Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Improved Dual Decomposition for Distributed Model Predictive Control ⋆

Pontus Giselsson *

* *Electrical Engineering, Stanford University*
(e-mail: pontusg@stanford.edu).

Abstract:
In dual decomposition, the dual to an optimization problem with a specific structure is solved in distributed fashion using (sub)gradient and recently also fast gradient methods. The traditional dual decomposition suffers from two main short-comings. The first is that the convergence is often slow, although fast gradient methods have significantly improved the situation. The second is that computation of the optimal step-size requires centralized computations, which hinders a fully distributed implementation of the algorithm. In this paper, the first issue is addressed by providing a tighter quadratic approximation of the dual function than what has previously been reported in the literature. Then a distributed algorithm is presented in which the provided dual function approximation is minimized in each step. Since the approximation is more accurate than the approximation used in standard and fast dual decomposition, the convergence properties are improved. For the second issue, we extend a recent result to allow for a fully distributed parameter selection in the algorithm. Further, we show how to apply the algorithm to optimization problems arising in distributed model predictive control (DMPC) and that the proposed algorithm enjoys distributed reconfiguration, i.e. *plug-and-play*, in the DMPC context.

1. INTRODUCTION

Optimization problems with a separable cost and sparse constraints can be solved in distributed fashion by distributed optimization algorithms. Most distributed algorithms exploit the property that the (sub)gradient to the dual of such optimization problems can be computed in distributed fashion, which enables for distributed implementation of dual (sub)gradient algorithms. This approach is referred to as dual decomposition and originates from Everett (1963); Danzig and Wolfe (1961); Benders (1962). The use of sub-gradient or gradient methods to solve the dual problem usually results in poor convergence properties of the algorithm. A remedy to this is to instead use a fast gradient method. The first fast gradient method was presented in Nesterov (1983) and has since been extended and generalized in Beck and Teboulle (2009); Nesterov (2003); Tseng (2008). The benefit of using fast gradient methods is that, with negligible increase in iteration complexity, the convergence rate is improved from $O(1/k)$ in standard gradient methods to $O(1/k^2)$, where k is the iteration number. In Giselsson et al. (2013) this approach was taken, i.e. dual decomposition algorithms based on fast gradient methods were presented, and significantly improved convergence behavior was reported, compared to using standard dual decomposition methods. However, in many applications further improvements are necessary for realistic implementation.

⋆ During the preparation of this paper, the author was a member of the LCCC Linnaeus Center at Lund University. Financial support from the Swedish Research Council for the author's Postdoctoral studies at Stanford University is gratefully acknowledged. Eric Chu is also gratefully acknowledged for constructive feedback.

One prerequisite to apply fast gradient methods is that the function to be minimized is convex and differentiable and has a Lipschitz continuous gradient. These properties are equivalent to the existence of a quadratic upper bound with the same curvature in all directions (defined by the Lipschitz constant) to the function to be minimized. In gradient and fast gradient methods, the main step in every iteration is to minimize this quadratic upper bound. This is equivalent to taking a step in the negative gradient direction. If the bound does not well approximate the function to be minimized, slow convergence properties are expected. By instead letting the quadratic upper bound have different curvature in different directions, a closer fit between the bound and the function can be obtained. For an appropriate choice of non-uniform quadratic upper bound, this can significantly improve the convergence properties of fast gradient methods. The key result of this paper is a characterization of the set of matrices that can be used to describe a quadratic upper bound to the convex negative dual function, in the case of strongly convex primal cost function. This result generalizes previous results, e.g. Nesterov (2005), where a Lipschitz constant to the dual gradient is quantified. As a consequence of the presented result, quadratic upper bounds with different curvature in different directions can be used in dual decomposition methods. For an appropriate choice of quadratic upper bound, this can significantly improve the convergence in dual decomposition.

To enable distributed implementation of the dual fast gradient method, i.e. dual decomposition, the matrix that describes the quadratic upper bound must be block diagonal. In standard dual decomposition, this matrix is traditionally chosen as the reciprocal of the Lipschitz constant to

the dual gradient times the identity matrix. Besides giving slow convergence properties as previously discussed, this choice usually requires centralized computations. In this paper, we extend a recent result in Beck et al. (2014) to enable a fully distributed initialization procedure to select the block diagonal matrix that describes the quadratic upper bound. The extension relies on the new characterization of the set of matrices that can be used to describe the quadratic upper bound to the dual function.

In distributed model predictive control (DMPC), dual decomposition techniques have been used to distribute the computations over the subsystems Negenborn (2007); Doan et al. (2011); Giselsson et al. (2013). Although the use of fast gradient methods in dual decomposition have significantly improved the convergence, see Giselsson et al. (2013), it is not enough for realistic implementation in a distributed control system. In Giselsson (2013), a generalized version of dual decomposition was presented that allows for different curvature in different directions in the quadratic upper bound that is minimized in every iteration of the algorithm. This gives a significantly reduced number of iterations. The algorithm in Giselsson (2013) is restricted to problems having a quadratic cost, linear equality constraints, and linear inequality constraints. Dual variables for all these constraints are introduced, which results in the dual problem being a quadratic program. The algorithm in this paper is an extension and generalization of the algorithm in Giselsson (2013) that allows for any (local) convex inequality constraints. Also, only the equality constraints are dualized in this paper. These changes give rise to completely different technicalities since the dual function is implicitly defined though an optimization problem.

A feature of DMPC is that similar optimization problems are repeatedly solved online. This implies that much offline computational effort can be devoted to parameter selection in the algorithm to improve the online convergence. In this paper, the offline computational effort is devoted to choose a matrix that describes the quadratic upper bound to the negative dual function. The numerical evaluation suggests that this can significantly reduce the number of iterations in the algorithm. Besides favorable convergence properties, the presented algorithm also enjoys distributed configuration and reconfiguration, commonly referred to as plug-and-play. Distributed reconfiguration or plug-and-play is the property that if a subsystem is added to (or removed from) the system, only neighboring subsystems need to be invoked to reconfigure the algorithm for the new setup.

For space consideration, all proofs are omitted from this paper and can be found in the full version paper, Giselsson (2014).

2. PRELIMINARIES AND NOTATION

2.1 Notation

We denote by $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{m\times n}$, the sets of real numbers, vectors, and matrices. $\mathbb{S}^n \subseteq \mathbb{R}^{n\times n}$ is the set of symmetric matrices, and $\mathbb{S}^n_{++} \subseteq \mathbb{S}^n$, $[\mathbb{S}^n_+] \subseteq \mathbb{S}^n$, are the sets of positive [semi] definite matrices. We also use notation $\langle x, y\rangle = x^T y$, $\|x\|_2 = \sqrt{x^T x}$, and $\|x\|_H = \sqrt{x^T H x}$. Finally, $I_{\mathcal{X}}$ denotes the indicator function for the set $\mathcal{X}$, i.e. $I_{\mathcal{X}}(x) \triangleq \begin{cases} 0, & x\in\mathcal{X} \\ \infty, & \text{else} \end{cases}$.

2.2 Preliminaries

In this section, we introduce generalizations of already well used concepts. We generalize the notion of strong convexity as well as the notion of Lipschitz continuity of the gradient of convex functions. We also define conjugate functions and state a known result on dual properties of a function and its conjugate.

For differentiable and convex functions $f : \mathbb{R}^n \to \mathbb{R}$ that have a Lipschitz continuous gradient with constant L, we have that

$$\|\nabla f(x_1) - \nabla f(x_2)\|_2 \leq L\|x_1 - x_2\|_2 \quad (1)$$

holds for all $x_1, x_2 \in \mathbb{R}^n$. This is equivalent to that

$$f(x_1) \leq f(x_2) + \langle \nabla f(x_2), x_1 - x_2\rangle + \frac{L}{2}\|x_1 - x_2\|_2^2 \quad (2)$$

holds for all $x_1, x_2 \in \mathbb{R}^n$ (Nesterov, 2003, Theorem 2.1.5). In this paper, we allow for a generalized version of the quadratic upper bound (2) to f, namely that

$$f(x_1) \leq f(x_2) + \langle \nabla f(x_2), x_1 - x_2\rangle + \frac{1}{2}\|x_1 - x_2\|_{\mathbf{L}}^2 \quad (3)$$

holds for all $x_1, x_2 \in \mathbb{R}^n$ where $\mathbf{L} \in \mathbb{S}^n_+$. The bound (2) is obtained by setting $\mathbf{L} = LI$ in (3).

Remark 1. For concave functions f, i.e. where $-f$ is convex, the Lipschitz condition (1) is equivalent to that the following quadratic lower bound

$$f(x_1) \geq f(x_2) + \langle \nabla f(x_2), x_1 - x_2\rangle - \frac{L}{2}\|x_1 - x_2\|_2^2 \quad (4)$$

holds for all $x_1, x_2 \in \mathbb{R}^n$. The generalized counterpart naturally becomes that

$$f(x_1) \geq f(x_2) + \langle \nabla f(x_2), x_1 - x_2\rangle - \frac{1}{2}\|x_1 - x_2\|_{\mathbf{L}}^2 \quad (5)$$

holds for all $x_1, x_2 \in \mathbb{R}^n$.

Next, we state a Lemma on equivalent characterizations of the condition (3).

Lemma 2. Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable. The condition that

$$f(x_1) \leq f(x_2) + \langle \nabla f(x_2), x_1 - x_2\rangle + \frac{1}{2}\|x_1 - x_2\|_{\mathbf{L}}^2 \quad (6)$$

holds for some $\mathbf{L} \in \mathbb{S}^n_+$ and all $x_1, x_2 \in \mathbb{R}^n$ is equivalent to that

$$\langle \nabla f(x_1) - \nabla f(x_2), x_1 - x_2\rangle \leq \|x_1 - x_2\|_{\mathbf{L}}^2. \quad (7)$$

holds for all $x_1, x_2 \in \mathbb{R}^n$.

The standard definition of a differentiable and strongly convex function $f : \mathbb{R}^n \to \mathbb{R}$ is that it satisfies

$$f(x_1) \geq f(x_2) + \langle \nabla f(x_2), x_1 - x_2\rangle + \frac{\sigma}{2}\|x_1 - x_2\|_2^2 \quad (8)$$

for any $x_1, x_2 \in \mathbb{R}^n$, where the modulus $\sigma \in \mathbb{R}_{++}$ describes a lower bound of the curvature of the function. In this paper, the definition (8) is generalized to allow for a quadratic lower bound with different curvature in different directions.

Definition 3. A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is *strongly convex with matrix H* if and only if

$$f(x_1) \geq f(x_2) + \langle \nabla f(x_2), x_1 - x_2\rangle + \frac{1}{2}\|x_1 - x_2\|_H^2$$

holds for all $x_1, x_2 \in \mathbb{R}^n$, where $H \in \mathbb{S}^n_{++}$.

Remark 4. The traditional definition of strong convexity (8) is obtained from Definition 3 by setting $H = \sigma I$.

Lemma 5. Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable and strongly convex with matrix H. The condition that

$$f(x_1) \geq f(x_2) + \langle \nabla f(x_2), x_1 - x_2 \rangle + \frac{1}{2}\|x_1 - x_2\|_H^2 \quad (9)$$

holds for all $x_1, x_2 \in \mathbb{R}^n$ is equivalent to that

$$\langle \nabla f(x_1) - \nabla f(x_2), x_1 - x_2 \rangle \geq \|x_1 - x_2\|_H^2 \quad (10)$$

holds for all $x_1, x_2 \in \mathbb{R}^n$.

The condition (9) is a quadratic lower bound on the function value, while the condition (3) is a quadratic upper bound on the function value. These two properties are linked through the conjugate function

$$f^\star(y) \triangleq \sup_x \left\{ y^T x - f(x) \right\}.$$

More precisely, we have the following result.

Proposition 6. Assume that $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is closed, proper, and strongly convex with modulus σ on the relative interior of its domain. Then the conjugate function $f^\star$ is convex and differentiable, and $\nabla f^\star(y) = x^\star(y)$, where $x^\star(y) = \arg\max_x \left\{ y^T x - f(x) \right\}$. Further, $\nabla f^\star$ is Lipschitz continuous with constant $L = \frac{1}{\sigma}$.

A straight-forward generalization is given by the chain-rule and was proven in (Nesterov, 2005, Theorem 1) (which also proves the less general Proposition 6).

Corollary 7. Assume that $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is closed, proper, and strongly convex with modulus σ on the relative interior of its domain. Further, define $g^\star(y) \triangleq f^\star(Ay)$. Then $g^\star$ is convex and differentiable, and $\nabla g^\star(y) = A^T x^\star(Ay)$, where $x^\star(Ay) = \arg\max_x \left\{ (Ay)^T x - f(x) \right\}$. Further, $\nabla g^\star$ is Lipschitz continuous with constant $L = \frac{\|A\|_2^2}{\sigma}$.

For the case when $f(x) = \frac{1}{2}x^T H x + g^T x$, i.e. f is a quadratic, a tighter Lipschitz constant to $\nabla g^\star(y) = \nabla f^\star(Ay)$ was provided in (Richter et al., 2013, Theorem 7), namely $L = \|AH^{-1}A^T\|_2$.

3. PROBLEM FORMULATION

We consider optimization problems of the form

$$\begin{array}{ll} \text{minimize} & f(x) + h(x) \\ \text{subject to} & Ax = b \end{array} \quad (11)$$

where the decision variables are partitioned as $x = (x_1, \ldots, x_M) \in \mathbb{R}^n$ where $x_i \in \mathbb{R}^{n_i}$, the cost functions are separable, i.e. $f(x) = \sum_{i=1}^M f_i(x_i)$ and $h(x) = \sum_{i=1}^M h_i(x_i)$, the matrices $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ are decomposed as

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1M} \\ \vdots & \ddots & \vdots \\ A_{M1} & \cdots & A_{MM} \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ \vdots \\ b_M \end{bmatrix}$$

where $A_{ij} \in \mathbb{R}^{m_i \times n_j}$ for all $i \in \{1, \ldots, M\}$ and $j \in \{1, \ldots, M\}$ and $b_i \in \mathbb{R}^{m_i}$ for all $i \in \{1, \ldots, M\}$. We assume that for all $i \in \{1, \ldots, M\}$, $A_{ij} = 0$ for some $j \in \{1, \ldots, M\}$, i.e., that the A-matrix is block sparse. The sparsity structure induced by this assumption is represented by the sets $\mathcal{N}_i$ and $\mathcal{M}_i$, where $\mathcal{N}_i$ contains indices for non-zero blocks of block row i and $\mathcal{M}_i$ contains indices for non-zero blocks of block column i. More precisely, we have

$$\mathcal{N}_i = \{j \in \{1, \ldots, M\} \mid A_{ij} \neq 0\},$$
$$\mathcal{M}_i = \{j \in \{1, \ldots, M\} \mid A_{ji} \neq 0\}.$$

We also introduce concatenated matrices $A_{\mathcal{N}_i} \in \mathbb{R}^{m_i \times n_{\mathcal{N}_i}}$, where $n_{\mathcal{N}_i} = \sum_{j \in \mathcal{N}_i} n_j$, that contain all non-zero submatrices A_{ij}, e.g., if $\mathcal{N}_1 = \{1, 2, 6\}$ then $A_{\mathcal{N}_1} = [A_{11}\ A_{12}\ A_{16}]$. Similarly, we introduce $A_{\mathcal{M}_i} \in \mathbb{R}^{m_{\mathcal{M}_i} \times n_i}$, where $m_{\mathcal{M}_i} = \sum_{j \in \mathcal{M}_i} m_j$; if $\mathcal{M}_1 = \{1, 4, 6\}$, then $A_{\mathcal{M}_1} = [A_{11}^T\ A_{41}^T\ A_{61}^T]^T$. We also introduce consistent notation for the variables, namely $x_{\mathcal{N}_i} \in \mathbb{R}^{n_{\mathcal{N}_i}}$, i.e. $x_{\mathcal{N}_1} = (x_1, , x_2, x_6)$ in the above example. This implies that $\sum_{j \in \mathcal{N}_i} A_{ij} x_j = A_{\mathcal{N}_i} x_{\mathcal{N}_i}$.

These assumptions and the introduced notation imply that the optimization problem (11) can equivalently be written

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i=1}^M \{f_i(x_i) + h_i(x_i)\} & \\ \text{subject to} & A_{\mathcal{N}_i} x_{\mathcal{N}_i} = b_i, & i = \{1, \ldots, M\}. \end{array} \quad (12)$$

Throughout this paper we assume the following.

Assumption 8.

(a) The functions $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$ are strongly convex with matrix $H_i \in \mathbb{S}^{n_i}_{++}$.
(b) The functions $h_i : \mathbb{R}^{n_i} \to \mathbb{R} \cup \{\infty\}$ are proper, closed, and convex.
(c) The matrix $A \in \mathbb{R}^{m \times n}$ has full row rank.

Remark 9. Assumption 8(a) implies that $f = \sum_{i=1}^M f_i$ is strongly convex with matrix H, where

$$H := \text{blkdiag}(H_1, \ldots, H_M). \quad (13)$$

Assumption 8(b) is satisfied if, e.g., h_i are indicator functions to convex constraint sets. If Assumption 8(c) is not satisfied, redundant equality constraints can, without affecting the solution of (11), be removed to satisfy the assumption.

To form the dual problem, we introduce dual variables $\lambda = (\lambda_1, \ldots, \lambda_M) \in \mathbb{R}^m$, where $\lambda_i \in \mathbb{R}^{m_i}$. We also introduce a notation for dual variables that correspond to the concatenated matrices $A_{\mathcal{M}_i}$, namely $\lambda_{\mathcal{M}_i} \in \mathbb{R}^{m_{\mathcal{M}_i}}$. In the above example with $\mathcal{M}_1 = \{1, 4, 6\}$ we have $\lambda_{\mathcal{M}_1} = (\lambda_1, \lambda_4, \lambda_6)$. This gives the following Lagrange dual problem

$$\begin{aligned} &\sup_\lambda \inf_x \left\{ f(x) + h(x) + \lambda^T (Ax - b) \right\} \\ &= \sup_\lambda \inf_x \sum_{i=1}^M \left\{ f_i(x_i) + h_i(x_i) + \lambda_i^T \left(A_{\mathcal{N}_i} x_{\mathcal{N}_i} - b_i \right) \right\} \\ &= \sup_\lambda \sum_{i=1}^M \left[\inf_{x_i} \left\{ f_i(x_i) + h_i(x_i) + x_i^T A_{\mathcal{M}_i}^T \lambda_{\mathcal{M}_i} \right\} - \lambda_i^T b_i \right] \\ &= \sup_\lambda \sum_{i=1}^M \left\{ -F_i^\star(-A_{\mathcal{M}_i}^T \lambda_{\mathcal{M}_i}) - \lambda_i^T b_i \right\} \end{aligned} \quad (14)$$

where $F_i^\star$, $i = \{1, \ldots, M\}$, are conjugate functions to $F_i := f_i + h_i$. We get from (14) that the conjugate function $F^\star$ to $F = g + h$ is given by $F^\star = \sum_{i=1}^M F_i^\star$. To evaluate $F^\star$ or $F_i^\star$, a minimization problem is solved. We denote

the minimands to these optimization problems by $x^\star(\lambda)$ and $x_i^\star(\lambda_{\mathcal{M}_i})$ respectively and they are given by

$$x^\star(\lambda) = \arg\min_x \left\{ f(x) + h(x) + \lambda^T A x \right\}, \quad (15)$$

$$x_i^\star(\lambda_{\mathcal{M}}) = \arg\min_{x_i} \left\{ f_i(x_i) + h_i(x_i) + \lambda_{\mathcal{M}_i}^T A_{\mathcal{M}_i} x_i \right\}. \quad (16)$$

We define the local dual functions to be

$$d_i(\lambda_{\mathcal{M}_i}) := -F_i^\star(-A_{\mathcal{M}_i}^T \lambda_{\mathcal{M}_i}) - b_i^T \lambda_i \quad (17)$$

and the dual function to be

$$\begin{aligned} d(\lambda) &:= \min_x \left\{ f(x) + h(x) + \lambda^T (Ax - b) \right\} \quad (18) \\ &= -F^\star(-A^T\lambda) - b^T\lambda \end{aligned}$$

where $d = \sum_{i=1}^M d_i$. From Corollary 7 we have that d_i and d are differentiable with gradients

$$\begin{aligned} \nabla d_i(\lambda_{\mathcal{M}_i}) &= A_{\mathcal{M}_i} x_i^\star(\lambda_{\mathcal{M}_i}) - \hat{b}_i \\ \nabla d(\lambda) &= A x^\star(\lambda) - b. \end{aligned}$$

respectively, where $\hat{b}_i = (0, \ldots, 0, b_i, 0, \ldots, 0)$. Further, differentiation of the dual function w.r.t. λ_i is given by

$$\nabla_{\lambda_i} d(\lambda) = A_{\mathcal{N}_i} x_{\mathcal{N}_i}^\star(\lambda_i) - b_i.$$

Corollary 7 further implies that the gradients to d_i and d are Lipschitz continuous with constants $L_i = \|A_{\mathcal{M}_i}\|_2^2/\lambda_{\min}(H_i)$ and $L = \|A\|_2^2/\lambda_{\min}(H)$ respectively. As previously discussed, this is equivalent to the existence of a quadratic lower bound given by (4) to the concave dual function, with curvature L_i and L respectively. In the following section we will show that the dual function (18) and local dual functions (17) satisfy the following tighter lower bounds

$$d(\lambda_1) \geq d(\lambda_2) + \langle \nabla d(\lambda_2), \lambda_1 - \lambda_2 \rangle - \tfrac{1}{2}\|\lambda_1 - \lambda_2\|^2_{AH^{-1}A^T} \quad (19)$$

for all $\lambda_1, \lambda_2 \in \mathbb{R}^m$ and

$$\begin{aligned} d_i(\lambda_{\mathcal{M}_i}^1) \geq d_i(\lambda_{\mathcal{M}_i}^2) + \langle \nabla d_i(\lambda_{\mathcal{M}_i}^2), \lambda_{\mathcal{M}_i}^1 - \lambda_{\mathcal{M}_i}^2 \rangle - \\ - \tfrac{1}{2}\|\lambda_{\mathcal{M}_i}^1 - \lambda_{\mathcal{M}_i}^2\|_{A_{\mathcal{M}_i} H_i^{-1} A_{\mathcal{M}_i}^T} \quad (20) \end{aligned}$$

for all $\lambda_{\mathcal{M}_i}^1, \lambda_{\mathcal{M}_i}^2 \in \mathbb{R}^{m_{\mathcal{M}_i}}$ respectively.

4. DUAL FUNCTION PROPERTIES

The following results show that the dual and local dual functions satisfy (19) and (20) respectively. The results in this section are proven in the longer version of this paper, Giselsson (2014).

Theorem 10. Suppose that Assumption 8 holds and that f is strongly convex with matrix $H \in \mathbb{S}_{++}^n$. The dual function d defined in (18) is concave, differentiable and satisfies

$$d(\lambda_1) \geq d(\lambda_2) + \langle \nabla d(\lambda_2), \lambda_1 - \lambda_2 \rangle - \tfrac{1}{2}\|\lambda_1 - \lambda_2\|^2_{\mathbf{L}} \quad (21)$$

for every $\lambda_1, \lambda_2 \in \mathbb{R}^m$ and any $\mathbf{L} \in \mathbb{S}_+^m$ such that $\mathbf{L} \succeq AH^{-1}A^T$.

Corollary 11. The local dual functions d_i defined in (17) are concave, differentiable and satisfy

$$\begin{aligned} d_i(\lambda_{\mathcal{M}_i}^1) \geq d_i(\lambda_{\mathcal{M}_i}^2) + \langle \nabla d_i(\lambda_{\mathcal{M}_i}^2), \lambda_{\mathcal{M}_i}^1 - \lambda_{\mathcal{M}_i}^2 \rangle - \\ - \tfrac{1}{2}\|\lambda_{\mathcal{M}_i}^1 - \lambda_{\mathcal{M}_i}^2\|^2_{\mathbf{L}_{\mathcal{M}_i}} \end{aligned}$$

for all $\lambda_{\mathcal{M}_i}^1, \lambda_{\mathcal{M}_i}^2 \in \mathbb{R}^{m_{\mathcal{M}_i}}$ and any $\mathbf{L}_{\mathcal{M}_i} \in \mathbb{S}_{++}^{m_{\mathcal{M}_i}}$ such that $\mathbf{L}_{\mathcal{M}_i} \succeq A_{\mathcal{M}_i} H_i^{-1} A_{\mathcal{M}_i}^T$.

Next, we show that if f is a strongly convex quadratic function and h satisfies certain conditions, then Theorem 10 gives the best possible bound of the form (21).

Proposition 12. Assume that $f(x) = \frac{1}{2}x^T H x + \zeta^T x$ with $H \in \mathbb{S}_{++}^n$ and $\zeta \in \mathbb{R}^n$ and that there exists a set $\mathcal{X} \subseteq \mathbb{R}^n$ with non-empty interior on which h (besides being proper, closed, and convex) is linear, i.e. $h(x) = \xi_{\mathcal{X}}^T x + \theta_{\mathcal{X}}$ for all $x \in \mathcal{X}$. Further, assume that there exists $\widetilde{\nu}$ such that $x^\star(\widetilde{\nu}) \in \operatorname{int}(\mathcal{X})$. Then for any matrix $\mathbf{L} \not\succeq CH^{-1}C^T$, there exist ν_1 and ν_2 such that (21) does not hold.

Proposition 12 shows that the bound in Theorem 10 is indeed the best obtainable bound of the form (21) if f is a quadratic and h specifies the stated assumptions. Examples of functions that satisfy the assumptions on h in Proposition 12 include linear functions, indicator functions of closed convex constraint sets with non-empty interior, and the 1-norm.

Theorem 10 and Corollary 11, provide a tighter quadratic lower bound to the dual function compared to what has previously been presented in the literature, i.e. compared to Proposition 6 and Corollary 7. These results can be exploited to construct more efficient distributed algorithms.

5. DISTRIBUTED OPTIMIZATION ALGORITHM

Two main disadvantages are associated with standard dual decomposition using gradient methods or fast gradient methods. The first is the often slow convergence, especially if the standard gradient method is used, and the second is that global information is needed to compute the optimal step-size. In this section, we will describe generalized fast gradient methods. These methods together with the results on the tighter quadratic lower bound to the dual function presented in the previous section, enables for significantly improved convergence properties compared to standard dual decomposition. This is a remedy for the first issue. For the second issue, we present a fully distributed initialization procedure of the algorithm that computes the algorithm parameters.

Generalized fast gradient methods are applicable to solve problems of the form

$$\text{minimize } g(x)$$

where $x \in \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable and satisfies

$$g(x_1) \leq g(x_2) + \langle \nabla g(x_2), x_1 - x_2 \rangle + \tfrac{1}{2}\|x_1 - x_2\|^2_{\mathbf{L}} \quad (22)$$

for all $x_1, x_2 \in \mathbb{R}^n$ and some $\mathbf{L} \in \mathbb{S}_{++}^n$. The main step of the algorithm is to minimize the r.h.s. of (22), i.e. to take a gradient step, since

$$\begin{aligned} &\arg\min_x \left\{ g(y) + \langle \nabla g(y), x - y \rangle + \tfrac{1}{2}\|x - y\|^2_{\mathbf{L}} \right\} \\ &= y - \mathbf{L}^{-1}\nabla g(y). \end{aligned}$$

The algorithm is stated next, see Zuo and Lin (2011).

Algorithm 1. **Generalized fast gradient method**

Set: $y^1 = x^0 \in \mathbb{R}^n, t^1 = 1$
For $k \geq 1$

$$\begin{aligned} x^k &= y^k - \mathbf{L}^{-1}\nabla g(y^k) \\ t^{k+1} &= \tfrac{1+\sqrt{1+4(t^k)^2}}{2} \\ y^{k+1} &= x^k + \left(\tfrac{t^k - 1}{t^{k+1}}\right)(x^k - x^{k-1}) \end{aligned}$$

Since the right hand side of (22) is minimized in every step of the algorithm, it serves as an approximation of the function g to be minimized. By restricting $\mathbf{L} = LI$ (where $L \in \mathbb{R}_{++}$ is a Lipschitz constant to the gradient) in the approximation (22) we get the standard fast gradient method, see e.g. Beck and Teboulle (2009). Using $\mathbf{L} = LI$, the r.h.s. of (22) has the same curvature in all directions given by L. If g is not well approximated by this approximation, slow convergence is expected. By choosing a matrix $\mathbf{L}$ that better captures the shape of the function to be minimized, a tighter characterization of g can be obtained, and a faster convergence of the algorithm is expected. The convergence rate of Algorithm 1 is (see Zuo and Lin (2011))

$$g(x^k) - g(x^\star) \leq \frac{2\|x^\star - x^0\|_{\mathbf{L}}^2}{(k+1)^2}. \tag{23}$$

The convergence rate of the standard fast gradient method is given by setting $\mathbf{L} = LI$ in (23), see Beck and Teboulle (2009).

From Theorem 10 we know that the dual function (18) satisfies the properties required to apply the generalized fast gradient method, i.e. that the negative dual function satisfies (22). This implies that Algorithm 1, when applied to solve the dual problem (14), converges for any $\mathbf{L} \in \mathbb{S}_{++}^m$ that satisfies $\mathbf{L} \succeq AH^{-1}A^T$. Since d is concave, and $\nabla d(\lambda) = Ax^\star(\lambda) - b$, Algorithm 1 applied to solve the dual problem (14) becomes

Algorithm 2.
Generalized fast dual gradient method

Set: $z^1 = \lambda^0 \in \mathbb{R}^m, t^1 = 1$
For $k \geq 1$

$$x^k = \arg\min_x \left\{ f(x) + h(x) + (z^k)^T(Ax - b) \right\}$$
$$\lambda^k = z^k + \mathbf{L}^{-1}(Ax^k - b)$$
$$t^{k+1} = \frac{1+\sqrt{1+4(t^k)^2}}{2}$$
$$z^{k+1} = \lambda^k + \left(\frac{t^k - 1}{t^{k+1}}\right)(\lambda^k - \lambda^{k-1})$$

When solving separable problems of the form (12), Algorithm 2 can be implemented in distributed fashion by restricting $\mathbf{L} \in \mathbb{S}_{++}^m$ to be block diagonal, i.e. of the form $\mathbf{L} = \text{blkdiag}(\mathbf{L}_1, \ldots, \mathbf{L}_M)$ where $\mathbf{L}_i \in \mathbb{S}_{++}^{m_i}$. The distributed implementation is presented next.

Algorithm 3.
Distributed generalized fast dual gradient method

Initialize $z_i^1 = \lambda_i^0 \in \mathbb{R}^{m_i}, t^1 = 1$.
In every node, $i = \{1, \ldots, M\}$, do the following steps
For $k \geq 1$

(1) Send z_i^k to each $j \in \mathcal{N}_i$, receive z_j^k from each $j \in \mathcal{M}_i$
(2) Form $z_{\mathcal{M}_i}^k = (\ldots, z_j^k, \ldots)$ with all $j \in \mathcal{M}_i$
(3) Update local primal variables according to
$$x_i^k = \arg\min_x \left\{ f_i(x) + h_i(x) + x_i^T A_{\mathcal{M}_i}^T z_{\mathcal{M}_i}^k \right\}$$
(4) Send x_i^k to each $j \in \mathcal{M}_i$, receive x_j^k from each $j \in \mathcal{N}_i$
(5) Form $x_{\mathcal{N}_i}^k = (\ldots, x_j^k, \ldots)$ with all $j \in \mathcal{N}_i$
(6) Update local dual variables according to

$$\lambda_i^k = z_i^k + \mathbf{L}_i^{-1}(A_{\mathcal{N}_i} x_{\mathcal{N}_i}^k - b_i)$$
$$t^{k+1} = \frac{1+\sqrt{1+4(t^k)^2}}{2}$$
$$z_i^{k+1} = \lambda_i^k + \left(\frac{t^k - 1}{t^{k+1}}\right)(\lambda_i^k - \lambda_i^{k-1})$$

In the following proposition we state the convergence rate properties of Algorithm 3.

Proposition 13. Suppose that Assumption 8 holds. If $\mathbf{L} = \text{blkdiag}(\mathbf{L}_1, \ldots, \mathbf{L}_M) \in \mathbb{S}_{++}^m$ is chosen such that $\mathbf{L} \succeq AH^{-1}A^T$. Then Algorithm 3 converges with the rate

$$d(\lambda^\star) - d(\lambda^k) \leq \frac{2\left\|\lambda^\star - \lambda^0\right\|_{\mathbf{L}}^2}{(k+1)^2}, \forall k \geq 1 \tag{24}$$

where k is the iteration number, when solving problems of the form (12).

Remark 14. By forming a specific running average of previous primal variables, it is possible to prove a $O(1/k)$ convergence rate for the distance to the primal variable optimum and a $O(1/k^2)$ convergence rate for the worst case primal infeasibility, see Patrinos and Bemporad (2014).

Remark 15. Due to error accumulation of the fast gradient method, see Devolder et al. (2013), the inner minimizations, i.e. the x_i^k-updates, should be solved to high accuracy.

A remaining issue with the distributed Algorithm 3 is how to compute the $\mathbf{L}$-matrix and how to distribute these computations. This is the topic of the following section.

5.1 Distributed computation of the $\mathbf{L}$*-matrix*

The (optimal) step-size selection in standard fast dual gradient methods relies on computing a (tight) Lipschitz constant to the dual gradient. This Lipschitz constant is usually computed by taking the Euclidean operator norm of the equality constraint matrix A (see Corollary 7). This requires centralized computations. In this section we will extend a recent result in Beck et al. (2014) to allow for distributed selection of the $\mathbf{L}$-matrix that is used in the algorithm.

The requirements on the $\mathbf{L}$-matrix are that it should be block diagonal, i.e. $\mathbf{L} = \text{blkdiag}(\mathbf{L}_1, \ldots, \mathbf{L}_M)$ to facilitate a distributed implementation, and that it should satisfy $\mathbf{L} \succeq AH^{-1}A^T$ to guarantee convergence of the algorithm. We will see that Corollary 11 can be used to compute a matrix $\mathbf{L}$ that satisfies these requirements, using local computations and neighboring communication only. From Corollary 11 we have that any matrix $\mathbf{L}_{\mathcal{M}_i} \in \mathbb{S}_{++}^{m_{\mathcal{M}_i}}$ that describe a quadratic upper bound to the local dual functions d_i must satisfy $\mathbf{L}_{\mathcal{M}_i} \succeq A_{\mathcal{M}_i} H_i^{-1} A_{\mathcal{M}_i}^T$. To allow for a distributed implementation, we further restrict $\mathbf{L}_{\mathcal{M}_i}$ to be block-diagonal, i.e. if $\mathcal{M}_1 = \{1, 4, 6\}$ then $\mathbf{L}_{\mathcal{M}_1} = \text{blkdiag}(\mathbf{L}_{\mathcal{M}_1,1}, \mathbf{L}_{\mathcal{M}_1,4}, \mathbf{L}_{\mathcal{M}_1,6})$ where $\mathbf{L}_{\mathcal{M}_i,j} \in \mathbb{S}_{++}^{m_j}$. These restrictions on the local matrices $\mathbf{L}_{\mathcal{M}_i}$ are summarized in the following set notation

$$\mathcal{L}_{\mathcal{M}_i} = \big\{ \mathbf{L}_{\mathcal{M}_i} \in \mathbb{S}_{++}^{m_{\mathcal{M}_i}} \mid \mathbf{L}_{\mathcal{M}_i} \succeq A_{\mathcal{M}_i} H_i^{-1} A_{\mathcal{M}_i}^T, \; \mathbf{L}_{\mathcal{M}_i} = \text{blkdiag}(\ldots, \mathbf{L}_{\mathcal{M}_i,j}, \ldots) \text{ with all } j \in \mathcal{M}_i, \mathbf{L}_{\mathcal{M}_i,j} \in \mathbb{S}_{++}^{m_j} \big\}.$$

Using this set notation, we propose the following distributed initialization procedure for Algorithm 3.

Algorithm 4.
Distributed initialization of Algorithm 3

For each $i \in \{1, \dots, M\}$
Do

(1) Choose $\mathbf{L}_{\mathcal{M}_i} = \text{blkdiag}(\dots, \mathbf{L}_{\mathcal{M}_i,j}, \dots) \in \mathcal{L}_{\mathcal{M}_i}$
(2) Send $\mathbf{L}_{\mathcal{M}_i,j}$ to all $j \in \mathcal{M}_i$
Receive $\mathbf{L}_{\mathcal{M}_j,i}$ from all $j \in \mathcal{N}_i$
(3) Compute $\mathbf{L}_i = \sum_{j \in \mathcal{N}_i} \mathbf{L}_{\mathcal{M}_j,i}$

From this initialization we get local $\mathbf{L}_i$-matrices that are used in each local node i and in all iterations of Algorithm 3. In the following proposition we show that Algorithm 3 converges with the rate (24) when initialized using Algorithm 4.

Proposition 16. Suppose that Assumption 8 holds. If $\mathbf{L}_i \in \mathbb{S}^{m_i}_{++}$ is computed using Algorithm 4. Then Algorithm 3 converges with the rate (24) when solving problems of the form (12).

The first step in the distributed initialization algorithm is still not completely specified, i.e., we have not yet discussed how to choose $\mathbf{L}_{\mathcal{M}_i}$. In the following section we will discuss how to choose $\mathbf{L}_{\mathcal{M}_i}$ in context of distributed model predictive control.

6. DISTRIBUTED MODEL PREDICTIVE CONTROL

Distributed model predictive control (DMPC) is a distributed optimization-based control scheme applied to control systems consisting of several subsystems that have a sparse dynamic interaction structure. The local dynamics are described by

$$x_i(t+1) = \sum_{j \in \mathcal{N}_i} \Phi_{ij} x_j(t) + \Gamma_{ij} u_j(t), \qquad x_i(0) = \bar{x}_i$$

for all $i \in \{1, \dots, M\}$, where $x_i \in \mathbb{R}^{n_{x_i}}$, $u_i \in \mathbb{R}^{n_{u_i}}$, $\Phi_{ij} \in \mathbb{R}^{n_{x_i} \times n_{x_j}}$, $\Gamma_{ij} \in \mathbb{R}^{n_{x_i} \times n_{u_j}}$, and $\bar{x}_i \in \mathbb{R}^{n_{x_i}}$ is a measurement of the current state. In DMPC, it is common to have local state and control constraint sets $x_i \in \mathcal{X}_i$, $u_i \in \mathcal{U}_i$, where $\mathcal{X}_i$ and $\mathcal{U}_i$ are non-empty, closed, and convex sets. The cost function is usually chosen as the following sum over a horizon N

$$\sum_{i=1}^{M} \left(\sum_{t=0}^{N-1} \frac{1}{2} \begin{bmatrix} x_i(t) \\ u_i(t) \end{bmatrix}^T \begin{bmatrix} Q_i & 0 \\ 0 & R_i \end{bmatrix} \begin{bmatrix} x_i(t) \\ u_i(t) \end{bmatrix} \right) + \frac{1}{2} \|x_i(N)\|^2_{Q_{i,f}}$$

where $Q_i \in \mathbb{S}^{n_{x_i}}_{++}$, $R_i \in \mathbb{S}^{n_{u_i}}_{++}$, and $Q_{i,f} \in \mathbb{S}^{n_{x_i}}_{++}$. By stacking the local state and control vectors into $y_i = [x_i(0)^T, \dots, x_i(N)^T, u_i(0)^T, \dots, u_i(N-1)^T]^T$ we get an optimization problem of the form

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{M} \{f_i(y_i) + h_i(y_i)\} \\ \text{subject to} \quad & \sum_{j \in \mathcal{N}_i} A_{ij} y_j = b_i \bar{x}_i \end{aligned} \tag{25}$$

where $f_i(y_i) = \frac{1}{2} y_i^T H_i y_i$, $h_i(y_i) = I_{\mathcal{Y}_i}(y_i)$, and H_i, $\mathcal{Y}_i$, A_{ij}, and b_i are structured according to the stacked vector y_i. The optimization problem (25) is structured as (12) and can therefore be solved in distributed fashion using Algorithm 3. In DMPC, the optimization problem (25) is solved repeatedly in distributed fashion and the first control action $u_i(0)$ is applied to the plant in every subsystem $i \in \{1, \dots, M\}$ after each optimization. Since many similar optimization problems are solved repeatedly online, much offline computational effort can be devoted to ease the online computational burden. In this case, the offline computational effort is devoted to run the distributed initialization in Algorithm 4. The first step of Algorithm 4 does not state how to choose the $\mathbf{L}_i$ matrices, only that they should be chosen. In the DMPC context, we propose to solve the following local optimization problem in step 1 and for each $i \in \{1, \dots, M\}$:

$$\begin{aligned} \text{minimize} \quad & \text{tr } \mathbf{L}_{\mathcal{M}_i} \\ \text{subject to} \quad & \mathbf{L}_{\mathcal{M}_i} = \text{blkdiag}(\dots, \mathbf{L}_{\mathcal{M}_i,j}, \dots) \in \mathcal{L}_{\mathcal{M}_i}. \end{aligned} \tag{26}$$

This is a convex semi-definite program (SDP) that can readily be solved using standard software.

Due to the distributed structure of the initialization procedure, the DMPC scheme enjoys distributed reconfiguration, commonly referred to as *plug-and-play*. Distributed reconfiguration or plug-and-play refers to the feature that if an additional subsystem is connected to (or removed from) the system, the only updates needed in the algorithm involve computations in the direct neighborhood of the added (removed) subsystem. This is the case for Algorithm 4 since if a reconfiguration is needed due to addition or removal of subsystem i, only subsystems $j \in \mathcal{M}_i$ need to be invoked for the reconfiguration.

7. NUMERICAL EXAMPLE

The proposed algorithm is evaluated by applying it to a system consisting of 100 subsystems that have a sparse dynamic interaction. The dynamic interaction structure is decided using the method in (Kraning et al., 2013 §6.1). The number of states in each subsystem is randomly chosen from the interval $\{10, 11, \dots, 20\}$ and the number of inputs are three or four. The control horizon is chosen to be $N = 10$ and the total number of decision variables in the DMPC optimization problem is 18020. The entries of the dynamics and input matrices are randomly chosen from the intervals $[-0.7\ 1.3]$ and $[-1\ 1]$ respectively. Then the dynamics matrix is re-scaled to get a spectral radius of 1.2. The states and inputs are upper and lower bounded by random bounds generated from the intervals $[0.2\ 1.2]$ and $[-1.2\ -0.2]$ respectively. The state and input cost matrices are diagonal and each diagonal entry is randomly chosen from the interval $[1\ 1000]$.

The proposed algorithm is evaluated by comparing it to fast dual decomposition and standard dual decomposition. Fast dual decomposition is achieved by setting $\mathbf{L}_i = \|AH^{-1}A^T\|_2 I$ for all i in Algorithm 3, where A and H are the global equality constraint and cost matrices respectively. This choice of $\mathbf{L}_i$ is optimal if restricted to being a multiple of the identity matrix, and if all $\mathbf{L}_i$ are restricted to be equal (as in fast dual decomposition). By standard dual decomposition, we refer to dual decomposition with standard gradient steps. This is achieved by using the optimal step size, i.e. $\mathbf{L}_i = \|AH^{-1}A^T\|_2 I$ in Algorithm 3, and letting the last line of point 6) in the algorithm be $z_i^{k+1} = \lambda_i^k$. This removes the Nesterov acceleration from the method and results in a standard dual gradient method, i.e. the standard dual decomposition. In the

Table 1. Numerical comparison of Algorithm 3 and fast and standard dual decomposition.

	nbr iters	
Algorithm	avg.	max
Algorithm 3 initialized using Alg. 4	133.7	200
fast dual decomposition	788.5	1309
standard dual decomposition	45784	105651

numerical evaluation, the proposed distributed algorithm, i.e. Algorithm 3, is initialized using the initialization in Algorithm 4. In step 1) of this initialization procedure, the optimization problem (26) is solved in each node i.

The evaluation in Table 1 is obtained by generating 1000 random initial conditions from the state constraint set and solving the corresponding optimal control problems using the three different methods. Table 1 reports the average and max number of iterations required to solve these problems.

The numerical evaluation reveals that the introduction of acceleration significantly improves the convergence of dual decomposition, i.e. fast dual decomposition significantly outperforms standard dual decomposition. The distributed algorithm proposed in this paper, i.e. Algorithm 3, further reduces the communication requirement in dual decomposition with more than a factor of five compared to fast dual decomposition.

8. CONCLUSIONS

We have proposed a generalization of fast dual decomposition. In this generalization, a quadratic upper bound to the negative dual function with different curvature in different directions is minimized in each step in the algorithm. This differs from traditional dual decomposition methods where the main step is to minimize a quadratic upper bound to the negative dual function that has the same curvature in all directions. This generalization is made possible by the main contribution of this paper that characterizes the set of matrices that can be used to describe this quadratic upper bound. Further, we show that the algorithm can be initialized and reconfigured using distributed computations only. This is traditionally not the case in dual decomposition where the norm of a matrix that involve variables from all subsystems is used to compute the optimal step size. For the example considered in this paper, the proposed algorithm converges more than five times faster than fast dual decomposition and more than 300 times faster than standard dual decomposition.

REFERENCES

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, October 2009.

A. Beck, A. Nedic, A. Ozdaglar, and M. Teboulle. Optimal distributed gradient methods for network resource allocation problems. *IEEE Transactions on Control of Network Systems*, 2014. To appear.

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

G. B. Danzig and P. Wolfe. The decomposition algorithm for linear programming. *Econometrica*, 4:767–778, 1961.

O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, pages 1–39, 2013.

M. D. Doan, T. Keviczky, and B. De Schutter. An iterative scheme for distributed model predictive control using Fenchel's duality. *Journal of Process Control*, 21(5): 746–755, June 2011. Special Issue on Hierarchical and Distributed Model Predictive Control.

H. Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11:399–417, 1963.

P. Giselsson. A generalized distributed accelerated gradient method for DMPC with iteration complexity bounds. In *Proceedings of 2013 American Control Conference*, pages 327–333, Washington D.C., June 2013.

P. Giselsson. Improving fast dual ascent for MPC - Part I: The distributed case. *Automatica*, 2014. Submitted. Available `http://arxiv.org/abs/1312.3012`.

P. Giselsson, M. D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, 49(3):829–833, 2013.

M. Kraning, E. Chu, J. Lavaei, and S. Boyd. Dynamic network energy management via proximal message passing. *Foundations and Trends in Optimization*, 1(2):70–122, 2013.

R. R. Negenborn. *Multi-Agent Model Predictive Control with Applications to Power Networks*. PhD thesis, TU Delft, 2007.

Y. Nesterov. A method of solving a convex programming problem with convergence rate O $(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Netherlands, 1st edition, 2003. ISBN 1402075537.

Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, May 2005.

P. Patrinos and A. Bemporad. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Transactions on Automatic Control*, 59(1):18–33, 2014.

S. Richter, C. N. Jones, and M. Morari. Certification aspects of the fast gradient method for solving the dual of parametric convex programs. *Mathematical Methods of Operations Research*, 77(3):305–321, 2013.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. Technical report. Available: `http://www.csie.ntu.edu.tw/~b97058/tseng/papers/apgm.pdf`, May 2008.

W. Zuo and Z. Lin. A generalized accelerated proximal gradient approach for total-variation-based image restoration. *IEEE Transactions on Image Processing*, 20(10):2748–2759, October 2011.