Parallelizing Exact Inference in Bayesian Networks

Vasanth Krishna Namasivayam, Animesh Pathak and Viktor K. Prasanna
Department of Electrical Engineering, University of Southern California
{namasiva, animesh, prasanna}@usc.edu

Introduction

Exact inference is an approach for computing probabilities in Bayesian networks. Recently, this has been studied by various teams in the DARPA ACIP program. The popular Lauritzen Speigelhalter algorithm [2] converts a given Bayesian Network to a Junction Tree representation and then computes the exact inference on the same. The reason for this intermediate tree representation is that the commonly used inference algorithms will give erroneous results for Bayesian networks which have undirected cycles, whereas they can be easily modified and applied to the resultant junction tree. In this work, we present a parallel implementation of the exact inference problem on a general Bayesian Network. Our experiments show that the implementation scales well with increasing number of processors while the Intel PNL library [7] does not scale.

Background

Bayesian Networks: A Bayesian network [1] is represented by a directed acyclic graph (DAG), with each node representing a random variable. An edge between two nodes indicates a relation between the variables and the direction indicates the causality. If a node has a known value, it is said to be an *evidence node*. The joint distribution for each variable is a function of all its parents and is stored as a conditional probability table (CPT).

Exact Inference: Exact inference in a Bayesian network involves determining the probabilities of the query variables, given the exact state of the evidence variables. When we get new information about variables in the network, we update the conditional probability tables to reflect this new information. This updating is known as evidence propagation [2]. Once all the beliefs are updated, the conditional probability tables contain the most recent beliefs in any variable and can be queried like a simple database to evaluate probabilities.

Junction Trees: A *junction tree* [6] of a G is a tree J such that each maximal clique C of G is a node in J, and all the cliques represented by the nodes of the junction tree J satisfy the running intersection property [4]. Each edge in J is labeled with the intersection of the cliques represented by its bordering vertices. These labels are called *separator sets* or *sepsets*. The *clique width* of a junction tree J is defined as the maximum number of random variables in the clique represented by any node in J.

Our Parallel Algorithm

We assume a CREW PRAM model of computation and use adjacency matrices to represent graphs. Starting with a Bayesian network G with maximum degree k and clique width w $(=k+1)$, our parallel algorithm for p processors proceeds in the following stages:

Moralization: A *moral graph* of a DAG G is an undirected graph where all the parents of each node in G form a clique. We first moralize G in parallel, spending $O(n^2/p + nk^2/p)$ time, resulting in the graph G_{mor}.

Triangulation and Identifying Maximal Cliques: An undirected graph is *triangulated* iff every cycle of length four or greater contains a chord between any two non adjacent nodes in the cycle. To triangulate G_{mor}, we take its vertices one by one, connecting all its children and parents to each other. The vertices are chosen in increasing order of degree, and we perform this sorting in parallel in log p time. The cliques thus formed are inserted in to the set of vertices C of the Junction tree if they are maximal. In our parallel algorithm, we combined the triangulation step with the maximal clique selection step. The total time complexity of this stage is $O(n^3/p + n\log p + n^2 + nw^2)$.

Constructing the Junction Tree: After identifying the cliques C_i, we proceed to connect them to obtain the junction tree. From the *running intersection property*, for every j, there exists an $i < j$ such that:

$$C_j \cap (C_1 \cup C_2 \cup \ldots \cup C_{j-1}) \subseteq C_i$$

For every j, we choose one i for which this property holds, and connect C_j to C_i [5]. Finally, for each edge in this tree J, we compute the intersection of the cliques bordering the edge and insert it as a *separator set node* between them. Our parallel algorithm for this stage takes $O(wn^2/p)$ time.

Potential Table Calculation: In this step, we reintroduce the information about the relation between the variables by computing potential tables for each clique C_i from the information originally stored in G. Since each node in J represents a Clique of w nodes in the original Bayesian network, and each variable can take one of r values, there are r^w combined random experiments represented by each node in J. We calculate the potential tables for all cliques and edges in parallel, and this step takes $O(wr^w n/p)$ time.

Exact Inference Calculation: Our parallel algorithm for exact inference on Junction trees [3] assumes that the evidence comes on a single variable that is present in the root node of the tree. The algorithm involves rooting the

Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) through the Department of the Interior National Business Center under grant number NBCH104009.

tree at any clique that contains the evidence variable, and then propagating the belief all through the tree, in a way similar to the way it is performed in a tree Bayesian network. However, in this case, the messages passed between nodes consist of potential tables. We have pointer jumping to parallelize this computation on junction trees. The complexity of this stage is $O(r^w \log(n).n/p)$.

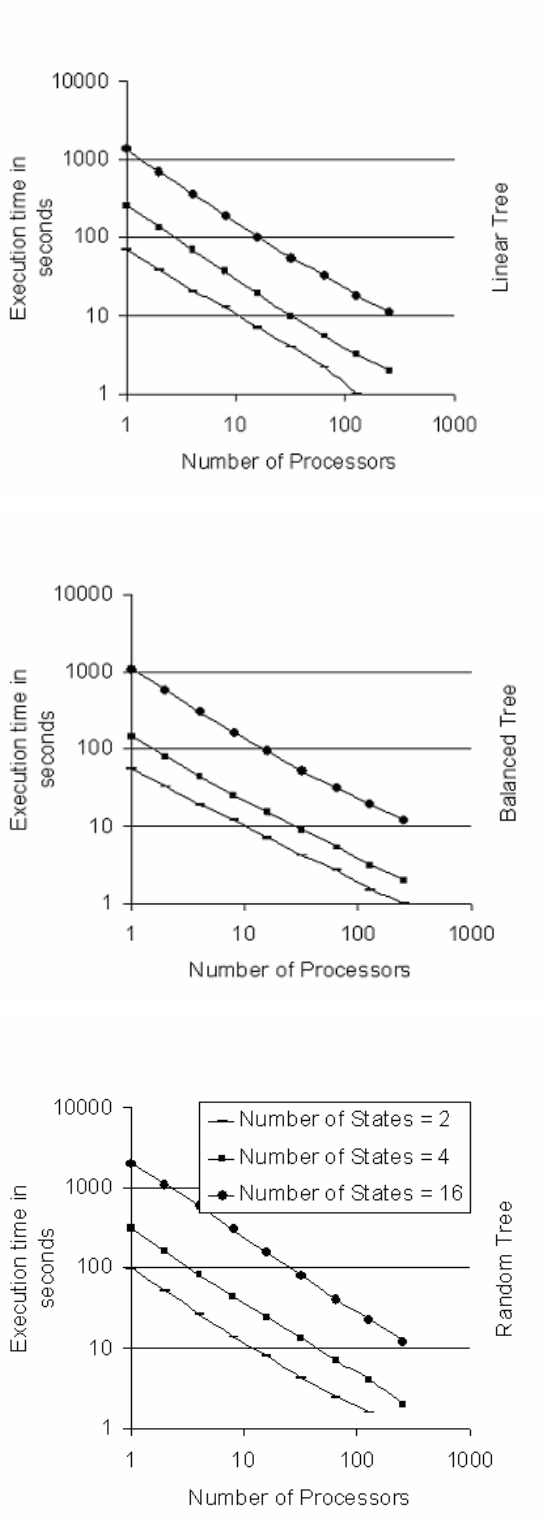

Figure 1: Overall Execution Time for Exact Inference.

Sequential Complexity: If all the above stages are performed sequentially, the overall time complexity is seen to be $O(n^3 + n^2w + wr^wn + nr^w)$.

Experimental Results

We used three machines for our experiments. The Shared Memory Processor at USC is a SunFire 15K system with 64 UltraSPARC III 1.2 GHz processors and a 150 MHz Sun Fireplane redundant 18X18 data, address, and response crossbar interconnect. We also accessed the computing resources at the San Diego Supercomputer Center. One of the machines is a DataStar cluster with 1024 IBM P655 nodes running at 1.5 GHz with 2 GB of memory per processor. The theoretical peak performance of this machine is 15 TeraFLOPS. Furthermore each node is connected to a GPFS (parallel file system) through a fiber channel. The second large machine we ran the experiments on is the Teragrid machine - SGI Altix. It has 1.6 GHz Itanium 2 processors with 1024 Gbytes of shared memory. It runs SGI ProPack 3.4 with OpenMP.

We used linear, balanced, and arbitrary Bayesian networks with 1024 nodes and node in(out) degrees being 1(1), 1(2), and 5(5) respectively. We ran the experiments with variables having 2, 4, and 16 states. The OpenMP directive used were `omp parallel` and `omp parallel for`.

Our experiments (Figure 1) show that our implementation of the exact inference algorithm is scalable. In addition, we performed the experiments with the widely available Intel PNL Library, and found that their implementation does not scale well.

Conclusion

We have presented an implementation of a parallel algorithm for exact inference on arbitrary Bayesian networks. Our experiments show that our solution scales with the number of processors used, while the Intel PNL library does not.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A modern approach*. Prentice Hall, 1995.

[2] S. L. Lauritzen and D. J. Spiegelhalter, "*Local computations with probabilities on graphical structures and their application to expert systems*," in J. of the Royal Statistical Society, 1988, pp. 157–224.

[3] V. K. Namasivayam and V. K. Prasanna, "*Scalable parallel implementation of exact inference in bayesian networks*," in Twelfth International Conference on Parallel and Distributed Systems (ICPADS), July 2006.

[4] A. V. Kozlov, "*Parallel implementations of probabilistic inference*," in Computer 29(12), Dec 96, pp. 33–40.

[5] D. M. Pennock, "*Logarithmic time parallel Bayesian inference*," in [UAI-98], USA, July 1998, pp. 431– 438.

[6] B. A. and G. D., "*A sufficiently fast algorithm for finding close to optimal junction trees*," in [UAI-96], USA, 1996.

[7] *Intel Probabilistic Network Library*, http://www.intel.com/technology/computing/pnl/index.htm.