Improving Scalability Using Inference Algorithms

Sonika Malik, Anupama Kaushik, Prabhjot Kaur
Maharaja Surajmal Institute of Technology, GGSIPU, Delhi
sonika.malik@msit.in, anupama@msit.in, prabhjot.kaur@msit.in

Abstract

As we can see there has been an increasing demand for a more and more scalable algorithm, owing to the increase in the complexity of our day to problems and situations. In this paper, we basically compare the inferencing algorithm in two circumstances. One is not much complex while the other one is complex. Till now various researchers proposed mc-sat with a great future scope for being better than others but no practical and explained proof was there. Gibbs sampling was considered as the best among all the inferencing algorithms but we observed that MC-SAT, an inference algorithm that combines ideas from Markov chain Monte Carlo(MCMC) and satisfiability is much better as the situation gets complicated. MC-SAT is basically based on Markov logic, which defines Markov networks using weighted clauses in first-order logic. MC-SAT improves the scalability of the unbbayes software and also Multi Entity Bayesian Network(MEBN) by reducing the time complexity and space complexity. We successfully implemented the algorithms and compared them for the examples we provide.

Keywords: OWL, MEBN, probabilistic ontology, probabilistic reasoning, uncertainty, Bayesian networks, BN, inference, logic, semantic web, MC-SAT, MCMC

DOI Number: 10.48047/nq.2022.20.19.NQ99096 **NeuroQuantology2022;20(19): 1056-1071**

1. Introduction

Inferencing algorithms are based on statistical inference that is they work on previous stats and knowledge of the problem set which helps us reduce our sample size by reducing the original sample set to most probable set thus data analysts and scientist are following approach from the algorithms for processing data to the amount of information they process.

The process of inferring conclusion from multiple observations is called inductive reasoning. The resultmay be correct or incorrect, or may be correct in certain situations.

Two possible definitions of "inference" are:

1. A conclusion reached on the basis of evidence and reasoning.
2. The process of reaching such a conclusion.

1.1 Inference in Bayesian Networks

Inference over a Bayesian network can come in two forms.

The first is simply evaluating the joint probability of a particular assignment of values for each variable (or a subset) in the network. For this, we already have a factorized form of the joint distribution, so we simply evaluate that product using the provided conditional probabilities. If we only care about a subset of variables, we will need to marginalize out the ones we are not interested in. In many cases, this mayresult in underflow, so it is common to take the logarithm of that product, which is equivalent to adding up the individual logarithms of each term in the product.

The second, more interesting inference task, is to find P(x|e) as in Eq. 1, or, to find the probability of someassignment of a subset of the variables (x) given assignments of other variables (our evidence, e). In the above example, an example of this could be to find P (Sprinkler, WetGrass | Cloudy), where {Sprinkler, WetGrass} is our x, and {Cloudy} is our e. In order to calculate this, we use the fact that P(x|e) = P (x, e) / P(e) = αP (x, e), where α is a normalization constant that we will calculate at the end such that P(x|e) + P (¬x | e) = 1. In order to calculate P (x, e), we must marginalize the joint probability distribution over the variables that do not appear in x or e, which we will denote as Y.

$$P(x|e) = \alpha \sum_{\forall y \in Y} P(x, e, Y) \tag{1}$$

For the given example, we can calculate P (Sprinkler, WetGrass | Cloudy) as follows:We would calculate P (¬x | e) in the same fashion, just setting the value of the variables in x to false instead of true. Once both P (x | e) and P (¬x | e) are calculated, we can solve for α, which equals 1 /(P (x | e) + P (¬x | e)).

Note that in larger networks, Y will most likely be quite large, since most inference tasks will only directly use a small subset of the variables. In scenarios like these, exact inference as shown above is very complex to calculate, so methods must be used to reduce the amount of computation. One moreefficient method of exact inference is through variable elimination, which takes advantage of the factthat each factor only involves a small number of variables. This means that the summations can be rearranged such that only factors involving a given variable are used in the marginalization of that variable. Alternatively, many networks are too large even for this method, so approximate inference methods such as MCMC are instead used; these provide probability estimations that require significantly less computation than exact inference methods.

1.2 Scalability

Scalability is the ability of a program to scale. For example, if you can do something on a small database (say less than 1000 records), a program that is highly scalable would work well on a small set as well as working well on a large set (say millions, or billions of records).

Like gap said, it would have a linear growth of resource requirements. Look up Big-O notation for more details about how programs can require more computation the larger the data input gets. Something parabolic like Big-O(x^2) is far less efficient with large x inputs than something linear like Big-O(x).

1.3 MEBN

Probabilistic approach to solve uncertainty is the most powerful conceptual tool that we have yet developed to understand and manage uncertainty. A Bayesian network, Bayes network, belief network, Bayesian) model or probabilistic directed acyclic graphical model is graphical probabilistic model in which the dependency graph is an acyclic directed graph.

Multi-Entity Bayesian Networks (Laskey, 2008) is a language for representing first order probabilistic knowledge base. MEBN integrate first order logic with Bayesian probability. MEBN logic expresses probabilistic knowledge as a collection of MEBN fragments (MFrags) organized into MEBN Theories (MTheories). An MFrag represents a conditional probability distribution of the instances of its resident random variables given the values of instances of their parents in the Fragment graphs and given the context constraints. A collection of MFrags represents a joint probability distribution over an unbounded, possibly infinite number of instances of its random variables. The joint distribution is specified by means of the local distributions together with the conditional independence relationships implied by the fragment graphs. Context terms are used to specify constraints under which the local distributions apply. MEBN combines First order logic with inferential power of BN, with this combination MEBN is able to provide a consistent treatment of uncertainty.

In traditional BN reasoning was about a fixed no. of attributes whereas a MEBN can deal with a varying number of attributes. Also, it provides a means of defining probability over an unbounded and varying no. of interrelated hypothesis using a syntax, set of models construction and interface process and semantics

Example-If all we have is BNs, and there are M months of data with N variables per month, we must build a BN with MxN nodes, and fill in identical arcs and local probability distributions at each time step. With MEBNs, we can write a single BN Frag relating the variables at time t with the variables at time t+1 and say repeat for all t's.

Multi-Entity Bayesian Networks (MEBN), a first-order language for specifying probabilistic knowledge bases as parameterized fragments of Bayesian networks.

This parameterization helps to add additional information and values, thus making our model more accurate and efficient to handle uncertainty.

We begin the paper by first explaining all the required knowledge for the algorithms. Then we provided all the algorithm used in MEBN and compared it with one of the MLN algorithm which is MC-SAT and provided the result for showing which one more scalable under different circumstances. Then at last we have our future work and conclusion for the research work.

1.4 Markov-logic networks

A first-order KB can be seen as a set of hard constraints on the set of possible worlds: if a world violates even one formula, it has zero probability. The basic idea in MLNs is to soften these constraints: when a world violates one formula in the KB it is less probable, but not impossible. The fewer formulas a world violates, the more probable it is. Each formula has an associated weight that reflects how strong a constraint it is: the higher the weight, the greater the difference in log probability between a world that satisfies the formula and one that does not, other things being equal.

1.5 Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods comprise a class of algorithms for sampling from a probability distribution. By constructing a Markov chain that has the desired distribution as its equilibrium distribution, one can obtain a sample of the desired distribution by observing the chain after a number of steps. The more steps there are, the more closely the distribution of the sample matches the actual desired distribution.

2. Evolution of Inference algorithms

First in 1962 the likelihood function has proved to be such a powerful tool for inference that it has been extended and generalized to semi-parametric models and non-parametric models, and various pseudo likelihood functions have been proposed for more complex models then came the Lazy DpLL in 1962, it formed the basis for modern SAT. After this, in 1966, MCMC, it is basically implemented for MLN (Markov logic networks). A simple approach to combining first-order logic and probabilistic graphical models in a single representation. A Markov logic network (MLN) is a first-order knowledge base with a weight attached to each formula, then before 1980 Junction tree (also known as 'Clique Tree') is a method used in machine learning to extract marginalization in general graphs. After this Backward Sampling in 1980, this forward–backward algorithm is an inference algorithm for hidden Markov models which computes the posterior marginal of all hidden state variables given a sequence of observations/emissions then BELIEF PROPAGATION in 1982 it was first proposed by Judea Pearl in 1982, who formulated it as an exact inference algorithm on trees, which was later extended to polytrees. While it is not exact on general graphs anymore, it has been shown to be a useful approximate algorithm. Then came Gibbs 1984, in statistics, Gibbs sampling or a Gibbs sampler is a Markov chain Monte Carlo (MCMC) algorithm for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution, when direct sampling is difficult. It is designed only for Bayesian network as compared to the MCMC which is for MLN only then Backward Sampling with children in1985 then SampleSearch in between (1985-1995) then Sample Search with Choco solver in between (1990-2000) then SampleSearch with Back jumping in between (1990-2000) then Backward Sampling with priors- 1987 Backward SampleSearch in 1988 the Ace in1990 Blocked Gibbs sampler - 1992- A blocked Gibbs sampler groups two or more variables together and samples from their joint distribution conditioned on all other variables, rather than sampling from each one individually. For example, in a hidden Markov model, a blocked Gibbs sampler might sample from all the latent variables making up the Markov chain in one go, using the forward-backward algorithm. Collapsed Gibbs sampler in1994 is a collapsed Gibbs sampler integrates out (marginalizes over) one or more variables when sampling for some other variable then Backward SampleSearch with back jumping in 1994 Gaussian BP in between (2000-2006) modified. In this contribution, we develop a solution based upon Gaussian belief propagation (GaBP) that does not involve direct matrix inversion. BP is for direct problems which have small sample space while Gaussian BP is for larger sample space then sample sat in 2004 then SAT in 2006(concept introduced-1972) it is very important because it is NP-Complete. To understand what this means you need a clear notion of Complexity classes. Often however this is not feasible. Where it is feasible is, when no other fast algorithm is known, such as the puzzle you mention. In this

case you do not have to develop an algorithm for the puzzle, but can use any of the highly optimized SAT-Solvers out there and you will end up with a reasonable fast algorithm for your puzzle then Mc-sat in 2006 it basically combines MCMC and sat algorithm in order to form a better and faster algorithm for complex problems in MLN. It is better than Gibbs, if the sample space for the problem is very large and lazy-mc-sat is an extension of the alchemy system 2007. It is much faster than Sat Algorithm.

3. Algorithms comparison

The comparison of existing algorithms is given in Table 1.

Table 1. Comparison of Existing Algorithms

S. No	Algorithm	Year	Principle	Domain	Base
1	Likelihood	1962	Likelihood	MLN, MEBN,	FOL

	Inferencing		Ratio	BN	
2	Gibbs	1984	Randomized Gibbs Sampler	Markov Logic Networks, Bayesian Networks	FOL
3	Backward Sampling	1987	Posterior marginals, Dynamic Programming	BN, MLN	Numeric
4	Sample Search	1985-1995	Mean Inferencing	BN, MLN	Numeric
5	Belief Propagation	1982	Sum-Product Message Passing	MLN, BN	Numeric
6	Gausian	2000-2006	Guassian Process	MLN, BN	Numeric
7	MC-SAT	2006	Combination of MCMC and Satisfiability	MLN, MEBN	FOL

3.1 Gibbs Sampling

Gibbs also use MCMC method. It not only checks the recent results but all the earlier cases and outcomes of the problem.

We select one random variable at a time. Then check it for all cases and reassemble it then take another variable and do the same iteration until all the variables are checked once. In likelihood and similar other algorithm all the samples generated (reassembled variables) they are independent of their samples. Samples are those which are created after reassembling. Basically every iteration is treated as a sample until we get the desired result.
This is dependent sampling algorithm which generates by past experience. Gibbs sampling is commonly used as a means of statistical inference, especially Bayesian inference. It is a randomized algorithm (i.e. an algorithm that makes use of random numbers), and is an alternative to deterministic algorithms for statistical inference such as the expectation-maximization algorithm (EM).

As with other MCMC algorithms, Gibbs sampling generates a Markov chain of samples, each of which is correlated with nearby samples. As a result, care must be taken if independent samples are desired. Generally, samples from the beginning of the chain (the *burn-in period*) may not accurately represent the desired distribution and are usually discarded. It has been shown, however, that using a longer chain instead (e.g. a chain that is *n* times as long as the initially considered chain using a thinning factor of *n*) leads to better estimates of the true posterior. Thus, *thinning* should only be applied when time or computer memory are restricted.

```
ALGORITHM
We have probabilistic approach for 3 instances(assumed)-
        P (Q1, Q2, Q3)
1) Finding initial conditional probability for given instances.
2) For t1: T (for t a number from 1 to T)
        $Q_1^t \sim P(Q1/Q_2^{t-1}, Q_3^{t-1})$
        $Q_2^t \sim P(Q2/Q_1^t, Q_3^{t-1})$
        $Q_3^t \sim P(Q3/Q_1^t, Q_2^t)$
3) Step 2 repeated until constant value acquired.
```

As we can see from the algorithm that first we find conditional probability of each instances and then keep on repeating it until all the instances have similar value which will be done after 1000 of iterations. Hence we don't reject any parameters and every variable will be accessed. But we clearly see that as the complexity or no. of instances to be traversed increases he Gibbs will take more time and become quite complex.

3.2 LIKELIHOOD

In statistics, maximum likelihood estimation (MLE) is a method of estimating the parameters of a statistical model, given observations. MLE attempts to find the parameter values that maximize the likelihood function, given the observations. The resulting estimate is called a maximum likelihood estimate, which is also abbreviated as MLE.

An example, suppose that we are interested in the heights of adult female penguins, but are unable to measure the height of every penguin in a population (due to cost or time constraints). Assuming that the heights are normally distributed with some unknown mean and variance, the mean and variance can be estimated with MLE while only knowing the heights of some sample of the overall population. MLE would accomplish that by taking the mean and variance as parameters and finding particular parametric values that make the observed results the most probable given the normal model.

We start with a given parametric model, f (y; θ), the probability density function for a random variable Y. At least initially we assume that y is a vector of n components $y_1, \ldots, y_n$, $y_i \in R$, and θ ∈". In regular statistical models, " is very often taken to be Rd or a subset of Rd. The likelihood function for this parametric model is

$$L(\theta; y) = c(y) f(y; \theta), \quad (1)$$

viewed as a function of θ, for fixed y. While some authors define the likelihood function without the arbitrary function c(y), this definition shows explicitly that the value of the likelihood function is only meaningful in relative terms. It is usually more convenient to work with the log-likelihood function

$$(\theta; y) = a(y) + \log f(y; \theta);$$

Y = (Y1, ..., Yn) are independent and identically distributed normal random variables, with mean μ and variance σ 2, then

(θ; y) = −n 2 log σ 2 − 1 2σ 2! (yi − μ) 2,

where θ = (μ, σ 2), " = R × R+

(θ; y) = −n 2 log σ 2 − 1 2σ 2!" yi − xTi β #2,

where θ = (β, σ 2).

Functional invariance

The maximum likelihood estimator selects the parameter value which gives the observed data the largest possible probability (or probability density, in the continuous case). If the parameter consists of a number of components, then we define their separate maximum likelihood estimators, as the corresponding component of the MLE of the complete parameter. Consistent with this, if is the MLE for, and if is any transformation of, then the MLE for is by definition.
It maximizes the so-called profile likelihood: The MLE is also invariant with respect to certain transformations of the data. If where is one to one and does not depend on the parameters to be estimated, then the density functions satisfy and hence the likelihood functions for and differ only by a factor that does not depend on the model parameters. For example, the MLE parameters of the log normal distribution are the same as those of the normal distribution fitted to the logarithm of the data.

Higher-order properties

As noted above, the maximum likelihood estimator is √*n* -consistent and asymptotically efficient, meaning that it reaches the Cramér–Rao bound: where is the Fisher information matrix: In particular, it means that the bias of the maximum likelihood estimator is equal to zero up to the order $^1/\sqrt{n}$. However, when we consider the higher-order terms in the expansion of the distribution of this estimator, it turns out that *ϑ*mle has bias of order $^1/_n$. This bias is equal to (component wise) where denotes the (*j, k*)-thcomponent of the *inverse* Fisher information matrix, and Using these formulae it is possible to estimate the second-order bias of the maximum likelihood estimator, and *correct* for that bias by subtracting it:

This estimator is unbiased up to the terms of order $^1/_n$, and is called the **bias-corrected maxim likelihood estimator**. This bias-corrected estimator is *second-order efficient* (at least within the curved exponential family), meaning that it has minimal mean squared error among all second-order bias corrected estimators, up to the terms of the order $^1/n^2$. It is possible to continue this process, that is to derive the third-order bias-correction term, and so on. However, as was shown by Kano (1996), the maximum likelihood estimator is **not** third-order efficient.

Relation to Bayesian inference

A maximum likelihood estimator coincides with the most probable Bayesian estimator given a uniform prior distribution on the parameters. Indeed, the maximum a posteriori estimate is the parameter *ϑ* that maximizes the probability of *ϑ* given the data, given by Bayes' theorem: where is the prior distribution for the parameter *ϑ* and where is the probability of the data averaged over all parameters. Since the denominator is independent of *ϑ*, the Bayesian estimator is obtained by maximizing with respect to *ϑ*. If we further assume that the prior is a uniform distribution, the Bayesian estimator is obtained by maximizing the likelihood function. Thus the Bayesian estimator coincides with the maximum likelihood estimator for a uniform prior distribution.

3.3 MC-SAT

MC-SAT, an MCMC algorithm that is able to handle deterministic and near-deterministic dependencies by using Wei et al.'s. SampleSAT as a subroutine to efficiently jump between isolated or near-isolated regions of non-zero probability, while preserving detailed balance.

MC-SAT accepts problems defined in Markov logic, a very general language that has both Markov networks and finite first-order logic as special cases (Richardson & Domingo's 2006). MC-SAT deals with of both probabilistic and deterministic information. For example, entity resolution (the problem of determining which observations correspond to the same object) involves both probabilistic inferences (e.g., observations with similar properties are more likely to be the same object)

MC-SAT applies slice sampling to Markov logic, using SampleSAT to sample a new state given the auxiliary variables.

Algorithm 1 MC-SAT (clauses, weights, num samples)

```
$x^{(0)}$ ← Satisfy (hard clauses)
for i ← 1 to num samples do
M ← ∅
for all $c^k$∈clauses satisfied by $x^{(i-1)}$ do with probability $1 - e^{-wk}$add ck to M
end for
 Sample $x^{(i)} \sim$USAT(M)
end for
```

MC-Sat works by applying slice sampling to Markov logic using Sample SAT in order to sample a given state based on the previous results. For each given ground clause we define a potential function ckcorresponds to the ϕ k (x) = exp (w k f k (x)). Grounding is a method of replacing variable by a constant i.e. We define the ground clause as a base condition for inferencing the next state. Assume for the moment that all weights are non-negative. On the ithiteration of MC-SAT, if c k is not satisfied by the current state x (i), u k is drawn uniformly from [0, 1]; therefore, u k ≤ 1 and u k ≤ e w k, and there is no requirement that it be satisfied in the next state. If c k is satisfied, u k is drawn uniformly from [0, e w k], and with probability 1−e −w k it will be greater than 1, in which case the next state must satisfy c k. Thus, sampling all the auxiliary variables determines a random subset M of the currently satisfied clauses that must also be satisfied in the next state. All Hard clause needs to be satisfied initially, so for a given state x (0) we satisfy all the hard clauses. We then take as the next state a uniform sample from the set of states SAT (M) that satisfy M. (Notice that SAT (M) is never empty, because it always contains at least the current state.) The initial state is found by applying a satisfiability solver to the set of all hard clauses in the network (i.e., all clauses with infinite weight). If this set is unsatisfiable, the output of MC-SAT is undefined.

3.4 Gaussian Inference

Gaussian Inference works on the basis of Gaussian Process. Supports graphical models, such as Bayesian networks and Markov random fields. Gaussian process (GP) regression is a fully probabilistic method for performing non-linear regression. In a Bayesian framework, regression models can be made robust by using heavy-tailed distributions instead of using normal distribution for modelling noise. This work focuses on estimation of parameters for robust GP regression. In literature, these are learned by maximizing the approximate marginal likelihood of data. However, gradient-based optimization algorithms which are used for this purpose can be unstable or may require tuning. A Gaussian process can be thought of as a Gaussian distribution over functions (thinking of functions as infinitely long vectors containing the value of the function at every input). Formally let the input space X and f: X → R a function from the input space to the reals, then we say f is a Gaussian process if for any vector of inputs x = [x_1, x_2, . . ., x_n] ^T such that xi ∈X for all i, the vector of output f(x) = [f(x_1), f(x_2), . . ., f(x_n)] ^T is Gaussian distributed. Belief Propagation may be described as a special case of Gaussian Inference.

One drawback of the Gaussian Process is that it scales very badly with the number of observations N. Solving for the coefficients α defining the mean function requires O (N^3) computations

3.5 Belief Propagation:

Belief propagation, also known as sum-product message passing, is a message-passing algorithm for performing inference on graphical models, such as Bayesian networks and Markov random fields. It calculates the marginal distribution for each unobserved node, conditional on any observed nodes. Belief propagation is commonly used in artificial intelligence and information theory and has demonstrated empirical success in numerous applications including low-density parity-check codes, turbo codes, free energy approximation, and satisfiability.

The algorithm was first proposed by Judea Pearlin 1982, who formulated it as an exact inference algorithm on trees, which was later extended to polytrees. While it is not exact on general graphs anymore, it has been shown to be a useful approximate algorithm.
If $X=\{X_i\}$ is a set of discrete random variables with a jointmass function *p*, the marginal distribution of a single X_i is simply the summation of *p* over all other variables: However, this quickly becomes computationally prohibitive: if there are 100 binary variables, then one needs to sum over $2^{99} \approx 6.338 \times 10^{29}$ possible values. By exploiting the polytrees structure, belief propagation allows the marginal to be computed much more efficiently.

Algorithm:

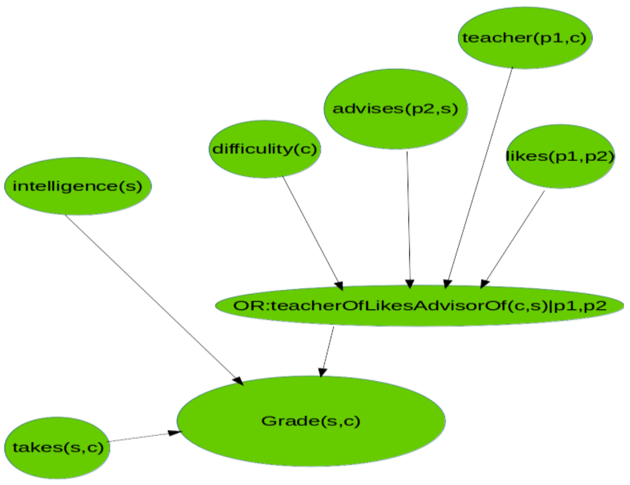
```
1.  initialize BP messages;
2.  initialize U =∅;
3.  for t=1,2…. N:
4.  run BP until convergence;
5.      choose i ∈ V\U;
6.      compute the BP marginal $v_i$ ($x_i$)
7.      choose $x_i^*$ distributed according to $v_i$;
8.      fix $x_i$ = $x_i^*$ and set U<-U ∪ $\{i\}$;
9.      add a factor I ($x_i$ =$x_i^*$) to the graphical model;
10. end
11. return $\underline{x}^*$.
```

Observations

a) Graph with smaller number of dependencies.

We have curated a sample example with defined declarations and evidence and tried to test run the same using different algorithms. The Comparison of existing algorithms based on time as given in Table 1.

Declarations:
type Professor;
type Student;
type Course;
type Grade;
guaranteed Grade None, A, B, C, D, F;
type DifficultyLevel;
guaranteed DifficultyLevel Easy, Hard;
type IntelligenceLevel;
guaranteed IntelligenceLevel Weak, Average, Smart;
random Boolean teaches (Professor, Course);
random Boolean advises (Professor, Student);
random Boolean takes (Student, Course);
random Boolean likes (Professor, Professor);
random DifficultyLevel difficulty(Course);
random IntelligenceLevel intelligence(Student);
random Grade (Student, Course);
random Boolean teacherOfLikesAdvisorOf (Course, Student);



advises (Moriarty, John) = True
grade (John, Stat10) = C
grade (Mary, Phil80) = A
grade (Fred, Stat10) = C

Fig. 1 Example (small no of dependencies)

PREDICATE LOGIC:

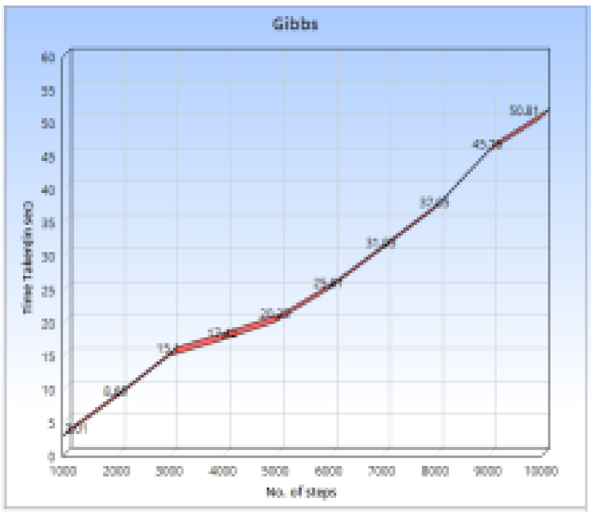
// everybody likes him- or herself likes(p,p).

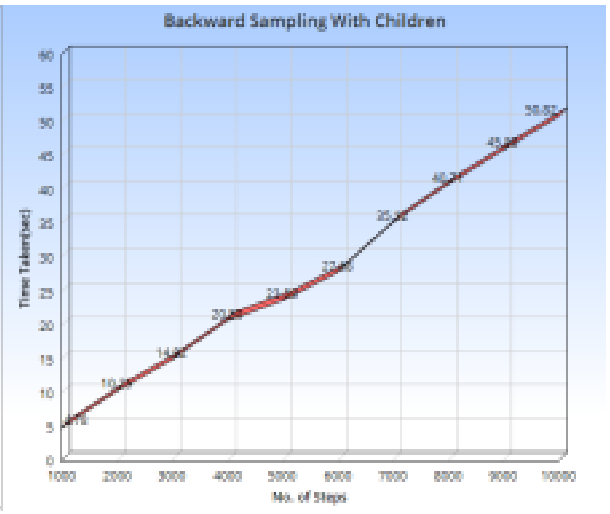
EVIDENCE:

takes (John, Stat10) = True
takes (John, CS106) = True
takes (Mary, Phil80) = True
takes (Mary, CS106) = True
takes (Fred, Stat10) = True
takes (Fred, CS106) = True
teaches (Smith, CS106) = True
teaches (Jones, CS106) = True
teaches (Moriarty, Phil80) = True
teaches (Jones, Stat10) = True

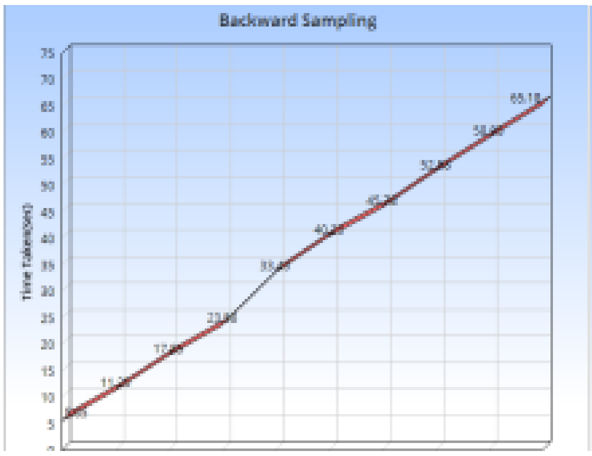
Table 2: Comparison on basis of time taken

S. No	Inferencing Algorithm	Time Taken (Sec)
1	GIBBS	50.81
2	Backward Sampling with Children	50.82
3	Backward Sampling	65.18
4	Sample Score	75.00
5	Sample Search	85.78
6	Likelihood	85.93
7	Sample Search with backjumping	88.20
8	MC-SAT	129.00
9	Variable Elimination	230.00
10	Belief propagation with junction tree	301.00
11	Gaussian	319.00

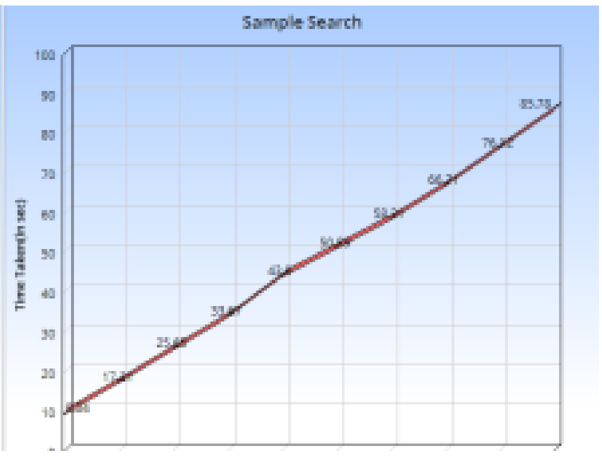




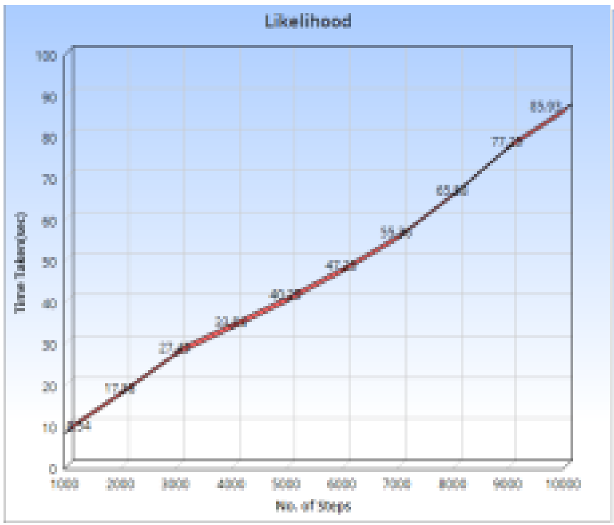
2(a): GIBBS 2(b): Backward Sampling with Children



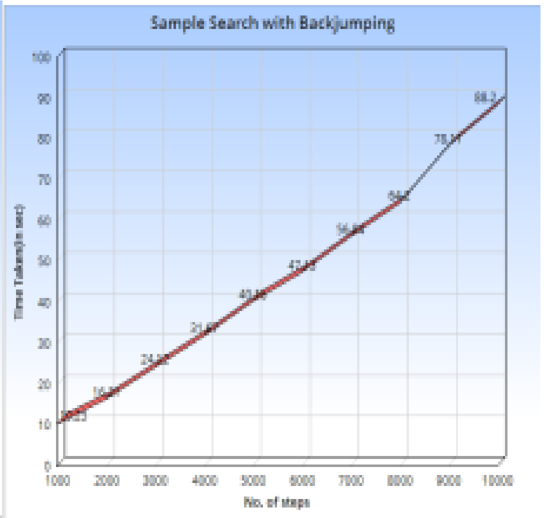
2(c): Backward Sampling



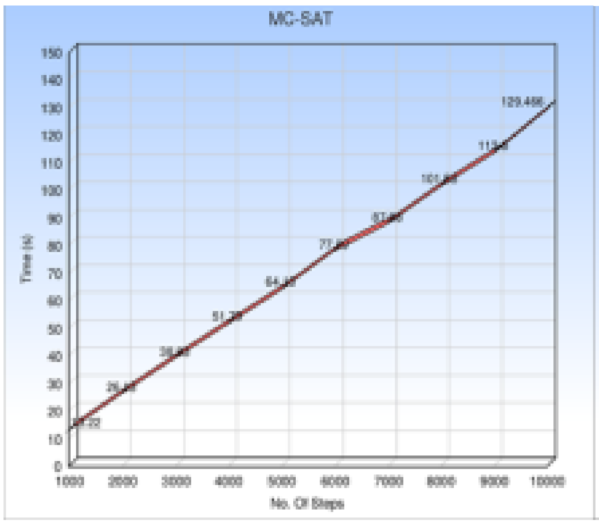
2(d): Sample Search



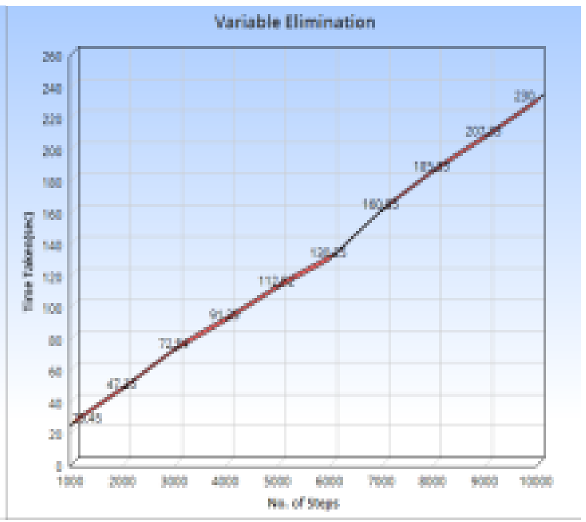
2(e): Likelihood



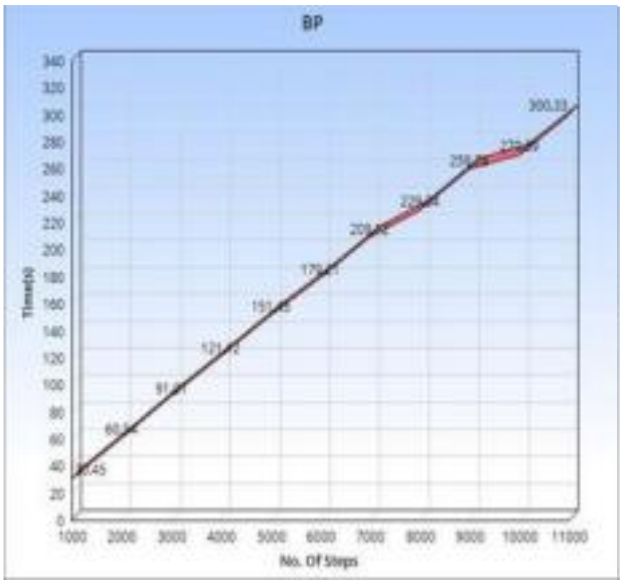
2(f): Sample Search with Back jumping

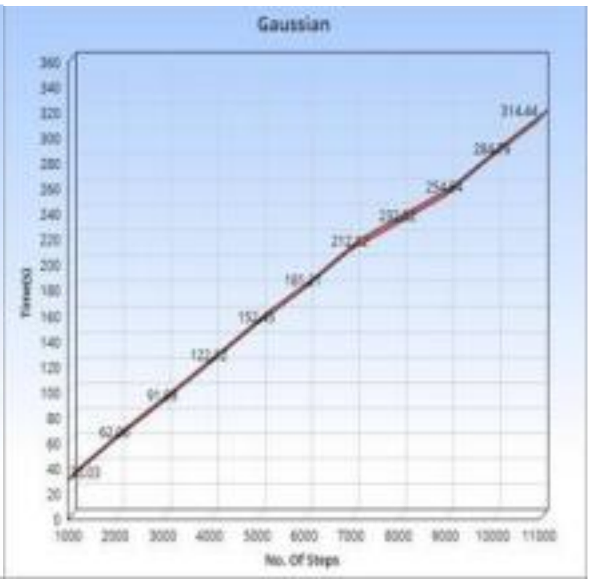


2(g): MC-SAT



2(h): Variable Elimination





2(i):

2(j)

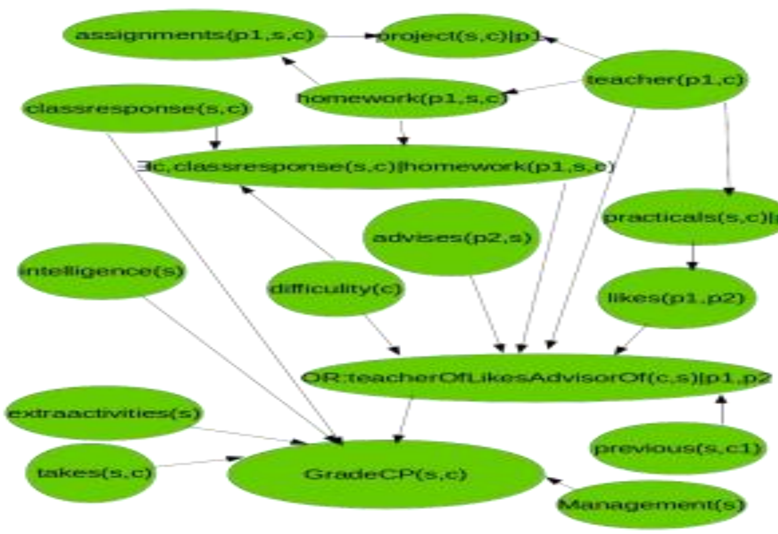


Fig 2(a)-2(j): Graphical Representation for Time taken by inference Algorithms

b) Graph with larger number of dependencies.

Declarations:

type Professor;
type Student;
type Course;
type project
guaranteed Project Completes, Not Completes
type assignments
guaranteed Project Completes, Not Completes
type class response
guaranteed Class Response Good, Bad
type practicals
type extraactivities
type management
type homework
type looks
type Grade;

```
guaranteed Grade None, A, B, C, D,
F;
type DifficultyLevel;
guaranteed DifficultyLevel Easy,
Hard;
type IntelligenceLevel;
guaranteed IntelligenceLevel Weak, Average, Smart;
random Boolean project(s)
random Boolean assignments (s, c)
random Boolean classresponse(s)
random Boolean practicals (s, c)
random extraactivities(s)
random Boolean management(s)
random Boolean homework (s, c)
random looks(s)
random Boolean teaches (Professor, Course);
random Boolean advises (Professor, Student);
random Boolean takes (Student, Course);
random Boolean likes (Professor, Professor);
random DifficultyLevel difficulty(Course);
random IntelligenceLevel intelligence(Student);
random Grade (Student, Course);
random Boolean teacherOfLikesAdvisorOf (Course, Student);
```

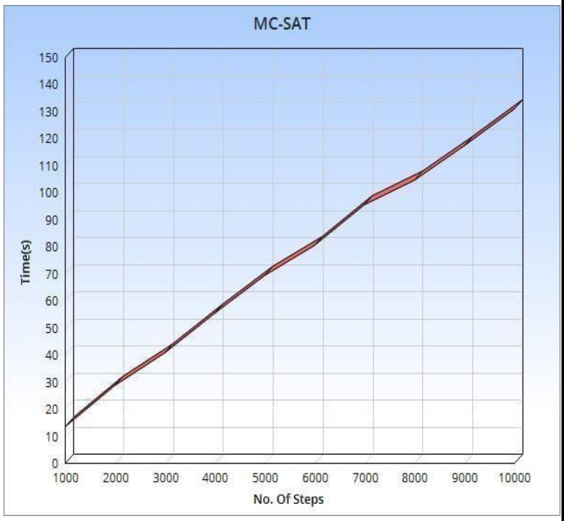
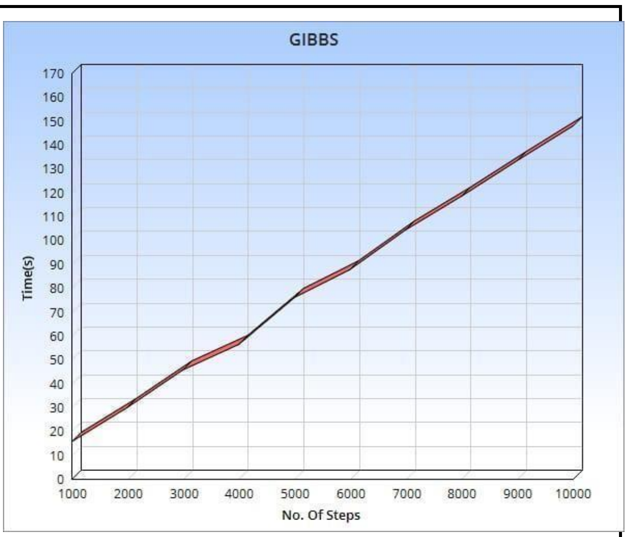
Fig 3: Example (Large no of dependencies)

```
PREDICATE LOGIC:
// everybody likes him- or herself
likes (p, p).
```

OUTCOMES

We don’t really observe any significant results in another algorithm but in this complex scenario MC-SAT outperforms Gibbs as shown in Fig 4(a) and 4(b).

MC-SAT	GIBBS

 Fig-4. (a)	 Fig. 4 (b)
131 secs	**148 secs**

Conclusion& Future Work:

Daily day to day life problems require certain level of reasoning and inferencing. With the large data being available deriving important relationship and dependency among them is a crucial step in learning, thus importance of highly efficient inferencing algorithms has increased which can infer with the minimum resources and time. Various algorithms have been proposed for the same and on the basis of derived results we can be sure that inferencing algorithms are progressing rapidly. All theinferencing algorithms are tested on this open source software called ProbCog it's coded with a mix of java and python, backend is java based and front end is python based though all the algorithms were available still we had go through modifying these algorithms according to our example. Introduction of slice sampling in MC-SAT increased its current performance under complex multi-entity relationship and thus it outperforms all other algorithms on the given scenario. As we see for simple example Gibbs provide best time complexity. Taking only 55 seconds and MC-SAT taking much more time than that. But as the no. Of instances increases and complexity increases the time taken increases in Gibbs because it needs to find conditional probability for every sample present. Both use MCMCwhich works on this procedure but in MC-SAT slice sampling is the new word introduced which basically breaks down the problem into simple parts. Then we find approximate values of it and then combine it at the end to get the final outcome. Hence in mc sat iterating over every instance at once is avoided which in return saves a bit of time. Hence in complex scenario Mc-Sat takes 131 seconds to complete whereas Gibbs takes 148 seconds.

Directions for future work include generation of a plug-in to include MC-Sat in unbbayes framework, which further will improve and reduce the problem of scalability in MEBN.

References:

[1] Markov Logic Networks by - Matthew Richardson and Pedro DomingosBacchus, F. (1990). Representing and reasoning with probabilistic knowledge.

[2] Cambridge, MA: MIT Press. Bacchus, F., Grove, A. J., Halpern, J. Y., &Koller, D. (1996). From statistical knowledge bases to degrees of belief. Artificial Intelligence, 87, 75–143.

[3] Bergadano, F., &Giordana, A. (1988). A knowledge-intensive approach to concept induction. Proceedings of the Fifth International Conference on Machine Learning (pp. 305–317).

[4] Ann Arbor, MI: Morgan Kaufmann.Berners-Lee, T., Hendler, J., &Lassila, O. (2001).

[5] The Semantic Web. Scientific American, 284 (5), 34–43. Besag, J. (1975). Statistical analysis of non-lattice data.

[6] The Statistician, 24, 179–195. Buntine, W. (1994). Operations for learning with graphical models Journal of Artificial Intelligence Research, 2, 159–225.

[7] Scalability and Knowledge Reusability in Ontology Modeling by-Mustafa Jarrar and Robert Meersman. Sound and Efficient Inference with Probabilistic and Deterministic Dependencies by- Hoifung Poon Pedro Domingos.

[8] A General Method for Reducing the Complexity of Relational Inference and its Application to MCMC by-Hoifung Poon Pedro Domingos Marc Sumner.

[9] PR-OWL - A Language for Defining Probabilistic Ontologies by-Rommel N. Carvalho, Kathryn B. Laskeyb, Paulo C. G. Costab.

[10] Handling Uncertainty in Ontology Construction Based on Bayesian Approaches: A Comparative Study by-FoniAgusSetiawan, WahyuCatur Wibowo1, and Novita Br Ginting. [11] On Lifting the Gibbs Sampling Algorithm-Deepak Venugopal Department of Computer Science The University of Texas at Dallas Richardson, TX, 75080, USA

[12] PR-OWL - A Language for Dening Probabilistic OntologiesI-By Rommel N. Carvalhoa, Kathryn B. Laskeyb, Paulo C. G. Costab.

[13] Allen, D. &Darwiche, A. 2003. New advances in inference by recursive conditioning. In UAI-03.

[14] Bartels, C. &Bilmes, J. 2004. Elimination is not enough: Non-minimal triangulations for graphical models. Technical report UWEETR-2004-00010, Univ. of Washington.

[15] Damien, P.; Wakefield, J.; Walker, S. 1999. Gibbs sampling for Bayesian non-conjugate and hierarchical models by auxiliary variables. Journal of the Royal Statistical Society B, 61:2.

[16] Dechter, R. &Mateescu, R. 2004. Mixtures of deterministic-probabilistic nets and their search space. In UAI-04. Dietterich, T.; Getoor, L.; Murphy, K., eds. 2004.

[17] Proc. ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields.

[18] IMLS. Gilks, W. R.; Richardson, S.; Spiegelhalter, D. J., eds.1996. Markov Chain Monte Carlo in Practice. Chapman and Hall.

[19] Kautz, H.; Selman, B.; Jiang, Y. 1997. A general stochastic approach to solving problems with hard and soft constraints. In The Satisfiability Problem: Theory and Applications.

[20] AMS. Kok, S.; Singla, P.; Richardson, M.; Domingos, P. 2005. The Alchemy system for statistical relational AI. http://www.cs.washington.edu/ai/alchemy/.