Modified Belief Propagation Algorithm for Energy Saving in Wireless and Sensor Networks

Danny Bickson and Danny Dolev and Yair Weiss
School of Computer Science and Engineering,
The Hebrew University of Jerusalem.
Email: {daniel51,dolev,yweiss}@cs.huji.ac.il

September 26, 2005

Abstract

In this paper we develop a simple variant of the belief propagation (BP) algorithm targeted towards reducing the communication cost of broadcast supporting networks such as sensor networks. Our modified algorithm can be proven to achieve identical computation results on each node, while at the same time significantly reducing the number of transmitted messages. Furthermore, the number of computation steps in each node is not increased. We pay in additional storage space, as each node saves the last message received from each of his neighbors.

Many recent works in the field of sensor networks use the BP algorithm for solving various problems. Our variant could be easily applied to any of these proposed systems, thus improving energy consumption and running time. One of the important issues in sensor networks is how to minimize of the sensors' energy consumption, reducing the number of transmitted messages by a large factor significantly improves the energy consumption. Another advantage of our algorithm is a decrease in the number of transmission slots needed, since in wireless environment acquiring a transmission slot can be costly in terms of time and power.

1 Introduction

1.1 Belief Propagation Algorithm

In this section we briefly outline the original max-product belief propagation algorithm. We will describe the algorithm for the undirected graphs, although the modifications we propose can be easily applied to the directed case as well. We relate to the max-product belief propagation algorithm. Similar arguments can be used for the sum-product algorithm. We describe the Belief propagation for discrete variables. Similar technique can be applied for the continues variable as well.

An undirected graphical model G consists of a set of vertices V and a set of edges E connecting them. Each vertex v_i is associated with a random variable (X_i). We assume that the joint probability distribution factors into a product of terms involving node pairs and single nodes. These factors are called edge potentials $psi_{ij}(Xi, Xj)$ and local (or self) potentials $\psi_{ii}(X_i)$.

For the rest of this paper we use the term nodes to refer to vertices.

The max-product belief propagation algorithm [5] is a distributed inference algorithm that enables us to calculate the marginal probabilities of the nodes, otherwise known as the "beliefs". It is a distributed message-passing algorithm and is therefore suitable for communication networks [3, 2, 6]. The Belief Propagation(BP) algorithm gives exact results on trees. We have no guarantee for the algorithm performance on graphs with cycles. The input to the BP algorithm is a graphical model with self potentials $\psi_{ii}(X_i)$ and edge potentials $\psi_{ij}(X_i, X_j)$. The output of the algorithm is the vector of node beliefs (posteriori probabilities). The algorithm

is an iterative distributed message passing algorithm where messages sent between nodes are determined by the following update rule:

$$m_{ij}^{(t+1)} = \alpha \max_{X_i} \psi_{ij}(X_i, X_j)\psi_i(X_i) \prod_{X_k \in N(X_i) - X_j} m_{ki}^{(t)}(X_j) \tag{1}$$

Where $m_{ij}(X_j)$ is a message sent from node X_i to node X_j, α is a normalization factor and $N(X_j)$ is the set of neighbors of node X_j. Finally each node calculates the belief:

$$b_i(X_i) = \alpha\psi_i(X_i) \prod_{X_j \in N(Xi)} m_{ji}(X_i) \tag{2}$$

The algorithm converges when the node receives identical messages from all neighbors for two consecutive rounds. In networks containing cycles the algorithm might not converge. However, in practice, there are several applications where the algorithm produces very good results even for graphs with loops, for example, the case of Turbo codes [**?**].

1.2 BP applications in sensor networks

Many recent works use the BP algorithm within a sensor network setting. For example, [3, 6] discusses the task of self calibration of sensors: sensors calibrate their reading based upon their local neighborhood to reduce inaccuracy of the sensors or noisy readings. [2] uses the BP algorithm for the aggregation of information in sensor networks. Other works use BP for clustering [7], calculating averages [4] and for hypothesis testing [1]. These works are given as examples of recent research in this area.

2 Our algorithm

Our algorithm is composed of two steps. In the first step - the broadcast step - each node broadcasts a modified message to all of its neighbors. The second step is a correction step, in which each node computes the original values it would obtain if the original BP algorithm where used.

We use the notation n_{i*} to denote the messages in our algorithm, so as to avoid confusion with the messages m_{ij} of the original BP algorithm. In the original algorithm, m_{ij} refers to a message from node i to node j. In our algorithm each node sends a unique message using broadcast to all of its neighbors, and so the notation n_{i*} is used for an outgoing message broadcast from node i to all its neighbors. α is a normalization constant.

Broadcast step: A single message n_{i*} is broadcasted from node i to all of its neighbors.

$$n_{i*}^{(t+1)} = \alpha\psi_i(X_i) \prod_{X_k \in N(X_i)} n_{k*}^{(t)}(X_j) \tag{3}$$

Correction step: The original BP message m_{ij} is then calculated by node j using the received broadcast messages.

$$m_{ij}^{(t+1)} = \frac{\max_{x_j} \psi_{ij}(X_i, X_j) n_{i*}^{(t)}}{m_{ji}^{(t)}} \tag{4}$$

We assume that both BP messages and the modified algorithm messages are initialized with uniform values.

$$n_i^{(1)} = m_{ij}^{(1)} = v_0, v_0(i) = 1/K \tag{5}$$

2.1 Algorithm Properties

Claim 1. The modified algorithm after the correction step achieves the exact computation result as in the original BP algorithm for the same input.

Proof. It is easy to show using induction on t, the round number, that both algorithms generate the same output after each round. For $t = 1$ the equivalence is trivial, since the self potentials are identically initialized in both algorithms before the first round.

For the induction step, we assume that after round t in the modified algorithm the messages are calculated correctly, and we want to prove it for round $t+1$. We note that in the modified algorithm, the message $n_{i*}^{(t+1)}$ has two changes relative to the BP message. First, the edge potential factor, $\psi_{ij}(X_i, X_j)$, is omitted (see equation (1)). Second, we multiply all the incoming messages - including the message $m_{ji}^{(t)}$ (which is excluded from the multiplication in the original BP algorithm). In the correction step, we correct the calculation by first multiplying by $\psi_{ij}(X_i, X_j)$ (i.e. by the initially omitted factor), and then by dividing by the message $m_{ji}^{(t)}$. After applying these corrections we obtain the original BP message. The cost is an additional $O(d)$ storage where d is the node degree (or the number of neighbors which the node is within the wireless signal range of the node). This is because we now need to store the calculated message, $m_{ji}^{(t)}$ from round t.

Claim 2. The computation needed by the modified algorithm can be computed locally without additional information, for any original message values.

Proof. In the correction step, nodes i and j both have mutual knowledge of their edge potentials $\psi_{ij}(X_i, X_j)$. Thus, node j can correct the computation by multiplying the edge potential with the incoming message $n_{*i}^{(t)}$. Furthermore, it is possible to divide by the vector $m_{ji}^{(t)}$ since this message was calculated and stored in the previous round. We should only be careful that there are no zero values in the divisor. In case there are zero values, we omit the matching row or column of the edge potential matrix.

Claim 3. The modified algorithm uses the same number of computation steps or less relative to the original BP algorithm.

Proof. Since we moved the calculation of multiplication of the factor $\psi_{ij}(X_i, X_j)$ locally from node i to node j we did not change the total number of computation steps. Furthermore, our modified algorithm allows us to calculate the product $\prod_{X_k \in N(X_i)} n_{*i}^{(t)}(X_j)$ only once, thus saving multiplications on the sender side (as opposed to the receiver side, which now has an extra division calculation for each neighbor). Calculating the product once is a standard implementation speedup for BP (e. g. Wainwright et. al in [8]). Therefore, in total, the number of computation steps needed by our algorithm is equal or lower to the total number of computation steps needed by the BP algorithm.

Claim 4. The number of messages sent using the modified algorithm is reduced by factor d, where d is the average node degree in the graph.

Proof. In the original BP algorithm each node sends a message to each neighbor on each round. In our algorithm we send only one message on each round and then calculate the original algorithm message values and so we save a number of transmissions proportional to the average node degree in the graph. In sensor networks this means a reduced need to acquire the communication medium and therefore savings of energy and time.

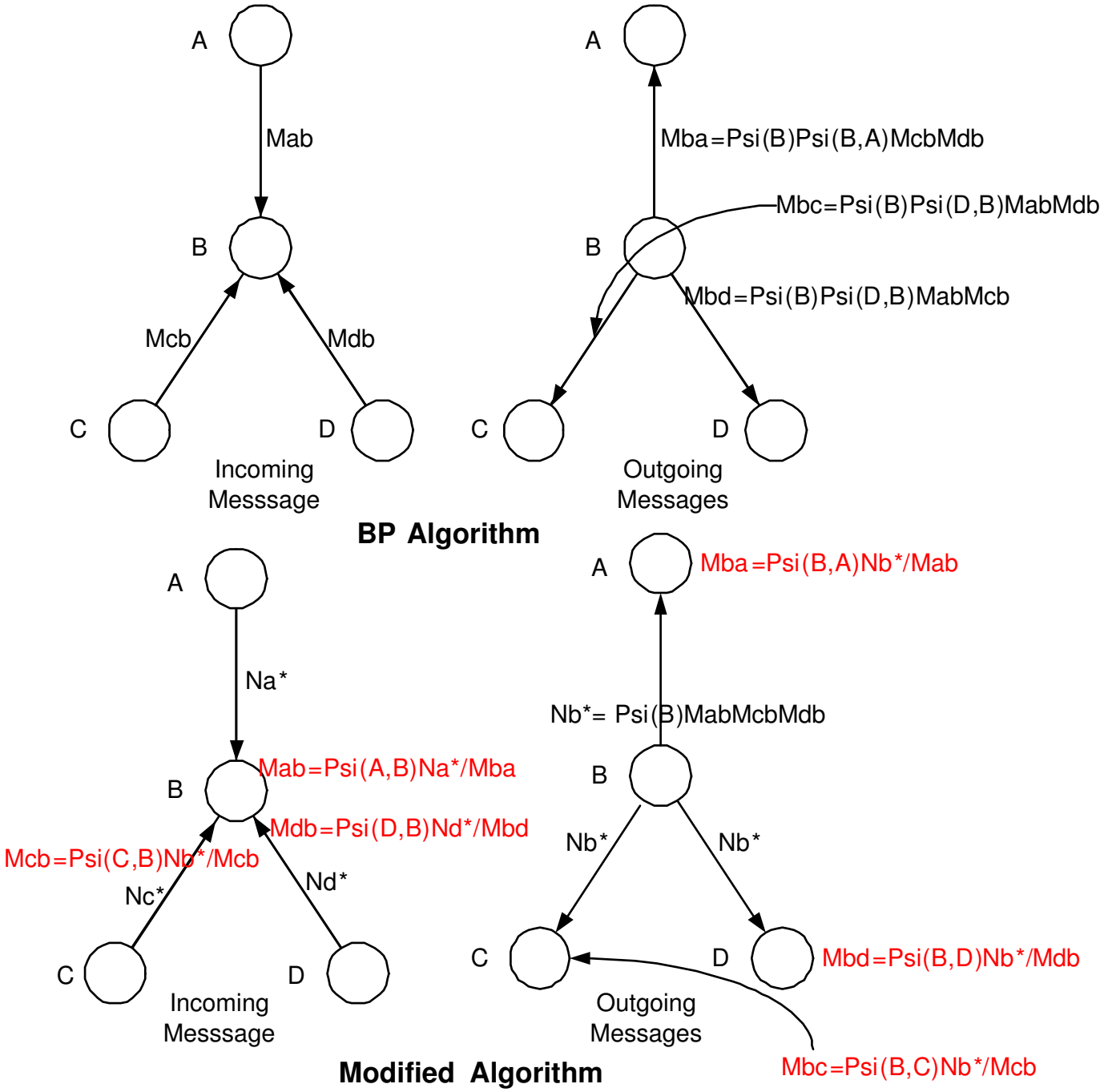

Figure 1: Graphical comparison of both algorithms for a network of 4 nodes. Node B's incoming and outgoing messages are shown. The upper diagram shows the original BP algorithm messages while the lower diagram shows the modified broadcast algorithm. Black arrows signify messages where red labels are local computations made at the nodes. M_{ij} is a message from node i to node j in the BP algorithm while N_{i*} is a broadcast message in the modified algorithm from node i to all of his neighbors.

3 Conclusion

We presented a modified BP algorithm which allows each node to broadcast the same message to all its neighbors. The modified algorithm exploits the broadcast network capabilities by sending a single broadcast message to all neighbors, avoiding the need of sending a distinct message to each neighbor. Furthermore, this algorithm reduces the number of computation steps needed at the cost of additional storage space in each node.

We believe that this algorithm can be applied as a building block for a wide variety of systems which use the BP algorithm in sensor network environments, in order to significantly reduce the energy consumption of the sensors. The proposed algorithm is not restricted to sensor networks. It can be used in other broadcast supporting networks like wireless and broadcast networks.

References

[1] M. Alanyali, S. Venkatesh, and O. S. S. Aeron. Distributed bayesian hypothesis testing in sensor networks. In *American Control Conference 2004*.

[2] C. Crick and A. Pfeffer. Loopy belief propagation as a basis for communication networks. In *In Proceedings of the 19th Conference on Uncertainty in AI, 2003*.

[3] Ihler, Fisher, Moses, and Willsky. Nonparametric belief propagation for self-calibration in sensor networks. In *Information Processing in Sensor Networks 2004*.

[4] C. C. Moallemi and B. V. Roy. Consensus propoagation. In *Technical Report, Stanford, June 2005*.

[5] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *In proc. of Uncertainty in AI, 1999*, pages 467–475.

[6] M. Paskin and C. Guestrin. Robust probabilistic inference in distributed systems. In *In the Twentieth Conference on Uncertainty in Art ficial Intelligence (UAI 2004), Banff, Canada, July 2004*.

[7] R. Rosales, K. Achan, and B. Frey. Learning to cluster using local neighborhood structure. In *Proc. of the 21st int. conf. on machine learning.*

[8] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-based reparameterization for approximate estimation on loopy graphs. In *NIPS 2001. Dec. 2001.*