Polynomial Linear Programming with Gaussian Belief Propagation

Danny Bickson, Yoav Tock
IBM Haifa Research Lab
Mount Carmel
Haifa 31905, Israel
Email: {dannybi,tock}@il.ibm.com

Ori Shental
Center for Magnetic Recording Research
UCSD, San Diego
9500 Gilman Drive
La Jolla, CA 92093, USA
Email: oshental@ucsd.edu

Danny Dolev
School of Computer Science and Engineering
Hebrew University of Jerusalem
Jerusalem 91904, Israel
Email: dolev@cs.huji.ac.il

***Abstract*—Interior-point methods are state-of-the-art algorithms for solving linear programming (LP) problems with polynomial complexity. Specifically, the Karmarkar algorithm typically solves LP problems in time $O(n^{3.5})$, where n is the number of unknown variables. Karmarkar's celebrated algorithm is known to be an instance of the log-barrier method using the Newton iteration. The main computational overhead of this method is in inverting the Hessian matrix of the Newton iteration. In this contribution, we propose the application of the Gaussian belief propagation (GaBP) algorithm as part of an efficient and distributed LP solver that exploits the sparse and symmetric structure of the Hessian matrix and avoids the need for direct matrix inversion. This approach shifts the computation from realm of linear algebra to that of probabilistic inference on graphical models, thus applying GaBP as an efficient inference engine. Our construction is general and can be used for any interior-point algorithm which uses the Newton method, including non-linear program solvers.**

I. Introduction

In recent years, considerable attention has been dedicated to the relation between belief propagation message passing and linear programming schemes. This relation is natural since the maximum a-posteriori (MAP) inference problem can be translated into integer linear programming (ILP) [1].

Weiss *et al.* [1] approximate the solution to the ILP problem by relaxing it to a LP problem using convex variational methods. In [2], tree-reweighted belief propagation (BP) is used to find the global minimum of a convex approximation to the free energy. Both of these works apply discrete forms of BP. Globerson *et al.* [3], [4] assume convexity of the problem and modify the BP update rules using dual-coordinate ascent algorithm. Hazan *et al.* [5] describe an algorithm for solving a general convex free energy minimization. In both cases the algorithm is guaranteed to converge to the global minimum as the problem is tailored to be convex.

In the present work we take a different path. Unlike most of the previous work which uses gradient-descent methods, we show how to use interior-point methods which are shown to have strong advantages over gradient and steepest descent methods. (For a comparative study see [6, §9.5,p. 496].) The main benefit of using interior point methods is their rapid convergence, which is quadratic once we are close enough to the optimal solution. Their main drawback is that they require heavier computational effort for forming and inverting the Hessian matrix, needed for computing the Newton step. To overcome this, we propose the use of Gaussian BP (GaBP) [7], [8], which is a variant of BP applicable when the underlying distribution is Gaussian. Using GaBP, we are able to reduce the time associated with the Hessian inversion task, from $O(n^{2.5})$ to $O(np\log(\epsilon)/\log(\gamma))$ at the worst case, where $p < n$ is the size of the constraint matrix $\mathbf{A}$, ϵ is the desired accuracy, and $1/2 < \gamma < 1$ is a parameter characterizing the matrix $\mathbf{A}$. This computational saving is accomplished by exploiting the sparsity of the Hessian matrix.

An additional benefit of our GaBP-based approach is that the polynomial-complexity LP solver

can be implemented in a distributed manner, enabling efficient solution of large-scale problems.

We also provide what we believe is the first theoretical analysis of the convergence speed of the GaBP algorithm.

The paper is organized as follows. In Section II, we reduce standard linear programming to a least-squares problem. Section III shows how to solve the least-squares problem using the GaBP algorithm. In Section IV, we extend our construction to the primal-dual method. We give our convergence results for the GaBP algorithm in Section V, and demonstrate our construction in Section VI using an elementary example. We present our conclusions in Section VII.

II. Standard Linear Programming

Consider the standard linear program

$$\begin{aligned} &\text{minimize}_{\mathbf{x}} && \mathbf{c}^T\mathbf{x} && \text{(1a)}\\ &\text{subject to} && \mathbf{A}\mathbf{x}=\mathbf{b}, \quad \mathbf{x}\geq 0 && \text{(1b)}\end{aligned}$$

where $\mathbf{A}\in\mathbb{R}^{n\times p}$ with $\mathbf{rank}\{\mathbf{A}\}=p<n$. We assume the problem is solvable with an optimal $\mathbf{x}^*$ assignment. We also assume that the problem is strictly feasible, or in other words there exists $\mathbf{x}\in\mathbb{R}^n$ that satisfies $\mathbf{A}\mathbf{x}=\mathbf{b}$ and $\mathbf{x}>0$.

Using the log-barrier method [6, §11.2], one gets

$$\begin{aligned} &\text{minimize}_{\mathbf{x},\mu} && \mathbf{c}^T\mathbf{x}-\mu\Sigma_{k=1}^{n}\log x_k && \text{(2a)}\\ &\text{subject to} && \mathbf{A}\mathbf{x}=\mathbf{b}. && \text{(2b)}\end{aligned}$$

This is an approximation to the original problem (1a). The quality of the approximation improves as the parameter $\mu\to 0$.

Now we would like to use the Newton method in for solving the log-barrier constrained objective function (2a), described in Table I. Suppose that we have an initial feasible point $\mathbf{x}_0$ for the canonical linear program (1a). We approximate the objective function (2a) around the current point $\tilde{\mathbf{x}}$ using a second-order Taylor expansion

$$f(\tilde{\mathbf{x}}+\Delta\mathbf{x})\simeq f(\tilde{\mathbf{x}})+f'(\tilde{\mathbf{x}})\Delta\mathbf{x}+1/2\Delta\mathbf{x}^T f''(\tilde{\mathbf{x}})\Delta\mathbf{x}. \quad (3)$$

Finding the optimal search direction $\Delta\mathbf{x}$ yields the computation of the gradient and compare it to zero

$$\frac{\partial f}{\partial\Delta\mathbf{x}}=f'(\tilde{\mathbf{x}})+f''(\tilde{\mathbf{x}})\Delta\mathbf{x}=0, \quad (4)$$

$$\Delta\mathbf{x}=-f''(\tilde{\mathbf{x}})^{-1}f'(\tilde{\mathbf{x}}). \quad (5)$$

Denoting the current point $\tilde{\mathbf{x}}\triangleq(\mathbf{x},\mu,\mathbf{y})$ and the Newton step $\Delta\mathbf{x}\triangleq(\mathbf{x},\mathbf{y},\mu)$, we compute the gradient

$$f'(\mathbf{x},\mu,\mathbf{y})\equiv(\partial f(\mathbf{x},\mu,\mathbf{y})/\partial\mathbf{x},\partial f(\mathbf{x},\mu,\mathbf{y})/\partial\mu, ,\partial f(\mathbf{x},\mu,\mathbf{y})/\partial\mathbf{y})$$

The Lagrangian is

$$\mathcal{L}(\mathbf{x},\mu,\mathbf{y})=\mathbf{c}^T\mathbf{x}-\mu\Sigma_k\log x_k+\mathbf{y}^T(\mathbf{b}-\mathbf{A}\mathbf{x}), \quad (7)$$

$$\frac{\partial\mathcal{L}(\mathbf{x},\mu,\mathbf{y})}{\partial\mathbf{x}}=\mathbf{c}-\mu\mathbf{X}^{-1}\mathbf{1}-\mathbf{y}^T\mathbf{A}=0, \quad (8)$$

$$\frac{\partial^2\mathcal{L}(\mathbf{x},\mu,\mathbf{y})}{\partial\mathbf{x}}=\mu\mathbf{X}^{-2}, \quad (9)$$

where $\mathbf{X}\triangleq\text{diag}(\mathbf{x})$ and $\mathbf{1}$ is the all-one column vector. Substituting (8)-(9) into (4), we get

$$\mathbf{c}-\mu\mathbf{X}^{-1}\mathbf{1}-\mathbf{y}^T\mathbf{A}+\mu\mathbf{X}^{-2}\mathbf{x}=0, \quad (10)$$

$$\mathbf{c}-\mu\mathbf{X}^{-1}\mathbf{1}+\mathbf{x}\mu\mathbf{X}^{-2}=\mathbf{y}^T\mathbf{A}, \quad (11)$$

$$\frac{\partial\mathcal{L}(\mathbf{x},\mu,\mathbf{y})}{\partial\mathbf{y}}=\mathbf{A}\mathbf{x}=0. \quad (12)$$

Now multiplying (11) by $\mathbf{A}\mathbf{X}^2$, and using (12) to eliminate $\mathbf{x}$ we get

$$\mathbf{A}\mathbf{X}^2\mathbf{A}^T\mathbf{y}=\mathbf{A}\mathbf{X}^2\mathbf{c}-\mu\mathbf{A}\mathbf{X}\mathbf{1}. \quad (13)$$

These normal equations can be recognized as generated from the linear least-squares problem

$$\min_{\mathbf{y}}||\mathbf{X}\mathbf{A}^T\mathbf{y}-\mathbf{X}\mathbf{c}-\mu\mathbf{A}\mathbf{X}\mathbf{1}||_2^2. \quad (14)$$

Solving for $\mathbf{y}$ we can compute the Newton direction $\mathbf{x}$, taking a step towards the boundary and compose one iteration of the Newton algorithm. Next, we will explain how to shift the deterministic LP problem to the probabilistic domain and solve it distributively using GaBP.

TABLE I
THE NEWTON ALGORITHM [6, §9.5.2] .

Given		feasible starting point $\mathbf{x}_0$ and tolerance $\epsilon > 0$, $k = 1$
Repeat	1	Compute the Newton step and decrement $\Delta\mathbf{x} = f''(\mathbf{x})^{-1} f'(\mathbf{x}), \quad \lambda^2 = f'(\mathbf{x})^T \Delta\mathbf{x}$
	2	Stopping criterion. quit if $\lambda^2/2 \leq \epsilon$
	3	Line search. Choose step size t by backtracking line search.
	4	Update. $\mathbf{x}_k := \mathbf{x}_{k-1} + t\Delta\mathbf{x}, \quad k = k + 1$

III. FROM LP TO PROBABILISTIC INFERENCE

We start from the least-squares problem (14), changing notations to

$$\min_{\mathbf{y}} ||\mathbf{F}\mathbf{y} - \mathbf{g}||_2^2, \tag{15}$$

where $\mathbf{F} \triangleq \mathbf{X}\mathbf{A}^T, \mathbf{g} \triangleq \mathbf{X}\mathbf{c} + \mu\mathbf{A}\mathbf{X}\mathbf{1}$. Now we define a multivariate Gaussian

$$p(\hat{\mathbf{x}}) \triangleq p(\mathbf{x}, \mathbf{y}) \propto \exp(-1/2(\mathbf{F}\mathbf{y} - \mathbf{g})^T \mathbf{I}(\mathbf{F}\mathbf{y} - \mathbf{g})). \tag{16}$$

It is clear that $\hat{\mathbf{y}}$, the minimizing solution of (15), is the MAP estimator of the conditional probability

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) =$$

$$= \mathcal{N}((\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{g}, (\mathbf{F}^T\mathbf{F})^{-1}). \tag{17}$$

Recent results by Bickson and Shental *et al.* [7]–[9] show that the pseudoinverse problem (17) can be computed efficiently and distributively by using the GaBP algorithm.

The formulation (16) allows us to shift the least-squares problem from an algebraic to a probabilistic domain. Instead of solving a deterministic vector-matrix linear equation, we now solve an inference problem in a graphical model describing a certain Gaussian distribution function. Following [9] we define the joint covariance matrix

$$\mathbf{C} \triangleq \begin{pmatrix} -\mathbf{I} & \mathbf{F} \\ \mathbf{F}^T & \mathbf{0} \end{pmatrix} \tag{18}$$

and the shift vector $\mathbf{b} \triangleq \{\mathbf{0}^T, \mathbf{g}^T\}^T \in \mathbb{R}^{(p+n)\times 1}$.

Given the covariance matrix $\mathbf{C}$ and the shift vector $\mathbf{b}$, one can write explicitly the Gaussian density function, $p(\hat{\mathbf{x}})$, and its corresponding graph $\mathcal{G}$ with edge potentials ('compatibility functions') ψ_{ij} and self-potentials ('evidence') ϕ_i. These graph potentials are determined according to the following pairwise factorization of the Gaussian distribution $p(\mathbf{x}) \propto \prod_{i=1}^n \phi_i(x_i) \prod_{\{i,j\}} \psi_{ij}(x_i, x_j)$, resulting in $\psi_{ij}(x_i, x_j) \triangleq \exp(-x_i C_{ij} x_j)$, and $\phi_i(x_i) \triangleq \exp\left(b_i x_i - C_{ii} x_i^2/2\right)$. The set of edges $\{i, j\}$ corresponds to the set of non-zero entries in $\mathbf{C}$ (18). Hence, we would like to calculate the marginal densities, which must also be Gaussian,

$$p(x_i) \sim \mathcal{N}(\mu_i = \{\mathbf{C}^{-1}\mathbf{g}\}_i, P_i^{-1} = \{\mathbf{C}^{-1}\}_{ii}),$$

$$\forall i > p,$$

where μ_i and P_i are the marginal mean and inverse variance (a.k.a. precision), respectively. Recall that, according to [9], the inferred mean μ_i is identical to the desired solution $\hat{y}$ of (17). The GaBP update rules are summarized in Table II.

It is known that if GaBP converges, it results in exact inference [10]. However, in contrast to conventional iterative methods for the solution of systems of linear equations, for GaBP, determining the exact region of convergence and convergence rate remain open research problems. All that is known is a sufficient (but not necessary) condition [11], [12] stating that GaBP converges when the spectral radius satisfies $\rho(|\mathbf{I}_K - \mathbf{A}|) < 1$. A stricter sufficient condition [10], determines that the matrix $\mathbf{A}$ must be diagonally dominant (*i.e.*, $|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \forall i$) in order for GaBP to converge. Convergence speed is discussed in Section V.

IV. EXTENDING THE CONSTRUCTION TO THE PRIMAL-DUAL METHOD

In the previous section we have shown how to compute one iteration of the Newton method using GaBP. In this section we extend the technique for

TABLE II
COMPUTING $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ VIA GABP [7].

#	Stage	Operation
1.	*Initialize*	Compute $P_{ii} = A_{ii}$ and $\mu_{ii} = b_i/A_{ii}$. Set $P_{ki} = 0$ and $\mu_{ki} = 0$, $\forall k \neq i$.
2.	*Iterate*	Propagate P_{ki} and μ_{ki}, $\forall k \neq i$ such that $A_{ki} \neq 0$. Compute $P_{i\backslash j} = P_{ii} + \sum_{k\in\mathrm{N}(i)\backslash j} P_{ki}$ and $\mu_{i\backslash j} = P_{i\backslash j}^{-1}(P_{ii}\mu_{ii} + \sum_{k\in\mathrm{N}(i)\backslash j} P_{ki}\mu_{ki})$. Compute $P_{ij} = -A_{ij}P_{i\backslash j}^{-1}A_{ji}$ and $\mu_{ij} = -P_{ij}^{-1}A_{ij}\mu_{i\backslash j}$.
3.	*Check*	If P_{ij} and μ_{ij} did not converge, return to #2. Else, continue to #4.
4.	*Infer*	$P_i = P_{ii} + \sum_{k\in\mathrm{N}(i)} P_{ki}$, $\mu_i = P_i^{-1}(P_{ii}\mu_{ii} + \sum_{k\in\mathrm{N}(i)} P_{ki}\mu_{ki})$.
5.	*Output*	$x_i = \mu_i$

computing the primal-dual method. This construction is attractive, since the extended technique has the same computation overhead.

The dual problem ([13]) conforming to (1a) can be computed using the Lagrangian

$$\mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{z}) = \mathbf{c}^T\mathbf{x} + \mathbf{y}^T(\mathbf{b} - \mathbf{A}\mathbf{x}) - \mathbf{z}^T\mathbf{x}, \quad \mathbf{z} \geq 0,$$

$$g(\mathbf{y},\mathbf{z}) = \inf_x \mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{z}), \tag{19a}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0. \tag{19b}$$

while

$$\frac{\partial\mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{z})}{\partial\mathbf{x}} = \mathbf{c} - \mathbf{A}^T\mathbf{y} - \mathbf{z} = 0. \tag{20}$$

Substituting (20) into (19a) we get

$$\begin{aligned} \text{maximize}_{\mathbf{y}} \quad & \mathbf{b}^T\mathbf{y} \\ \text{subject to} \quad & \mathbf{A}^T\mathbf{y} + \mathbf{z} = \mathbf{c}, \quad \mathbf{z} \geq 0. \end{aligned}$$

Primal optimality is obtained using (8) [13]

$$\mathbf{y}^T\mathbf{A} = \mathbf{c} - \mu\mathbf{X}^{-1}\mathbf{1}. \tag{22}$$

Substituting (22) in (21a) we get the connection between the primal and dual

$$\mu\mathbf{X}^{-1}\mathbf{1} = \mathbf{z}.$$

In total, we have a primal-dual system (again we assume that the solution is strictly feasible, namely $\mathbf{x} > 0, \mathbf{z} > 0$)

$$\begin{aligned} \mathbf{A}\mathbf{x} = \mathbf{b}, \quad & \mathbf{x} > 0, \\ \mathbf{A}^T\mathbf{y} + \mathbf{z} = \mathbf{c}, \quad & \mathbf{z} > 0, \\ \mathbf{X}\mathbf{z} = \mu\mathbf{1}. & \end{aligned}$$

The solution $[\mathbf{x}(\mu), \mathbf{y}(\mu), \mathbf{z}(\mu)]$ of these equations constitutes the central path of solutions to the logarithmic barrier method [6, 11.2.2]. Applying the Newton method to this system of equations we get

$$\begin{pmatrix} 0 & \mathbf{A}^T & I \\ \mathbf{A} & 0 & 0 \\ \mathbf{Z} & 0 & \mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{b} - \mathbf{A}\mathbf{x} \\ \mathbf{c} - \mathbf{A}^T\mathbf{y} - \mathbf{z} \\ \mu\mathbf{1} - \mathbf{X}\mathbf{z} \end{pmatrix}. \tag{24}$$

The solution can be computed explicitly by

$$\begin{aligned} \Delta\mathbf{y} = \; & (\mathbf{A}\mathbf{Z}^{-1}\mathbf{X}\mathbf{A}^T)^{-1}\cdot \\ & (\mathbf{A}\mathbf{Z}^{-1}\mathbf{X}(\mathbf{c} - \mu\mathbf{X}^{-1}\mathbf{1} - \mathbf{A}^T\mathbf{y}) + \mathbf{b} - \mathbf{A}\mathbf{x}), \\ \Delta\mathbf{x} = \; & \mathbf{X}\mathbf{Z}^{-1}(\mathbf{A}^T\Delta\mathbf{y} + \mu\mathbf{X}^{-1}\mathbf{1} = \mathbf{c} + \mathbf{A}^T\mathbf{y}), \\ \Delta\mathbf{z} = \; & -\mathbf{A}^T\Delta\mathbf{y} + \mathbf{c} - \mathbf{A}^T\mathbf{y} - \mathbf{z}. \end{aligned}$$

The main computational overhead in this method is the computation of $(\mathbf{A}\mathbf{Z}^{-1}\mathbf{X}\mathbf{A}^T)^{-1}$, which is derived from the Newton step in (5).

Now we would like to use GaBP for computing the solution. We make the following simple change to (24) to make it symmetric: since $\mathbf{z} > 0$, we can multiply the third row by $\mathbf{Z}^{-1}$ and get a modified symmetric system

$$\begin{pmatrix} 0 & \mathbf{A}^T & I \\ \mathbf{A} & 0 & 0 \\ I & 0 & \mathbf{Z}^{-1}\mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{b} - \mathbf{A}\mathbf{x} \\ \mathbf{c} - \mathbf{A}^T\mathbf{y} - \mathbf{z} \\ \mu\mathbf{Z}^{-1}\mathbf{1} - \mathbf{X} \end{pmatrix}.$$

Defining $\tilde{\mathbf{A}} \triangleq \begin{pmatrix} 0 & \mathbf{A}^T & I \\ \mathbf{A} & 0 & 0 \\ I & 0 & \mathbf{Z}^{-1}\mathbf{X} \end{pmatrix}$, and $\tilde{\mathbf{b}} \triangleq \begin{pmatrix} \mathbf{b} - \mathbf{A}\mathbf{x} \\ \mathbf{c} - \mathbf{A}^T\mathbf{y} - \mathbf{z} \\ \mu\mathbf{Z}^{-1}\mathbf{1} - \mathbf{X} \end{pmatrix}$. one can use GaBP iterative algorithm shown in Table II.

In general, by looking at (4) we see that the solution of each Newton step involves inverting the Hessian matrix $f''(\mathbf{x})$. The state-of-the-art approach in practical implementations of the Newton step is first computing the Hessian inverse $f''(\mathbf{x})^{-1}$ by using a (sparse) decomposition method like (sparse) Cholesky decomposition, and then multiplying the result by $f'(\mathbf{x})$. In our approach, the GaBP algorithm computes directly the result $\Delta\mathbf{x}$, without computing the full matrix inverse. Furthermore, if the GaBP algorithm converges, the computation of $\Delta\mathbf{x}$ is guaranteed to be accurate.

V. New Convergence Results

In this section we give an upper bound on the convergence rate of the GaBP algorithm. As far as we know this is the first theoretical result bounding the convergence speed of the GaBP algorithm.

Our upper bound is based on the work of Weiss *et al.* [10, Claim 4], which proves the correctness of the mean computation. Weiss uses the pairwise potentials form[1], where

$$\begin{aligned} p(\mathbf{x}) &\propto \Pi_{i,j}\psi_{ij}(x_i,x_j)\Pi_i\psi_i(x_i), \\ \psi_{i,j}(x_i,x_j) &\equiv \exp(-1/2(x_i\ x_j)^T\mathbf{V}_{ij}(x_i\ x_j)), \\ \mathbf{V}_{ij} &\equiv \begin{pmatrix} \tilde{a}_{ij} & \tilde{b}_{ij} \\ \tilde{b}_{ji} & \tilde{c}_{ij} \end{pmatrix}. \end{aligned}$$

Assuming the optimal solution is $\mathbf{x}^*$, for a desired accuracy $\epsilon||\mathbf{b}||_\infty$ where $||\mathbf{b}||_\infty \equiv \max_i |\mathbf{b}_i|$, and $\mathbf{b}$ is the shift vector, we need to run the algorithm for at most $t = \lceil \log(\epsilon)/\log(\beta) \rceil$ rounds to get an accuracy of $|x^* - x_t| < \epsilon||\mathbf{b}||_\infty$ where $\beta = \max_{ij} |\tilde{b}_{ij}/\tilde{c}_{ij}|$.

The problem with applying Weiss' result directly to our model is that we are working with different parameterizations. We use the *information form* $p(\mathbf{x}) \propto \exp(-1/2\mathbf{x}^T\mathbf{A}\mathbf{x}+\mathbf{b}^T\mathbf{x})$. The decomposition of the matrix $\mathbf{A}$ into pairwise potentials is not unique. In order to use Weiss' result, we propose such a decomposition. Any decomposition from the canonical form to the pairwise potentials form should be subject to the following constraints [10]

$$\tilde{b}_{ij} = \mathbf{A}_{ij}, \qquad \Sigma_j \tilde{c}_{ij} = \mathbf{A}_{ii}.$$

We propose to initialize the pairwise potentials as following. Assuming the matrix $\mathbf{A}$ is diagonally dominant, we define ε_i to be the non negative gap

$$\varepsilon_i \triangleq |\mathbf{A}_{ii}| - \Sigma_j |\mathbf{A}_{ij}| > 0.$$

and the following decomposition

$$\tilde{b}_{ij} = \mathbf{A}_{ij}, \qquad \tilde{c}_{ij} = \mathbf{A}_{ij} + \varepsilon_i/|N(i)|,$$

where $|N(i)|$ is the number of graph neighbors of node i. Following Weiss, we define γ to be

$$\begin{aligned} \gamma = \max_{i,j} \frac{|\tilde{b}_{ij}|}{|\tilde{c}_{ij}|} = \frac{|a_{ij}|}{|a_{ij}| + \varepsilon_i/|N(i)|} = \\ = \max_{i,j} \frac{1}{1 + (\varepsilon_i)/(|a_{ij}||N(i)|)} < 1. \end{aligned} \tag{25}$$

In total, we get that for a desired accuracy of $\epsilon||\mathbf{b}||_\infty$ we need to iterate for $t = \lceil \log(\epsilon)/\log(\gamma) \rceil$ rounds. Note that this is an upper bound and in practice we indeed have observed a much faster convergence rate.

The computation of the parameter γ can be easily done in a distributed manner: Each node locally computes ε_i, and $\gamma_i = \max_j 1/(1 + |a_{ij}|\varepsilon_i/N(i))$. Finally, one maximum operation is performed globally, $\gamma = \max_i \gamma_i$.

A. Applications to Interior-Point Methods

We would like to compare the running time of our proposed method to the Newton interior-point method, utilizing our new convergence results of the previous section. As a reference we take the Karmarkar algorithm [14] which is known to be an instance of the Newton method [15]. Its running time is composed of n rounds, where on each round one Newton step is computed. The cost of computing one Newton step on a dense Hessian matrix is $O(n^{2.5})$, so the total running time is $O(n^{3.5})$.

Using our approach, the total number of Newton iterations, n, remains the same as in the Karmarkar algorithm. However, we exploit the special structure of the Hessian matrix, which is both symmetric and sparse. Assuming that the size of the constraint matrix $\mathbf{A}$ is $n \times p, \quad p < n$, each iteration of GaBP for computing a single Newton step takes $O(np)$, and based on our new convergence analysis for accuracy $\epsilon||\mathbf{b}||_\infty$ we need to iterate for $r = \lceil \log(\epsilon)/\log(\gamma) \rceil$ rounds, where γ is defined in (25). The total computational burden for a single Newton step is $O(np\log(\epsilon)/\log(\gamma))$. There are at most n rounds, hence in total we get $O(n^2 p\log(\epsilon)/\log(\gamma))$.

[1] Weiss assumes scalar variables with zero means.

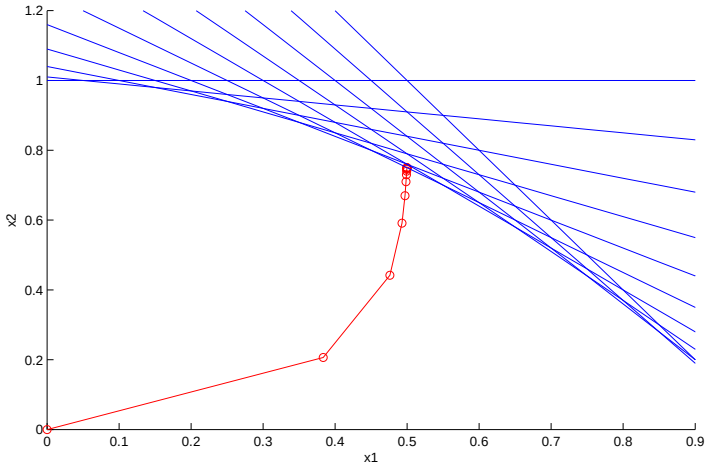

Fig. 1. A simple example of using GaBP for solving linear programming with two variables and eleven constraints. Each red circle shows one iteration of the Newton method.

VI. Experimental Results

We demonstrate the applicability of the proposed algorithm using the following simple linear program borrowed from [16]

$$\begin{aligned} \text{maximize} \quad & x_1 + x_2 \\ \text{subject to} \quad & 2px_1 + x_2 \leq p^2 + 1\,, \\ & p = 0.0, 0.1, \cdots, 1.0\ . \end{aligned}$$

Fig. 1 shows execution of the affine-scaling algorithm [17], a variant of Karmarkar's algorithm [14], on a small problem with two variables and eleven constraints. Each circle is one Newton step. The inverted Hessian is computed using the GaBP algorithm, using two computing nodes. Matlab code for this example can be downloaded from [18].

Regarding larger scale problems, we have observed rapid convergence (of a single Newton step computation) on very large scale problems. For example, [19] demonstrates convergence of 5-10 rounds on sparse constraint matrices with several millions of variables. [20] shows convergence of dense constraint matrices of size up to $150,000 \times 150,000$ in 6 rounds, where the algorithm is run in parallel using 1,024 CPUs. Empirical comparison with other iterative algorithms is given in [8].

VII. Conclusion

In this paper we have shown how to efficiently and distributively solve interior-point methods using an iterative algorithm, the Gaussian belief propagation algorithm. Unlike previous approaches which use discrete belief propagation and gradient descent methods, we take a different path by using continuous belief propagation applied to interior-point methods. By shifting the Hessian matrix inverse computation required by the Newton method, from linear algebra domain to the probabilistic domain, we gain a significant speedup in performance of the Newton method. We believe there are numerous applications that can benefit from our new approach.

Acknowledgement

O. Shental acknowledges the partial support of the NSF (Grant CCF-0514859). D. Bickson would like to thank Nati Linial from the Hebrew University of Jerusalem for proposing this research direction. The authors are grateful to Jack Wolf and Paul Siegel from UCSD for useful discussions and for constructive comments on the manuscript.

References

[1] C. Yanover, T. Meltzer, and Y. Weiss, "Linear programming relaxations and belief propagation – an empirical study," in *Journal of Machine Learning Research*, vol. 7. Cambridge, MA, USA: MIT Press, 2006, pp. 1887–1907.

[2] Y. Weiss, C. Yanover, and T. Meltzer, "Map estimation, linear programming and belief propagation with convex free energies," in *The 23th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

[3] A. Globerson and T. Jaakkola, "Fixing max-product: Convergent message passing algorithms for map lp-relaxations," in *Advances in Neural Information Processing Systems (NIPS)*, no. 21, Vancouver, Canada, 2007.

[4] M. Collins, A. Globerson, T. Koo, X. Carreras, and P. Bartlett, "Exponentiated gradient algorithms for conditional random fields and max-margin markov networks," in *Journal of Machine Learning Research. Accepted for publication*, 2008.

[5] T. Hazan and A. Shashua, "Convergent message-passing algorithms for inference over general graphs with convex free energy," in *The 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, Helsinki, July 2008.

[6] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, March 2004.

[7] O. Shental, D. Bickson, P. H. Siegel, J. K. Wolf, and D. Dolev, "Gaussian belief propagation solver for systems of linear equations," in *IEEE Int. Symp. on Inform. Theory (ISIT)*, Toronto, Canada, July 2008.

[8] D. Bickson, O. Shental, P. H. Siegel, J. K. Wolf, and D. Dolev, "Linear detection via belief propagation," in *Proc. 45th Allerton Conf. on Communications, Control and Computing*, Monticello, IL, USA, Sept. 2007.

[9] ——, "Gaussian belief propagation based multiuser detection," in *IEEE Int. Symp. on Inform. Theory (ISIT)*, Toronto, Canada, July 2008.

[10] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, no. 10, pp. 2173–2200, 2001.

[11] J. K. Johnson, D. M. Malioutov, and A. S. Willsky, "Walk-sum interpretation and analysis of Gaussian belief propagation," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006, pp. 579–586.

[12] D. M. Malioutov, J. K. Johnson, and A. S. Willsky, "Walk-sums and belief propagation in Gaussian graphical models," *Journal of Machine Learning Research*, vol. 7, Oct. 2006.

[13] S. Portnoy and R. Koenker, "The gaussian hare and the laplacian tortoise: Computability of squared- error versus absolute-error estimators," in *Statistical Science*, vol. 12, no. 4. Institute of Mathematical Statistics, 1997, pp. 279–296.

[14] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1984, pp. 302–311.

[15] D. A. Bayer and J. C. Lagarias, "Karmarkar's linear programming algorithm and newton's method," in *Mathematical Programming*, vol. 50, no. 1, March 1991, pp. 291–330.

[16] `http://en.wikipedia.org/wiki/Karmarkar's_algorithm`.

[17] R. J. Vanderbei, M. S. Meketon, and B. A. Freedman, "A modification of karmarkar's linear programming algorithm," in *Algorithmica*, vol. 1, no. 1, March 1986, pp. 395–407.

[18] `http://www.cs.huji.ac.il/labs/danss/p2p/gabp/`.

[19] D. Bickson and D. Malkhi, "A unifying framework for rating users and data items in peer-to-peer and social networks," in *Peer-to-Peer Networking and Applications (PPNA) Journal, Springer-Verlag*, April 2008.

[20] D. Bickson, D. Dolev, and E. Yom-Tov, "A gaussian belief propagation solver for large scale support vector machines," in *5th European Conference on Complex Systems*, Jerusalem, Sept. 2008.