1 Design and Implementation

1.1 Input format

The input is the **Bayesian Interchange Format** (BIF) which is a file with .bif extension. In general, Four types of blocks are defined.
The Network Block:
A network block defines the name of the network and lists the properties. The example below specify the network block for the Asia Bayesian Network:

```
network asia{
   property version 1.1;
   property author ..;
}
```

The Varable Block:
Variable blocks define the variables in a network. These blocks used to be called node blocks in the BNIF; it seems that variable conveys more of a statistical meaning while node just refers to a graphical concept. The example below is the bronc node from asia.bif

```
variable bronc{
   type discrete [2] {yes, no};
}
```

Blocks for standard nodes
Standard nodes have to define the probabilities for each discrete parent instantiation. An example of a standard probability block is:

```
probability (dysp | bronc, either){
   (yes, yes) 0.9, 0.1;
   (no, yes) 0.7, 0.3;
   (yes, no) 0.8, 0.2;
   (no, no) 0.1, 0.9;
}
```

Probability blocks:
Probability blocks are another way to specify the (conditional) probability tables (CPTs). For these variables, and hence the topology of the network. The block indicates the variables of the probability distribution right after the keyword probability.

```
probability (v_10_8 v_10_7 v_9_8){
    table 0.586357 0.667473 0.789088 0.466932
          0.413643 0.332527 0.210912 0.533068;
}
```

A more detailed explanation of the BIF is explained here: [1]

1.2 Output

The output of the program is the cnf file following the DIMACS format which is a widely accepted standard format for representing CNF clauses and it's also a format used by most of the model counters mentioned in section 3.3.

- A comment line starts with a *c*

[1]http://www.cs.washington.edu/dm/vfml/appendixes/bif.htm

- A line p cnf var clauses specify the instance in CNF format, in which *vars* is the number of variables used in the file and *clauses* is the number of clauses in the CNF.
- Each CNF variable is denoted by a nonzero number smaller than *vars*, the negation of a variable is denoted by a negative number.
- Each clauses contains one or several CNF variables, 0 specifies the end of the clause.

Consider the following CNF:

$$x_1 \lor x_2 \lor \neg x_3$$
$$x_1 \lor x_4 \lor x_5$$
$$\neg x_3 \lor \neg x_4$$

The clauses in DIMACS format

```
c
c A sample DIMAC file
c
p cnf 5 3
1 2 −3 0
1 4 5 0
−3 −4 0
```

1.3 Fetching variables values from Bayesian Network

Pgmpy library does not support fetching CPT values using table indexing, the only way to fetch value is through specifying evidence and variables and query the value through Variable Elimination method. The Variable Elimination method is the bayesian inference method that is Insert time complexety. To solve the problem, I extended the pgmpy library to support fetching variable values without querying using Variable Elimination.

The method get_cpds returns a conditional probability distribution of the node, the returned type is defined as TabularCPD that contains the name of the node, cardinality, variables which are stores as a nested list, the list of evidences and their corresponding cardinalities. An example is given below:

```
cpd = TabularCPD('dysp', 2, [[0.9, 0.7, 0.8, 0.1],
                             [0.1, 0.3, 0.2, 0.9]],
                            ['bronc', 'either'], [2, 2])
```

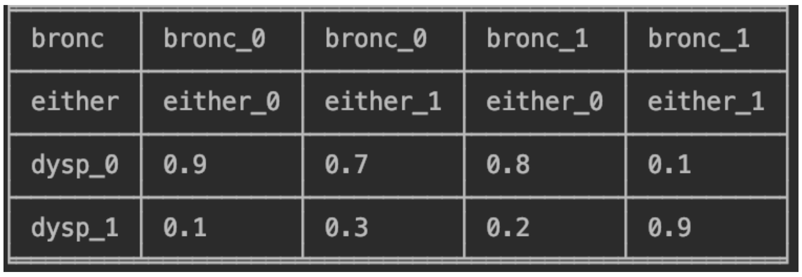
In our case for both bron and either, the evidence all equals to 2, the convention for the order is shown in figure 1. The list which store the values are the transpose of the matrix in BIF format.

1.4 Implementation of Full encoding

Generating Indicator variables:

Generating Parameter variables:

Generating Parameter variables: Write files: Storing weights



bronc	bronc_0	bronc_0	bronc_1	bronc_1
either	either_0	either_1	either_0	either_1
dysp_0	0.9	0.7	0.8	0.1
dysp_1	0.1	0.3	0.2	0.9

Figure 1: A sample printed CPT of a node dyps

1.5 Simplified Full encoding

1.6 Implementation of Improved Encoding

1.7 Implementation of Group Encoding

1.7.1 Splitting the CPT

1.7.2 QM algorithm and extension for multi-variate simplification

2 Project Management

2.1 Original plan

2.2 Time line

2.3 Git contribution

3 Results and Evaluation

3.1 Experiment Setup

3.2 Variable elimination and bayesian with weighted model counting

insert table here

3.3 Number of Clauses

3.4 Model counting results