Convergence of Min-Sum-Min Message-Passing for Quadratic Optimization

Guoqiang Zhang* and Richard Heusdens

Department of Circuits and Systems,
Delft University of Technology,
Delft, The Netherlands
{g.zhang-1,r.heusdens}@tudelft.nl

Abstract. We propose a new message-passing algorithm for the quadratic optimization problem. As opposed to the min-sum algorithm, the new algorithm involves two minimizations and one summation at each iteration. The new min-sum-min algorithm exploits feedback from last iteration in generating new messages, resembling the Jacobi-relaxation algorithm. We show that if the feedback signal is large enough, the min-sum-min algorithm is guaranteed to converge to the optimal solution. Experimental results show that the min-sum-min algorithm outperforms two reference methods w.r.t. the convergence speed.

Keywords: quadratic optimization, Gaussian belief propagation, min-sum, min-sum-min.

1 Introduction

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

$$\min_{x\in\mathbb{R}^n} f(x) \triangleq \min_{x\in\mathbb{R}^n} \left(\frac{1}{2}x^\top Jx - h^\top x\right), \tag{1}$$

where the quadratic matrix $J \in \mathbb{R}^{n\times n}$ is real symmetric positive definite and $h \in \mathbb{R}^n$. It is known that the optimal solution is given by $x^* = J^{-1}h$. We suppose that the quadratic matrix J is sparse and the dimensionality n is large. In this situation, the 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.

A common approach that exploits the sparsity of J is to associate the function $f(x)$ with an undirected graph $G = (V, E)$. That is, the graph has a node for each variable x_i and an edge between node i and j only if the element J_{ij} is nonzero. By doing so, the sparsity of J is fully captured by the graph. As a consequence, the function can be decomposed with respect to $G = (V, E)$ as

$$f(x) = \sum_{i\in V} f_i(x_i) + \sum_{(i,j)\in E} f_{ij}(x_i, x_j), \tag{2}$$

* This work was supported by the COMMIT program, The Netherlands.

T. Calders et al. (Eds.): ECML PKDD 2014, Part III, LNCS 8726, pp. 353–368, 2014.

where each edge-function $f_{ij}(x_i, x_j)$ characterizes the interaction of x_i and x_j as specified by J_{ij}. With the graphic model (2), distributed quadratic optimization (DQO) boils down to how to spread the global information of (J, h) in (1) over the graph efficiently by exchanging local information between neighboring nodes.

In the literature, the Jacobi algorithm is a natural approach for solving the problem over the associated graph [1]. At each iteration, the algorithm performs node-oriented minimizations over all the nodes in the graph, of which the messages are in a form of linear functions (see Table 1). It is known that when the matrix J is walk-summable[1], the Jacobi algorithm converges to the optimal solution [3,5]. To fix the convergence for a general matrix J, the Jacobi algorithm was under-relaxed by incorporating an estimate of x^* from last iteration in computing a new estimate (see Table 1). The Jacobi-relaxation algorithm possesses a guaranteed convergence if the relaxation parameter is properly chosen [1]. For the above two algorithms, once a node-estimate is updated, this estimate is broadcast to all its neighbors. Because the information transmitted is general, and not edge-specific, the two algorithms are known to converge slowly [1].

Table 1. Algorithm comparison. The min-sum-min algorithm is a new method that we will present in the paper.

J is walk-summable	J is general
Jacobi Alg.: * node-oriented minimization * linear message	Jacobi-relaxation Alg.: * introduce feedback in Jacobi Alg.
LiCD Alg.: * pairwise minimization * linear message	GLiCD Alg.: * introduce feedback in LiCD Alg.
min-sum Alg.: * pairwise minimization * quadratic message	min-sum-min Alg.: * introduce feedback in min-sum Alg.

To accelerate the convergence of the Jacobi algorithm, the linear coordinate descent (LiCD) algorithm was proposed in [9]. At each iteration, the LiCD algorithm performs pairwise minimizations over all the edges in the graph, of which the messages are in a form of linear functions (see Table 1). As shown in [9], if the quadratic matrix J is walk-summable, the LiCD algorithm converges to the optimal solution. To fix the convergence for a general matrix J, the LiCD algorithm was further extended in [10] by incorporating feedback from last iteration in computing new messages, which is referred to as the *generalized LiCD* (GLiCD) algorithm.

An alternative scheme for solving the quadratic problem is by using the framework of probability theory. The optimal solution x^* is viewed as the mean value of a random vector $x \in \mathbb{R}^n$ with Gaussian distribution

[1] See subsection 2.3 for the definition.

$$p(x) \propto \exp\left(-\frac{1}{2}x^\top Jx + h^\top x\right). \tag{3}$$

The min-sum (also known as max-product) algorithm is one popular approach to estimate both the mean value $x^* = J^{-1}h$ and individual variances [8,2]. At each iteration, the algorithm essentially performs pairwise minimizations over all the edges in the graph, of which the messages are in a form of quadratic functions (see Table 1). For a graph with a tree-structure, the min-sum algorithm converges to the optimal solution in finite steps [8]. The question of convergence for loopy graphic models has been proven difficult. In [3,5,6], it was shown when the matrix J is walk-summable, the min-sum algorithm converges to the optimal solution (see Table 1). In [4], a double-loop algorithm has been proposed to compute the optimal solution for a general matrix J, where the min-sum algorithm is used as a subroutine. We note that the double-loop algorithm is time-consuming. This motivates us to develop a single-loop min-sum based algorithm.

In this paper, we complete Table 1 by proposing a (single-loop) min-sum-min algorithm for a general quadratic optimization problem. Our primary motivation is to fix the convergence failure of the min-sum algorithm when the matrix J is general. Inspired by the GLiCD algorithm, the min-sum-min algorithm also incorporates feedback from last iteration in computing new messages. Compared to the min-sum algorithm, the min-sum-min algorithm involves one more minimization at each iteration. The additional minimization is performed to compute the estimate of x^*, which is used to construct the feedback signal in generating new messages.

We also study the convergence of the min-sum-min algorithm. We show that by setting the feedback signal large enough in computing new messages, the algorithm possesses a guaranteed convergence. Experimental results show that the min-sum-min algorithm converges faster than the Jacobi-relaxation and GLiCD algorithms.

2 Min-Sum-Min Message-Passing

In this section, we present the min-sum-min algorithm for the quadratic optimization problem. In particular, we describe how to construct feedback signal in update the messages.

2.1 Message-Passing Framework

Consider the quadratic optimization problem (1). Without loss of generality, we assume the quadratic matrix J is of unit diagonal (i.e., $J_{ii} = 1$, $i = 1, \ldots, n$). By using the sparsity of the matrix J, the quadratic function $f(x)$ can be decomposed w.r.t. a graph $G = (V, E)$

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

where the node and edge functions are given by

$$f_i(x_i) = \frac{1}{2}x_i^2 - h_i x_i \qquad i \in V \tag{4}$$

$$f_{ij}(x_i, x_j) = J_{ij} x_i x_j \qquad (i,j) \in E. \tag{5}$$

An edge exists between node i and j in the graph only if $J_{ij} \neq 0$. For each node $i \in V$, we denote the set of its neighbors as $N(i) \triangleq \{j \in V : (i,j) \in E\}$. For each edge $(i,j) \in E$, we use $[j,i]$ to denote the directed edge from node i to j. Correspondingly, we denote the set of all directed edges of the graph as $\overrightarrow{E}$.

The min-sum-min algorithm intends to minimize the quadratic function in an iterative, synchronous message-passing fashion. At time t, each node i keeps track of a message and an estimate of x_i^* from each neighbor $u \in N(i)$. We denote the message and the estimate from node u to i as $m_{ui}^{(t)}(x_i)$ and $\hat{x}_i^{u,(t)}$, respectively. Correspondingly, we use $\hat{x}_{edge}^{(t)}$ to denote the vector of all the estimates at time t. $\hat{x}_{edge}^{(t)}$ is of dimension $|\overrightarrow{E}|$, of which each component $\hat{x}_i^{j,(t+1)}$ corresponds to a directed edge $[i,j] \in \overrightarrow{E}$. Note that for each node $i \in V$, the estimates $\{\hat{x}_i^{u,(t)}, u \in N(i)\}$ reveal information about the optimal solution x_i^*. Thus, the estimates obtained at time t can be used as feedback in computing new messages and new estimates at time $t+1$.

Formally, we use the estimates at time t to construct $|E|$ penalty functions, one for each edge in the graph. In particular, we define the penalty function $p_{ij}^{(t)}(x_i, x_j)$ for $(i,j) \in E$ to be a quadratic function:

$$p_{ij}^{(t)}(x_i, x_j) = \frac{s}{2}\left(x_i - \hat{x}_i^{j,(t)}\right)^2 + \frac{s}{2}\left(x_j - \hat{x}_j^{i,(t)}\right)^2, \tag{6}$$

where the weighting factor $1 > s \geq 0$. Note that each penalty function only involves the estimates that are computed along the associated edge. The particular form of the penalty function enables the performance analysis of the algorithm (see Section 3.2).

With the penalty functions (6), we define new node and edge functions at time t as

$$g_i^{(t)}(x_i) = (1-s) f_i(x_i) + \sum_{u \in N(i)} m_{ui}^{(t)}(x_i) \quad i \in V \tag{7}$$

$$\begin{aligned} g_{ij}^{(t)}(x_i, x_j) =& (1-s) f_{ij}(x_i, x_j) - m_{ji}^{(t)}(x_i) - m_{ij}^{(t)}(x_j) \\ &+ p_{ij}^{(t)}(x_i, x_j) \quad (i,j) \in E. \end{aligned} \tag{8}$$

As opposed to (4)-(5), the new edge and node functions (7)-(8) include both the current messages and the penalty functions.

In next subsection, we explain how to use (7)-(8) in computing new messages and estimates. Note that as the weighting factor s approaches to one, the original function $f(x)$ has less and less impact on the new local functions (7)-(8). While at the same time, the penalty function enlarges the impact of the estimates when computing new estimates and messages in next iteration.

Remark 1. We point out that when $s = 0$, the local-function formation (7)-(8) coincides with that of the min-sum algorithm [6]. It is the penalty functions that make the node and edge functions special.

2.2 Message-Updating Expressions

We have thus far presented the message-passing framework. In particular, we have defined the penalty functions (6). In this subsection, we derive the updating expressions for the messages and estimates. We then point out the difference between the min-sum-min and min-sum algorithms.

Suppose that the messages at time t take a quadratic form: (see [6] for a similar definition)

$$m_{ui}^{(t)}(x_i) = -\frac{1}{2}\gamma_{ui}^{(t)}(1-s)^2 J_{ui}^2 x_i^2 + z_{ui}^{(t)} x_i, \quad \forall\ [u,i] \in \overrightarrow{E}, \tag{9}$$

where $\{\gamma_{ui}\}$ and $\{z_{ui}\}$ are quadratic parameters and linear parameters, respectively. The weighting factor s is involved in (9) because of the penalty functions. We use $\gamma^{(t)}$ to denote the vector of all the quadratic parameters at time t. Similarly, we use $z^{(t)}$ to denote the vector of all the linear parameters. Both $\gamma^{(t)}$ and $z^{(t)}$ are of dimension $|\overrightarrow{E}|$.

We now compute the new estimates and messages for time $t+1$ given the information at time t. Without loss of generality, we focus on computing $\{m_{ij}^{(t+1)}, m_{ji}^{(t+1)}\}$ and $\{\hat{x}_i^{j,(t+1)}, \hat{x}_j^{i,(t+1)}\}$ that are associated with the edge $(i,j) \in E$. Note that the old messages $\{m_{ij}^{(t)}, m_{ji}^{(t)}\}$ and estimates $\{\hat{x}_i^{j,(t)}, \hat{x}_j^{i,(t)}\}$ are only involved in three local functions $\{g_i^{(t)}(x_i), g_j^{(t)}(x_j), g_{ij}^{(t)}(x_i, x_j)\}$. Thus, we use the three local functions in computing the corresponding new messages and estimates.

Formally, we define a function $L_{ij}^{(t)}(x_i, x_j)$ for $(i,j) \in E$ to be

$$L_{ij}^{(t)}(x_i, x_j) \stackrel{\Delta}{=} g_i^{(t)}(x_i) + g_j^{(t)}(x_j) + g_{ij}^{(t)}(x_i, x_j). \tag{10}$$

The function $L_{ij}^{(t)}(x_i, x_j)$ is in a quadratic form. For the time being, we assume that $L_{ij}^{(t)}(x_i, x_j)$ is a strictly convex quadratic function. In other words, the 2×2 quadratic matrix in $L_{ij}^{(t)}(x_i, x_j)$ is assumed to be symmetric positive definite. In next subsection, we explain under what conditions the assumption holds. We compute the new estimates $\{\hat{x}_i^{j,(t+1)}, \hat{x}_j^{i,(t+1)}\}$ by minimizing the function $L_{ij}^{(t)}(\cdot,\cdot)$ over x_i and x_j:

$$\left(\hat{x}_i^{j,(t+1)}, \hat{x}_j^{i,(t+1)}\right) = \arg\min_{x_i, x_j} L_{ij}^{(t)}(x_i, x_j). \tag{11}$$

Since $L_{ij}^{(t)}(x_i, x_j)$ is a quadratic function, $\hat{x}_i^{j,(t+1)}$ and $\hat{x}_j^{i,(t+1)}$ have closed-form expressions.

Note that the information about $\hat{x}_i^{j,(t+1)}$ or $\hat{x}_j^{i,(t+1)}$ is embedded in both node i and j. We design the message $m_{ji}^{(t+1)}(x_i)$ with the purpose to bring all the information about $\hat{x}_i^{j,(t+1)}$ that is contained in node j to node i. In doing so, we reconsider the minimization of $L_{ij}^{(t)}(x_i, x_j)$:

$$\begin{aligned}\min_{x_i,x_j} L_{ij}^{(t)}(x_i,x_j) &= \min_{x_i}\left[g_i^{(t)}(x_i) + \min_{x_j}\left(g_j^{(t)}(x_j) + g_{ij}^{(t)}(x_i,x_j)\right)\right] \\ &= \min_{x_i}\Bigg[g_i^{(t)}(x_i) + \frac{s}{2}(x_i - \hat{x}_i^{j,(t)})^2 - m_{ji}^{(t)}(x_i) \\ &\qquad + \min_{x_j}\Big((1-s)f_j(x_j) + (1-s)f_{ij}(x_i,x_j) \\ &\qquad + \sum_{v\in N(j)\setminus i} m_{vj}^{(t)}(x_j) + \frac{s}{2}\big(x_j - \hat{x}_j^{i,(t)}\big)^2\Big)\Bigg]. \end{aligned} \tag{12}$$

By following (12), we define $m_{ji}^{(t+1)}(x_i)$ to be

$$\begin{aligned} m_{ji}^{(t+1)}(x_i) \triangleq \min_{x_j}\Bigg((1-s)f_j(x_j) + (1-s)f_{ij}(x_i,x_j) \\ + \sum_{v\in N(j)\setminus i} m_{vj}^{(t)}(x_j) + \frac{s}{2}\big(x_j - \hat{x}_j^{i,(t)}\big)^2\Bigg) + \kappa, \end{aligned} \tag{13}$$

where κ represents an arbitrary offset term. The derivation of $m_{ij}^{(t+1)}(x_j)$ follows a similar procedure.

Based on the above computation guideline, we present the final expressions for the new messages and estimates. By combining (4)-(5), (9) and (13), we obtain the expressions for $\gamma_{ji}^{(t+1)}$ and $z_{ji}^{(t+1)}$ of $m_{ji}^{(t+1)}(x_i)$ as

$$\gamma_{ji}^{(t+1)} = \frac{1}{1 - \sum_{v\in N(j)\setminus i}\gamma_{vj}^{(t)}(1-s)^2 J_{vj}^2}, \tag{14}$$

$$z_{ji}^{(t+1)} = (1-s)J_{ij}\gamma_{ji}^{(t+1)}\left((1-s)h_j + s\hat{x}_j^{i,(t)} - \sum_{v\in N(i)\setminus i} z_{vj}^{(t)}\right). \tag{15}$$

The parameters $\gamma_{ij}^{(t+1)}$ and $z_{ij}^{(t+1)}$ of $m_{ij}^{(t+1)}(x_i)$ can be computed similarly. By combining (11)-(15), we obtain the expressions for $\hat{x}_i^{j,(t+1)}$ and $\hat{x}_j^{i,(t+1)}$ as

$$\begin{pmatrix}\hat{x}_i^{j,(t+1)} \\ \hat{x}_j^{i,(t+1)}\end{pmatrix} = \frac{1}{(1-s)J_{ij}}\begin{pmatrix}1 & (1-s)J_{ij}\gamma_{ij}^{(t+1)} \\ (1-s)J_{ij}\gamma_{ji}^{(t+1)} & 1\end{pmatrix}^{-1}\begin{pmatrix}z_{ij}^{(t+1)} \\ z_{ji}^{(t+1)}\end{pmatrix}. \tag{16}$$

The above expression fully characterizes the relationship between the estimates and the linear parameters. With (14)-(16) at hand, one can easily work out the updating-expressions of the messages and estimates associated with other edges in the graph.

Finally we reconsider the expression (12). Note that there are two minimizations and one summation involved in (12). As indicated in (13), the minimization over x_j and the summation of the incoming messages excluding $m_{ij}^{(t)}$ originate from the min-sum algorithm. The second minimization over x_i in (12) computes an estimate of x_i^*, which is used as feedback in generating new messages and estimates in next iteration. This is how the name *min-sum-min* message-passing comes up.

Remark 2. It is worth noting that when $s = 0$ in (14)-(15), we actually obtain the message-updating expressions for the min-sum algorithm. In other words, the min-sum-min algorithm includes the min-sum algorithm as a special case by setting $s = 0$.

2.3 Algorithm Implementation

In this subsection, we consider the algorithm implementation. We mainly study under what conditions the minimization problem (11) is well defined for $t \geq 0$ and for any $(i, j) \in E$.

Before formally presenting the algorithm implementation, we first provide the definition of the walk-summability of a positive definite matrix below. We emphasize that the min-sum algorithm converges to the optimal solution if the matrix J in (1) is walk-summable [3,5,6].

Definition 1. *[3,5] A symmetric 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., $\rho(\bar{R}) < 1$).*

To facilitate the analysis, we set the initial estimates and messages to be zero, i.e., $\hat{x}_{edge}^{(0)} = 0$, $\gamma^{(0)} = 0$ and $z^{(0)} = 0$. We note that $\hat{x}_{edge}^{(0)}$ and $z^{(0)}$ have to satisfy Equation (16). In order for the algorithm to evolve continuously by following (14)-(16), the minimization problem (11) should be correctly posed for any $t \geq 0$. By working on (11), a sufficient condition can be derived:

$$1 > (1 - s)^2 J_{ij}^2 \gamma_{ij}^{(t)} \gamma_{ji}^{(t)} \quad \forall (i, j) \in E, \tag{17}$$

$$\gamma^{(t)} > 0, \tag{18}$$

where $t = 1, 2, \ldots$. Note that the above two equations only involve the quadratic vector $\gamma^{(t)}$ and J.

Next we argue that if the parameter s is chosen such that the matrix

$$J_s = sI + (1 - s)J \tag{19}$$

is walk-summable, (17)-(18) hold for any $t \geq 1$. Note that J_s is again of unit-diagonal. It is not difficult to show that when $s \in (\lfloor 1 - 1/\rho(\bar{R})\rfloor_+, 1)$, J_s is walk-summable. The operation $\lfloor w\rfloor_+ = \max(0, w)$ for $w \in \mathbb{R}$. From [6], it is known that if J_s is walk-summable, then $\gamma^{(t)}$ converges to a fixed point γ_s^* by following (14). Further,

$$\gamma_s^* \geq \gamma^{(t+1)} \geq \gamma^{(t)} \quad \forall t \geq 0. \tag{20}$$

Considering the fixed point γ_s^* in (17), we have

$$\begin{aligned} & 1 > (1-s)^2 J_{ij}^2 \gamma_{\{s,ij\}}^* \gamma_{\{s,ji\}}^* \quad \forall (i,j) \in E \\ \Longleftrightarrow & 1 \overset{(a)}{>} \sum_{u \in N(i)} (1-s)^2 J_{ui}^2 \gamma_{\{s,ui\}}^* \quad \forall (i,j) \in E, \end{aligned} \tag{21}$$

where step (a) follows from (14) and the fact that γ_s^* is stable. (21) holds when J_s is walk-summable [6,7]. By using (20)-(21) and the initialization $\gamma^{(0)} = 0$, it can be easily shown that (17)-(18) hold when J_s is walk-summable. We summarize the result in a lemma below:

Lemma 1. *if s is chosen from $(\lfloor 1 - 1/\rho(\bar{R}) \rfloor_+, 1)$ such that J_s is walk-summable and $\gamma^{(0)} = 0$, then the minimization problem (11) is well defined for any $(i,j) \in E$, $t \geq 0$. The quadratic vector $\gamma^{(t)}$ monotonically converges to γ_s^*.*

Besides the quadratic vector $\gamma^{(0)}$, we also have to initialize $\hat{x}_{edge}^{(0)}$ and $z^{(0)}$. Due to the expression (16), we only need to initialize $\hat{x}_{edge}^{(0)}$, the linear vector $z^{(0)}$ can be computed accordingly. If the algorithm converges to the optimal solution as $t \to \infty$, we have

$$\hat{x}_i^{u,(\infty)} = x_i^* \quad \forall u \in N(i) \text{ and } i \in V.$$

For the estimation vector $\hat{x}_{edge}^{(t)}$, $t \geq 0$, we denote its corresponding optimal solution as x_{edge}^*. In practice, one can measure the difference of the estimates $\{\hat{x}_i^{u,(t)}, u \in N(i)\}$ for each variable x_i to terminate the iteration procedure.

To briefly summarize,the min-sum-min algorithm generalizes the min-sum algorithm by introducing the penalty functions. Our goal in this paper is to study whether the min-sum-min algorithm converges for an arbitrary positive definite matrix J by choosing the weighting factor s properly.

3 Convergence of Min-Sum-Min Algorithm

In this section, we study the convergence of the min-sum-min algorithm. We first reformulate the message updating-expressions into vector forms. We then present the convergence analysis for the min-sum-min algorithm.

3.1 Reformulation of the Message Updating-Expressions

In this subsection, we reformulate the two updating expressions (15)-(16) into vector forms. The vector forms provide a big picture of the evolution of the algorithm.

We first consider the evolution of the linear vector $z^{(t)}$. From (15), we have

$$\begin{aligned} z^{(t+1)} = & (1-s)^2 B D^{(t)} y + s(1-s) B D^{(t)} \hat{x}_{edge}^{(t)} \\ & -(1-s) B D^{(t)} C z^{(t)} \quad t \geq 0 \end{aligned} \tag{22}$$

where the matrices $D^{(t)}, B, C \in \mathbb{R}^{|\overrightarrow{E}|\times|\overrightarrow{E}|}$, and the vector $y \in \mathbb{R}^{|\overrightarrow{E}|}$, are given by

$$D^{(t)}_{ij,uk} = \begin{cases} \gamma^{(t)}_{ij} & u = i, k = j \text{ and } [i,j] \in \overrightarrow{E} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{ij,uk} = \begin{cases} J_{ij} & u = i, k = j \text{ and } [i,j] \in \overrightarrow{E} \\ 0 & \text{otherwise} \end{cases}$$

$$C_{ij,uk} = \begin{cases} 1 & u \neq j, k = i \text{ and } [i,j], [u,k] \in \overrightarrow{E} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ij} = h_i \qquad [i,j] \in \overrightarrow{E}.$$

$D^{(t)}$ and B are two diagonal matrices. In particular $\gamma^{(t)} = D^{(t)} \cdot \mathbf{1}$, where $\mathbf{1}$ is the all-one vector. As $\gamma^{(t)} \to \gamma^*_s$ over time, $D^{(t)}$ converges to D^*_s.

Next we consider the evolution of $\hat{x}^{(t)}_{edge}$. By combining (16) and (22), we have

$$\begin{aligned} \hat{x}^{(t+1)}_{edge} = {} & (1-s)A^{(t)}D^{(t)}y + sA^{(t)}D^{(t)}\hat{x}^{(t)}_{edge} \\ & -(1-s)A^{(t)}D^{(t)}CBA^{(t)-1}\hat{x}^{(t)}_{edge} \quad t \geq 0, \end{aligned} \tag{23}$$

where the matrix $A^{(t)} \in \mathbb{R}^{|\overrightarrow{E}|\times|\overrightarrow{E}|}$ is given by

$$A^{(t)}_{ij,uk} = \begin{cases} \frac{1}{1-(1-s)^2 J^2_{ij}\gamma^{(t)}_{ij}\gamma^{(t)}_{ji}} & u = i, k = j \text{ and } [i,j] \in \overrightarrow{E} \\ \frac{-(1-s)J_{ij}\gamma^{(t)}_{ij}}{1-(1-s)^2 J^2_{ij}\gamma^{(t)}_{ij}\gamma^{(t)}_{ji}} & u = j, k = i \text{ and } [i,j] \in \overrightarrow{E} \\ 0 & \text{otherwise} \end{cases}.$$

The matrix $A^{(t)}$ converges to A^*_s as $\gamma^{(t)} \to \gamma^*_s$.

Upon obtaining (23), the remaining work is to study under what conditions $\hat{x}^{(t)}_{edge}$ converges to the optimal solution x^*_{edge}. To achieve this goal, we analyze (23) in two steps. In the first step, we consider the extreme case with D^*_s and A^*_s in (23). In this situation, $\hat{x}^{(t)}_{edge}$ can be alternatively expressed as

$$\hat{x}^{(t)}_{edge} = (1-s)A^*_s \sum_{i=0}^{t} \left[D^*_s(sA^*_s - (1-s)CB)\right]^i D^*_s y. \tag{24}$$

It is immediate from (24) that if the spectral radius of the matrix $D^*_s(sA^*_s - (1-s)CB)$ is less than 1 (i.e., $\rho(D^*_s(sA^*_s - (1-s)CB)) < 1$), $\hat{x}^{(t)}_{edge}$ converges to a fixed point as $t \to \infty$. We note that at this moment it is unclear if the fixed point $\hat{x}^{(\infty)}_{edge}$ is identical to x^*_{edge}.

In the second step, we consider the overall convergence specified by (23). We assume J_s is walk-summable and the spectral radius of $D^*_s(sA^*_s - (1-s)CB)$ is less than 1. By using the result of Lemma 1, it is known that there exists an integer K such that when $t \geq K$, the spectral radius of $D^{(t)}(sA^{(t)} - (1-s)CB)$ is less than one. This implies that $\hat{x}^{(t)}_{edge}$ in (23) also converges to a fixed point

provided with sufficient time. In fact, both (23) and (24) converge to the same fixed point. Due to limited space, we will not provide the proof here. One can refer to Section VI of [6] for a detailed argument on proving a similar result.

Based on the above analysis, we summarize the result in a lemma below.

Lemma 2. *Under the initialization $\gamma^{(0)} = 0$, if the matrix J_s in (19) is walk-summable and the spectral radius of $D_s^*(sA_s^* - (1-s)CB)$ is less than 1, the estimation vector $\hat{x}_{edge}^{(t)}$ converges to a fixed point. In particular, the fixed point is given by*

$$\lim_{t\to\infty} \hat{x}_{edge}^{(t)} = (1-s)A_s^*\big(I - sD_s^*A_s^* + (1-s)D_s^*CB\big)^{-1}D_s^*y. \tag{25}$$

Lemma 2 provides a general sufficient convergence condition for the min-sum-min algorithm. For the situation that the algorithm converges, one natural question is if the fixed point $\hat{x}_{edge}^{(\infty)}$ is identical to the optimal solution x_{edge}^*. To clarify, x_{edge}^* is constructed from x^*, and is of dimension $|\overrightarrow{E}|$. We show in the following that $\hat{x}_{edge}^{(\infty)} = x_{edge}^*$ when the algorithm converges. We let $\gamma^{(0)} = \gamma_s^*$ to simplify the argument.

Lemma 3. *Under the initialization $\gamma^{(0)} = \gamma_s^*$, if the matrix J_s in (19) is walk-summable and the spectral radius of $D_s^*(sA_s^* - (1-s)CB)$ is less than 1, the fixed point $\hat{x}_{edge}^{(\infty)}$ in (25) is the same as x_{edge}^**

$$x_{edge}^* = (1-s)A_s^*\big(I - sD_s^*A_s^* + (1-s)D_s^*CB\big)^{-1}D_s^*y. \tag{26}$$

Proof. From Lemma 2, it is clear that when the algorithm converges, the fixed point $\hat{x}_{edge}^{(\infty)}$ is independent of the initial vector $\hat{x}_{edge}^{(0)}$. In other words, any initialization would result in the same fixed point. In order to prove the lemma, we consider a special initialization for the estimation vector. That is $\hat{x}_{edge}^{(0)} = x_{edge}^*$. It is immediate from (10)-(11) and (16) that $\hat{x}_{edge}^{(t)} = \hat{x}_{edge}^*$ for any $t \geq 0$. The optimal solution x_{edge}^* is the fixed point. The proof is complete. □

Remark 3. In fact, one can generalize Lemma 2 by considering more general initializations. See [6] for how to initialize $\gamma^{(0)}$ and $z^{(0)}$. In this paper, we consider the special initialization for simplicity.

3.2 Convergence Analysis

We have known from (6) that the parameter s determines the amount of feedback in computing new messages and estimates. We show in the following that when s approaches to 1, the min-sum-min algorithm converges. We use the Taylor expansions in the argument.

As indicated in Lemma 3, the key point in proving the algorithm convergence is to study the spectral radius of the matrix $D_s^*(sA_s^* - (1-s)CB)$. Note that the two matrices D_s^* and A_s^* take complicated forms while the matrix CB is much simple. We now study the properties of A_s^* and D_s^* in detail. Due to the special structure of A_s^*, its inverse can be easily computed:

$$A_s^{*-1} = I + (1-s)D_s^* H, \tag{27}$$

where

$$H_{\{ij,uk\}} = \begin{cases} J_{ij} & u = j, k = i \text{ and } [i,j] \in \overrightarrow{E} \\ 0 & \text{otherwise} \end{cases}.$$

By using (27), the matrix A_s^* can be represented by an infinite series in terms of $D_s^* H$, which is given by $A_s^* = \sum_{i=0}^{\infty}(-1)^i(1-s)^i(D_s^* H)^i$. By using algebra on the infinite series, we obtain

$$A_s^* = I - (1-s)A_s^* D_s^* H. \tag{28}$$

Similarly, by applying the Taylor expansion on D_s^*, we have

$$D_s^* = I + (1-s)^2 D_s^* P_s, \tag{29}$$

where the matrix P_s is given by

$$P_{\{s,ij,uk\}} = \begin{cases} \sum_{v \in N(i)\setminus j} J_{vi}^2 \gamma_{\{s,vi\}}^* & [i,j] = [u,k] \text{ and } [i,j] \in \overrightarrow{E} \\ 0 & \text{otherwise} \end{cases}.$$

Now we are ready to study the matrix $D_s^*(sA_s^* - (1-s)CB)$. By applying (28)-(29), the matrix can be rewritten as

$$\begin{aligned} & D_s^*(sA_s^* - (1-s)CB) \\ &= D_s^*(sI - s(1-s)A_s^* D_s^* H - (1-s)CB) \\ &= D_s^*\big[sI - (1-s)CB - (1-s)D_s^* H \\ &\quad + (1-s)^2 D_s^* H + s(1-s)^2 A_s^*(D_s^* H)^2\big] \\ &= sI - (1-s)(CB + H) \\ &\quad + (1-s)^2 g(A_s^*, D_s^*, P_s, H, CB), \end{aligned} \tag{30}$$

where $g(\cdot)$ is a matrix function in terms of the matrices $\{A_s^*, D_s^*, P_s, H, CB\}$. Note that the last term in (30) is of second order of $(1-s)$. Also, as $s \to 1$, γ_s^* converges to $\mathbf{1}$. This implies that the matrices A_s^*, D_s^* and P_s are bounded when $s \in (\lfloor 1 - 1/\rho(\bar{R})\rfloor_+,\ 1)$. Thus, as $s \to 1$, the last term in (30) can be ignored, which results in

$$D_s^*(sA_s^* - (1-s)CB) \approx sI - (1-s)(CB + H), \quad \text{as } s \to 1.$$

To facilitate the analysis in the following, we denote $Q_s = sI - (1-s)(CB+H)$.

Next we derive the eigenvalues of the matrix Q_s. Denote the eigenvalues of J as $\{\lambda_i > 0, i = 1, \ldots, |V|\}$. We first note that the matrix $CB + H$ takes the form

$$(CB + H)_{ij,uk} = \begin{cases} J_{ui} & k = i \text{ and } [i,j],[u,k] \in \overrightarrow{E} \\ 0 & \text{otherwise} \end{cases}.$$

By relating the matrix $CB + H$ with $R = I - J$, one can show that all the non-zero eigenvalues of $CB + H$ are $\{\lambda_i - 1, i = 1, \ldots, |V|\}$. Finally, the eigenvalues of Q_s are give by

$$\{s + (1-s)(1-\lambda_i), i = 1, \ldots, |V|\} \bigcup \{s\}.$$

Using the fact that $\lambda_i > 0$ for all i, it can be shown that when $1 > s > \left\lfloor \frac{\rho(R)-1}{\rho(R)+1} \right\rfloor_+$ (i.e., $R = I - J$), the spectral radius of Q_s is less than 1. Further, as $s \to 1$, all the eigenvalues of Q_s approach to 1. As $\rho(R) \leq \rho(\bar{R})$ (see Corollary 6.3 in [1]), it is immediate that

$$\left\lfloor \frac{\rho(R)-1}{\rho(R)+1} \right\rfloor_+ \leq \lfloor 1 - 1/\rho(\bar{R}) \rfloor_+.$$

Thus, we can safely say that when $1 > s > \lfloor 1 - 1/\rho(\bar{R}) \rfloor_+$, the spectral radius of Q_s is less than 1.

The above analysis shows that if s is sufficiently close to 1, the min-sum-min algorithm converges, which we summarize in a theorem below.

Theorem 1. *If the parameter s is sufficiently close to 1 from below, the spectral radius of the matrix $D_s^*(sA_s^* - (1-s)CB)$ is less than 1. Consequently, the min-sum-min algorithm converges to the optimal solution.*

Remark 4. We point out that the matrix Q_s can be used to construct the message-updating expression of the Jacobi-relaxation algorithm [1]. In particular, the expression takes the form

$$\hat{x}_{edge}^{(t)} = \sum_{k=0}^{t-1} Q_s^k y + Q_s^t \hat{x}_{edge}^{(0)}.$$

Compared with Jacobi-relaxation algorithm, the min-sum-min algorithm updates the estimates nonlinearly in terms of the elements of J (see (23)), resulting in the last term in (30).

4 Dynamic Adaption of the Weighting Factor s

We have known thus far that when the weighting factor s is sufficiently close to 1, the min-sum-min algorithm converges to the optimal solution. Right now we cannot provide a fixed support region for s with guaranteed convergence. On the other hand, in practice, we have to choose some value for s. Intuitively

Table 2. The min-sum-min algorithm with dynamic parameter s

Initialization: $\gamma^{(0)} = 0$, $\hat{x}_{edge}^{(0)} = 0$, Flag = 0,
$s = \lfloor 1 - 1/\rho(\|R\|_1 \rfloor_+$, $s_{best} = s$, $r_{best} = 1$
repeat{min-sum-min iteration: t=1,2,...}
  if $r^{(t)}$ is stable AND Flag=0 **then**
    if $r^{(t)} < r_{best}$ **then**
      $r_{best} = r^{(t)}$, $s_{best} = s$, $s = s + 0.1$
    else
      Flag=1, $s = s_{best}$
    end if
  end if
until it terminates

speaking, if the parameter s is chosen to be very close to 1, the min-sum-min algorithm may take many iterations to reach the stoping criterion, making the algorithm less valuable. This motivate us to dynamically adjust the weighting factor s when running the min-sum-min algorithm.

We now explain how we adjust the weighting factor s in the algorithm. we first compress the estimation vector $\hat{x}_{edge}^{(t)}$ from dimension $\overrightarrow{E}$ to $|V|$. In particular, we compute an estimate $\hat{x}_i^{(t)}$ for each optimal component x_i^* by using $\{\hat{x}_i^{u,(t)}, u \in N(i)\}$:

$$x_i^{(t)} = \frac{1}{|N(i)|} \sum_{u \in N(i)} \hat{x}_i^{u,(t)}. \tag{31}$$

We denote the resulting estimation vector as $\hat{x}^{(t)} = [\hat{x}_i^{(t)}, \ldots, \hat{x}_{|V|}^{(t)}]$, which is of dimension $|V|$.

With the vector $\hat{x}^{(t)}$, we then define a new sequence $\{r^{(t)}, t \geq 2\}$:

$$r^{(t)} = \frac{\|\hat{x}^{(t)} - \hat{x}^{(t-1)}\|^2}{\|\hat{x}^{(t-1)} - \hat{x}^{(t-2)}\|^2}. \tag{32}$$

For a fixed parameter s, the sequence $\{r^{(t)}, t \geq 2\}$ would become stable after a number of iterations. We search for a value of s in $[\lfloor 1 - 1/\|R\|_1 \rfloor_+, 1)$ such that the corresponding stable value of the sequence $\{r^{(t)}, t \geq 2\}$ is as small as possible, which we denote as s_{best}. We note that once the value s_{best} is found after a number iterations, it will remain the same in the following iterations.

The pseudo-code of the min-sum-min algorithm with dynamic parameter s is provided in Table 2. The stepsize Δs for searching for the value s_{best} is set to be 0.1. The parameter "Flag" is used to indicate if the value s_{best} has been found or not.

Table 3. Numbers of iterations of the two algorithms for seven pairs of (J, h)

	1	2	3	4	5	6	7
$(\|V\|, \|E\|)$	(10,26)	(10,16)	(15,75)	(15,39)	(20,56)	(20,70)	(25, 182)
Jacobi-relaxation	1011	4635	4239	3498	6078	24130	15005
GLiCD	587	398	3050	2276	4568	18775	11516
min-sum-min	330	188	2474	1513	3691	16047	9801

4.1 Experiments with Synthetic Data

We tested the min-sum-min algorithm with the synthetic data. In the implementation of the min-sum-min algorithm in Table 2, we measured the error $|r^{(t)} - r^{(t-1)}|$ for checking the stability of the sequence $\{r^{(t)}, t \geq 2\}$. The threshold for $|r^{(t)} - r^{(t-1)}|$ was set as 10^{-4}.

We also implemented the Jacobi-relaxation and GLiCD algorithms for comparison (see Table 1). The GLiCD also has a free parameter s required to be adjusted in order to guarantee its convergence (see [10]). We adapted a similar procedure of Table 2 to adjust the parameter s for GLiCD. For the Jacobi-relaxation algorithm, it is known that when $s = \lfloor 1 - 1/\|R\|_1 \rfloor_+$, it converges to the optimal solution [1]. Therefore, we fixed the parameter $s = \lfloor 1 - 1/\|R\|_1 \rfloor_+$ in implementing the Jacobi-relaxation algorithm. To terminate the iterations of the three algorithms, the infinite norm between an estimate and the optimal solution was measured. The convergence threshold was set as 10^{-5}.

Seven pairs of (J, h) were randomly generated and tested by the three algorithms. The experimental results are displayed in Table 3. It is seen that the Jacobi-relaxation algorithm performs the worst in terms of number of iterations for all the seven optimization problems. Conversely, the min-sum-min algorithm performs the best. This might be because the quadratic messages carry more information than the linear messages (see Table 1 for the algorithmic comparison).

4.2 Experiments with Real Data

We also tested the three algorithms for the J matrices downloaded from the Matrix Market website repository [11], where the matrices originated from some real applications. The vector h in (1) were randomly generated. The implementation of the three algorithms were the same as for the synthetic data.

Fig. 1 displays the performance results of the three algorithms for two particular J matrices (one is of size 48×48 and the other one is of size 468×468). The min-sum-min and GLiCD algorithms converges significantly faster than the Jacobi-relaxation algorithm. This may be because the Jacobi-relaxation algorithm only involves linear updates of the estimates while the other two algorithms apply nonlinear updates of the estimate (See Remark 4 and [10]). Also we have observed that for the J matrix of size 48×48, the convergence speeds

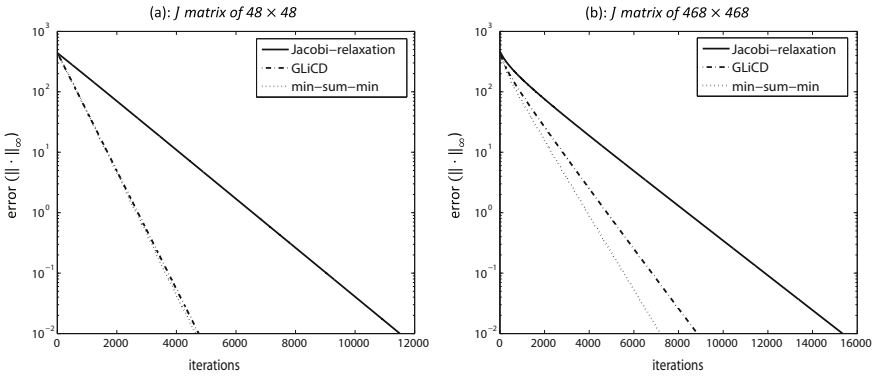

Fig. 1. Performance comparison for J matrices downloaded from [11]

of the min-sum-min and GLiCD algorithms are quite similar. For the above particular case, the GLiCD algorithm is favorable because it only transmits linear messages within the graph, thus saving half of transmission energy required for the min-sum-min algorithm. Other matrices were also tested and similar results were obtained.

5 Conclusion

In this paper, we have proposed the min-sum-min algorithm for the quadratic optimization problem. The min-sum-min algorithm parallels with the Jacobi-relaxation and GLiCD algorithms (See Table 1). Also we have studied the convergence of the min-sum-min algorithm. We have shown that if the feedback signal is set to be large enough (i.e., the parameter s is close to 1), the min-sum-min algorithm converges to the optimal solution. Experimental results show that the min-sum-min algorithm is advantageous over the Jacobi-relaxation and GLiCD algorithms in terms of the convergence speed.

References

1. Bertsekas, D.P., Tsitsikis, J.N.: Parallel and distributed Computation: Numerical Methods. Athena Scientific, Belmont (1997)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2007)
3. Johnson, J.K., Malioutov, D.M., Willsky, A.S.: Walk-sum Interpretation and Analysis of Gaussian Belief Propagation. In: Advances in Neural Information Processing Systems, vol. 18. MIT Press, Cambridge (2006)
4. Johnson, J.K., Bickson, D., Dolev, D.: Fixing Convergence of Gaussian Belief Propagation. In: The International Symposium on Information Theory (2009)
5. Malioutov, D.M., Johnson, J.K., Willsky, A.S.: Walk-Sums and Belief Propagation in Gaussian Graphical Models. J. Mach. Learn. Res. 7, 2031–2064 (2006)
6. Moallemi, C.C., Van Roy, B.: Convergence of Min-Sum Message Passing for Quadratic Optimization. IEEE Trans. Inf. Theory 55(5), 2413–2423 (2009)

7. Moallemi, C.C., Van Roy, B.: Convergence of Min-Sum Message Passing for Convex Optimization. IEEE Trans. Inf. Theory 56(4), 2041–2050 (2010)
8. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufman Publishers (1988)
9. Zhang, G., Heusdens, R.: Linear Coordinate-Descent Message-Passing for Quadratic Optimization. Neural Computation 24(12), 3340–3370 (2012)
10. Zhang, G., Heusdens, R.: Convergence of Generalized Linear Coordinate-Descent Message-Passing for Quadratic Optimization. In: IEEE International Symposium on Information Theory Proceedings, pp. 1997–2001 (2012)
11. Matrix Market: Harwell Boeing Collection, `http://math.nist.gov/MatrixMarket/index.html`