Capita Selecta AI - Probabilistic Programming Inference for SRL

Thierry Deruyttere (r0660485) & Armin Halilovic (r0679689)

April 2, 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:

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

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

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	0.3 KB	0.053s	1.0	1.0 KB	0.050s
Cachet	0.0	?	0.016s	1.0	?	0.016s

Asia network (small)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	1.0	1.2 KB	0.049s	1.0	1.9 KB	0.05s
Cachet	0.0	?	0.018s	1.0	?	0.017s

Sachs network (small)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	0.99707	16.8 KB	0.075s	1.0	13.4 KB	0.07s
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	0.051s	1.0	1.0 KB	0.05s
Cachet	0.0	?	0.016s	1.0	?	0.017s

Survey network (small)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	1.0	0.5 KB	0.035s	1.0	2.0 KB	0.052s
Cachet	0.0	?	0.016s	1.0	?	0.016s

Alarm network (medium)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	0.999	451.1 KB	0.217s	0.999	143.4 KB	0.093s
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	0.076s	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	0.999	591.8 MB	46.065s	1.0	25.1MB	2.73s
Cachet	0	?	58.86s	1	?	15.21s

Andes network (very large)

	ENC1			ENC2		
	Prob	**Memory**	**Runtime**	**Prob**	**Memory**	**Runtime**
Minic2d	1.0	5.5GB	266.605s	1.0	122.8 MB	5.646s
Cachet	?	?	$> 4h(killed)$	?	?	$> 4h(killed)$

For Cachet, we didn't find a way to output memory usage, so we cannot compare that aspect to MiniC2D. From the results listed above, we can conclude that ENC2 is a better encoding than ENC1 when it comes to computational requirements. In most of the cases, it has lower memory usage and a lower runtime. Concerning the weighted model counters, we clearly see that Cachet is faster than MiniC2D on small to medium networks. When we use larger networks, Cachet is a lot slower than MiniC2D, and for the very large network, Cachet couldn't even finish in 4 hours!

1.4 Knowledge compilation

Vtree with the most compact circuit

For this task, we used both the SDD and MiniC2D packages to generate vtrees and SDDs. The resulting vtrees and their corresponding SDDs can be found under report/1.4/. Subdirectories have been made for each WMC's output for each of the CNFs. The file name in each of these directories follow the following naming convention: (SDD_vtree_type | MiniC2D_vtree_method)[_SDD_minimize_cardinality]_(vtree | sdd)

- **ENC1**: SDD built the best vtree for the most compact circuit with the "-t balanced -m" flags. The next best circuit can be created via the vtree created by MiniC2D with "-m 3" (reverse elimination order).
- **ENC2**: For this encoding, we have different options that result in the best circuits. The options "-t balanced" for SDD and "-m 3" for MiniC2D

are again one of the best. Like for ENC1, the vtree created by SDD is better than the one from MiniC2D.

- **Monty Hall**: Here we see that by using the option "-t balanced -m", we receive the most compact circuit and yet again closely followed by "-m 3" from MiniC2D.

Pattern for a good vtree

From our tests, we've noticed that vtrees that are more shallow result in the best circuits. This seems logical to us, since a vtree is a binary search tree, and with such a tree you also prefer it to be 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`.

Reporting of statistics other than the resulting probabilities only happens with our pipeline when MiniC2D is selected as model counter (flag –model_counter minic2d).

2.2 Pipeline with previous tasks

2.2.1 Cancer Bayesian network

- Probabilities:

```
cancer             : 0.01163
dyspnoea           : 0.3040705
pollution("high"): 0.1
pollution("low")  : 0.9
smoker             : 0.3
xray("negative")  : 0.7918589999999999
xray("positive")  : 0.208141
```

- Total runtime: 0.594s
- Runtime of the separate parts:

```
Time to parse ground problog:  0.011s
Time to convert ground problog to first order logic:
    0.000s
Time to convert first order logic to CNF:  0.002s
Time for all model counting:  0.58s
```

- Number of variables and lines in CNF: 17, 77
- Depth of vtree: 11
- Number of edges and nodes in the circuit: 81, 94

2.2.2 Monty Hall

- Probabilities:

```
prize(1)       : 0.3333333333333333
prize(2)       : 0.3333333333333333
prize(3)       : 0.3333333333333333
select_door(1): 1.0
win_keep       : 0.3333333333333333
win_switch     : 0.6666666666666666
```

- Total runtime: 0.469s
- Runtime of the separate parts:

```
Time to parse ground problog: 0.011s
Time to convert ground problog to first order logic:
    0.0s
Time to convert first order logic to CNF: 0.001s
Time for all model counting: 0.458s
```

- Number of variables and lines in CNF: 10, 26
- Depth of vtree: 8
- Number of edges and nodes in the circuit: 52, 44

2.3 Pipeline on Bayesian learning example

- Probabilities:

```
weight(c1, 0.1): 0.0
weight(c1, 0.3): 2.167977370376305e-06
weight(c1, 0.5): 0.004149743263716653
weight(c1, 0.7): 0.13174850211546205
weight(c1, 0.9): 0.8640995866434509
weight(c2, 0.1): 3.227661043218289e-06
weight(c2, 0.3): 0.02410999702877393
weight(c2, 0.5): 0.6672475437022279
weight(c2, 0.7): 0.3062862583084689
weight(c2, 0.9): 0.0023529732994859343
```

- Total runtime: 3.633s
- Runtime of the separate parts:

```
Time to parse ground problog: 0.338s
Time to convert ground problog to first order logic:
    0.024s
Time to convert first order logic to CNF: 0.088s
Time for all model counting: 3.183s
```

- Number of variables and lines in CNF: 192, 1062
- Depth of vtree: 95
- Number of edges and nodes in the circuit: 646, 525

2.4 Pipeline on alarm Bayesian network

We had to drop many nodes in order for the pipeline to be able to process the ALARM Bayesian network. This is due to how we implemented the conversion from First Order Logic to CNF. We have come across methods with better performance, but were out of time to try other implementations. Only 21 of the 37 nodes are left in the network.

We used the pipeline to convert files/networks/alarm.net to CNF with the following command.

```
python3 scripts/inference.py --bayesian_network files/
    networks/alarm.net
```

We kept dropping leaf nodes until a CNF was created within reasonable time (a couple seconds). Then, we used the ground problog file in the output of the pipeline and stored it in files/problog/alarm.pl. Finally, to this file, we added queries for all the nodes in the network.

The probabilities shown below are only those of the nodes lower in the network. To see all of the probabilities, run:

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

- Probabilities:

```
cVP("HIGH")                : 0.1541757517134
cVP("LOW")                 : 0.1129475165156
cVP("NORMAL")              : 0.7160774170523999
lVEDVOLUME("HIGH")         : 0.2095
lVEDVOLUME("LOW")          : 0.0886
lVEDVOLUME("NORMAL")       : 0.7019
pAP("HIGH")                : 0.0575
pAP("LOW")                 : 0.0496
pAP("NORMAL")              : 0.8929
pCWP("HIGH")               : 0.2064165610039
pCWP("LOW")                : 0.1129475165156
pCWP("NORMAL")             : 0.6755051316624
sHUNT("HIGH")              : 0.103095
sHUNT("NORMAL")            : 0.8969050000000001
sTROKEVOLUME("HIGH")       : 0.0404
sTROKEVOLUME("LOW")        : 0.1808
sTROKEVOLUME("NORMAL")     : 0.7787999999999999
tPR("HIGH")                : 0.2971
tPR("LOW")                 : 0.3068
tPR("NORMAL")              : 0.3961
```

```
vENTMACH("HIGH")        : 0.056
vENTMACH("LOW")         : 0.056
vENTMACH("NORMAL")      : 0.838
vENTMACH("ZERO")        : 0.05
```

- Total runtime: 6.413s
- Runtime of the separate parts:

```
Time to parse ground problog:  0.037s
Time to convert ground problog to first order logic:
    0.003s
Time to convert first order logic to CNF:  0.148s
Time for all model counting:  6.225s
```

- Number of variables and lines in CNF: 112, 1118
- Depth of vtree: 27
- Number of edges and nodes in the circuit: 974, 711

Parameter Learning

We have extended the pipeline with 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

The following results were reached after 45 iterations. Our current implementation determines convergence is reached when all parameters change by less than 0.005.

```
smoker                    : 0.3757309044335211
pollution("low")          : 0.6944157672162523
pollution("high")         : 0.1
p_cancer_0                : 1.7136773738417463e-05
p_cancer_1                : 1.5696846453312893e-12
p_cancer_2                : 0.05290598927777933
p_cancer_3                : 0.07266786066467187
p_xray("positive")_0:  0.9581146900452404
p_xray("negative")_0:  0.25501789141868203
p_xray("positive")_1:  0.0
p_xray("negative")_1:  0.8571427918086106
p_dyspnoea_0              : 0.15338511324535153
p_dyspnoea_1              : 0.0
```

3.2.2 Parameters with 100 interpretations

The pipeline was terminated after 1.5 hours at 39 iterations.

```
smoker                    : 0.3295379131986182
pollution("low")          : 0.9041099224149188
pollution("high")         : 0.09589007758508108
p_cancer_0                : 0.1110126093038839
p_cancer_1                : 0.00048335141926639366
p_cancer_2                : 0.37732149668302517
p_cancer_3                : 0.1502500596905257
p_xray("positive")_0:  0.8927878392142493
p_xray("negative")_0:  0.11856852583630233
p_xray("positive")_1:  0.1408494511343412
p_xray("negative")_1:  0.8749301634679101
p_dyspnoea_0              : 0.09252183100200292
p_dyspnoea_1              : 0.30160276817000836
```

3.2.3 Parameters with 1000 interpretations

The pipeline was terminated after 1.5 hours at 3 iterations.

```
probabilities after iteration 3:
```

```
smoker                  : 0.29233350800326574
pollution("low")        : 0.9148055069262087
pollution("high")       : 0.08519449307379132
p_cancer_0              : 0.09403271863945559
p_cancer_1              : 0.025426090633783687
p_cancer_2              : 0.733743251011375
p_cancer_3              : 0.3007407777225963
p_xray("positive")_0:  0.3741545373041979
p_xray("negative")_0:  0.8205193870361593
p_xray("positive")_1:  0.22686346009032624
p_xray("negative")_1:  0.773832837404553
p_dyspnoea_0            : 0.23145524434562476
p_dyspnoea_1            : 0.33029011323407503
```

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:

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

Parameter clauses:

$$(\neg\, \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$$

$$\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:

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

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

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

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

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

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.