Parallel Multigrid Acceleration for the Finite-Element Gaussian Belief Propagation Algorithm

Yousef El-Kurdi, Warren J. Gross, and Dennis Giannacopoulos

Department of Electrical and Computer Engineering, McGill University, Montreal, QC H3A 0E9, Canada

We introduce a novel parallel multigrid algorithm, referred to as the finite-element multigrid Gaussian belief propagation (FMGaBP), to accelerate the convergence of the recently introduced finite-element Gaussian belief propagation solver. The FMGaBP algorithm processes the FEM computation in a fully distributed and parallel manner, with stencil-like element-by-element operations, demonstrating high parallel efficiency. The results for both sequential as well as parallel message scheduling versions of FMGaBP demonstrate high convergence rates independent of the scale of discretization on the finest mesh. In comparison with the multigrid preconditioned conjugate gradient (MG-PCG) solver, the FMGaBP algorithm demonstrates considerable iteration reductions as tested by Laplace benchmark problems. In addition, the parallel implementation of FMGaBP shows a speedup of 2.9 times over the parallel implementation of MG-PCG using eight CPU cores.

Index Terms—**FEMs, Gaussian belief propagation (GaBP), multigrid.**

I. Introduction

GAUSSIAN belief propagation (GaBP) is a variant of the belief propagation (BP) algorithm originally used to compute marginal distributions on graphical models [1]. GaBP was also introduced as a distributed solver for linear systems of equations [2], [3] that demonstrated great potential for parallel hardware implementations for FEMs [4], [5]. In addition, GaBP has been shown to outperform classical iterative solvers such as Gauss–Seidel and Jacobi [2]. However, such algorithms, which are derived based on pairwise interconnect assumptions on the underlying graphical model, suffer mostly from lack of convergence when diagonal dominance properties are not met. To address these convergence shortcomings and improve the parallel efficiency of the FEM computation the FEM-GaBP (FGaBP) algorithm was recently introduced, which was derived directly from a FEM variational formulation [6]. The FGaBP algorithm solves the FEM in parallel element by element without the need to assemble a global sparse matrix. FGaBP can be shown empirically to reach high parallel efficiency as the scale of the FEM problem increases [6]. However, like most of the iterative solvers, the FGaBP convergence rate tends to stall when executed on fine meshes.

In this paper, we address this issue by introducing the novel finite-element multigrid GaBP (FMGaBP) algorithm. To the best of our knowledge, this is the first multigrid formulation for continuous domain GaBP algorithms that is derived directly from a variational formulation of the FEM. Our empirical results for FMGaBP demonstrate that the new algorithm achieves high convergence rates, typical of multigrid, which is independent of the scale of the finest grid. In addition, we show for a benchmark Laplace problem that FMGaBP requires considerably less iterations than the multigrid preconditioned conjugate gradient (MG-PCG) solver. The eight-thread multicore implementation of FMGaBP demonstrates speedups of up to 2.9 times over the MG-PCG solver implemented by the multithreaded library deal.II [7]. This paper is organized as follows. In the following section, we present a brief summary of the FGaBP algorithm. Then, we present the multigrid formulation of the FMGaBP algorithm. Finally, we conclude with the performance results.

Manuscript received June 28, 2013; revised August 19, 2013; accepted September 24, 2013. Date of current version February 21, 2014. Corresponding author: Y. El-Kurdi (e-mail: yousef.el-kurdi@mail.mcgill.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMAG.2013.2284483

II. FGaBP Algorithm

Multigrid techniques can vary widely using different relaxation algorithms with different smoothing properties, the approach of generating the course grid operators, and the types of grid transfer operations that involve interpolation and restriction generally independent of the nature of the problem [8]. In this paper, we introduce an efficient adaptation of multigrid techniques to the parallel FGaBP algorithm resulting in the FMGaBP algorithm. The FMGaBP algorithm considerably improves the FGaBP convergence rate while maintaining high parallel efficiency of stencil-like, element-by-element, operations. To later illustrate the FMGaBP algorithm, we present here a brief background on the FGaBP algorithm.

The FGaBP algorithm can be derived directly from the FEM variational formulation of the scalar Helmholtz equation. The FEM approximates the solution to the Helmholtz equation by rendering stationary the following functional [9]:

$$\mathcal{F}(U) = \sum_{s \in \mathcal{S}} \mathcal{F}_s(U_s) \tag{1}$$

where $\mathcal{S}$ is the set of all finite elements, U_s are the field unknowns for element s, and $\mathcal{F}_s$ is the energy contribution of each finite element which can, generally, be stated as

$$\mathcal{F}_s(U_s) = \frac{1}{2} U_s^T M_s U_s - B_s^T U \tag{2}$$

in which we refer to M_s as the element characteristic matrix with dimensions $n \times n$ where n is the number of unknowns per element and B_s is the element source vector.

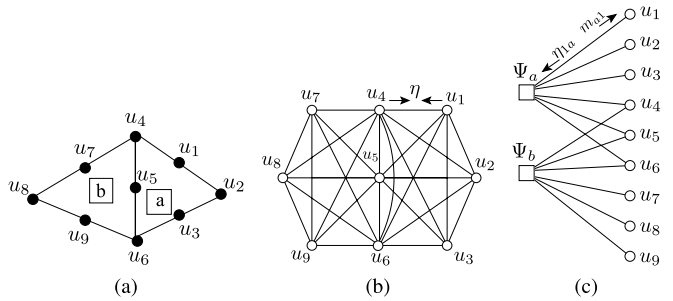

Fig. 1. (a) Sample FEM mesh of two second-order triangles. Two graphical model representations are shown in (b) and (c); the messages η in a graphical model are communicated recursively either sequentially or in parallel. (b) Conventional pairwise graphical model. (c) New FEM-FG model with reduced number of communication links and improved parallelism.

The FGaBP algorithm solves the FEM problem using BP inference to find the stationary point of the energy functional (1). We start by formulating FEM as a variational inference problem by modifying (1) as follows:

$$\mathcal{P}(U) = \tfrac{1}{Z} \textstyle\prod_{s\in\mathcal{S}} \Psi_s(U_s) \tag{3}$$

where Z is a normalizing constant and $\Psi_s(U_s)$ are local factor functions of the local finite element variables U_s defined as

$$\Psi_s(U_s) = \exp\left[-\frac{1}{2}U_s^T M_s U_s + B_s^T U\right]. \tag{4}$$

It is worth noting here that Ψ_s as defined in (4) takes a multivariate Gaussian form albeit unnormalized when the characteristic matrix M_s is symmetric and positive definite.

It can be shown that $\mathcal{P}$ as in (3) represents a Gaussian multivariate probability distribution when $\mathcal{F}$ is convex quadratic [10]. Hence, the optimality condition of $\mathcal{F}$ can be restated as

$$\arg\min_U \mathcal{F} = \arg\max_U \mathcal{P} \tag{5}$$

where the variables vector U represents joint Gaussian random variables under the distribution $\mathcal{P}$. Since $\mathcal{P}$ is maximized when $U = \mu$, where μ is the mean vector of $\mathcal{P}$, the FEM minimization problem of the functional $\mathcal{F}$ is transformed into a computational inference problem of finding the marginal means of the random variables U under the distribution $\mathcal{P}$. Hence, BP inference algorithms can be employed to compute the marginal means of the random variables U.

Given the probability distribution $\mathcal{P}$ being represented in a factored form, as shown in (3), we can define a graphical model to facilitate the derivation of the BP update rules to infer the marginal distributions of the random variables U. Such a graphical model is referred to as an FG [11], which, in our setting, we refer to as a FEM-FG. The FEM-FG is shown in Fig. 1(c). Using the FEM-FG, we can execute the general BP update rules [1] by passing two types of messages. A factor node message $m_{a\to i}$ sent from a factor node Ψ_a to a connected variable node u_i; and a variable node message $\eta_{i\to a}$ sent back from the variable node u_i to the factor node Ψ_a. The general BP update rules are stated as follows:

$$m_{ai}(u_i) = \int_{u_{\mathcal{N}(a)\setminus i}} \Psi_a(u_a) \prod_{j\in\mathcal{N}(a)\setminus i} \eta_{ja}(u_j)\, \mathrm{d}u_{\mathcal{N}(a)\setminus i} \tag{6}$$

$$\eta_{ia}(u_i) = \prod_{k\in\mathcal{N}(i)\setminus a} m_{ki}(u_i) \tag{7}$$

$$b_i(u_i) \propto \prod_{k\in\mathcal{N}(i)} m_{ki}(u_i) \tag{8}$$

where $\mathcal{N}(a)$ is the set of all nodes' indices connected to node index a, referred to as the neighborhood set of a; $\mathcal{N}(a)\setminus i$ is the neighborhood set of a minus node i; $b_i(u_i)$ is referred to as the belief at node i. By representing the variable nodes u_i as Gaussian random variables, the BP updates can be solved in a closed form resulting in the FGaBP algorithm as follows.

1) For each variable node i, compute variable node messages $(\alpha_{ia}, \beta_{ia})$ to each factor node $a \in \mathcal{N}(i)$ as follows:

$$\alpha_i = \sum_{k\in\mathcal{N}(i)} a_{ki} \qquad \alpha_{ia} = \alpha_i - \alpha_{ai} \tag{9}$$

$$\beta_i = \sum_{k\in\mathcal{N}(i)} \beta_{ki} \qquad \beta_{ia} = \beta_i - \beta_{ai}. \tag{10}$$

2) To compute the factor node messages we first define $\mathcal{A}$, $\mathcal{B}$, W, and K for each factor node a as follows:

$$\mathcal{A} = \begin{bmatrix} \alpha_{i_1 a} & 0 & \cdots & 0 \\ 0 & \alpha_{i_2 a} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_{i_n a} \end{bmatrix}, \mathcal{B} = \begin{bmatrix} \beta_{i_1 a} \\ \beta_{i_2 a} \\ \vdots \\ \beta_{i_n a} \end{bmatrix}, \tag{11}$$

$$W = M + \mathcal{A}, \qquad K = B + \mathcal{B},$$

where $\{i_1, i_2, \ldots, i_n\} = \mathcal{N}(a)$, M and B are the element a characteristic matrix and source vector as defined in (4) with $s = a$. We then partition W and K as follows:

$$W = \begin{bmatrix} W_{\mathcal{L}(i)} & V^T \\ V & \bar{W} \end{bmatrix}, \qquad K = \begin{bmatrix} K_{\mathcal{L}(i)} \\ \bar{K} \end{bmatrix}$$

where $\mathcal{L}(i)$ is the local index corresponding to the global node i. Now, compute and send factor node a messages $(\alpha_{ai}, \beta_{ai})$ to each variable node i as follows:

$$\alpha_{ai} = M_{\mathcal{L}(i)} - V^T \bar{W}^{-1} V \tag{12}$$

$$\beta_{ai} = B_{\mathcal{L}(i)} - \bar{K}^T \bar{W}^{-1} V. \tag{13}$$

3) *A priori*, all the messages are initialized as $\alpha = 1$ and $\beta = 0$. Messages can be communicated according to any schedule. Once the messages converge, the nodal means, or solutions, μ_i can be obtained by

$$\mu_i = \frac{\beta_i}{\alpha_i}. \tag{14}$$

Finally, essential boundary conditions can be incorporated directly into Ψ_s in (4) on an element-by-element basis. It is worth noting here that the FGaBP update rules in (12) and (13) are based on distributed local computations performed using matrices of order $(n-1)$. In addition, the update rules, as shown here, are applicable for any arbitrary element geometrical shape or interpolation order, hence introducing great flexibility in FGaBP implementations.

III. FMGaBP Algorithm

The distinguishing feature of the FGaBP algorithm is solving the FEM in parallel, element by element, without needing to assemble a large sparse matrix or performing any global algebraic operation such as sparse matrix–vector multiplication. These key advantages, which have important implications for parallel hardware implementations, are maintained by the new multigrid formulation for FMGaBP. This is accomplished using the FGaBP as a level smoother for the FMGaBP algorithm. As a result, the multigrid transfer operations take a different form, which is more computationally decoupled and localized. To illustrate the FMGaBP formulation, we define a quantity, associated with each individual finite element a, referred to as the factor node belief b_a. The belief b_a takes the form of a multivariate Gaussian distribution as

$$b_a(u_a) \propto \exp\left[-\frac{1}{2}u_a^T W_a u_a + K_a^T u_a\right] \tag{15}$$

where W_a is the inverse covariance of the factor node belief, K_a is the source vector with the same dimension, and u_a is a vector of random unknowns linked to the finite element a. Matrix W_a and vector K_a are updated each iteration according to (11). By observing the dynamics of the message update rules of the BP algorithm in (6)–(8), we can deduce that at message convergence, the joint mean vector of b_a, given by $\bar{u}_a = W_a^{-1} K_a$ as computed from the local factor perspective, will be equal to the marginal means of each of the random unknowns u_a as computed from global perspective by (14). Using this observation, we can formulate a quantity referred to as the belief residual r_a given by

$$r_a = K_a - W_a \bar{u}_a. \tag{16}$$

Using multiple grids with refinement by splitting and using a consistent local node numbering between each set of parent–child elements, the belief residuals of each group of child elements can be locally restricted into the parent element as

$$K_a^H = \mathcal{R}_l r_a^h \tag{17}$$

where K_a^H is the source vector of the parent element, r_a^h is the accumulated local residual of child elements, and $\mathcal{R}_l$ is the child–parent local restriction operator. Similarly, we can use local operations for interpolation to apply the corrections from the coarse elements as follows:

$$\bar{u}_a^h \leftarrow \bar{u}_a^h + \mathcal{I}_l \bar{u}_a^H. \tag{18}$$

Using the level updated $\bar{u}_a^h$, we can reinitialize the corresponding level local messages using (16) with $r_a \to 0$ and solve for K_a. In this setup, we keep the α_{ai} messages fixed for each level and only reinitialize the β_{ai} messages. For self-adjoint problems, $\mathcal{R}_l$ is typically the transpose of $\mathcal{I}_l$.

It is important to note that both operators $\mathcal{I}_l$ and $\mathcal{R}_l$ do not require any storage nor do they contain any references to global node numbers; hence, their effect can be produced using a generic local node numbering association between each parent–child element set, as shown in Fig. 2(d). As a result, the restriction and interpolation functions can be implemented in FMGaBP using embarrassingly parallel and efficient stencil-like operations. Since it can be shown that convergence is achieved when the level restricted local residuals approach zero, the resulting multigrid FMGaBP algorithm becomes a fixed-point algorithm for the stationary solution point.

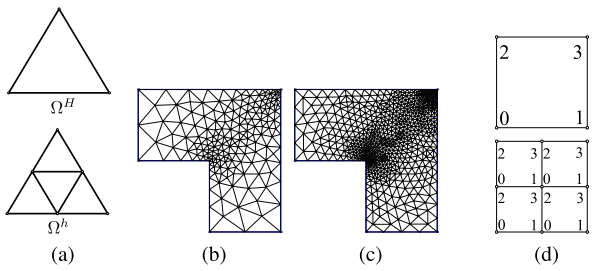

Fig. 2. (a) Mesh refinement by splitting each triangle in mesh Ω^H into four geometrically similar subtriangles to produce a finer mesh Ω^h. (b) Course irregular mesh. (c) Refined mesh by splitting. (d) Local node numbering for each parent–child element set using quadrilaterals.

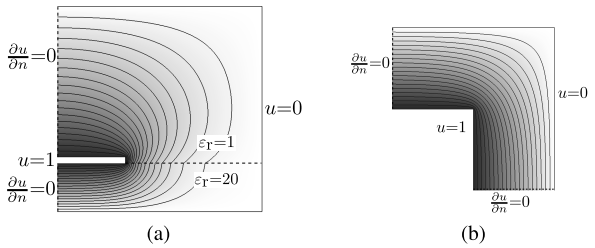

Fig. 3. Laplace equipotential contours obtained using FMGaBP, the dimensions of both structures are within the unit interval [0, 1] cm. (a) Right symmetrical side of the shielded strip between two different materials. (b) Top-right symmetrical corner of the square coaxial conductor.

IV. Result

The performance of the FMGaBP algorithm is demonstrated using two Laplace benchmark problems, as shown in Fig. 3. The first problem is the shielded strip conductor (SSC) using two different material properties; and the second is the L-shaped corner of the square coaxial conductor (LSC). Both problems employ Dirichlet and Neumann boundary conditions.

The FMGaBP is coded using C++ object-oriented programming approach. The element matrices and source vectors are the only input to the FMGaBP algorithm. Two open-source software libraries were used to formulate the FEM problem and assemble the local element matrices and vectors. The SSC problem was formulated using the library GetFEM++ [12] with semi-irregular triangular meshes; while the LSC problem was formulated using the library deal.II [7] with hierarchical quadrilateral meshes. Since deal.II inherently supports hierarchical meshes and multigrid solvers using parallel implementations, it was used for performance comparisons. A V-cycle multigrid scheme is used in all our experiments. All the solvers are terminated when the residual l^2-norm is dropped to 10^{-15}. The FMGaBP algorithm uses the iterative FGaBP as the coarsest level solver; whereas, a householder direct solver was used in deal.II for the MG-PCG solver.

Table I shows the FMGaBP convergence results for the SSC problem. The results demonstrate a convergence performance almost independent of the number of unknowns in the finest level. However, a trend of slightly increasing number of

TABLE I
ITERATIONS OF THE FMGaBP FOR THE SSC PROBLEM USING FIVE LEVELS OF REFINEMENT

Refinement Level	Unknowns	Triangles	*Iterations* $v_1 = 1,\ v_2 = 1$ *
1-coarse	222	382	—
2	825	1,528	*9*
3	3,177	6,112	*11*
4	12,465	24,448	*13*
5	49,377	97,792	*15*
6	196,545	391,168	*16*

* Parameters v_1 and v_2 are pre and post smoothing iterations.

TABLE II
FMGaBP SPEEDUP OVER DEAL.II FOR BOTH SETUP AND SOLVE TIME USING EIGHT THREADS ON 2 × QUADCORE PROCESSORS

Unknowns (million)	Quadrilaterals (million)	FMGaBP			MG-PCG			*Speedup*	
		itr	t_{st}	t_{sv} *	itr	t_{st}	t_{sv}	*setup*	*solve*
0.788	0.786	6	2.6	0.98	10	6.14	2.56	*2.4*	*2.6*
3.15	3.15	6	10.5	3.74	10	27.5	10.4	*2.6*	*2.8*
12.6	12.6	6	43.1	14.2	10	108	41.6	*2.5*	*2.9*

* The symbols t_{st} and t_{sv} refer to the setup and solve time respectively reported in seconds. The acronym itr refers to iterations.

iterations is observed. This increase is expected as a result of the strongly varying material coefficients.

Table II shows the FMGaBP results for the LSC problem compared with the results from deal.II. The timing results were obtained using a compute node with two quadcore processors of type Intel Xeon X5560 Nehalem with 2.8-GHz clock speed and 8-MB cache. Problem sizes up to 12.6 million unknowns are solved. OpenMP was used to parallelize the FMGaBP. A coloring algorithm is used to guarantee thread safety. The coloring algorithm required minimal overhead, by virtue of using the hierarchical structure of the mesh provided by deal.II. For all timing cases, the best of 40 runs is reported. The MG-PCG solver, used for comparison, is provided by deal.II configured with multithreading enabled. In the timing analysis, we focus only on 2-D problems since 3-D problems require a much larger memory footprint for a single node to handle. Therefore, it is best to analyze large 3-D problems using MPI implementations on multinode clusters. Such implementations will require sophisticated partitioning algorithms, which will be investigated as a future work. Nevertheless, the FMGaBP algorithm was tested to work successfully with 3-D problems as well.

As shown in Table II, the FMGaBP algorithm requires close to half the iteration count of MG-PCG. In addition, the FMGaBP algorithm demonstrates considerable time speedups for both the total setup time and the solve time. Setup operations are not parallelized in these experiments, therefore sequential execution time is reported for the setup phase. The major reductions in setup time were because the FMGaBP, contrary to the deal.II library, does not require the setup of globally large sparse matrices, or level transfer matrices. The FMGaBP algorithm also demonstrated parallel solve time speedups of up to 2.9 times over deal.II's

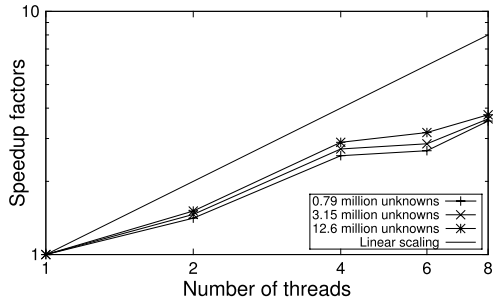

Fig. 4. Speedup factors for FMGaBP over the sequential execution on 2 × quadcore processors node.

MG-PCG solver. The speedups were due to mainly two factors, the considerable iteration reductions and the higher parallel efficiency of FMGaBP. As a key factor, the FMGaBP implements the level transfer operations as embarrassingly parallel and local stencil-like operations that are inherently thread safe.

Fig. 4 shows the speedup factors of FMGaBP for three problem sizes scaled for eight threads. FMGaBP demonstrates approximately linear trends of speedups with increasing number of threads. In addition, the graph trends show increased parallel efficiency as the problem size increases.

V. CONCLUSION

The novel FMGaBP algorithm was introduced and was shown to achieve high parallel performance. In addition, we showed for a benchmark Laplace problem that FMGaBP requires considerably less iterations than the MG-PCG solver. The eight-thread multicore implementation of FMGaBP demonstrates speedups of up to 2.9 times over deal.II's implementation of MG-PCG.

REFERENCES

[1] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, USA; Morgan Kaufmann, 1988.
[2] O. Shental, P. Siegel, J. Wolf, D. Bickson, and D. Dolev, "Gaussian belief propagation solver for systems of linear equations," in *Proc. IEEE ISIT*, Jun./Nov. 2008, pp. 1863–1867.
[3] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Comput.*, vol. 13, no. 10, pp. 2173–2200, 2001.
[4] Y. El-Kurdi, W. Gross, and D. Giannacopoulos, "Efficient implementation of Gaussian belief propagation solver for large sparse diagonally dominant linear systems," *IEEE Trans. Magn.*, vol. 48, no. 2, pp. 471–474, Feb. 2012.
[5] Y. El-Kurdi, D. Giannacopoulos, and W. Gross, "Relaxed Gaussian belief propagation," in *Proc. IEEE ISIT*, Jul. 2012, pp. 2002–2006.
[6] Y. El-Kurdi, W. Gross, and D. Giannacopoulos, "Parallel solution of the finite element method using Gaussian belief propagation," in *Proc. 15th IEEE CEFC*, Nov. 2012, p. 141.
[7] W. Bangerth, R. Hartmann, and G. Kanschat, "Deal. II—A general purpose object oriented finite element library," *ACM Trans. Math. Softw.*, vol. 33, no. 4, pp. 1–27, Aug. 2007.
[8] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed. Philadelphia, PA, USA: SIAM, 2000.
[9] J. Jin, *The Finite Element Method in Electromagnetics*, 2nd ed. New York, NY, USA: Wiley, 2002.
[10] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, nos. 1–2, pp. 1–305, Jan. 2008.
[11] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," vol. 47, no. 2, pp. 498–519, Feb. 2001.
[12] Y. Renard and J. Pommier. (2007). *GetFEM++—An Open-Source Finite Element Library* [Online]. Available: http://download.gna.org/getfem/html/homepage/