LINEAR COORDINATE-DESCENT MESSAGE-PASSING FOR QUADRATIC OPTIMIZATION

Guoqiang Zhang and Richard Heusdens

Department of Mediamatics
Delft University of Technology
Delft, the Netherlands
Email: {g.zhang-1,r.heusdens}@tudelft.nl

ABSTRACT

In this paper we propose a new message-passing algorithm for quadratic optimization. The design of the new algorithm is based on linear coordinate-descent between neighboring nodes. The updating messages are in a form of linear functions as compared to the min-sum algorithm of which the messages are in a form of quadratic functions. Therefore, the linear coordinate-descent (LiCD) algorithm has simpler updating rules than the min-sum algorithm. It is shown that when the quadratic matrix is walk-summable, the LiCD algorithm converges. As an application, the LiCD algorithm is utilized in solving general linear systems. The performance of the LiCD algorithm is found empirically to be comparable to that of the min-sum algorithm, but at lower complexity in terms of computation and storage.

Index Terms— Distributed optimization, message passing, walk summable, coordinate decent

1. INTRODUCTION

In this paper we consider solving a quadratic optimization problem, namely,

$$\min_{x\in R^n} f(x) = \min_{x\in R^n} \big(\frac{1}{2}x^\top Jx - h^\top x\big), \tag{1}$$

where $J \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix and $h \in \mathbb{R}^n$. Without loss of generality, we assume J has unit diagonal. It is known that the optimal solution x^* satisfies the linear equation $Jx^* = h$. We suppose that the matrix J is sparse and the dimensionality n is large. In this situation, direct computation (without using the sparse structure of J) of the optimal solution may be expensive and unscalable. One natural question is how to exploit the sparse geometry to efficiently obtain the optimal solution. To achieve this goal, the quadratic function $f(x)$ can be decomposed in a pairwise fashion according to an undirected graph $G = (V, E)$, so that

$$f(x) = \sum_{i\in V} f_i(x_i) + \sum_{(i,j)\in E} f_{ij}(x_i, x_j).$$

The algorithms that exploit the sparse geometry exchange information between nodes in the graph until reaching consensus.

As a popular algorithm, the min-sum message-passing for solving the quadratic optimization problem has been well studied. If the graph has a tree structure (i.e., no loops involved), it is well known that the min-sum algorithm converges to the optimal solution [1, 2]. The question of convergence to the optimal solution has proved difficult for loopy graph models. In [2, 3], it has been shown that if the min-sum algorithm converges for general graphs, it computes the optimal solution. In particular, Weiss and Freeman [2] established a convergence condition where the quadratic matrix J is required to be diagonally dominant[1]. Later on, Johnson et al. [4, 5] discovered a more general convergence condition. They found that if the matrix J is walk-summable[2], the min-sum algorithm always converges. Recent work by Moallemi and Roy provided a geometrical meaning to the walk-summability of J [6]. In [7], the min-sum algorithm for quadratic optimization was applied in solving general linear systems.

We point out that the updating messages of the min-sum algorithm for solving the quadratic problem are in a form of quadratic functions. This implies that a node has to transmit two coefficients (one corresponds to the quadratic term and the other corresponds to the linear term) for each message to its neighbors. One natural question is if an efficient algorithm exits where the messages are in a form of linear functions only. Linear-functional messages have significant importance for sensor-network related problems, where the transmission power and storage capacity are highly valuable.

In this paper we design a new algorithm with linear-functional messages for the quadratic problem. The messages are updated by performing linear coordinate-descent (LiCD) between neighboring nodes in the graph. The design of the LiCD algorithm is inspired by the block coordinate-descent (BCD) algorithms developed for discrete graph models [8, 9, 10] (the variables take discrete values). While the message forms of the BCD algorithms are determined by the discrete alphabet, the LiCD algorithm explicitly imposes linear-functional messages. The BCD algorithms require that the messages are updated asynchronously. The messages of the LiCD algorithm, on the other hand, can be updated either in parallel or asynchronously. To save space in the paper, we only consider parallel message-updating for the LiCD algorithm.

We will show that when the matrix J is walk-summable, the LiCD algorithm converges to the optimal solution. We emphasize that the walk-summability of J is only a sufficient condition for the algorithm convergence. In addition, we consider applying the LiCD algorithm in solving general linear systems. We follow the line of work in [7], where the min-sum algorithm is exploited in solving the same problem. We found by experiment that the LiCD algorithm is comparable to the min-sum algorithm in convergence speed, but has a lower computational complexity and requires less storage capacity.

2. LINEAR COORDINATE-DESCENT ALGORITHM

In this section we first present the LiCD algorithm by deriving the updating expressions. After that, we study the convergence of the LiCD algorithm.

[1]The matrix J is diagonally dominant if $|J_{ii}| > \sum_{j\neq i} |J_{ij}|$ for all i.

[2]See subsection 2.4 for the definition of walk-summability. It can be shown by algebra that if a matrix is diagonal dominant, then it is also walk-summable.

2.1. Message passing framework

Consider the quadratic objective function (1). Correspondingly, the local functions are given by

$$f_{ij}(x_i, x_j) = J_{ij}x_i x_j \quad \forall (i,j) \in E \tag{2}$$

$$f_i(x_i) = \frac{1}{2}x_i^2 - h_i x_i \quad \forall i \in V. \tag{3}$$

The local functions f_i and f_{ij} are often called self-potentials and edge potentials, respectively. f_{ij} captures the interaction between node i and j. For the quadratic problem, an edge between node i and j exists in the graph only if $J_{ij} \neq 0$. We use $N(i)$ to denote the set of all neighbors of node i. For each edge $(i,j) \in E$, we use $[j,i]$ and $[i,j]$ to denote its two directed edges. Correspondingly, we use $\vec{E}$ to denote the set of all directed edges.

The LiCD algorithm exchanges information between nodes iteratively until reaching consensus. In particular, at time t, each node i collects incoming messages $\left\{m_{ui}^{(t)}(x_i) | u \in N(i)\right\}$ from all neighboring nodes. These messages are then combined to produce new outgoing messages, one for each neighbor $u \in N(i)$. We suppose that the message sent from node j to i at time t takes a linear form

$$m_{ji}^{(t)}(x_i) = -z_{ji}^{(t)} x_i. \tag{4}$$

The new self and edge potentials at time t can be defined as

$$f_{ij}^{(t)}(x_i, x_j) = J_{ij}x_i x_j + z_{ji}^{(t)} x_i + z_{ij}^{(t)} x_j \quad \forall (i,j) \in E \tag{5}$$

$$f_i^{(t)}(x_i) = \frac{1}{2}x_i^2 - \left(h_i + \sum_{u \in N(i)} z_{ui}^{(t)}\right) x_i \quad \forall i \in V. \tag{6}$$

It is straightforward that

$$f(x) = \sum_{i \in V} f_i^{(t)}(x_i) + \sum_{(i,j) \in E} f_{ij}^{(t)}(x_i, x_j). \tag{7}$$

Thus, the overall objective function remains the same. The new potentials (5)-(6) can be viewed as a reformulation of the objective function.

At time t, each node computes an estimate of its variable by minimizing the self-potential:

$$\hat{x}_i^{(t)} = \arg\min_{x_i} f_i^{(t)}(x_i) \quad i \in V. \tag{8}$$

If the LiCD algorithm converges to the optimal solution x^*, we have

$$\lim_{t \to \infty} \hat{x}_i^{(t)} = x_i^* \quad \forall i \in V.$$

2.2. Updating expressions

The LiCD algorithm performs pairwise minimizations between neighboring nodes in computing new messages. In this subsection, we compute the updating expressions for the messages.

Without loss of generality, we focus on deriving the expressions for $z_{ji}^{(t+1)}$ and $z_{ij}^{(t+1)}$, $(i,j) \in E$, given the potential functions at time t. Before going to the details, we first present the basic idea of the linear coordinate-descent. In computing $\{z_{ji}^{(t+1)}, z_{ij}^{(t+1)}\}$, node i and j build a joint function $L_{ij}^{(t)}(x_i, x_j)$ defined as

$$L_{ij}^{(t)}(x_i, x_j) = f_i^{(t)}(x_i) + f_j^{(t)}(x_j) + f_{ij}^{(t)}(x_i, x_j).$$

This particular form of the joint function is motivated from the fact that $L_{ij}^{(t)}(x_i, x_j)$ is a sub-summation of (7) for the objective function $f(x)$. As a result, $L_{ij}^{(t)}(x_i, x_j)$ does not involve $z_{ij}^{(t)}$ and $z_{ji}^{(t)}$ any more. We first minimize $L_{ij}^{(t)}(x_i, x_j)$ over $\{x_i, x_j\}$. Denote the resulting optimal solution as $\{\hat{x}_i^{j,(t+1)}, \hat{x}_j^{i,(t+1)}\}$, where the superscript j (or i) indicates that the estimate $\hat{x}_i^{j,(t+1)}$ (or $\hat{x}_j^{i,(t+1)}$) is computed by utilizing the information from node j (or node i). After obtaining $\hat{x}_i^{j,(t+1)}$, we then choose $z_{ji}^{(t+1)}$ such that

$$\hat{x}_i^{j,(t+1)} = \arg\min_{x_i} \left(\frac{1}{2}x_i^2 - \left(h_i + \sum_{u \in N(i) \setminus j} z_{ui}^{(t)} + z_{ji}^{(t+1)}\right) x_i\right).$$

The estimate $\hat{x}_i^{j,(t+1)}$ is different from $\hat{x}_i^{(t)}$ in (8). The parameter $z_{ji}^{(t+1)}$ brings all the information about $\hat{x}_i^{j,(t+1)}$ that is contained in node j to node i. Since we only operate on the linear terms of the potential functions, this is how the name *linear coordinate-descent* comes up.

Based on the above computation guidline, we derive the expressions for $z_{ij}^{(t+1)}$ and $z_{ji}^{(t+1)}$. From (5)-(6), the expression for $L_{ij}^{(t)}(x_i, x_j)$ is given by

$$L_{ij}^{(t)}(x_i, x_j) = \frac{1}{2}\begin{pmatrix} x_i & x_j \end{pmatrix} \begin{bmatrix} 1 & J_{ij} \\ J_{ij} & 1 \end{bmatrix} \begin{pmatrix} x_i \\ x_j \end{pmatrix} - \begin{pmatrix} h_i + \sum_{u \in N(i) \setminus j} z_{ui}^{(t)} & h_j + \sum_{v \in N(j) \setminus i} z_{vj}^{(t)} \end{pmatrix} \begin{pmatrix} x_i \\ x_j \end{pmatrix}.$$

As the 2×2 quadratic matrix of $L_{ij}^{(t)}(x_i, x_j)$ is positive definite, the optimal solution $(\hat{x}_i^{j,(t+1)}, \hat{x}_j^{i,(t+1)})$ is finite. By computation, the expressions for $z_{ji}^{(t+1)}$ and $z_{ij}^{(t+1)}$ are given by

$$z_{ji}^{(t+1)} = \frac{J_{ij}}{1 - J_{ij}^2}\left(J_{ij}\big(h_i + \sum_{u \in N(i) \setminus j} z_{ui}^{(t)}\big) - \big(h_j + \sum_{v \in N(j) \setminus i} z_{vj}^{(t)}\big)\right) \tag{9}$$

$$z_{ij}^{(t+1)} = \frac{J_{ij}}{1 - J_{ij}^2}\left(J_{ij}\big(h_j + \sum_{v \in N(j) \setminus i} z_{vj}^{(t)}\big) - \big(h_i + \sum_{u \in N(i) \setminus j} z_{ui}^{(t)}\big)\right) \tag{10}$$

The updating expressions for other parameters $\{z_{kl}^{(t+1)}, z_{lk}^{(t+1)}, (k,l) \in E\}$ take similar forms as in (9)-(10).

2.3. Algorithm implementation

In the implementation of the LiCD algorithm, we let $z_{ji}^{(0)} \in \mathbb{R}$ for all $[j,i] \in \vec{E}$. As the messages evolve according to (9)-(10) over time, each node keeps track of the most recent parameters from neighboring nodes, one from each neighbor. Compared to that of the min-sum algorithm, this operation is obviously memory-efficient. At each time, a node computes an estimate of its variable by applying (8):

$$\hat{x}_i^{(t)} = h_i + \sum_{u \in N(i)} z_{ui}^{(t)} \quad \forall i \in V, t = 0, 1, \ldots \tag{11}$$

If the algorithm converges, the parameters $\{z_{ij}\}$ and the estimates $\{\hat{x}_i\}$ would be stable after some time. Thus one can terminate the iteration by examining $\{z_{ij}\}$ and/or $\{\hat{x}_i\}$. Alternatively, one can define a termination criterion by involving $\hat{x}_i^{(t)}$ and x_i^*. In particular,

one can check the error $\epsilon^{(t)} = \|\hat{x}^{(t)} - x^*\|_\infty$ at the end of iteration $t, t = 1, 2, \ldots$. We briefly summarize the algorithm in Table 1.

	Stage	Operation
1	*Initialize*	$z_{ij} \in \mathbb{R}, \forall [i,j] \in \vec{E}$
2	*Iterate*	For all $[i,j] \in \vec{E}$ Update z_{ij} using (10). End
3	*Check*	If $\{\hat{x}_i^{(t)}\}$ or $\{\epsilon^{(t)}\}$ have converged, go to 4; else, return to 2.
4	*Output*	Return $\hat{x}_i, \forall i$.

Table 1. LiCD message-passing for computing $x^* = \arg\min_x \frac{1}{2}x^\top Jx - h^\top x$.

2.4. On convergence of the LiCD algorithm

The walk-summability of J was originally discovered as a sufficient condition for the convergence of the min-sum algorithm [4, 5, 6]. We show in the following that the walk-summability of J is also a sufficient condition for the LiCD algorithm to converge. For completeness, we provide the definition of the walk-summability below.

Definition 2.1 *[4, 5] A positive definite matrix $J \in \mathbb{R}^{n\times n}$, with all ones on its diagonal, is walk-summable if the spectral radius of the matrix $\bar{R}$, where $R = I - J$ and $\bar{R} = [|R_{ij}|]_{i,j=1}^n$, is less than one (i.e., $\rho(\bar{R}) < 1$).*

We briefly describe how the walk-summability of J is connected to the optimal solution of the quadratic problem (see [4, 5] for detailed information). First, note that the optimal solution for the quadratic optimization problem (1) is given by

$$x^* = J^{-1}h = (I - R)^{-1}h.$$

If we assume that the matrix $\bar{R}$ has spectral radius less than 1, then the spectral radius of R is also less than 1 (i.e., $\rho(R) < 1$). Under the condition $\rho(R) < 1$, we may express the solution x^* by the infinite series

$$x^* = \sum_{t=0}^{\infty} R^t h, \tag{12}$$

where $R_{ii} = 0$ and $R_{ij} = -J_{ij}$, if $i \neq j$. The condition $\rho(\bar{R}) < 1$ guarantees that the series $\sum_{t=0}^{\infty} R^t$ is absolutely convergent. In other words, the convergence of $\sum_{t=0}^{\infty} R^t$ does not depend on how its components are arranged in the summation. By establishing the equivalence between the iterates (9)-(11) and (12) when $\rho(\bar{R}) < 1$, we then obtain the convergence results. We present the results in two theorems below: one for the special initialization $\{z_{ij}^{(0)} = 0\}$ and the other for a general initialization.

Theorem 2.2 *If the quadratic matrix J in (1) is walk-summable (i.e., $\rho(\bar{R}) < 1$) and $z_{ij}^{(0)} = 0$ for all $[i,j] \in \vec{E}$, then the iterates of the LiCD algorithm satisfy*

$$\|\hat{x}^{(t)} - x^*\|_2 \leq \frac{\rho(\bar{R})^{t+1}}{1 - \rho(\bar{R})}\|\bar{h}\|_2, \tag{13}$$

where $\bar{h} = [|h_i|]$.

Theorem 2.3 *If the quadratic matrix J in (1) is walk-summable (i.e., $\rho(\bar{R}) < 1$) and $z_{ij}^{(0)} \in \mathbb{R}$ for all $[i,j] \in \vec{E}$, then the LiCD algorithm converges to the optimal solution x^*, i.e., $\hat{x}_i^{(t)} \to x_i^*$ for all $i \in V$.*

As the proofs for the two theorems are rather long, we will present them in [11].

3. APPLICATION TO GENERAL LINEAR SYSTEMS

It is known that solving a general linear system of equations (possibly over-constrained) is a fundamental problem in computer science and engineering. Recently, the min-sum algorithm has been successfully applied for the problem [12, 7]. As an alternative solution, we consider applying the LiCD algorithm in solving the same problem. In particular, we focus on the efficiency of the LiCD algorithm.

Suppose $A \in \mathbb{R}^{n\times k}$, $n \geq k$, is full column rank matrix. For a given shift vector $b \in \mathbb{R}^{n\times 1}$, we consider solving the following linear least square problem

$$x^* = \arg\min_x \|Ax - b\|_2^2,$$

where the vector x is of dimensionality k. By using algebra, it is straightforward that the optimal solution is given by

$$x^* = (A^\top A)^{-1}A^\top b. \tag{14}$$

We let $J = (A^\top A)$. The matrix J is positive definite, but not necessarily walk-summable. Thus, the convergence of the LiCD or the min-sum algorithm is not guaranteed here.

3.1. Scheme 1

In [7], the linear equation (14) was relaxed to a sequence of linear equations

$$\hat{x}^{(t+1)} = (J + \alpha I)^{-1}(A^\top b + \alpha\hat{x}^{(t)}) \quad t = 0, 1, \ldots, \tag{15}$$

where $\alpha > 0$ is a free parameter. We refer to the construction (15) as *Scheme 1*. Denote $J_\alpha = J + \alpha I$. It was shown in [7] that by following the iteration (15), $\hat{x}^{(t)}$ converges to the solution (14) for any initialization $\hat{x}^{(0)}$. Given $\hat{x}^{(t)}$ at time t, the new estimate $\hat{x}^{(t+1)}$ is computed by exploiting the min-sum algorithm. To guarantee the convergence of the min-sum algorithm, α is chosen to ensure that J_α is walk-summable. This approach involves two loops: the outer loop follows the iteration (15), and the inner loop implements the min-sum algorithm for each time step in (15).

Correspondingly, we use the same iteration (15) in our work. Then at time t in (15), we apply the LiCD algorithm to compute $\hat{x}^{(t+1)}$. In order to use (9)-(11), (15) is rescaled to make J_α unit diagonal. Since α is chosen such that J_α is walk-summable, the LiCD algorithm always converges.

3.2. Scheme 2

Note that Scheme 1 requires the pre-computation of J and $A^\top b$. To avoid such computation, another construction of linear equations was proposed in [12, 7]. The basic idea is to introduce an auxiliary vector $z \in \mathbb{R}^{n\times 1}$ in addition to x. With the new variable vector $y = [x^\top, z^\top]^\top$, the computation for x^* in (14) is realized by solving a new sequence of linear equations

$$\tilde{J}_\alpha \hat{y}^{(t+1)} = h^{(t)}, \quad t = 0, 1, \ldots, \tag{16}$$

where

$$\tilde{J}_\alpha = \begin{pmatrix} I_{k\times k} & A^\top \\ A & -\alpha I_{n\times n} \end{pmatrix} \in \mathbb{R}^{(k+n)\times(k+n)}$$

$$\hat{y}^{(t+1)} = [\ \hat{x}^{(t+1)\top} \quad \hat{z}^{(t+1)}\ ]^\top$$

$$h^{(t)} = [\ \hat{x}^{(t)\top} \quad b^\top\ ]^\top.$$

The two iterations (15) and (16) are equivalent in computing $\hat{x}^{(t+1)}$. At time t, the min-sum algorithm is used to compute $\hat{y}^{(t+1)}$ given

$h^{(t)}$ in [7]. We refer to the construction (16) as *Scheme 2*.

In the following, we consider applying the LiCD algorithm to compute $\hat{y}^{(t+1)}$ given $h^{(t)}$ in (16). Since J_α is not positive definite, Theorem 2.2 or 2.3 cannot be directly applied here. We briefly argue that for large enough α, the LiCD algorithm still converges by following (9)-(11). To apply (9)-(10), (16) is rescaled such that the rescaled matrix $\breve{J}_\alpha$ has unit diagonal. In particular, $\breve{J}_\alpha$ takes the form

$$\breve{J}_\alpha = \begin{pmatrix} I_{k\times k} & \frac{1}{\sqrt{\alpha}}A^\top i \\ \frac{1}{\sqrt{\alpha}}Ai & I_{n\times n} \end{pmatrix} \in \mathbb{C}^{(k+n)\times(k+n)},$$

where i represents the imaginary symbol. Define $\breve{R}_\alpha \triangleq I - \breve{J}_\alpha$ and $\bar{R}_\alpha \triangleq [|\breve{R}_{\alpha,ij}|]$. It is clear that for large enough α, there is $\rho(\bar{R}_\alpha) < 1$. By following the same argument as was used in Theorem 2.2 and 2.3, it can be shown that the LiCD algorithm converges by following (9)-(11).

3.3. Experimental results

It is clear that the min-sum and the LiCD algorithm have the same applicability. In this subsection we study the efficiency of the LiCD algorithm with the min-sum algorithm as a reference.

In the experiment, we let $n = 20$ and $k = 5$. The components in A and b were generated from the uniform distribution $\mathcal{U}(0, 1)$. To terminate the iterations of the two algorithms, the error of the estimate w.r.t. its optimal solution was measured. Convergence threshold for the outer and inner loops were set as 10^{-5} and 10^{-10}, respectively.

In the first experiment, we tested the two algorithms for five different linear systems (different realizations of A and b). For each linear system, the parameter α was chosen such that J_α is walk-summable. The results are displayed in Table. 2. It is seen that the performance of the two algorithms is comparable. Other linear systems (different parameters A and b) were also tested, of which the performance is similar to Table 2.

		1	2	3	4	5
S1	LiCD	6866	4125	5697	6335	7216
	min-sum	7073	4200	5783	6503	7182
S2	LiCD	14131	8470	11318	12273	14886
	min-sum	13153	7772	10437	11179	13395

Table 2. Number of iterations of the two algorithms under the two schemes. In particular, S1 stands for Scheme 1, and S2 stands for Scheme 2.

In the second experiment, we study the impact of the parameter α on the performance of the two algorithms. In doing so, we focus on a particular linear system (A, b). The results are displayed in Fig. 1. One observes that Scheme 2 always need more iterations than Scheme 1. Also, there exists a particular value for α where the two algorithms are most efficient for each scheme.

4. CONCLUSION

We have proposed a new message-passing algorithm for quadratic optimization by performing linear coordinate-decent operations. Compared to the min-sum algorithm of which the messages are quadratic functions, the LiCD algorithm has linear-functional messages. Thus, the LiCD algorithm saves half of the transmission bandwidth used for the min-sum algorithm for each iteration. Further, we have shown that the LiCD has the same convergence condition as the min-sum algorithm. Finally we successfully applied

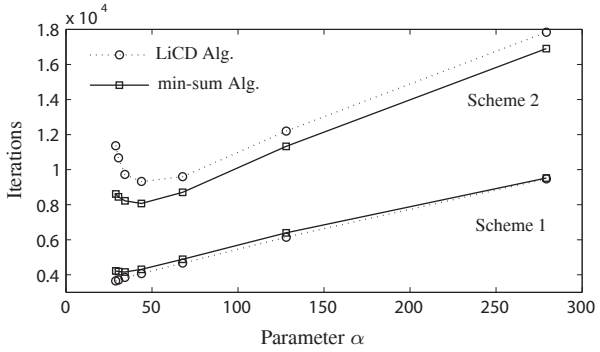

Fig. 1. Impact of the parameter α on the performance of the two algorithms.

the LiCD algorithm in solving general linear systems, of which the performance is comparable to that of the min-sum algorithm.

5. REFERENCES

[1] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference," *Morgan Kaufman Publishers*, 1988.

[2] Y. Weiss and W. T. Freeman, "Correctness of Belief Propagation in Gaussian Graphical Models of Arbitrary Topology," *Neural Computation*, vol. 13, pp. 2173–2200, 2001.

[3] P. Rusmevichientong and B. Ban Roy, "An analysis of Belief Propagation on the Turbo Decoding Graph with Gaussian Densities," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 745–765, 2001.

[4] 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*, Cambridge, MA: MIT Press, 2006, vol. 18.

[5] D. M. Malioutov, J. K. Johnson, and A. S. Willsky, "Walk-Sums and Belief Propagation in Gaussian Graphical Models," *J. Mach. Learn. Res.*, vol. 7, pp. 2031–2064, 2006.

[6] C. C. Moallemi and B. Van Roy, "Convergence of Min-Sum Message Passing for Quadratic Optimization," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2413–2423, 2009.

[7] J. K. Johnson, D. Bickson, and D. Dolev, "Fixing Convergence of Gaussian Belief Propagation," in *the International Symposium on Information Theory*, 2009.

[8] A. Globerson and T. Jaakkola, "Fixing Max-Product: Convergent Message Passing Algorithms for MAP LP-Relaxations," *Advances in Neural Information Processing Systems 21*.

[9] D. Sontag, T. Meltzer, A. Globerson, Y. Weiss, and T. Jaakkola, "Tightening LP Relaxations for MAP using Message-Passing," in *UAI*, 2008.

[10] D. Sontag, A. Globerson, and T. Jaakkola, "Introduction to Dual Decomposition for Inference," in *Optimization for Machine Learning*. 2011, MIT Press.

[11] G. Zhang and R. Heusdens, "Linear Coordinate-Descent Message-Passing for Quadratic Optimization," in preparation for submission to Neural Computation.

[12] D. Bickson, D. Dolev, O. Shental, P. H. Siegel, and J. K. Wolf, "Gaussian Belief Propagation Based Multiuser Detection," in *IEEE International Symposium on Information Theory*, 2008, pp. 1878–1882.