EFIX: Exact Fixed Point Methods for Distributed Optimization

Dušan Jakovetić * Nataša Krejić *
Nataša Krklec Jerinkić *

July 29, 2022

Abstract

We consider strongly convex distributed consensus optimization over connected networks. EFIX, the proposed method, is derived using quadratic penalty approach. In more detail, we use the standard reformulation – transforming the original problem into a constrained problem in a higher dimensional space – to define a sequence of suitable quadratic penalty subproblems with increasing penalty parameters. For quadratic objectives, the corresponding sequence consists of quadratic penalty subproblems. For generic strongly convex case, the objective function is approximated with a quadratic model and hence the sequence of the resulting penalty subproblems is again quadratic. EFIX is then derived by solving each of the quadratic penalty subproblems via a fixed point (R)-linear solver, e.g., Jacobi Over-Relaxation method. The exact convergence is proved as well as the worst case complexity of order $\mathcal{O}(\epsilon^{-1})$ for the quadratic case. In the case of strongly convex generic functions, the standard result for penalty methods is obtained. Numerical results indicate that the method is highly competitive with state-of-the-art exact first order methods, requires smaller computational and communication effort, and is robust to the choice of algorithm parameters.

Key words: Fixed point methods, quadratic penalty method, distributed optimization, strongly convex problems

1 Introduction

We consider problems of the form

$$\min_{y\in\mathbb{R}^n} f(y) = \sum_{i=1}^{N} f_i(y), \tag{1}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ are strongly convex local cost functions. A decentralized optimization framework is considered, more precisely, we assume decentralized but connected network of N nodes.

Distributed consensus optimization over networks has become a mainstream research topic, e.g., [2, 3, 4, 6, 9, 11], motivated by numerous applications in signal processing [13], control [16], Big Data analytics [23], social networks [1], etc. Various methods have been proposed in the literature, e.g., [22, 24, 28, 29, 30, 31, 33, 34, 35, 37].

While early distributed (sub)gradient methods exhibit several useful features, e.g., [18], they also have the drawback that they do not converge to the exact problem solution when applied with a constant step-size; that is, for exact convergence, they need to utilize a diminishing step-size [39]. To address this issue, several different mechanisms have been proposed. Namely, in [25] two different weight-averaging matrices at two consecutive iterations are used. A gradient-tracking technique where the local updates are modified so that they track the network-wide average gradient of the nodes' local cost functions is proposed and analyzed in [12, 21]. The authors of [2] incorporate multiple consensus steps per each gradient update to obtain the convergence to the exact solution.

In this paper we investigate a different strategy to develop a novel class of exact distributed methods by employing quadratic penalty approach. The method is defined by the standard reformulation of distributed problem (1) into constrained problem in $\mathbb{R}^{nN}$ with constraints that penalize the differences in local approximations of the solution. The reformulated constrained problem is then solved by a quadratic penalty method. Given that the sequence of penalty subproblems is quadratic, we employ a fixed point linear solver to find zeroes of the corresponding gradients. Thus, we abbreviated the method as EFIX - Exact Fixed Point.

The proposed approach is as follows. The constrained distributed problem in $\mathbb{R}^{nN}$ is reformulated by adding a quadratic penalty term that penalizes the differences of solution estimates at neighbouring nodes across the network.

Then the sequence of penalty problems are solved inexactly, wherein the corresponding penalty parameters increase over time to make the algorithm exact. The algorithm parameters, the penalty parameter sequence and the levels of inexactness of the (inner) penalty problems, are designed such that the overall algorithm exhibits efficient behaviour. We consider two types of strongly convex objective functions - quadratic and generic strongly convex function. For quadratic objective function the subproblems are quadratic, while in the generic case we approximate the objective function at the current iteration with a quadratic model. Solving these problems boils down to finding zeroes of the gradients, i.e. to solving systems of linear equations and one can employ any distributed linear solver like fixed point iterative methods. The proposed framework is general and we exemplify it by employing the Jacobi Over-Relaxation (JOR) method for solving the penalty subproblems. Numerical tests on both simulated and real data sets demonstrate that the resulting algorithms are (at least) comparable with existing alternatives like [21], [36] in terms of the required computational and communication costs, as well as the required knowledge of global system parameters such as the global (maximal) Lipschitz constant of the local gradients L, strong convexity constant μ and the network parameters.

From the theoretical point of view the following results are established. First, for the quadratic cost functions, we show that either a sequence generated by the EFIX method is unbounded or it converges to the exact solution of the original problem (1). The worst-case complexity result of order $\mathcal{O}(\epsilon^{-1})$ is proved. For strongly convex costs with Lipschitz continuous gradients, the obtained result corresponds to the well-known result in the classical, centralized optimization - if the iterative sequence converges then its limit is the solution of the original problem. Admittedly, this result is weaker than what is known for existing alternatives like, e.g., [21], but are enough to theoretically certify the methods and are in line with the general theory of quadratic penalty methods; see, e.g., [19]. Numerical examples nevertheless demonstrate advantages of the proposed approach. Moreover, the convergence results of the proposed method are obtained although the Linear Independence Constraint Qualification, LICQ is violated.

It is worth noting that penalty approaches have been studied earlier in the context of distributed consensus optimization, e.g., [14, 15, 27, 40]. The authors of [40] allow for nondifferentiable costs, but their analysis relies on Lagrange multipliers and the distance from a closed, convex feasible set which plays a crucial role in the analysis. In [27], a differentiable exact penalty function is employed, but the problem under consideration assumes local constraints and separable objective function. Moreover, LICQ is assumed to hold. In our case, separating the objective function yields the constrained optimization problem (2) where the LICQ is violated. The authors of [15] consider more general problems with possibly nondiffrenetiable part of the objective function and linear constraints and provide the analysis for the decentralized distributed optimization problems (Section 4 of [15]). They show the convergence to an exact solution by carefully designing the penalty parameters and the step size sequence. The proposed algorithm boils down to the distributed gradient with time-varying step sizes. The convergence is of the order $\mathcal{O}(1/\sqrt{k})$, i.e., $\mathcal{O}(1/k)$ for the accelerated version. We notice that EFIX algorithm needs the gradient calculations only in the outer iterations, whenever the penalty parameter is increased and a new subproblem is generated, which makes it computationally less demanding. The numerical efficiency of the method in [15] is not documented to the best of out knowledge, although the convergence rate results are very promising. The strong convexity is not imposed in [15], and possibilities for relaxation of convexity requirements in EFIX are going to be the subject of further research. The algorithm presented in [14] is also based on penalty approach. A sequence of subproblems with increasing penalty parameters is defined and solved by accelerated proximal gradient method. Careful adjustment of algorithmic parameters yields a better complexity result than the results presented here. However, with respect to existing work, the proposed EFIX framework is more general in terms of the subsumed algorithms and can accommodate arbitrary R-linearly-converging solver for quadratic penalty subproblems. Finally, another important advantage of EFIX is the robustness with respect to algorithmic parameters.

The paper is organized as follows. In Section 2 we give some preliminaries. EFIX method for quadratic problems is defined and analyzed in Section 3. The analysis is extended to general convex case in Section 4 and the numerical results for both quadratic and general case are presented in Section 5. Some conclusions are drawn in Section 6. All proofs are moved to the Appendix.

2 Preliminaries

The notation we will use further is the following. With $\mathbb{A}, \mathbb{B}, \ldots$ we denote matrices in $\mathbb{R}^{nN \times nN}$ with block elements $\mathbb{A} = [A_{ij}],\ A_{ij} \in \mathbb{R}^{n \times n}$ and elements

$a_{ij} \in \mathbb{R}$. The vectors of corresponding dimensions will be denoted as $\mathbf{x} \in \mathbb{R}^{nN}$ with sub-blocks $x_i \in \mathbb{R}^n$ as well as $y \in \mathbb{R}^n$. The vector (matrix) norm $\|\cdot\|$ is the Euclidean (spectral) norm.

Let us specify more precisely the setup we consider here. The network of connected computational nodes is represented by a graph $G = (V, E)$, where V is the set of nodes $\{1, ..., N\}$ and E is the set of undirected edges (i, j). Denote by O_i the set of neighbors of node i and let $\bar{O}_i = O_i \bigcup \{i\}$. Let W and $L = I - W$ denote the communication matrix and the corresponding Laplacian matrix. The communication matrix W is a weighted adjacency matrix of G with additional properties as stated in Assumption 1 below. We also define augmented communication matrix $\mathbb{W} = W \otimes I \in \mathbb{R}^{nN \times nN}$ and augmented weighted Laplacian matrix $\mathbb{L} = \mathbb{I} - \mathbb{W}$, where $\mathbb{I} \in \mathbb{R}^{nN \times nN}$ is the identity matrix and $\otimes$ denotes the Kronecker product. The properties of the communication matrix W are stated as follows.

A 1. *The matrix $W \in \mathbb{R}^{N \times N}$ is symmetric, doubly stochastic and*

$$w_{ij} > 0 \text{ if } j \in \bar{O}_i, \; w_{ij} = 0 \text{ if } j \notin \bar{O}_i$$

The network G is connected and undirected.

Let us assume that each of N nodes has its local cost function f_i and has access to the corresponding derivatives of this local function. Under the assumption A1, the problem (1) has the equivalent form

$$\min_{\mathbf{x} \in \mathbb{R}^{nN}} F(\mathbf{x}) := \sum_{i=1}^{N} f_i(x_i) \quad \text{s. t.} \quad \mathbb{L}^{1/2}\mathbf{x} = 0, \tag{2}$$

where $\mathbf{x} = (x_1; ...; x_N) \in \mathbb{R}^{nN}$. Notice that under Assumption A1 the constraint $\mathbb{L}^{1/2}\mathbf{x} = 0$ is actually stating that the feasible vectors $\mathbf{x} = (x_1; ...; x_N)$ have the property $x_i = x_j, \; i, j = 1, \ldots, N$. Hence, the equivalence of (1) and (2) is in the following sense. Let us denote by $y^* \in \mathbb{R}^n$ the solution of (1). Then $\mathbf{x}^* = (y^*, \ldots, y^*) \in \mathbb{R}^{nN}$ is a solution of (2). Conversely, let $\mathbf{x}^*$ be the solution to (2). Then, $\mathbf{x}^* = (y^* ...., y^*)$, where y^* is the solution to (1). Now, the quadratic penalty reformulation of this problem associated with graph G is

$$\min_{\mathbf{x} \in \mathbb{R}^{nN}} \Phi_\theta(\mathbf{x}) := F(\mathbf{x}) + \frac{\theta}{2}\mathbf{x}^T \mathbb{L}\mathbf{x}, \tag{3}$$

where $\theta > 0$ is the penalty parameter. Clearly, each solution of (3) depends on the value of θ. Given that the quadratic penalty is not exact, to achieve equivalence we need $\theta \to \infty$. Otherwise, for any fixed value of θ one can show that the solution of (3), say $\tilde{x} \in \mathbb{R}^{nN}$ has the property that for any $i = 1, .., N$, there holds $\|\tilde{x}_i - y^*\| = \mathcal{O}(\theta)$. For further details one can see [38]. The EFIX method proposed in the sequel follows the sequential quadratic programming framework where the sequence of problems (3) with increasing values of the penalty parameters are solved approximately. Therefore, asymptotically we reach an exact solution of (1) as the penalty parameter goes to infinity. The communication matrix W influences the rate of convergence of the method but given that $\mathbb{W}\mathbf{x} = \mathbf{x}$ is the constraint that ensures $x_1 = x_2 = \ldots = x_N$ for all W that satisfy A1, it does not influence the equivalence of the reformulation, i.e., equivalence of the problems (1) and (2). Regarding the influence of matrix W on the relation between (2) and (3), it can be shown that, for a fixed θ, the difference between the solutions of (2) and (3) is on the order $\mathcal{O}(1/(1-\lambda_2))$, where λ_2 is the modulus of the second largest in modulus eigenvalue of W. See, e.g., Theorem 4 in [38] or equations (7) and (8) in [7].

3 EFIX-Q: Quadratic problems

Quadratic costs are very important subclass of problems that we consider. One of the typical example is linear least squares problem which comes from linear regression models, data fitting etc. We start the analysis with the quadratic costs given by

$$f_i(y) = \frac{1}{2}(y - b_i)^T B_{ii}(y - b_i), \tag{4}$$

where $B_{ii} = B_{ii}^T \in \mathbb{R}^{n \times n}, b_i \in \mathbb{R}^n$. Let us denote by $\mathbb{B} = diag(B_{11}, ..., B_{NN})$ the block-diagonal matrix and $\mathbf{b} = (b_1; ...; b_N) \in \mathbb{R}^{nN}$. Then, the penalty function defined in (3) with the quadratic costs (4) becomes

$$\Phi_\theta(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{b})^T \mathbb{B}(\mathbf{x} - \mathbf{b}) + \frac{\theta}{2}\mathbf{x}^T \mathbb{L}\mathbf{x}$$

and

$$\nabla\Phi_\theta(\mathbf{x}) = (\mathbb{B} + \theta\mathbb{L})\mathbf{x} - \mathbb{B}\mathbf{b}.$$

Thus, solving $\nabla\Phi_\theta(\mathbf{x}) = 0$ is equivalent to solving the linear system

$$\mathbb{A}(\theta)\mathbf{x} = \mathbf{c}, \quad \mathbb{A}(\theta) := \mathbb{B} + \theta\mathbb{L}, \quad \mathbf{c} := \mathbb{B}\mathbf{b}. \tag{5}$$

Clearly, matrix $\mathbb{A}(\theta)$ depends on parameter theta as well as a number of matrices and vectors derived from $\mathbb{A}$ below. To simplify the notation we will omit θ further on if θ is a generic parameter and place it whenever θ has some specific value. Under the following assumptions, this system can be solved in a distributed, decentralized manner by applying a suitable linear solver. To make the presentation more clear we concentrate here on the JOR method, without loss of generality.

A 2. *Each function $f_i,\ i = 1, \ldots, N$ is μ-strongly convex.*

This assumption implies that the diagonal elements of Hessian matrices B_{ii} are positive, bounded by μ from below. This can be easily verified by the fact that $y^T B_{ii} y \geq \mu \|y\|^2$ for $y = e_j, j = 1, ..., n$ where e_j is the j-th column of the identity matrix $I \in \mathbb{R}^{n \times n}$. Clearly, the diagonal elements of $\mathbb{A}$ are positive. Moreover, $\mathbb{A}$ is positive definite with minimal eigenvalue bounded from below with μ. Therefore, for arbitrary $\mathbf{x}_0 \in \mathbb{R}^{nN}$ and $\mathbb{A}, \mathbf{c}$ given in (5), we can define the JOR iterative procedure as

$$\mathbf{x}^{k+1} = \mathbb{M}\mathbf{x}^k + \mathbf{p}, \tag{6}$$

$$\mathbb{M} = q\mathbb{D}^{-1}\mathbb{G} + (1-q)\mathbb{I}, \quad \mathbf{p} = q\mathbb{D}^{-1}\mathbf{c}, \tag{7}$$

where $\mathbb{D}$ is a diagonal matrix with $d_{ii} = a_{ii}$ for all $i = 1, ..., nN$, $\mathbb{G} = \mathbb{D} - \mathbb{A}$, $\mathbb{I}$ is the identity matrix and q is the relaxation parameter. The structure of $\mathbb{A}$ and $\mathbb{M}$ makes the iterative method specified in (6) completely distributed assuming that each node i has the corresponding (block) row of $\mathbb{M}$, and thus we do not need any additional adjustments of the linear solver to the distributed network.

The JOR method (6)-(7) can be stated in the distributed manner as follows. Notice that the blocks of $\mathbb{A}$ are given by

$$A_{ii} = B_{ii} + \theta(1 - w_{ii})I, \quad \text{and} \quad A_{ij} = -\theta w_{ij} I \quad \text{for } i \neq j. \tag{8}$$

Therefore, we can represent JOR iterative matrix $\mathbb{M}$ in similar manner, i.e., $\mathbb{M} = [M_{ij}]$ where

$$M_{ii} = qD_{ii}^{-1}G_{ii} + (1-q)I, \quad M_{ij} = q\theta w_{ij} D_{ii}^{-1} \quad \text{for} \quad i \neq j, \tag{9}$$

and $\mathbf{p} = (p_1; ...; p_N)$ is calculated as

$$p_i = qD_{ii}^{-1}B_{ii}b_i. \tag{10}$$

Thus, each node i can update its own vector x_i by

$$x_i^{k+1} = \sum_{j \in \bar{O}_i} M_{ij} x_j^k + p_i. \tag{11}$$

Notice that (11) requires only the neighbouring x_j^k, and the corresponding elements of $M_{ij}, j \in \bar{O}_j$, i.e. the method is fully distributed.

The convergence interval for the relaxation parameter q is well known in this case, see e.g. [8].

Lemma 3.1. *Suppose that the assumptions A1-A2 are satisfied. Then the JOR method converges for all* $q \in (0, 2/\sigma(\mathbb{D}^{-1}\mathbb{A}))$*, with* $\sigma(\mathbb{D}^{-1}\mathbb{A}))$ *being the spectral radius of* $(\mathbb{D}^{-1}\mathbb{A}))$.

Lemma 3.1 gives the interval for relaxation parameter q that ensures that the spectral radius of $\mathbb{M}$ is smaller than 1 and hence gives the sufficient and necessary condition for convergence. Estimating the spectral radius of $\mathbb{D}^{-1}\mathbb{A}$ is not an easy task in general and several results are derived for specific matrix classes that specify the interval for q such that a sufficient condition for convergence holds, i.e. values of q that give $\|\mathbb{M}\|_p < \rho \leq 1$ for $p = 1, 2, \infty$.

Let us now estimate the interval stated in Lemma 3.1. We have

$$\sigma(\mathbb{D}^{-1}\mathbb{A}) \leq \|\mathbb{D}^{-1}\mathbb{A}\| \leq \|\mathbb{D}^{-1}\|\|\mathbb{A}\|.$$

Since the diagonal elements of B_{ii} are positive and $\mathbb{D}$ is the diagonal matrix with elements $d_{ii} = b_{ii} + \theta\ell_{ii}, i = 1, \ldots, nN$, with $\mathbb{L} = [\ell_{ij}] \in \mathbb{R}^{nN \times nN}$, we can upper bound the norm of $\mathbb{D}^{-1}$ as follows

$$\|\mathbb{D}^{-1}\| \leq \frac{1}{\theta(1 - \bar{w})},$$

where $\bar{w} := \max_i w_{ii} < 1$. On the other hand,

$$\|\mathbb{A}\| \leq \|\mathbb{B}\| + 2\theta \leq \max_i l_i + 2\theta := L + 2\theta,$$

where l_i is the largest eigenvalue of B_{ii}. So, the convergence interval for the relaxation parameter can be set as

$$q \in (0, \frac{2\theta(1 - \bar{w})}{L + 2\theta}). \tag{12}$$

Alternatively, one can use the infinity norm and obtain a bound as above with $\bar{B} := \max_i \|B_{ii}\|_\infty$ instead of L.

The iterative matrix $\mathbb{M}$ depends on the penalty parameter and the EFIX algorithm we define further on solves a sequence of penalty problems defined by a sequence of penalty parameters $\theta_s, s = 0, 1, \ldots$. Thus (12) can be updated for each penalty subproblem, defined with a new penalty parameter. However the upper bound in (12) is monotonically increasing with respect to θ, so one can set $q \in (0, 2\theta_0(1-\bar{w})/(L + 2\theta_0))$ without updating with the change of θ. In the test presented in Section 5 we use $\theta_0 = 2L$, which further implies that the JOR parameter can be fixed to any positive value smaller than $4(1-\bar{w})/5$.

The globally convergent algorithm for problem (1) with quadratic functions (4) is given below. In each subproblem we have to solve a linear system of type (5). The algorithm is designed such that these linear systems are solved within an inner loop defined by (11). The penalty parameters $\{\theta_s\}$ with the property $\theta_s \to \infty, \; s \to \infty$, and the number of inner iterations $k(s)$ of type (11) are assumed to be given. Also, we assume that the relaxation parameters $q(s)$ are defined by a rule that fulfills (12). Thus, for given θ_s the linear system $\mathbb{A}(\theta_s)\mathbf{x} = \mathbf{c}$ is solved approximately in each outer iteration, with the iterative matrix

$$\mathbb{M}(\theta_s) = q(s)\mathbb{D}^{-1}\mathbb{G} + (1 - q(s))\mathbb{I}.$$

The global constants L and $\bar{w}$ are needed for updating the relaxation parameter in each iteration but the nodes can settle them through initial communication at the beginning of iterative process. Thus, they are also treated as input parameters for the algorithm. Notice that constants L and $\overline{w}$ represent maxima of certain scalar quantities distributed across nodes in the network. Let us consider L, while similar arguments hold for $\overline{w}$ as well. Assuming that each node i knows the Lipschitz constant l_i of ∇f_i, then each node can obtain L after the nodes perform a distributed algorithm to calculate L that can be taken as $L = \max_i l_i$. There are several ways to calculate maximum in a fully distributed way inexpensively, e.g., [26]. Such algorithm converges in $\mathcal{O}(diam)$ iterations (communication rounds), where *diam* is the network diameter.

Algorithm EFIX-Q.

Given: $\{\theta_s\}$, $x_i^0 \in \mathbb{R}^n, i = 1, ..., N$, $\{k(s)\} \subset \mathbb{N}, L, \bar{w}$. Set $s = 0$.

S1 Set $k = 0$ and choose q according to (12) with $\theta = \theta_s$. Let $\mathbb{M} = \mathbb{M}(\theta_s), z_i^0 = x_i^s, i = 1, \ldots, N$.

S2 For each $i = 1, \dots, N$ compute the new local solution estimates

$$z_i^{k+1} = \sum_{j \in \bar{O}_i} M_{ij} z_j^k + p_i$$

and set $k = k + 1$.

S3 If $k < k(s)$ go to step S2. Else, set $\mathbf{x}^{s+1} = (z_1^k, \dots, z_N^k),\ s = s + 1$ and go to step S1.

The above algorithm relies on distributed implementation of the fixed point solver JOR. Thus each node has a set of local information, to be more specific each node i has the corresponding ith block-row of the matrix $\mathbb{M}$ and the corresponding vector p_i. In fact, having in mind the structure of matrix $\mathbb{M}$, one can see that each node i, besides the input parameters, only needs to store the following: the Hessian of the local cost function, i.e., $B_{ii} \in \mathbb{R}^{n \times n}$; the vector $b_i \in \mathbb{R}^n$; and the weights $(w_{i1}, w_{i2}, ..., w_{iN}) \in \mathbb{R}^N$, i.e., the ith row of the matrix W. For instance, notice that the block M_{ij} of the matrix $\mathbb{M}$ can be derived from the stored data since it varies through j directly with w_{ij} and D_{ii} is the diagonal matrix with the diagonal which coincides with the diagonal of $A_{ii} = B_{ii} + \theta(1 - w_{ii})I$. Additionally, node i can also store the vectors $p_i, diag(D_{ii}^{-1}) \in \mathbb{R}^n$ and the matrix $M_{ii} \in \mathbb{R}^{n \times n}$ in order to avoid unnecessary calculations within inner iterations. Each node computes z_i^{k+1} in Step 2, using $z_j^k, j \in \bar{O}_i$ from its neighbors and computing $M_{ij} z_j^k$ and after that transmits the new approximation z_i^{k+1} to the neighbors. Thus, at each iteration, each node i sends to its immediate neighbors in graph G vector $z_i^{k+1} \in \mathbb{R}^n$ and receives the corresponding estimates of the neighboring nodes $z_j^{k+1} \in \mathbb{R}^n, j \in O_i$.

Our analysis relies on the quadratic penalty method, so we state the framework algorithm (see [19] for example). We assume again that the sequence of penalty parameters $\{\theta_s\}$ has the property $\theta_s \to \infty$ and that the tolerance sequence $\{\varepsilon_s\}$ is such that $\varepsilon_s \to 0$.

Algorithm QP.

Given: $\{\theta_s\},\ \{\varepsilon_s\}$. Set $s = 0$.

S1 Find $\mathbf{x}^s$ such that

$$\|\nabla \Phi_{\theta_s}(\mathbf{x}^s)\| \leq \varepsilon_s. \tag{13}$$

S2 Set $s = s + 1$ and return to S1.

Let us demonstrate that the EFIX-Q fits into the framework of Algorithm QP, that is given a sequence $\{\varepsilon_s\}$ such that $\varepsilon_s \to 0$, there exists a proper choice of the sequence $\{k(s)\}$ such that (13) is satisfied for all penalty subproblems.

Lemma 3.2. *Suppose that the assumptions A1-A2 are satisfied. If* $\|\nabla \Phi_{\theta_s}(\mathbf{x}^s)\| \leq \varepsilon_s$ *then* $\|\nabla \Phi_{\theta_{s+1}}(\mathbf{x}^{s+1})\| \leq \varepsilon_{s+1}$ *for*

$$k(s) = \left\lceil \left| \frac{\log(\mu \varepsilon_{s+1}) - \log(L + 2\theta_{s+1})(\varepsilon_s + 2\bar{c})}{\log(\rho_{s+1})} \right| \right\rceil, \tag{14}$$

where ρ_{s+1} *is a constant such that* $\|\mathbb{M}(\theta_{s+1})\| \leq \rho_{s+1} < 1$ *and* $\bar{c} = \|\mathbf{c}\|$.

The previous lemma shows that EFIX-Q fits into the framework of quadratic penalty methods presented above if we assume $\varepsilon_s \to 0$ and set $k(s)$ as in (14), with $\{\mathbf{x}^s\}$ being the outer iterative sequence of Algorithm EFIX-Q. Notice that the inner iterations (that rely on the JOR method) stated in steps S2-S3 of EFIX-Q can be replaced with any solver of linear systems or any optimizer of quadratic objective function which can be implemented in decentralized manner and exhibits linear convergence with factor ρ_s. Moreover, it is enough to apply a solver with R-linear convergence, i.e., any solver that satisfies

$$\|\mathbf{z}^k - \mathbf{x}^*_{\theta_{s+1}}\| \leq C_{s+1} \|\mathbf{x}^s - \mathbf{x}^*_{\theta_{s+1}}\| \rho^k_{s+1},$$

where C_{s+1} is a positive constant. In this case, the slightly modified $k(s)$ with $(L + 2\theta_{s+1})$ multiplied with C_{s+1} in (14) fits the proposed framework.

Although the LICQ does not hold for (2), following the steps of the standard proof and modifying it to cope with LICQ violation, we obtain the global convergence result presented below.

Theorem 3.1. *Suppose that the assumptions A1-A2 are satisfied. Assume that* $\varepsilon_s \to 0$ *and* $k(s)$ *is defined by* (14). *Let* $\{\mathbf{x}^s\}$ *be a sequence generated by algorithm EFIX-Q. Then, either the sequence* $\{\mathbf{x}^s\}$ *is unbounded or it converges to a solution* $\mathbf{x}^*$ *of the problem* (2) *and* x_i^* *is the solution of problem* (1) *for every* $i = 1, ..., N$.

The previous theorem states that the only requirement on $\{\varepsilon_s\}$ is that it is a positive sequence that tends to zero. On the other hand, quadratic penalty function is not exact penalty function and the solution $\mathbf{x}^*_\theta$ of the penalty problem (3) is only an approximation of the solution y^* of problem

(1). Moreover, it is known (see Corollary 9 in [38]) that for every $i = 1, ..., N$, there holds

$$e^1_{i,\theta} := \|x^*_{i,\theta} - y^*\| = \mathcal{O}(\theta^{-1}).$$

More precisely, denoting by λ_2 the second largest eigenvalue of W in modulus, we have

$$e^1_{i,\theta_s} \leq \frac{LJ}{\theta_s \kappa(1-\lambda_2)} \sqrt{4 - 2\kappa\theta_s^{-1}} + \frac{J}{\theta_s(1-\lambda_2)}, \tag{15}$$

where $\kappa = \mu L/(\mu + L)$ and $J = \sqrt{2Lf(0)}$ since the optimal value of each local cost function is zero. Thus, looking at an arbitrary node i and any outer iteration s we have

$$\|x_i^s - y^*\| \leq \|x_i^s - x^*_{i,\theta_s}\| + \|x^*_{i,\theta_s} - y^*\| := e^2_{i,\theta_s} + e^1_{i,\theta_s}. \tag{16}$$

So, there is no need to solve the penalty subproblem with more accuracy than $e^1_{i,\theta}$ - the accuracy of approximating the original problem. Therefore, using (27) and (15) and balancing these two error bounds we conclude that a suitable value for ε_s, see (27), can be estimated as

$$\varepsilon_s = \mu \left(\frac{LJ}{\theta_s \kappa(1-\lambda_2)} \sqrt{4 - 2\kappa\theta_s^{-1}} + \frac{J}{\theta_s(1-\lambda_2)} \right) \tag{17}$$

Similar idea of error balance is used in [39], to decide when to decrease the step size.

Assume that we define ε_s as in (17) Together with (27) we get

$$\|x_i^s - x^*_{i,\theta_s}\| = \mathcal{O}\left(\frac{1}{\theta_s}\right).$$

Furthermore, using (15) and (16) we obtain

$$\|x_i^s - y^*\| = \mathcal{O}\left(\frac{1}{\theta_s}\right).$$

Therefore, the following result concerning the outer iterations holds.

Proposition 3.1. *Suppose that the assumptions of Theorem 3.1 hold and that ε_s is defined by (17). Let $\{\mathbf{x}^s\}$ be a bounded sequence generated by EFIX-Q . Then for every $i = 1, ..., N$ there holds*

$$\|x_i^s - y^*\| = \mathcal{O}\left(\frac{1}{\theta_s}\right).$$

The complexity result stated below for the special choice of penalty parameters, $\theta_s = s$ can be easily derived using the above Proposition.

Corollary 3.1. *Suppose that the assumptions of Proposition 3.1 hold and* $\theta_s = s$ *for* $s = 1, 2, \ldots$. *Then after at most*

$$\bar{s} = \left\lceil \frac{2J(3 + 2L/\mu)}{(1 - \lambda_2)} \epsilon^{-1} \right\rceil$$

iterations we have $\|x_i^{\bar{s}} - y^*\| \leq \epsilon$ *for all* $i = 1, ..., N$ *and any* $\epsilon > 0$, *where* J *and* λ_2 *are as in* (15).

Notice that the number of outer iterations $\bar{s}$ to obtain the ϵ-optimal point depends directly on J, i.e., on $f(0)$ and the Lipschitz constant L. Moreover, it also depends on the network parameters - recall that λ_2 represents the second largest eigenvalue of the matrix W, so the complexity constant can be diminished if we can chose the matrix W such that λ_2 is as small as possible for the given network.

4 EFIX-G: Strongly convex problems

In this section, we consider strongly convex local cost functions f_i that are not necessarily quadratic. The main motivation comes from machine learning problems such as logistic regression where the Hessian is easy to calculate and, under regularization, satisfies Assumption A2. The main idea now is to approximate the objective function with a quadratic model at each outer iteration s and exploit the previous analysis. Instead of solving (13), we form a quadratic approximation $Q_s(\mathbf{x})$ of the penalty function $\Phi_{\theta_s}(\mathbf{x})$ defined in (3) as

$$\begin{aligned} Q_s(\mathbf{x}) \quad &:= \quad F(\mathbf{x}^{s-1}) + \nabla^T F(\mathbf{x}^{s-1})(\mathbf{x} - \mathbf{x}^{s-1}) + \\ &+ \quad \frac{1}{2}(\mathbf{x} - \mathbf{x}^{s-1})^T \nabla^2 F(\mathbf{x}^{s-1})(\mathbf{x} - \mathbf{x}^{s-1}) + \frac{\theta_s}{2}\mathbf{x}^T \mathbb{L}\mathbf{x} \end{aligned} \tag{18}$$

and search for $\mathbf{x}^s$ that satisfies

$$\|\nabla Q_s(\mathbf{x}^s)\| \leq \varepsilon_s. \tag{19}$$

In other words, we are solving the system of linear equations

$$\mathbb{A}_s \mathbf{x} = \mathbf{c}_s,$$

where

$$\mathbb{A}_s := \nabla^2 F(\mathbf{x}^{s-1}) + \theta_s \mathbb{L},$$
$$\mathbf{c}_s := \nabla^2 F(\mathbf{x}^{s-1})\mathbf{x}^{s-1} - \nabla F(\mathbf{x}^{s-1}).$$

Under the stated assumptions, $\mathbb{A}_s$ is positive definite with eigenvalues bounded with μ from below and the diagonal elements of $\mathbb{A}_s$ are strictly positive. Therefore, using the same notation and formulas as in the previous section with $\nabla^2 f_i(x_i^{s-1})$ instead of B_{ii} in (8) we obtain the same bound for the JOR parameter, (12).

Before stating the algorithm, we repeat the formulas for completeness. The matrix $\mathbb{A}_s = [A_{ij}]$ has blocks $A_{ij} \in \mathbb{R}^{n \times n}$ given by

$$A_{ii} = \nabla^2 f_i(x_i^{s-1}) + \theta_s(1 - w_{ii})I, \quad \text{and} \quad A_{ij} = -\theta_s w_{ij} I \quad \text{for } i \neq j. \tag{20}$$

The JOR iterative matrix is $\mathbb{M}_s = [M_{ij}]$ where

$$M_{ii} = q_s D_{ii}^{-1} G_{ii} + (1 - q)I, \quad M_{ij} = q_s \theta_s w_{ij} D_{ii}^{-1} \quad \text{for} \quad i \neq j, \tag{21}$$

and the vector $\mathbf{p}_s = (p_1; ...; p_N)$ is calculated as $\mathbf{p}_s = q\mathbb{D}_s^{-1}\mathbf{c}_s$, where $\mathbb{D}_s$ is a diagonal matrix with $d_{ii} = a_{ii}$ for all $i = 1, ..., nN$ and $\mathbb{G}_s = \mathbb{D}_s - \mathbb{A}_s$, i.e.,

$$p_i = qD_{ii}^{-1} c_i, \quad \text{where} \quad c_i = \nabla^2 f_i(x_i^{s-1})x_i^{s-1} - \nabla f_i(x_i^{s-1}). \tag{22}$$

The algorithm presented below is a generalization of EFIX-Q and we assume the same initial setup: the global constants L and $\bar{w}$ are known, the sequence of penalty parameters $\{\theta_s\}$ and the sequence of inner iterations counters $\{k(s)\}$ are input parameters for the algorithm.

Algorithm EFIX-G.

Input: $\{\theta_s\}$, $x_i^0 \in \mathbb{R}^n, i = 1, ..., N$, $\{k(s)\} \subset \mathbb{N}, L, \bar{w}$. Set $s = 0$.

S1 Each node i sets q according to (12) with $\theta = \theta_s$.

S2 Each node calculates $\nabla f_i(x_i^s)$ and $\nabla^2 f_i(x_i^s)$. Define $\mathbb{M} = \mathbb{M}_s$ given by (21), $z_i^0 = x_i^s, i = 1, \ldots, N$ and set $k = 0$.

S3 For $i = 1, \ldots, N$ update the solution estimates

$$z_i^{k+1} = \sum_{j \in \bar{O}_i} M_{ij} z_j^k + p_i$$

and set $k = k + 1$.

S4 If $k < k(s)$ go to step S3. Else, set $\mathbf{x}^{s+1} = (z_1^k; \ldots; z_N^k)$, $s = s+1$ and go to step S1.

The algorithm differs from the quadratic case EFIX-Q in step S2, where the gradients and the Hessians are calculated in a new point at every outer iteration. Each node i, besides the input parameters, stores the weights $(w_{i1}, w_{i2}, ..., w_{iN}) \in \mathbb{R}^N$. Moreover, it calculates the Hessian of the local cost function $\nabla^2 f_i(x_i^s) \in \mathbb{R}^{n \times n}$ and the corresponding gradient $\nabla f_i(x_i^s) \in \mathbb{R}^n$ at each outer iteration and stores them through the inner iterations. Similarly to the EFIX-Q case, node i can also store the vectors $p_i, diag(D_{ii}^{-1}) \in \mathbb{R}^n$ and the matrix $M_{ii} \in \mathbb{R}^{n \times n}$ calculated at each outer iteration in order to avoid unnecessary calculations within the corresponding inner iterations. At each iteration, each node exchanges the current estimates of the solution (vectors z_j^k) with its immediate neighbors as explained in EFIX-Q case.

Following the same ideas as in the proof of Lemma 3.2, we obtain the similar result under the following additional assumption.

A 3. *For each $y \in \mathbb{R}^n$ there holds $\|\nabla^2 f_i(y)\| \leq l_i$, $i = 1, ..., N$.*

Notice that this assumption implies that $\|\nabla^2 F(\mathbf{x})\| \leq L := \max_i l_i$.

Lemma 4.1. *Suppose that Assumptions A1-A3 hold. If $\|\nabla Q_s(\mathbf{x}^s)\| \leq \varepsilon_s$ holds then $\|\nabla Q_{s+1}(\mathbf{x}^{s+1})\| \leq \varepsilon_{s+1}$ for*

$$k(s) = \left\lceil \left| \frac{\log(\mu \varepsilon_{s+1}) - \log(L + 2\theta_{s+1})(\varepsilon_s + \bar{c}_s + \bar{c}_{s+1})}{\log(\rho_{s+1})} \right| \right\rceil, \tag{23}$$

where ρ_{s+1} is a constant such that $\|\mathbb{M}_{s+1}\| \leq \rho_{s+1} < 1$ and $\bar{c}_s = \|\mathbf{c}_s\|$.

The Lemma above implies that EFIX-G is a penalty method with the penalty function Q instead of Φ, i.e., with (19) instead of (13). Notice that due to assumption A2, without loss of generality we can assume that the functions f_i are nonnegative and thus the relation between ε_s and θ_s can remain as in (17). We have the following convergence result which corresponds to the classical statement in centralized optimization, [19].

Theorem 4.1. *Let the assumptions A1-A3 hold. Assume that $\{\mathbf{x}^s\}$ is a sequence generated by Algorithm EFIX-G such that $k(s)$ is defined by* (23) *and $\varepsilon_s \to 0$. If $\{\mathbf{x}^s\}$ is bounded then every accumulation point of $\{\mathbf{x}^s\}$ is feasible for the problem* (2). *Furthermore, if $\lim_{s \to \infty} \mathbf{x}^s = \mathbf{x}^*$ then $\mathbf{x}^*$ is the solution of problem* (2), *i.e., x_i^* is the solution of problem* (1) *for every $i = 1, ..., N$.*

5 Numerical results

5.1 Quadratic case

We test EFIX-Q method on a set of quadratic functions (4) defined as in [12]. Vectors b_i are drawn from the Uniform distribution on $[1, 31]$, independently from each other. Matrices B_{ii} are of the form $B_{ii} = P_i S_i P_i$, where S_i are diagonal matrices with Uniform distribution on $[1, 101]$ and P_i are matrices of orthonormal eigenvectors of $\frac{1}{2}(C_i + C_i^T)$ where C_i have components drawn independently from the standard Normal distribution.

The network is formed as follows, [12]. We sample N points randomly and uniformly from $[0, 1] \times [0, 1]$. Two points are directly connected if their distance, measured by the Euclidean norm, is smaller than $r = \sqrt{\log(N)/N}$. The graph is connected. Moreover, if nodes i and j are directly connected, we set $w_{i,j} = 1/\max\{deg(i), deg(j)\}$, where $deg(i)$ stands for the degree of node i and $w_{i,i} = 1 - \sum_{j \neq i} w_{i,j}$. We test on graphs with $N = 30$ and $N = 100$ nodes.

The error metrics is the following

$$e(x^k) := \frac{1}{N} \sum_{i=1}^{N} \frac{\|x_i^k - y^*\|}{\|y^*\|}, \tag{24}$$

where $y^* \neq 0$ is the exact (unique) solution of problem (1).

The parameters are set as follows. The Lipschitz constant is calculated as $L = \max_i l_i$, where l_i is the largest eigenvalue of B_{ii}. The strong convexity constant is calculated as $\mu = \min_i \mu_i$, where $\mu_i > 0$ is the smallest eigenvalue of B_{ii}.

The proposed method is denoted by EFIX-Q $k(s)$ balance to indicate that we use the number of inner iterations given by (14) where $L, \mu, \bar{c}$ are calculated at the initial phase of the algorithm and imposing (17) to balance two types of errors as discussed in Section 3. The initial value of the penalty parameter is set to $\theta_0 = 2L$. The choice is motivated by the fact that the usual step size bound in many gradient-related methods is $\alpha < 1/(2L)$ and $1/\alpha$ corresponds to the penalty parameter. Hence, we set $\theta \geq 2L$. Further, the penalty parameter is updated by $\theta_{s+1} = (s+1)\theta_s$. The inner solver used at Step 3 of EFIX-Q method is the Jacobi method, i.e., JOR method with $q = 1$. In the quadratic case, the Jacobi method converged and the bounds derived in (12) were not needed. The Jacobi method (JOR method in general) is used to solve the sequence of quadratic problems up to accuracy determined $\{\varepsilon_s\}$. Clearly, the precision, measured by ε_s determines the computational costs. On the other hand it is already discussed that the error in solving a particular quadratic problem should not be decreased too much given that the quadratic penalty is not an exact method and hence each quadratic subproblem is only an approximation of the original constrained problem, depending on the penalty parameter θ_s. Therefore, we tested several choices of the inner iteration counter and parameter update, to investigate the error balance and its influence on the convergence. The method abbreviated as EFIX-Q $k(s)$ is obtained with $\varepsilon_0 = \theta_0 = 2L$, $\varepsilon_s = \varepsilon_0/s$ for $s > 0$, and $k(s)$ defined by (14). Furthermore, to demonstrate the effectiveness of $k(s)$ stated in (14) we also report the results from the experiments where the inner iterations are terminated only if (13) holds, i.e. without a predefined sequence $k(s)$. We refer to this method as EFIX-Q stopping. Notice that the exit criterion of EFIX-Q is not computable in the distributed framework and the test reports here are performed only to demonstrate the effectiveness of (14).

The proposed method is compared with the state-of-the-art method [21, 17] abbreviated as DIGing $1/(mL)$, where $1/(mL)$ represents the step size, i.e., $\alpha = 1/(mL)$ for different values of $m \in \{2, 3, 10, 20, 50, 100\}$. This method is defined as follows

$$x_i^{k+1} = \sum_{j=1}^{N} w_{ij} x_j^k - \alpha u_i^k,\ u_i^{k+1} = \sum_{j=1}^{N} w_{ij} u_j^k + B_{ii}(x_i^{k+1} - x_i^k), u_i^0 = \nabla f_i(x_i^0).$$

We model the total computational cost by counting the total number of n-dimensional SPs (scalar products of two n-dimensional real vectors) per node evaluated during the algorithm run, i.e., we let the unit computational cost be a single n-dimensional SP evaluation. Here, we model a single N-dimensional scalar product computational cost as $\xi := N/n$ unit costs. Thus, the computational cost of DIGing method per node, per iteration, can be estimated to $n + 2n\xi$ since $\sum_j w_{ij} x_j^k$ takes $n\xi$ unit computational costs (n N-dimensional SPs) as well as $\sum_j w_{ij} u_j^k$, and $B_{ii}(x_i^{k+1} - x_i^k)$ takes n unit computational costs (n n-dimensional SPs). In the sequel, we refer to unit computational costs as SPs.

In order to compare the costs, we unfold the proposed EFIX-Q method considering all inner iterations consecutively (so k below is the cumulative counter for all inner iterations) as follows

$$x_i^{k+1} = qD_{ii}^{-1}G_{ii}x_i^k + (1-q)x_i^k + q\theta D_{ii}^{-1}\sum_{i\neq j} w_{ij}x_j^k + qD_{ii}^{-1}B_{ii}b_i.$$

Since D_{ii} is a diagonal matrix, $qD_{ii}^{-1}G_{ii}x_i^k$ takes $n+1$ SPs and $D_{ii}^{-1}\sum_{i\neq j} w_{ij}x_j^k$ takes $1+n\xi$ SPs. Moreover, $B_{ii}b_i$ is calculated only once, at the initial phase, so $D_{ii}^{-1}B_{ii}b_i$ costs only 1 SPs (unit costs). Thus, the cost of EFIX-Q method can be estimated to $n+3+n\xi$ SPs per node, per iteration. The difference between EFIX-Q and DIGing can be significant especially for larger value of N, given the relative difference between quantities 3 and $n\xi = N$. Moreover, DIGing method requires at each iteration the exchange of two vectors, x_j and u_j among all neighbors, while EFIX requires only the exchange of x_j, so it is 50% cheaper than the DIGing method in terms of communication costs per iteration.

We set $\mathbf{x}^0 = 0$ for all the tested methods and consider $n = 10$ and $n = 100$. Figure 1 presents the errors $e(\mathbf{x}^k)$ throughout iterations k for $N = 30$ and $N = 100$. The results for different values of n appear to be very similar and hence we report only the case $n = 100$.

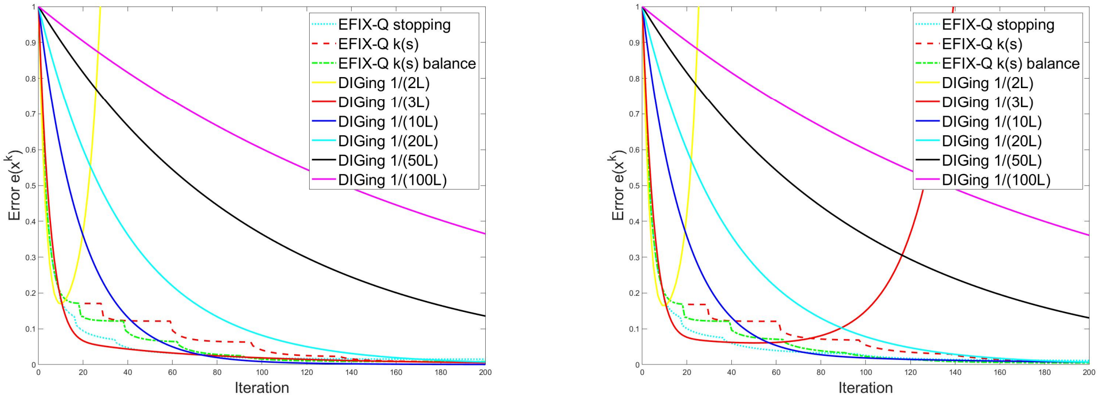

Figure 1: The EFIX methods (dotted lines) versus the DIGing method, error (24) propagation through iterations for $n = 100$, $N = 30$ (left) and $n = 100$, $N = 100$ (right).

Comparing the number of iterations of all considered methods, from Figure 1 one can see that EFIX-Q methods are highly competitive with the best

DIGing method in the case of $N = 30$. Furthermore, EFIX-Q outperforms all the convergent DIGing methods in the case of $N = 100$. Moreover, we can see that EFIX-Q $k(s)$ balance behaves similarly to EFIX-Q stopping, so the number of inner iterations $k(s)$ given in Lemma 3.2 is well estimated. Also, EFIX-Q $k(s)$ balance improves the performance of EFIX-Q $k(s)$ and the balancing of errors yields a more efficient method.

We compare the tested methods in terms of computational costs, measured by scalar products and communication costs as well. The results are presented in Figure 2 where we compare EFIX-Q $k(s)$ balance with the best convergent DIGing method in the cases $n = 10, N = 30$ (top) and $n = 100, N = 100$ (bottom). The results show clear advantages of EFIX-Q, especially in the case of larger n and N.

5.2 Strongly convex problems

EFIX-G method is tested on the binary classification problems for data sets: Mushrooms [32] ($n = 112$, total sample size $T = 8124$), CINA0 [5] ($n = 132$, total sample size $T = 16033$) and Small MNIST [20] ($n = 100$, total sample size $T = 7603$). For each of the problems, the data is divided across 30 nodes of the graph described in Subsection 5.1 The logistic regression with the quadratic regularization is used and thus the local objective functions are of the form

$$f_i(y) = \sum_{j \in J_i} \log(1 + e^{-\zeta_j d_j^T y}) + \frac{\mu}{2}\|y\|^2 := \sum_{j \in J_i} \tilde{f}_j(y),$$

where J_i collects the indices of the data points assigned to node i, $d_j \in \mathbb{R}^n$ is the corresponding vector of attributes and $\zeta_j \in \{-1, 1\}$ represents the label. The gradient and the Hessian of $\tilde{f}_j(y)$ are given by

$$\nabla \tilde{f}_j(y) = \frac{1 - \psi_j(y)}{\psi_j(y)} \zeta_j d_j + \mu y, \ \nabla^2 \tilde{f}_j(y) = \frac{\psi_j(y) - 1}{\psi_j^2(y)} d_j d_j^T + \mu I,$$

$$\psi_j(y) := 1 + e^{-\zeta_j d_j^T y}.$$

Thus, evaluating the gradient of $\tilde{f}_i(y)$ costs 1 SPs. Also, we estimate the cost of calculating the Hessian of $\tilde{f}_i(y)$ with $n/2$ SPs. Moreover, $(\psi_j(y) - 1)/\psi_j^2(y) \in (0, 1)$ and thus all the local cost functions are μ-strongly convex. The data is scaled in a such way that the Lipschitz constants l_i are 1 and thus $L = 1 + \mu$. We set $\mu = 10^{-4}$.

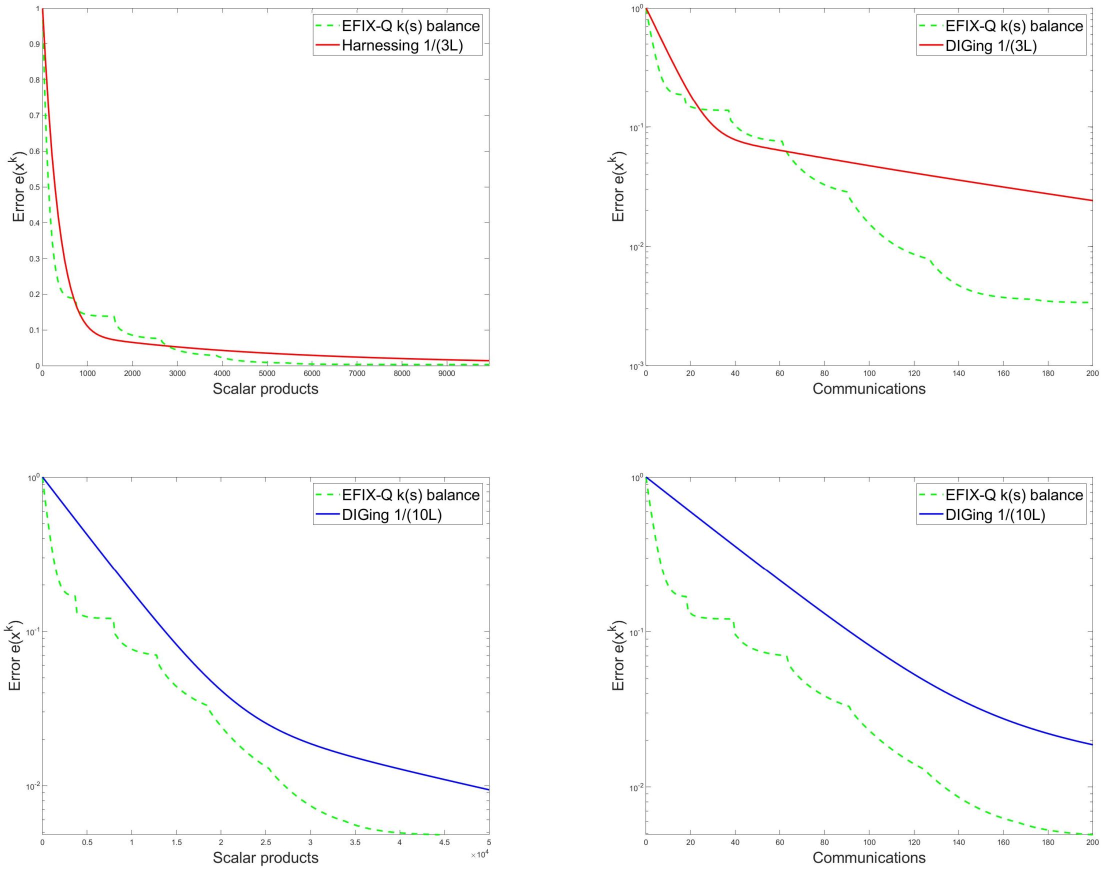

Figure 2: The proposed method (dotted line) versus the DIGing method, error (24) and the computational cost (left) and communications (right) for $n = 10$, $N = 30$ (top) and $n = 100$, $N = 100$ (bottom).

We test EFIX-G $k(s)$ balance, the counterpart of the quadratic version EFIX-Q $k(s)$ balance, with $k(s)$ defined by (23). The JOR parameter q_s is set according to (12), more precisely, we set $q = 2\theta_s(1-\bar{w})/(L+2\theta_s)$. We report here that, unlike the quadratic case, Jacobi method did not converge and we had to use the estimate (12). A rough estimation of $\bar{c}_s$ is s $3L\sqrt{N}$ since

$$\|\mathbf{c}_s\| \leq \|\nabla^2 F(\mathbf{x}^{s-1})\|\|\mathbf{x}^{s-1}\| + \|\nabla F(\mathbf{x}^{s-1}) - \nabla F(\tilde{\mathbf{x}})\| \leq L3\max\{\|\mathbf{x}^{s-1}\|, \|\tilde{\mathbf{x}}\|\},$$

where $\tilde{\mathbf{x}}$ is a stationary point of the function F. The remaining parameters are set as in the quadratic case.

Since the solution is unknown in general, the different error metric is used - the average value of the original objective function f across the nodes' estimates

$$v(\mathbf{x}^k) = \frac{1}{N}\sum_{i=1}^{N} f(x_i^k) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} f_j(x_i^k). \tag{25}$$

We compare the proposed method with DIGing which takes the following form for general, non-quadratic problems

$$x_i^{k+1} = \sum_{j=1}^{N} w_{ij}x_j^k - \alpha u_i^k,\ u_i^{k+1} = \sum_{j=1}^{N} w_{ij}u_j^k + \nabla f_i(x_i^{k+1}) - \nabla f_i(x_i^k), u_i^0 = \nabla f_i(x_i^0).$$

For each of the data sets we compare the methods with respect to iterations, communications and computational costs (scalar products). The communications of the DIGing method are twice more expensive than for the proposed method, as in the quadratic case. Denote $\tilde{\xi} = |J_i|/n$. The computational cost of the DIGing method is estimated to $2n\xi + n\tilde{\xi} + |J_i|$ SPs per iteration, per node: weighted sum of x_j^k ($n\xi$ SPs); weighted sum of u_j^k ($n\xi$ SPs); evaluating $\nabla f_i(x_i^k)$ ($n\tilde{\xi} + |J_i|$ SPs) because evaluating of each gradient $\nabla \tilde{f}_j(x_i^k), j \in J_i$ costs 1 SP (for $d_j^T x_i^k$ needed for calculating $\psi_j(x_i^k)$) and evaluating the gradient $\nabla f_i(x_i^k)$ takes the weighted sum of d_j vectors

$$\nabla f_i(x_i^k) = \sum_{j\in J_i} \frac{1-\psi_j(x_i^k)}{\psi_j(x_i^k)}\zeta_j d_j + \mu x_i^k,$$

which costs $n\tilde{\xi}$ SPs (n $|J_i|$-dimensional scalar products). On the other hand, the cost of EFIX-G $k(s)$ balance per node remains $n\xi + n + 3$ scalar products at each inner iteration while in the outer iterations (s) we have additional $|J_i|+2n\tilde{\xi}+\tilde{\xi}n^2/2$ scalar products for evaluating Hessian ($\tilde{\xi}n^2/2$) and c_i given by

$$c_i = \sum_{j\in J_i} \frac{\psi_j(x_i^{s-1})-1}{\psi_j^2(x_i^{s-1})} d_j d_j^T x_i^{s-1} + \mu x_i^{s-1} - \nabla f_i(x_i^{s-1}),$$

i.e., in order to evaluate c_i we have $|J_i|$ scalar products of the form $d_j^T x_i^{s-1}$, a weighted sum od d_j vectors which costs $n\tilde{\xi}$ SPs and the gradient $\nabla f_i(x_i^{s-1})$ which costs only $n\tilde{\xi}$ SPs since the scalar products $d_j^T x_i^{s-1}$ are already evaluated and calculated in the first sum.

The results are presented in Figure 3, (y-axes is in the log scale). The first column contains graphs for EFIX - G $k(s)$balance and all DIGing methods with error metrics through iterations. Obviously, the EFIX -G method is either comparable or better in comparison with DIGing methods in all tested problems, except for Small MNIST dataset . To emphasize the difference in computational costs we plot in column two the graphs of error metrics with respect to SPs for EFIX -G and the two best DIGing method. The same is done in column three of the graph for the communication costs.

5.3 Additional comparisons

We provide additional comparisons with the very recent algorithm termed OPTRA in [36], as a further representative of state of the art. The authors of [36] show that, up to universal constants and in the smooth convex (non strongly convex) setting, OPTRA matches theoretical lower complexity bounds in terms of communication and computation, with respect to oracles as defined in [36]. Also, the numerical examples in [36] show that OPTRA is competitive with several state of the art alternatives. Therefore, we compare EFIX also with OPTRA. We perform tests on well connected communication matrix W defined at the beginning of Subsection 5.1 and on ring structure represented by W_{ring} communication matrix to examine the behavior on the network which is not well connected. In this case, we set diagonal elements of W_{ring} to 0.5 and the relevant (nonzero) off-diagonal elements are 0.25.

All the parameters for EFIX methods are the same as in the previous two subsections and we consider the same set test problems. We test OPTRA algorithm with number of inner consensus iterations set to $K = 2$ and the parameter which influences the step size set to $\nu = 100$. The choice was motivated by the the numerical results presented in [36]. The total number

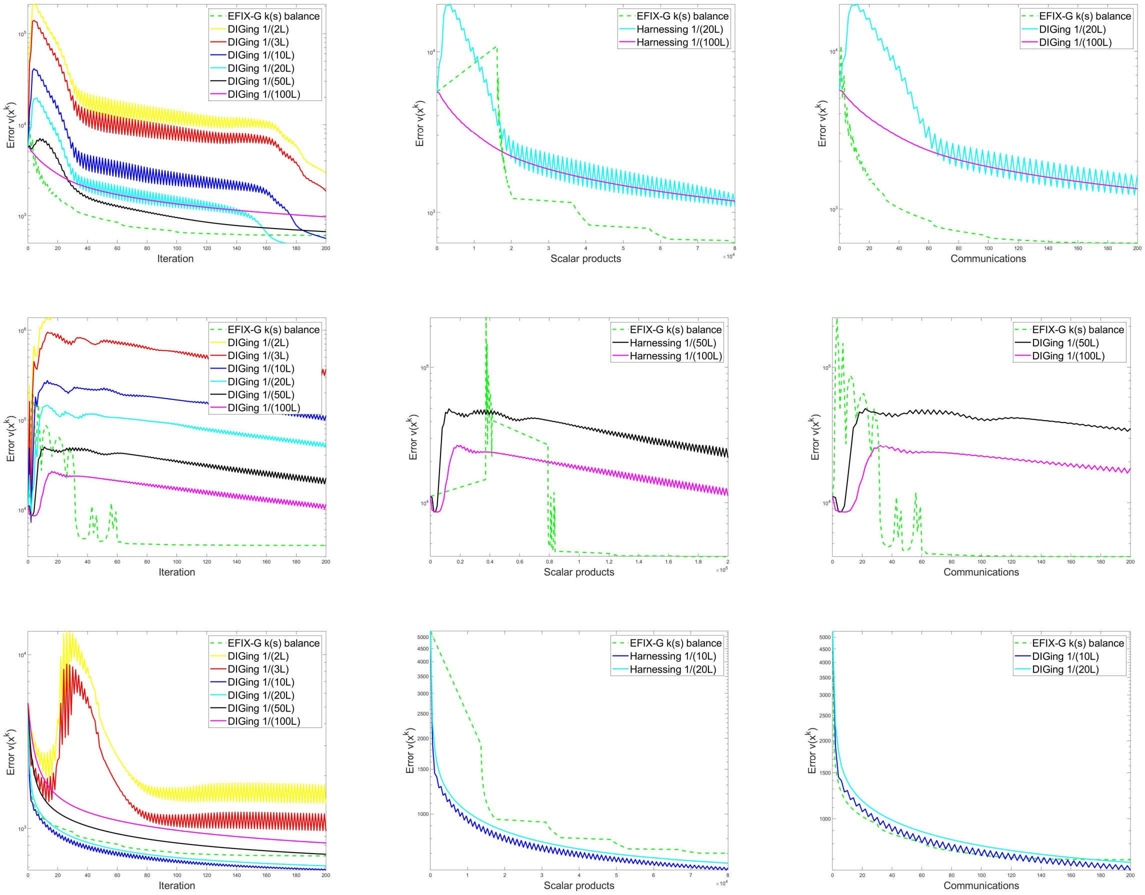

Figure 3: The proposed method (dotted line) versus the DIGing method on Mushrooms (top), CINA0 (middle) and Small MNIST data set (bottom).

of (outer) iterations along which the algorithms will be run is set to $T = 200$. This is also the number of EFIX overall (total number of inner) iterations. Notice that this is relevant for OPTRA as certain OPTRA's parameters explicitly depend on T. On the other hand, for EFIX, no parameter depends on T.

The metrics and the cost measures are retained as in the previous subsections. Using the same logic, we conclude that the communication cost of each OPTRA iteration is $2K$ (i.e., $2K$ times more costly that EFIX iteration). This comes from the fact that each OPTRA iteration calls the inner procedure named AccGossip two times and each AccGossip performs K consensus steps. For the quadratic case, the computational cost (in SPs) is $2Kn\xi + n$ per node per iteration - the cost of calculating the gradient is n SPs per node and the of each consensus step is $n\xi$ SPs. For the logistic regression case, the cost of calculating the gradient is $n\tilde{\xi} + |J_i|$ SPs and we obtain the total cost of $(2K\xi + 1)n + |J_i| + n\tilde{\xi}$ SPs per node, per iteration.

The results are presented at Figures 4-6. Figure 4 represents the results obtained on quadratic costs for the two types of communication graphs and $n = 100, N = 30$. Figures 5-6 correspond to logistic regression problem with datasets Small MNIST, Mushrooms and CINA0. Figure 5 represents the results on well connected graph, while Figure 6 deals with the ring graph.

Notice that, for the considered iteration horizon T, OPTRA achieves a very precise final accuracy for certain experiments, like for datasets Small MNIST and Mushrooms in Figures 5-6 (top and bottom rows). On the other hand, OPTRA seems to saturate at a plateau or progresses very slowly on other experiments, like for the quadratic case in Figure 4. This behavior is not in contradiction with the theory of OPTRA in [36], where the authors are concerned with providing the number of iterations needed to reach a prescribed finite accuracy, and are not concerned with asymptotic convergence as k tends to infinity. (See Theorem 7 in [36]). Both methods exhibit initial oscillatory behavior on CINA0 dataset in Figures 5-6 (middle), but it seems that EFIX stabilizes sooner, while OPTRA continues to oscillate. So, for CINA0 dataset, EFIX method outperforms OPTRA. On the other hand, after the initial advantage of EFIX in terms of communication costs and iterations, OPTRA takes the lead and outperforms EFIX. The advantage of OPTRA is obvious in terms of computational costs measured in SPs (Figures 5-6, middle). Notice that the conclusions are rather similar on both tested graphs. Taking into account all the presented results, the tested methods appear to be competitive.

We also comment on an advantage of EFIX with respect to OPTRA in terms of parameter tuning. Notice that for each problem at hand, we apply the same universal rules to set the EFIX parameters. In contrast, OPTRA has a free parameter $\nu > 0$ that seems to be difficult to tune. In terms of guidelines for setting ν, reference [36] suggests (up to universal constants) a theoretical optimized value of ν. For such value of ν, OPTRA achieves the lower complexity bounds with respect to the oracle defined in [36]; however, the optimized value of ν depends on the gradient of the cost function at the solution and is hence difficult to specify. Reference [36] does not give guidelines how to approximate the optimized value of ν, but it rather hand-tunes ν for each given data set and each given network. Another advantage of EFIX over OPTRA is that OPTRA's parameters τ and γ depend on the total iteration budget T. In other words, for different total iteration budgets T, OPTRA parameters should be set differently. In contrast, EFIX parameters are set universally irrespective of a value of T set beforehand.

6 Conclusions

The quadratic penalty framework is extended to distributed optimization problems. Instead of standard reformulation with quadratic penalty for distributed problems, we define a sequence of quadratic penalty subproblems with increasing penalty parameters. Each subproblem is then approximately solved by a distributed fixed point linear solver. In the paper we used the Jacobi and Jacobi Over-Relaxation method as the linear solvers, to facilitate the explanations. The first class of optimization problems we consider are quadratic problems with positive definite Hessian matrices. For these problems we define the EFIX-Q method, discuss the convergence properties and derive a set of conditions on penalty parameters, linear solver precision and inner iteration number that yield an iterative sequence which converges to the solution of the original, distributed and unconstrained problem. Furthermore, the complexity bound of $\mathcal{O}(\epsilon^{-1})$ is derived. In the case of strongly convex generic function we define EFIX-G method. It follows the reasoning for the quadratic problems and in each outer iteration we define a quadratic model of the objective function and couple that model with the quadratic penalty. Hence, we are again solving a sequence of quadratic subproblems. The convergence statement is weaker in this case but nevertheless corresponds to the classical statement in the centralized penalty methods - we

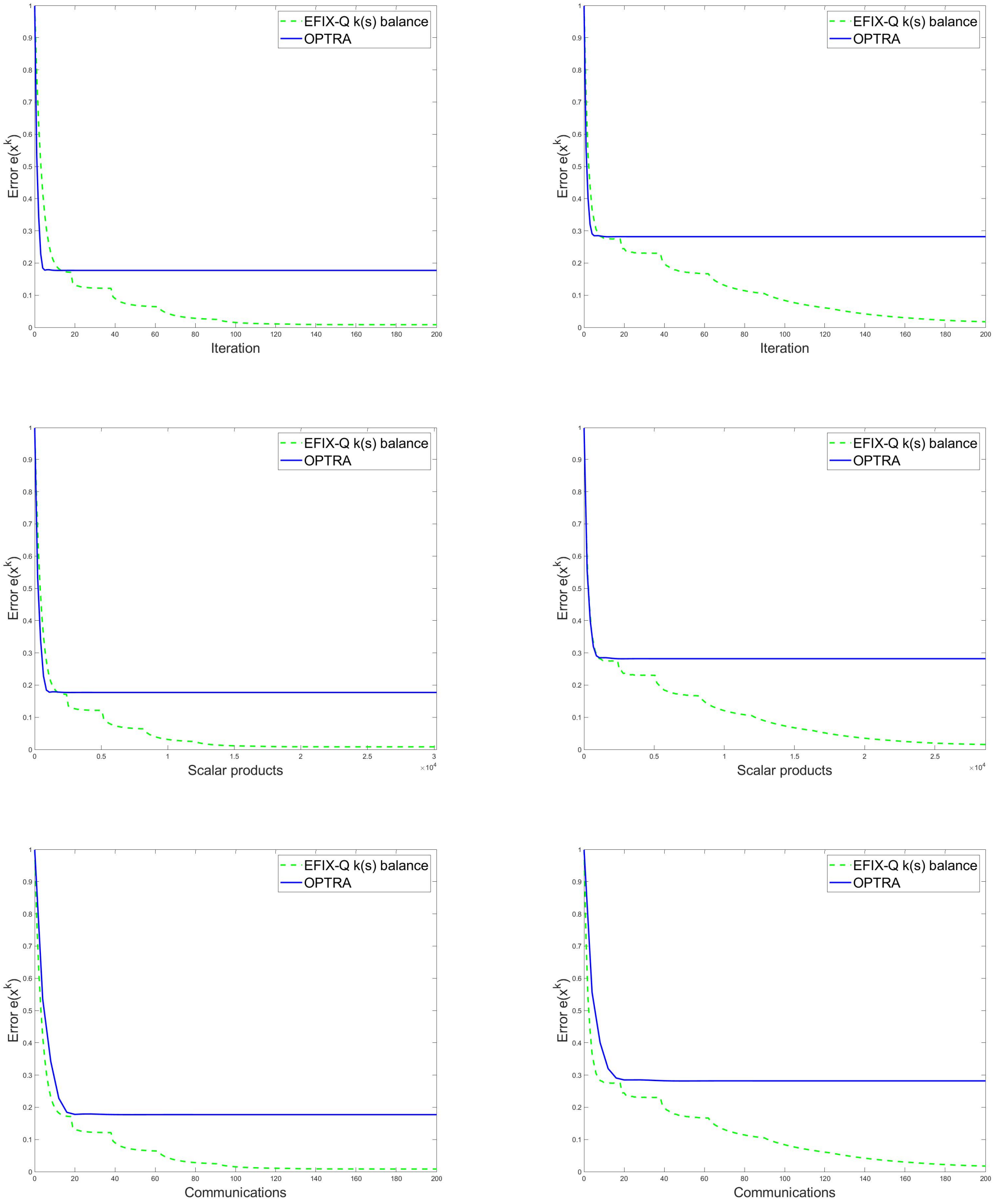

Figure 4: The proposed method (dotted line) versus the OPTRA methods on strongly convex quadratic functions ($n = 100, N = 30$) on well connected graph represented by W (left) and ring graph represented by W_{ring} (right).

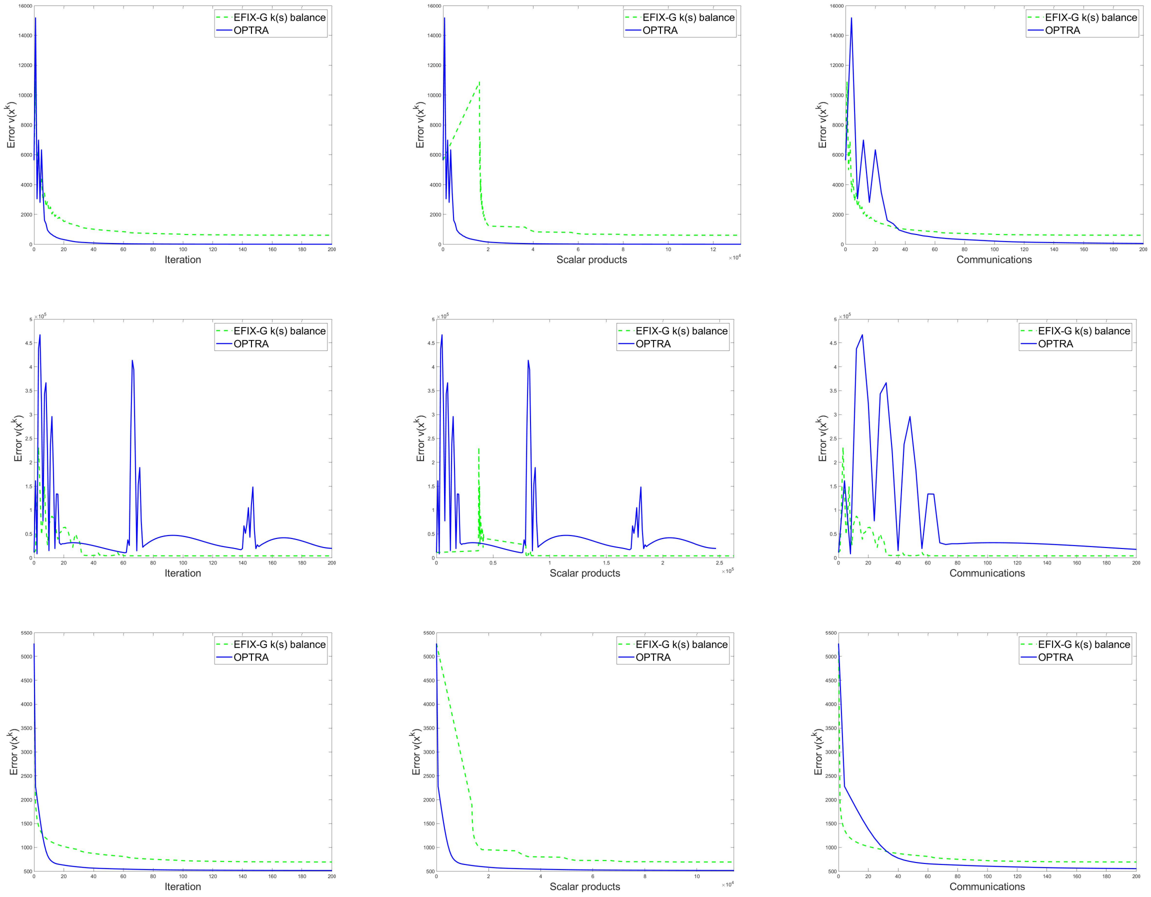

Figure 5: The proposed method (dotted line) versus the OPTRA methods on Mushrooms (top), CINA0 (middle) and Small MNIST data set (bottom) on well connected graph with 30 nodes.

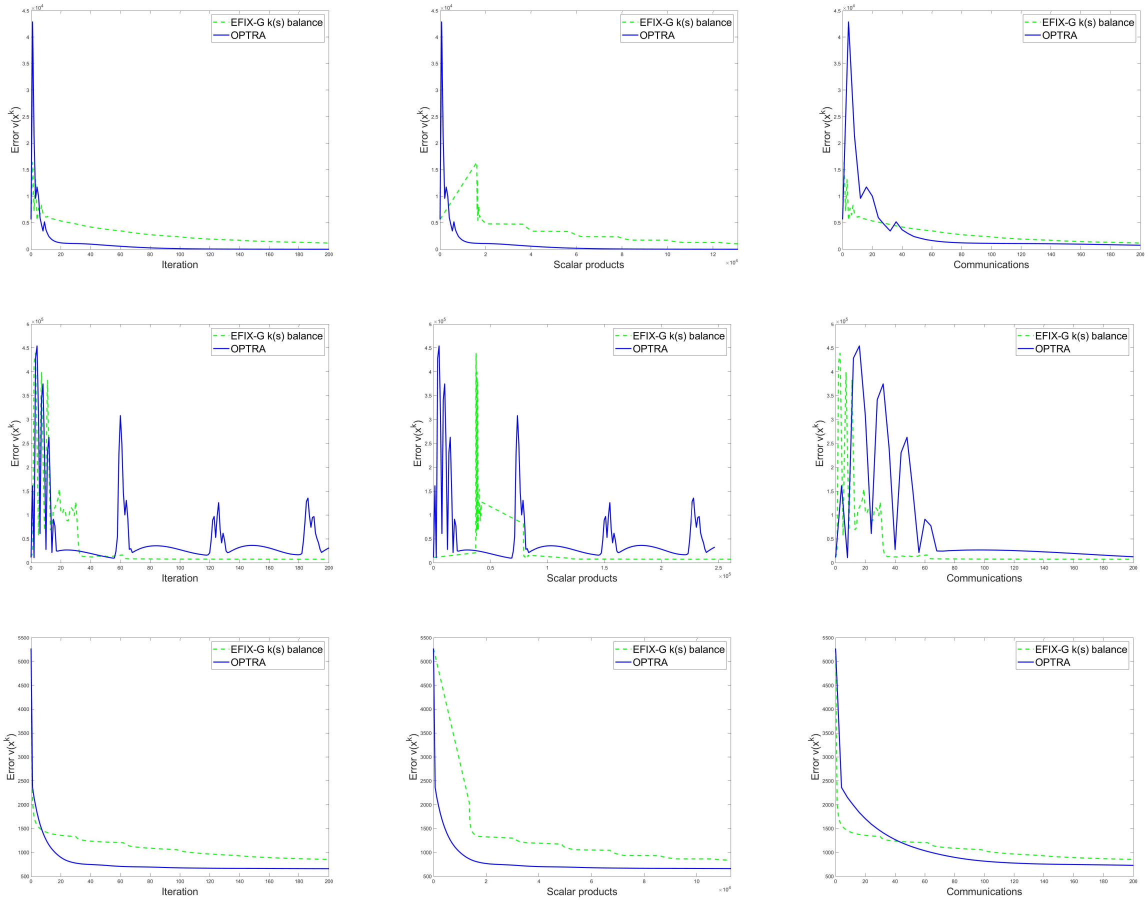

Figure 6: The proposed method (dotted line) versus the OPTRA methods on Mushrooms (top), CINA0 (middle) and Small MNIST data set (bottom) on ring graph with 30 nodes.

prove that if the sequence converges then its limit is a solution of the original problem. The method is dependent on penalty parameters, precision of the linear solver for each subproblem and consequently, the number of inner iterations for subproblems. As quadratic penalty function is not exact, the approximation error is always present and hence we investigated the mutual dependence of different errors. A suitable choice for the penalty parameters, subproblem accuracy and inner iteration number is proposed for quadratic problems and extended to the generic case. The method is tested and compared with the state-of-the-art first order exact method for distributed optimization, DIGing. It is shown that EFIX is comparable with DIGing in terms of error propagation with respect to iterations and that EFIX computational and communication costs are lower in comparison with DIGing methods. EFIX is also compared to recently developed primal-dual method - OPTRA. The comparison is made on both well connected and weekly connected graphs and the EFIX method proves to be at least competitive with the tested counterpart with respect to practical performance, while the advantage of EFIX lies in universal parameter settings.

Acknowledgements

The authors are grateful to the Associate Editor and the referees for comments which helped us to improve the paper.

Funding This work is supported by the Ministry of Education, Science and Technological Development, Republic of Serbia. The work is also supported in part by the European Union's Horizon 2020 Research and Innovation program under grant agreement No 871518.

Availability statement The data sets analyzed during the current study are available in the the MNIST database of handwritten digits (http://yann.lecun.com/exdb/mnist/), LIBSVM Data: Classification (Binary Class, https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html) and UCI Machine Learning Repository (https://archive.ics. uci.edu/ml/index.php).

References

[1] Baingana, B., Giannakis, G., B., Joint Community and Anomaly Tracking in Dynamic Networks, IEEE Transactions on Signal Processing,

64(8), (2016), pp. 2013-2025.

[2] A. S. Berahas, A. S., Bollapragada, R., Keskar, N. S., Wei, E., Balancing Communication and Computation in Distributed Optimization, IEEE Transactions on Automatic Control, 64(8), (2019), pp. 3141-3155.

[3] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine Learning, 3(1), (2011) pp. 1-122.

[4] Cattivelli, F., Sayed, A. H., Diffusion LMS strategies for distributed estimation, IEEE Transactions on Signal Processing, 58(3), (2010) pp. 1035–1048.

[5] Causality workbench team, a marketing dataset, http://www.causality.inf.ethz.ch/data/CINA.html.

[6] Di Lorenzo, P., Scutari, G., Distributed nonconvex optimization over networks, in IEEE International Conference on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), (2015), pp. 229-232.

[7] Fodor, L., Jakoveti, D., Kreji, N., Krklec Jerinki, N., Skrbi, S., Performance evaluation and analysis of distributed multi-agent optimization algorithms with sparsified communication, EURASIP Journal on Advances in Signal Processing, 25, (2021), https://doi.org/10.1186/s13634-021-00736-4.

[8] Greenbaum, A., Iterative Methods for Solving Linear Systems, SIAM, 1997.

[9] Jakovetić, D., A Unification and Generalization of Exact Distributed First Order Methods, IEEE Transactions on Signal and Information Processing over Networks, 5(1), (2019), pp. 31-46.

[10] Jakovetić, D., Krejić, N., Krklec Jerinkić, N. , Malaspina, G. , Micheletti, A., Distributed Fixed Point Method for Solving Systems of Linear Algebraic Equations, arXiv:2001.03968, (2020).

[11] Jakovetić, D., Xavier, J., Moura, J. M. F., Fast distributed gradient methods, *IEEE Transactions on Automatic Control*, 59(5), (2014) pp. 1131–1146.

[12] Jakovetić, D., Krejić, N., Krklec Jerinkić, N., Exact spectral-like gradient method for distributed optimization, Computational Optimization and Applications, 74, (2019), pp. 703728.

[13] Lee, J. M., Song, I., Jung, S., Lee, J., A rate adaptive convolutional coding method for multicarrier DS/CDMA systems, MILCOM 2000 Proceedings 21st Century Military Communications. Architectures and Technologies for Information Superiority (Cat. No.00CH37155), Los Angeles, CA, (2000), pp. 932-936.

[14] Li, H., Fang, C., Yin, W., Lin, Z., Decentralized Accelerated Gradient Methods With Increasing Penalty Parameters, IEEE Transactions on Signal Processing, 68, pp. 4855-4870, (2020).

[15] Li, H., Fang, C., Lin, Z., Convergence Rates Analysis of The Quadratic Penalty Method and Its Applications to Decentralized Distributed Optimization, arxiv preprint, arXiv:1711.10802, (2017).

[16] Mota, J., Xavier, J., Aguiar, P., Püschel, M., Distributed optimization with local domains: Applications in MPC and network flows, IEEE Transactions on Automatic Control, 60(7), (2015), pp. 2004-2009.

[17] Nedic, A., Olshevsky, A., Shi, W., Uribe, C.A., Geometrically convergent distributed optimization with uncoordinated step-sizes, 2017 American Control Conference (ACC), Seattle, WA, 2017, pp. 3950-3955, doi: 10.23919/ACC.2017.7963560

[18] Nedić, A., Ozdaglar, A., Distributed subgradient methods for multiagent optimization, IEEE Transactions on Automatic Control, 54(1), (2009), pp. 48–61.

[19] Nocedal, J., Wright, S. J., Numerical Optimization, Springer, 1999.

[20] Outlier Detection Datasets (ODDS) http://odds.cs.stonybrook.edu/mnist-dataset/.

[21] Qu, G., Li, N., Harnessing smoothness to accelerate distributed optimization, IEEE Transactions on Control of Network Systems, 5(3), (2018), pp. 1245-1260.

[22] Saadatniaki, F., Xin, R., Khan, U. A., Decentralized optimization over time-varying directed graphs with row and column-stochastic matrices, IEEE Transactions on Automatic Control, (2018).

[23] Scutari, G., Sun, Y., Parallel and Distributed Successive Convex Approximation Methods for Big-Data Optimization, arXiv:1805.06963, (2018).

[24] Scutari, G., Sun, Y., Distributed Nonconvex Constrained Optimization over Time-Varying Digraphs, Mathematical Programming, 176(1-2), (2019), pp. 497-544.

[25] Shi, W., Ling, Q., Wu, G., Yin, W., EXTRA: an Exact First-Order Algorithm for Decentralized Consensus Optimization, SIAM Journal on Optimization, 2(25), (2015), pp. 944-966.

[26] Shi, G., Johansson, K. H., Finite-time and asymptotic convergence of distributed averaging and maximizing algorithms, arXiv:1205.1733, (2012).

[27] Srivastava, P., Corts, J., Distributed Algorithm via Continuously Differentiable Exact Penalty Method for Network Optimization, 2018 IEEE Conference on Decision and Control (CDC), Miami Beach, FL, (2018), pp. 975-980.

[28] Sun, Y., Daneshmand, A., Scutari, G., Convergence Rate of Distributed Optimization Algorithms based on Gradient Tracking, arXiv:1905.02637, (2019).

[29] Sundararajan, A., Van Scoy, B., Lessard, L., Analysis and Design of First-Order Distributed Optimization Algorithms over Time-Varying Graphs, arXiv:1907.05448, (2019).

[30] Tian, Y., Sun, Y., Scutari, G., Achieving Linear Convergence in Distributed Asynchronous Multi-agent Optimization, IEEE Trans. on Automatic Control, (2020).

[31] Tian, Y., Sun, Y., Scutari, G., Asynchronous Decentralized Successive Convex Approximation, arXiv:1909.10144, (2020).

[32] UCI Machine Learning Expository, https://archive.ics.uci.edu/ml/datasets/Mushroom.

[33] Xiao, L., Boyd, S. and Lall, S., Distributed average consensus with time-varying metropolis weights, Automatica, (2006).

[34] Xin, R., Khan, U. A., Distributed Heavy-Ball: A Generalization and Acceleration of First-Order Methods With Gradient Tracking, IEEE Transactions on Automatic Control, 65(6), (2020), pp. 2627-2633.

[35] Xin, R., Xi, C., Khan, U. A., FROST–Fast row-stochastic optimization with uncoordinated step-sizes, EURASIP Journal on Advances in Signal Processing, Special Issue on Optimization, Learning, and Adaptation over Networks, 1, (2019).

[36] Xu, J., Tian, Y., Sun, Y., Scutari G., Accelerated primal-dual algorithmsfor distributed smooth convex optimization over networks, International Conference on Artificial Intelligence and Statistics, PMLR, (2020), pp. 2381-2391.

[37] Xu, J., Tian, Y., Sun, Y., Scutari G., Distributed Algorithms for Composite Optimization: Unified Framework and Convergence Analysis, arXiv:2002.11534, (2020).

[38] Yuan, K., Ling, Q., Yin, W., On the convergence of decentralized gradient descent, SIAM Journal on Optimization 26(3), (2016), pp. 18351854.

[39] Yousefian, F., Nedić, A., Shanbhag, U. V., On stochastic gradient and subgradient methods with adaptive steplength sequences, Automatica, 48(1), (2012), pp. 56-67.

[40] Zhou, H., Zeng, X., Hong, Y., Adaptive Exact Penalty Design for Constrained Distributed Optimization, IEEE Transactions on Automatic Control, 64(11), (2019), pp. 4661-4667.

7 Appendix

Proof of Lemma 3.2

Proof. Notice that $\mathbb{A}(\theta)$ is positive definite for all $\theta > 0$ and thus there exists an unique stationary point $\mathbf{x}_\theta^*$ of Φ_θ, i.e., an unique solution of $\mathbb{A}(\theta)\mathbf{x} = \mathbf{c}$. With notation $\mathbf{z}^k = (z_1^k; \dots ; z_N^k), \mathbf{z}^0 = \mathbf{x}^s$, we have

$$\begin{aligned} \|\nabla \Phi_{\theta_{s+1}}(\mathbf{z}^k)\| &= \|\nabla \Phi_{\theta_{s+1}}(\mathbf{z}^k) - \nabla \Phi_{\theta_{s+1}}(\mathbf{x}_{\theta_{s+1}}^*)\| \\ &\leq \|\mathbb{A}(\theta_{s+1})\| \|\mathbf{z}^k - \mathbf{x}_{\theta_{s+1}}^*\| \\ &\leq (L + 2\theta_{s+1})\|\mathbf{z}^k - \mathbf{x}_{\theta_{s+1}}^*\| \\ &\leq (L + 2\theta_{s+1})\rho_{s+1}^k \|\mathbf{x}^s - \mathbf{x}_{\theta_{s+1}}^*\| \\ &\leq (L + 2\theta_{s+1})\rho_{s+1}^k (\|\mathbf{x}^s - \mathbf{x}_{\theta_s}^*\| + \|\mathbf{x}_{\theta_s}^* - \mathbf{x}_{\theta_{s+1}}^*\|). \end{aligned} \tag{26}$$

Let us now estimate the norms in the final inequality. First, notice that

$$\nabla \Phi_{\theta_s}(\mathbf{x}^s) = \nabla \Phi_{\theta_s}(\mathbf{x}^s) - \nabla \Phi_{\theta_s}(\mathbf{x}_{\theta_s}^*) = \mathbb{A}(\theta_s)(\mathbf{x}^s - \mathbf{x}_{\theta_s}^*).$$

Thus, since $\mu \mathbb{I} \preceq \mathbb{A}(\theta_s)$ we obtain

$$\|\mathbf{x}^s - \mathbf{x}_{\theta_s}^*\| \leq \|\mathbb{A}^{-1}(\theta_s)\| \|\nabla \Phi_{\theta_s}(\mathbf{x}^s)\| \leq \frac{\varepsilon_s}{\mu}. \tag{27}$$

Moreover, for any θ we have

$$\|\mathbf{x}_\theta^*\| \leq \|\mathbb{A}^{-1}(\theta)\| \|\mathbf{c}\| \leq \frac{\bar{c}}{\mu}. \tag{28}$$

Putting (27) and (28) into (26) we obtain

$$\|\nabla \Phi_{\theta_{s+1}}(\mathbf{z}^k)\| \leq \frac{(L + 2\theta_{s+1})\rho_{s+1}^k(\varepsilon_s + 2\bar{c})}{\mu}.$$

Imposing the inequality

$$\frac{(L + 2\theta_{s+1})\rho_{s+1}^k(\varepsilon_s + 2\bar{c})}{\mu} \leq \varepsilon_{s+1},$$

and then applying the logarithm and rearranging, we obtain that $\|\nabla \Phi_{\theta_{s+1}}(\mathbf{z}^k)\| \leq \varepsilon_{s+1}$ for all $k \geq k(s)$ defined by (14). Therefore, for $\mathbf{z}^{k(s)} = \mathbf{x}^{s+1}$ we get the statement. □

Proof of Theorem 3.1

Proof. Assume that $\{\mathbf{x}^s\}$ is bounded and consider the problem (2), i.e.,

$$\min F(\mathbf{x}), \text{ s.t. } h(\mathbf{x}) = 0$$

where

$$h(\mathbf{x}) = \mathbb{L}^{1/2}\mathbf{x}.$$

Let $\mathbf{x}^*$ be an arbitrary accumulation point of the bounded sequence $\{\mathbf{x}^s\}$ generated by algorithm EFIX-Q, i.e., let

$$\lim_{s \in K_1} \mathbf{x}^s = \mathbf{x}^*.$$

The inequality (13) implies

$$\theta_s \|\nabla^T h(\mathbf{x}^s) h(\mathbf{x}^s)\| - \|\nabla F(\mathbf{x}^s)\| \leq \varepsilon_s. \tag{29}$$

Since $\nabla^T h(\mathbf{x}^s) = (\mathbb{L}^{1/2})^T = \mathbb{L}^{1/2}$, we obtain

$$\nabla^T h(\mathbf{x}^s) h(\mathbf{x}^s) = \mathbb{L}\mathbf{x}^s,$$

and (29) implies

$$\|\mathbb{L}\mathbf{x}^s\| \leq \frac{1}{\theta_s}(\|\nabla F(\mathbf{x}^s)\| + \varepsilon_s). \tag{30}$$

Taking the limit over K_1 we have $\mathbb{L}\mathbf{x}^* = 0$, i.e., $h(\mathbf{x}^*) = 0$, so $\mathbf{x}^*$ is a feasible point. Therefore $\mathbb{W}\mathbf{x}^* = \mathbf{x}^*$, or equivalently $x_1^* = x_2^* = ... = x_N^*$, so the consensus is achieved.

Now, we prove that $\mathbf{x}^*$ is an optimal point of problem (2). Let us define $\lambda_s := \theta_s h(\mathbf{x}^s)$. Considering the gradient of the penalty function we obtain

$$\nabla \Phi_{\theta_s}(\mathbf{x}^s) = \nabla F(\mathbf{x}^s) + \theta_s \mathbb{L}\mathbf{x}^s = \nabla F(\mathbf{x}^s) + \mathbb{L}^{1/2}\lambda_s. \tag{31}$$

Since $\mathbf{x}^s \to \mathbf{x}^*$ over K_1 and $\varepsilon_s \to 0$, from (30) we conclude that $\zeta_s := \theta_s \mathbb{L}\mathbf{x}^s$ must be bounded over K_1. Therefore, $\lambda_s = \theta_s \mathbb{L}^{1/2}\mathbf{x}^s$ is also bounded over K_1 and thus, there exist $K_2 \subseteq K_1$ and λ^* such that

$$\lim_{s \in K_2} \lambda_s = \lambda^*. \tag{32}$$

Indeed, by the eigenvalue decomposition, we obtain $\mathbb{L} = \mathbb{U}\mathbb{V}\mathbb{U}^T$, where $\mathbb{U}$ is an unitary matrix and $\mathbb{V}$ is the diagonal matrix with eigenvalues of $\mathbb{L}$. Let us denote them by v_i. The matrix is positive semidefinite, so $v_i \geq 0$ for all

i and we also know that $\mathbb{L}^{1/2} = \mathbb{U}\mathbb{V}^{1/2}\mathbb{U}^T$. Since ζ_s is bounded over K_1, the same is true for the sequence $\mathbb{U}^T\zeta_s = \mathbb{V}\theta_s\mathbb{U}^T\mathbf{x}^s := \mathbb{V}\nu^s$. Consequently, all the components $v_i[\nu^s]_i$ are bounded over K_1 and the same is true for $\sqrt{v_i}[\nu^s]_i$. By unfolding we get that $\mathbb{V}^{1/2}\theta_s\mathbb{U}^T\mathbf{x}^s$ is bounded over K_1 and thus the same holds for

$$\mathbb{U}\mathbb{V}^{1/2}\theta_s\mathbb{U}^T\mathbf{x}^s = \theta_s\mathbb{L}^{1/2}\mathbf{x}^s = \lambda_s.$$

Now, using (32) and taking the limit over K_2 in (31) we get

$$0 = \nabla F(\mathbf{x}^*) + \mathbb{L}^{1/2}\lambda^*,$$

i.e., $\nabla F(\mathbf{x}^*) + \nabla^T h(\mathbf{x}^*)\lambda^* = 0$, which means that $\mathbf{x}^*$ is a KKT point of problem (2) with λ^* being the corresponding Lagrange multiplier. Since F is assumed to be strongly convex, $\mathbf{x}^*$ is also a solution of the problem (2). Finally, notice that x_i^* is a solution of the problem (1) for any given node $i = 1, ..., N$.

We have just proved that, for an arbitrary i, every accumulation point of the sequence $\{x_i^s\}$ is the solution of problem (1). Since the function f is strongly convex, the solution of problem (1) must be unique. So, assuming that there exist accumulation points $\mathbf{x}^*$ and $\tilde{\mathbf{x}}$ such that $\mathbf{x}^* \neq \tilde{\mathbf{x}}$ yields contradiction. Therefore we conclude that all the accumulation points must be the same, i.e., the sequence $\{\mathbf{x}^s\}$ converges. This completes the proof. □

Proof of Corollary 3.1

Proof. Notice that (16), (15) and (27) imply for arbitrary i

$$\begin{aligned}\|x_i^s - y^*\| &\leq \frac{\varepsilon_s}{\mu} + e^1_{i,\theta_s} \leq 2\left(\frac{LJ}{\theta_s\kappa(1-\lambda_2)}\sqrt{4 - 2\kappa\theta_s^{-1}} + \frac{J}{\theta_s(1-\lambda_2)}\right)\\ &\leq \frac{2J}{\theta_s(1-\lambda_2)}\left(\frac{2(\mu+L)}{\mu} + 1\right) \leq \frac{2J}{\theta_s(1-\lambda_2)}(3 + 2L/\mu).\end{aligned}$$

For $\theta_s = s$, the right-hand side of the above inequality is smaller than ϵ for

$$s \geq \frac{2J(3+2L/\mu)}{(1-\lambda_2)}\epsilon^{-1} \tag{33}$$

which completes the proof. □

Proof of Lemma 4.1

Proof. Notice that Q_s is strongly convex for all $\theta > 0$, i.e., for all s. Moreover, its Hessian $\mathbb{A}_s$ satisfies $\mu\mathbb{I} \preceq \mathbb{A}_s \preceq (L + 2\theta_s)\mathbb{I}$. Thus, there exists an unique stationary point $\mathbf{x}^*_{\theta_s}$ of Q_s, i.e., an unique solution of $\mathbb{A}_s\mathbf{x} = \mathbf{c}_s$. With notation $\mathbf{z}^k = (z^k_1; \ldots ; z^k_N)$, $\mathbf{z}^0 = \mathbf{x}^s$, we have

$$\begin{aligned} \|\nabla Q_{s+1}(\mathbf{z}^k)\| &= \|\nabla Q_{s+1}(\mathbf{z}^k) - \nabla Q_{s+1}(\mathbf{x}^*_{\theta_{s+1}})\| \\ &\leq \|\mathbb{A}_{s+1}\|\|\mathbf{z}^k - \mathbf{x}^*_{\theta_{s+1}}\| \\ &\leq (L + 2\theta_{s+1})\|\mathbf{z}^k - \mathbf{x}^*_{\theta_{s+1}}\| \\ &\leq (L + 2\theta_{s+1})\rho^k_{s+1}\|\mathbf{x}^s - \mathbf{x}^*_{\theta_{s+1}}\| \\ &\leq (L + 2\theta_{s+1})\rho^k_{s+1}(\|\mathbf{x}^s - \mathbf{x}^*_{\theta_s}\| + \|\mathbf{x}^*_{\theta_s} - \mathbf{x}^*_{\theta_{s+1}}\|). \end{aligned} \tag{34}$$

Let us now estimate the norms in the final inequality. First, notice that

$$\nabla Q_s(\mathbf{x}^s) = \nabla Q_s(\mathbf{x}^s) - \nabla Q_s(\mathbf{x}^*_{\theta_s}) = \mathbb{A}_s(\mathbf{x}^s - \mathbf{x}^*_{\theta_s}).$$

Thus, we obtain

$$\|\mathbf{x}^s - \mathbf{x}^*_{\theta_s}\| \leq \|\mathbb{A}_s^{-1}\|\|\nabla Q_s(\mathbf{x}^s)\| \leq \frac{\varepsilon_s}{\mu}. \tag{35}$$

Moreover, for any s we have

$$\|\mathbf{x}^*_{\theta_s}\| \leq \|\mathbb{A}_s^{-1}\|\|\mathbf{c}_s\| \leq \frac{\bar{c}_s}{\mu}. \tag{36}$$

Putting (35) and (36) into (34) we obtain

$$\|\nabla Q_{s+1}(\mathbf{z}^k)\| \leq \frac{(L + 2\theta_{s+1})\rho^k_{s+1}(\varepsilon_s + \bar{c}_s + \bar{c}_{s+1})}{\mu}.$$

Imposing the inequality

$$\frac{(L + 2\theta_{s+1})\rho^k_{s+1}(\varepsilon_s + \bar{c}_s + \bar{c}_{s+1})}{\mu} \leq \varepsilon_{s+1},$$

and then applying the logarithm and rearranging, we obtain that $\|\nabla Q_{s+1}(\mathbf{z}^k)\| \leq \varepsilon_{s+1}$ for all $k \geq k(s)$ defined by (14). Therefore, for $\mathbf{z}^{k(s)} = \mathbf{x}^{s+1}$ we get the statement. □

Proof of Theorem 4.1

Proof. Let us consider the problem (2) and denote $h(\mathbf{x}) = \mathbb{L}^{1/2}\mathbf{x}$. Let $\tilde{\mathbf{x}} = \lim_{s \in K} \mathbf{x}^s$ be an arbitrary accumulation point. Notice that for the penalty function Φ_{θ_s} defined in (3) there holds

$$\begin{aligned} & \|\nabla \Phi_{\theta_s}(\mathbf{x}^s) - \nabla Q_s(\mathbf{x}^s)\| \\ = \;& \|\nabla F(\mathbf{x}^s) - \nabla F(\mathbf{x}^{s-1}) + \nabla^2 F(\mathbf{x}^{s-1})(\mathbf{x}^s - \mathbf{x}^{s-1})\| \\ \leq \;& 2L\|\mathbf{x}^s - \mathbf{x}^{s-1}\| := r_s \end{aligned} \tag{37}$$

and thus the error of the quadratic model r_s is also bounded over K. Now, inequality (19) together with the previous inequality implies that

$$\|\nabla \Phi_{\theta_s}(\mathbf{x}^s)\| \leq \varepsilon_s + r_s, \tag{38}$$

i.e., we obtain

$$\|\mathbb{L}\mathbf{x}^s\| \leq \frac{1}{\theta_s}(\|\nabla F(\mathbf{x}^s)\| + \varepsilon_s + r_s).$$

Taking the limit over K in the previous inequality, we conclude that $\mathbb{L}\tilde{\mathbf{x}} = 0$, so the feasibility condition is satisfied, i.e., we have $\tilde{x}_1 = \tilde{x}_2 = ... = \tilde{x}_N$.

If $\lim_{s \to \infty} \mathbf{x}^s = \mathbf{x}^*$ we have that the error in quadratic model converges to zero from (37), i.e. $\lim_{s \to \infty} r_s = 0$ and thus (38) implies that

$$\lim_{s \in K} \nabla \Phi_{\theta_s}(\mathbf{x}^s) = 0.$$

Following the same steps as in the second part of the proof of Theorem 3.1, we conclude that $\mathbf{x}^*$ is optimal and the statement follows. □