A Unifying View of Message Passing algorithms for Gaussian MRFs

Ananth Ranganathan

November 13, 2007

Abstract

We illustrate the relationship between message passing algorithms in GMRFs and matrix factorization algorithms. Specifically, we show that message passing on trees is equivalent to Gaussian elimination, while Loopy Belief Propagation is equivalent to Gauss-Seidel relaxation. Similarly, recently introduced message passing algorithms such as the Extended Message Passing, and the Embedded Subgraphs algorithm are also shown to be equivalent to commonly known matrix methods. We describe efficient extensions to message passing algorithms based on exploiting these relationships.

1 Introduction

Gaussian graphical models arise in a number of varied applications such as sensor networks [7], robot mapping [1], and stereo-based depth computations [5]. Consequently, estimation algorithms for such models have become popular. A number of these algorithms are based on message passing and thus, are amenable to distributed or parallel implementation making them of further interest.

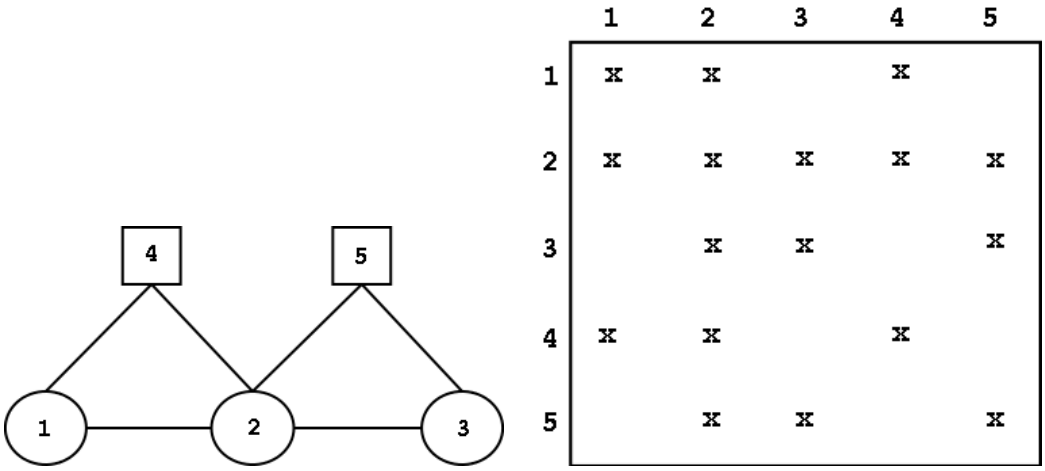
In this technical report, we describe the close relationship between popular message passing algorithms for GMRFs and matrix-based algorithms for solving linear systems. We consider three different algorithms - belief propagation (BP), both on trees [9] and on graphs with loops [13], the Extended Message Passing algorithm of Plarre and Kumar [10], and the Embedded Subgraphs algorithm of Delouille et. al. [3]. These algorithms together span the gamut of currently available message passing algorithms. BP on trees is exact and non-iterative as is the Extended Message Passing (EMP) algorithm which works on any graph. Loopy Belief Propagation (LBP) and Embedded Subgraphs (ES) are iterative algorithms that converge to the exact MAP solution but do not provide the correct marginal covariances. Additionally BP, LBP and ES are parallelizable whereas EMP is not. Thus, these algorithms form a representative set of message passing algorithms. As our aim is to foster a qualitative and intuitive understanding of these algorithms, we will not specify them formally. If required, these can be found in the respective references.

Any GMRF is equivalent to a linear system of the form $Ax = b$, where x is the vector of unknowns, b is the error vector, and the covariance of the system is $\Sigma = A^{-1}$. Solving large systems by direct matrix inversion as $x = A^{-1}b,\ \Sigma = A^{-1}$ is highly inefficient, and hence the search for alternate algorithms.

As the running example in this paper, we will consider the graph in Figure 1 along with its corresponding system matrix. Since we will only be concerned with the pattern of non-zero elements in the matrix, these are shown using crosses at the appropriate locations, and we omit the exact values of these elements. Further, we will ignore the error vector b of the linear system for ease of exposition, noting that all the operations performed on the matrix are also to be performed on the error vector.

2 Belief Propagation

We start with the standard belief propagation (BP) algorithm on acyclic graphs. BP works in two phases. In the first phase, messages flow up the tree with each node gathering messages from its children and using these to compute a message to its parent. The second phase consists of a similar downward flow of messages from the root to the leaves.



(a) (b)

Figure 1: The graph and system matrix for the running example used in the paper

BP on acyclic GMRFs is exactly equivalent to Gaussian elimination. The first phase consists of column operations that reduce the system matrix to triangular form. The second phase subsequently performs back-substitution to obtain the mean values and marginal covariances. Since the first phase triangularizes the matrix, it can be viewed as performing a Cholesky factorization of the system matrix. This fact will be of use in understanding message passing algorithms discussed later.

Consider, as an example, the tree in Figure 2(a) along with its system matrix in Figure 2(b). The operations of message passing in the first phase are shown in Figures 2(c)-(d). Note that no new elements (fill-in) are introduced during the triangularization process. This is true only in the case of acyclic graphs and can be intuitively seen from the structure of the system matrix. Each row of the matrix has only one element to the left of the diagonal, whose column gives the parent of the node corresponding to the row. Similarly the non-zero elements to the right of the diagonal correspond to the children of the node.

The scheme discussed above is only possible if we identify a root node, and hence a tree structure of the graph under consideration. However, in the Generalized Belief Propagation (GBP) algorithm described by Yedidia et. al. [14], all the nodes are considered equal and message passing works simply through the exchange of messages between neighboring nodes. To see the relationship between this asynchronous message passing scheme and the two-phase algorithm described above, we note that the message passed from a node, say A, to another node B does not depend on whether A is a parent or a child of B. Hence, each node can consider itself to be the root of the tree and perform message-passing accordingly. This results in the asynchronous message passing scheme. In effect, if the graph contains n nodes, this scheme solves n Gaussian elimination problems (as discussed above) simultaneously, each corresponding to a tree with a different root node, as illustrated in Figure 3. Since there is no backsubstitution phase required now, another way of seeing this is as a message passing implementation of the Gauss-Jordan algorithm, which results in the system matrix being reduced to a block diagonal matrix from which the MAP solution and marginal covariances can be recovered directly.

2.1 Loopy Belief Propagation

If the graph has loops then local message passing schemes based on Gaussian Elimination break down, since the system matrix does not get triangularized. This is because in the process of zeroing out entries to the right of the diagonal, a message may potentially introduce new non-zero entries. However, LBP has been proven to always converge to the correct MAP solution for GMRFs [13]. How do we explain this?

LBP can be shown to be equivalent to a modified version of the Jacobi iteration method. Jacobi iterations are based

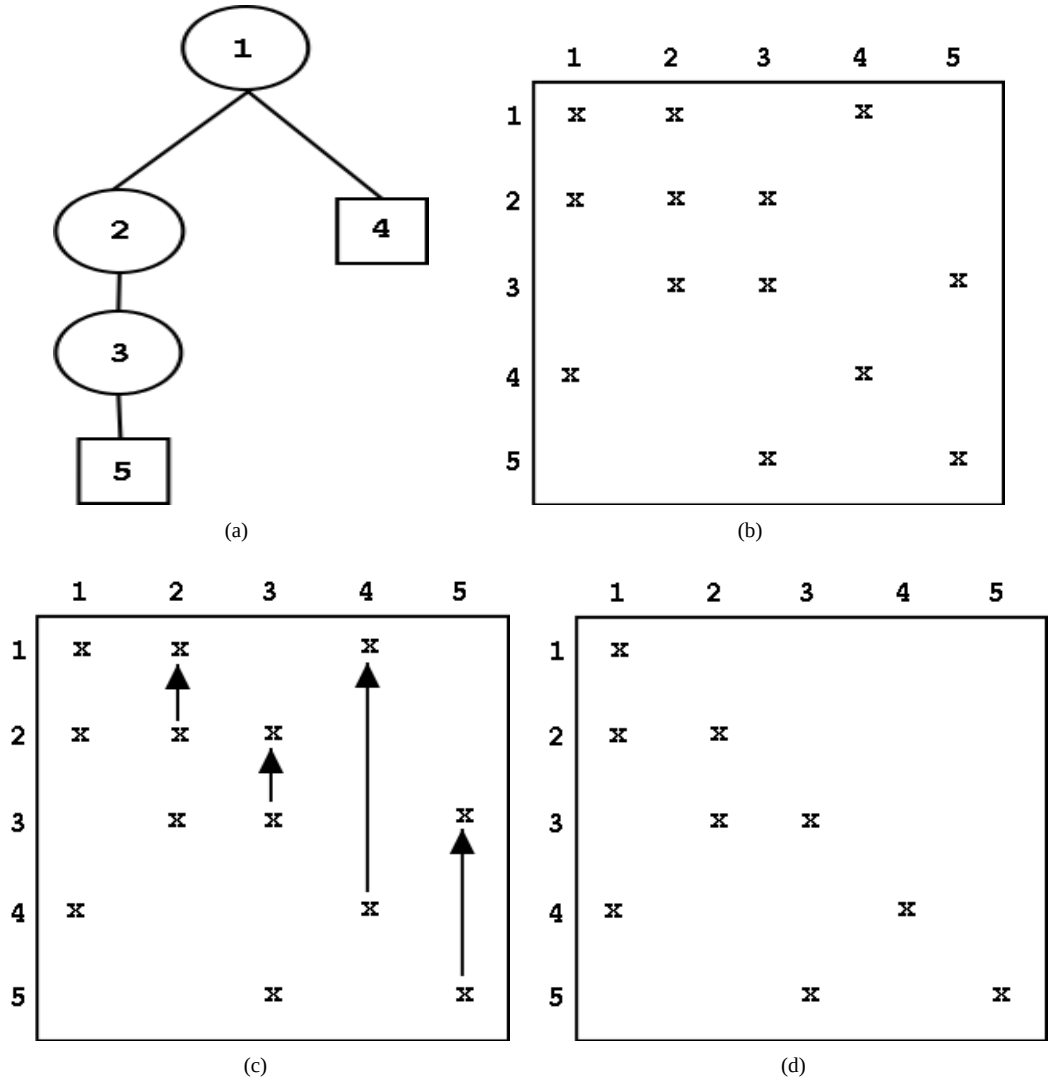


Figure 2: (a) An example tree illustrating the effect of BP on the system matrix (b) The system matrix is shown with rows and columns conforming to a pre-ordering on the tree (c) The messages from each node to its parent performs a row-based elimination operation, zeroing out the elements at the end of the arrows (d) The result is a triangular matrix

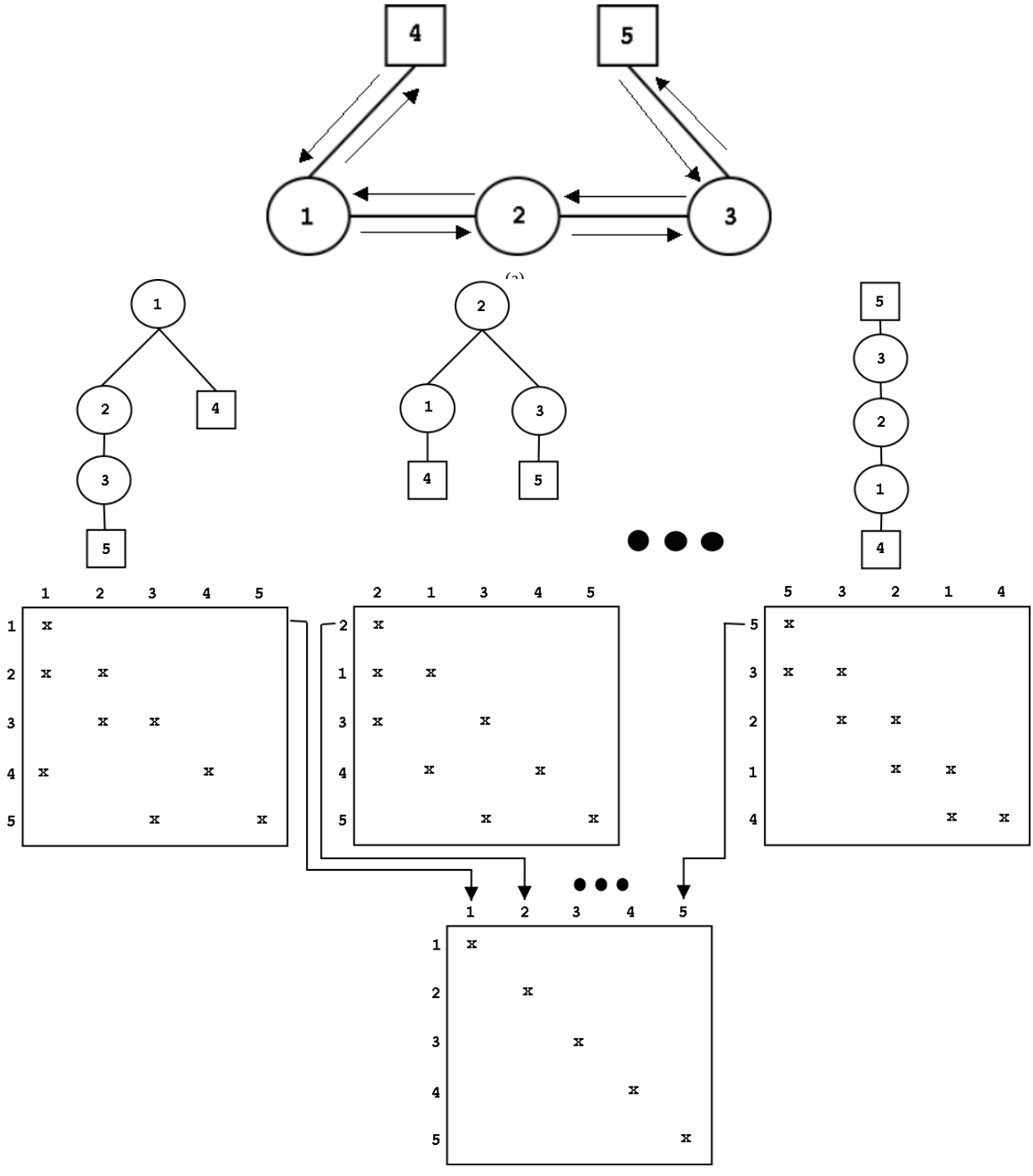


Figure 3: BP on an undirected graph with message passing between neighboring nodes (shown in (a)) is equivalent to performing BP simultaneously on the set of trees with all possible roots (shown in (b)). The result, as in Gauss-Jordan elimination for matrices, is a block diagonal matrix. Note that the matrix ordering corresponding to each tree is a pre-order sorting of the tree.

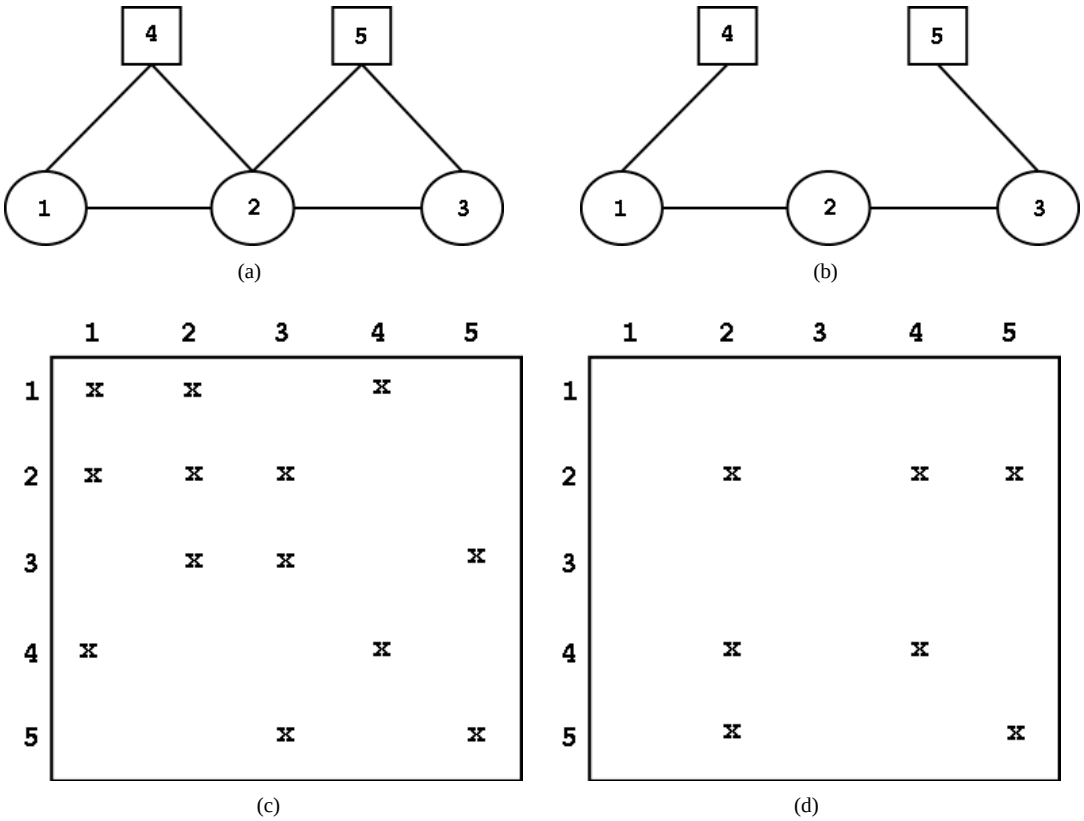


Figure 4: (a) The example graph and the (b) spanning tree used to illustrate EBP (c) The matrix A_T corresponding to the spanning tree and the (d) cutting matrix K

on the equation

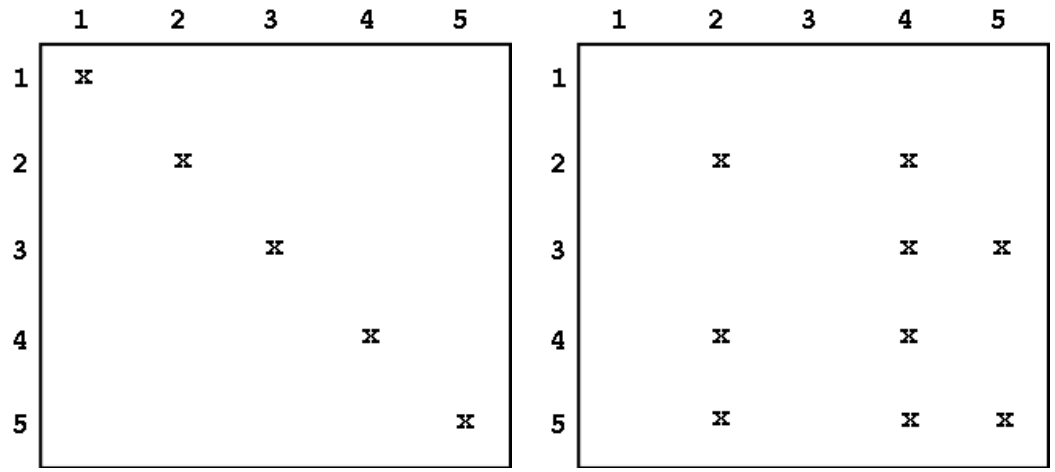
$$Dx^{(k)} \quad = \quad (L+U)x^{(k-1)} + b$$

where D, $-L$, and $-U$ are respectively the diagonal, strictly lower triangular, and strictly upper triangular portions of the system matrix. While, for LBP, the correspondence between the system matrix A and the Jacobi iterations is not straight-forward, the equivalence can be proven. Details of the proof are not provided here but can be found in [**?**]. Note that this proof relates asynchronous LBP to Gauss-Seidel relaxation. Synchronous LBP can similarly be shown to relate to the Jacobi method.

3 Extended Message Passing

LBP does not give the exact MAP solution in a non-iterative manner only because we restrict the algorithm to only local message passing. What if we allow non-local message passing, i.e. messages between nodes that are not adjacent in the graph? This is precisely the approach taken by the EMP algorithm [10]. However, we still do not want every node to pass messages to every other node as this would result in a $O(n^2)$ algorithm just to compute the messages. The EMP algorithm overcomes this by splitting the graph into two pieces, one of which is a tree and can be solved exactly using BP while the second piece is solved using Gaussian elimination implemented via non-local message passing.

EMP works by splitting the system matrix A into two matrices so that $A = A_T + K$, where the matrix A_T corresponds to a spanning tree of the graph and K, called the cutting matrix, corresponds to the edges in the graph not in the spanning tree. The spanning tree is a good choice for dissecting the graph since it minimizes the rank of the cutting matrix that solved using relatively expensive non-local message passing.



(a) (b)

Figure 5: The spanning tree matrix and the cutting matrix after performing BP on the spanning tree.

Consider the example in Figure 4 showing the EMP setup for out running example. EMP first performs standard BP on the spanning tree and reduces the spanning tree matrix to a diagonal matrix. The messages also simultaneously keep track of the changes to the cutting matrix K during this process. The resulting matrices are shown in Figure 5. These matrices are then combined to obtain a modified system.

The crucial observation now is that the nodes involved in the cutting matrix, i.e the nodes with edges that are not in the spanning tree, form an independent subsystem that can be solved without referencing the other nodes in the system. This is demonstrated in Figure 6. EMP uses this property to solve the cutting matrix subsystem using the Gauss-Jordan algorithm implemented via message passing as described in Section 2. Note that in the worst case, messages may need to be passed between every pair of nodes in this subsystem but for sparse graphs the number of messages will, in general, be a lot fewer.

The mean values for the cutting matrix subsystem can be obtained straightaway after the elimination step while recovering the means of the other nodes involves a back-substitution step, which may again require non-local information.

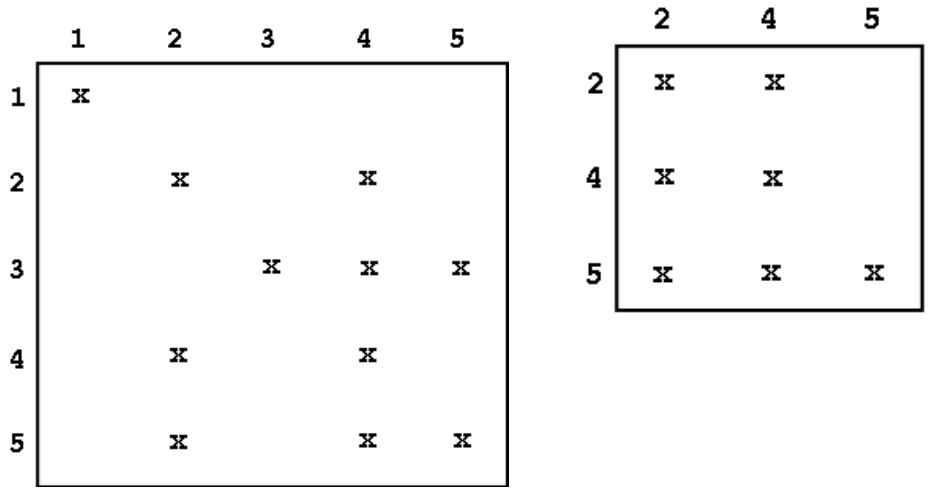
Marginal covariances can be obtained by changing the right side of the linear system. Instead of the measurement errors, we use a block-diagonal matrix containing identity diagonal blocks corresponding to the nodes involved in the cutting matrix. As before, we modify this matrix during message passing to reflect the BP phase on the tree and then perform a Gauss-Jordan inversion of the cutting matrix subsystem to obtain the marginal covariances of the nodes in the subsystem. Covariances of the remaining nodes are again obtained using back-substitution.

3.1 An incremental version of EMP

Often, the need arises in many applications for incremental inference updates to the graph. This is especially the case in smoothing applications where additional edges (corresponding to new measurements) or new nodes (corresponding to new unknowns) may be added to the graph at each time step. Efficiency requires that we do not calculate all the unknowns when the graph changes slightly but instead update the values only of the affected variables. An example application where the need for an incremental inference algorithm arises is given in [**?**].

As it stands, the EMP algorithm is not incremental, i.e. the addition of even a single edge or node into the graph requires a full re-computation of all the messages and unknowns. The reason for this is the use of Gauss-Jordan elimination to solve the spanning tree and the cutting matrix subsystem. By (block-)diagonalizing the matrix, Gauss-Jordan elimination bakes in information from the right-side of the linear system into the final diagonized matrix. A change in the right-side, caused due to an additional node or edge, can only be dealt with by recomputing the diagonalization starting from scratch.

Our understanding of the relationship between matrix algorithms and EMP now comes in handy and can be used



(a) (b)

Figure 6: (a)The combined system after putting together the spanning tree matrix and the cutting matrix after performing BP on the spanning tree (b) The nodes involved in the cutting matrix form an independent subsystem that is solved using the Gauss-Jordan algorithm.

to make EMP incremental. Instead of diagonalizing the matrix we triangularize it, i.e. compute the Cholesky factor. Since the Cholesky factor can be updated incrementally when new rows/columns/elements are added either to the matrix or to the right side, the result is an incremental algorithm.

In practise, this means that we do not perform two-way BP on the spanning tree but instead only computing the messages flowing from the leaves to the root, as described in Section 2. Note that we now need to specify a root and a tree ordering on the nodes whereas this was not required by the EMP algorithm. After addition of new nodes or edges, the messages differ significantly around the newly added components but the differences gradually die out as we move farther away in the tree. Thus, not all the messages on the tree will have to be recomputed. Additionally, if the changes to the tree are close to the leaves (as is usually the case), the computational savings will be much larger than if they are close to the root.

The cutting matrix subsystem is also dealt with similarly. We enforce an ordering on the cutting matrix subsystem so that each node only sends messages to nodes below it in the ordering. However, computational savings are less likely in this case since the nodes in the system are more densely connected. This emphasizes our previous statement about the importance of keeping the subsystem as small as possible.

4 Embedded Subgraphs

The Embedded Subgraphs (ES) algorithm [3] tries to overcome the non-local nature of EMP by clustering the graph so that most of the loopy nature of the graph is subsumed in the node clusters. The graph is segmented into arbitrary independent subgraphs, where independence implies that the subgraphs do not share any nodes. We can then form a "supergraph" consisting of the subgraphs and other singleton nodes that do not belong to any of the subgraphs. We will call the vertices in the supergraph as "supernodes".

The subgraphs, since they are independent, form a block diagonal matrix for some ordering of the nodes in the graph. ES is essentially a block version of LBP, where each iteration consists of two stages -

1. Each supernode exchanges messages with its adjacent. In practise, this is accomplished by restricting message passing to the nodes containing edges linking different components in the supergraph.

2. Each supernode solves for its unknowns through internal message passing. This is done locally by each node inverting its supernode matrix block. Singleton nodes do not communicate in this step.

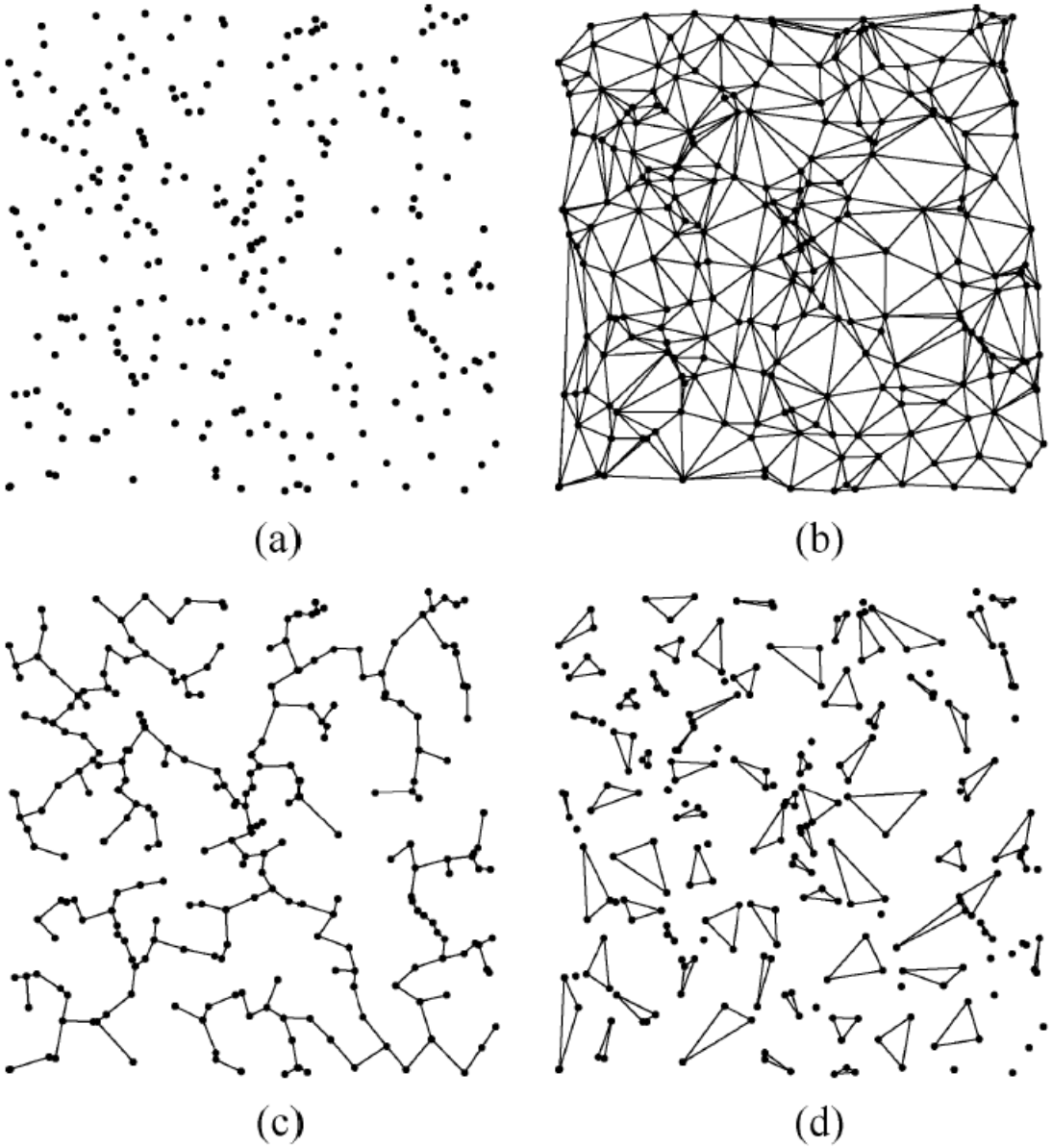


Figure 7: (a) Example sensor network with 250 uniformly distributed sensor nodes. (b) Graphical model for the hidden variables based on the Delaunay triangulation of the sensor locations. (c) Minimum spanning tree embedded in the Delaunay graph. (d) A set of disjoint subgraphs. Each subgraph is a triangle or a singleton node embedded in the Delaunay triangulation. Disjoint refers to the fact that no two subgraphs share a common node. Figure and caption from [3].

Since LBP (or equivalently Gauss-Seidel relaxation) is only guaranteed to compute the correct means but not the marginal covariances, ES is also limited in the same manner. However, the block structuring results in faster convergence than LBP.

Clearly, selection of the subgraphs is an important step in the algorithm. While large subgraphs increase the convergence rate, each iteration takes more time due to the matrix inversion involved therein. Subgraphs in the form of triangles [4] (Figure 7) and polygons [2] have been successfully used.

4.1 Embedded Subgraphs and the Multifrontal algorithm

Consider the case where the supergraph is acyclic, i.e the clustering results in the subsumption of every loop in the original graph. Clearly, in this case, the algorithm is not iterative since the ES algorithm is simply performing BP on the supergraph.

Now consider an additional twist where we mandate a local coordinate frame for each subgraph. We can do this by selecting one node per subgraph to be the base node and connecting all the other nodes in the subgraph to this base node. The nodes of the subgraph are now transformed to lie in a coordinate frame with the base node is the origin. The base node of a subgraph is thus involved in all its external edges. Additionally, we combine the nodes involved in these external edges (i.e the inter-subgraph edges) into a "separator" set since these nodes separate the subgraphs from each other.

The above described procedure transforms each subgraph into a clique and the supergraph into a clique tree. The algorithm proceeds by solving each clique internally and then exchanging messages with neighboring cliques. Crucially, the inter-clique message exchange only changes the value of the base node in each subgraph but does not affect the other nodes.

The algorithm described above is equivalent to the junction tree algorithm, and in the matrix case, to solving the system using a Multifrontal Cholesky factorization. A detailed and easy to follow explanation of the use of clique trees to solve sparse linear systems is available in [8]. The distributed Multifrontal factorization algorithm, implemented through message passing on a clique tree, is described in [11]. An example showing the relationship between subgraphs and Multifrontal Cholesky is given in Figure 8. Note that the Multifrontal algorithm always convertes the GMRF into a clique tree while the Embedded Subgraphs algorithm will in general operate on a cyclic supergraph containing subgraphs and singleton nodes.

In light of the above connection, we can view the ES algorithm as a "halfway house" between LBP on the graph and the complete junction tree algorithm. While the junction tree algorithm aims to construct a tree (the clique tree) from the loopy graph, the ES algorithm tries to reduce the number of loops so as to increase the convergence rate of LBP while keeping the algorithm local. The advantage of the ES algorithm lies in its combination of efficiency and locality which can be crucial in some applications, an example being sensor networks where long range communication is usually impossible. The disadvantage is the inability to obtain exact marginal covariances which is only possible in general through non-local communication.

5 Conclusion

We presented the relationship between some common and popular message passing algorithms and the equivalent matrix algorithms. In general, every message passing algorithm for GMRFs can be viewed as a distributed implementation of some matrix-based algorithm for solving linear systems. However, an additional constraint in most message passing algorithms is the need for strong locality in communication. In spite of this constraint, we believe that understanding the link between message passing and matrix methods can lead to new and improved distributed algorithms.

We presented two new algorithms obtained by modifying existing message passing algorithms through their underlying matrix counterparts. These are respectively

1. an incremental version of the Extended Message Passing algorithm
2. an exact Embedded Subgraph algorithm based on the Multifrontal Cholesky algorithm that can, again, be made incremental

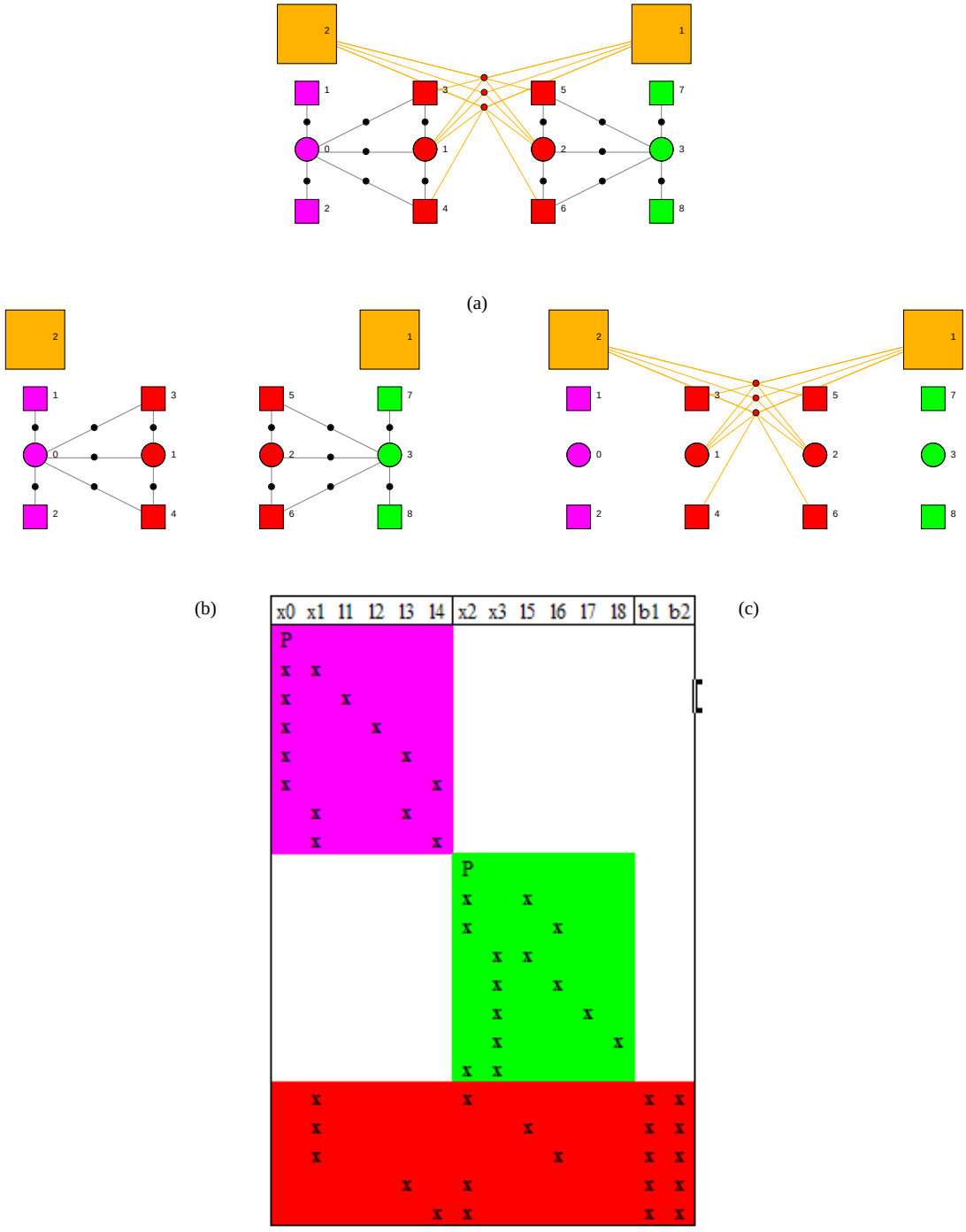


Figure 8: Illustration of the Multifrontal algorithm and its relationship to Embedded Subgraphs. (a) The example GMRF (small circles on the edges denote measurements) divided into subgraphs with the base nodes of the subgraphs shown in yellow. The subgraphs are shown in purple and green respectively. The red nodes are involved in edges across subgraphs and form the separator set (b) edges involved in the intra-subgraph solution phase (c) edges involved in the inter-subgraph solution phase (d) the reordered system matrix has a block diagonal component corresponding to the independent subgraphs (in purple and green) and a separator component corresponding to the separator set (in red). Figures taken from [6]

A major omission in the algorithms we have discussed here is the Embedded Trees algorithm of Sudderth [12]. This algorithm is unique in being an iterative algorithm capable of computing the exact means *and* marginal covariances. However, the algorithm is related to fixed point Richardson iterations and hence, is not amenable to a simple matrix characterization like the other algorithms discussed are.

In our presentation, we have omitted mention of an entire class of matrix algorithms based on bipartite measurement graphs rather than GMRFs. The most common algorithm in this category (analogous to the Cholesky factorization for GMRFs) is QR factorization. The reason for this omission is mainly due to the confusion regarding the semantics of a measurement node in a bipartite graph. For example, while a sensor in a sensor network is clearly a physical realization of a node in a GMRF, no such counterpart exists for a measurement node in a bipartite graph. Hence, it is the lack of applications rather than a lack of understanding that is the reason for the absence of message passing algorithms on bipartite graphs.

References

[1] F. Dellaert. Square Root SAM: Simultaneous location and mapping via square root information smoothing. In *Robotics: Science and Systems (RSS)*, 2005.

[2] V. Delouille, R. Neelamani, and R. Baraniuk. Robust distributed estimation in sensor networks using the embedded polygons algorithm. In *Information Processing in Sensor Networks*, 2004.

[3] V. Delouille, R. Neelamani, and R. Baraniuk. Robust Distributed Estimation Using the Embedded Subgraphs Algorithm. *IEEE Transactions on Signal Processing*, 54(8), August 2006.

[4] V. Delouille, R. Neelamani, V. Chandrasekaran, and R. Baraniuk. The Embedded Triangles Algorithm for Distributed Estimation in Sensor Networks. In *IEEE Workshop on Statistical Signal Processing (SSP)*, 2003.

[5] M. Isard and J. MacCormick. Dense motion and disparity estimation via loopy belief propagation. Technical Report MSR-TR-2005-163, Microsoft Research, 2005.

[6] Alexander Kipp, Peter Krauthausen, and Frank Dellaert. A Partially Fixed Linearization Approach for Submap-Parametrized Smoothing and Mapping. Technical Report GIT-GVU-05-25, Georgia Institute of Technology, 2005.

[7] M.A. Paskin and C.E. Guestrin. Robust probabilistic inference in distributed systems. In *Proc. 20^{th} Conf. on Uncertainty in AI (UAI)*, Banff, Canada, July 2004.

[8] Mark A. Paskin and Gregory D. Lawrence. Junction tree algorithms for solving sparse linear systems. Technical Report UCB/CSD-03-1271, EECS Department, University of California, Berkeley, 2003.

[9] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[10] K. Plarre and P. R. Kumar. Extended message passing algorithm for inference in loopy gaussian graphical models. *Ad Hoc Networks*, 2:153–169, 2004.

[11] A. Pothen and C. Sun. Distributed multifrontal factorization using clique trees. In *Proc. of the Fifth SIAM Conf. on Parallel Processing for Scientific Computing*, pages 34–40. Society for Industrial and Applied Mathematics, 1992.

[12] E. B. Sudderth, M. J. Wainwright, and A. S. Willsky. Embedded trees: Estimation of Gaussian Processes on graphs with cycles. *IEEE Transactions on Signal Processing*, 52(11):3136–3150, 2004.

[13] Yair Weiss and William T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. In *Advances in Neural Information Processing Systems (NIPS)*, pages 673–679, 1999.

[14] J.S. Yedidia, W.T. Freeman, and Y.Weiss. Understanding belief propagation and its generalizations. Technical Report TR-2001-22, Mitsubishi Electric Research Laboratories, January 2002.