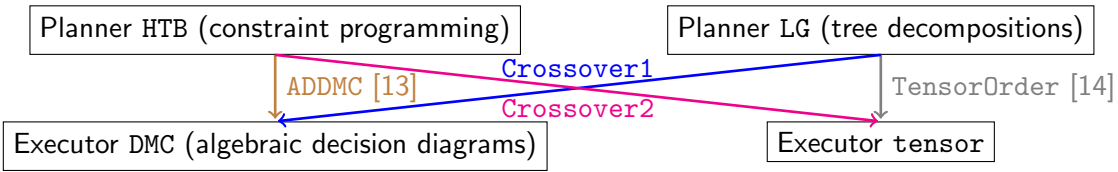
Model count $\#\varphi = 3$

Model-Counting Framework and Implementation

Framework:



Implementation:



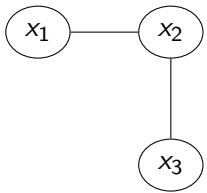
Performance:

Crossover1 > ADDMC > TensorOrder > Crossover2

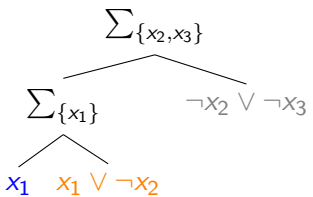
Planner HTB (Heuristic Tree Builder)

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

Primal graph



Project-join tree (**one-shot**)



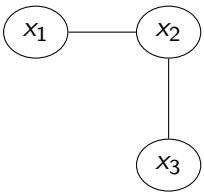
Planner HTB constructs project-join tree with constraint-programming heuristics

- Variable ordering: maximal-cardinality search [15], minimal fill-in [16]
- Clause ordering: bucket elimination [17], Bouquet's Method [18]

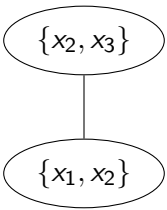
Planner LG (Line Graph)

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

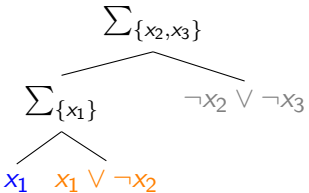
Primal graph



Tree decompositions



Project-join trees (**anytime**)



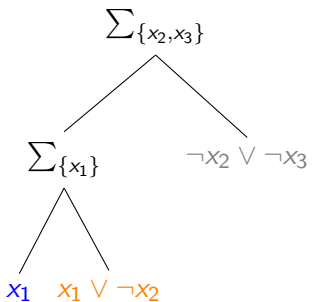
Planner LG constructs project-join trees with tree decompositions [19]

- Winning tools from heuristic-treewidth track of PACE Challenge 2017 [20]: `Tamaki` [21], `FlowCutter` [22], `htd` [23]

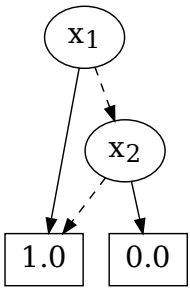
Executor DMC (Diagram Model Counter)

Node in project-join tree represents pseudo-Boolean function

Project-join tree



Algebraic decision diagram (ADD)



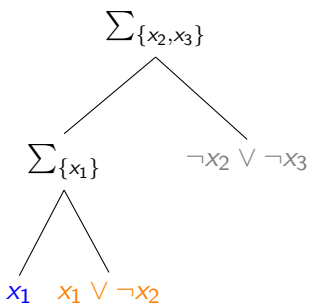
Executor DMC valuates project-join tree using **sparse** ADDs [24]

- ADD package CUDD [25]

Executor `tensor`

Node in project-join tree represents pseudo-Boolean function

Project-join tree



Tensor (multi-dimensional array)

- 0-dimension (scalar)
- 1-dimension (list)
- 2-dimension (matrix)
- ...

Executor `tensor` valuates project-join tree using **dense** tensors

- Tensor package `NumPy` [26]

Empirical Evaluation: 1976 Benchmarks (No Preprocessing)

Bayesian inference: 1080 CNF formulas [4]

- Deterministic Quick Medical Reference
- Grid Networks
- Plan Recognition

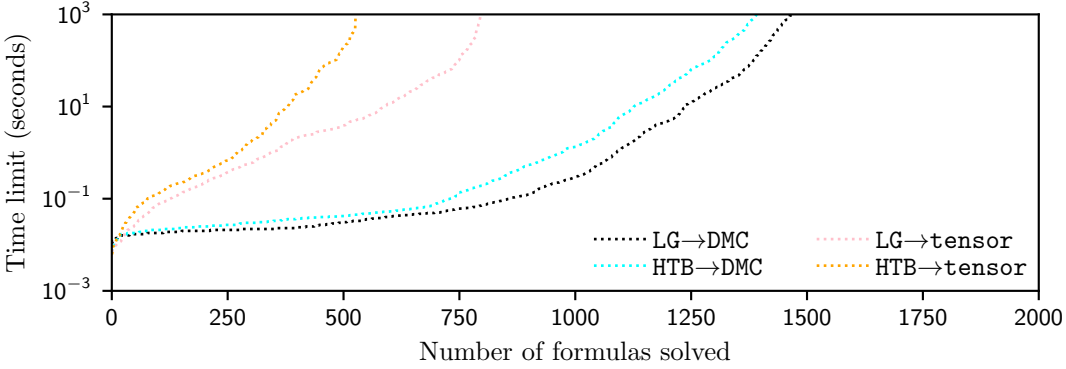
Other domains: 896 CNF formulas [27–30]

- Planning
- Bounded Model Checking
- Circuit
- Configuration
- Quantitative Information Flow
- Scheduling

Linux cluster at Rice University:

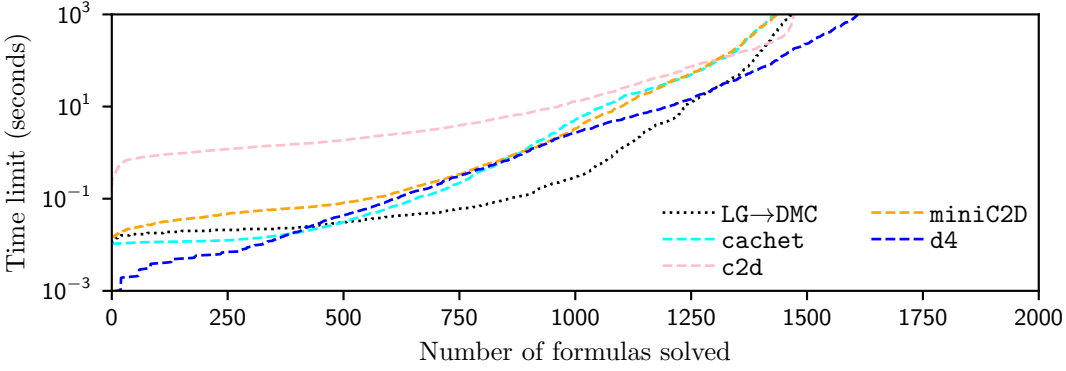
- CPU: 2.60GHz Xeon E5-2650 v2 (single-core solvers)
- RAM: 30GB

Empirical Evaluation: Planners and Executors



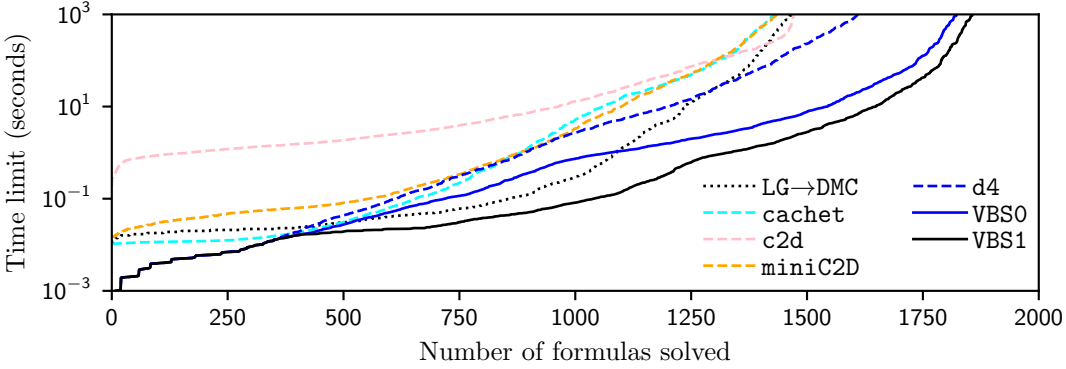
- `LG` (*anytime* tree decompositions) outperforms `HTB` (*one-shot* constraint programming)
- `DMC` (*sparse* ADDs) outperforms `tensor` (*dense* tensors)

Empirical Evaluation: Model Counters



- Exact weighted model counters: `cachet` [6], `c2d` [8], `miniC2D` [9], `d4` [10]
- `LG→DMC` fastest on 471 formulas (24%); `DPMC` (all combinations) fastest on 584 (30%)

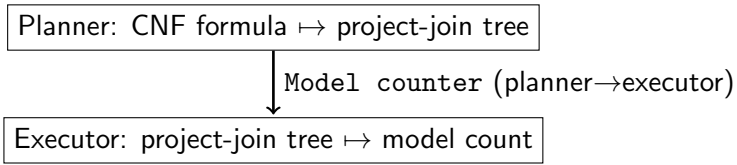
Empirical Evaluation: Virtual Best Solvers



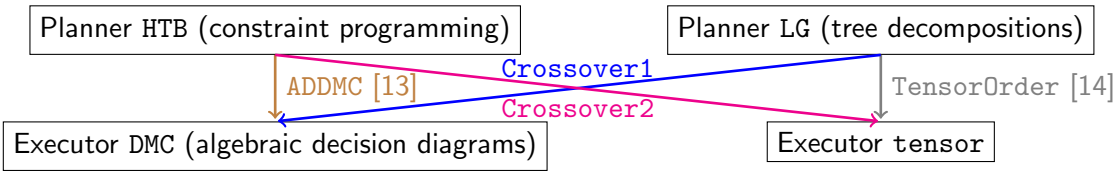
- Virtual best solver (VBS): simulation of running actual solvers in parallel
- `VBS1` (with `LG→DMC`) is faster than `VBS0` (without `LG→DMC`)

Summary: Model Counting by Dynamic Programming

Framework:



Implementation:



Model Counting Competition 2021:

- Unprojected tracks: DPMC [1, CP 2020] (https://github.com/vardigroup/DPMC)
- Projected track: ProCount [31, SAT 2021] (talk at 3:30 pm CEST on Thursday)

References I

[1] Jeffrey M Dudek, Vu HN 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*. Springer. 2020, pp. 211–230.

[2] Leslie G Valiant. "The complexity of enumeration and reliability problems". In: *SICOMP* 8.3 (1979), pp. 410–421. DOI: 10.1137/0208032.

[3] Michael A Shwe et al. "Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base". In: *Methods of Information in Medicine* (1991). URL: https://europepmc.org/article/med/1762578.

[4] Tian Sang, Paul Beame, and Henry A Kautz. "Performing Bayesian inference by weighted model counting". In: *AAAI*. Vol. 1. AAAI Press. 2005, pp. 475–482. URL: https://dl.acm.org/doi/10.5555/1619332.1619409.

References II

[5] Leonardo Duenas-Osorio et al. "Counting-based reliability estimation for power-transmission grids". In: *AAAI*. 2017. URL: https://dl.acm.org/doi/10.5555/3298023.3298219.

[6] Tian Sang et al. "Combining component caching and clause learning for effective model counting". In: *SAT* 4 (2004), pp. 20–28. URL: http://www.satisfiability.org/SAT04/accepted/65.html.

[7] Wikipedia contributors. *DPLL algorithm — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-March-2021]. 2020. URL: https://en.wikipedia.org/w/index.php?title=DPLL_algorithm&oldid=994404116.

[8] Adnan Darwiche. "New advances in compiling CNF to decomposable negation normal form". In: *ECAI*. 2004, pp. 318–322. URL: https://dl.acm.org/doi/10.5555/3000001.3000069.

References III

[9] Umut Oztok and Adnan Darwiche. "A top-down compiler for sentential decision diagrams". In: *IJCAI*. 2015. URL: https://dl.acm.org/doi/10.5555/2832581.2832687.

[10] Jean-Marie Lagniez and Pierre Marquis. "An improved decision-DNNF compiler". In: *IJCAI*. 2017, pp. 667–673. DOI: 10.24963/ijcai.2017/93.

[11] Randal E Bryant. "Graph-based algorithms for Boolean function manipulation". In: *IEEE TC* 100.8 (1986), pp. 677–691. DOI: 10.1109/TC.1986.1676819.

[12] Wikipedia contributors. *Binary decision diagram — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-March-2021]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Binary_decision_diagram&oldid=1011828301.

[13] Jeffrey M. Dudek, Vu H. N. Phan, and Moshe Y. Vardi. "ADDMC: weighted model counting with algebraic decision diagrams". In: *AAAI*. Vol. 34. 2020, pp. 1468–1476. DOI: 10.1609/aaai.v34i02.5505.

References IV

[14] Jeffrey M Dudek, Leonardo Dueñas-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). URL: https://arxiv.org/abs/1908.04381.

[15] 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: *SICOMP* 13.3 (1984), pp. 566–579. DOI: 10.1137/0213035.

[16] Rina Dechter. *Constraint processing*. Morgan Kaufmann, 2003. DOI: 10.1016/B978-1-55860-890-0.X5000-2.

[17] Rina Dechter. "Bucket elimination: a unifying framework for reasoning". In: *AIJ* 113.1-2 (1999), pp. 41–85. DOI: 10.1016/S0004-3702(99)00059-4.

[18] 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. URL: https://www.theses.fr/1999AIX11011.

References V

[19] Neil Robertson and Paul D Seymour. "Graph minors. X. Obstructions to tree-decomposition". In: *J Combinatorial Theory B* 52.2 (1991), pp. 153–190. DOI: 10.1016/0095-8956(91)90061-N.

[20] Holger Dell et al. "The PACE 2017 parameterized algorithms and computational experiments challenge: The second iteration". In: *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2018.

[21] Hisao Tamaki. "Positive-instance-driven dynamic programming for treewidth". In: *J Comb Optim* 37.4 (2019), pp. 1283–1311. DOI: 10.1007/s10878-018-0353-z.

[22] Ben Strasser. "Computing tree decompositions with FlowCutter: PACE 2017 submission". In: *arXiv preprint arXiv:1709.08949* (2017). URL: https://arxiv.org/abs/1709.08949.

References VI

[23] Michael Abseher, Nysret Musliu, and Stefan Woltran. "htd–a free, open-source framework for (customized) tree decompositions and beyond". In: *CPAIOR*. Springer. 2017, pp. 376–386. DOI: 10.1007/978-3-319-59776-8_30.

[24] R Iris Bahar et al. "Algebraic decision diagrams and their applications". In: *Form Method Syst Des* 10.2-3 (1997), pp. 171–206. DOI: 10.1023/A:1008699807402.

[25] Fabio Somenzi. *CUDD: CU decision diagram package–release 3.0.0*. University of Colorado at Boulder. 2015. URL: https://github.com/ivmai/cudd.

[26] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006. URL: https://dl.acm.org/doi/book/10.5555/2886196.

[27] Edmund Clarke et al. "Bounded model checking using satisfiability solving". In: *Form Method Syst Des* 19.1 (2001), pp. 7–34. DOI: 10.1023/A:1011276507260.

References VII

[28] Carsten Sinz, Andreas Kaiser, and Wolfgang Küchlin. "Formal methods for the validation of automotive product configuration data". In: *AI EDAM* 17.1 (2003), pp. 75–97. DOI: 10.1017/S0890060403171065.

[29] Hector Palacios and Hector Geffner. "Compiling uncertainty away in conformant planning problems with bounded width". In: *JAIR* 35 (2009), pp. 623–675. URL: https://dl.acm.org/doi/10.5555/1641503.1641518.

[30] Vladimir Klebanov, Norbert Manthey, and Christian Muise. "SAT-based analysis and quantification of information flow in programs". In: *QEST*. 2013, pp. 177–192. DOI: 10.1007/978-3-642-40196-1_16.

[31] Jeffrey M Dudek, Vu HN 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*. Springer. 2021, pp. 152–170.