Capita Selecta AI - Probabilistic Programming
Inference for SRL

Thierry Deruyttere (r0660485) & Armin Halilovic (r0679689)

April 1, 2018

Probabilistic Inference Using Weighted Model Counting

1.1 PGM to CNF

1.1.1 ENC 1

Our ENC1 encoding for the Cancer Bayesian network can be found in appendix 4.1. The CNF in dimacs format can be found under report/encodings/cancer/.

1.1.2 ENC 2

Our ENC2 encoding for the Cancer Bayesian network can be found in appendix 4.2. The CNF in dimacs format can be found under report/encodings/cancer/.

1.2 SRL to CNF

1.2.1 Encoding of Monty Hall as CNF

An encoding of problog programs can be generated by our program as follows:

```
python3 scripts/inference.py --problog files/problog/
    monty_hall.pl
```

The CNF will be shown using the program's predicates. A version of the CNF in dimacs format will be shown as well. See `README.MD` for more information.

Our CNF encoding for the given Monty Hall ProbLog program is:

$\wedge(open\_door(2) \vee prize(2) \vee prize(3) \vee \neg p\_open\_door(2)\_0)$
$\wedge(open\_door(2) \vee prize(2) \vee \neg prize(3))$
$\wedge(\neg open\_door(2) \vee \neg prize(2) \vee \neg prize(2))$
$\wedge(\neg open\_door(2) \vee \neg prize(2) \vee prize(3))$
$\wedge(\neg open\_door(2) \vee \neg prize(3) \vee \neg prize(2))$
$\wedge(\neg open\_door(2) \vee \neg prize(3) \vee prize(3))$
$\wedge(\neg open\_door(2) \vee p\_open\_door(2)\_0 \vee \neg prize(2))$
$\wedge(\neg open\_door(2) \vee p\_open\_door(2)\_0 \vee prize(3))$
$\wedge(open\_door(3) \vee prize(2) \vee prize(3) \vee \neg p\_open\_door(3)\_0)$
$\wedge(open\_door(3) \vee prize(3) \vee \neg prize(2))$
$\wedge(\neg open\_door(3) \vee \neg prize(2) \vee \neg prize(3))$
$\wedge(\neg open\_door(3) \vee \neg prize(2) \vee prize(2))$
$\wedge(\neg open\_door(3) \vee \neg prize(3) \vee \neg prize(3))$
$\wedge(\neg open\_door(3) \vee \neg prize(3) \vee prize(2))$
$\wedge(\neg open\_door(3) \vee p\_open\_door(3)\_0 \vee \neg prize(3))$
$\wedge(\neg open\_door(3) \vee p\_open\_door(3)\_0 \vee prize(2))$
$\wedge(win\_keep \vee \neg prize(1))$
$\wedge(\neg win\_keep \vee prize(1))$
$\wedge(win\_switch \vee \neg prize(2) \vee open\_door(2))$
$\wedge(win\_switch \vee \neg prize(3) \vee open\_door(3))$
$\wedge(\neg win\_switch \vee prize(2) \vee prize(3))$
$\wedge(\neg win\_switch \vee prize(2) \vee \neg open\_door(3))$
$\wedge(\neg win\_switch \vee \neg open\_door(2) \vee prize(3))$
$\wedge(\neg win\_switch \vee \neg open\_door(2) \vee \neg open\_door(3))$
$\wedge(\neg prize(1) \vee \neg prize(2))$
$\wedge(\neg prize(1) \vee \neg prize(3))$
$\wedge(\neg prize(2) \vee \neg prize(3))$
$\wedge(prize(1) \vee prize(2) \vee prize(3))$

Weights:

$W(p\_open\_door(2)\_0) = 0.5$	$W(\neg p\_open\_door(2)\_0) = 0.5$
$W(p\_open\_door(3)\_0) = 0.5$	$W(\neg p\_open\_door(3)\_0) = 0.5$
$W(select\_door(1)) = 1.00$	$W(\neg select\_door(1)) = 0.00$
$W(prize(1)) = 0.33$	$W(\neg prize(1)) = 1.00$
$W(prize(2)) = 0.33$	$W(\neg prize(2)) = 1.00$
$W(prize(3)) = 0.33$	$W(\neg prize(3)) = 1.00$
$W(open\_door(2)) = 1.00$	$W(\neg open\_door(2)) = 1.00$
$W(open\_door(3)) = 1.00$	$W(\neg open\_door(3)) = 1.00$
$W(win\_keep) = 1.00$	$W(\neg win\_keep) = 1.00$
$W(win\_switch) = 1.00$	$W(\neg win\_switch) = 1.00$

1.3 Weighted Model Counting

1.3.1 Weighted model counters on above CNFs

We have selected MiniC2D and Cachet as weighted model counters and have executed them on DIMACS versions of the CNFs of the previous tasks. The DIMACS files can be found under report/encodings. The output of the model counters is listed below.

MiniC2D

MiniC2D needs to be executed with the $-W$ flag in order for it to do weighted model counting. The resulting probability can be read next to "Count".

Listing 1.1: MiniC2D on ENC1 encoding of Cancer network

```
Constructing CNF... DONE
CNF stats:
  Vars=30 / Clauses=74
  CNF Time        0.000s
Constructing vtree (from primal graph)... DONE
Vtree stats:
  Vtree widths: con<=5, c_con=48 v_con=5
  Vtree Time      0.001s
Counting... DONE
  Learned clauses        0
Cache stats:
  hit rate        75.0%
  lookups         16
  ent count       4
  ent memory      0.2 KB
  ht  memory      152.6 MB
  clists          1.0 ave, 1 max
  keys            3.0b ave, 3.0b max, 3.0b min
Count stats:
  Count Time      0.000s
  Count           0.9999999999999999
Total Time: 0.012s
```

Listing 1.2: MiniC2D on ENC2 encoding of Cancer network

```
Constructing CNF... DONE
CNF stats:
  Vars=20 / Clauses=30
  CNF Time        0.000s
Constructing vtree (from primal graph)... DONE
Vtree stats:
  Vtree widths: con<=6, c_con=16 v_con=6
  Vtree Time      0.000s
Counting... DONE
  Learned clauses        0
```

```
Cache stats:
  hit rate        23.1%
  lookups         26
  ent count       20
  ent memory      1.0 KB
  ht  memory      152.6 MB
  clists          1.0 ave, 1 max
  keys            1.8b ave, 3.0b max, 1.0b min
Count stats:
  Count Time      0.000s
  Count           1.0000000000000000
Total Time: 0.012s
```

Listing 1.3: MiniC2D on CNF encoding of Monty Hall

```
Constructing CNF... DONE
CNF stats:
  Vars=10 / Clauses=26
  CNF Time        0.000s
Constructing vtree (from primal graph)... DONE
Vtree stats:
  Vtree widths: con<=4, c_con=22 v_con=4
  Vtree Time      0.000s
Counting... DONE
  Learned clauses        0
Cache stats:
  hit rate        20.0%
  lookups         5
  ent count       4
  ent memory      0.2 KB
  ht  memory      152.6 MB
  clists          1.0 ave, 1 max
  keys            3.2b ave, 4.0b max, 3.0b min
Count stats:
  Count Time      0.000s
  Count           1.0000000000000000
Total Time: 0.011s
```

Cachet

For Cachet, there is no need to use extra parameters to get a probability. It is reported next to "Satisfying probability".

Listing 1.4: Cachet on ENC1 encoding of Cancer network

```
Number of total components              11
Number of split components              2
Number of non-split components          5
Number of SAT residual formula          12
Number of trivial components            0
```

```
Number of changed components          0
Number of adjusted components         0
First component split level           1

Number of Decisions                   11
Max Decision Level                    5
Number of Variables                   30
Original Num Clauses                  74
Original Num Literals                 172
Added Conflict Clauses                0
Added Conflict Literals               0
Deleted Unrelevant clauses            0
Deleted Unrelevant literals           0
Number of Implications                124
Total Run Time                        0.0163

Satisfying probability                8.72319e-08
Number of solutions                   93.6645
```

Listing 1.5: Cachet on ENC2 encoding of Cancer network

```
Number of total components            11
Number of split components            2
Number of non-split components        5
Number of SAT residual formula        12
Number of trivial components          0
Number of changed components          0
Number of adjusted components         0
First component split level           1

Number of Decisions                   11
Max Decision Level                    5
Number of Variables                   20
Original Num Clauses                  30
Original Num Literals                 84
Added Conflict Clauses                0
Added Conflict Literals               0
Deleted Unrelevant clauses            0
Deleted Unrelevant literals           0
Number of Implications                72
Total Run Time                        0.017372

Satisfying probability                1
Number of solutions                   1.04858e+06
```

Listing 1.6: Cachet on WCNF encoding of Monty Hall

```
Number of total components            4
Number of split components            1
Number of non-split components        2
```

```
Number of SAT residual formula          5
Number of trivial components            0
Number of changed components            0
Number of adjusted components           0
First component split level             2

Number of Decisions                     4
Max Decision Level                      4
Number of Variables                     10
Original Num Clauses                    26
Original Num Literals                   73
Added Conflict Clauses                  0
Added Conflict Literals                 0
Deleted Unrelevant clauses              0
Deleted Unrelevant literals             0
Number of Implications                  26
Total Run Time                          0.016062

Satisfying probability                  0.444444
Number of solutions                     455.111
```

For ENC1, we see that with Cachet reports a satisfying probability of almost 0. Similarly, for Monty Hall, we see that we get a probability of 0.44. This is due to the fact that with ENC1, the weights of negated literals are 1, but Cachet expects that $weight(x) + weight(-x) = 1$. In the Monty Hall encoding, we also have weights of negated literals equalling 1, which gives the same problem as with ENC1.

1.3.2 Difference between the selected WMCs

MiniC2D Vs Cachet

MiniC2D and Cachet are weighted model counters that work in different ways. In short, MiniC2D is a top down compiler that compiles CNFs into SDDs, while Cachet uses formula caching combined with clause learning and component analysis [[1], [2]].

Both weighted model counters use concepts from the SAT literature. They both use clause learning and component caching in order to reuse components that later appear again during search.

Cachet also uses other methods from SAT literature, like an explicit on the fly calculation of connected components. This is different in MiniC2D, as it relies on vtrees to identify disconnected CNF components. MiniC2d creates vtrees for CNFs and then creates SDDs based on the created vtrees.

1.3.3 Overview of computational requirements

We have executed the model counters with various CNFs to build an overview of computational requirements. The files we used for testing can be found under report/encodings. We have used scripts to convert the ".dsc" files to ENC1 and ENC2 encodings in DIMACS format. We downloaded the ".dsc" files from `http://www.bnlearn.com/bnrepository/`.

Cancer network (small)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	1.0	0.2 KB	0.155s	1.0	1.0 KB	0.000s
Cachet	0.0	?	0.016s	1.0	?	0.016s

Asia network (small)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	1.0	0.9 KB	0.145s	1.0	2.0 KB	0.139s
Cachet	0.0	?	0.018s	1.0	?	0.017s

Sachs network (small)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	0.99707	14.3 KB	0.184s	1.0	14.5 KB	0.154s
Cachet	0.0	?	0.019s	1.0	?	0.017s

Earthquake network (small)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	1.0	0.6 KB	0.137s	1.0	1.0 KB	0.153s
Cachet	0.0	?	0.016s	1.0	?	0.017s

Survey network (small)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	1.0	1.6 KB	0.125s	1.0	2.0 KB	0.125s
Cachet	0.0	?	0.016s	1.0	?	0.016s

Alarm network (medium)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	0.999	819.4 KB	0.215s	0.999	147.2 KB	0.092s
Cachet	0.0	?	0.176s	1.0	?	0.222s

Child network (medium)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	1.0	45.8KB	0.076	1.0	30.8 KB	0.059s
Cachet	0.0	?	0.03s	1.0	?	0.03s

Hailfinder network (large)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	1.0	129.8 MB	0.999	1.0	13.1 MB	1.854
Cachet	val1	val2	a	b	val3	val4

Andes network (very large)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	1.0	5.1GB	246.998s	1.0	364.1 MB	12.06s
Cachet	?	?	$> 4h(killed)$	b	val3	?

From the results listed above, we can conclude that TODO ...

1.4 Knowledge compilation

Vtree with the most compact circuit

During our tests

Pattern for a good vtree

As a vtree is a binary tree, which means that a good vtree is compact. We want thus a vtree that is shallow.

Build an Inference Engine

2.1 Implementation

We have implemented the pipeline using python. Information about installation and usage can be found in `README.MD`.

2.2 Pipeline with previous tasks

2.2.1 Cancer Bayesian network

- Probability:
- Total runtime:
- Runtime of the separate parts:
- Number of variables and lines in CNF:
- Depth of vtree:
- Number of edges and nodes in the circuit:

2.2.2 Monty Hall

- Probability:
- Total runtime:
- Runtime of the separate parts:
- Number of variables and lines in CNF:
- Depth of vtree:
- Number of edges and nodes in the circuit:

2.3 Pipeline on Bayesian learning example

- Probability:
- Total runtime:
- Runtime of the separate parts:
- Number of variables and lines in CNF:
- Depth of vtree:
- Number of edges and nodes in the circuit:

2.4 Pipeline on alarm Bayesian network

- Probability:
- Total runtime:
- Runtime of the separate parts:
- Number of variables and lines in CNF:
- Depth of vtree:
- Number of edges and nodes in the circuit:

Parameter Learning

We have extended the pipeline with limited support for parameter learning. For this functionality, our program expects a file containing tunable probabilities and another file containing values for all probabilities (the ground truth). The ground truth is necessary for generation of interpretations. The amount of interpretations to be generated can be set as well. More information about this feature can be found in `README.MD`.

3.1 Generated interpretations

Four interpretations can be generated with the following command:

```
python3 scripts/inference.py --problog_learn files/
    problog/cancer_learn.pl --problog_learn_truth files/
    problog/cancer.pl --learning_interpretations 4
```

Observations will be dropped with a probability of 30% automatically and the resulting interpretations will be written to src/files/interpretations.txt.

Here is an example of generated interpretations with the command listed above:

```
evidence(\+cancer).
evidence(\+xray("positive")).
evidence(\+dyspnoea).
evidence(\+pollution("high")).
evidence(\+smoker).
==========================================================
evidence(smoker).
evidence(dyspnoea).
evidence(\+pollution("high")).
evidence(pollution("low")).
==========================================================
evidence(xray("negative")).
evidence(\+dyspnoea).
evidence(\+pollution("high")).
evidence(\+smoker).
==========================================================
evidence(smoker).
evidence(dyspnoea).
```

```
evidence(\+pollution("high")).
evidence(pollution("low")).
```

3.2 Pipeline with interpretations

3.2.1 Parameters with 10 interpretations

3.2.2 Parameters with 100 interpretations

3.2.3 Parameters with 1000 interpretations

3.3 Observations for different number of interpretations

We notice that TODO ... Voorspelling: Met minder interpretaties zijn de iteraties sneller (logisch, want minder queries per iteratie) dan met meer interpretaties. Met meer interpretaties gebeurt de convergence wel in minder iteraties omdat de EM dan beter werkt. Wel ook de total runtime er bij zetten.

Appendix

4.1 ENC1

Indicator clauses:

$$
\begin{gathered}
(\neg\, \lambda_{PollutionLow} \vee \neg\, \lambda_{PollutionHigh}) \wedge (\lambda_{PollutionLow} \vee \lambda_{PollutionHigh}) \wedge (\neg \\
\lambda_{SmokerTrue} \vee \neg\, \lambda_{SmokerFalse}) \wedge (\lambda_{SmokerTrue} \vee \lambda_{SmokerFalse}) \wedge (\neg \\
\lambda_{CancerTrue} \vee \neg\, \lambda_{CancerFalse}) \wedge (\lambda_{CancerTrue} \vee \lambda_{CancerFalse}) \wedge (\neg \\
\lambda_{XrayPositive} \vee \neg\, \lambda_{XrayNegative}) \wedge (\lambda_{XrayPositive} \vee \lambda_{XrayNegative}) \wedge (\neg \\
\lambda_{DyspnoeaTrue} \vee \neg\, \lambda_{DyspnoeaFalse}) \wedge (\lambda_{DyspnoeaTrue} \vee \lambda_{DyspnoeaFalse})
\end{gathered}
$$

Parameter clauses:

$$
\begin{gathered}
(\neg\, \lambda_{PollutionLow} \vee \theta_{PollutionLow}) \wedge (\lambda_{PollutionLow} \vee \neg\, \theta_{PollutionLow}) \wedge (\neg \\
\lambda_{PollutionHigh} \vee \theta_{PollutionHigh}) \wedge (\lambda_{PollutionHigh} \vee \neg\, \theta_{PollutionHigh}) \wedge (\neg \\
\lambda_{SmokerTrue} \vee \theta_{SmokerTrue}) \wedge (\lambda_{SmokerTrue} \vee \neg\, \theta_{SmokerTrue}) \wedge (\neg \\
\lambda_{SmokerFalse} \vee \theta_{SmokerFalse}) \wedge (\lambda_{SmokerFalse} \vee \neg\, \theta_{SmokerFalse}) \wedge (\neg \\
\lambda_{PollutionLow} \vee \neg\, \lambda_{SmokerTrue} \vee \neg\, \lambda_{CancerTrue} \vee \\
\theta_{CancerTrue|PollutionLow,SmokerTrue}) \wedge (\lambda_{PollutionLow} \vee \neg \\
\theta_{CancerTrue|PollutionLow,SmokerTrue}) \wedge (\lambda_{SmokerTrue} \vee \neg \\
\theta_{CancerTrue|PollutionLow,SmokerTrue}) \wedge (\lambda_{CancerTrue} \vee \neg \\
\theta_{CancerTrue|PollutionLow,SmokerTrue}) \wedge (\neg\, \lambda_{PollutionLow} \vee \neg\, \lambda_{SmokerTrue} \vee \neg \\
\lambda_{CancerFalse} \vee \theta_{CancerFalse|PollutionLow,SmokerTrue}) \wedge (\lambda_{PollutionLow} \vee \neg \\
\theta_{CancerFalse|PollutionLow,SmokerTrue}) \wedge (\lambda_{SmokerTrue} \vee \neg \\
\theta_{CancerFalse|PollutionLow,SmokerTrue}) \wedge (\lambda_{CancerFalse} \vee \neg \\
\theta_{CancerFalse|PollutionLow,SmokerTrue}) \wedge (\neg\, \lambda_{PollutionLow} \vee \neg\, \lambda_{SmokerFalse} \vee \neg \\
\lambda_{CancerTrue} \vee \theta_{CancerTrue|PollutionLow,SmokerFalse}) \wedge (\lambda_{PollutionLow} \vee \neg \\
\theta_{CancerTrue|PollutionLow,SmokerFalse}) \wedge (\lambda_{SmokerFalse} \vee \neg \\
\theta_{CancerTrue|PollutionLow,SmokerFalse}) \wedge (\lambda_{CancerTrue} \vee \neg \\
\theta_{CancerTrue|PollutionLow,SmokerFalse}) \wedge (\neg\, \lambda_{PollutionLow} \vee \neg\, \lambda_{SmokerFalse} \vee \neg \\
\lambda_{CancerFalse} \vee \theta_{CancerFalse|PollutionLow,SmokerFalse}) \wedge (\lambda_{PollutionLow} \vee \neg \\
\theta_{CancerFalse|PollutionLow,SmokerFalse}) \wedge (\lambda_{SmokerFalse} \vee \neg \\
\theta_{CancerFalse|PollutionLow,SmokerFalse}) \wedge (\lambda_{CancerFalse} \vee \neg \\
\theta_{CancerFalse|PollutionLow,SmokerFalse}) \wedge (\neg\, \lambda_{PollutionHigh} \vee \neg\, \lambda_{SmokerTrue} \vee \neg \\
\lambda_{CancerTrue} \vee \theta_{CancerTrue|PollutionHigh,SmokerTrue}) \wedge (\lambda_{PollutionHigh} \vee \neg \\
\theta_{CancerTrue|PollutionHigh,SmokerTrue}) \wedge (\lambda_{SmokerTrue} \vee \neg \\
\theta_{CancerTrue|PollutionHigh,SmokerTrue}) \wedge (\lambda_{CancerTrue} \vee \neg \\
\theta_{CancerTrue|PollutionHigh,SmokerTrue}) \wedge (\neg\, \lambda_{PollutionHigh} \vee \neg\, \lambda_{SmokerTrue} \vee \neg \\
\lambda_{CancerFalse} \vee \theta_{CancerFalse|PollutionHigh,SmokerTrue}) \wedge (\lambda_{PollutionHigh} \vee \neg
\end{gathered}
$$

$\theta_{CancerFalse|PollutionHigh,SmokerTrue}) \wedge (\lambda_{SmokerTrue} \vee \neg \theta_{CancerFalse|PollutionHigh,SmokerTrue}) \wedge (\lambda_{CancerFalse} \vee \neg \theta_{CancerFalse|PollutionHigh,SmokerTrue}) \wedge (\neg \lambda_{PollutionHigh} \vee \neg \lambda_{SmokerFalse} \vee \neg \lambda_{CancerTrue} \vee \theta_{CancerTrue|PollutionHigh,SmokerFalse}) \wedge (\lambda_{PollutionHigh} \vee \neg \theta_{CancerTrue|PollutionHigh,SmokerFalse}) \wedge (\lambda_{SmokerFalse} \vee \neg \theta_{CancerTrue|PollutionHigh,SmokerFalse}) \wedge (\lambda_{CancerTrue} \vee \neg \theta_{CancerTrue|PollutionHigh,SmokerFalse}) \wedge (\neg \lambda_{PollutionHigh} \vee \neg \lambda_{SmokerFalse} \vee \neg \lambda_{CancerFalse} \vee \theta_{CancerFalse|PollutionHigh,SmokerFalse}) \wedge (\lambda_{PollutionHigh} \vee \neg \theta_{CancerFalse|PollutionHigh,SmokerFalse}) \wedge (\lambda_{SmokerFalse} \vee \neg \theta_{CancerFalse|PollutionHigh,SmokerFalse}) \wedge (\lambda_{CancerFalse} \vee \neg \theta_{CancerFalse|PollutionHigh,SmokerFalse}) \wedge (\neg \lambda_{CancerTrue} \vee \neg \lambda_{XrayPositive} \vee \theta_{XrayPositive|CancerTrue}) \wedge (\lambda_{CancerTrue} \vee \neg \theta_{XrayPositive|CancerTrue}) \wedge (\lambda_{XrayPositive} \vee \neg \theta_{XrayPositive|CancerTrue}) \wedge (\neg \lambda_{CancerTrue} \vee \neg \lambda_{XrayNegative} \vee \theta_{XrayNegative|CancerTrue}) \wedge (\lambda_{CancerTrue} \vee \neg \theta_{XrayNegative|CancerTrue}) \wedge (\lambda_{XrayNegative} \vee \neg \theta_{XrayNegative|CancerTrue}) \wedge (\neg \lambda_{CancerFalse} \vee \neg \lambda_{XrayPositive} \vee \theta_{XrayPositive|CancerFalse}) \wedge (\lambda_{CancerFalse} \vee \neg \theta_{XrayPositive|CancerFalse}) \wedge (\lambda_{XrayPositive} \vee \neg \theta_{XrayPositive|CancerFalse}) \wedge (\neg \lambda_{CancerFalse} \vee \neg \lambda_{XrayNegative} \vee \theta_{XrayNegative|CancerFalse}) \wedge (\lambda_{CancerFalse} \vee \neg \theta_{XrayNegative|CancerFalse}) \wedge (\lambda_{XrayNegative} \vee \neg \theta_{XrayNegative|CancerFalse}) \wedge (\neg \lambda_{CancerTrue} \vee \neg \lambda_{DyspnoeaTrue} \vee \theta_{DyspnoeaTrue|CancerTrue}) \wedge (\lambda_{CancerTrue} \vee \neg \theta_{DyspnoeaTrue|CancerTrue}) \wedge (\lambda_{DyspnoeaTrue} \vee \neg \theta_{DyspnoeaTrue|CancerTrue}) \wedge (\neg \lambda_{CancerTrue} \vee \neg \lambda_{DyspnoeaFalse} \vee \theta_{DyspnoeaFalse|CancerTrue}) \wedge (\lambda_{CancerTrue} \vee \neg \theta_{DyspnoeaFalse|CancerTrue}) \wedge (\lambda_{DyspnoeaFalse} \vee \neg \theta_{DyspnoeaFalse|CancerTrue}) \wedge (\neg \lambda_{CancerFalse} \vee \neg \lambda_{DyspnoeaTrue} \vee \theta_{DyspnoeaTrue|CancerFalse}) \wedge (\lambda_{CancerFalse} \vee \neg \theta_{DyspnoeaTrue|CancerFalse}) \wedge (\lambda_{DyspnoeaTrue} \vee \neg \theta_{DyspnoeaTrue|CancerFalse}) \wedge (\neg \lambda_{CancerFalse} \vee \neg \lambda_{DyspnoeaFalse} \vee \theta_{DyspnoeaFalse|CancerFalse}) \wedge (\lambda_{CancerFalse} \vee \neg \theta_{DyspnoeaFalse|CancerFalse}) \wedge (\lambda_{DyspnoeaFalse} \vee \neg \theta_{DyspnoeaFalse|CancerFalse})$

Weights:
W($\lambda_{PollutionLow}$) = 1.00
W($\neg\lambda_{PollutionLow}$) = 1.00
W($\lambda_{PollutionHigh}$) = 1.00
W($\neg\lambda_{PollutionHigh}$) = 1.00
W($\lambda_{SmokerTrue}$) = 1.00
W($\neg\lambda_{SmokerTrue}$) = 1.00
W($\lambda_{SmokerFalse}$) = 1.00
W($\neg\lambda_{SmokerFalse}$) = 1.00
W($\lambda_{CancerTrue}$) = 1.00
W($\neg\lambda_{CancerTrue}$) = 1.00
W($\lambda_{CancerFalse}$) = 1.00
W($\neg\lambda_{CancerFalse}$) = 1.00
W($\lambda_{XrayPositive}$) = 1.00
W($\neg\lambda_{XrayPositive}$) = 1.00
W($\lambda_{XrayNegative}$) = 1.00
W($\neg\lambda_{XrayNegative}$) = 1.00
W($\lambda_{DyspnoeaTrue}$) = 1.00
W($\neg\lambda_{DyspnoeaTrue}$) = 1.00
W($\lambda_{DyspnoeaFalse}$) = 1.00
W($\neg\lambda_{DyspnoeaFalse}$) = 1.00

$W(\theta_{PollutionLow}) = 0.90$
$W(\neg\theta_{PollutionLow}) = 1.00$
$W(\theta_{PollutionHigh}) = 0.10$
$W(\neg\theta_{PollutionHigh}) = 1.00$
$W(\theta_{SmokerTrue}) = 0.30$
$W(\neg\theta_{SmokerTrue}) = 1.00$
$W(\theta_{SmokerFalse}) = 0.70$
$W(\neg\theta_{SmokerFalse}) = 1.00$
$W(\theta_{CancerTrue|PollutionLow,SmokerTrue}) = 0.03$
$W(\neg\theta_{CancerTrue|PollutionLow,SmokerTrue}) = 1.00$
$W(\theta_{CancerFalse|PollutionLow,SmokerTrue}) = 0.97$
$W(\neg\theta_{CancerFalse|PollutionLow,SmokerTrue}) = 1.00$
$W(\theta_{CancerTrue|PollutionLow,SmokerFalse}) = 0.00$
$W(\neg\theta_{CancerTrue|PollutionLow,SmokerFalse}) = 1.00$
$W(\theta_{CancerFalse|PollutionLow,SmokerFalse}) = 1.00$
$W(\neg\theta_{CancerFalse|PollutionLow,SmokerFalse}) = 1.00$
$W(\theta_{CancerTrue|PollutionHigh,SmokerTrue}) = 0.05$
$W(\neg\theta_{CancerTrue|PollutionHigh,SmokerTrue}) = 1.00$
$W(\theta_{CancerFalse|PollutionHigh,SmokerTrue}) = 0.95$
$W(\neg\theta_{CancerFalse|PollutionHigh,SmokerTrue}) = 1.00$
$W(\theta_{CancerTrue|PollutionHigh,SmokerFalse}) = 0.02$
$W(\neg\theta_{CancerTrue|PollutionHigh,SmokerFalse}) = 1.00$
$W(\theta_{CancerFalse|PollutionHigh,SmokerFalse}) = 0.98$
$W(\neg\theta_{CancerFalse|PollutionHigh,SmokerFalse}) = 1.00$
$W(\theta_{XrayPositive|CancerTrue}) = 0.90$
$W(\neg\theta_{XrayPositive|CancerTrue}) = 1.00$
$W(\theta_{XrayNegative|CancerTrue}) = 0.10$
$W(\neg\theta_{XrayNegative|CancerTrue}) = 1.00$
$W(\theta_{XrayPositive|CancerFalse}) = 0.20$
$W(\neg\theta_{XrayPositive|CancerFalse}) = 1.00$
$W(\theta_{XrayNegative|CancerFalse}) = 0.80$
$W(\neg\theta_{XrayNegative|CancerFalse}) = 1.00$
$W(\theta_{DyspnoeaTrue|CancerTrue}) = 0.65$
$W(\neg\theta_{DyspnoeaTrue|CancerTrue}) = 1.00$
$W(\theta_{DyspnoeaFalse|CancerTrue}) = 0.35$
$W(\neg\theta_{DyspnoeaFalse|CancerTrue}) = 1.00$
$W(\theta_{DyspnoeaTrue|CancerFalse}) = 0.30$
$W(\neg\theta_{DyspnoeaTrue|CancerFalse}) = 1.00$
$W(\theta_{DyspnoeaFalse|CancerFalse}) = 0.70$
$W(\neg\theta_{DyspnoeaFalse|CancerFalse}) = 1.00$

4.2 ENC2

Indicator clauses

$(\neg\ \lambda_{PollutionLow} \vee \neg\ \lambda_{PollutionHigh}) \wedge (\lambda_{PollutionLow} \vee \lambda_{PollutionHigh}) \wedge (\neg\ \lambda_{SmokerTrue} \vee \neg\ \lambda_{SmokerFalse}) \wedge (\lambda_{SmokerTrue} \vee \lambda_{SmokerFalse}) \wedge (\neg$

$\lambda_{CancerTrue} \vee \neg\, \lambda_{CancerFalse}) \wedge (\lambda_{CancerTrue} \vee \lambda_{CancerFalse}) \wedge (\neg\, \lambda_{XrayPositive} \vee \neg\, \lambda_{XrayNegative}) \wedge (\lambda_{XrayPositive} \vee \lambda_{XrayNegative}) \wedge (\neg\, \lambda_{DyspnoeaTrue} \vee \neg\, \lambda_{DyspnoeaFalse}) \wedge (\lambda_{DyspnoeaTrue} \vee \lambda_{DyspnoeaFalse})$

Parameter clauses

$(\neg\, \rho_{PollutionLow} \vee \lambda_{PollutionLow}) \wedge (\rho_{PollutionLow} \vee \lambda_{PollutionHigh}) \wedge (\neg\, \rho_{SmokerTrue} \vee \lambda_{SmokerTrue}) \wedge (\rho_{SmokerTrue} \vee \lambda_{SmokerFalse}) \wedge (\neg\, \lambda_{PollutionLow} \vee \neg\, \lambda_{SmokerTrue} \vee \neg\, \rho_{CancerTrue|PollutionLow,SmokerTrue} \vee \lambda_{CancerTrue}) \wedge (\neg\, \lambda_{PollutionLow} \vee \neg\, \lambda_{SmokerTrue} \vee \rho_{CancerTrue|PollutionLow,SmokerTrue} \vee \lambda_{CancerFalse}) \wedge (\neg\, \lambda_{PollutionLow} \vee \neg\, \lambda_{SmokerFalse} \vee \neg\, \rho_{CancerTrue|PollutionLow,SmokerFalse} \vee \lambda_{CancerTrue}) \wedge (\neg\, \lambda_{PollutionLow} \vee \neg\, \lambda_{SmokerFalse} \vee \rho_{CancerTrue|PollutionLow,SmokerFalse} \vee \lambda_{CancerFalse}) \wedge (\neg\, \lambda_{PollutionHigh} \vee \neg\, \lambda_{SmokerTrue} \vee \neg\, \rho_{CancerTrue|PollutionHigh,SmokerTrue} \vee \lambda_{CancerTrue}) \wedge (\neg\, \lambda_{PollutionHigh} \vee \neg\, \lambda_{SmokerTrue} \vee \rho_{CancerTrue|PollutionHigh,SmokerTrue} \vee \lambda_{CancerFalse}) \wedge (\neg\, \lambda_{PollutionHigh} \vee \neg\, \lambda_{SmokerFalse} \vee \neg\, \rho_{CancerTrue|PollutionHigh,SmokerFalse} \vee \lambda_{CancerTrue}) \wedge (\neg\, \lambda_{PollutionHigh} \vee \neg\, \lambda_{SmokerFalse} \vee \rho_{CancerTrue|PollutionHigh,SmokerFalse} \vee \lambda_{CancerFalse}) \wedge (\neg\, \lambda_{CancerTrue} \vee \neg\, \rho_{XrayPositive|CancerTrue} \vee \lambda_{XrayPositive}) \wedge (\neg\, \lambda_{CancerTrue} \vee \rho_{XrayPositive|CancerTrue} \vee \lambda_{XrayNegative}) \wedge (\neg\, \lambda_{CancerFalse} \vee \neg\, \rho_{XrayPositive|CancerFalse} \vee \lambda_{XrayPositive}) \wedge (\neg\, \lambda_{CancerFalse} \vee \rho_{XrayPositive|CancerFalse} \vee \lambda_{XrayNegative}) \wedge (\neg\, \lambda_{CancerTrue} \vee \neg\, \rho_{DyspnoeaTrue|CancerTrue} \vee \lambda_{DyspnoeaTrue}) \wedge (\neg\, \lambda_{CancerTrue} \vee \rho_{DyspnoeaTrue|CancerTrue} \vee \lambda_{DyspnoeaFalse}) \wedge (\neg\, \lambda_{CancerFalse} \vee \neg\, \rho_{DyspnoeaTrue|CancerFalse} \vee \lambda_{DyspnoeaTrue}) \wedge (\neg\, \lambda_{CancerFalse} \vee \rho_{DyspnoeaTrue|CancerFalse} \vee \lambda_{DyspnoeaFalse})$

Weights

$W(\lambda_{PollutionLow}) = 1.00$
$W(\neg\, \lambda_{PollutionLow}) = 1.00$
$W(\lambda_{PollutionHigh}) = 1.00$
$W(\neg\, \lambda_{PollutionHigh}) = 1.00$
$W(\lambda_{SmokerTrue}) = 1.00$
$W(\neg\, \lambda_{SmokerTrue}) = 1.00$
$W(\lambda_{SmokerFalse}) = 1.00$
$W(\neg\, \lambda_{SmokerFalse}) = 1.00$
$W(\lambda_{CancerTrue}) = 1.00$
$W(\neg\, \lambda_{CancerTrue}) = 1.00$
$W(\lambda_{CancerFalse}) = 1.00$
$W(\neg\, \lambda_{CancerFalse}) = 1.00$
$W(\lambda_{XrayPositive}) = 1.00$
$W(\neg\, \lambda_{XrayPositive}) = 1.00$
$W(\lambda_{XrayNegative}) = 1.00$
$W(\neg\, \lambda_{XrayNegative}) = 1.00$
$W(\lambda_{DyspnoeaTrue}) = 1.00$
$W(\neg\, \lambda_{DyspnoeaTrue}) = 1.00$
$W(\lambda_{DyspnoeaFalse}) = 1.00$
$W(\neg\, \lambda_{DyspnoeaFalse}) = 1.00$
$W(\rho_{PollutionLow}) = 0.90$
$W(\neg\, \rho_{PollutionLow}) = 0.10$

$W(\rho_{SmokerTrue}) = 0.30$
$W(\neg\ \rho_{SmokerTrue}) = 0.70$
$W(\rho_{CancerTrue|PollutionLow,SmokerTrue}) = 0.03$
$W(\neg\ \rho_{CancerTrue|PollutionLow,SmokerTrue}) = 0.97$
$W(\rho_{CancerTrue|PollutionLow,SmokerFalse}) = 0.00$
$W(\neg\ \rho_{CancerTrue|PollutionLow,SmokerFalse}) = 1.00$
$W(\rho_{CancerTrue|PollutionHigh,SmokerTrue}) = 0.05$
$W(\neg\ \rho_{CancerTrue|PollutionHigh,SmokerTrue}) = 0.95$
$W(\rho_{CancerTrue|PollutionHigh,SmokerFalse}) = 0.02$
$W(\neg\ \rho_{CancerTrue|PollutionHigh,SmokerFalse}) = 0.98$
$W(\rho_{XrayPositive|CancerTrue}) = 0.90$
$W(\neg\ \rho_{XrayPositive|CancerTrue}) = 0.10$
$W(\rho_{XrayPositive|CancerFalse}) = 0.20$
$W(\neg\ \rho_{XrayPositive|CancerFalse}) = 0.80$
$W(\rho_{DyspnoeaTrue|CancerTrue}) = 0.65$
$W(\neg\ \rho_{DyspnoeaTrue|CancerTrue}) = 0.35$
$W(\rho_{DyspnoeaTrue|CancerFalse}) = 0.30$
$W(\neg\ \rho_{DyspnoeaTrue|CancerFalse}) = 0.70$

Bibliography

[1] Umut Oztok and Adnan Darwiche. A top-down compiler for sentential decision diagrams. *In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3141–3148, 2015.

[2] Paul Beame Tian Sang and Henry Kautz. Heuristics for fast exact model counting. *Eighth International Conference on Theory and Applications of Satisfiability Testing*, 2005.