Inference in Belief Network using Logic Sampling and Likelihood Weighing algorithms

Jasmine K.S[a], PrathviRaj S. Gavani[b], Rajashekar P Ijantakar[b], Sumithra Devi.K.A[c]

[a] Associate Professor ,Department of MCA, jasmineks@rvce.edu.in
[b] Students, Department of MCA , prathviraj@gmail.com, rijantakar1@gmail.com
[c] Director Dept of MCA, sumithraka@gmail.com
R.V. College of Engineering, Bangalore,

KEYWORD

Belief network
Logic sampling
Likelihood weighing
Dynamic decision making
Uncertainty

ABSTRACT

Over the time in computational history, belief networks have become an increasingly popular mechanism for dealing with uncertainty in systems. It is known that identifying the probability values of belief network nodes given a set of evidence is not amenable in general. Many different simulation algorithms for approximating solution to this problem have been proposed and implemented. This paper details the implementation of such algorithms, in particular the two algorithms of the belief networks namely Logic sampling and the likelihood weighing are discussed. A detailed description of the algorithm is given with observed results. These algorithms play crucial roles in dynamic decision making in any situation of uncertainty.

1. Introduction

A belief network is a formal knowledge representation and inference technique consisting of a directed graph and a set of conditional probabilities. Belief networks are an elegant, well founded way to reason with uncertainty, but in general, inference with them is computationally difficult to manage.

Belief networks are used to model uncertainty in a domain. The term "Belief networks" encompasses a whole range of different but related techniques which deal with reasoning under uncertainty. Both quantitative and qualitative techniques are used. Influence diagrams are an extension to belief networks; they are used when working with decision making.

This report describes the implementation and use of stochastic simulation algorithms for doing approximate inference with belief networks. The two main algorithm discussed in this paper are Logic sampling algorithm and Likelihood weighing algorithm.

1.1. Background

Many approaches to reasoning with uncertain knowledge have been proposed. The argument is favour of belief networks, made by Pearl and others, is that only belief networks are based on a really firm theoretical foundation: probability theory.

One use of belief networks has been in the field of expert systems [1]. A rule based system might contain a rule, “If A, then conclude B with certainty C.” In probabilistic terms, this would correspond to the conditional probability statement, $p(B \mid A) = C$. In general, without making conditional probability assumptions, $p(B \mid A)$ does not tell us much, because if there are any other variables in the system, we must consider them as well as B before we can determine the probability of B. Belief networks provide a graphical means of specifying which other variables a variable depends on, and more importantly, which variables can be ignored.

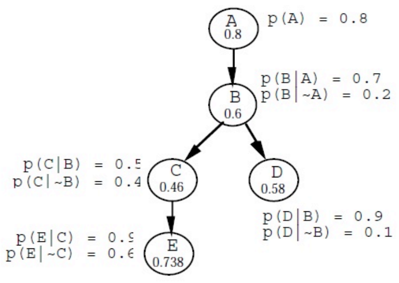


Fig 1: A simple belief network with its associated conditional probabilities and posterior marginal probabilities on nodes.

2. Proposed System

The proposed approach deals with two basic algorithms which facilitates dynamic decision making in node to node communication. Our intent here is to give enough detail for the reader to get an intuitive understanding of the algorithms.

We first describe a brute force algorithm for propagating belief network values. Consider a network with two nodes, A and B (Figure 2) where node A has three states and node B has two states (Boolean). This network has 3*2 = 6 states. For each state of the network, we can calculate a joint probability value of the state (Table 1).

The probability value of any variable in any state can be calculated by simply summing over all rows in which the variable is assigned the state.

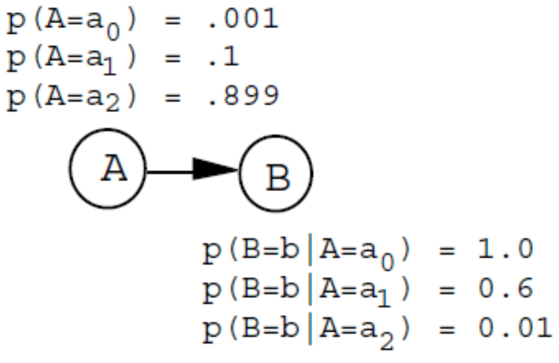


Fig 2: A trivial 2 node belief network

Table 1: Exhaustive list of joint probabilities for a trivial belief network.

A	B	Value		
a_0	b	$p(A=a_0)p(B=b \mid A=a_0)$	$0.001 * 1.0 =$	0.001
a_0	$\bar{b}$	$p(A=a_0)p(B=\bar{b} \mid A=a_0)$	$0.001 * 0.0 =$	0.0
a_1	b	$p(A=a_1)p(B=b \mid A=a_1)$	$0.1 * 0.6 =$	0.06
a_1	$\bar{b}$	$p(A=a_1)p(B=\bar{b} \mid A=a_1)$	$0.1 * 0.4 =$	0.04
a_2	b	$p(A=a_2)p(B=b \mid A=a_2)$	$0.899 * 0.01 =$	0.00899
a_2	$\bar{b}$	$p(A=a_2)p(B=\bar{b} \mid A=a_2)$	$0.899 * 0.99 =$	0.89001

Evidence restricts the cases to those in which the evidence node is in the desired state, and then normalizing. We normalize a variable by dividing each of its possible states by the sum of its possible states. For example, if B is set to !b, we only consider the 3 rows where B is false. In this case, the calculated value for A=a2 would be 0.89001 /0.0+0.04+0.89001=0.957.

Unfortunately, the size of this table is exponential in the number of nodes, so this algorithm quickly becomes intractable as the number of nodes increases. Simulation algorithms select a subset of the rows and use the values calculated to estimate the values for the variables. The simulation algorithms we consider here differ primarily in the method they use to select rows. Each time a row is selected, the probability of selecting that row, p-selecting, is used to normalize the values.

2.1. Logic Sampling Algorithm

The simplest simulation algorithm, logic sampling[4], randomly chooses a state for each node in the network from among the possible states by giving an equal chance to all states. The value of p selecting for this algorithm is a constant ($\prod$ 1/states$_i$ where states$_i$ is the number of states of the node i), but since normalization will negate the e effects of this constant, we avoid the computation and use 1.0. The following pseudo code describes this algorithm [6].

```
**Loop for the number of simulations**
    **For each non-evidence node in the network**
        **Set the state of the node to one of its possible states at random**
    **End For**
    **Set p-selecting to 1.0**
    **Calculate a score (total probabilities / p-selecting)**
    **Score the net (using traditional or Markov blanket scoring)**
```

End Loop
Normalize the node values

For example, consider a very simple graph with two nodes, A and B. Node A represents the arrival time of a student a summer job, and is one of three mutually exclusive and exhaustive states:

a_0 means the student arrives before 7:30, a_1 means the student arrives between 7:30 and 9:00, and a_2 means the student arrives after 9:00. Node B represents the proposition that the student will find a parking space within a ten minute walk of the laboratory, and is conditioned on node A. For

Simplicity, we will let a_i stand for the expression A=ai, and b or !b stand for the expressions B=b or B=!b. The initial probabilities would then be:

p(a0) = 0.001 It is very unlikely the student will arrive before 7:30

p(a1) = 0.1 It is unlikely the student will arrive before 9:00

p(a2) = 0.899 Usually the student shows up after 9:00

$p(b|a_0)$ = 1.0 Before7:30 there is always parking

$p(b|a_1)$ = 0.6 There is usually parking between 7:30 and 9:00

$p(b|a_2)$ = 0.01 It is hard to park close after 9:00

Flowchart for the algorithm is depicted below

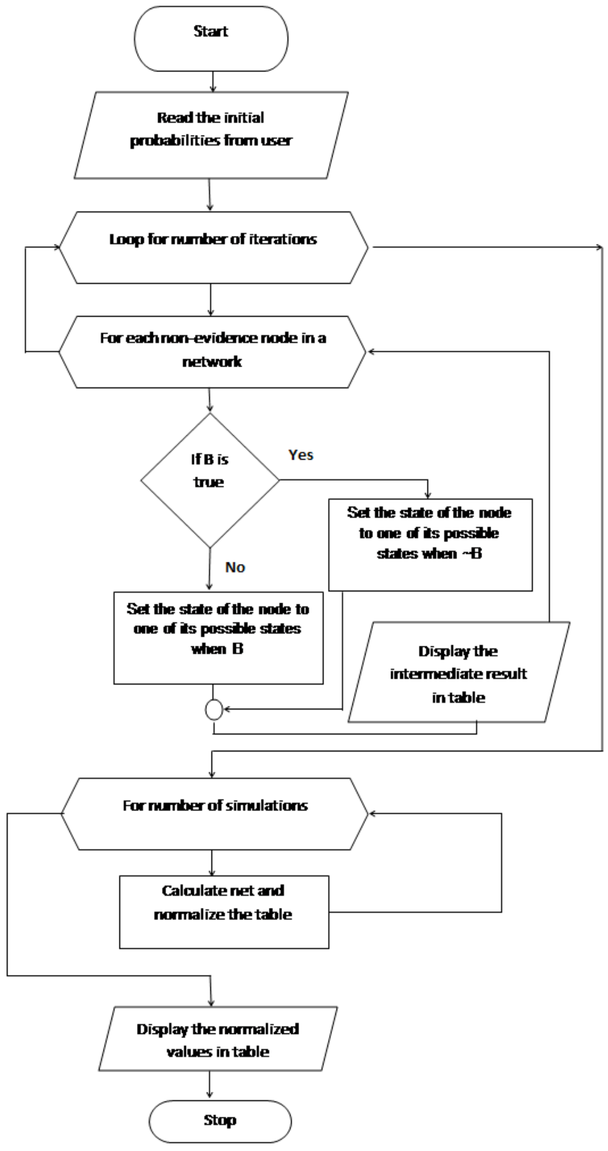

Start
Read the initial probabilities from user
Loop for number of iterations
For each non-evidence node in a network
If B is true
Yes
No
Set the state of the node to one of its possible states when ~B
Set the state of the node to one of its possible states when B
Display the intermediate result in table
For number of simulations
Calculate net and normalize the table
Display the normalized values in table
Stop

Fig 3: Flowchart of Logical sampling algorithm

2.2. Likelihood Weighing Algorithm

This algorithm is considered to be more advanced compared to the Logic sampling algorithm[5]. This uses more information than the Logic Sampling to choose its state.

Likelihood Weighing is different from the logic sampling by weighing the node states before selecting them. The node states are weighed by their prior probabilities.

The likelihood algorithm is as follows[6]

```
Sort the nodes of the graph so that parents always proceed children
Loop for the number of simulations
    p-selecting = 1.0
    For each non-evidence node N in the network (in graph order) where N has k parents $P_1….P_k$ in states $SP_i$
        Choose a state $S_N$ according to the conditional probability of the state p-selecting = p-selecting * $p(N=S_N \mid P_1=sP_1,….,P_K = SP_K)$
    End For
    Calculate a score (total probabilities / p-selecting)
    Score the net (using traditional or Markov blanket scoring)
End Loop
Normalize the node values
```

The flowchart for the algorithms is depicted in the figure 4

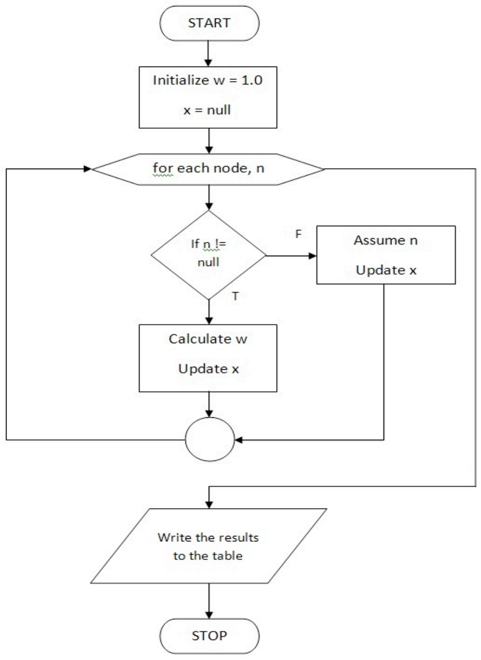


Fig. 4 Flow chart of the Likelihood weighing algorithm.

3. Result obtained

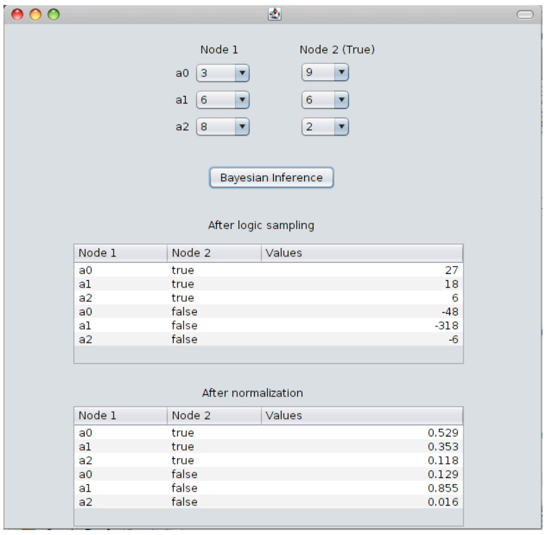
3.1. Logic Sampling Algorithm

The below figures can be understood well with the example mentioned in section 2.1.

In Fig 5 and Fig 7 the values of a0,a1,a2 nodes are in increasing order and the value for node a0 still stands out against the values of node a1 and a2.

In Fig 6 we get quite obvious result where the node a0 gets higher values than other nodes

In Fig 7 assigns highest values to node a0. So the value of node a0 gets maximum value in return.



	Node 1	Node 2 (True)
a0	3	9
a1	6	6
a2	8	2

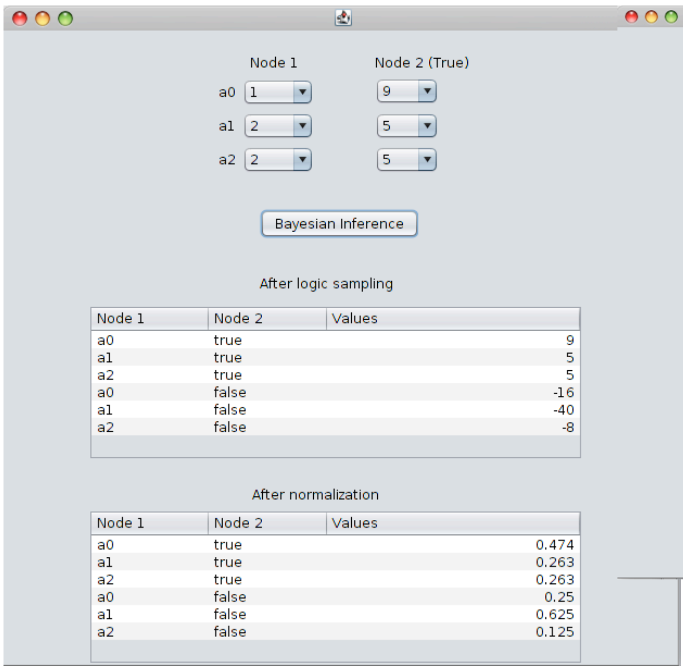
After logic sampling

Node 1	Node 2	Values
a0	true	27
a1	true	18
a2	true	6
a0	false	-48
a1	false	-318
a2	false	-6

After normalization

Node 1	Node 2	Values
a0	true	0.529
a1	true	0.353
a2	true	0.118
a0	false	0.129
a1	false	0.855
a2	false	0.016

Fig 5 . Output Screen-1



	Node 1	Node 2 (True)
a0	1	9
a1	2	5
a2	2	5

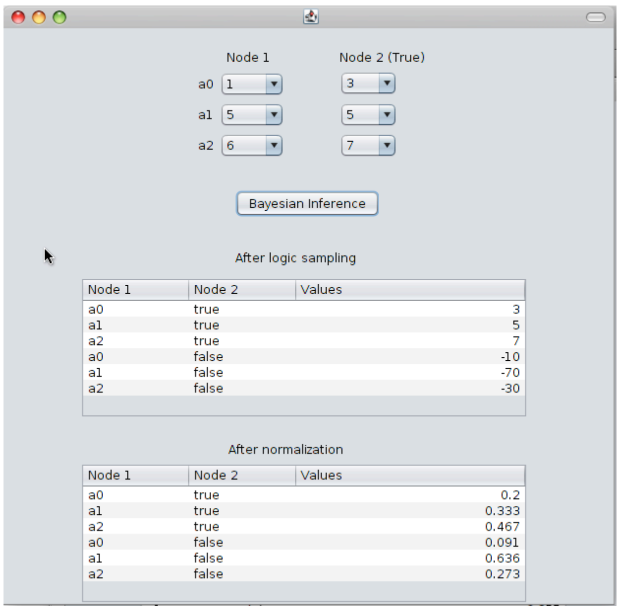
After logic sampling

Node 1	Node 2	Values
a0	true	9
a1	true	5
a2	true	5
a0	false	-16
a1	false	-40
a2	false	-8

After normalization

Node 1	Node 2	Values
a0	true	0.474
a1	true	0.263
a2	true	0.263
a0	false	0.25
a1	false	0.625
a2	false	0.125

Fig 6 . Output Screen-2



	Node 1	Node 2 (True)
a0	1	3
a1	5	5
a2	6	7

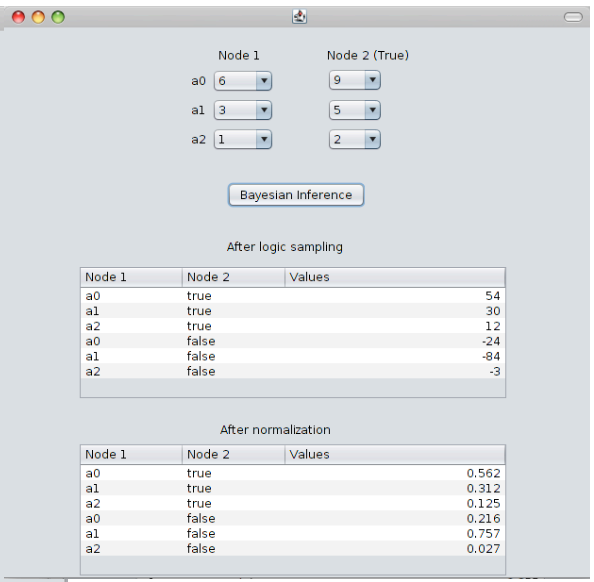
After logic sampling

Node 1	Node 2	Values
a0	true	3
a1	true	5
a2	true	7
a0	false	-10
a1	false	-70
a2	false	-30

After normalization

Node 1	Node 2	Values
a0	true	0.2
a1	true	0.333
a2	true	0.467
a0	false	0.091
a1	false	0.636
a2	false	0.273

Fig 7.Output Screen-3



	Node 1	Node 2 (True)
a0	6	9
a1	3	5
a2	1	2

After logic sampling

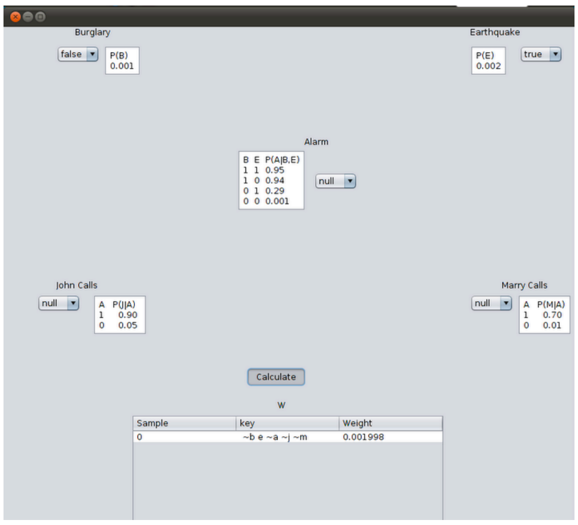
Node 1	Node 2	Values
a0	true	54
a1	true	30
a2	true	12
a0	false	-24
a1	false	-84
a2	false	-3

After normalization

Node 1	Node 2	Values
a0	true	0.562
a1	true	0.312
a2	true	0.125
a0	false	0.216
a1	false	0.757
a2	false	0.027

Fig 8.Output Screen-4

3.2. Likelihood Weighing Algorithm



Burglary: false

P(B)
0.001

Earthquake: true

P(E)
0.002

Alarm: null

B	E	P(A|B,E)
1	1	0.95
1	0	0.94
0	1	0.29
0	0	0.001

John Calls: null

A	P(J|A)
1	0.90
0	0.05

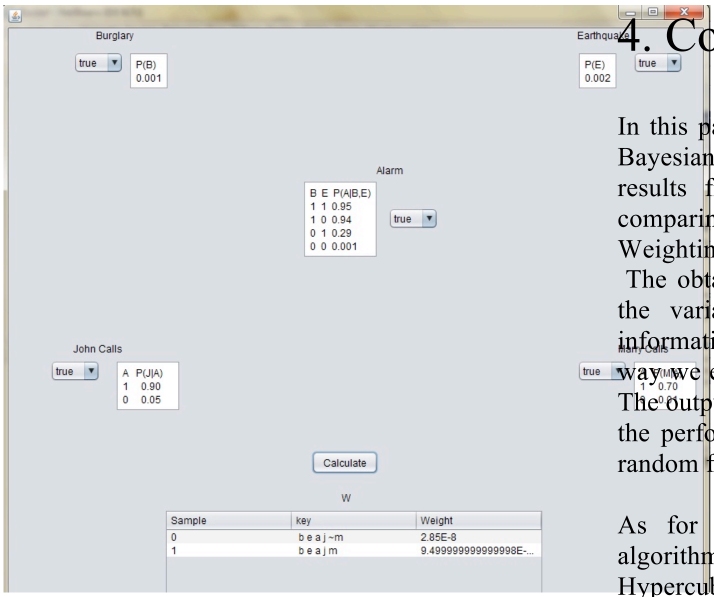
Marry Calls: null

A	P(M|A)
1	0.70
0	0.01

W

Sample	key	Weight
0	~b e ~a ~j ~m	0.001998

Fig 9. Output Screen-5



P(B)
0.001

P(E)
0.002

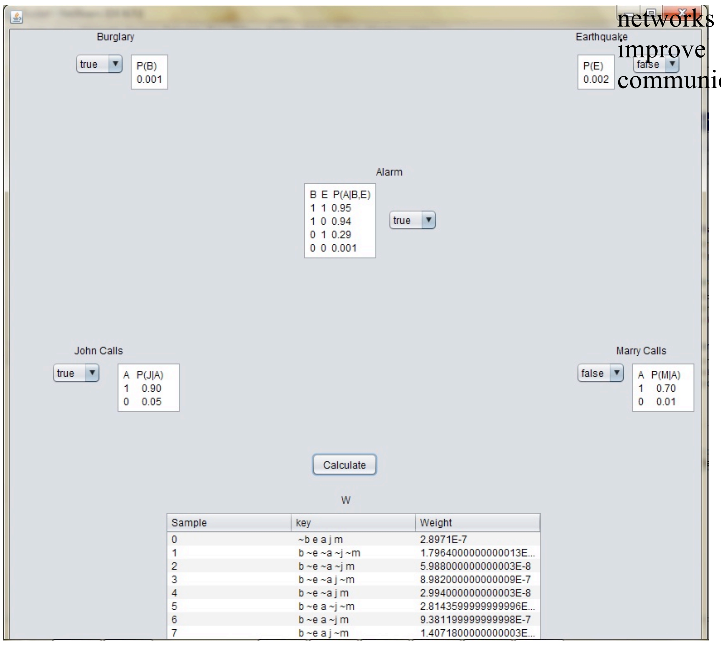
B	E	P(A|B,E)
1	1	0.95
1	0	0.94
0	1	0.29
0	0	0.001

A	P(J|A)
1	0.90
0	0.05

A	P(M|A)
1	0.70
0	0.01

Sample	key	Weight
0	b e a j ~m	2.85E-8
1	b e a j m	9.4999999999999998E-...

Fig.10. Output Screen-6



P(B)
0.001

P(E)
0.002

B	E	P(A|B,E)
1	1	0.95
1	0	0.94
0	1	0.29
0	0	0.001

A	P(J|A)
1	0.90
0	0.05

A	P(M|A)
1	0.70
0	0.01

Sample	key	Weight
0	~b e a j m	2.8971E-7
1	b ~e ~a ~j ~m	1.7964000000000013E...
2	b ~e ~a ~j m	5.9880000000000003E-8
3	b ~e ~a j ~m	8.9820000000000009E-7
4	b ~e ~a j m	2.9940000000000003E-8
5	b ~e a ~j ~m	2.8143599999999996E...
6	b ~e a ~j m	9.3811999999999998E-7
7	b ~e a j ~m	1.4071800000000003E...

Fig 11. Output Screen-7

4. Conclusion

In this paper, two algorithms for implementing Bayesian belief network are discussed. The results from a set of empirical experiments comparing Logic Sampling and Likelihood Weighting are presented as part of results.
The obtained results will help one to simulate the variables and the functions with more information to weight the simulations. In this way we expect to obtain more uniform weights. The output screens from 1 to 11 help to compare the performance of the proposed algorithms in random fashion.

As for future work, many new simulation algorithm methods are proposed, such as Latin Hypercube Sampling and Systemic sampling which are better to deal with extreme distributions, if we can use these random networks as benchmark to test and we can improve our work for more complicated communication network.

References

[1] Cooper GF. Current research directions in the development of expert systems based on belief networks. Applied Stochastic Models and Data Analysis 1989; 5:39 -52.

[2] Changyun Wang. Bayesian Belief Network Simulation. Florida state university, February 2003:30-35.

[3] Pearl J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA: Morgan Kaufmann, 1988.

[4] Shachter R, Peot M. Simulation approaches to general probabilistic inference on belief networks. In: Henrion M, Shachter R, Kanal L, Lemmer J, eds. Uncertainty in Artificial Intelligence 5. Amsterdam: Elsevier Science Publishers B.V. (North-Holland), 1990:221-31

[5] Chavez RM, Cooper GF. A fully polynomial randomized approximation scheme for the Bayesian inference problem (working paper). Technical report. Stanford University, Stanford California. Fall, 1988.

[6] Steve B. Cousins ,William Chen, Mark E. Frisse,, CABeN:A Collection of algorithms for belief networks,Proc. Fifteenth Annual Symposium on computer Applications in Medical care, USA, November 1991