Adaptive Belief Propagation

Georgios Papachristoudis GEOPAPA@MIT.EDU
John W. Fisher III FISHER@CSAIL.MIT.EDU
CSAIL, MIT, Cambridge, MA 02139, USA

Abstract

Graphical models are widely used in inference problems. In practice, one may construct a single large-scale model to explain a phenomenon of interest, which may be utilized in a variety of settings. The latent variables of interest, which can differ in each setting, may only represent a small subset of all variables. The marginals of variables of interest may change after the addition of measurements at different time points. In such adaptive settings, naïve algorithms, such as standard belief propagation (BP), may utilize many unnecessary computations by propagating messages over the entire graph. Here, we formulate an efficient inference procedure, termed *adaptive BP* (AdaBP), suitable for adaptive inference settings. We show that it gives exact results for trees in discrete and Gaussian Markov Random Fields (MRFs), and provide an extension to Gaussian loopy graphs. We also provide extensions on finding the most likely sequence of the entire latent graph. Lastly, we compare the proposed method to standard BP and to that of (Sümer et al., 2011), which tackles the same problem. We show in synthetic and real experiments that it outperforms standard BP by orders of magnitude and explore the settings that it is advantageous over (Sümer et al., 2011).

1. Introduction

Many estimation problems can be cast as inference in graphical models, where nodes represent variables of interest and edges between them indicate dependence relations. Naïve inference may have exponential complexity in the number of variables. Message passing algorithms, such as BP (Pearl, 1982) reduce the complexity significantly. Despite the fact that BP performs exact inference only on trees, it is often applied to loopy graphs (for which it is approximate) due to its computational efficiency.

We consider the problem of inference in large-scale models. Such models which arise, for example, in complex spatio-temporal phenomena, may be utilized in multiple settings. It is often the case that only a subset of latent variables is of interest for different applications which may vary from instance to instance (Rosales & Jaakkola, 2005; Chechetka & Guestrin, 2010; Wick & McCallum, 2011). Additionally, the set of available measurements may vary with use or become available at different points in time. The latter is common for any sequential estimation problem. In such situations, general-purpose inference algorithms, such as BP may utilize many unnecessary computations when only a small subset is desired. The complexity of such approaches becomes prohibitive as the size of graph increases, e.g., due to constant re-evaluation of messages. There exist several examples that fall into this category of problems. Patient monitoring provides one such practical example. Large-scale systems may monitor the health status of many patients; however, different physicians limit their interest to patients under their immediate care. Temperature monitoring sensors provide data over time and space, but sensitive areas (e.g., server room) may require more careful examination for the timely response in case of abnormal behavior. Lastly, in computational biology, the effects of mutations are explored (*computational mutagenesis*), with each putative mutation resulting in a very similar problem.

This motivates methods for problems where measurements are added incrementally and the interest is in a subset of node marginals at a given time point or the MAP sequence of the full latent graph. This is the problem of *adaptive inference*, where the goal is to take advantage of previously computed quantities instead of performing inference from scratch. In these cases, standard BP results in many redundant computations. Consequently, we develop an adaptive inference approach which avoids redundant computations and whose average-case performance shows significantly lower complexity compared to BP. The main idea is to send only messages between the node where a measurement has

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

been obtained from (w_ℓ) and the node whose marginal is of interest (v_ℓ).[1] The correctness of this approach is guaranteed by propagating messages between consecutive measurement nodes $w_{\ell-1}, w_\ell$ at every iteration as shown in Fig. 1. As a result, we only send the necessary messages to guarantee that the incoming messages to the node of interest v_ℓ are correct. We call this minimal messaging schedule *adaptive BP*. We show that it gives exact results on trees (as standard BP) and provide an extension for Gaussian loopy graphs that still guarantees exactness in the evaluation of marginals.

The proposed method requires a preprocessing step of $\mathcal{O}(N \log N)$ time, where N is the number of latent nodes. In the worst case, when relative distance between consecutive "measurement" nodes is approximately the tree diameter and the diameter is on the order of N (highly unbalanced tree), the performance is comparable – yet still faster to– standard BP. However, for height-balanced trees worst-case performance results in $\mathcal{O}(\log N)$ messages per update as compared to $\mathcal{O}(N)$ for standard BP. In the worst case, if distance of consecutive nodes is very small, the computation of the node marginal is obtained in constant time per iteration. We provide an extension of the method for MAP inference and for Gaussian loopy MRFs and show how it can be used to suggest nearly optimal measurement schedules. We compare the proposed method to (Sümer et al., 2011) and examine settings under one approach may have advantages over the other. Lastly, we empirically demonstrate the performance of our method in a variety of synthetic datasets, as well for two real applications.

Related Work. Sümer et al. consider the same problem in the context of factor graphs (Sümer et al., 2011) utilizing the factor elimination algorithm to evaluate node marginals (Darwiche & Hopkins, 2001). They construct a balanced representation of the elimination tree in $\mathcal{O}(|\mathcal{X}|^{3t_w} N)$ time, which allows for computation of a node marginal in $\mathcal{O}(|\mathcal{X}|^{2t_w} \log N)$, where N is the number of nodes, t_w is the elimination tree width (size of the largest clique in the chordal graph minus one) and $|\mathcal{X}|$ the alphabet size. However, the preprocessing step becomes prohibitive as $|\mathcal{X}|$ and t_w grow large, thus making this method inappropriate for dense loopy graphs. For trees, the width of the elimination tree is one and the complexity of updating the model reduces to $\mathcal{O}(|\mathcal{X}|^3 \log N)$ as compared to $\mathcal{O}(|\mathcal{X}|N)$ for standard BP. Note that they address the discrete case only. As we show later, the computational complexity is impacted significantly by not taking into account the relative distances between consecutive nodes of interest. (Wick & McCallum, 2011) consider the focused inference problem, where they propose a prioritized scheme of Metropolis Hastings algorithm that focuses on sampling variables from a set of interest. (Chechetka & Guestrin, 2010) examine the same problem. They create a prioritized message schedule weighted towards messages to which the set of interest is most sensitive. Their approach is limited to discrete graphs, the set of interest is fixed and the preprocessing time depends on the number of edges and neighbors of the nodes. In contrast, our proposed method extends to loopy Gaussian models, has a reduced pre-processing time, and allows for varying sets of interest.

[1] We will refer to w_ℓ as "measurement" node for abbreviation.

2. Problem Statement

Consider the Markov Random Field (MRF) which represents a graph $G = (V, \mathcal{E})$ of N latent variables, $X = \{X_1, \ldots, X_N\}$, whose direct dependencies are represented by edge set $\mathcal{E}$. Let's denote the neighbors of latent node X_k with $\mathcal{N}(k)$ and assume each latent node X_k is linked to m_k measurements $\{Y_{k,1:m_k}\}$. The set $\{Y_{k,1:m_k}\}$ will be called *observation set* and be denoted by $\mathcal{S}_k$. In addition, each $X_k \in \mathcal{X}$. A feedback vertex set (FVS) $\mathcal{F}$, is a set of nodes whose removal results in a cycle–free graph $\mathcal{T} = V \setminus \mathcal{F}$ (forest of trees). Obviously, $\mathcal{F} = \emptyset$ in the case of trees. We denote $|\mathcal{F}| = K$ to be the size of FVS. We would also call all nodes in $\mathcal{T}$ that are neighbors to an FV node as *anchors* and denote them by $\mathcal{A}$. That is, $\mathcal{A} = \{i \mid i \in \mathcal{T}, i \in \mathcal{N}(p), \forall p \in \mathcal{F}\}$.

For the purpose of our analysis, we focus on discrete MRFs, but the proposed method generalizes straightforwardly to Gaussian MRFs (Weiss & Freeman, 2001). Lastly, a common assumption is that measurements are conditionally independent on X. We focus on pairwise MRFs since it can be shown that MRFs with larger cliques can be reduced to pairwise ones (Wainwright et al., 2005). We consider three types of potentials; node potentials of latent variables, $\varphi_k^{(0)}(x_k)$, pairwise potentials between latent and observed variables, $\chi_{k\ell}(x_k, y_\ell)$, and pairwise potentials between latent variables, $\psi_{ij}(x_i, x_j)$. In the gaussian case, we assume that the variables follow a multivariate Gaussian with parameters h, J, where h is the full potential vector and J the full precision (information) matrix. Absence of an edge between two nodes i, j implies that $J_{ij} = 0$ and vice versa. We are interested in problems where a measurement is added at a time and only one or a few marginals are of interest at any point. The total number of available measurements is $M = \sum_{k=1}^{N} m_k$. Lastly, we are given a *measurement plan (order)* $\boldsymbol{w} = \{w_1, \ldots, w_M\} = \{w_{1:M}\}$, which provides the order of taking measurements from each set. That is, a measurement is obtained from set $\mathcal{S}_{w_1}$, then from $\mathcal{S}_{w_2}$, and so on. We call *marginal order*, $\boldsymbol{v} = \{v_{1:M}\}$, the sequence of the latent nodes whose marginal is of interest at each step.

Belief Propagation. Belief propagation is a message passing algorithm where two messages are propagated on each edge (i, j), one on each direction. A message from node i

to node j essentially contains all the information from the subtree rooted at i. In its serial version, an arbitrary node is chosen as a root, then messages are passed from leaves to the root, and then back to the leaves. In the discrete case, messages and marginal of interest are computed as

$$m_{i\to j}(x_j) = \sum_{x_i} \varphi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k\in\mathcal{N}(i)\setminus j} m_{k\to i}(x_i) \tag{1}$$

$$p_{\mathsf{X}_i}(x_i) \propto \varphi_i(x_i) \prod_{k\in\mathcal{N}(i)\setminus j} m_{k\to i}(x_i), \tag{2}$$

while in the gaussian case as

$$h_{i\to j} = -J_{ji}J_{i\setminus j}^{-1}h_{i\setminus j} \qquad J_{i\to j} = -J_{ji}J_{i\setminus j}^{-1}J_{ij} \tag{3}$$

$$\hat{h}_i = h_i + \sum_{k\in\mathcal{N}(i)} h_{k\to i} \qquad \hat{J}_i = J_{ii} + \sum_{k\in\mathcal{N}(i)} J_{k\to i}, \tag{4}$$

where $h_{i\setminus j} = h_i + \sum_{k\in\mathcal{N}(i)\setminus j} h_{k\to i}, J_{i\setminus j} = J_{ii} + \sum_{k\in\mathcal{N}(i)\setminus j} J_{k\to i}$. Belief propagation is a dynamic programming algorithm that takes advantage of the tree structure to avoid recomputing recurring quantities. The complexity of sending a message in the discrete case is $\mathcal{O}(|\mathcal{X}|^2)$, where $|\mathcal{X}|$ is the alphabet size. It is $\mathcal{O}(d^3)$ for the Gaussian case, where d is the dimension of the latent node. The overall complexity in its serial version is $\mathcal{O}(N|\mathcal{X}|^2)$.

Updating node potentials. Observed nodes are essentially absorbed into the node potential of the latent node they link to. If a measurement $\mathsf{Y}_{w_\ell,u} = y_u$ is added at iteration ℓ linking to node X_{w_ℓ}, it updates node X_{w_ℓ} potential as

$$\varphi_{w_\ell}^{(\ell)}(x_{w_\ell}) = \varphi_{w_\ell}^{(\ell-1)}(x_{w_\ell})\chi_{w_\ell u}(x_{w_\ell}, y_u),$$

where $\varphi_{w_\ell}^{(\ell-1)}(\cdot)$ is the node potential before the incorporation of measurement y_u. Updating the node potential requires $\mathcal{O}(|\mathcal{X}|)$ time. In the Gaussian case, incorporating a measurement in node w_ℓ is as straightforward. It becomes clear that all the information about observed nodes is easily absorbed into the latent node potentials. Thus, we will henceforth only consider the graph of latent nodes.

3. Method Description

For the purposes of analysis, we would delay the discussion to general Gaussian MRFs until Sec. 5. We will consider trees here and show later an extension to Gaussian loopy graphs. As a reminder, we obtain one measurement at every step and are interested in finding the marginal at a given node. We show in the supplement how this can be extended to multiple measurements or nodes of interest at a time. The measurement order $\boldsymbol{w} = \{w_1, \dots, w_M\}$ is the order that measurements are obtained, while the marginal order $\boldsymbol{v} = \{v_1, \dots, v_M\}$ determines the marginals of interest at any time. Again, the key idea is to propagate messages in the paths $(w_{\ell-1}, w_\ell)$ and $(w_\ell, v_\ell), \forall \ell$. The path between any two nodes can be determined trivially if their *lowest common ancestor* (lca) is known. The problem of finding the lca of two nodes is well-studied and is related to the famous Range Minimum Query (RMQ) problem, which returns the index of the minimum element between two specified indices of an array $\boldsymbol{A}$. This algorithm, as discussed in (Harel & Tarjan, 1984), requires the building of a structure $\boldsymbol{M}$ of size $N \times L$, where $L = \lceil \log_2 N \rceil + 1$, which returns the index of the minimum array element between two specified indices of $\boldsymbol{A}$ in constant time. It is extremely well-suited for problems with a large number of queries, R, where $R \gg N$, since it is linear in R. It turns out that the LCA can be reduced to the RMQ problem as shown below (Czumaj et al., 2007), (Fischer & Heun, 2007).

Lowest Common Ancestor. For a specified root, each node is labeled in a breadth-first manner. That is, the root is assigned label 1, and all other nodes are labeled accordingly in a top-down, left-right approach (cf. Fig. 1). As we mentioned above, the index of the minimum element in the subarray $\boldsymbol{A}[i \dots j]$ is provided in constant time by building the RMQ structure. Now, suppose we recover the Euler tour $\boldsymbol{E}$ of the tree starting from the root. As a reminder, the *Euler tour* of a strongly connected, directed graph G is a cycle that traverses each edge of G exactly once, although it may visit a node more than once (Cormen et al., 2009). Since we are dealing with undirected graphs here, we assume for the purposes of analysis that each undirected edge is equivalent to two directed edges of opposing direction. The number of edges entering or exiting a node is called the *in-degree* or *out-degree*, respectively. Since by construction each node has equal out- and in-degree, the Euler tour is always a cycle, that is, it starts and ends on the same node (here, the root). If we denote by $\boldsymbol{H}$ the vector which stores the index of the first occurrence of each node in $\boldsymbol{E}$, the lca of nodes $w_{\ell-1}, w_\ell$ would be somewhere in $\boldsymbol{E}[\boldsymbol{H}_{w_{\ell-1}}, \dots, \boldsymbol{H}_{w_\ell}]$ due to the way the Euler tour is constructed (depth-first manner). Since the nodes are labeled in a breadth-first manner, the lca of $w_{\ell-1}, w_\ell$ would be the one with the smallest label and hence the smallest depth in the range $\boldsymbol{E}[\boldsymbol{H}_{w_{\ell-1}}, \dots, \boldsymbol{H}_{w_\ell}]$. It becomes apparent that we need to introduce a vector $\boldsymbol{D}_e$ which would store the depth of the corresponding nodes in the Euler tour. For example, the depth of the first node in the Euler tour is $[\boldsymbol{D}_e]_1 = 0$, because it is the root. Since the lca of $w_{\ell-1}, w_\ell$ is the node with the smallest depth in $\boldsymbol{E}[\boldsymbol{H}_{w_{\ell-1}}, \dots, \boldsymbol{H}_{w_\ell}]$, the index of the minimum element of subarray $\boldsymbol{D}_e[\boldsymbol{H}_{w_{\ell-1}}, \dots, \boldsymbol{H}_{w_\ell}]$ would give us the $\text{lca}(w_{\ell-1}, w_\ell)$.

It remains now to build a matrix $\boldsymbol{M}$ that would provide answers to queries of the type $\arg\min \boldsymbol{D}_e[\boldsymbol{H}_{w_{\ell-1}}, \dots, \boldsymbol{H}_{w_\ell}]$ in constant time. The size of this matrix would be $N \times L$, where $L = \lceil \log_2 N \rceil + 1$, while element $[\boldsymbol{M}]_{i,j}$ would represent the index of the minimum element of the subarray

$\boldsymbol{D}_e$ that starts at i and has length 2^{j-1}:

$$[\boldsymbol{M}]_{i,j} = \begin{cases} [\boldsymbol{M}]_{i,j-1} & , [\boldsymbol{D}_e]_{[\boldsymbol{M}]_{i,j-1}} \leq [\boldsymbol{D}_e]_{[\boldsymbol{M}]_{r,j-1}} \\ [\boldsymbol{M}]_{\min\{i+2^{j-2},N\},j-1} & , \text{otherwise,} \end{cases}$$

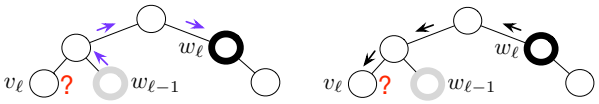
where $i = 1, \ldots, N$, $j = 1, \ldots, L$, $r = \min\{i + 2^{j-2}, N\}$ and $[\boldsymbol{M}]_{i,1} = i$. The absolute index of the minimum value of $\boldsymbol{D}_e[i, \ldots, j]$ is recovered in constant time as

$$\text{RMQ}_{\boldsymbol{D}_e}(i,j) = \begin{cases} [\boldsymbol{M}]_{i,k+1} & , [\boldsymbol{D}_e]_{[\boldsymbol{M}]_{i,k+1}} \leq [\boldsymbol{D}_e]_{[\boldsymbol{M}]_{s,k+1}} \\ [\boldsymbol{M}]_{j-2^k+1,k+1} & , \text{otherwise,} \end{cases}$$

where $k = \lfloor \log_2(j - i + 1) \rfloor$ and $s = j - 2^k + 1$. The lca of $w_{\ell-1}$ and w_ℓ is simply

$$\text{lca}(w_{\ell-1}, w_\ell) = \begin{cases} \boldsymbol{E}_{\text{RMQ}_{\boldsymbol{D}_e}(\boldsymbol{H}_{w_{\ell-1}}, \boldsymbol{H}_{w_\ell})}, \boldsymbol{H}_{w_{\ell-1}} < \boldsymbol{H}_{w_\ell} \\ \boldsymbol{E}_{\text{RMQ}_{\boldsymbol{D}_e}(\boldsymbol{H}_{w_\ell}, \boldsymbol{H}_{w_{\ell-1}})}, \text{otherwise.} \end{cases} \tag{5}$$

Adaptive BP. With a careful inspection, we observe that after the incorporation of a measurement at node X_{w_ℓ}, the evaluation of the messages along the unique path from node w_ℓ to node v_ℓ is sufficient for the determination of node v_ℓ's marginal. This is the key point of the *adaptive BP* algorithm. The above procedure guarantees to give the correct marginals along this path as long as all the incoming messages to node w_ℓ are correct. This is possible, if we additionally propagate messages from $w_{\ell-1}$ to w_ℓ at every iteration. The algorithm is described as follows. During initialization, all node potentials we propagate messages along the entire graph in both directions. At this point, as a new measurement arrives from set $\mathcal{S}_{w_1}$, the messages from w_1 to v_1 are computed. This way, the marginals of the nodes in the path that connects w_1, v_1 (incl. w_1, v_1) are correctly updated. Then, we propagate messages from w_1 to w_2, update the node potential of X_{w_2} and send messages from w_2 to v_2. We continue with this procedure for each ℓ. If $w_\ell = w_{\ell-1}$, no messages are propagated from $w_{\ell-1}$ to w_ℓ, while if $w_\ell = v_{\ell-1}$, only the node potential X_{w_ℓ} is updated. Obviously, the path from node $w_{\ell-1}$ to w_ℓ is directly related to the lca$(w_{\ell-1}, w_\ell)$. Similarly, for the pair (w_ℓ, v_ℓ). Therefore, at every iteration, we need to determine the lcas of these two pairs, which is accomplished in constant time, with the reduction to the RMQ problem. Once we find the lca of pair $(w_{\ell-1}, w_\ell)$, we can trivially determine the directed path from $w_{\ell-1}$ to w_ℓ by traversing from $w_{\ell-1}$ up to lca$(w_{\ell-1}, w_\ell)$ and then down to w_ℓ. We will denote the messages in this path by $\mathcal{M}(w_{\ell-1} \to w_\ell)$. Similarly, we denote the messages of the directed path from w_ℓ to v_ℓ by $\mathcal{M}(w_\ell \to v_\ell)$. Note here that both of the above schedules contain only the single-direction messages from one node to another. The update is done in the same manner as in the serial version of BP, that is, we propagate messages from $w_{\ell-1}$ to the lca$(w_{\ell-1}, w_\ell)$ and then down to w_ℓ. The procedure is the same for the pair (w_ℓ, v_ℓ). A flow of the algorithm and a detailed description are provided in Fig. 1 and Alg. 1, respectively.



(a) Messages from $w_{\ell-1}$ to w_ℓ (b) Messages from w_ℓ to v_ℓ

Figure 1. The bold node in black represents the current measurement node w_ℓ, while the bold node in gray the previous measurement node $w_{\ell-1}$. The node in question mark represents the node of interest v_ℓ. **(a)** In the first phase of an iteration, we propagate messages from $w_{\ell-1}$ to w_ℓ (depicted in purple color). **(b)** In the second phase, we propagate messages from w_ℓ to v_ℓ (depicted in black color) after we have updated the node potential at w_ℓ, which has changed due to the addition of a new measurement.

Messages are updated as

$$m_{i\to j}(x_j) = \sum_{x_i} \varphi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in \mathcal{N}(i)\setminus j} m_{k\to i}(x_i),$$
$$\forall (i,j) \in \mathcal{M}(w_{\ell-1} \to w_\ell) \text{ and } \mathcal{M}(w_\ell \to v_\ell),$$

while the marginal of node of interest v_ℓ, is computed from Eq. (2). If the latent graph is a chain, there is no need to find the lca: we simply propagate from $w_{\ell-1}$ to w_ℓ, update the node potential of w_ℓ and propagate messages to v_ℓ.

Proposition 1. *Alg. 1 correctly updates the messages in path* $\mathcal{M}(w_{\ell-1} \to w_\ell), \forall \ell$.

(Proof of this and subsequent corollaries in the supplement)

Corollary 1. *Alg. 1 correctly updates the messages in path* $\mathcal{M}(w_\ell \to v_\ell), \forall \ell$.

Corollary 2. *Alg. 1 provides the exact marginals of all nodes in path* $\mathcal{M}(w_\ell \to v_\ell), \forall \ell$.

Algorithm 1 ADAPTIVE BP

Preprocessing
Determine Euler tour $\boldsymbol{E}$, depths of elements in the Euler tour $\boldsymbol{D}_e$, vector $\boldsymbol{H}$ which stores the index of the first occurrence of node i in $\boldsymbol{E}$, and matrix $\boldsymbol{M}$ which stores the index of the minimum value of the subarray of $\boldsymbol{D}_e$ starting at i and having length 2^{j-1}.
Initialization
Initialize the node, pairwise potentials and messages.
Iteration
for $\ell = 1, \ldots, M$ **do**
Find lca$(w_{\ell-1}, w_\ell)$ from Eq. (5).
Determine the schedule $\mathcal{M}(w_{\ell-1} \to w_\ell)$.
Compute messages $m_{i\to j}(x_j)$ in $\mathcal{M}(w_{\ell-1} \to w_\ell)$.
Update the node potential at X_{w_ℓ}.
Find lca(w_ℓ, v_ℓ) from Eq. (5).
Determine the schedule $\mathcal{M}(w_\ell \to v_\ell)$.
Compute messages $m_{i\to j}(x_j)$ in $\mathcal{M}(w_\ell \to v_\ell)$.
Compute the marginal of interest $p_{X_{v_\ell}}(x_{v_\ell})$.
end for

Complexity. If the depth of each node (D) is not known in advance, it can be retrieved in $\mathcal{O}(N)$ time, in a depth-first approach. Similarly, the Euler tour is also retrievable in linear time. The same holds for vectors $\boldsymbol{D}_e$ and $\boldsymbol{H}$. Lastly, the creation of matrix $\boldsymbol{M}$, takes $\mathcal{O}(N \log_2 N)$ time and space. Therefore, the overall complexity of preprocessing is $\mathcal{O}(N \log_2 N)$. For adaptive BP, we only need to send messages along the directed paths $\mathcal{M}(w_{\ell-1} \to w_\ell)$ and $\mathcal{M}(w_\ell \to v_\ell)$. The number of messages to be sent in step ℓ is $\text{dist}(w_{\ell-1}, w_\ell) + \text{dist}(w_\ell, v_\ell)$.[2] The overall complexity is $\mathcal{O}(\sum_{\ell=1}^{M}(\text{dist}(w_{\ell-1}, w_\ell) + \text{dist}(w_\ell, v_\ell))|\mathcal{X}|^2)$. Compare this with standard BP, where $2(N-1)$ messages are sent at each iteration resulting in an overall complexity of $\mathcal{O}(mN^2|\mathcal{X}|^2)$, assuming that the number of measurements from each set is the same, $m_k = m, \forall k$. As we see, the complexity of adaptive BP directly depends on the context of the measurement and marginal order, while standard BP has a fixed cost per iteration. We will analyze the worst, best and average complexity of adaptive BP for balanced and unbalanced trees. In the worst-case, when the tree is highly unbalanced (tree diameter on the order of N) and the relative distance between $(w_{\ell-1}, w_\ell)$, (w_ℓ, v_ℓ) is comparable to the diameter for all ℓ, we need to transmit $\mathcal{O}(N)$ messages at every iteration. In this case, the order of the number of messages to be sent is the same with standard BP. If, instead, the latent graph is a balanced tree, with each node having approximately q children, $\mathcal{O}\left(\lfloor \log_q N \rfloor\right)$ messages are propagated at every iteration in the worst case. In the best-case scenario, if $w_{\ell-1}, w_\ell, v_\ell$ are akin to each other (e.g., parent-child or siblings) for every ℓ, then only one or two messages are propagated at every iteration, which reduces the overall complexity to just $\mathcal{O}\left(mN|\mathcal{X}|^2\right)$. As expected, when there is small distance between pairs of nodes $(w_{\ell-1}, w_\ell)$, (w_ℓ, v_ℓ), the complexity is substantially reduced. Complexity only depends on the relative distance between consecutive terms. Structure comes only into consideration, in the worst case, when the relative distance between $(w_{\ell-1}, w_\ell)$ and (w_ℓ, v_ℓ) are consistently comparable to the tree diameter.

4. Extension to Max-Product

In the case of max-product, we replace sum with max and introduce a new type of messages, called *delta messages*, that will be used for the recovery of the MAP sequence. A delta message indicates the value of the source node that corresponds to the MAP sequence of the subtree rooted at the source node (excl. the branch containing the target node) for a specific value of the target node. In order to recover the MAP sequence, we need to propagate delta messages from $w_{\ell-1}$ to w_ℓ and then backtrack from w_ℓ down to the leaves (considering w_ℓ as the root).

[2]The distance between nodes w, v is the length of the path connecting them and equals $\text{dist}(w, v) = D_v + D_w - 2D_{\text{lca}(w,v)}$.

In general, obtaining the MAP sequence is a linear operation in the number of nodes. However, local changes in node potentials might induce only small changes in the MAP sequence. We should note that the only delta messages pointing towards the root w_ℓ that change, are the ones the path $\mathcal{M}(w_{\ell-1} \to w_\ell)$, which are correctly updated during iteration ℓ. A visualization of the algorithm is provided in the supplement. This observation can help us recover the MAP sequence in a more efficient way. We create an indicator sparse matrix, whose rows represent the source and columns the target of a delta message. We assign value 1 to any delta message that became "dirty" (changed) in the most recent iteration. That is, every message in $\mathcal{M}(w_{\ell-1} \to w_\ell)$. Therefore, when we backtrack from w_ℓ down to the leaves, we must consider the effect that these changed messages have in the MAP sequence. Nevertheless, if a node's maximizing value remains the same (with the previous iteration), then the MAP subsequences of the subtrees rooted at the neighbors of this node will also remain the same since the delta messages from these neighbors to the node stayed intact. Because of that, there is no need to backtrack further down to the subtrees of a node's neighbors, if this node's maximizing value did not change between consecutive iterations.

5. Extension to Gaussian Loopy MRFs

Adaptive BP can be extended to Gaussian loopy graphs by using the Feedback Message Passing (FMP) algorithm by (Liu et al., 2012). Their algorithm provides a way to evaluate the exact means and variances in loopy Gaussian MRFs. On the first step, an FVS $\mathcal{F}$ is determined from one of the existing algorithms. Then, the exact means and variances are obtained in two rounds. In the first round, BP runs on $\{h_\mathcal{T}, J_\mathcal{T}\}$, where $h_\mathcal{T}, J_\mathcal{T}$ correspond to the part of full potential vector and block of full information matrix that contain only nodes in $\mathcal{T}$. We also run BP $|\mathcal{F}| = K$ more times with parameters $\{h^p, J_\mathcal{T}\}_{p\in\mathcal{F}}$, where $h^p = J_{\mathcal{T}p}$. In other words, h^p is the part of column of J that corresponds to node p and contains only the rows that correspond to nodes in $\mathcal{T}$. This provides us with "partial" means and variances $\hat{\mu}_i^\mathcal{T}, \hat{\Sigma}_{ii}^\mathcal{T}, \forall i \in V$ as well as "feedback gains" $g_i^p, \forall i \in V, p \in \mathcal{F}$ that will be used in the second round of the method. In the second round, the means and variances in FVS $\mathcal{F}$ are evaluated by inverting $\hat{J}_\mathcal{F}$ and the exact means and variances are retrieved by running BP one more time and adding correction terms to the estimated variances from the first round, $\hat{\Sigma}_{ii}^\mathcal{T}$ (see Alg. 2 for more details).

After briefly outlining FMP, we move on by describing the extension of AdaBP to Gaussian loopy graphs. As a reminder, once we determine the FVS $\mathcal{F}$, the remaining graph $\mathcal{T}$ is a tree. Let's denote by $w_\ell^\mathcal{T}$ the node from $\mathcal{T}$

Algorithm 2 FEEDBACK MESSAGE PASSING (FMP)

1. Construct K potential vectors: $h^p = J_{\mathcal{T}p}, \forall p \in \mathcal{F}$.
2. Run BP $K+1$ times on $\mathcal{T}$ with parameters $\{h_{\mathcal{T}}, J_{\mathcal{T}}\}$, $\{h^p, J_{\mathcal{T}}\}_{p\in\mathcal{F}}$, which will produce messages $h^{\mathcal{T}}_{i\to j}, h^p_{i\to j}, J^{\mathcal{T}}_{i\to j}$ and marginals $\hat{\mu}^{\mathcal{T}}_i, g^p_i, \hat{\Sigma}^{\mathcal{T}}_{ii}$.
3. Obtain the K–sized graph with updated parameters $\hat{h}_{\mathcal{F}}, \hat{J}_{\mathcal{F}}$ as
$$[\hat{J}_{\mathcal{F}}]_{pq} = J_{pq} - \sum_{i\in\mathcal{N}(p)\cap\mathcal{T}} J_{pi} g^q_i, \ \forall p,q \in \mathcal{F} \quad (6)$$
$$[\hat{h}_{\mathcal{F}}]_p = h_p \ - \sum_{i\in\mathcal{N}(p)\cap\mathcal{T}} J_{pi}\hat{\mu}^{\mathcal{T}}_i, \ \forall p \in \mathcal{F} \quad (7)$$
and solve for $\Sigma_{\mathcal{F}} = \hat{J}^{-1}_{\mathcal{F}}$ and $\mu_{\mathcal{F}} = \Sigma_{\mathcal{F}}\hat{h}_{\mathcal{F}}$.
4. Revise the potential vector in $\mathcal{T}$ as $\tilde{h}_i = h_i - \sum_{j\in\mathcal{N}(i)\cap\mathcal{F}} J_{ij}[\mu_{\mathcal{F}}]_j, \ \forall i \in \mathcal{T}$ and obtain the exact means by running BP one more time on the revised potential vector (the corresponding messages will be denoted by $\tilde{h}^{\mathcal{T}}_{i\to j}$).
5. Correct the variances with
$$\Sigma_{ii} = \hat{\Sigma}^{\mathcal{T}}_{ii} + \sum_{p\in\mathcal{F}}\sum_{q\in\mathcal{F}} g^p_i[\Sigma_{\mathcal{F}}]_{pq} g^q_i, \ \forall i \in \mathcal{T}. \quad (8)$$

where a measurement has been obtained most recently

$$w^{\mathcal{T}}_\ell = \begin{cases} w_\ell & \text{, if } w_\ell \in \mathcal{T} \\ w^{\mathcal{T}}_{\ell-1} & \text{, otherwise.} \end{cases}$$

Let's further denote the subtrees of $\mathcal{T}$ rooted at $w^{\mathcal{T}}_\ell, v_\ell \in \mathcal{T}$ with the nodes in $\mathcal{A}$ as their leaves by $\mathcal{T}^w_\ell$ and $\mathcal{T}^v_\ell$, respectively. Depending on the size of the anchor set $\mathcal{A}$, and the allocation of its nodes inside $\mathcal{T}$, subtrees $\mathcal{T}^w_\ell, \mathcal{T}^v_\ell$ can be much smaller than $\mathcal{T}$, $|\mathcal{T}^w_\ell|, |\mathcal{T}^v_\ell| \ll |\mathcal{T}|$.

As we see from steps 4 and 5 of Alg. 2, the evaluation of the marginal at v_ℓ requires the knowledge of $\mu_{\mathcal{F}}, \Sigma_{\mathcal{F}}$ which in turn require the knowledge of "partial" means $\hat{\mu}^{\mathcal{T}}_i$ and "feedback gains" g^p_i at the anchors $\mathcal{A}, \forall p \in \mathcal{F}$. The quantities $\hat{\mu}^{\mathcal{T}}_i, g^p_i, \forall i \in \mathcal{A}$ will be correct as long as the messages between w_ℓ and $\mathcal{A}$ are correct, which is guaranteed by sending messages between consecutive measurement nodes as we did in the tree case. The difference in the loopy case is that since some measurement nodes might belong to the FVS, we should always propagate messages from the most recent measurement node in $\mathcal{T}$ (that is, $w^{\mathcal{T}}_\ell$), to the next measurement node $w_\ell \in \mathcal{T}$ to ensure consistency. If $w_\ell \notin \mathcal{T}$, propagation is not necessary. For the evaluation of v_ℓ's marginal, we further need to propagate from $w^{\mathcal{T}}_\ell$ to all nodes in $\mathcal{A}$, which ensures the correctness of all incoming messages to nodes in $\mathcal{A}$. After that, the potential vector $\hat{h}_{\mathcal{F}}$ and information matrix $\hat{J}_{\mathcal{F}}$ are updated correctly from Eqs. (6), (7) which leads to the right mean and covariance $\mu_{\mathcal{F}}, \Sigma_{\mathcal{F}}$ of the FVS $\mathcal{F}$. If $v_\ell \in \mathcal{F}$, we retrieve the marginal from $\mu_{\mathcal{F}}, \Sigma_{\mathcal{F}}$ as $[\mu_{\mathcal{F}}]_{v_\ell}, [\Sigma_{\mathcal{F}}]_{v_\ell v_\ell}$. Otherwise, we revise the potential vectors according to Step 4 of Alg. 2 and propagate messages $h^p_{i\to j}, \forall p$ from $w^{\mathcal{T}}_\ell$ to v_ℓ. Tables 1, 2, 3 summarize the messaging protocol for every iteration ℓ. A more detailed discussion is provided in the supplement.

Table 1. Messages between $w_{\ell-1}, w_\ell$

	$w_\ell \in \mathcal{T}$	$w_\ell \in \mathcal{F}$
$w_{\ell-1} \in \{\mathcal{F}, \mathcal{T}\}$	Send $J^{\mathcal{T}}_{i\to j}, h^{\mathcal{T}}_{i\to j}, h^p_{i\to j}$ in $\mathcal{M}(w^{\mathcal{T}}_{\ell-1} \to w_\ell)$	–

Table 2. First-round messages between w_ℓ, v_ℓ

	$v_\ell \in \{\mathcal{F}, \mathcal{T}\}$
$w_\ell \in \{\mathcal{F}, \mathcal{T}\}$	Send $J^{\mathcal{T}}_{i\to j}, h^{\mathcal{T}}_{i\to j}, h^p_{i\to j}$ in $\mathcal{M}(w^{\mathcal{T}}_\ell \to \mathcal{A})$

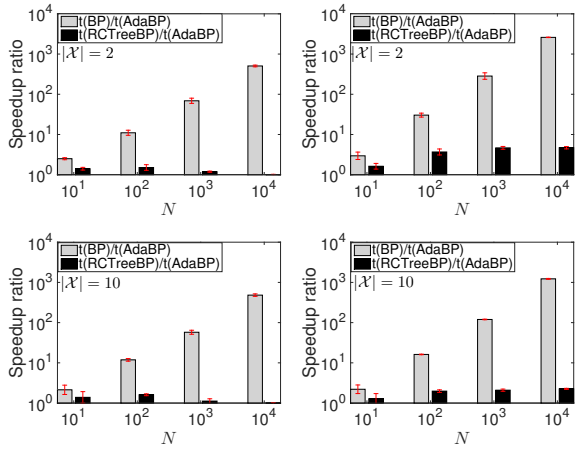
Table 3. Second-round messages between w_ℓ, v_ℓ

	$v_\ell \in \mathcal{T}$	$v_\ell \in \mathcal{F}$
$w_\ell \in \{\mathcal{F}, \mathcal{T}\}$	Send $\tilde{h}^{\mathcal{T}}_{i\to j}$ in $\mathcal{M}(\mathcal{A} \to v_\ell)$ Send $h^p_{i\to j}$ in $\mathcal{M}(w^{\mathcal{T}}_\ell \to v_\ell)$	– –

In terms of complexity, we first need to determine the FVS. Even though, finding the minimum FVS is NP-complete, there are approximate algorithms that find an FVS with size comparable to the optimal. For example, (Bafna et al., 1999) provide a 2-approximation, which runs in $\mathcal{O}(\min\{|\mathcal{E}|\log N, N^2\})$ time. At every iteration we need to send $(K+2)\text{dist}(w^{\mathcal{T}}_{\ell-1}, w_\ell)$ messages between $w^{\mathcal{T}}_{\ell-1}$ and w_ℓ, if $w_\ell \in \mathcal{T}$ and $(K+2)(|\mathcal{T}^w_\ell| - 1)$ messages between $w^{\mathcal{T}}_\ell$ and nodes in $\mathcal{A}$. If, in addition, $v_\ell \in \mathcal{T}$, the propagation of $(|\mathcal{T}^v_\ell| - 1)$ $\tilde{h}^{\mathcal{T}}_{i\to j}$ messages between the anchors $\mathcal{A}$ and v_ℓ is necessary, plus $K\text{dist}(w^{\mathcal{T}}_\ell, v_\ell)$ $h^p_{i\to j}$ messages from $w^{\mathcal{T}}_\ell$ to v_ℓ. Therefore, we send $\mathcal{O}(K(\text{dist}(w^{\mathcal{T}}_{\ell-1}, w_\ell) + \text{dist}(w^{\mathcal{T}}_\ell, v_\ell) + |\mathcal{T}^w_\ell|) + |\mathcal{T}^v_\ell|)$ messages per iteration. Compare this to the $\mathcal{O}(K|\mathcal{T}|)$ messages per iteration of standard FMP. To understand the difference in complexity, let's assume for the shake of exposition that $|\mathcal{T}^w_\ell| \geq \text{dist}(w^{\mathcal{T}}_{\ell-1}, w_\ell), \text{dist}(w^{\mathcal{T}}_\ell, v_\ell), |\mathcal{T}^v_\ell|$. This means that the complexity of adaptive BP is $\mathcal{O}(K|\mathcal{T}^w_\ell|)$, which results in a speedup on the order of $\mathcal{O}(|\mathcal{T}|/|\mathcal{T}^w_\ell|)$, since it always holds that $|\mathcal{T}^w_\ell| \leq |\mathcal{T}|$. Therefore, adaptive BP is consistently faster than standard FMP.

6. Experiments

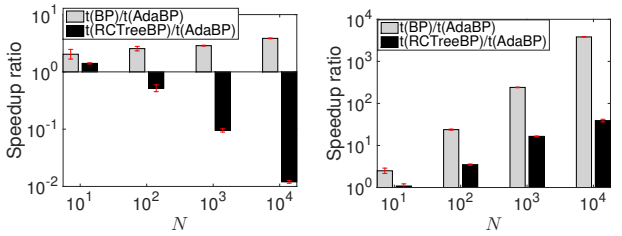
Henceforth, we refer to the proposed algorithm as AdaBP, the method of (Sümer et al., 2011) as RCTreeBP, and standard BP as BP. We use a publicly available version of RCTreeBP. In addition, when we make use of the term "consecutive elements", we mean consecutive measurement elements $w_{\ell-1}, w_\ell$ and concurrent measurement and marginal



(a) Unconstrained $\boldsymbol{w}$ (b) Constrained $\boldsymbol{w}$

Figure 2. Comparison of the total running times of AdaBP against standard BP (gray) and RCTreeBP (black) over different alphabet sizes, $|\mathcal{X}| \in \{2, 10\}$. **(a)** Distance between consecutive elements $\mathbb{E}[\text{dist}(w_{\ell-1}, w_\ell)]$ is unconstrained. **(b)** $\mathbb{E}[\text{dist}(w_{\ell-1}, w_\ell)] \leq |\mathcal{X}| \log N$. For average distance $\mathbb{E}[\text{dist}(w_{\ell-1}, w_\ell)]$ smaller than $|\mathcal{X}| \log N$, AdaBP is 1.3–4.7 faster than RCTreeBP.

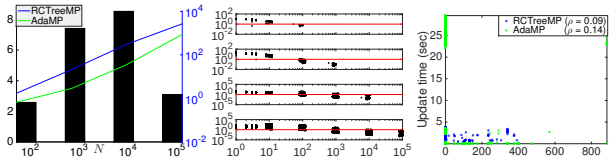
elements w_ℓ, v_ℓ. Recall that updates per iteration in RCTreeBP have complexity $\mathcal{O}(|\mathcal{X}|^3 \log N)$ (for trees), while complexity is $\mathcal{O}(|\mathcal{X}|^2(\text{dist}(w_{\ell-1}, w_\ell) + \text{dist}(w_\ell, v_\ell)))$ for AdaBP. Our experiments demonstrate that AdaBP is consistently orders of magnitude faster than standard BP (except in the worst case), and outperforms RCTreeBP when the average distance between consecutive elements is less than $|\mathcal{X}| \log N$ (see Fig. 2(b)). Conversely, if the tree diameter is much greater than $|\mathcal{X}| \log N$ and the average distance between consecutive elements is comparable to the tree diameter, AdaBP yields worse performance than RCTreeBP. We consider the following synthetic experiment where we construct unbalanced trees of sizes $N \in \{10, 10^2, 10^3, 10^4\}$. We repeat the above procedure $R = 10$ times for each N, by randomly constructing a new tree. For each tree, we randomly generate different $\boldsymbol{w}$ orders of size N and for simplicity of analysis we set $\boldsymbol{v} = \boldsymbol{w}$, so that only the distance between consecutive measurement nodes affects the computation. Figs. 2(a) and 2(b) compare the ratios of running times of AdaBP against standard BP and RCTreeBP (different rows correspond to different alphabet sizes). In all cases both AdaBP and RCTreeBP significantly outperform standard BP. Fig. 2(a) considers the case of randomly generated $\boldsymbol{w}$. When there is no restriction on the distance between consecutive elements, both AdaBP and RCTreeBP are comparable. However, for average distance between consecutive elements less than $|\mathcal{X}| \log N$, AdaBP is 1.3–4.7 times faster than RCTreeBP. Fig. 3(a) and 3(b) consider worst and best case performance of AdaBP, respectively. In the former, we generate several different instances of a Markov chain of varying sizes and con-



(a) $\mathbb{E}\{\text{dist}(w_{\ell-1}, w_\ell)\} \sim \mathcal{O}(N)$ (b) $\mathbb{E}\{\text{dist}(w_{\ell-1}, w_\ell)\} = 2$

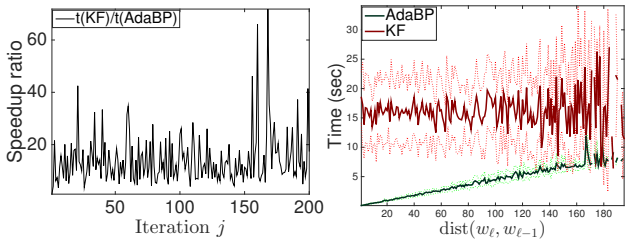
Figure 3. **(a)** Worst case. Distance between consecutive elements is on the order of N. AdaBP is comparable to standard BP (still being 2–4 times faster) and orders of magnitude slower than RCTreeBP. **(b)** Best case. Consecutive elements are very close to each other. Only a constant number of updates is required per step for AdaBP.

struct the measurement and marginal orders, $\boldsymbol{w}$ and $\boldsymbol{v}$ such that there is at least $2N/3$ distance between consecutive elements. In the latter case, we consider different instances of a star graph (tree diameter: 2) of varying sizes and randomly create measurement and marginal orders (which by construction do not have consecutive elements of more than 2 nodes apart). As expected, in Fig. 3(a), RCTreeBP outperforms AdaBP for worst-case $\boldsymbol{w}$ (that is, when there is large distance between consecutive elements), yet still outperforms BP by a factor of 2–4. However, in Fig. 3(b) we see that AdaBP is 4–49 times faster than RCTreeBP and up to thousand times faster than BP. Next, we consider application of AdaMP (MP denotes max-product) to biological data. Specifically, we explore the effects of pointwise mutations in DNA sequences to the birth or disappearance of CpG islands. CpG islands are regions of DNA with high percentage of cytosine (C) occurring next to guanine (G) nucleotides and are believed to be responsible for upstream gene regulation. Usually, CpG island detection is modeled as an HMM problem where hidden nodes are binary variables which indicate the presence (or absence) of a CpG region and observed variables correspond to the observed DNA sequence comprised of the four nucleotides {A,T,C,G}. The goal is to find the MAP sequence (CpG regions) that best explains the observed data (DNA sequence). In *computational mutagenesis*, changes in the location of CpG islands are of interest due to mutations in the DNA sequence (Acar et al., 2009). We compare AdaMP and RCTreeMP on varying-size stretches (10^2–10^5 bp) of human chromosome 21 obtained from the NCBI database. We train the parameters of HMM with one of the standard CpG prediction tools, CpG Island Searcher (Takai & Jones, 2002). We perform a mutation every other nucleotide for each DNA-pair stretch and compare the running times of both methods under different criteria in Fig. 4. In this experiment, $v_\ell = w_\ell, \forall \ell$. Fig. 4(a) shows the speedup of AdaMP over RCTreeMP for varying sizes of DNA sequence. For medium to large sequences, AdaMP exhibits



(a) (b) (c)

Figure 4. **(a)** Speedups of AdaMP over RCTreeMP for varying-size stretches of chr 21 (10^2–10^5 bp). **(a)** Left y-axis shows the speedup over RCTreeMP, while right y-axis the actual running times in sec (represented as lines). **(b)** Ratios of update times of AdaMP over RCTreeMP for different values of dist$(w_{\ell-1}, w_\ell)$ (x-axis: dist$(w_{\ell-1}, w_\ell)$, y-axis: speedup). The four log-log plots correspond to four different DNA stretches of $10^2, 10^3, 10^4, 10^5$ bp size, respectively. For smaller distances, AdaMP outperforms, but for distances closer to the graph size N, RCTreeMP is preferable. Red line indicates ratio of 1. **(c)** Both methods are not very sensitive to changes in the MAP sequence between consecutive iterations (x-axis: # of bp that differ between consecutive MAP sequences).



(a) (b)

Figure 5. **(a)** This figure shows the speedup over Kalman filter (KF). AdaBP is 1–42 times faster than standard kalman filtering/smoothing techniques. **(b)** Running time per iteration of AdaBP and KF as a function of consecutive distance between elements. AdaBP is much more sensitive to dist$(w_{\ell-1}, w_\ell)$ and as the figure suggests it is much faster than KF when dist$(w_\ell, w_{\ell-1})$ is small. Dotted plots represents deviation due to different runs of the experiment.

better performance, however, for very large sequences of size $\sim 10^5$, the computational cost of determining the MAP sequence is nearly linear with the graph size (even though the cost of updating the delta messages remains remarkably low). In contrast, RCTreeMP depends only on the number of variables which changed since the previous iteration. Fig. 4(b) examines the relationship in performance to the distance between consecutive elements for DNA stretches of varying size (10^2–10^5 bp). As expected, AdaMP is very sensitive to the distance between consecutive elements dist$(w_{\ell-1}, w_\ell)$. On the contrary, RCTreeMP depends only on the graph size N. AdaMP is preferred for measurement schedules with low average dist$(w_{\ell-1}, w_\ell)$ (points above the red line), while RCTreeMP average distance comparable to graph size (points below the red line). Lastly, Fig. 4(c) shows that both methods are not very sensitive to changes in the MAP sequence between consecutive iterations.

As a second experiment, we analyzed temperature measurements collected from 53 wireless sensors at 30 sec intervals from the Intel Berkeley Research lab. We modeled the latent temperatures in the various locations of the lab as a grid graph. We assume that measurements obtained from a sensor are a noisy representation of the temperatures around its close vicinity. We further assume that temperatures evolve over time following linear dynamics as $X_t = AX_{t-1} + V_{t-1}$, where $V_{t-1} \sim \mathcal{N}(0, Q)$ and X_t represents the temperatures of the lab at time t. We learn parameters A and Q by training the data between Feb 28 and Mar 7, 2004 on a Normal-inverse-Wishart model. One of the primary goals in this setting is to estimate the covariance of the latent variables after the incorporation of measurements. Since the problem is modeled as a Gaussian HMM, we can use the Kalman filter/smoothing updates. We use measurements in a 6-hour window on Feb 28, 2014 on a random order and compare the update times of AdaBP versus standard Kalman filter/smoothing updates (RCTreeBP is not included for comparison here, since it is not applicable to Gaussian models). We see in Fig. 5(a), that AdaBP is consistently (1–42 times) faster than Kalman filtering/smoothing. Also, in Fig. 5(b), we observe the direct dependence of AdaBP to distance between consecutive elements $(w_{\ell-1}, w_\ell)$, (w_ℓ, v_ℓ), which makes it more appropriate for problems with small average distance.

7. Discussion

We presented a new algorithm, AdaBP, which is particularly suited to sequential inference problems, when there is little or no knowledge of the measurement schedule in advance. In addition, when we can design the measurement order, we show in the supplement how to propose a nearly optimal schedule by casting it as a shortest Hamiltonian path problem. We compared the method to standard BP and RCTreeBP. In the case of trees, standard BP incurs a prohibitive cost ($\mathcal{O}(N)$ messages per iteration), while AdaBP sends only the necessary messages between consecutive elements. We provided an extensive analysis of the algorithmic complexity with respect to the measurement $\boldsymbol{w}$ and marginal schedule $\boldsymbol{v}$. We showed that when consecutive measurement and marginal elements $w_{\ell-1}, w_\ell, v_\ell$ are akin to each other, the complexity per iteration is of order $\mathcal{O}(1)$ and hence the overall complexity is $\mathcal{O}(N)$ (provided the sizes of $\boldsymbol{w}, \boldsymbol{v}$ are close to N). In addition, advanced knowledge of the measurement nodes allows for a nearly minimal schedule design. Lastly, we showed extensions of the algorithm to Gaussian loopy graphs and to the most likely sequence problem (which applies on the full latent graph). An implementation of this algorithm is publicly available at `https://github.com/geopapa11/adabp`

Acknowledgments

The authors would like to thank the reviewers for their valuable comments and suggestions. This work was partially supported by the Army Research Office (ARO) Multidisciplinary Research Initiative (MURI) program (Award number W911NF-11-1-0391), NSF/DNDO Collaborative Research ARI-LA (Award ECCS-1348328), and by the Department of Energy (NA-22) Consortium for Verification Technology.

References

Acar, U. A., Ihler, A. T., Mettu, R. R., and Sümer, Ö. Adaptive Updates for MAP Configurations with Applications to Bioinformatics. In *IEEE/SP 15th Workshop on Statistical Signal Processing (SSP)*, August 2009.

Bafna, V., Berman, P., and Fujito, T. A 2-Approximation Algorithm for the Undirected Feedback Vertex Set Problem. *SIAM Journal on Discrete Mathematics*, 12(3): 289–297, Sep 1999. ISSN 0895-4801.

Chechetka, A. and Guestrin, C. Focused Belief Propagation for Query-Specific Inference. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, May 2010.

Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. *Introduction to Algorithms*. McGraw-Hill Higher Education, 3rd edition, 2009.

Czumaj, A., Kowaluk, M., and Lingas, A. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. *Theoretical Computer Science*, 380(1-2): 37–46, July 2007.

Darwiche, A. and Hopkins, M. Using recursive decomposition to construct elimination orders, jointrees, and dtrees. In *Trends in Artificial Intelligence, Lecture Notes in AI*, pp. 180–191. Springer-Verlag, 2001.

Fischer, J. and Heun, V. A New Succinct Representation of RMQ-Information and Improvements in the Enhanced Suffix Array. In *Proceedings of the 1st International Conference on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, ESCAPE'07, pp. 459–470. Springer-Verlag, 2007.

Harel, D. and Tarjan, R. E. Fast Algorithms for Finding Nearest Common Ancestors. *SIAM Journal on Computing*, 13(2):338–355, May 1984.

Liu, Y., Chandrasekaran, V., Anandkumar, A., and Willsky, A. S. Feedback Message Passing for Inference in Gaussian Graphical Models. *IEEE Transactions on Signal Processing*, 60(8):4135–4150, Aug 2012.

Pearl, J. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the American Association of Artificial Intelligence National Conference (AAAI)*, pp. 133–136, 1982.

Rosales, R. and Jaakkola, T. S. Focused Inference. In *Proceedings of the 10th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 317–324, January 2005.

Sümer, Ö., Acar, U. A., Ihler, A. T., and Mettu, R. R. Adaptive exact inference in graphical models. *Journal of Machine Learning Research*, 12:3147–3186, Nov 2011.

Takai, D. and Jones, P. A. Comprehensive analysis of CpG islands in human chromosomes 21 and 22. *Proceedings of the National Academy of Sciences (PNAS)*, 99 (6):3740–3745, March 2002.

Wainwright, M. J., Jaakkola, T. S., and Willsky, A. S. MAP estimation via agreement on trees: Message-passing and linear programming. *IEEE Transactions on Information Theory*, 51:2005, 2005.

Weiss, Y. and Freeman, W. T. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13:2173–2200, 2001.

Wick, M. L. and McCallum, A. Query-Aware MCMC. In *Advances in Neural Information Processing Systems 24*, pp. 2564–2572, 2011.