DISTRIBUTED FIRST AND SECOND ORDER METHODS WITH INCREASING NUMBER OF WORKING NODES

Dušan Jakovetić[1,2]*, Nataša Krklec Jerinkić*[2]*, Nataša Krejić*[2]*, and Dragana Bajović*[1,3]

[1]BioSense Institute, Novi Sad, Serbia
[2]Faculty of Sciences, University of Novi Sad, Serbia
[3]Faculty of Technical Sciences, University of Novi Sad, Serbia

ABSTRACT

We consider distributed optimization problems where nodes in a connected network collaboratively minimize the sum of their locally known convex costs subject to a common (vector-valued) optimization variable. In this paper, we present a mechanism to significantly improve the computational and communication efficiency of some recently proposed first and second order distributed methods for solving such problems. The presented mechanism relaxes the requirement that all nodes are active (i.e., update their solution estimates and communicate with neighbors) at all iterations k. Instead, each node is active at iteration k with probability p_k, where p_k is increasing to unity, while the activations are independent both across nodes and across iterations. Assuming strongly convex and twice continuously differentiable local costs and that p_k grows to one linearly, both first and second order methods with the idling schedule exhibit very similar theoretical convergence and convergence rate properties as if all nodes were active at all iterations. Simulation examples demonstrate that incorporating the idling schedule in first and second order distributed methods significantly improves their computational and communication efficiencies.

Index Terms— Distributed optimization, distributed gradient method, distributed quasi Newton method, increasing number of working nodes, convergence rate, consensus.

1. INTRODUCTION

We consider unconstrained distributed optimization problems where N nodes are situated in a generic, connected network, each node has access to its local convex cost f_i, and the nodes' common goal is to minimize the aggregate sum of their local costs subject to a common (vector-valued) optimization variable. To solve this and related problems, a number of first order distributed methods, e.g., [1, 2, 3], and second order distributed methods, e.g., [4, 5, 6] has been proposed and analyzed, and their applicability has been demonstrated on various distributed learning, estimation, and control use cases.

Research of N. Krejić and N. Krklec-Jerinkić is supported by Ministry of Education, Science and Technological Development, Republic of Serbia, grant no. 174030.

In this paper, we present a mechanism, based on an idling schedule of nodes, that significantly improves the efficiency of such distributed methods, both in terms of communication and computational costs. With this mechanism, each node i, at each iteration k, is working–active with probability p_k, and is inactive with probability $1 - p_k$, while different nodes' activations are independent both across nodes and across iterations. Active nodes perform updates and exchange their solution estimates with their neighboring (active) nodes, while inactive nodes stay idle. Quantity p_k is increasing over iterations k to unity, so that, on average, as the algorithm progresses, an increasing number of nodes is working and becomes involved in the optimization process.

The motivation for the proposed approach comes from centralized (hybrid) stochastic-deterministic gradient methods to minimize a sum of component functions f_i's, [7]; see also [8, 9]. Reference [7] shows that, if the sample size (number of component functions f_i's involved) appropriately increases as the iteration counter k grows, the algorithm can achieve practically the same convergence rate as if the full sample size is used across all iterations, thereby yielding significant computational savings with respect to the standard method that always utilizes the full sample.

Specifically, we apply here the idling mechanism to the distributed first order method in [1] and the distributed second order method in [6]. We assume that p_k grows to one at a linear (geometric) rate, and that costs f_i's are twice continuously differentiable with bounded Hessian. Under this setting, we present results that show that, with both first and second order distributed methods, the corresponding method with the incorporated idling mechanism converges to the same solution – both in the mean square sense (MSS) and almost surely – as if all nodes were active at all iterations; moreover, the MSS convergence occurs at a globally linear (geometric) rate. Furthermore, with the distributed first order method and an appropriately tuned p_k's convergence factor, the method with idling schedule achieves practically the same linear convergence factor as if all nodes were active at all iterations. Simulation examples demonstrate that the presented idling mechanism indeed allows for significant computational and com-

munication savings.

This paper is a continuation of [10] where we proposed the idling schedule and considered (projected) first order distributed methods for constrained problems with compact, convex constraints. Here, we consider *unconstrained problems* and both *first and second order* distributed methods. Proofs of the results presented here are omitted due to lack of space and will be included in an extended companion paper.

The remainder of the paper is organized as follows. The next paragraph sets notation. Section 2 describes the network, optimization, and generic idling schedule models. Sections 3 and 4 present, respectively, the first and second order distributed algorithms with idling schedules and results on convergence and convergence rates for the methods. Section 5 gives simulation examples. Finally, we conclude in Section 6.

We use the following notation. We denote by: $\mathbb{R}$ the set of real numbers; $\mathbb{R}^d$ the d-dimensional Euclidean real coordinate space; A_{ij} the entry in the i-th row and j-th column of a matrix A; I, and 0, respectively, the identity matrix and the zero matrix. $A \succ 0\,(A \succeq 0)$ means that the symmetric matrix A is positive definite (respectively, positive semidefinite); $\|\cdot\|$ the Euclidean (respectively, spectral) norm of its vector (respectively, matrix) argument; $\lambda_i(\cdot)$ the i-th largest eigenvalue; $\nabla h(w)$ and $\nabla^2 h(w)$ the gradient and Hessian, respectively, evaluated at w of a function $h : \mathbb{R}^d \to \mathbb{R}$, $d \geq 1$; $\mathbb{P}(\mathcal{A})$ and $\mathbb{E}[u]$ the probability of an event $\mathcal{A}$ and expectation of a random variable u, respectively. Finally, for two positive sequences η_n and χ_n, we have: $\eta_n = O(\chi_n)$ if $\limsup_{n\to\infty} \frac{\eta_n}{\chi_n} < \infty$.

2. MODEL AND IDLING MECHANISM

Optimization model. We consider the following unconstrained optimization problem:

$$\text{minimize} \;\; \textstyle\sum_{i=1}^N f_i(x) =: f(x). \tag{1}$$

Here, recall that N is the number of nodes, and $f_i : \mathbb{R}^d \mapsto \mathbb{R}$ is a function known only to node i. For all i, we assume that $f_i : \mathbb{R}^d \mapsto \mathbb{R}$ is twice continuously differentiable and has a bounded Hessian, i.e., there exist positive constants μ and L, $\mu \leq L$, such that, for all $x \in \mathbb{R}^d$: $\mu\, I \preceq \nabla^2 f_i(x) \preceq L\, I$. Under this Assumption, problem (1) is solvable and has the unique solution, which we denote by $x^\star$. The requirement on the existence of (continuous) second derivative of the f_i's can be relaxed with the first order method in Section 3; see [10] for details.

Network model. Nodes are connected in a generic undirected network $\mathcal{G} = (\mathcal{V}, E)$, where $\mathcal{V}$ is the set of N nodes and E is the set of edges, i.e., (unordered) node pairs $\{i, j\}$ that can exchange messages.The network $\mathcal{G} = (\mathcal{V}, E)$ is assumed connected, undirected, and simple (no self-loops nor multiple links). Denote by Ω_i the neighborhood set of node i (excluding i). We associate with $\mathcal{G}$ a $N \times N$ symmetric weight matrix W, which is also stochastic (rows sum to one and all the entries are non-negative). We let W_{ij} be strictly positive for each $\{i, j\} \in E$, $i \neq j$; $W_{ij} = 0$ for $\{i, j\} \notin E$, $i \neq j$; and $W_{ii} = 1 - \sum_{j\neq i} W_{ij}$. We assume that there exist two constants $0 < w_{min} \leq w_{max} < 1$, such that $W_{ii} \in [w_{min},\, w_{max}]$, for all i. Further, we let W be positive definite, i.e., $\lambda_N(W) > 0$. This are mild assumptions; see, e.g., [10], on how a matrix W that fulfills these assumptions can be set in a distributed way. It can be shown that, under the above assumptions on W, $\lambda_1(W) = 1$, and $\lambda_2(W) < 1$.

Time model and idling mechanism. With both the first order and the second order distributed methods considered in Sections 3 and 4, we assume that all nodes are synchronized according to a global clock and simultaneously (in parallel) perform algorithm iterations $k = 0, 1, ...$ Each node has an internal Bernoulli state variable $z_i^{(k)}$. If $z_i^{(k)} = 1$, node i is active (or working) at iteration k, i.e., it performs computations and communicates with neighbors. If $z_i^{(k)} = 0$, node i does not perform an update nor it communicates; we say that, in this case, node i is idle. At each k, each node i generates $z_i^{(k)}$ independently from the previous iterations, and independently from other nodes. We denote by $p_k := \mathbb{P}\,(z_i(k) = 1)$. The quantity p_k is, for simplicity, assumed common for all nodes. Also, for all k, $p_k \geq p_{\min}$, for a positive constant $p_{\min}$. For future reference, we denote by $\Omega_i^{(k)}$ the set of *active* neighbors of node i at k, i.e., all nodes $j \in \Omega_i$ with $z_j^{(k)} = 1$.

3. DISTRIBUTED FIRST ORDER METHOD

We now describe the distributed first order algorithm with the idling schedule. At each iteration k, each node i updates its solution estimate $x_i^{(k)} \in \mathbb{R}^d$, with arbitrary initialization $x_i^{(0)} \in \mathbb{R}^d$. The update of node i is as follows. If $z_i^{(k)} = 0$, node i is idle and sets $x_i^{(k+1)} = x_i^{(k)}$. Otherwise, if $z_i^{(k)} = 1$, node i broadcasts its state to all its working neighbors $j \in \Omega_i^{(k)}$. The idle neighbors do not receive $x_i^{(k)}$; e.g.,, with wireless sensor networks, this corresponds to switching-off the receiving antenna. Likewise, node i receives $x_j^{(k)}$ from all $j \in \Omega_i^{(k)}$. Upon reception, node i updates $x_i^{(k)}$ as follows:

$$\begin{aligned} x_i^{(k+1)} \;&=\; \left(1 - \sum_{j\in\Omega_i^{(k)}} W_{ij}\right) x_i^{(k)} \\ &+\; \sum_{j\in\Omega_i^{(k)}} W_{ij}\, x_j^{(k)} - \frac{\alpha}{p_k} \nabla f_i(x_i^{(k)}). \end{aligned} \tag{2}$$

In (2), $\alpha > 0$ is a constant; we let $\alpha \leq \lambda_N(W)/L$. The step size in (2) is multiplied by $1/p_k$, to compensate for non-working (idle) nodes over iterations. Also, note that the standard distributed (sub)gradient method in [11] is recovered with $p_k = 1$, $\forall k$. With the method in [11], it is known that node i's solution estimate converges to a solution

neighborhood, i.e., to a point $x_i^\bullet$, where $\|x_i^\bullet - x^\star\| = O(\alpha)$, $i = 1, ..., N$, e.g., [12]. We have the following result.

Theorem 1 Consider algorithm (2) with $\alpha \leq \lambda_N(C)/L$. Further, let $p_k = 1 - \delta^{k+1}$, $\forall k$, for some $\delta \in (0, 1)$. Then, $x_i^{(k)}$ converges, both in MSS and almost surely, to the convergence point $x_i^\bullet$ of the method in [1], $i = 1, ..., N$. Furthermore, let $\eta := \max\{1 - \alpha\mu, \delta^{1/2}\}$, and $\sqrt{\delta} \leq 1 - \alpha\mu$. Then: $\mathbb{E}\left[\|x^{(k)} - x^\bullet\|\right] = O\left(k(1-\alpha\mu)^k\right) = O\left((1 - \alpha\mu + \epsilon)^k\right)$, for arbitrarily small $\epsilon > 0$.

Theorem 1 states that, under appropriately set δ, (2) converges at a practically same rate as [1]. (It can be shown that the bound $O\left((1-\alpha\mu)^k\right)$ is tight for [1].) This still does not explicitly quantify savings with (2), due to the involved constants, but significant savings indeed occur in practice; see [10].

4. DISTRIBUTED SECOND ORDER METHOD

We now incorporate the idling mechanism in the distributed second order method in [6]. We refer to [6] for the rationale behind the method and derivations, while here we present the method variant with the idling schedule.[1] Like with the distributed first order method in Section 3, each node i can be in two possible states (active or idle) depending on the realization of the Bernoulli state $z_i^{(k)}$. Also, each node i still maintains its solution estimate $x_i^{(k)} \in \mathbb{R}^d$ over iterations. However, now local Hessians $\nabla^2 f_i(x_i^{(k)})$ are also evaluated and incorporated in the update rule. The distributed second order method with idling is presented in Algorithm 1 below.

Algorithm 1

At each node i, require $\alpha, \rho, \beta > 0$, and p_k, $k = 0, 1, ...$

(1) Initialization: Each node i sets $k = 0$ and $x_i^{(0)} \in \mathbb{R}^d$.

(2) Each node i generates $z_i^{(k)}$; if $z_i^{(k)} = 0$, node i stays idle, sets $x_i^{(k+1)} = x_i^{(k)}$, and goes to step (9); otherwise, node i is active and performs steps (2)-(9) (in parallel with other active nodes).

(3) Each active node i transmits $x_i^{(k)}$ to all its active neighbors $j \in \Omega_i^{(k)}$ and receives $x_j^{(k)}$ from all $j \in \Omega_i^{(k)}$.

(4) Each active node i calculates

$$g_i{}^{(k)} = \left(A_i^{(k)}\right)^{-1} \left[\frac{\alpha}{p_k} \nabla f_i(x_i^{(k)}) + \sum_{j \in \Omega_i^{(k)}} W_{ij} \left(x_i^{(k)} - x_j^{(k)}\right) \right], \quad (3)$$

$$\text{where } A_i^{(k)} = \alpha \nabla^2 f_i(x_i^{(k)}) + (1 - W_{ii})\, I. \quad (4)$$

(5) Each active node i transmits $g_i{}^{(k)}$ to all its active neighbors $j \in \Omega_i^{(k)}$ and receives $g_j^{(k)}$ from all $j \in \Omega_i^{(k)}$.

(6) Each active node i chooses a diagonal $d \times d$ matrix $\Lambda_i^{(k)}$, such that $\|\Lambda_i^{(k)}\| \leq \rho$.

(7) Each active node i calculates: $s_i^{(k)} = -g_i{}^{(k)} + \Lambda_i^{(k)} \sum_{j \in \Omega_i^{(k)}} W_{ij}\, g_j^{(k)}$.

(8) Each active node i updates its solution estimate as: $x_i^{(k+1)} = x_i^{(k)} + \beta\, s_i^{(k)}$.

(9) Set $k = k + 1$ and go to step (2).

We now briefly comment on the algorithm and the parameters and quantities involved. First, it is easy to see that, setting $\beta = 1$, $\Lambda_i^{(k)} = 0$, and replacing matrix $A_i^{(k)}$ with the identity, we recover the first order method in Section 3. However, intuitively, matrix $A_i^{(k)}$ allows each node to "scale" its search direction with its locally available second order information. Furthermore, taking non-zero matrices $\Lambda_i^{(k)}$'s allows one to capture second-order information beyond the "node-decoupled block-diagonal approximation" – attainable through the $A_i^{(k)}$'s only (see [6] for details). Note that Algorithm 1 with $\Lambda_i^{(k)} = 0$ involves a single communication round per k, just like the method in Section 3; taking a non-zero $\Lambda_i^{(k)}$ induces an additional communication round – two rounds per k in total. Parameter β is the algorithm step size, and, theoretically, it should be taken small enough to ensure a global linear convergence, similarly to standard (centralized) Newton methods. Quantity $\rho > 0$ is a safeguarding parameter that should be appropriately set (below a threshold value) to ensure that Algorithm 1 is a descent method (in the MSS); see also [6]. For details on specific choices of $\Lambda_i^{(k)}$ (without idling), we refer to [6]. Here, although the theoretical results presented here allow for generic $\Lambda_i^{(k)}$, in Section 5 we consider easy-to-set choices $\Lambda_i^{(k)} = 0$ and $\Lambda_i^{(k)} = -I$. We have the following result.

Theorem 2 Consider Algorithm 1 with $p_k = 1 - \delta^{k+1}$, $k = 0, 1, ...$, for some $\delta \in (0, 1)$. Then, there exist constants $\overline{\rho} > 0$ and $\overline{\beta} > 0$, dependent on $\delta, \alpha, \mu, L, w_{min}$, and w_{max}, such that, for any $\rho \in (0, \overline{\rho})$, and for any $\beta \in (0, \overline{\beta})$, $x_i^{(k)}$ converges to $x_i^\bullet$ almost surely and in the MSS, for all i. Moreover, the MSS convergence is (globally) linear, i.e., $\mathbb{E}\left[\|x_i^k - x_i^\bullet\|^2\right]$ converges to zero at a linear rate.

Theorem 2 says that, despite the idling, the distributed second order method achieves a globally linear convergence rate (in MSS), which matches the order of convergence of the method when all nodes are active across all iterations [6]. This Theorem does not explicitly establish savings in communications and computations with the idling schedule introduced; however, savings actually occur in practice, as demonstrated on examples in the next section.

[1] The method in [6] is presented here for simplicity with the parameter θ therein set to zero.

5. SIMULATION EXAMPLE

We now present a simulation example that demonstrates the effectiveness of incorporating the idling schedule in the distributed second order method in [6]. We refer to [10] for several simulation examples on the distributed first order method (more precisely, its variant for constrained problems) that also show significant gains achieved through the idling mechanism.

The simulation setup is as follows. We consider a connected network with $N = 30$ nodes, generated as a random geometric graph: nodes are randomly (uniformly) placed on a unit square, and the node pairs whose distance is less than a radius are connected by an edge. Each node's local cost is quadratic, i.e., $f_i(x) = \frac{1}{2}(x - a_i)^\top B_i(x - a_i)$, $i = 1, ..., N$, where $d = 4$, $B_i \in \mathbb{R}^{d\times d}$ is a positive definite (symmetric matrix), and $a_i \in \mathbb{R}^d$ is a vector. Matrices B_i and vectors a_i are generated randomly, as described in [6]. We compare the second order distributed method [6] without the idling mechanism ($p_k = 1$, for all k), and the method with the idling mechanism, where $p_k = 1 - \delta^{k+1}$, $k = 0, 1, ...$, with $\delta = 0.995$. With both variants, we initialize all nodes' solution estimates to zero. We use the weights $W_{ij} = \frac{1}{1+2\max\{d_i,d_j\}}$, for $\{i, j\} \in E$ (where d_i is node i's degree); $W_{ij} = 0$, for $\{i, j\} \notin E$, $i \neq j$; and $W_{ii} = 1 - \sum_{j\neq i} W_{ij}$, $i = 1, ..., N$. We let $\alpha = 1/(200L)$, where the Lipschitz constant $L = \max_{i=1,...,N} \|B_i\|$. Further, we set $\beta = 1$, and $\rho = +\infty$ (no safeguarding).[2]

Figure 1 (top) compares the method with idling (increasing number of working nodes) and without idling (all nodes working at all iterations) for $\Lambda_i^{(k)} = 0$, for all i, k. The x-axis shows the total cost–total number of activations per node up to iteration k (which corresponds to the communication and computational cost up to k), while y-axis shows the relative error $\frac{1}{N}\sum_{i=1}^N \frac{\|x_i^{(k)}-x^\star\|}{\|x^\star\|}$, where we recall that $x^\star$ is the solution to (1), $x^\star \neq 0$. We can see that the idling mechanism significantly improves the algorithm efficiency: to reach the "steady state" relative error (≈ 0.0075), the method with idling has a total cost around 750, while the method without idling has the total cost around 950; this represents savings of more than 20%. Figure 1 (bottom) repeats the plots for $\Lambda_i^{(k)} = -I$, for all i, k. First, we can see that the methods with $\Lambda_i^{(k)} = -I$ converge faster in terms of the number of activations than the corresponding methods with $\Lambda_i^{(k)} = 0$. This is expected, as the choice $\Lambda_i^{(k)} = 0$ captures only "block-diagonal" Hessian information; also, it utilizes one communication per node, per activation, while the other choice utilizes two. The relative savings of incorporating the idling mechanism are similar ($\approx 20\%$.)

[2]Note that the latter choice for β and ρ is not covered theoretically by Theorem 2, but extensive simulations on strongly convex quadratic costs demonstrate convergence in this case as well.

6. CONCLUSION

We presented an idling mechanism to improve efficiency of first and second order distributed methods. With this mechanism, each node i, at each iteration k, is active with probability p_k (performing communications and solution estimate updates) and stays idle with probability $1 - p_k$, while the activations are independent across nodes and across iterations. Specifically, we incorporated the idling mechanism in the distributed first order method in [1] and in the distributed second order method in [6]. We demonstrated theoretically that, when p_k increases to one at a geometric rate, both first and second order methods exhibit similar convergence and convergence rate properties as the corresponding methods where all nodes are constantly active. Finally, we demonstrated by simulation that the incorporation of the idling mechanism yields significant computational and communication savings.

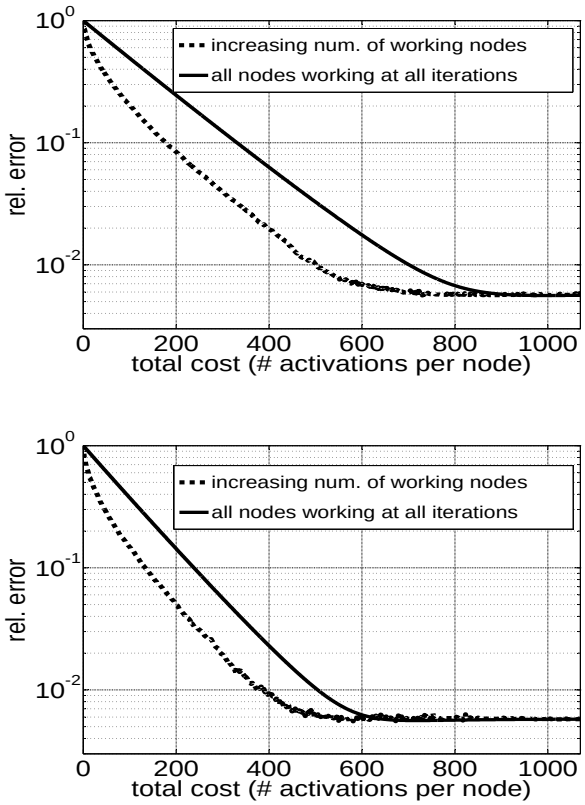

Fig. 1. Comparison of the second order distributed method in [6] with increasing number of working nodes (with the idling mechanism), and with all nodes working all the time (no idling mechanism); Top: $\Lambda_i^{(k)}$ in Algorithm 1 is set to zero, for all i, k; Bottom: $\Lambda_i^{(k)} = -I$, for all i, k.

7. REFERENCES

[1] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, January 2009.

[2] S.S. Ram, A. Nedic, and V.V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *J. Optim. Theory Appl.*, vol. 147, no. 3, pp. 516–545, 2011.

[3] D. Jakovetic, J. Xavier, and J. M. F. Moura, "Fast distributed gradient methods," *IEEE Trans. Autom. Contr.*, vol. 59, no. 5, pp. 1131–1146, May 2014.

[4] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton-part I: Algorithm and convergence," *submitted to IEEE Transactions on Signal Processing*, 2015, available at: http://arxiv.org/abs/1504.06017.

[5] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton-part II: Convergence rate and implementation," *IEEE Transactions on Signal Processing*, 2015, available at: http://arxiv.org/abs/1504.06020.

[6] D. Bajovic, D. Jakovetic, N. Krejic, and N. Krklec Jerinkic, "Newton-like method with diagonal correction for distributed optimization," 2015., available at: http://arxiv.org/abs/1509.01703.

[7] M. Friedlander and M. Schmidt, "Hybrid deterministic-stochastic methods for data fitting," *SIAM J. Sci. Comput.*, vol. 34, no. 3, pp. A1380–A1405, 2012, DOI: http://dx.doi.org/10.1137/110830629.

[8] T. Homem de Mello, "Variable-sample methods for stochastic optimization," *ACM Transactions on Modeling and Computer Simulation*, vol. 13, no. 2, pp. 108–133, 2003.

[9] E. Polak and J. O. Royset, "Efficient sample sizes in stochastic nonlinear programing," *Journal of Computational and Applied Mathematics*, vol. 217, no. 2.

[10] D. Jakovetic, D. Bajovic, N. Krejic, and N. Krklec-Jerinkic, "Distributed gradient methods with variable number of working nodes," *to appear in IEEE Trans. Sig. Process.*, 2016.

[11] A. Nedic, A. Ozdaglar, and A.P. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, April 2010.

[12] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *to appear in SIAM Journal on Optimization*, 2015, available at: http://arxiv.org/abs/1310.7063.