Distributed, scalable and gossip-free consensus optimization with application to data analysis

Sina Khoshfetrat Pakazad, Christian A. Naesseth, Fredrik Lindsten, Anders Hansson, Member, IEEE

Abstract—Distributed algorithms for solving additive or consensus optimization problems commonly rely on first-order or proximal splitting methods. These algorithms generally come with restrictive assumptions and at best enjoy a linear convergence rate. Hence, they can require many iterations or communications among agents to converge. In many cases, however, we do not seek a highly accurate solution for consensus problems. Based on this we propose a controlled relaxation of the coupling in the problem which allows us to compute an approximate solution, where the accuracy of the approximation can be controlled by the level of relaxation. The relaxed problem can be efficiently solved in a distributed way using a combination of primal-dual interior-point methods (PDIPMs) and message-passing. This algorithm purely relies on second-order methods and thus requires far fewer iterations and communications to converge. This is illustrated in numerical experiments, showing its superior performance compared to existing methods.

Index Terms—Distributed optimization, data analysis, consensus, primal-dual method, data analysis.

I. INTRODUCTION

Many optimization problems in e.g. machine learning, control and signal processing [3], [9], [3], [19] can be formulated as

$$\min_x \frac{1}{N} \sum_{i=1}^{N} f_i(x) + g_i(x),$$

(1)

where the convex functions $f_i : \mathbb{R}^p \to \mathbb{R}$ and $g_i : \mathbb{R}^p \to \mathbb{R}$ are smooth and non-smooth, respectively, such that $F_i = f_i + g_i$ is Lipschitz continuous. Here we assume that $x \in \mathbb{R}^p$, where $p$ is not overly large, whereas $N$ can be potentially large.

It is sometimes impossible to solve these problems using centralized optimization algorithms. This is commonly due to computational issues, e.g., when $N$ is very large, or due to privacy requirements. In these cases, a solution is provided by distributed algorithms, which solve the optimization problem using a network of computational agents that can collaborate and communicate with one another. These algorithms commonly rely on consensus formulations of the problem and are typically based on first-order splitting methods, see e.g., [2], [17], [5], [20], [3], [6]. Thus, these methods are slow with sub-linear or at best linear convergence rates, [3], [20], [17]. Moreover, they sometimes require further assumptions, such as smoothness or strong convexity of the cost function, and are commonly sensitive to the scaling of the problem. Among these algorithms the ones based on proximal point methods or proximal method of multipliers are less sensitive to scaling, see e.g., [7], [3], [6], and require less iterations to converge. However, each iteration is generally far more computationally demanding than that of gradient-based or subgradient-based algorithms.

In order to reduce the required number of iterations and sensitivity to scaling, attempts have been made to combine first and second order methods, see e.g., [1], [21], [16]. However due to reliance on first-order methods, these algorithms still require many iterations to converge. In a distributed setting, this means that they require many communications among computational agents, which gives rise to non-negligible communication overhead. In order to address all these issues, we set out to devise algorithms that solely rely on second-order methods, which allow for (i) more efficient handling of the problem data, (ii) convergence in few iterations, and (iii) efficient use of parallel/distributed computational platforms and cloud/edge computing for solving the problem. To reach this goal, we face two main hurdles. Firstly, using second-order methods for solving (1) in a distributed or parallelized manner is generally not possible due to the fact that the subproblems in (1) are all fully coupled. Secondly, second-order methods cannot be applied directly for solving this problem, due to the fact that the functions $F_i$ are non-differentiable or not continuously differentiable.

In this paper, we show how these hurdles can be overcome. In Section II we present a controlled relaxation to the coupling among the subproblems in (1). By solving the relaxed formulation we obtain an approximate solution for (1), where the accuracy of the approximation is controlled by the level of relaxation. This is motivated by the fact that in many applications we do not seek an exact solution to (1), for instance due to the uncertainty present in the problem data. A similar relaxation was considered in [13], but based on a different motivation and and for handling streamed data.

Next, we propose to use primal-dual interior-point methods (PDIPMs) (reviewed in Section III) for solving the relaxed problem. The convergence of these methods is well-established, see e.g., [4], [22]. The proposed relaxation allows us to impose a coupling structure on the problem that can be represented as a tree. This opens up for using message-passing to compute the search directions and other parameters exactly through conducting recursions over the tree structure (see also (1)). Hence, distributing the computations does not jeopardize the convergence of the PDIPM. Message-passing is briefly discussed in Section IV. The resulting algorithm provides superior performance in comparison to recently developed high-performance distributed algorithms, as shown in Section V using multiple numerical experiments.

II. A CONTROLLED RELAXATION AND REFORMULATION

In order to impose a desirable structure on (1), let us consider a relaxation of the problem given as

$$\min_{x,x^i} \frac{1}{N} \sum_{i=1}^{N} F_i(x^i)$$

(2a)

subject to $\|x - x^i\|^2 \leq \varepsilon^2, \quad i = 1, \ldots, N.$

(2b)

Note that the terms in (2a) are decoupled. The approximate coupling among these terms are now described using the
Algorithm 1: Primal-dual Interior-point Method [22, 4]

1: Given feasible iterates
2: repeat
3: Compute the primal-dual search directions
4: Compute appropriate primal and dual step sizes
5: Update primal and dual iterates
6: Update the perturbation parameter
7: until stopping criteria is satisfied

Constraints in (25). Let us denote an optimal solution of the problem in (1) with \( x^* \) and that of (2) with \( x^*_{\text{rel}} \) and \( x^{i*} \). It is possible to compute satisfactory suboptimal solutions for (1) by solving (2), as quantified by the following theorem.

**Theorem I:** Let us assume that the Lipschitz constant for each \( F_i \) is denoted by \( L_i \). Then we have

\[
\frac{1}{N} \sum_{i=1}^{N} F_i(x^*_{\text{rel}}) - F_i(x^*) < \frac{\varepsilon}{N} L_i
\]

where \( L = \sum_{i=1}^{N} L_i \). Furthermore, if the cost function is strongly convex with modulus \( m \), we have \( \|x^*_{\text{rel}} - x^*\| \leq \frac{\varepsilon}{NmL} \).

Proof 1: See Appendix B.

If the tolerated suboptimality of the solution is \( \varepsilon_{\text{tol}} \), choosing \( \varepsilon = N\varepsilon_{\text{tol}}/L \) guarantees that \( x^*_{\text{rel}} \) gives a satisfactory solution. Moreover, if the problem is strongly convex, given a threshold \( \varepsilon_{\text{var}} \) concerning the accuracy of the solution, if we choose \( \varepsilon = \frac{Nm^{\text{var}}}{2L} \), we can guarantee that the obtained solution will satisfy the accuracy requirements. It goes without saying that the smaller the \( \varepsilon \), the more accurate the computed solution. However, care must be taken as choosing extremely small values for this parameter can give rise to numerical issues. In general, e.g., provided that the data is normalized, we can compute accurate enough solutions using moderately small values of \( \varepsilon \), see Section V.

In this paper, we devise a distributed algorithm for solving the relaxed problem (2), purely relying on second-order methods. Due to non-smoothness of the objective function, however, second-order algorithms cannot be directly applied. Instead, we introduce additional variables and constraints in order to equivalently reformulate the problem as (see e.g., [4]).

**Algorithm 2:**

\[
\min_{x, z, t^i} \frac{1}{N} \sum_{i=1}^{N} h_i(x^i, t^i)
\]

subject to

\[
G^i(x^i, t^i) \leq 0, \quad i = 1, \ldots, N
\]

\[
A^i x = b^i, \quad i = 1, \ldots, N
\]

\[
\|x - x^i\| \leq \varepsilon^2, \quad i = 1, \ldots, N
\]

where \( h_i : \mathbb{R}^{p+d_i} \to \mathbb{R} \) is smooth, the variables \( t^i \in \mathbb{R}^{d_i} \) denote the additional variables, and the constraints in (4b) and (4c) are the additional inequality and equality constraints, with \( A^i \in \mathbb{R}^{n_i \times (p+d_i)} \) and \( G^i : \mathbb{R}^{p+d_i} \to \mathbb{R}^{m_i} \). For instance when the non-smooth terms in the objective function are indicator functions for convex sets, the problem can be reformulated by removing these terms and adding the corresponding constraints to the problem. Another common approach for reformulating e.g., problems of the form in (4) as in [4], is through the use of epigraph reformulations [4].

Other approaches are also possible, e.g., [4], but for the sake of brevity we do not discuss them here.

III. PRIMAL-DUAL INTERIOR-POINT METHODS

PDIPMs are the state-of-the-art iterative solvers for problems like (4). A generic description of these methods is given in Algorithm 1 [22, 4]. Let us denote the dual variables for the constraints in (4b), (4c) and (4d) with \( z^i, v^i \) and \( \lambda_i \), respectively. At each iteration \( k \), given feasible primal and dual iterates, i.e., such that \( G^i(x^{i,(k)}, t^{i,(k)}) < 0 \), \( \|x^{i,(k)} - x^{i,(k)}\| < \varepsilon^2 \), \( z^{i,(k)} > 0 \) and \( \lambda_i^{(k)} > 0 \), one way of computing the search directions requires solving an equality constrained quadratic program, see [11]. Particularly, for the problem in (4), the QP that needs to be solved takes the form

\[
\begin{align*}
\minimize_{x, z^i} & \quad \sum_{i=1}^{N} \left[ \frac{1}{2} \|\Delta x^i\|^2 + H_{pd}^{i,(k)} \frac{\partial}{\partial x^i} \|\Delta x^i\|^2 \right] + H_{lg}^{i,(k)} \frac{\partial}{\partial x^i} \|\Delta x^i\| \\text{subject to} \quad A^i \frac{\partial}{\partial x^i} \|\Delta x^i\| = x^{i,(k)} - x^{i,(k)} \end{align*}
\]

Through solving (5) we can compute the primal directions \( \Delta x^{i,(k+1)} \), \( \Delta t^{i,(k+1)} \) and \( \Delta x^{i,(k+1)} \) together with the dual directions \( \Delta z^{i,(k+1)} \), see [11]. It is then possible to compute the remaining dual variables’ directions \( \Delta z^{i,(k+1)} \) and \( \Delta \lambda_i^{(k+1)} \); the explicit expressions are provided in Appendix A together with expressions for the data matrices appearing in (5). For more details on how these matrices are formed, see e.g., [22] or [11] Sec. 5 and 6.

At this point, we can compute an appropriate step size, e.g., using back-tracking line search that assures feasibility of the iterates and persistent decrease of the norm of primal and dual residuals, [22], [18], [4]. We can then update the iterates and the procedure is continued until certain stopping criteria are satisfied. These are commonly based on primal and dual residuals norms and the so-called surrogate duality gap, see [22], [11] for more details.

During the run of a PDIPM, the main computational burden arises from the computation of the search directions, which requires solving (5). Indeed, the cost of this can be prohibitive in many cases. Also, for problems that come with privacy requirements, the computations cannot be done in a centralized manner. However, due to the coupling structure of the problem in (4), which is also inherited by (5), it is possible to distribute the computations at each iteration of the PDIPM using message-passing (or dynamic programming) over trees as discussed in [11]. Next, we show how this can be done for the problem under study.

IV. DISTRIBUTED COMPUTATIONS

Let us reconsider the problem in (4). This problem is made up of \( N \) subproblems, each of which is defined by a term in the
cost function and its corresponding constraint set described by each term in (4b)–(4d). The coupling structure of this problem can be represented using the tree illustrated in Figure 1.

Recall that the main computational burden of a PDIPM applied to (4) corresponds to the computations of the search directions, i.e., solving the QP in (5). Note that this QP inherits the same tree representation and, hence, we can solve the problem in (3) by conducting message-passing upward and downward through the tree, see [11], [12]. For this purpose, we first assign each subproblem in (5) to each leaf i of the tree. Then, considering the star-shape structure of the tree, at each iteration k, agents at the leaves of the tree compute their messages to the root of the tree simultaneously. Specifically each agent first computes the search directions for the local variables $\Delta x^i$ and $\Delta v^i$ as a function of $\Delta x$ by solving

$$\begin{bmatrix} H_{ii}^{(k)} & (A^i)^T \\ A^i & 0 \end{bmatrix} \begin{bmatrix} \Delta x^i \\ \Delta v^i \end{bmatrix} = \begin{bmatrix} -r_{primal}^{(k)} \\ 0 \end{bmatrix}.$$

The result of which can be written compactly as

$$\begin{bmatrix} \Delta x^i \\ \Delta v^i \\ \Delta u^i \end{bmatrix} = \begin{bmatrix} u_1^{i,(k)} \\ u_2^{i,(k)} \\ \hat{u}_2^{i,(k)} \end{bmatrix} \Delta x.$$

By inserting the solutions from (6) into the cost function of $\Delta x$, we obtain quadratic functions in $\Delta x$, with Hessian and linear terms given by

$$Q^{i,(k)} = H_{ii}^{i,(k)} + (r_{primal}^{i,(k)})^T H_{ii}^{i,(k)} U_1^{i,(k)} + (U_1^{i,(k)})^T H_{ii}^{i,(k)} U_1^{i,(k)},$$

$$q^{i,(k)} = r_{primal}^{i,(k)} + (U_1^{i,(k)})^T r_{primal}^{i,(k)} + (H_{ii}^{i,(k)})^T u_1^{i,(k)} + (U_1^{i,(k)})^T H_{ii}^{i,(k)} u_1^{i,(k)},$$

respectively. These quadratic functions are then sent to the root. The agent at the root will then form and solve the optimization problem

$$\text{minimize}_{\Delta x} \frac{1}{2} \sum_{i=1}^{N} \Delta x^T Q^{i,(k)} \Delta x + \Delta x^T q^{i,(k)}$$

which gives the search direction $\Delta x$, that it then communicates downwards to its children. Each agent at the leaves of the tree, having received $\Delta x$, can then compute its local variables’ search directions using (6).

Notice that computing stepsizes and residuals, updating the perturbation parameters and checking the termination condition require conducting summing or computing minimum or maximum of local quantities over the tree, and hence, we can use the same computational structure for this purpose, see [11] Sec. 6.3. Consequently, combining message-passing and PDIPMs results in a scalable and distributed algorithm, that purely relies on second-order methods, for solving the problem in (2). A generic summary of the proposed algorithm is given in Algorithm 2. Note that mixing message-passing and PDIPMs does not affect their convergence and the proposed method thus inherits properties such as superlinear convergence and finite termination of PDIPMs. See Appendix C for further discussion.

**Remark 1:** The coupling structure presented in Figure 1 is imposed based on the way the constraints in (2b) are introduced. It is possible to impose other structures, e.g., chain-like or balanced trees, by modifying the way these constraints are introduced. Doing so requires recomputing the bounds calculated in Section II to match this structure.

Note that all agents have access to their local variables updates $x^{i,(k)}$ and that of the global one $x^{(k)}$ as depicted in Figure 1 and all agents consider the solution for $x$ as the computed parameters. This means that we have exact consensus among agents.

### Algorithm 2 Distributed Primal-dual Algorithm (DPDA)

1: Given $k = 0$, $\mu > 1$, $\varepsilon > 0$, $\epsilon > 0$, $\epsilon_{max} > 0$, $x^{(k)}$, $x^{(0)}$, $x^{(0)}$, $v^{(0)}$, and $q^{(0)}$, such that $G'(x^{(0)},v^{(0)}) < 0$, $\|x^{(0)} - x^{(0)}\|^2 < \varepsilon^2$, $z^{(0)}, \lambda_1^{(0)} > 0$ for all $i = 1, \ldots, N$, $\eta^{(0)}$ and $\delta = \mu \mu m/\eta^{(k+1)}$.

2: repeat
3: Perform message-passing upwards and downwards through the tree in Figure 1 to compute the search directions
4: Compute a proper step size, $\alpha^{(k+1)}$, by performing upward-downward passes through the tree, see [11] Sec. 6.3 for details.
5: Update the primal and dual iterates using the computed search directions and step size
6: Perform upward-downward pass through the tree to decide whether to terminate the algorithm and/or to update the perturbation parameter $\delta = \mu \mu m/\eta^{(k+1)}$.
7: $k = k + 1$.
8: until the algorithm is terminated

### V. NUMERICAL EXPERIMENTS

In this section we apply the proposed algorithm DPDA to robust least squares and logistic regression problems, and compare its performance with that of alternating direction method of multipliers (ADMM), [3], and algorithms presented in [20] and [16]. We refer to these algorithms as EXTRA and ESOM, respectively. These algorithms are chosen based on their superior performance in comparison to commonly used algorithms for distributedly solving problems of the form (1). We compare the performance of the algorithms based on their iterations count and computational time. We do not claim that any of the algorithms (including DPDA) has been implemented in their most efficient manner, which can potentially affect the reported computational time, whereas the iteration count to be less susceptible to this. Another reason for considering the iteration count is that it corresponds to the number of communications among agents. This is a good performance measure since for many existing algorithms the communication overhead is the most significant bottleneck which can create significant latency, see e.g., [10], [14].
DPDA we use the equivalent reformulation (see [4])

ESOM cannot be applied to this problem and in order to use

smooth, it is not twice continuously differentiable. This means

\[ \phi \] generated randomly in the interval

parameters are chosen as

\[ [0, 1] \] based on a uniform distribution in the interval

\[ N \] the measurement noise. In this experiment,

\[ A \] ond, in order to study the effect of the scaling of the problem,

matrices

\[ \rho \] for a coefficient matrix

\[ A \] are tuned manually to maximize performance.

\[ \beta \] was 34 and we have

\[ p = 34 \] and we have considered 350 data points, that we assume are divided among

\[ N = 10 \] agents. This means that \( n_i = 35 \). The results

from these experiments are illustrated in Figure 2 (right). The computational times are reported in Table I (the third row).

Similar to the previous experiment we see that DPDA clearly outperforms the other algorithms, and ADMM has the highest per-iteration computational complexity. This is because each agent at each iteration needs to solve a robust least squares problem for updating its local variables. Due to this, ADMM in fact has the worst per-iteration computational complexity among these algorithms. The computational time for this experiment are reported in Table [I](the first two rows). As can be seen from the table, EXTRA and DPDA provide comparable performance, and DPDA clearly outperforms EXTRA for worse conditioned problems. Having said that, DPDA will potentially perform worse on problems with large number of features. However, as was mentioned in the introduction these problems are not the focus of this paper.

B. Logistic Regression Problem

We continue our investigation of the performance of DPDA by conducting an experiment based on a logistic regression problem. A logistic regression problem can be written as

\[ \max_x \sum_{i=1}^{N} \sum_{j=1}^{n_i} \phi_M(A^i_j x - Y^i_j), \quad (7) \]

where \( A^i \in \mathbb{R}^{n_i \times p} \) with \( A^i_j \) denoting the \( j \)th row of \( A^i \), and \( \phi: \mathbb{R} \rightarrow \mathbb{R} \) is the Huber penalty function defined as

\[ \phi_M(u) = \begin{cases} u^2 & |u| \leq M \\ M(|u| - M) & |u| > M \end{cases} \]

We assume that each agent \( i \) has access to its own measurements \( Y^i = A^i x + e^i \) with \( A^i \in \mathbb{R}^{n_i \times p} \) and \( e^i \sim N(0, \sigma^2 I) \) is the measurement noise. In this experiment, \( N = 10 \), \( n_i = 20 \) and \( p = 10 \), the matrices \( A^i \) have been generated randomly based on a uniform distribution in the interval \([0, 1]\) and the parameters \( x \) used for producing the data have been also generated randomly in the interval \([0, 20]\).

Notice that although the cost function for this problem is smooth, it is not twice continuously differentiable. This means ESOM cannot be applied to this problem and in order to use DPDA we use the equivalent reformulation (see [4])

\[ \min_{x, u^i, v^i} \sum_{i=1}^{N} \|u^i\|^2 + M1^T v^i - u^i - v^i \leq A^i x - Y^i \leq u^i + v^i \]

subject to

\[ 0 \leq v^i \leq M1 \]

\[ v^i \geq 0 \]

Note that ESOM cannot be used for solving this formulation either, as it includes constraints. We set \( M = 1 \) and the relaxation level for DPDA to \( \varepsilon = 10^{-3} \). The algorithm parameters are chosen as \( \mu = 10 \), \( \beta = 0.4 \) and \( \alpha = 0.1 \). The matrices \( W \) and \( \tilde{W} \) used in EXTRA are chosen in the same way as in [20] Sec. 4 for the star-shaped graph in Figure 1. The other parameters in this algorithm, and those in ADMM, are tuned manually to maximize performance.

The results are shown in Figure 2 (left). The figure reports results for two experiments. First for a coefficient matrix \( A \) generated as described above (condition number 6.56). Second, in order to study the effect of the scaling of the problem, for a coefficient matrix \( A \) obtained by manipulating its singular values to increase the condition number by almost a factor ten (to 56.92). The optimal solution for \( x^* \), has been computed using CVX, [8]. From these figures, we observe that DPDA requires far fewer iterations than ADMM and EXTRA, and hence communications among agents, to converge to a solution. Furthermore, it is much less sensitive to the scaling of the problem. Despite this, one should note that since EXTRA is a gradient-based method, the computational complexity of each of its iterations is linear in the number of variables. This is in contrast to DPDA where the computational cost of each of iterations is cubic in the number of variables. Moreover, ADMM has a higher computational complexity EXTRA and DPDA. This is because at each iteration of the ADMM, each agent needs to solve a robust least squares problem for updating its local variables. Due to this, ADMM in fact has the worst per-iteration computational complexity among these algorithms. The computational time for this experiment are reported in Table I (the first two rows). As can be seen from the table, EXTRA and DPDA provide comparable performance, and DPDA clearly outperforms EXTRA for worse conditioned problems. Having said that, DPDA will potentially perform worse on problems with large number of features. However, as was mentioned in the introduction these problems are not the focus of this paper.

VI. Conclusions

In this paper we proposed a distributed PDPM for computing approximate solutions for convex consensus problems.
This was done by first proposing a relaxed formulation of the consensus problem. Solving this problem results in an approximate solution of the consensus problem, where we showed how the accuracy of the computed solution can be controlled by the relaxation level. The imposed coupling structure in the relaxed problem enabled us to distribute the computations of each iteration of a PDIPM using message-passing. We showed the performance of the proposed algorithm using numerical experiments based on robust least squares and logistic regression problems, using both synthetic and real data. In this paper we did not discuss a standard approach for choosing \( \varepsilon \). We plan to address this as a future line of research, possibly through introduction of efficient methodologies for global or local scaling of the problem data.

**References**

[1] M. Annergren, S. Khoshfetrat Pakazad, A. Hansson, and B. Wahlberg. A distributed primal-dual interior-point method for loosely coupled problems using ADMM. *ArXiv e-prints*, February 2015.

[2] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

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

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] V. Cevher, S. Becker, and M. Schmidt. Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics. *IEEE Signal Processing Magazine*, 31(5):32–43, Sept 2014.

[6] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, volume 49 of Springer Optimization and Its Applications, pages 185–212. Springer New York, 2011.

[7] J. Eckstein. Splitting methods for monotone operators with application to parallel optimization. PhD dissertation, Massachusetts Institute of Technology, 1989.

[8] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming.

[9] T. Hastie, J. Friedman, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2003.

[10] M. Jaggi, V. Smith, M. Takáč, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 3068–3076, 2014.

[11] S. Khoshfetrat Pakazad, A. Hansson, M. S. Andersen, and I. Nielsen. Distributed primal-dual interior-point methods for solving tree-structured coupled convex problems using message-passing. *Optimization Methods and Software*, 32(3):401–435, 2017.

[12] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[13] A. Koppel, B. M. Sadler, and A. Ribeiro. Proximity without consensus in online multi-agent optimization. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3726–3730, 2016.

[14] M. Li, D. G. Andersen, A. J. Smola, and K. Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pages 19–27, 2014.

[15] M. Lichman. UCI machine learning repository, 2013.

[16] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro. A decentralized second-order method with exact linear convergence rate for consensus optimization. *ArXiv e-prints*, February 2016.

[17] A. Nedic, A. Ozdaglar, and P.A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, April 2010.

[18] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2006.

[19] H. Ohlsson, T. Chen, S. Khoshfetrat Pakazad, L. Ljung, and S. Shankar Sastry. Scalable Anomaly Detection in Large Homogenous Populations. *Automatica*, 50(5):1459 – 1465, 2014.

[20] W. Shi, Q. Ling, G. Wu, and W. Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[21] E. Wei, A. Ozdaglar, and A. Jadbabaie. A distributed Newton method for network utility maximization–I: Algorithm. *IEEE Transactions on Automatic Control*, 58(9):2162–2175, 2013.

[22] S. J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, 1997.

**Appendix A**

**Description of the data matrices in (5)**

Here we present a detailed description of the data matrices defining (5). Let us start with the Hessian matrix of the cost function that is given as

\[ H_{\text{pd}}^{i,(k)} = \begin{bmatrix} \nabla^2 f_i^{(k)} & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & \nabla^2 G_j^{i,(k)} & 0 \\ 0 & \ldots & 0 & \nabla^2 G_j^{i,(k)} \end{bmatrix} - \begin{bmatrix} \nabla^2 f_i^{(k)} & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & \nabla^2 G_j^{i,(k)} & 0 \\ 0 & \ldots & 0 & \nabla^2 G_j^{i,(k)} \end{bmatrix} + \sum_{j=1}^{m_i} \varepsilon_j^{i,(k)} \begin{bmatrix} \nabla^2 G_j^{i,(k)} \end{bmatrix}^{T} + \sum_{j=1}^{m_i} \varepsilon_j^{i,(k)} \begin{bmatrix} \nabla^2 G_j^{i,(k)} \end{bmatrix} \]

The elements of statistical methodology for global or local scaling of the problem data.

where \( 2\lambda_i^{(k)} \) is defined similarly. The coefficient vector defining the linear term in the cost function can be extracted as below

\[ r_{\text{dual}}^{i,(k)} = \left[ \nabla x_i h_i^{(k)} \right] + \left[ D_x G_0^{i,(k)} \right] T x_i - \lambda_i^{(k)} \]

and

\[ 2\lambda_i^{(k)} \left[ \begin{array}{c} \nabla x_i h_i^{(k)} T \\ \nabla x_i h_i^{(k)} \end{array} \right] - \left[ \begin{array}{c} \nabla x_i h_i^{(k)} T \\ \nabla x_i h_i^{(k)} \end{array} \right] T x_i^{i,(k)} - x_i^{i,(k)} = \lambda_i^{(k)} \]

\[ \varepsilon_i^{i,(k)} = -\text{diag}(z_i^{(k)}) G_i^{i,(k)} - \frac{1}{\delta} \]

\[ \varepsilon_i^{i,(k)} = -\frac{1}{\delta} \left[ \begin{array}{c} \lambda_i^{(k)} \\ \lambda_i^{(k)} \end{array} \right] \]

\[ \varepsilon_i^{i,(k)} = \frac{1}{\delta} \frac{1}{\delta} \left[ \begin{array}{c} \lambda_i^{(k)} \\ \lambda_i^{(k)} \end{array} \right] \]

\[ \varepsilon_i^{i,(k)} = \frac{1}{\delta} \frac{1}{\delta} \left[ \begin{array}{c} \lambda_i^{(k)} \\ \lambda_i^{(k)} \end{array} \right] \]
where for the sake of notational ease we have denoted a function \( G(x) \) evaluated at \( x^{(k)} \) with \( G^{(k)} \). Here \( \delta \) is referred to as the perturbation parameter. Having computed the primal variables’ directions \( \Delta x^{i,(k+1)} \), \( \Delta t^{i,(k+1)} \) and \( \Delta x^{i,(k+1)} \) together with the dual variables’ directions \( \Delta \delta^{i,(k+1)} \) by solving (5), we can compute the remaining dual variables’ directions as

\[
\begin{align*}
\Delta z^{i,(k+1)} &= - \text{diag}(G^{i,(k)})^{-1} \left( \text{diag}(z^{i,(k)}) \times 
\begin{bmatrix}
D_x^{i,(k)} & D_t^{i,(k)}
\end{bmatrix}
\begin{bmatrix}
\Delta x^{i,(k+1)} \\
\Delta t^{i,(k+1)}
\end{bmatrix} - r^{i,(k)} \right) \\
\Delta \lambda^{i,(k+1)} &= \frac{1}{\|x^{(k)} - x^{i,(k)}\|^2 - \varepsilon^2} \times 
\begin{bmatrix}
\lambda^{i,(k)} \\
\lambda^{i,(k)}
\end{bmatrix}
\begin{bmatrix}
x^{i,(k)} - x^{(k)} \\
x^{i,(k)} - x^{i,(k)}
\end{bmatrix}^T 
\begin{bmatrix}
\Delta x^{i,(k+1)} \\
\Delta t^{i,(k+1)}
\end{bmatrix} - r^{i,(k)}
\end{align*}
\] (9a)

\[
\hat{\eta}^{(k+1)} = \sum_{i=1}^N - \lambda^{i,(k+1)} \left( \|x^{i,(k+1)} - x^{(k+1)}\|^2 - \varepsilon^2 \right) - (z^{i,(k+1)})^T G^{i,(k+1)}
\]

denoting the surrogate duality gap.

**APPENDIX B**

**PROOF OF THEOREM**

Firstly, notice that we have

\[
\frac{1}{N} \sum_{i=1}^N F_i(x_{\text{rel}}^*) \leq \frac{1}{N} \sum_{i=1}^N F_i(x^*) \leq \frac{1}{N} \sum_{i=1}^N F_i(x_{\text{rel}}),
\]

where the first inequality follows from the fact that (2) is a relaxation of (1) and the second inequality follows from the fact that \( x^* \) is optimal for (1) but \( x_{\text{rel}}^* \) is not. Then we have

\[
\sum_{i=1}^N F_i(x_{\text{rel}}^*) - F_i(x^*) \leq N(U_b - L_b) \leq \sum_{i=1}^N \|F_i(x_{\text{rel}}^*) - F_i(x_{\text{rel}})\| \leq \sum_{i=1}^N L_i \|x_{\text{rel}}^* - x_{\text{rel}}\| \leq \varepsilon L.
\]

which proves (5). Under the assumption that the cost function is strongly convex, we have

\[
\|x_{\text{rel}}^* - x^*\|^2 \leq \frac{2}{Nm} \sum_{i=1}^N F_i(x_{\text{rel}}) - F_i(x^*) \leq \frac{2\varepsilon L}{Nm}
\]

which completes the proof.

**APPENDIX C**

**CONVERGENCE PROPERTIES OF THE PROPOSED METHOD**

PDIPMs have been shown to converge and that they enjoy favorable convergence properties such as superlinear convergence and finite termination, see [22] Ch. 6 and 7] for a full discussion and technical presentation. Algorithm [2] is obtained by distributing the computations of each iteration of the PDIPM. As was shown in [11] Sec. 6.1 and Thm 6.4, the computed search directions using message-passing are exact. Furthermore the exact computation of the parameters can be trivially distributed within a message-passing framework, see [11] Sec. 6.3. This means that the distributed computations do not jeopardize the convergence of the PDIPM and does not affect its convergence properties.