Generating Random WMC Instances

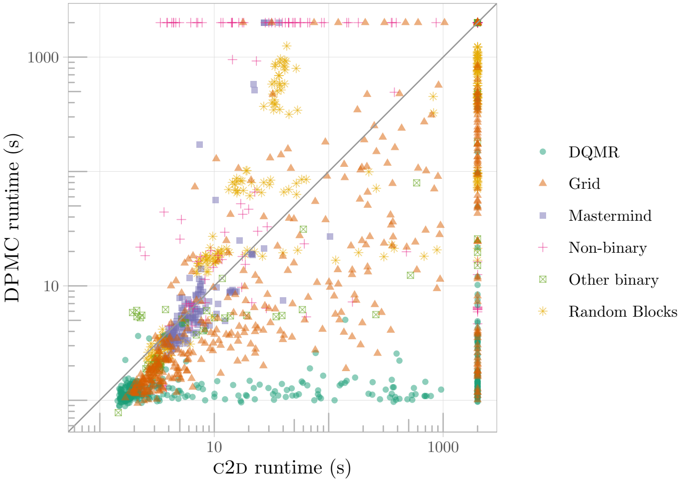
An Empirical Analysis with Varying Primal Treewidth

Paulius Dilkas

National University of Singapore

30th May 2024

Which Algorithm Is Better? It Depends on the Data



The runtime data is from Dilkas and Belle (2021): various Bayesian networks encoded using the approach by Darwiche (2002)

The Problem: Weighted Model Counting (WMC)

- A generalisation of propositional model counting (#SAT)
- Applications:
  - graphical models
  - probabilistic programming
  - neuro-symbolic AI
- WMC algorithms use:
  - dynamic programming
  - knowledge compilation
  - SAT solvers

Example

$w(x) = 0.3,\ w(\neg x) = 0.7,$
$w(y) = 0.2,\ w(\neg y) = 0.8$

$\mathrm{WMC}(x \lor y) = w(x)w(y) + w(x)w(\neg y) + w(\neg x)w(y) = 0.44$

(Some of the) WMC Algorithms

- Cachet (Sang et al. 2004)
  - a SAT solver with clause learning and component caching
- c2d (Darwiche 2004)
  - knowledge compilation to d-DNNF
- d4 (Lagniez and Marquis 2017)
  - knowledge compilation to decision-DNNF
- miniC2D (Oztok and Darwiche 2015)
  - knowledge compilation to decision-SDDs
- DPMC (Dudek, Phan and Vardi 2020)
  - dynamic programming with ADDs and tree decomposition based planning

Frequently Asked Questions

Frequently Asked Questions

Q: Why isn't SharpSAT-TD included in the experiments?

A: Because I started setting up these experiments eight days after the SharpSAT-TD paper came out

Frequently Asked Questions

Q: Why isn't SharpSAT-TD included in the experiments?

A: Because I started setting up these experiments eight days after the SharpSAT-TD paper came out

Q: Why isn't GANAK included in the experiments?

A: Because it's easy to argue that probabilistic algorithms are out of scope (and I had lots of algorithms already)

Frequently Asked Questions

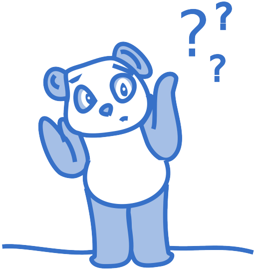
Q: Why isn't SharpSAT-TD included in the experiments?

A: Because I started setting up these experiments eight days after the SharpSAT-TD paper came out

Q: Why isn't GANAK included in the experiments?

A: Because it's easy to argue that probabilistic algorithms are out of scope (and I had lots of algorithms already)

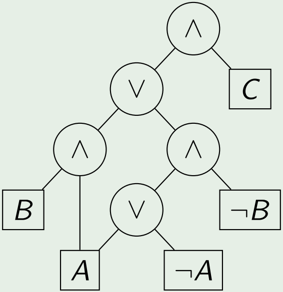
Q: Why am I giving a talk about this now?

A: 

Knowledge Compilation: d-DNNF and Decision-DNNF

Example

$C \land (A \lor \neg B)$ in d-DNNF/decision-DNNF:

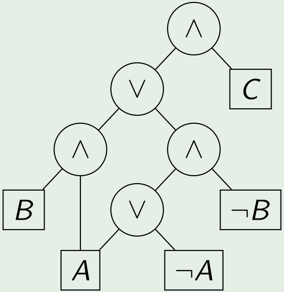


Knowledge Compilation: d-DNNF and Decision-DNNF

Negation normal form: $\neg$ is only applied to variables

Example

$C \wedge (A \vee \neg B)$ in d-DNNF/decision-DNNF:



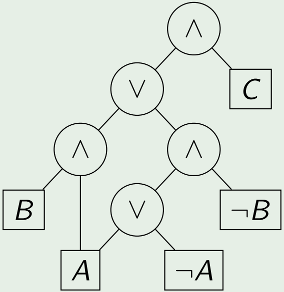
Knowledge Compilation: d-DNNF and Decision-DNNF

Negation normal form: $\neg$ is only applied to variables

Decomposability: for every $\alpha \land \beta$, $\text{Vars}(\alpha) \cap \text{Vars}(\beta) = \emptyset$

Example

$C \land (A \lor \neg B)$ in d-DNNF/decision-DNNF:



Knowledge Compilation: d-DNNF and Decision-DNNF

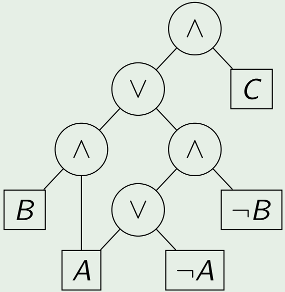
Negation normal form: $\neg$ is only applied to variables

Decomposability: for every $\alpha \land \beta$, $\mathrm{Vars}(\alpha) \cap \mathrm{Vars}(\beta) = \emptyset$

Determinism: for every $\alpha \lor \beta$, $\alpha \land \beta \equiv \bot$

Example

$C \land (A \lor \neg B)$ in d-DNNF/decision-DNNF:



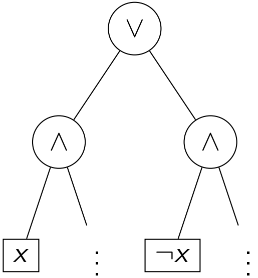
Knowledge Compilation: d-DNNF and Decision-DNNF

Negation normal form: $\neg$ is only applied to variables

Decomposability: for every $\alpha \wedge \beta$, $\mathsf{Vars}(\alpha) \cap \mathsf{Vars}(\beta) = \emptyset$

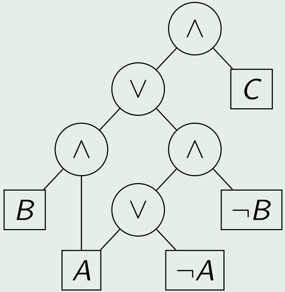
Determinism: for every $\alpha \vee \beta$, $\alpha \wedge \beta \equiv \bot$

Decision: all disjunctions are of the form



Example

$C \wedge (A \vee \neg B)$ in d-DNNF/decision-DNNF:



Tree Decompositions and Primal Treewidth

A formula in CNF:

$$\phi = \begin{aligned}[t] &(x_1 \lor x_2) \land (x_2 \lor x_3 \lor x_4) \land (x_1 \lor x_4) \land (x_3 \lor x_5) \land (x_4 \lor x_5 \lor x_6) \\ &\land (x_5 \lor x_6 \lor x_7) \land (x_6 \lor x_7) \end{aligned}$$

Tree Decompositions and Primal Treewidth

A formula in CNF:

$$\phi = \begin{aligned}(x_1 \vee x_2) \wedge (x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_4) \wedge (x_3 \vee x_5) \wedge (x_4 \vee x_5 \vee x_6)\\ \wedge (x_5 \vee x_6 \vee x_7) \wedge (x_6 \vee x_7)\end{aligned}$$

The primal graph of ϕ is:

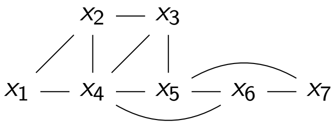
$$\begin{tikzcd}
 & x_2 \arrow[r, dash] \arrow[dl, dash] \arrow[d, dash] & x_3 \arrow[dl, dash] \arrow[d, dash] & & \\
x_1 \arrow[r, dash] & x_4 \arrow[r, dash] \arrow[rr, dash, bend right] & x_5 \arrow[r, dash] \arrow[rr, dash, bend left] & x_6 \arrow[r, dash] & x_7
\end{tikzcd}$$

Tree Decompositions and Primal Treewidth

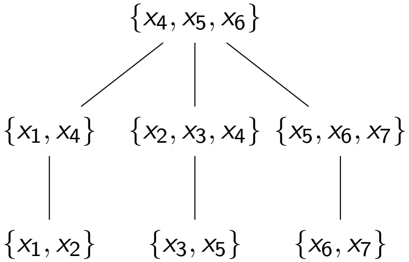
A formula in CNF:

$$\phi = \begin{array}{r}(x_1 \lor x_2) \land (x_2 \lor x_3 \lor x_4) \land (x_1 \lor x_4) \land (x_3 \lor x_5) \land (x_4 \lor x_5 \lor x_6) \\ \land (x_5 \lor x_6 \lor x_7) \land (x_6 \lor x_7)\end{array}$$

The primal graph of ϕ is:



Its minimum-width tree decomposition:

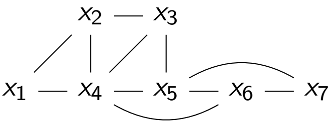


Tree Decompositions and Primal Treewidth

A formula in CNF:

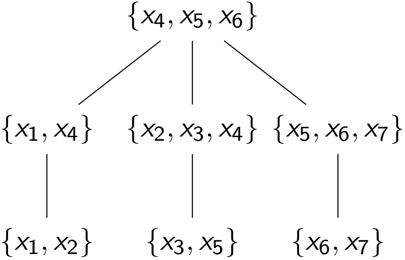
$$\phi = \begin{array}{r} (x_1 \vee x_2) \wedge (x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_4) \wedge (x_3 \vee x_5) \wedge (x_4 \vee x_5 \vee x_6) \\ \wedge (x_5 \vee x_6 \vee x_7) \wedge (x_6 \vee x_7) \end{array}$$

The primal graph of ϕ is:



$\therefore$ the primal treewidth of ϕ is 2

Its minimum-width tree decomposition:



Formally...

Definition (Robertson and Seymour 1984)

A tree decomposition of a graph G is a pair (T, χ), where T is a tree and $\chi\colon \mathcal{V}(T) \to 2^{\mathcal{V}(G)}$ is a labelling function, with the following properties:

- $\bigcup_{t \in \mathcal{V}(T)} \chi(t) = \mathcal{V}(G)$;
- for every edge $e \in \mathcal{E}(G)$, there is $t \in \mathcal{V}(T)$ s.t. e has both endpoints in $\chi(t)$;
- for all $t, t', t'' \in \mathcal{V}(T)$, if t' is on the path between t and t'', then $\chi(t) \cap \chi(t'') \subseteq \chi(t')$.

The width of tree decomposition (T, χ) is $\max_{t \in \mathcal{V}(T)} |\chi(t)| - 1$. The treewidth of graph G is the smallest w such that G has a tree decomposition of width w.

The Parameterised Complexity of WMC Algorithms

Let n be the number of variables and m be the number of clauses.

- Component caching (used in CACHET) is $2^{\mathcal{O}(w)}n^{\mathcal{O}(1)}$, where w is the branchwidth of the underlying hypergraph (Bacchus, Dalmao and Pitassi 2009)
  - Branchwidth is within a constant factor of primal treewidth
- C2D is based on an algorithm, which is $\mathcal{O}(2^w mw)$, where w is at most primal treewidth (Darwiche 2001; Darwiche 2004)
- DPMC can be shown to be $\mathcal{O}(4^w mn)$, where w is an upper bound on primal treewidth

Early History of Random SAT

- Goldberg, Purdom Jr. and Brown (1982) show that (simplified) Davis-Putnam procedures run in polynomial time on average on the following model:
  - Fix the numbers of variables and clauses
  - Let $p \in (0, 0.5)$
  - For each clause C and variable x:
    - Add x to C w.p. p
    - Or add $\neg x$ to C w.p. p
    - Or do nothing w.p. $1 - 2p$

Early History of Random SAT

- Goldberg, Purdom Jr. and Brown (1982) show that (simplified) Davis-Putnam procedures run in polynomial time on average on the following model:
  - Fix the numbers of variables and clauses
  - Let $p \in (0, 0.5)$
  - For each clause C and variable x:
    - Add x to C w.p. p
    - Or add $\neg x$ to C w.p. p
    - Or do nothing w.p. $1 - 2p$
- Franco and Paull (1983) were the first to propose a 'reasonable' random k-CNF model:
  1. Fix the numbers of variables, clauses, and clause width
  2. Sample each clause independently as a subset of all possible literals
  3. Reject clauses that have x and $\neg x$ for some variable x
  - They show that the Davis-Putnam procedure requires exponential time w.p. one

Phase Transitions

- Phase transitions for decision problems have been studied both experimentally and theoretically for a long time (see, e.g., Cheeseman, Kanefsky and Taylor (1991))

Phase Transitions

- Phase transitions for decision problems have been studied both experimentally and theoretically for a long time (see, e.g., Cheeseman, Kanefsky and Taylor (1991))
- Phase transitions are characterised by a parameter k such that:
  - For low values of k, the problem is easy
  - For high values of k, the problem is easy again
  - Average values of k is where the problem becomes hard

Phase Transitions

- Phase transitions for decision problems have been studied both experimentally and theoretically for a long time (see, e.g., Cheeseman, Kanefsky and Taylor (1991))
- Phase transitions are characterised by a parameter k such that:
  - For low values of k, the problem is easy
  - For high values of k, the problem is easy again
  - Average values of k is where the problem becomes hard
- For SAT-based problems, (clause) density is the standard parameter
  - i.e., the number of clauses divided by the number of variables
  - (usually parameterised by clause width)

Phase Transitions

- Phase transitions for decision problems have been studied both experimentally and theoretically for a long time (see, e.g., Cheeseman, Kanefsky and Taylor (1991))
- Phase transitions are characterised by a parameter k such that:
  - For low values of k, the problem is easy
  - For high values of k, the problem is easy again
  - Average values of k is where the problem becomes hard
- For SAT-based problems, (clause) density is the standard parameter
  - i.e., the number of clauses divided by the number of variables
  - (usually parameterised by clause width)
- The most comprehensive (experimental) study of phase transitions for SAT algorithms is by Coarfa et al. (2003)
  - They show that the transition from polynomial to exponential time depends on the solver

Generating Random WMC Instances: The Algorithm

$\phi \leftarrow$ empty CNF formula;
$G \leftarrow$ empty graph;
for $i \leftarrow 1$ **to** m **do**
    $X \leftarrow \emptyset$;
    for $j \leftarrow 1$ **to** k **do**
        $x \leftarrow$ `newVariable(`X`, `G`)`;
        $\mathcal{V}(G) \leftarrow \mathcal{V}(G) \cup \{ x \}$;
        $\mathcal{E}(G) \leftarrow \mathcal{E}(G) \cup \{ \{ x, y \} \mid y \in X \}$;
        $X \leftarrow X \cup \{ x \}$;
    $\phi \leftarrow \phi \cup \{ \{ l \leftsquigarrow \mathcal{U}\{ x, \neg x \} \mid x \in X \} \}$;

- the number of clauses
- clause width
- a function to pick a variable
- a (fair) coin flip

How to Pick a Variable

Parameter $\rho \in [0, 1]$ biases the probability distribution towards adding variables that would introduce fewer new edges.

Function `newVariable`(*set of variables* X, *primal graph* G)**:**

$N \leftarrow \{ e \in \mathcal{E}(G) \mid |e \cap X| = 1 \}$;

if $N = \emptyset$ **then return** $x \leftarrow\!\!\sim \mathcal{U}(\{ x_1, x_2, \ldots, x_n \} \setminus X)$;

return

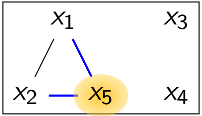
$$x \leftarrow\!\!\sim \left(\{ x_1, x_2, \ldots, x_n \} \setminus X, y \mapsto \frac{1-\rho}{n-|X|} + \rho \frac{|\{ z \in X \mid \{ y, z \} \in \mathcal{E}(G) \}|}{|N|} \right);$$

From Random SAT to Random WMC

We introduce parameter $\rho \in [0, 1]$ that biases the probability distribution towards adding variables that would introduce fewer new edges to the primal graph.

Example partially-filled formula:
$(\neg x_5 \lor x_2 \lor x_1) \land (x_5 \lor ? \lor ?)$

Its primal graph:



The probability distribution for the next variable

Base probability of each variable being chosen:

$$\frac{1-\rho}{4}.$$

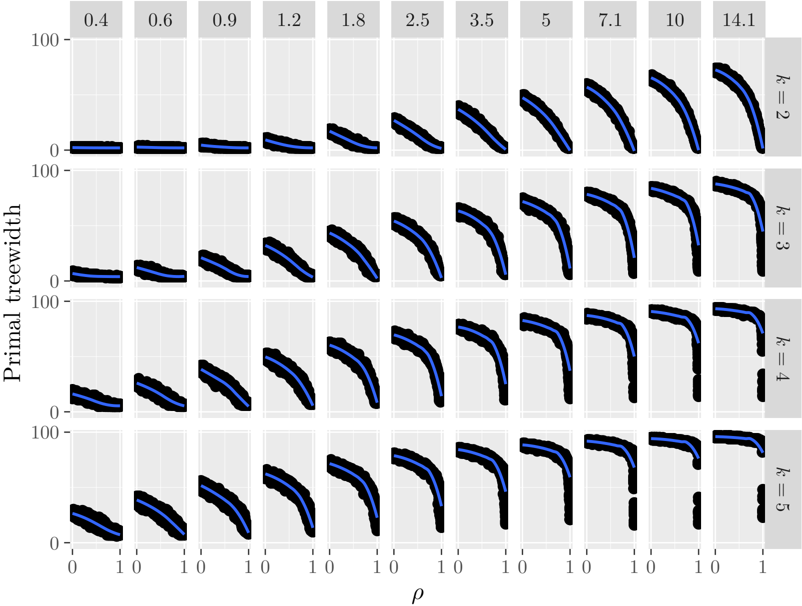
Both x_1 and x_2 get a bonus probability of $\rho/2$ for each being the endpoint of one out of the two neighbourhood edges.

The Relationship Between ρ and Primal Treewidth

Experiment 1 (Validation)

- Set the number of variables to 100
- Consider a geometric sequence of 11 densities from 0.4 to 14.1
- Let ρ range from zero to one in steps of 0.01
- Generate ten 2-, 3-, 4-, and 5-CNF formulas

The Relationship Between ρ and Primal Treewidth



Density, Primal Treewidth, and Runtime

Experiment 2 (The Main One)

- Set the number of variables to 70
- Consider densities ranging from 1 to 4.3 in steps of 0.3
- Let ρ range from 0 to 0.5 in steps of 0.01
- Generate one 3-CNF formula

Peak Hardness w.r.t. Density

Let μ denote the density, i.e., the number of clauses divided by the number of variables.

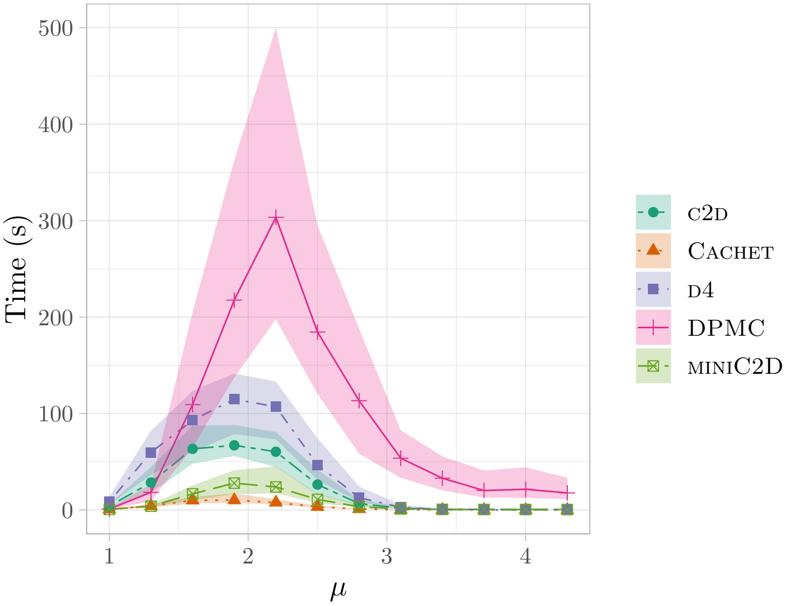
- CACHET is known to peak at $\mu = 1.8$ (Sang et al. 2004)
- Bayardo Jr. and Pehoushek (2000) show some #SAT algorithms to peak at $\mu = 1.2$ and $\mu = 1.9$

Peak Hardness w.r.t. Density

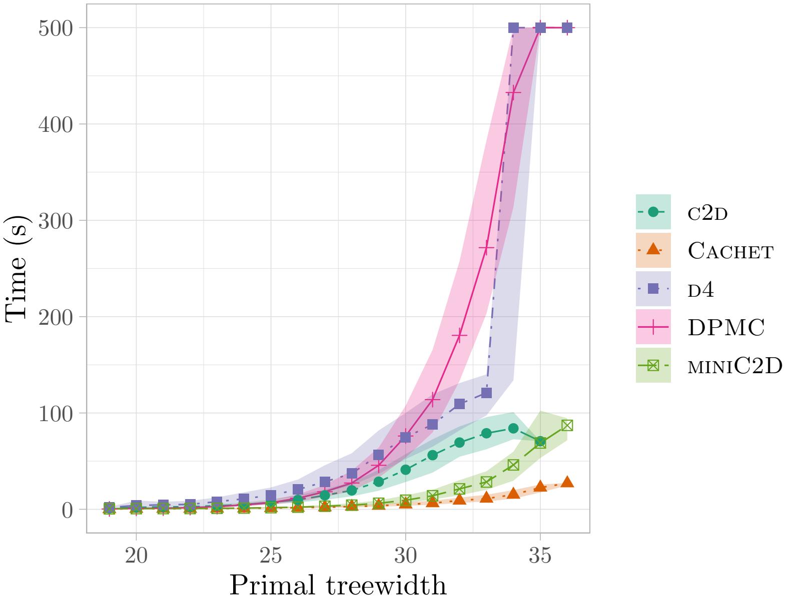
Let μ denote the density, i.e., the number of clauses divided by the number of variables.

- CACHET is known to peak at $\mu = 1.8$ (Sang et al. 2004)
- Bayardo Jr. and Pehoushek (2000) show some #SAT algorithms to peak at $\mu = 1.2$ and $\mu = 1.9$
- In our experiments:
  - DPMC peaks at $\mu = 2.2$
  - all other algorithms peak at $\mu = 1.9$

Peak Hardness w.r.t. Density (when $\rho = 0$)



Hardness w.r.t. Primal Treewidth (when $\mu = 1.9$)



Is The Relationship Exponential: Two Approaches

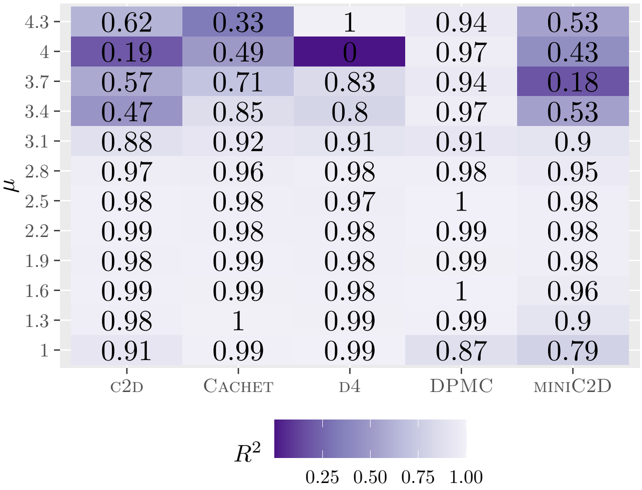
Linear Regression

We fit the model $\ln t \sim \alpha w + \beta$, i.e., $t \sim e^{\beta}(e^{\alpha})^{w}$, where t is runtime, and w is primal treewidth

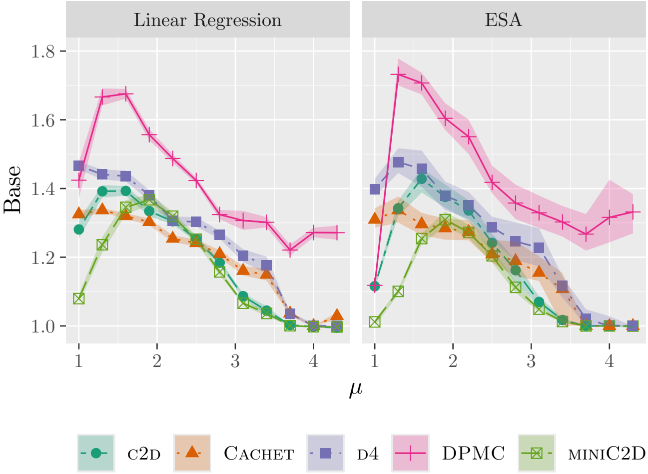
Empirical Scaling Analyzer (ESA) v2 (Pushak and Hoos 2020)

1. Prepare a list of hypotheses about scalability, e.g.:
   exponential: $t \sim \alpha\beta^{w}$,
   polynomial: $t \sim \alpha w^{\beta}$
2. Use $30\,\%$ of the data with the largest values of w for testing
3. For each hypothesis, ESA produces:
   - estimates of parameter values,
   - support loss, and
   - challenge loss

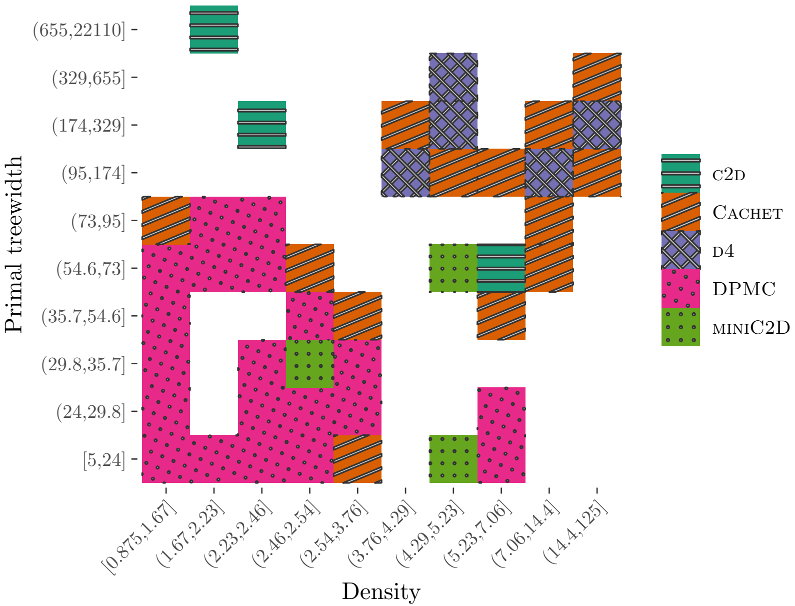
How Well Does Linear Regression Explain the Data?



The Base of the Exponential

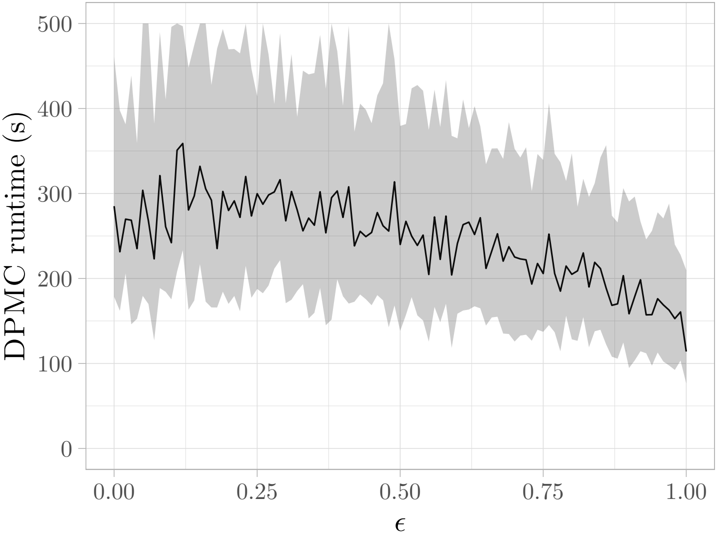


Does Real Data Confirm Our Observations?

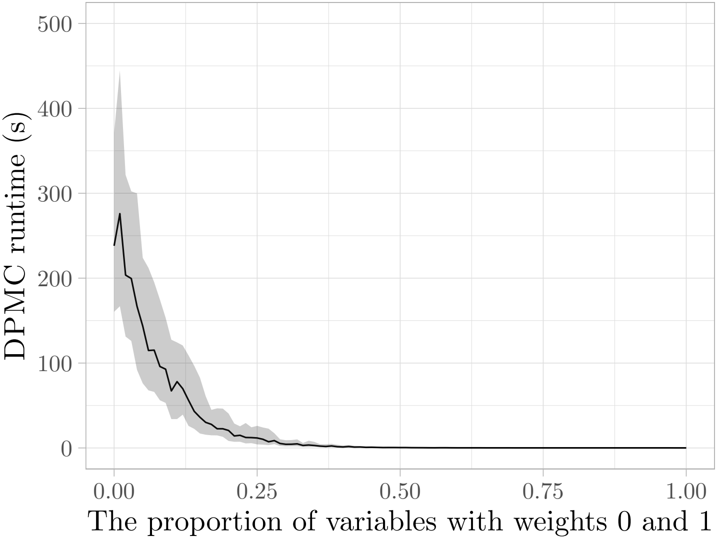


Bonus 1: How DPMC Reacts to Redundancy in Weights

Let ϵ be the proportion of variables x s.t. $w(x) = w(\neg x) = 0.5$



Bonus 2: 0/1 Weights Make Counting Easy



Summary

- This work introduced a random model for WMC instances with a parameter that indirectly controls primal treewidth
- Observations:
  - All algorithms scale exponentially w.r.t. primal treewidth
  - The running time of DPMC:
    - peaks at a higher density
    - and scales worse w.r.t. primal treewidth
- Future work:
  - A theoretical relationship between ρ and primal treewidth
  - Non-k-CNF instances
  - Algorithm portfolios for WMC

Future Work: (Per-Instance) Algorithm Selection

Definition (Bischl et al. 2016)

Given a set $\mathcal{I}$ of problem instances, a space of algorithms $\mathcal{A}$, and a performance measure $m\colon \mathcal{I} \times \mathcal{A} \to \mathbb{R}$, the algorithm selection problem is to find a mapping $s\colon \mathcal{I} \to \mathcal{A}$ that optimises $\mathbb{E}[m(i, s(i))]$.

Future Work: (Per-Instance) Algorithm Selection

Definition (Bischl et al. 2016)

Given a set $\mathcal{I}$ of problem instances, a space of algorithms $\mathcal{A}$, and a performance measure $m\colon \mathcal{I} \times \mathcal{A} \to \mathbb{R}$, the algorithm selection problem is to find a mapping $s\colon \mathcal{I} \to \mathcal{A}$ that optimises $\mathbb{E}[m(i, s(i))]$.

ϕ

Future Work: (Per-Instance) Algorithm Selection

Definition (Bischl et al. 2016)

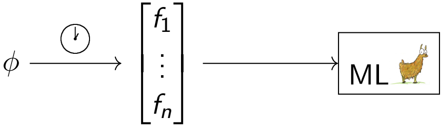
Given a set $\mathcal{I}$ of problem instances, a space of algorithms $\mathcal{A}$, and a performance measure $m\colon \mathcal{I} \times \mathcal{A} \to \mathbb{R}$, the algorithm selection problem is to find a mapping $s\colon \mathcal{I} \to \mathcal{A}$ that optimises $\mathbb{E}[m(i, s(i))]$.

$$\phi \longrightarrow \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Future Work: (Per-Instance) Algorithm Selection

Definition (Bischl et al. 2016)

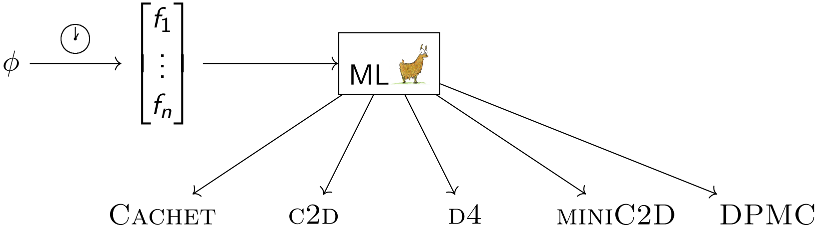
Given a set $\mathcal{I}$ of problem instances, a space of algorithms $\mathcal{A}$, and a performance measure $m\colon \mathcal{I} \times \mathcal{A} \to \mathbb{R}$, the algorithm selection problem is to find a mapping $s\colon \mathcal{I} \to \mathcal{A}$ that optimises $\mathbb{E}[m(i, s(i))]$.



Future Work: (Per-Instance) Algorithm Selection

Definition (Bischl et al. 2016)

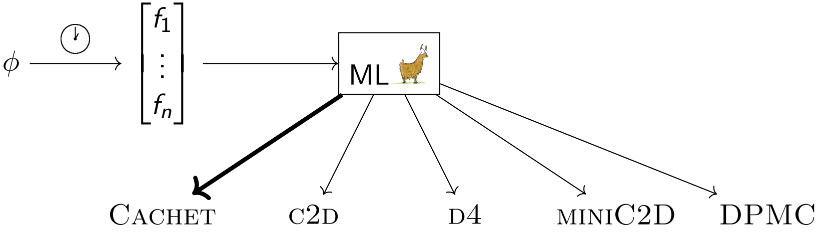
Given a set $\mathcal{I}$ of problem instances, a space of algorithms $\mathcal{A}$, and a performance measure $m \colon \mathcal{I} \times \mathcal{A} \to \mathbb{R}$, the algorithm selection problem is to find a mapping $s \colon \mathcal{I} \to \mathcal{A}$ that optimises $\mathbb{E}[m(i, s(i))]$.



Future Work: (Per-Instance) Algorithm Selection

Definition (Bischl et al. 2016)

Given a set $\mathcal{I}$ of problem instances, a space of algorithms $\mathcal{A}$, and a performance measure $m\colon \mathcal{I} \times \mathcal{A} \to \mathbb{R}$, the algorithm selection problem is to find a mapping $s\colon \mathcal{I} \to \mathcal{A}$ that optimises $\mathbb{E}[m(i, s(i))]$.



Future Work: (Per-Instance) Algorithm Selection

Definition (Bischl et al. 2016)

Given a set $\mathcal{I}$ of problem instances, a space of algorithms $\mathcal{A}$, and a performance measure $m\colon \mathcal{I} \times \mathcal{A} \to \mathbb{R}$, the algorithm selection problem is to find a mapping $s\colon \mathcal{I} \to \mathcal{A}$ that optimises $\mathbb{E}[m(i, s(i))]$.