Exploiting Data-Independence for Fast Belief-Propagation

Julian J. McAuley JULIAN.MCAULEY@NICTA.COM.AU
Tibério S. Caetano TIBERIO.CAETANO@NICTA.COM.AU
NICTA and Australian National University, Canberra ACT 0200 Australia

Abstract

Maximum a posteriori (MAP) inference in graphical models requires that we maximize the sum of two terms: a *data-dependent* term, encoding the conditional likelihood of a certain labeling given an observation, and a *data-independent* term, encoding some prior on labelings. Often, data-dependent factors contain fewer latent variables than data-independent factors – for instance, many grid and tree-structured models contain only first-order conditionals despite having pairwise priors. In this paper, we note that MAP-inference in such models can be made substantially faster by appropriately preprocessing their data-independent terms. Our main result is to show that message-passing in any such pairwise model has an expected-case exponent of only 1.5 on the number of states per node, leading to significant improvements over existing quadratic-time solutions.

1. Introduction

MAP-inference in a graphical model $\mathcal{G}$ consists of solving an optimization problem of the form

$$\hat{\mathbf{y}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} \sum_{C \in \mathcal{C}} \Phi_C(\mathbf{y}_C|\mathbf{x}_C), \tag{1}$$

where $\mathcal{C}$ is the set of maximal cliques in $\mathcal{G}$. This problem is often solved via *message-passing* algorithms such as the junction-tree algorithm, loopy belief-propagation, or inference in a factor graph (Aji & McEliece, 2000; Kschischang et al., 2001).

Computing messages between two intersecting cliques A and B in general involves solving a problem of the form

$$m_{A\to B}(\mathbf{y}_{A\cap B}) = \max_{\mathbf{y}_{A\setminus B}} \Phi_A(\mathbf{y}_A|\mathbf{x}_A) \sum_{D\in\Gamma(A)\setminus\{B\}} m_{D\to A}(\mathbf{y}_{A\cap D}), \tag{2}$$

where $\Gamma(A)$ is the set of cliques that intersect with A. If the nodes of our model have N states, solving (eq. 2) appears to require $\Theta(N^{|A|})$ operations, since there are $N^{|A|}$ possible values of $\mathbf{y}_A$.

Alternately, (eq. 1) can be expressed in the form

$$\hat{\mathbf{y}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} \underbrace{\sum_{F\in\mathcal{F}} \Phi_F(\mathbf{y}_F|\mathbf{x}_F)}_{\text{data dependent}} + \underbrace{\sum_{C\in\mathcal{C}} \Phi_C(\mathbf{y}_C)}_{\text{data independent}}, \tag{3}$$

where each $F \in \mathcal{F}$ is a subset of some $C \in \mathcal{C}$.

In this paper, we show that much faster algorithms can be developed whenever the model's *data-dependent* factors contain fewer latent variables than its *data-independent* factors, or equivalently when every $F \in \mathcal{F}$ is a *proper subset* of some $C \in \mathcal{C}$ in (eq. 3). Although our results apply to general models of this form, we shall mainly be concerned with the most common case, in which we have pairwise models with data-independent priors, or problems of the form

$$\hat{\mathbf{y}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}} \underbrace{\sum_{i\in\mathcal{N}} \Phi_i(y_i|x_i)}_{\text{node potential}} + \lambda \underbrace{\sum_{(i,j)\in\mathcal{E}} \Phi_{i,j}(y_i,y_j)}_{\text{edge potential}}. \tag{4}$$

This encompasses a wide variety of models, including grid-structured models for optical flow and stereo disparity as well as chain and tree-structured models for text or speech. Examples are shown in Figure 1. In all of these examples, we give a solution to (eq. 2) with an *expected-case running time* of only $O(N^{1.5})$, while to our knowledge, the best currently known solution is the naïve $\Theta(N^2)$ version. Our result is achieved by preprocessing the data-independent part of the model *offline*, simply by sorting the rows and columns of $\Phi_{i,j}$.

As our optimizations apply directly to the message passing equations themselves, they can be applied

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

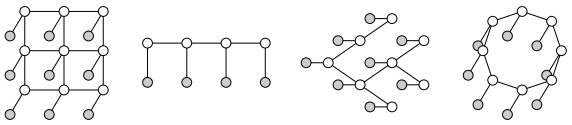


Figure 1. Some graphical models to which our results apply: *cliques containing observations have fewer latent variables than purely latent cliques*. Gray nodes correspond to the observation, white nodes to the labeling. In other words, cliques containing a gray node encode the *data likelihood*, whereas cliques containing only white nodes encode *priors*. We focus on the case where the gray nodes have degree one (i.e., they are connected to only one white node).

to many variants of belief-propagation, such as the junction-tree algorithm, loopy belief-propagation, and factor graphs. In particular, in models where belief-propagation is known to produce the correct solution, i.e., trees (and junction trees in general), our optimizations result in the asymptotically fastest solution for exact inference.

1.1. Related Work

There has been previous work on speeding-up message-passing algorithms by exploiting some type of structure in the graphical model. For example, Kersting et al. (2009) study the case where different cliques share the same potential function. In Felzenszwalb & Huttenlocher (2006), fast message-passing algorithms are provided for the case in which the potential of a 2-clique is only dependent on the *difference* of the latent variables (which is common in some computer vision applications); they also show how the algorithm can be made faster if the graphical model is a bipartite graph. In Kumar & Torr (2006), the authors provide faster algorithms for the case in which the potentials are *truncated*, whereas in Petersen et al. (2008) the authors offer speedups for models that are specifically grid-like.

The latter work is perhaps the most similar in spirit to ours, as it exploits the fact that certain factors can be *sorted* in order to reduce the search space of a certain maximization problem. In practice, this leads to linear speedups over a $\Theta(N^4)$ algorithm. We too shall rely on sorting to reduce the search space of a maximization problem, but additionally we exploit *data-independence* to reduce a $\Theta(N^2)$ algorithm to $O(N^{1.5})$.

Notably, our assumption of *data-independence in the prior* differs substantially from those above; it is arguably a much weaker assumption since it is in fact satisfied by most graphical models of practical interest.

2. Our Approach

We consider an (undirected) pairwise graphical model $\mathcal{G}(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of nodes, and $\mathcal{E}$ the set of edges, which factorizes according to (eq. 4). In such a model, computing a message between two neighboring cliques $A = (i, j)$ and $B = (i, k)$ is equivalent in complexity to solving

$$m_{A\to B}(y_i) = \Psi_i(y_i) + \max_{y_j} \Psi_j(y_j) + \Phi_{i,j}(y_i, y_j), \quad (5)$$

where $\Psi_i(y_i)$ is the sum of $\Phi(y_i|x_i)$ and any first-order messages over y_i (similarly for $\Psi_j(y_j)$). The general form of (eq. 5) encompasses many variants of belief-propagation. In all such cases, solving (eq. 5) appears to be a $\Theta(N^2)$ operation, since this is the number of possible arguments to $\Phi_{i,j}$.

For a specific value of $y_i = q$, solving (eq. 5) amounts to solving

$$m_{A\to B}(q) = \Psi_i(q) + \max_{y_j} \underbrace{\Psi_j(y_j)}_{\mathbf{v}_a} + \underbrace{\Phi_{i,j}(q, y_j)}_{\mathbf{v}_b}, \quad (6)$$

which appears to have linear time complexity, as it is equivalent to solving

$$\max_i \{\mathbf{v}_a[i] + \mathbf{v}_b[i]\}. \quad (7)$$

However, we note that solving (eq. 7) is only $O(\sqrt{N})$ *if we know the permutations that sort* $\mathbf{v}_a$ *and* $\mathbf{v}_b$. Our algorithm for solving (eq. 7) is given in Algorithm 1; the execution of this algorithm is explained in Figure 2. For now, we simply state the following theorem regarding the algorithm's running time:

Theorem 1. *The* ***expected*** *running time of Algorithm 1 is* $O(\sqrt{N})$. *This yields a speedup of at least* $\Omega(\sqrt{N})$ *in models containing pairwise priors and unary data-likelihoods.*

We have recently employed an algorithm of this type in McAuley & Caetano (2010), where we showed that fast algorithms can be developed for inference in graphical models whose maximal cliques are larger than their factors. Further discussion of this theorem can be found in Section 3; complete proofs are given in McAuley & Caetano (2009).

An apparent issue is that the cost of sorting every row of $\Phi_{i,j}$ is $\Theta(N^2 \log N)$ (i.e., more expensive than the naïve solution). However we make the observation that this cost can be circumvented *so long as only the data-independent part of the potential is maximal* (i.e., the prior, such as in (eq. 6)). In such cases, the data-independent part of the model can be sorted *offline*.

Algorithm 1 Find i that maximizes $\mathbf{v}_a[i] + \mathbf{v}_b[i]$

Require: permutation functions p_a and p_b that sort $\mathbf{v}_a$ and $\mathbf{v}_b$ in decreasing order

1: **Initialize:** $start \leftarrow 1$
2: $end_a \leftarrow p_a^{-1}[p_b[1]]$ {end_a is the index of the element in $\mathbf{v}_a$ corresponding to the largest element in $\mathbf{v}_b$; see the red line in Figure 2}
3: $end_b \leftarrow p_b^{-1}[p_a[1]]$
4: $best \leftarrow \text{argmax}_{i \in \{p_a[1], p_b[1]\}} \{\mathbf{v}_a[i] + \mathbf{v}_b[i]\}$
5: $max \leftarrow \mathbf{v}_a[best] + \mathbf{v}_b[best]$
6: **while** $start < end_a \wedge start < end_b$ **do**
7: {consider the indices $p_a[start]$ and $p_b[start]$}
8: $start \leftarrow start + 1$
9: **if** $\mathbf{v}_a[p_a[start]] + \mathbf{v}_b[p_a[start]] > max$ **then**
10: $best \leftarrow p_a[start]$
11: $max \leftarrow \mathbf{v}_a[best] + \mathbf{v}_b[best]$
12: **end if**
13: **if** $p_b^{-1}[p_a[start]] < end_b$ **then**
14: $end_b \leftarrow p_b^{-1}[p_a[start]]$
15: **end if**
16: {repeat Lines 9–15, interchanging a and b}
17: **end while**
18: **while** $start < end_a$ **do**
19: {we have considered all candidate values in p_b, but some may remain in p_a}
20: $start \leftarrow start + 1$
21: **if** $\mathbf{v}_a[p_a[start]] + \mathbf{v}_b[p_a[start]] > max$ **then**
22: $best \leftarrow p_a[start]$
23: $max \leftarrow \mathbf{v}_a[best] + \mathbf{v}_b[best]$
24: **end if**
25: **end while**
26: {repeat Lines 18–25, interchanging a and b}
27: **return** $best$ {this takes *expected time* $O(\sqrt{N})$}

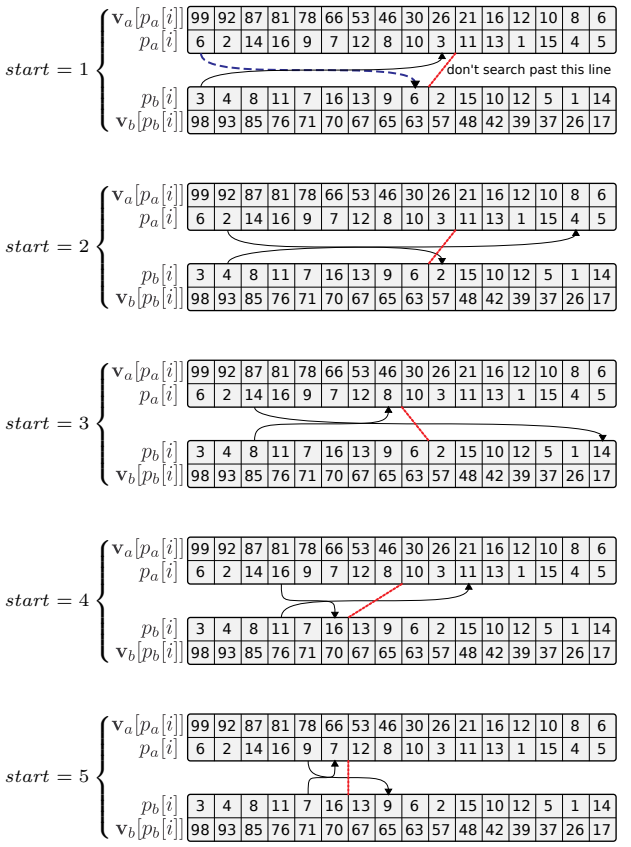


Figure 2. Algorithm 1 explained (best viewed in color): the two arrows connect $\mathbf{v}_a[p_a[start]]$ to $\mathbf{v}_b[p_a[start]]$, and $\mathbf{v}_a[p_b[start]]$ to $\mathbf{v}_b[p_b[start]]$; the red line connects end_a and end_b, which is updated each time an arrowhead lies to its left; we only consider those arrows that whose tail lies to the left of the red line – all others can be ignored; a dashed arrow is shown when a new maximum is found.

Once $\Phi_{i,j}$ has been sorted, (eq. 6) can be solved via Algorithm 2.[1]

Often the prior is *homogeneous* (every edge uses the same prior), meaning that $\Phi_{i,j}$ can be sorted *online*, so long as $|\mathcal{E}| \in \Omega(\log N)$ (i.e., the number of messages that must be computed is asymptotically larger than $\log N$). Similarly, when using iterative inference schemes (such as loopy belief-propagation), the sorting step takes place only during the first iteration; if inference is run for $\Omega(\log N)$ iterations, then speed improvements can still be obtained with online sorting.

3. Runtime Analysis

In this section, we compute the expected-case running time of Algorithm 2 under the assumption that the rows and columns of the data-independent terms have been sorted offline. First note that if Algorithm 1 solves (eq. 7) in $O(f(N))$, then Algorithm 2 must take $O(N \log(N) + N f(N))$; thus we need only compute $f(N)$. We shall demonstrate that $f(N) \in O(\sqrt{N})$ as stated in Theorem 1.

We consider the *expected-case* running time, under the assumption that the order statistics of $\mathbf{v}_a$ and $\mathbf{v}_b$ are independent. It is worth mentioning that we are limited to expected-case analysis, as there is provably no deterministic solution that is sub-linear in N: otherwise we could solve max-sum matrix multiplication (or 'funny matrix multiplication') in $O(N^{2.5})$, though it is known to have no deterministic sub-cubic solution (assuming that only addition and comparison operators are used) (Kerr, 1970; Alon et al., 1997).

The running time of Algorithm 1 depends on the permutation matrix that transforms the sorted values of

[1] C++ implementations of our algorithms are available at http://users.cecs.anu.edu.au/~julianm/

Algorithm 2 Solve (eq. 6) using Algorithm 1

Require: a set of permutation functions $\mathbf{p}$ such that p_i sorts the i^{th} row of $\Phi_{i,j}$ in decreasing order.

compute the permutation function p_a by sorting Ψ_j {takes $\Theta(N \log N)$}
for $q \in \{1 \dots N\}$ **do**
  $(\mathbf{v}_a, \mathbf{v}_b) \leftarrow (\Psi_j, \Phi_{i,j}(q, y_j | x_i, x_j))$
  $r \leftarrow$ *Algorithm 1* $(\mathbf{v}_a, \mathbf{v}_b, p_a, p_q)$ $\{O(\sqrt{N})\}$
  $m_{A \rightarrow B}(q) \leftarrow \Psi_i(q) + \Psi_j(r) + \Phi_{i,j}(q, r | x_i, x_j)$
end for {*expected-case* $O(N\sqrt{N})$}
return $m_{A \rightarrow B}$

$\mathbf{v}_a$ into the sorted values of $\mathbf{v}_b$, which in turn depends only on their order statistics.

Figure 3 gives examples of different permutation matrices, and the number of addition operations that each induces. Here we see that our method is $\Theta(1)$ when the two vectors have the same order statistics, and $\Theta(N)$ when the two vectors have 'opposite' order statistics. Our analysis considers the case that the order statistics of $\mathbf{v}_a$ and $\mathbf{v}_b$ are independent (i.e., every permutation matrix is equally likely). Further details and proofs are given in McAuley & Caetano (2009).

As stated in Figure 3, we can compute the number of additions as follows: starting from the top-left of the permutation matrix, find the smallest gray square that contains an entry; if this square has width M, we perform fewer than $2M$ additions. For a randomly chosen permutation matrix, simple inspection reveals that the probability that $M > m$ is given by

$$P_N(M > m) = \frac{(N-m)!(N-m)!}{(N-2m)!N!}, \tag{8}$$

and thus, using the identity $E(X) = \sum_{x=1}^{\infty} P(X \geq x)$, we can show that the expected value of M is

$$E_N(M) = \sum_{m=0}^{\lfloor N/2 \rfloor} \frac{(N-m)!(N-m)!}{(N-2m)!N!}, \tag{9}$$

which can be shown to be $\Omega(\log N)$ and $O(\sqrt{N})$ (Theorem 1). Thus we can solve (eq. 2) in expected-case $\Theta(NE_N(M))$ which is $O(N^{1.5})$. We shall verify these statements experimentally in Section 5.1.

3.1. Correlated Data

As shown in Figure 3, our algorithm will perform better or worse than the expected-case depending on the particular permutations that sort the data. If the data are positively correlated we will tend to observe permutations that lie close to the main diagonal (the left side of Figure 3); if the data are negatively correlated we will tend to observe permutations that lie close to the off-diagonal (the right side of Figure 3). In such cases, we shall observe better or worse performance (respectively) than the expected-case.

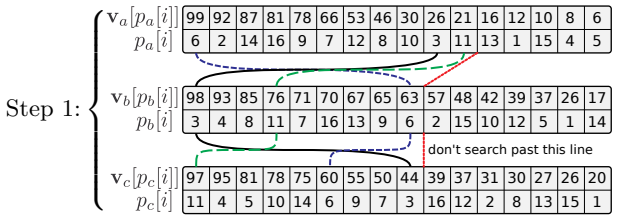


Figure 4. Algorithm 1 can be generalized to handle any number of lists. For K lists (corresponding to K^{th}-order priors), it has running time $O(KN^{\frac{K-1}{K}})$.

4. Generalizations

As we have suggested, our results apply not only to pairwise models, but to *any models whose data-dependent cliques have fewer latent variables than the data-independent cliques*; in this section we shall state our main result in this general case.

We have shown that Algorithm 1 solves (eq. 7) in $O(\sqrt{N})$. By similar reasoning, it can be shown that this algorithm can be adapted to solve

$$\max_i \{\mathbf{v}_1[i] \times \mathbf{v}_2[i] \times \dots \times \mathbf{v}_K[i]\} \tag{10}$$

in sub-linear time. As with our solution to (eq. 7), we can substantially reduce the search space if we know the permutations that sort the lists (see Figure 4). The running time of our algorithm is given by the following theorem (a proof is given in (McAuley & Caetano, 2009)):

Theorem 2. *Algorithm 1 generalizes to K lists with an expected running time of $O(KN^{\frac{K-1}{K}})$ (it can be adapted to be $O(\min(N, KN^{\frac{K-1}{K}}))$, if we carefully avoid rereading entries).*

This generalization can be applied as follows: if the *data-independent* factors are of dimension K (within cliques containing more than K terms), we can obtain an expected speedup of $\Omega(\frac{1}{K}N^{\frac{1}{K}})$; setting $K = 2$ recovers precisely the $\Omega(\sqrt{N})$ speedup for pairwise priors discussed so far in this paper.

An example application to which this generalization can be applied is that of Felzenszwalb (2005). This model contains a third-order geometric prior, while the *data-dependent* factors are only pairwise. Our method

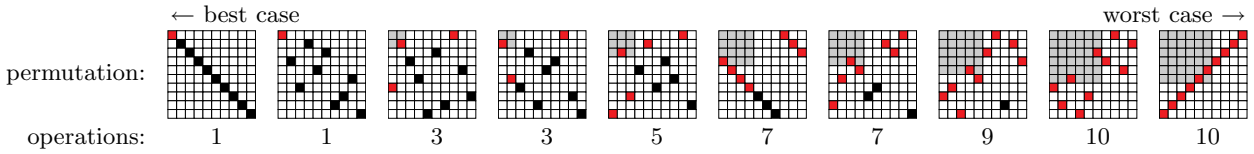


Figure 3. Different permutation matrices and their resulting cost (in terms of additions performed). Each permutation matrix transforms the *sorted* values of one list into the sorted values of the other, i.e., it transforms $\mathbf{v}_a$ as sorted by p_a into $\mathbf{v}_b$ as sorted by p_b. For instance, if there is an entry in the first row and fifth column, this indicates that $p_a[1] = p_b[5]$ (equivalently that $p_b^{-1}[p_a[1]] = 5$, or $p_a^{-1}[p_b[5]] = 1$), meaning that the largest value of $\mathbf{v}_a$ has the same index as the fifth largest value of $\mathbf{v}_b$. The red squares show the entries that must be read before the algorithm terminates (each corresponding to one addition). In reference to Algorithm 1, an entry in row number *start* corresponds to computing $\mathbf{v}_a[p_a[start]] + \mathbf{v}_b[p_a[start]]$; similarly, the entry in column number *start* corresponds to computing $\mathbf{v}_a[p_b[start]] + \mathbf{v}_b[p_b[start]]$. A simple method to determine the number of additions is as follows: starting from the top-left of the permutation matrix, find the smallest gray square that contains an entry; if this square has width M, we shall read fewer than $2M$ entries. Note that the width of this gray square is precisely the value of *start* when the algorithm terminates.

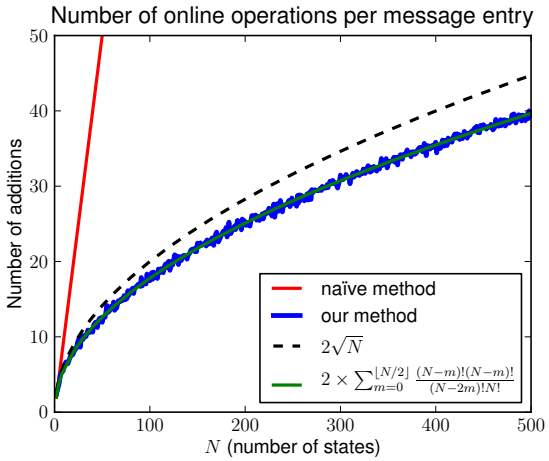


Figure 5. The number of addition operations required to compute each entry of a message (average of 10 trials). The naïve solution requires $\Theta(N)$ operations, whereas our method requires $O(\sqrt{N})$ in the expected-case. The exact expectation is also shown.

allows us to pass messages in this model in $O(N^{\frac{8}{3}})$, i.e., $\Omega(N^{\frac{1}{3}})$ faster than the standard cubic solution.

As the pairwise case is by far the most common, and as it gives the largest speedup, we shall focus on this case in our experiments.

5. Experiments

5.1. Number of Addition Operations

Figure 5 shows the number of addition operations required to solve to (eq. 7); multiplying by $N+1$ gives the number of operations required to compute (eq. 5), i.e., the entire message. $\mathbf{v}_a$ and $\mathbf{v}_b$ are chosen by sampling uniformly from $[0, 1)^N$, and the average of 10 trials is shown. The value reported is precisely the value of $2 \times$ *start* when Algorithm 1 terminates. This confirms that the expected value given in (eq. 9) matches the experimental performance, and also verifies that the expectation is upper-bounded by $2 \times \sqrt{N}$.

Due to the computational overhead of our solution, we expect the running time of our algorithm to differ from the value shown in Figure 5 by a multiplicative constant, except in cases where $\mathbf{v}_a$ and $\mathbf{v}_b$ are highly (positively or negatively) correlated.

5.2. Inference in Pairwise Models

In each of the following experiments we perform belief-propagation in models of the form given in (eq. 4). Thus each model is completely specified by defining the node potentials $\Phi_i(y_i|x_i)$, the edge potentials $\Phi_{i,j}(y_i, y_j)$, and the topology $(\mathcal{N}, \mathcal{E})$ of the graph.

Furthermore we assume that the edge potentials are *homogeneous*, i.e., that the potential for each edge is the same, or rather that they have the same order statistics (for example, they may differ by a multiplicative or additive constant). This means that the sorting can be done *online* without affecting the asymptotic complexity. When subject to heterogeneous potentials we need merely sort them *offline*; the online cost shall be similar to what we report here.

5.2.1. Chain-Structured Models

In this section, we consider *chain-structured* graphs. Here we have nodes $\mathcal{N} = \{1 \dots Q\}$, and edges $\mathcal{E} = \{(1, 2), (2, 3) \dots (Q-1, Q)\}$. The max-sum algorithm is known to compute the maximum-likelihood solution exactly for tree-structured models.

Figure 6 (top) shows the performance of our method on a model with *random* potentials, i.e., $\Phi_i(y_i|x_i) = U[0, 1)$, $\Phi_{i,i+1}(y_i, y_{i+1}) = U[0, 1)$, where $U[0, 1)$ is the

uniform distribution. Fitted curves are superimposed onto the running time, confirming that the performance of the standard solution grows quadratically with the number of states, while ours grows at a rate of $N^{1.5}$. The residual error r shows how closely the fitted curve approximates the running time; in the case of random potentials, both curves have similar constants.

Figure 6 (bottom) shows the performance of our method on a 'text-denoising' experiment. In this experiment random errors are introduced into a body of text, which the model aims to correct. Here we have $\Phi_i(y_i|x_i) = \lambda(1 - I_{\{x_i\}}(y_i))$, i.e., a constant cost λ is incurred in the event that x_i and y_i are not equal. $\Phi_{i,i+1}(y_i, y_{i+1})$ returns the frequency of the pair (y_i, y_{i+1}) in a training corpus. This experiment was performed on text from each language in the Leipzig Multilingual Corpora (Quasthoff et al., 2006). The first 100,000 characters were used to construct the pairwise statistics of $\Phi_{i,i+1}$, and the next 2,500 characters were used for the denoising experiment.

Although our algorithm results in a faster solution for all languages, we observe a higher constant of complexity than that obtained for random data. This suggests that the pairwise priors in different languages are not independent of the messages; the higher residual error may also suggest that different languages have different order-statistics.

5.2.2. Grid-Structured Models

Similarly, we can apply our method to *grid-structured* models. Here we resort to loopy belief-propagation to approximate the MAP solution, though indeed the same analysis applies in the case of factor graphs (Kschischang et al., 2001). We construct a 50×50 grid model and perform loopy belief-propagation using a random message-passing schedule for five iterations. In these experiments our nodes are $\mathcal{N} = \{1 \dots M\}^2$, and our edges connect the 4-neighbors (similar to the grid shown in Figure 1 (left)).

Figure 7 (top) shows the performance of our method on a grid with random potentials (similar to the experiment in Section 5.2.1). Figure 7 (bottom) shows the performance of our method on an optical flow task (Lucas & Kanade, 1981). Here the states encode *flow vectors*: for a node with N states, the flow vector is assumed to take integer coordinates in the square $[-\sqrt{N}/2, \sqrt{N}/2)^2$ (so that there are N possible flow vectors). For the unary potential we have

$$\Phi_{(i,j)}(y|x) = \left\| Im_1[i,j] - Im_2[(i,j) + f(y)] \right\|, \quad (11)$$

where $Im_1[a,b]$ and $Im_2[a,b]$ return the gray-level of the pixel at (a,b) in the first and second images (respectively), and $f(y)$ returns the flow vector encoded by y. The pairwise potentials simply encode the Euclidean distance between two flow vectors.

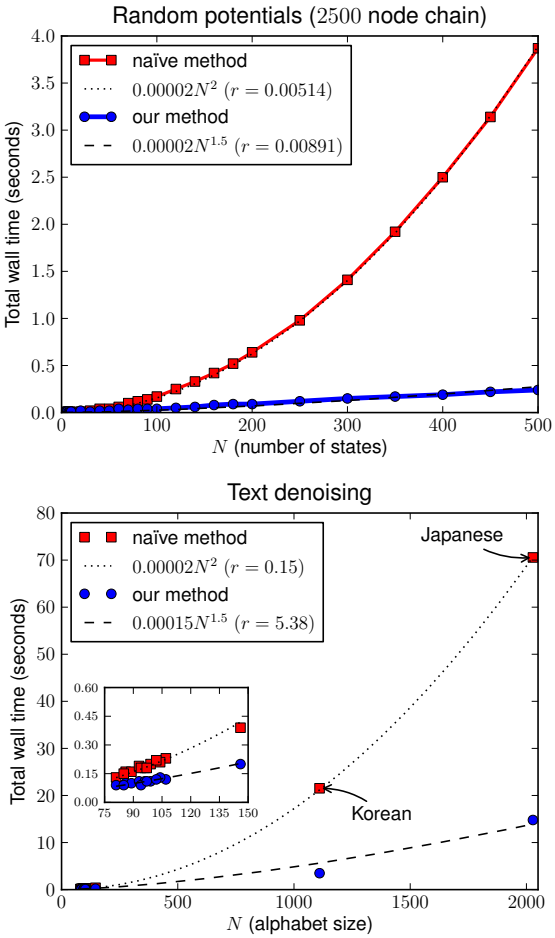


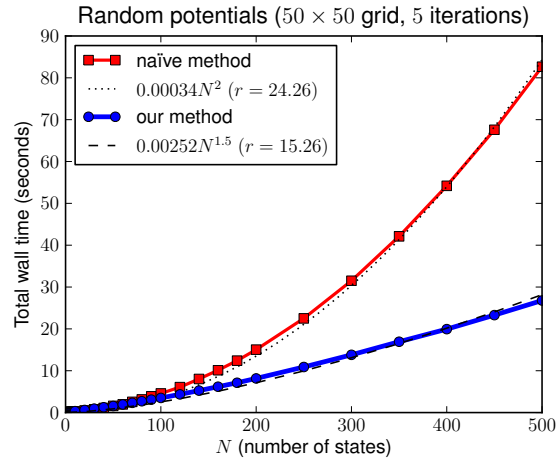
Figure 6. Running time of inference in chain-structured models: random potentials (top), and text denoising (bottom). Fitted curves confirm that the exponent of our method is indeed 1.5 (r denotes the residual error, i.e., the 'goodness' of the fitted curve).

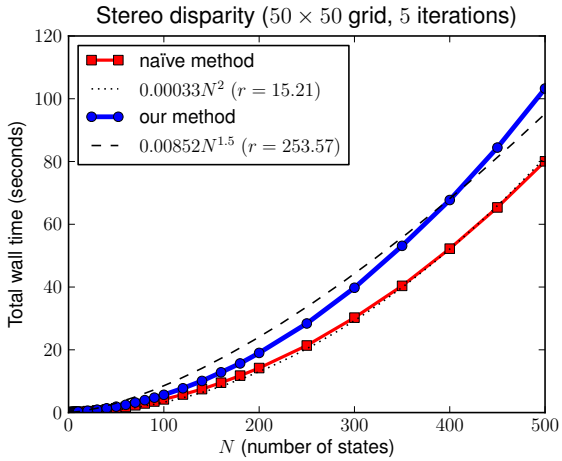
Our fitted curves in Figure 7 show $O(N^{1.5})$ performance for both random data and for optical flow.

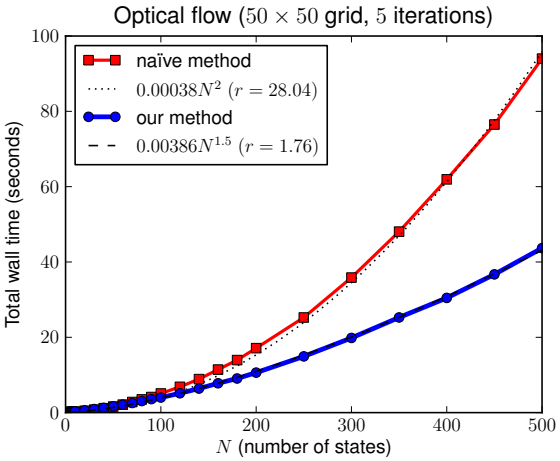
5.2.3. Failure Cases

In our previous experiments on text denoising and optical flow we observed running times similar to those for random potentials, indicating that there is no prevalent dependence structure between the order statistics of the messages and the potentials.

In certain applications the order statistics of these terms are highly dependent. The most straightforward example is that of *concave* potentials (or *convex* potentials in a min-sum formulation). For instance, in a







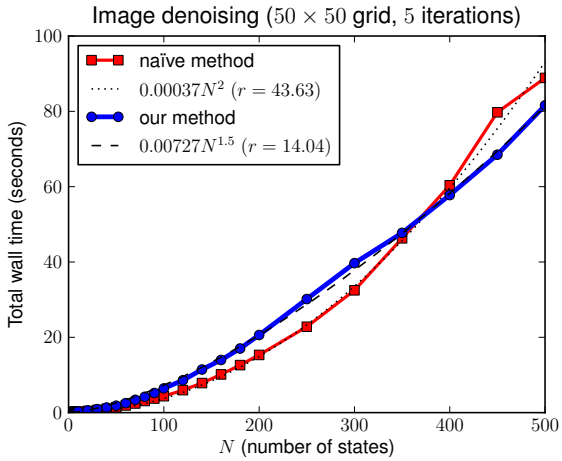


Figure 7. Running time of inference in grid-structured models: random potentials (top), and optical flow (bottom).

Figure 8. Two experiments whose potentials and messages have highly dependent order statistics: stereo disparity (top), and image denoising (bottom).

stereo disparity experiment, the unary potentials encode that the output should be 'close to' a certain value; the pairwise potentials encode that neighboring nodes should take similar values (Sun et al., 2003).

Whenever both $\mathbf{v}_a$ and $\mathbf{v}_b$ are concave in (eq. 7), the permutation matrix that transforms the sorted values of $\mathbf{v}_a$ to the sorted values of $\mathbf{v}_b$ is block-off-diagonal (see the sixth permutation in Figure 3). In such cases, our algorithm only decreases the number of addition operations by a multiplicative constant, and may in fact be slower due to its computational overhead. This is precisely the behavior shown in Figure 8 (top), in the case of stereo disparity.

It should be noted that there exist algorithms specifically designed for this class of potential functions (Kolmogorov & Shioura, 2007; Felzenszwalb & Huttenlocher, 2006), which are preferable in such instances.

We similarly perform an experiment on image denoising, where the unary potentials are again convex functions of the input (see Lan et al., 2006). Instead of using a pairwise potential that merely encodes smoothness, we extract the pairwise statistics from image data (similar to our experiment on text denoising); thus the potentials are no longer concave. We see in Figure 8 (bottom) that even if a small number of entries exhibit some 'randomness' in their order statistics, we begin to gain a modest speed improvement over the naïve solution (though indeed, the improvements are negligible compared to those shown in previous experiments).

6. Discussion and Future Work

At the core of our work is an $O(\sqrt{N})$ solution to (eq. 7); this solution has many applications beyond those covered in this paper. As suggested in Section 3, our analysis leads to an $O(N^{2.5})$ expected-time solution to 'funny matrix multiplication' – the analogue

of regular matrix multiplication where summation is replaced by maximization.

It can be shown that a sub-cubic solution to funny matrix multiplication has a variety of applications beyond those discussed here. For instance, it allows us to solve the all-pairs shortest path problem in $O(N^{2.5})$ (Aho et al., 1983).

We have also applied similar techniques to a different class of graphical models, by exploiting the fact that the *data-dependent* factors in triangulated graphical models often contain fewer terms than their maximal cliques. In such cases, exact inference in a junction-tree is equivalent to a generalized version of funny matrix multiplication. This leads to faster solutions to a number of computer-vision problems in which large maximal cliques factor into pairwise terms (McAuley & Caetano, 2010).

7. Conclusion

We have presented an algorithm for message passing in models whose data-dependent factors contain fewer latent variables than their data-independent factors. We find this to be useful in models with pairwise priors, as it allows us to do message passing in only $O(N^{1.5})$ for models with N states, thus substantially improving upon the standard quadratic-time solution. In practice, we find that in spite of the computational overhead of our model, speed improvements are gained even for modest values of N, resulting in substantial speedups in a variety of real-world applications.

Acknowledgements

We would like to thank Pedro Felzenszwalb, Johnicholas Hines, and David Sontag for comments on initial versions of this paper. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program

References

Aho, Alfred V., Hopcroft, John E., and Ullman, Jeffrey D. *Data Structures and Algorithms*. Addison-Wesley, 1983.

Aji, Srinivas M. and McEliece, Robert J. The generalized distributive law. *IEEE Trans. on Information Theory*, 46(2):325–343, 2000.

Alon, Noga, Galil, Zvi, and Margalit, Oded. On the exponent of the all pairs shortest path problem. *Journal of Computer and System Sciences*, 54(2):255–262, 1997.

Felzenszwalb, Pedro F. Representation and detection of deformable shapes. *IEEE Trans. on PAMI*, 27(2): 208–220, 2005.

Felzenszwalb, Pedro F. and Huttenlocher, Daniel P. Efficient belief propagation for early vision. *IJCV*, 70(1):41–54, 2006.

Kerr, Leslie R. The effect of algebraic structure on the computational complexity of matrix multiplication. *PhD Thesis*, 1970.

Kersting, Kristian, Ahmadi, Babak, and Natarajan, Sriraam. Counting belief propagation. In *UAI*, 2009.

Kolmogorov, Vladimir and Shioura, Akiyoshi. New algorithms for the dual of the convex cost network flow problem with application to computer vision. Technical report, University College London, 2007.

Kschischang, Frank R., Frey, Brendan J., and Loeliger, Hans-Andrea. Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory*, 47 (2):498–519, 2001.

Kumar, M. Pawan and Torr, Philip. Fast memory-efficient generalized belief propagation. In *ECCV*, 2006.

Lan, Xiang-Yang, Roth, Stefan, Huttenlocher, Daniel P., and Black, Michael J. Efficient belief propagation with learned higher-order markov random fields. In *ECCV*, 2006.

Lucas, Bruce D. and Kanade, Takeo. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

McAuley, Julian J. and Caetano, Tibério S. Faster Algorithms for Max-Product Message-Passing *CoRR*, abs/0910.3301, 2009.

McAuley, Julian J. and Caetano, Tibério S. Exploiting within-clique factorizations in junction-tree algorithms. *AISTATS*, 2010.

Petersen, K., Fehr, J., and Burkhardt, H. Fast generalized belief propagation for MAP estimation on 2D and 3D grid-like markov random fields. In *DAGM*, 2008.

Quasthoff, U., Richter, M., and Biemann, C. Corpus portal for search in monolingual corpora. In *Language Resources and Evaluation*, 2006.

Sun, Jian, Zheng, Nan-Ning, and Shum, Heung-Yeung. Stereo matching using belief propagation. *IEEE Trans. on PAMI*, 25(7):787–800, 2003.