An Efficient Parallel EM Algorithm: A Sparse Matrix Compaction Technique

Wei-Min Jeng and **Stephen Huang**
Department of Computer Science
University of Houston
Houston, TX 77204
E-Mail: {wjeng, shuang}@cs.uh.edu

Abstract

In recent years, the advances in microprocessors and high-speed networks are making it possible for scientific applications to run on clustered type of environment. Expectation Maximization (EM) algorithm is a popular iterative method that requires a significant amount of computation and memory in approximating the incomplete data for many real-world problems. In this paper we propose a sparse matrix compaction technique to speed up the computation by better manipulating the probability matrix. The sparse matrix compaction method is made more efficient by both taking advantage of the geometrical information of the application and removing of indirect addressing overheads that is associated with most of the compressed sparse matrix method. Load balancing is also achieved by a better distribution scheme implemented in the pre-processing phase of the algorithm. By using the proposed method in handling the sparse matrix, our study demonstrates a promising result for the parallelization of the algorithm.

1. Introduction

Expectation Maximization (EM) algorithm [1] was originally developed for estimating the incomplete data that exist in many real world application [2, 3]; however, the long computation time has impeded the routine use of it due to the size of the matrix involved in the operation. Various versions of the algorithm have been proposed to overcome the problem. For example, accelerating with numerical analysis approach [4] and speeding up with parallelization schemes [5, 6]. However, the high cost/performance ratio remains the same with the computational load required to perform for the algorithm.

In this paper, we present our approach to parallelization of the Shepp and Vardi EM algorithm [10] for PET [4, 5] using the sparse matrix based technique. Taking system geometry information into partition scheme, non-zero elements are extracted from the original matrix and indexed into compressed vectors in the preprocessing phase before the actual operation begins. The knowledge acquired from this stage then is stored and exploited in the matrix-vector multiplication stage to attain the efficiency. The pre-processing operation does not require any specific matrix structure thus can be applied to any of the applications that involve matrix-vector multiplication. Indirect addressing, a main disadvantage as discussed [7, 8, 9] in most of the sparse-matrix schemes, is fixed by better permutation of the multiplication operands. The results of running the parallel scheme on networks of workstations show that the partition-by-inclination strategy is the key to high performance.

2. EM Algorithm

The EM algorithm is founded based on probability model in approximating the unknown information. Suppose we have a model for the complete data Y, with associated probability density function $f(Y|\theta)$, where θ is the set of parameters of the density. Let $Y = (Y_{obs}, Y_{mis})$ where Y_{obs} denote the observed part of Y and Y_{mis} denote the missing information. The complete data Y is not observed directly, instead, the many-to-one mapping $Y_{obs}(Y)$ is the outcome observed. The probability density function of the incomplete data can be defined as $g(Y_{obs}|\theta) = \int f(Y|\theta)\, dY$. The basic goal of the algorithm is to maximize the the likelihood function

$$L(Y_{obs}, \theta) = \log g(Y_{obs}|\theta) = \log \int f(Y|\theta)\, dY$$

with respect to θ.

Since the information of complete data Y is unknown to us, the maximum likelihood function $g = \int f(Y|\theta)\, dY$ can not be obtained. The algorithm uses both the expectation of the likelihood function L given the observed data Y_{obs} and the estimate of θ iteratively until its convergence. Let $\theta^{[r]}$ represent the current estimate of the unknown parameters at the rth iteration. There are two stages in each iteration of the EM algorithm and they can be described as follows:

In the Expectation Step, compute

$$Q(\theta|\theta^{[r]}) = \mathrm{E}[\log f(Y|\theta)|Y_{obs}, \theta^{[r]}],$$

and in Maximization Step, choose $\theta^{[r+1]}$ which can maximize the Q function

$$\theta^{[r+1]} = \max Q(\theta|\theta^{[r]})$$

with respect to the first argument θ. The change from $\theta^{[r]}$ to $\theta^{[r+1]}$ increases the likelihood and its proof can be found at [10].

The EM algorithm as introduced in the above is an iterative method that approximates the incomplete data with maximum likelihood through convergence. One version of the algorithm proposed by Shepp and Vardi has offered the discrete form of solutions to the problem. The estimation process can be stated as the four separate steps in each iteration

Step 1) $\mathbf{x} = \mathrm{P}^{\mathrm{T}} \times \lambda$,
Step 2) $\Phi_{\mathrm{d}} = \mathbf{n}_{\mathrm{d}} / \mathbf{x}_{\mathrm{d}}$
Step 3) $\mathbf{y} = \mathrm{P} \times \Phi$, and
Step 4) $\lambda_{\mathrm{b}}^{\mathrm{new}} = \lambda_{\mathrm{b}}^{\mathrm{old}} \cdot \mathbf{y}_{\mathrm{b}}$,

where P is the input probability matrix that stays the same throughout the execution of the algorithm. Other variables such as λ and Φ are vectors used to hold intermediate results. Both Steps 1 and 3 in the above serial algorithm involve matrix P in matrix-vector type of multiplication. These two forms the performance obstacle in speeding up the algorithm because of the size of the matrix usually encountered. The matrix P that appears in both Steps 1 and 3 is pre-computed and consumes the majority of the execution time for any of the parallel implementations. It posts a serious problem due to the ever-increasing demand for a larger size matrix P. To overcome the problem, reduction in size of the matrix and better handling of the matrix is proposed in this paper for better performance.

3. The EM Algorithm with A Sparse Matrix Compaction (EM/SMC)

In the first and the third stages of traditional EM algorithm, matrix-vector multiplication operations represent the most complex parts of the algorithm in terms of their execution time spent. As in many other scientific applications, the matrix involved in the multiplication operation is very sparse. Unlike the matrix that has a regular structure, the probability matrix in the EM algorithm can not be processed using some of the known technique [11]. As measured by our own phantom study experiment, there are at most 4% of non-zero elements for the matrix encountered in the EM algorithm. It indicates that about 96% or more computation time are wasted in the unnecessary multiplication operations, which are too significant to ignore. The size of the original matrix posts two of the most serious problems to the implementation of the algorithm, both long computation and large memory requirement.

3.1 PET System Geometry with Example

Before the actual parallel scheme is introduced, first we show how the low-density matrix is formed with a typical PET system. Illustrations by the example below, we define the target system to be the one that has 8 detector pairs and 16 (4×4) image boxes. The probability matrix therefore is a sparse one of size 16×8 with high percentage of zero elements. Figure 1 shows the graphical system geometry and its corresponding probability information is then listed in Table 1. Note that adjacent detectors, such as detectors 1 and 8, cannot form detector pairs because of the inability of capturing any gamma ray coincidences with their physical setups. The probability information entries can appear as non-zeroes in Table 1 only if the image boxes are on the paths of their corresponding detector pairs.

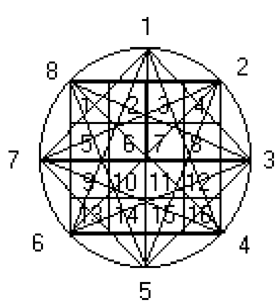


Figure 1: A PET system geometry information (8 detectors and 16 image boxes).

	Detector Pair																			
	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(2,4)	(2,5)	(2,6)	(2,7)	(2,8)	(3,5)	(3,6)	(3,7)	(3,8)	(4,6)	(4,7)	(4,8)	(5,7)	(5,8)	(6,8)
Box 1					X				X	X				X			X		X	X
Box 2			X	X	X				X	X				X			X			
Box 3	X	X	X				X	X	X	X				X						
Box 4	X	X				X	X	X	X	X				X						
Box 5				X	X				X				X	X		X	X		X	X
Box 6			X	X				X	X				X	X			X		X	
Box 7		X	X				X	X	X				X	X			X			
Box 8	X	X				X	X	X	X			X	X	X						
Box 9				X				X	X			X	X			X		X	X	X
Box 10			X	X			X	X	X			X	X			X	X		X	
Box 11		X	X				X	X				X	X	X		X	X		X	
Box 12		X				X	X				X	X	X	X		X	X			
Box 13				X				X				X			X	X		X	X	X
Box 14			X	X			X	X				X			X	X		X	X	
Box 15		X	X				X				X	X			X	X	X		X	
Box 16		X				X	X				X	X			X	X	X			

Table 1: Original probability information matrix , P, of size 16×20 where non-zero entries are represented by "X".

3.2 The Parallelization Scheme

3.2.1 Partition-by-Inclination

Our scheme works by first partition the detector pairs into groups by their inclinations. As shown in Figure 1, detectors are evenly distributed along the ring and that creates roughly the same number of detector pairs for different inclinations. Figure 2 shows the four different detector pair groups in different inclinations with respect to horizontal axis counter-clockwise. In the four different groups, each of which has three detector pairs with the same inclinations. Note that there are other inclinations additional to the four to make a complete partition. Assuming that two processors are available, we assign the first two groups to the first processor and the next two to the second processors. The advantages of using these particular schemes are two folds. First, the load balance can easily be achieved since the workloads are evenly distributed because of the intrinsically symmetric system geometry. More importantly, for each of the image boxes there is at most one detector pair from any inclination groups that has the non-zero probability entry in the original matrix using our setup. Instead of simple extraction operations of non-zeroes, the patterns of the system geometry can be found using this pre-processing scheme. The original matrix can then be rearranged to help extract the non-zeroes and group them in vectors as shown in Table 2.

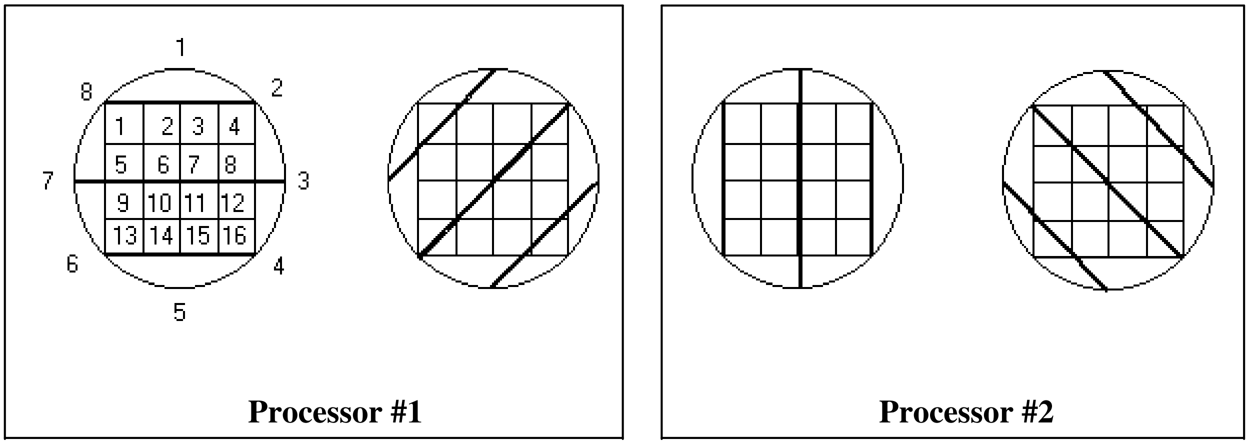


Figure 2: Geometric representations of ImagePixel/DetectorPair layouts partitioned into two processors. (only 4 inclinations are shown)

Detector Pair

	(2, 8)	(3,7)	(4,6)		(1,7)	(2,6)	(3,5)		(1,5)	(2,4)	(6,8)		(1,3)	(4,8)	(5,7)
Box 1	X			Box 1	X			Box 2	X			Box 3	X		
Box 2	X			Box 2	X			Box 3	X			Box 4	X		
Box 3	X			Box 5	X			Box 6	X			Box 8	X		
Box 4	X			Box 3		X		Box 7	X			Box 1		X	
Box 5		X		Box 4		X		Box 10	X			Box 2		X	
Box 6		X		Box 6		X		Box 11	X			Box 5		X	
Box 7		X		Box 7		X		Box 14	X			Box 6		X	
Box 8		X		Box 8		X		Box 15	X			Box 7		X	
Box 9		X		Box 9		X		Box 4		X		Box 10		X	
Box 10		X		Box 10		X		Box 8		X		Box 11		X	
Box 11		X		Box 11		X		Box 12		X		Box 12		X	
Box 12		X		Box 13		X		Box 16		X		Box 15		X	
Box 13			X	Box 14		X		Box 1			X	Box 16		X	
Box 14			X	Box 12			X	Box 5			X	Box 9			X
Box 15			X	Box 15			X	Box 9			X	Box 13			X
Box 16			X	Box 16			X	Box 13			X	Box 14			X

Table 2: Rearranged submatrix information exploiting the geometry information of the system.

Algorithm EM/SMC (Sparse Matrix based EM algorithm for Step 1)
Input: pre-processed partition-by-inclination and delimiter information
Output: Adjusting detector pair count estimate factor information

```
   /* N denotes the number of nodes
for all i, 0 <= i < N do in parallel
   j = 0;
   /* loop for number of inclination groups (Gi) of processor i
   while (j < Gi) do
      b = 0;
      d = 0;
      /* loop for number of detector pairs (Di,j) in each inclination group */
      while (d < Di,j) do
            /* loop for the number of elements in each detector pair vector */
            while (b < delimiter[d]) do
                  t[d] = t[d] + Pi,j[b] * λi,j[b];
                  b = b + 1;
            endwhile
            d = d + 1;
      endwhile
      j = j + 1;
   endwhile
endfor
```

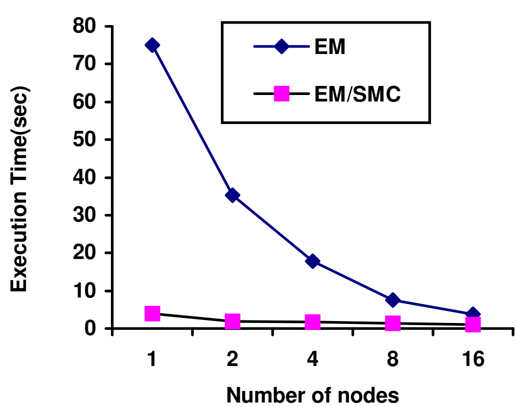
4. Experimental Results

4.1 The Platform

Different settings are selected to test their performance characteristics for both original EM and SM-EM algorithms. A network of 1, 2, 4, …,to 16 processing nodes of SUN workstations are employed to conduct the experiments and the data are collected and analyzed. We illustrate the use of communication operations for our parallel EM algorithm by using the MPI [12] parallel library.

4.2 EM versus EM/SMC

We implemented the EM/SMC algorithm with probability matrix generated by phantom study information. The 4096x4096 matrix was first pre-processed using the compressing scheme as described in Section 3.2 to obtain partition-by-inclination vectors before distributed into individual processors. Figures 3 and 4 illustrate the results of our experiments obtained from using both original and extracted matrices in different versions of the algorithm.



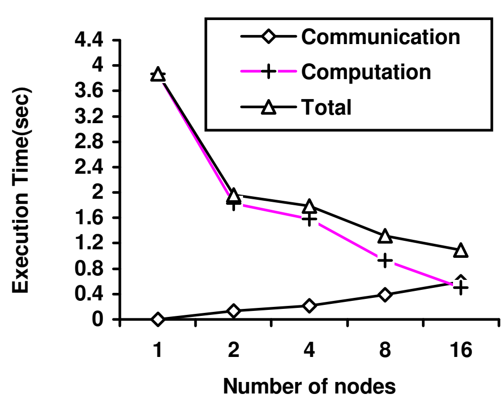


Figure 3: Execution time for EM and EM/SMC algorithms. Figure 4: Breakdown analysis for EM/SMC.

With about 4% density of non-zero elements in the original sparse matrix, the extracted data could make a significant difference in terms of execution time. According to the experiment results, the biggest saving could cut the execution time down to 5.15% of its original magnitude. There is a programming control overhead incurred due to the irregular pattern of the new generated data. The use of indexing rather than typical loop control also makes it more difficult for program optimization. The amount of savings exceeds the overhead by far even with the additional cost involved in the new scheme.

5. Summary

In this paper we presented an inclination-based sparse matrix-vector multiplication algorithm for distributed-memory parallel machine. The EM/SMC algorithm consists of two phases in carrying out the matrix-vector multiplication. In the pre-processing stage the nonzero elements are extracted into a few vectors with the use of both system geometry information and inherent patterns. The compressed vectors are distributed evenly by their inclinations to processors to balance the workload. In the actual computation phase, the delimiter information is used to perform the individual summation for distinguished detector pairs to prevent the overhead of indirect addressing. Depending on both matrix and configuration information, the proposed algorithm shows that significant performance gain can be predicted if the inter-processor communication is properly controlled. We believe this technique can be applied to sparse matrices in other applications.

Reference

[1] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from incomplete Data via the EM Algorithm," J. Roy. Statist. Soc., Vol. 39, pp. 1-38, 1977.

Alvaro Suarez, C.N. Ojeda-Guerra, "Overlapping Computations and Communications in Torus Networks," IEEE Proceedings of Parallel and Distributed Processing, pp. 162-169, 1996.

[2] A. Ansari and R. Viswanathan, "Application of EM Algorithm to Be Detection of Direct Sequence Signal in Pulsed Noise Jamming," IEEE Trans. Comm., Vol. 41, No. 8, pp. 1151-1154, 1993.

[3] R. E. Blahut, "Computation of Channel Capacity and Rate-distortion Functions," IEEE Trans. Infor. Th., Vol. 18, pp. 460-473, July 1972.

[4] Linda Kaufman, "Implementing and Accelerating the EM Algorithm for Positron Emission Tomography," IEEE Transactions on Medical Imaging, Vol. MI-6, pp. 37-50, March, 1987.

[5] Chung-Ming Chen and Soo-Young Lee, "On Parallelizing the EM Algorithm for PET Image Reconstruction," IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 8, pp. 260-273, August 1994.

[6] Wei-Min Jeng and Stephen Huang, "Inter-Iteration Optimization of Parallel EM Algorithm on Message-Passing Multicomputers," In Proceedings of International Conference on Parallel Conference, pp. 245-252, 1998.

[7] G. V. Paolini and G. Radicati di Brozolo, "Data Structures to Vectorize CG Algorithms for General Sparsity Patterns," BIT 29, pp. 703-718, 1989.

[8] D. Kincaid, T. Oppe, J. Respress, D. Young, ITPACKV2C User's Guide, Report CAN-191, The University of Texas at Austin, 1984.

[9] Wei-Min Jeng and Stephen Huang, "A Sparse Matrix Based Parallel Image Reconstruction for PET on SP2," 1999 IEEE Symposium on Computer-Based Medical Systems, to appear.

[10] L. A. Shepp and Y. Vardi, "Maximum Likelihood Reconstruction for Emission Tomography," IEEE Transactions on Medical Imaging, Vol. 1, pp. 113-122, Oct. 1982.

[11] K. Madsen, G. H. Rodrigue, and J. Karush, "Matrix Multiplication by Diagonals on a Vector/Parallel Processor," Information Processing Letter, No. 2, pp. 41-45, 1978.

[12] MPI Forum. "The MPI Message Passing Interface Standard," Http://www.mcs.anl.gov/mpi/mpi-report/mpi-report.html, November 1995.