Distributed Adaptive Importance Sampling on Graphical Models Using MapReduce

Ahsanul Haque, Swarup Chandra and Latifur Khan
The University of Texas at Dallas
Richardson TX, USA
Email: (ahsanul.haque, swarup.chandra, lkhan)@utdallas.edu

Charu Aggarwal
IBM T. J. Watson Research Center
Yorktown NY, USA
Email: charu@us.ibm.com

***Abstract*—In the case of a graphical model, machine learning algorithms used to evaluate a query can be broadly classified into exact and approximate inference algorithms. Exact inference algorithms use only network parameters to evaluate a query. However, these algorithms are typically intractable on large networks due to exponential time and space complexity. Approximate inference algorithms are widely used in practice to overcome this constraint, with a trade-off in accuracy. It includes sampling and propagation-based algorithms. These approximate algorithms may also suffer from scalability issues if applied on large networks, for achieving higher accuracy. To address this challenge, we have designed and implemented several MapReduce-based distributed versions of a specific type of approximate inference algorithm called Adaptive Importance Sampling (AIS). We compare and evaluate the proposed approaches using benchmark networks. Experimental results show that our proposed approaches achieve significant scaleup and speedup compared to the sequential method, while achieving similar accuracy asymptotically.**

Keywords—Adaptive Importance Sampling; Approximate Inference; MapReduce

I. INTRODUCTION

Probabilistic graphical model [1] is a representation of data distribution that concisely embodies stochastic relationships of a data generating process. Many real world applications [2] such as speech recognition, video analysis, bioinformatics, etc., using graphical model involve large amounts of data. These data may not be completely utilized for analytics which incorporate traditional inference and learning methods. Recently, many parallel and distributed architectures [3] have been proposed, commonly referred to as “BigData Frameworks”, which provide an efficient way to handle a large amount of data to perform analytical computations. Therefore, it is imperative to study and develop techniques involving graphical model computations using these technologies.

Inference algorithms on graphical models can be broadly classified into exact and approximate methods. Exact inference on a simple model provides accurate results. However, they are known to be resource intensive and quickly become computationally intractable when used on large complex graphs [4]. Approximate inference algorithms are instead used to evaluate queries within practical time limits. Sampling-based inference methods are one such class of algorithms that asymptotically approximates a data distribution using samples generated from a known distribution. In particular, *Adaptive Importance Sampling* (AIS) [5] is a technique in which a known distribution is periodically updated using a set of weighted samples, accelerating convergence towards the data distribution. This algorithm is used to reduce variance, especially when larger variance may be caused by rare events.

Applications of importance sampling or AIS techniques include variance reduction in communication systems ([6]), market evaluation [7], aeronautics [8], etc. These applications performing predictions and simulations typically involve large datasets. In addition to these large datasets, use of AIS over complex graph structures may also demand large computational resources [9]. With the use of existing parallel and distributed computational frameworks, these challenges can be addressed adequately. In this paper, we exploit the nature of sample independence and weight computation in AIS, and propose two MapReduce based approaches for performing faster inference over a given graphical model. Our experimental evaluation shows that the proposed approaches achieve substantial gain in scaleup and speedup compared to the sequential execution of AIS.

Contributions of this paper are as follows: 1) We design two distributed approaches that exploit inherent parallelism in a version of Adaptive Importance Sampling (AIS). In particular, we use the Rao-Blackwellized method [10] which combines sampling and Variable Elimination techniques to calculate probability of evidence, or partition function in AIS. 2) We implement all the proposed approaches using the MapReduce framework. 3) We evaluate these approaches using various benchmark graphical networks, and compare its effectiveness in terms of speedup and scaleup, with sequential approach over these networks.

The paper is organized as follows: In Section II, we provide some background information on probabilistic graphical models and sampling algorithms. In Section III, we point out certain design challenges in distributed AIS, and present our proposed approaches. Section IV provides the experimental results and shows comparison of performance among our proposed distributed approaches and the sequential execution. In Section V, we discuss some related works. Finally, Section VI concludes the paper.

II. BACKGROUND

In this section, we provide a background on exact and approximate inference methods including sampling techniques, over probabilistic graphical models.

A. Probabilistic Graphical Models

A probabilistic graphical model $\mathcal{G}$ is a collection of functions over a set of random variables $\mathbf{X} \in \boldsymbol{\mathcal{X}}$. It is generally represented as a network of nodes, each node denoting a data feature and edges denoting feature relationships. A directed acyclic graph, called a *Bayesian* network, captures conditional independence between these random variables. An undirected graph is called a *Markov* network where non-negative real-valued functions (or factors ϕ) are defined over cliques in the network. The joint distribution $P_{\mathcal{G}}$ [11] is given by-

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \prod_{k} \phi(x_{\{k\}}), \text{ where } Z = \sum_{\mathbf{x} \in \boldsymbol{\mathcal{X}}} \prod_{k} \phi(x_{\{k\}}) \tag{1}$$

where Z is the partition function, and $x_{\{k\}}$ is the state of the variable $x \in \mathbf{x}$ appearing in the k^{th} clique.

Probability of evidence or partition function Z is computed using inference algorithms [1]. Exact inference can be used for evaluating Z accurately. *Variable Elimination* (VE) is one such exact inference algorithm where non-query variables, i.e., $\mathbf{X} \setminus \mathbf{x}_e, \mathbf{x}_q$ are eliminated from the evidence instantiated network $\mathcal{G}_e$ in a certain order during query evaluation. Here, $\mathbf{x}_e$ represents a set of evidence variables, and $\mathbf{x}_q$ represents set of query variables. Both time and space complexity of running Variable Elimination (VE) on a graph is exponential in its *treewidth* and is known to be computationally intractable on large complex networks [4]. This motivates the use of approximate inference algorithms.

B. Sampling

Sampling-based approximate inference algorithms are used to approximate the posterior $\mathcal{P}$ with an empirical estimation from samples [1]. A sample is an instantiation (or assignment) of variables. Let $\mathcal{F}$ be a function of $\mathbf{x}$. A set of samples are used for estimating its expectation $\widetilde{F}$ with regard to the distribution $\mathcal{P}$ given by-

$$\widetilde{\mathcal{F}}(\mathbf{X}) = \sum_{i=0}^{N} \mathcal{F}(\mathbf{x}_i) * \mathcal{P}(\mathbf{x}_i) \tag{2}$$

In the case of Markov Chain Monte Carlo (MCMC) methods such as Gibbs sampling [12], samples are generated directly from a given probability distribution $\mathcal{P}$ represented by the network. However, this estimation may result in large variance of expected value due to occurrence of rare events. Instead, *Importance Sampling* uses another distribution, called a *proposal distribution* $\mathcal{Q}$ over which sampling is performed [1]. The proposal distribution is initialized randomly such that $\mathcal{Q}(x) > 0$ when $\mathcal{P}(x) > 0$, where $x \in \mathbf{X}$. The expectation in this case is given by-

$$\widetilde{\mathcal{F}}(\mathbf{X}) = \sum_{i=0}^{N} \mathcal{F}(\mathbf{x}_i) * \frac{\mathcal{P}(\mathbf{x}_i)}{\mathcal{Q}(\mathbf{x}_i)} \tag{3}$$

Variance can be further reduced by choosing $\mathcal{Q}$ similar to $\mathcal{P}$. However, this is not always possible since the distribution $\mathcal{P}$ might be unknown or complex. In such cases, an adaptive strategy is used where the proposal distribution $\mathcal{Q}$ is updated periodically using a set of empirically generated samples. This iterative process is known as *Adaptive Importance Sampling* (AIS) [5].

A variant of AIS that combines sampling and exact inference, is known as *Rao-Blackwellized AIS* (RB-AIS). Here, a subset of variables $\mathbf{W} \subset \mathbf{X} \setminus \mathbf{x}_e$ (called w-cutset [13]) are heuristically chosen. Samples generated from these variables are then conditioned on $\mathcal{G}_{\mathbf{e}}$, effectively reducing the treewidth of the resultant network $\mathcal{G}_{\mathbf{ew}}$, where $\mathbf{w}$ is an instantiation of variables in $\mathbf{W}$. This makes the network feasible to perform exact inference over $\widehat{\mathbf{Y}} = \mathbf{X} \setminus \mathbf{W}, \mathbf{x}_e$ for sufficiently large $\mathbf{W}$.

In this paper, we use the Rao-Blackwellized version of Adaptive Importance Sampling, by choosing an initial $\mathcal{Q}$ (uniform distribution) [14] over a set of discrete random variables $\mathbf{X}$ used to define the given distribution. Exact inference over $\mathcal{G}_{\mathbf{ew}}$ is performed using VE, where min-degree heuristic is used to obtain an ordering of variables. The subset of variables $\mathbf{W}$ is chosen such that the most occurring variable in a tree decomposition [1] is removed iteratively from all clusters to reduce the width of ordering to a desired value of $w + 1$. This desired value is set based on the trade-off between required amount of resources and quality of the estimate measured in terms of resulting accuracy. Smaller value of w may result in larger w-cutset size $|\mathbf{W}|$. This provides quicker evaluation with lower accuracy as VE is executed over a smaller set of variables. On the contrary, larger w typically results in smaller $|\mathbf{W}|$ with a higher accuracy since VE is executed over a larger set of variables. For instance, $|\mathbf{W}| = 0$ implies that no variables are considered for sampling, and VE is executed over the entire network $\widehat{\mathbf{Y}} = \mathbf{X} \setminus \mathbf{x}_e$, resulting in an accurate answer with larger execution time.

Samples on w-cutset variables are then weighed using the following equation-

$$\psi_{\mathbf{x}} = \frac{VE(\mathcal{G}_{\mathbf{ew}} | \mathbf{X} = \mathbf{x})}{\mathcal{Q}(\mathbf{X} = \mathbf{x})} \tag{4}$$

where $VE(\mathcal{G})$ is the result of VE on graph $\mathcal{G}$. This weight is computed for each sample, and is used to update $\mathcal{Q}$ periodically after every n weight calculations using-

$$\mathcal{Q}(X = x) = \frac{\sum_{t=1}^{n} \delta_{\mathbf{x}^t}(X = x) \psi_{\mathbf{x}^t}}{\sum_{t=1}^{n} \psi_{\mathbf{x}^t}} \tag{5}$$

Here $\mathbf{x}^1 \ldots \mathbf{x}^n$ are the sets of samples generated whose weights are used for updating $\mathcal{Q}$. Larger value of n typically results in quicker convergence of the algorithm [5]. δ is the Dirac function. The sum-accumulated weight ψ_{acc} is averaged over these samples to estimate the probability of evidence as $Z = \frac{\psi_{acc}}{N}$.

III. Proposed Approaches

In this section, we discuss certain design challenges and present our proposed approaches.

A. Design Challenges

The two main challenges in designing a parallel version of RB-AIS are sampling and corresponding weight computation for estimating new proposal distribution.

Parallel sampling in a distributed environment can be categorized into two types. One involves associating all graph nodes to each process, i.e. duplicating the network in each process, and the other involves associating one graph node per process. The former approach produces complete samples in each process. This is typically used when sampling variables under conditional independence assumption, such as in Gibbs sampling. However this process is known to have high mixing time [15]. The latter approach can be used when assuming that samples from each variable are independent of other variables. In RB-AIS, samples from the w-cutset variables are instantiated on the network to form a graphical model with bounded treewidth. Variables in w-cutset are systematically chosen to form independent elements in the set. Therefore, we can independently generate samples from these variables (called partial sample) by distributing the variables across multiple processes, or use a single process to sample all w-cutset variables (called complete sample).

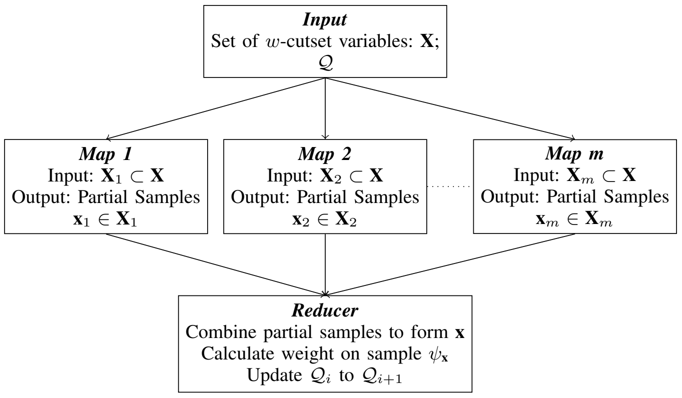
RB-AIS computes a weight for each complete sample. These weights are used to update the proposal distribution at the end of each iteration. This requires a method to combine partial samples or complete sample weights. Consequently, a parallel approach should consist of a synchronization method between processes for combining samples or sample weights at every iteration.

These challenges can be addressed to a significant extent by using the MapReduce framework, which has an inherent distributed and fault-tolerant mechanism along with a method to combine results of partial computations from multiple processes. We propose an approach called *Distributed Sampling in Mappers* (DSM) that leverages the *Mapper* function to perform parallel sampling, and a *Reduce* function to combine samples for weight computation and estimation of new proposal distribution $\mathcal{Q}$. However, the major resource intensive task in RB-AIS is the weight calculation procedure performed on each complete sample since this involves analytical computations i.e. Variable Elimination. Parallel sampling alone may not effectively employ the distributed architecture of MapReduce if weight calculation is performed using a single process. Therefore, we propose another approach called *Distributed Weight Calculation in Mappers* (DWCM) which computes sample weights in a distributed fashion using the *Mapper* functions. Here, the *Reducer* function is only used to combine sample weights and estimate a new proposal distribution $\mathcal{Q}$.

The proposed approaches use a set of MapReduce jobs to form a chain, each job performing computations of a single iteration in RB-AIS. For example, the i^{th} MapReduce job generates n samples. These are used for updating $\mathcal{Q}_i$ at the end of the job to produce $\mathcal{Q}_{i+1}$. The output of the i^{th} MapReduce job is given as input to the $(i+1)^{th}$ MapReduce job. A final distribution $\mathcal{Q}_I$ and an estimate of Z are obtained at the end of the chain execution.

B. Distributed Sampling in Mappers (DSM)

An overview of DSM is depicted in Figure 1. In this approach, w-cutset variables are distributed among available Mappers in each MapReduce job for sampling. The Mapper functionality detailed in Algorithm 1 shows the input key index as one of the assigned variable X, with $\mathcal{Q}[X]$ as its associated value. Each Mapper generates n random samples (or partial samples) from each of its assigned variables (line 3), which are emitted as output (line 4). Each Mapper output consists of sample index $s \in \{1 \dots n\}$ as a key with a tuple consisting of variable index X, sample x and $\mathcal{Q}[X = x]$, as value.


Input
Set of w-cutset variables: $\mathbf{X}$; $\mathcal{Q}$
Map 1
Input: $\mathbf{X}_1 \subset \mathbf{X}$
Output: Partial Samples $\mathbf{x}_1 \in \mathbf{X}_1$
Map 2
Input: $\mathbf{X}_2 \subset \mathbf{X}$
Output: Partial Samples $\mathbf{x}_2 \in \mathbf{X}_2$
Map m
Input: $\mathbf{X}_m \subset \mathbf{X}$
Output: Partial Samples $\mathbf{x}_m \in \mathbf{X}_m$
Reducer
Combine partial samples to form $\mathbf{x}$
Calculate weight on sample $\psi_{\mathbf{x}}$
Update $\mathcal{Q}_i$ to $\mathcal{Q}_{i+1}$
Fig. 1: Distributed Sampling in Mappers (DSM)

Algorithm 1: Pseudocode for Mapper in DSM

```
input : $X$ as key; $(\mathcal{Q}[X], \omega[X])$ as value
output: partial samples on $X$
1 begin
      // $-1$ is special key for $Z$
2     if $X! = -1$ then
3         $List[x] \leftarrow$ Generate partial samples on $X$;
4         Emit partial samples stored in $List[x]$ ;
5     else
6         Emit $(-1, Z)$;
```

As depicted in Figure 1, a single Reducer is used to combine the partial samples from each Mapper according to the sample index s (or $sampleIndex$), to form a set of n complete samples. Algorithm 2 details the Reducer operation. Here, partial samples having sample index s are combined to form a complete sample $\mathbf{x}$, as shown in line 4. Weight $\psi_{\mathbf{X}}$ for $\mathbf{x}$ is computed using VE in line 5, which is then used for updating the probability of evidence Z in line 6. Note that $\mathcal{G}$ is required to perform VE at the Reducer. Therefore, we place it in the $Distributed\ Cache$. After calculating weights for all samples, a new proposal distribution $\mathcal{Q}$ is estimated from ψ at line 2 of the $cleanup$ procedure, according to equation 5. Since the next MapReduce job in the chain requires w-cutset variables as input, the Reducer in this MapReduce job emits $< X, (\mathcal{Q}[X], \omega[X]) >$ as key-value pairs for all w-cutset variables $X \in \mathbf{W}$.

In addition, the value of Z is propagated through the MapReduce chain and is updated at the Reducer in each iteration. We use a special key of -1 to differentiate the value of Z from other key-value pairs.

C. Distributed Weight Calculation in Mappers (DWCM)

Instead of distributed sampling and sequential weight computation in DSM, we propose a sequential sampling and distributed weight computation method in this approach. Figure 2 depicts a high level architecture for DWCM. Applications of distributed RB-AIS may involve complex networks having

Algorithm 2: Pseudocode for Reducer in DSM

```
input : intermediate key value pairs from Mappers
output: key value pairs where key is $X$ and value is
        $(\mathcal{Q}[X], \omega[X])$
1 Procedure reduce
2     $sampleIndex \leftarrow$ Extract sample index from $key$;
      // −1 is special key for $Z$
3     if $sampleIndex$ != -1 then
4         $\mathbf{x} \leftarrow$ Combine partial samples from $List[value]$;
5         Calculate $\psi_{\mathbf{x}}$ ;
6         Update $\boldsymbol{\psi}$ and $Z$ based on $\psi_{\mathbf{x}}$ ;
7     else
8         Load $Z$ from $List[values]$ ;

1 Procedure cleanup
2     Update $\omega$ and $\mathcal{Q}$ based on $\boldsymbol{\psi}$ ;
3     For each $X \in \mathbf{X}$, Emit $(X, (\mathcal{Q}[X], \omega[X]))$ ;
4     Emit $(-1, Z)$;
```

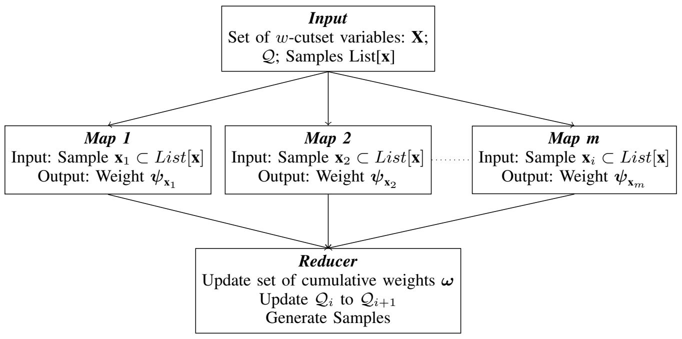

Input
Set of w-cutset variables: $\mathbf{X}$;
$\mathcal{Q}$; Samples List[$\mathbf{x}$]
Map 1
Input: Sample $\mathbf{x}_1 \subset List[\mathbf{x}]$
Output: Weight $\boldsymbol{\psi}_{\mathbf{x}_1}$
Map 2
Input: Sample $\mathbf{x}_2 \subset List[\mathbf{x}]$
Output: Weight $\boldsymbol{\psi}_{\mathbf{x}_2}$
Map m
Input: Sample $\mathbf{x}_i \subset List[\mathbf{x}]$
Output: Weight $\boldsymbol{\psi}_{\mathbf{x}_m}$
Reducer
Update set of cumulative weights $\boldsymbol{\omega}$
Update $\mathcal{Q}_i$ to $\mathcal{Q}_{i+1}$
Generate Samples

Fig. 2: Distributed Weight Calculation in Mappers (DWCM)

Algorithm 3: Pseudocode for Mapper in DWCM

```
input : $sampleIndex$ as key; $(\mathbf{x}, \mathcal{Q}(\mathbf{X} = \mathbf{x}))$ as value
output: $weightVal$ as key; $(\mathbf{x}, \psi_{\mathbf{x}})$ as value
1 Procedure map
2     $sampleIndex \leftarrow$ Extract sample index from $key$;
3     if $sampleIndex > 1$ then
4         Calculate $\psi_{\mathbf{x}}$ ;
5         Emit $(weightVal, (\mathbf{x}, \psi_{\mathbf{x}}))$ ;
6     else
7         Emit($key$, $value$);
```

Algorithm 4: Pseudocode for Reducer in DWCM

```
input : intermediate key value pairs from Mappers
output: key value pairs where key is $sampleIndex$ and
        value is $(\mathbf{x}, \mathcal{Q}(\mathbf{X} = \mathbf{x}))$
1 Procedure reduce
      // −1 is special key for $Z$
2     if $key$ = -1 then
3         Load $Z$ from $List[values]$;
      // −2 is special key for $\omega$
4     else
5         Update $\boldsymbol{\psi}$ based on $\psi_{\mathbf{x}}$ ;

1 Procedure cleanup
2     Update $\omega$, $\mathcal{Q}$ and $Z$ based on $\boldsymbol{\psi}$ ;
3     Emit$(-1, Z)$;
4     $List[\mathbf{x}] \leftarrow$ Generate samples on $\mathbf{X}$;
5     For each $\mathbf{x} \in List[\mathbf{x}]$, Emit $(sampleIndex,$
      $(\mathbf{x}, \mathcal{Q}(\mathbf{X} = \mathbf{x})))$ ;
```

large treewidth. RB-AIS reduces this to a network having smaller treewidth for tractable analytical computation. In this regard, we propose to perform sequential sampling of w-cutset variables whose weight calculations are performed using parallel resources. In particular, multiple Mappers are used for calculating weights of samples. Algorithm 3 details the Mapper functionality whose input consists of an index $s \in \{1 \ldots n\}$ of a sample $\mathbf{x}$ as key, and a tuple $(\mathbf{x}, \mathcal{Q}[\mathbf{X} = \mathbf{x}])$ as value. Sample $\mathbf{x}$ here denotes a complete sample. The weight $\psi_{\mathbf{x}}$ of a sample $\mathbf{x}$ is computed using VE at line 4. Here again, $\mathcal{G}$ is placed in the $Distributed\ Cache$ since VE is executed on multiple Mappers. The output of each Mapper consists a constant denoted as $weightVal$ as key, and a tuple ($\mathbf{x}, \psi_{\mathbf{x}}$) as corresponding value.

Sample weights are combined at a single Reducer in the job, which updates $\mathcal{Q}$ and Z as shown in the figure. Algorithm 4 details the Reducer operation. $\boldsymbol{\psi}$ is updated using all sample weights at line 5 of the *reduce* procedure. In the *cleanup* procedure, the Reducer updates ω, $\mathcal{Q}$ and Z at line 2, similar to DSM. However, the next MapReduce job in the chain require samples from w-cutset variables as input. Therefore, n samples are generated at the Reducer using these variables, as shown in lines 4-5. Similar to DSM, the value of Z in DWCM is propagated through the MapReduce chain using a special key of value -1.

IV. Experimental Results

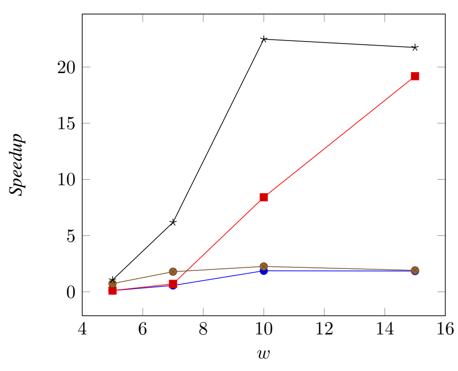
In this section, we evaluate and analyze the proposed approaches using various benchmark networks.

A. Performance Metrics

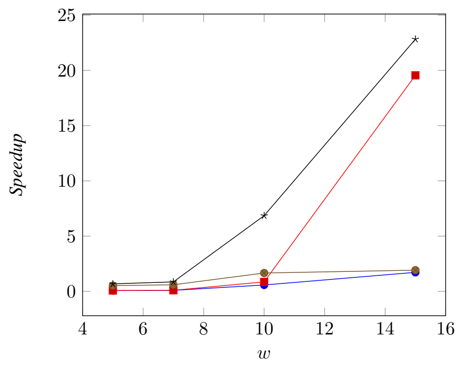
A major benefit of using a distributed approach over a sequential approach to perform any computation is improvement in overall execution time. We measure the performance of our approach using *Speedup* and *Scaleup*. Here, *Speedup* $= \frac{T_{sq}}{T_d}$ where T_{sq} and T_d are the execution time of sequential and distributed approach respectively on a particular network using a set of parameter values, and *Scaleup* $= \frac{T_s}{T_p}$ where T_s and T_p are the execution time when using a single and multiple processor or cores respectively. A higher value for both these measures indicate better performance of the distributed approach.

B. Experiment

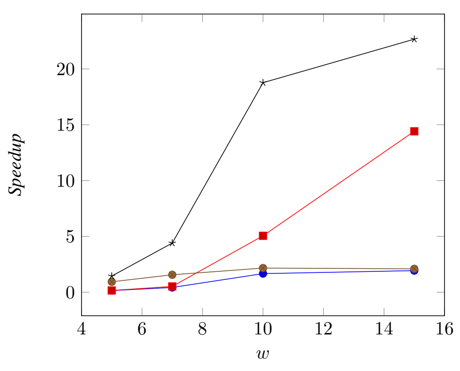
We systematically perform various experiments by varying parameter values including w in w-cutset, number of samples n per MapReduce job, and number of updates I to the proposal distribution $\mathcal{Q}$, on all the considered datasets. These experiments were performed on a Hadoop (version: 1.2.1) cluster containing 9 nodes, each with four 2.2GHz cores and 4 GB of RAM. We use publicly available benchmark



(a) 54.wcsp.uai



(b) 29.wcsp.uai

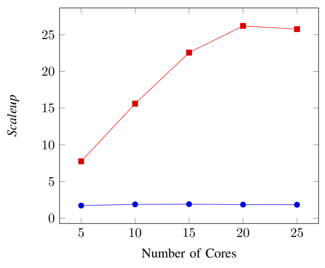


(c) 404.wcsp.uai

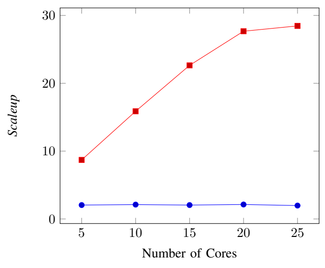
Fig. 3: *Speedup* achieved: ● DSM ($n = 100$); ■ DWCM ($n = 100$); ● DSM ($n = 1000$); ★ DWCM ($n = 1000$);

TABLE I: Datasets

Name of Network	Number of nodes	Number of factors
54.wcsp	67	271
29.wcsp	82	462
404.wcsp	100	710



(a) 54.wcsp



(b) 404.wcsp

Fig. 4: *Scaleup* with $n = 1000$: ● DSM, and ■ DWCM

datasets [16] to perform our experiments. These are listed in Table I with relevant properties that may affect the performance of distributed RB-AIS.

Figure 3 shows the speedup of DSM and DWCM approaches on each dataset, with $w = \{5, 7, 10, 15\}$ of w-cutset, number of samples per iteration $n = \{100, 1000\}$, and the total number of iterations $I = 10$. As discussed in Section III, it is expected that DWCM performs better than DSM since weight computation is a bottleneck compared to sampling in RB-AIS. From these figures, we can observe that DWCM exhibits better speedup than DSM, which supports our claim. For instance, with $w = 10$ and $n = 100$, we observe a speedup of 1.87 in DSM compared to that of 8.41 in DWCM, on the 54.wcsp network. Additionally, note that DSM performs slightly better than the sequential approach in most cases with speedup having a value close to 1. This is as expected since the total cost of sampling sequentially on the network is equivalent to the overhead cost of using the MapReduce framework. Therefore, any time gained in parallel sampling is reduced by this overhead cost.

We observe that in all cases, the performance of our approaches improved with higher w. For $w = 5$, the sequential approach performs better in most cases with speedup value less than 1. However, for $w = 15$, the distributed approach performs better than sequential approach in all cases with speedup greater than 1.7. Additionally, we also observe that distributed approaches perform better when more number of samples (n) are used for updating $\mathcal{Q}$. This can be attributed to the parallel resource utilization when compared to the sequential approach. Our empirical evaluation suggested that RB-AIS converges faster using larger w, and provides a better result with larger number of samples. However, this increases the runtime of the sequential approach to a great extent. Our approaches consistently show a better speedup for such parameter values. For example, in DWCM approach with $w = 15$ and $n = 1000$, the speedup obtained was 22.82 on 29.wcsp network and 22.66 on 404.wcsp network. Therefore, these proposed approaches can be used in situations where faster convergence is required with higher accuracy.

Figure 4 shows the scaleup achieved on both DSM and DWCM approaches for 54.wcsp and 404.wcsp networks, with $w = 15$ and $n = 1000$. A similar behavior was observed on 29.wcsp network as well (not reported due to space constraints). From these figures, it can be observed that the performance improves with an increase in number of cores. In addition, DWCM scales better than DSM as expected. In particular, DSM performs marginally better than the sequential approach with an increase in number of cores. Whereas, DWCM performs significantly better as expected with an average scaleup of 20.12 over all datasets, showing greater scalability compared to the sequential approach.

V. Related Work

There exist several studies on parallelizing inference algorithms in graphical models. Parallel approaches to exact inference have been proposed in studies such as [17]. Despite these efforts, exact inference algorithms on complex real world networks can still be computationally infeasible. Approximate inference algorithms such as Markov chain Monte Carlo (MCMC) methods, in particular Gibbs sampling [12] are popularly known to have an "embarrassingly" parallel structure [18] which can be easily exploited. Recently, a study proposes a parallel architecture for MCMC [19] using MapReduce. They distribute network variables into multiple processes, generating samples from these subposteriors in parallel. These are then combined to form a full-data posterior. We use a similar approach to perform parallel sampling over a given model. However, samples are generated from each variable independently. Further, importance sampling techniques typically used for variance reduction in certain networks, have not been studied with regards to a parallel architecture. We design and implement parallel approaches for a Rao-Blackwellised Adaptive Importance sampling algorithm (RB-AIS). In a recent work [20], we propose another approach that combines parallel sampling and distributed weight computation for RB-AIS.

Previous studies [21] list various challenges in developing parallel inference algorithms including different process speed, sample mixing time, etc. Proliferation of tools such as Hadoop [22] has provided sophisticated architectures to overcome most of these challenges. In particular, frameworks such as Pregel [23] and GraphLab [24] provide architectures for parallel computation on graph structures. These essentially adopt vertex-centric models to enable efficient message passing between nodes in a graph. Our study involves an iterative process which uses data structures over cliques in a graph. We argue that the mechanism of combining sample weights for distribution adaptation is most suitable to be implemented using MapReduce. However, frameworks such as Spark [25] can be leveraged for faster computation. We leave this for future work.

VI. Conclusion

In this paper, we design and implement distributed approaches for a Rao-Blackwellized Adaptive Importance Sampling (RB-AIS) algorithm which leverages the Hadoop MapReduce framework. One of the approaches performs parallel sampling (DSM), and the other performs parallel weight calculation (DWCM) over complex networks. We evaluate the performance of these approaches by measuring speedup and scaleup. From our experimental results, it can be seen that for large and complex networks, the distributed approach performs significantly better compared to the sequential approach in terms of scalability and computational speed without loosing accuracy, asymptotically.

Acknowledgment

This material is based upon work supported by NSF Award No. CNS-1229652 and DMS-1322353. We also thank Dr. Vibhav Gogate and Dr. Michael Baron for their valuable comments and suggestions.

References

[1] A. Darwiche, *Modeling and reasoning with Bayesian networks*. Cambridge University Press, 2009.

[2] M. I. Jordan, "Graphical models," *Statistical Science*, pp. 140–155, 2004.

[3] S. Sagiroglu and D. Sinanc, "Big data: A review," in *Collaboration Technologies and Systems (CTS), 2013 International Conference on*. IEEE, 2013, pp. 42–47.

[4] P. Dagum and M. Luby, "Approximating probabilistic inference in bayesian belief networks is np-hard," *Artif. Intell.*, vol. 60, no. 1, pp. 141–153, Mar. 1993.

[5] J. Cheng and M. J. Druzdzel, "Ais-bn: An adaptive importance sampling algorithm for evidential reasoning in large bayesian networks," *Journal of Artificial Intelligence Research*, vol. 13, pp. 13–155, 2000.

[6] D. Remondo, R. Srinivasan, V. F. Nicola, W. C. Van Etten, and H. E. Tattje, "Adaptive importance sampling for performance evaluation and parameter optimization of communication systems," *Communications, IEEE Transactions on*, vol. 48, no. 4, pp. 557–565, 2000.

[7] L. Hoogerheide and H. K. van Dijk, "Bayesian forecasting of value at risk and expected shortfall using adaptive importance sampling," *International Journal of Forecasting*, vol. 26, no. 2, pp. 231–247, 2010.

[8] J. Morio, "Non-parametric adaptive importance sampling for the probability estimation of a launcher impact position," *Reliability Engineering & System Safety*, vol. 96, no. 1, pp. 178–183, 2011.

[9] H. Guo and W. Hsu, "A survey of algorithms for real-time bayesian network inference," in *AAAI/KDD/UAI02 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*. Edmonton, Canada, 2002.

[10] G. Casella and C. P. Robert, "Rao-blackwellisation of sampling schemes," *Biometrika*, vol. 83, no. 1, pp. 81–94, March 1996.

[11] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[12] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 6, no. 6, pp. 721–741, Nov. 1984.

[13] B. Bidyuk and R. Dechter, "An empirical study of w-cutset sampling for bayesian networks," in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'03. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 37–46.

[14] V. Gogate and R. Dechter, "Approximate inference algorithms for hybrid bayesian networks with discrete constraints." in *UAI*. AUAI Press, 2005, pp. 209–216.

[15] J. Gonzalez, Y. Low, A. Gretton, and C. Guestrin, "Parallel gibbs sampling: From colored fields to thin junction trees," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 324–332.

[16] "The probabilistic inference challenge (pic2011)," http://www.cs.huji.ac.il/project/PASCAL/showNet.php, 2011, last updated on 07.08.2014.

[17] Y. Xia and V. K. Prasanna, "Junction tree decomposition for parallel exact inference," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1–12.

[18] J. Corander, M. Gyllenberg, and T. Koski, "Bayesian model learning based on a parallel mcmc strategy," *Statistics and Computing*, vol. 16, no. 4, pp. 355–362, Dec. 2006.

[19] W. Neiswanger, C. Wang, and E. P. Xing, "Asymptotically exact, embarrassingly parallel mcmc." *CoRR*, vol. abs/1311.4780, 2013.

[20] A. Haque, S. Chandra, L. Khan, and M. Baron, "Mapreduce guided approximate inference over graphical models," in *Symposium Series on Computational Intelligence*. IEEE, 2014, to appear.

[21] J. E. Gonzalez, Y. Low, C. Guestrin, and D. O'Hallaron, "Distributed parallel inference on large factor graphs," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 203–212.

[22] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–10.

[23] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 135–146.

[24] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: a framework for machine learning and data mining in the cloud," *Proceedings of the VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.

[25] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 2010, pp. 10–10.