Distributed Algorithm via Continuously Differentiable Exact Penalty Method for Network Optimization

Priyank Srivastava Jorge Cortés

Abstract— **This paper proposes a distributed optimization framework for solving nonlinear programming problems with separable objective function and local constraints. Our novel approach is based on first reformulating the original problem as an unconstrained optimization problem using continuously differentiable exact penalty function methods and then using gradient based optimization algorithms. The reformulation is based on replacing the Lagrange multipliers in the augmented Lagrangian of the original problem with Lagrange multiplier functions. The problem of calculating the gradient of the penalty function is challenging as it is non-distributed in general even if the original problem is distributed. We show that we can reformulate this problem as a distributed, unconstrained convex optimization problem. The proposed framework opens new opportunities for the application of various distributed algorithms designed for unconstrained optimization.**

I. Introduction

Recently, there has been a considerable interest in the analysis and control of large scale networked systems, like the control of multi-agent systems and power systems. In many cases, these applications could be modeled as optimization problems, and even as algebraic equations in some cases. Due to the communication constraints and the privacy of the agents, it is desirable to solve these problems in a distributed way. Distributed algorithms could also outperform the centralized ones with the parallel processing power available nowadays. However, most of the distributed algorithms available for solving constrained optimization problems either convert the problem into unconstrained optimization of a non-differentiable function or first construct the Lagrangian and then perform the optimization in both the primal and dual variables. In the former case, the non-differentiability of the objective functions prohibits the use of easily implementable gradient based methods and instead, there is a need for more complicated subgradient based methods. On the other hand, Lagrangian based methods often lack good convergence speed and need a good starting point. These considerations provide the motivation for this paper.

Literature Review: Given the anticipated growth in number of large scale network systems, there have been a lot of efforts to develop distributed algorithms to solve optimization problems. For distributed optimization, approaches in the literature could be divided into two main categories: consensus-based where all the agents agree on a common solution [1], [2], [3] and the ones where each agent computes only its component [4], [5]. Our algorithm is based on the latter approach. The existing methods under this kind of approach consist of Lagrangian based algorithms, e.g. the primal-dual algorithm [6], and exact reformulation using non-differentiable penalty functions [7]. There have been a lot of works on solving a constrained optimization problem by means of converting it into an unconstrained minimization problem of a single function [8], [9], [10], [11]. In this paper, we focus on continuously differentiable penalty functions which were introduced in [12]. The work [10] generalizes these penalty functions and proves that complete equivalence between the solutions of constrained and unconstrained problems could be established under some regularity assumptions on the constraint set with reference to some arbitrary compact set. The work [13] proposes a continuously differentiable exact penalty function that relaxes some of the assumptions in [10]. However, most works in the context of continuously differentiable exact penalty functions use the centralized optimization algorithms for the minimization of the unconstrained penalty function. Here, we focus on calculating the gradient of the penalty function in a distributed way which would enable us to use any gradient based algorithm, like gradient descent in a distributed way.

Statement of Contributions: We consider nonlinear programming problems with separable objective function and local constraints. Our starting point is the exact reformulation of this problem as an unconstrained optimization of a continuously differentiable exact penalty function. We propose a framework for the distributed optimization of the unconstrained penalty function. Our first contribution is to provide a distributed algorithm to solve a linear equation. This distributed algorithm is based on the reformulation of the problem of finding the solutions of the linear equation as an unconstrained convex optimization problem. Our second contribution is the calculation of the gradient of the penalty function in a distributed way by establishing the fact that the calculation of certain non-distributed terms in the gradient could be formulated as solving a linear equation. For reasons of space, all proofs are omitted and would appear elsewhere.

Organization: Section II introduces notation and basic concepts from graph theory and constrained optimization, providing background on continuously differentiable exact penalty methods. Section III introduces the constrained distributed optimization problem of interest. Section IV formulates the problem of calculating the multiplier functions and their gradients as a linear equation. Section V proposes an algorithm for solving linear equations in a distributed way. Section VI describes the proposed distributed algorithm to

The authors are with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, {psrivast,cortes}@ucsd.edu

solve the optimization problem. Section VII illustrates the effectiveness of the proposed dynamics through simulations and finally, Section VIII gathers our conclusions and ideas for future work.

II. Preliminaries

In this section, we present our notational conventions and review some basic concepts.

A. Notation

Let $\mathbb{R}$ and $\mathbb{Z}$ be the set of real numbers and integers, respectively. We let $|X|$ denote the cardinality of a set X. For a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$, we denote by $\nabla_x f$ the partial derivative of f with respect to its argument x. $A \otimes B$ denotes the Kronecker product of two matrices A and B. We use $\mathbf{0}$ and $\mathbf{1}$ to denote the vector or matrix of zeros and ones of appropriate dimension, respectively. $\text{diag}(v)$ denotes the diagonal matrix formed by the elements of vector v on the diagonal. We use $\mathcal{X}^o$ to denote the interior of a set $\mathcal{X}$.

B. Graph theory

Following [14], we present the basic concepts from graph theory. We denote an undirected graph by $G = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ as the set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ as the set of edges. $(i, j) \in \mathcal{E}$ if and only if $(j, i) \in \mathcal{E}$. A vertex $j \in \mathcal{V}$ is said to be a neighbor of i if and only if $(i, j) \in \mathcal{E}$. The set $\mathcal{N}_i$ denotes the set of all the neighbors of vertex i. The degree of a node is the number of edges connected to it. With $|\mathcal{V}| = n$, the degree matrix $D \in \mathbb{R}^{n\times n}$ of a graph is the diagonal matrix with $D_{ii} = \deg(v_i)$. The adjacency matrix $A \in \mathbb{R}^{n\times n}$ is a matrix with all the diagonal elements as zero and is defined such that $A_{ij=1}$ if the edge $(i, j) \subseteq \mathcal{E}$ and 0, otherwise. The Laplacian matrix of a graph $G \in \mathbb{R}^{n\times n}$ is defined as $L := D - A$. Note that, due to the structure of Laplacian matrix, $\mathbf{1}^T L = 0$.

C. Constrained optimization

Here, we introduce the basic concepts of constrained optimization following [15], [11]. Consider the following nonlinear optimization problem

$$\begin{aligned} \min_{x\in\mathcal{D}} \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0,\ h(x) = 0 \end{aligned} \tag{1}$$

where, $\mathcal{D}$ is a compact set assumed to be regular, that is $D = \overline{\mathcal{D}^o}$ and $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^p$ are assumed to be twice continuously differentiable with $p \leq n$. The feasible set of (1) is defined as $\mathcal{F} := \{x \mid x \in \mathcal{D}, g(x) \leq 0, h(x) = 0\}$. Let us define following index sets for the inequality constraints

$$\begin{aligned} I_0(x) &:= \{j \mid g_j(x) = 0\} \\ I_+(x) &:= \{j \mid g_j(x) \geq 0\} \end{aligned}$$

We state some regularity conditions for the constraints below

(a) The *linear independence constraint qualification* (LICQ) holds at a point $x \in \mathbb{R}^n$ if the gradients $\nabla g_j(x)$, $j \in I_0(x)$, ∇h_k, $\forall k = 1, \ldots, p$ are linearly independent.

(b) The *Mangasarian-Fromovitz constraint qualification* (MFCQ) holds at a point $x \in \mathbb{R}^n$ if the gradients ∇h_k, $\forall k$ are linearly independent and there exists a vector $z \in \mathbb{R}^n$ such that

$$\nabla g_j(x)' z < 0, \quad j \in I_0(x) \tag{2a}$$
$$\nabla h_k(x)' z = 0, \quad \forall k \tag{2b}$$

(c) The *extended Mangasarian-Fromovitz constraint qualification* (EMFCQ) is the extension of MFCQ with the set $I_0(x)$ in (2a) replaced by $I_+(x)$.

The Lagrangian function associated with (1) is the function $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ given by

$$L(x, \lambda, \mu) = f(x) + \lambda' g(x) + \mu' h(x)$$

where, $\lambda \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^p$ are the Lagrange multipliers or the dual variables associated with the inequality and equality constraints, respectively. A Karush-Kuhn-Tucker (KKT) point for problem (1) is a triplet $(\bar{x}, \bar{\lambda}, \bar{\mu})$ such that

$$\begin{aligned} \nabla_x L(\bar{x}, \bar{\lambda}, \bar{\mu}) &= 0 \\ \bar{\lambda} \geq 0 \quad \bar{\lambda}' g(\bar{x}) &= 0 \\ g(\bar{x}) \leq 0 \quad h(\bar{x}) &= 0 \end{aligned}$$

KKT conditions are the necessary conditions for a point to be local optimal if certain regularity condition (like the ones mentioned above) is satisfied at that point.

D. Continuously differentiable exact penalty functions

Here, we introduce the concept of continuously differentiable exact penalty functions following [12] and [10]. The key idea in the construction of continuously differentiable exact penalty functions is that of replacing the multiplier vectors (λ, μ) in the augmented Lagrangian of [16] by the multiplier functions $(\lambda(x), \mu(x))$. Following [12], we make the assumption

Assumption 1: LICQ is satisfied at any $x \in \mathbb{R}^n$.

Let $N : \mathbb{R} \to \mathbb{R}^{(m+p)\times(m+p)}$ be defined by

$$N(x) = \begin{bmatrix} \nabla g(x)' \nabla g(x) + \gamma^2 G^2(x) & \nabla g(x)' \nabla h(x) \\ \nabla h(x)' \nabla g(x) & \nabla h(x)' \nabla h(x) \end{bmatrix} \tag{3}$$

where, $\gamma \neq 0$ and $G(x) := \text{diag}(g(x))$. In the following result from [10], we provide the expressions for $(\lambda(x), \mu(x))$ and some of their properties.

Proposition 2.1: (Multiplier functions and their derivatives): For any $x \in \mathbb{R}^n$

(a) $N(x)$ is a positive definite matrix;

(b) $(\lambda(x), \mu(x))$ are given by

$$\begin{bmatrix} \lambda(x) \\ \mu(x) \end{bmatrix} = -N^{-1}(x) \begin{bmatrix} \nabla g(x)' \\ \nabla h(x)' \end{bmatrix} \nabla f(x); \tag{4}$$

(c) if $(\bar{x}, \bar{\lambda}, \bar{\mu})$ is a KKT triple for problem (1), then $\lambda(\bar{x}) = \bar{\lambda}$ and $\mu(\bar{x}) = \bar{\mu}$;

(d) the functions $\lambda(x)$ and $\mu(x)$ are continuously differentiable and the Jacobian matrices are given by

$$\begin{bmatrix}\nabla\lambda(x)'\\ \nabla\mu(x)'\end{bmatrix} = -N^{-1}(x)\begin{bmatrix}R(x)\\ S(x)\end{bmatrix} \tag{5}$$

where,

$$\begin{aligned} R(x) &:= \nabla g(x)'\nabla_x^2 L(x,\lambda(x),\mu(x)) \\ &\quad + \sum_{j=1}^{m} e_j^m \nabla_x L(x,\lambda(x),\mu(x))'\nabla^2 g_j(x) \qquad (6a)\\ &\quad + 2\gamma^2\Lambda(x)G(x)\nabla g(x)' \\ S(x) &:= \nabla h(x)'\nabla_x^2 L(x,\lambda(x),\mu(x)) \\ &\quad + \sum_{k=1}^{p} e_k^p \nabla_x L(x,\lambda(x),\mu(x))'\nabla^2 h_k(x) \qquad (6b)\end{aligned}$$

$$\begin{aligned}\nabla_x L(x,\lambda(x),\mu(x)) &:= [\nabla_x L(x,\lambda,\mu)]_{\substack{\lambda=\lambda(x)\\ \mu=\mu(x)}}\\ \nabla_x^2 L(x,\lambda(x),\mu(x)) &:= [\nabla_x^2 L(x,\lambda,\mu)]_{\substack{\lambda=\lambda(x)\\ \mu=\mu(x)}}\\ \Lambda(x) &:= \text{diag}(\lambda(x))\end{aligned}$$

and $e_j^m(e_k^p)$ denote the j-th(k-)th column of the $m\times m$ ($p\times p$) identity matrix.

With the expressions of $(\lambda(x),\mu(x))$ from the Proposition 2.1, we can define the continuously differentiable penalty function as

$$\begin{aligned} f^\epsilon(x) &:= f(x) + \lambda(x)'(g(x) + Y^\epsilon(x)y^\epsilon(x)) + \mu(x)'h(x)\\ &\quad + \frac{1}{\epsilon}\|g(x) + Y^\epsilon(x)y^\epsilon(x)\|^2 + \frac{1}{\epsilon}\|h(x)\|^2 \end{aligned} \tag{7}$$

where, $\epsilon > 0$

$$\begin{aligned} y_j^\epsilon(x) &:= \Big\{ -\min\Big[0, g_j(x) + \frac{\epsilon}{2}\lambda_j(x)\Big]\Big\}^{1/2} \quad \forall j\\ Y^\epsilon(x) &:= \text{diag}(y^\epsilon(x))\end{aligned}$$

Now, consider the unconstrained problem

$$\min_{x\in\mathcal{D}^o} f^\epsilon(x) \tag{8}$$

Before we provide the equivalence between (1) and (8), let us define the various notions of exactness following [10].

Definition 2.2 (Exactness of penalty function): (a) The function $f^\epsilon(x)$ is a weakly exact penalty function for problem (1) if $\exists\,\bar\epsilon$ s.t. $\forall\epsilon\in(0,\bar\epsilon]$, the set of global minimizers of (1) and (8) are equal.

(b) The function $f^\epsilon(x)$ is an exact penalty function for problem (1) if $\exists\,\bar\epsilon$ s.t. $\forall\epsilon\in(0,\bar\epsilon]$, $f^\epsilon(x)$ is a weakly exact penalty function for problem (1) and any local minimizer of (8) is a local minimizer of (1).

(c) The function $f^\epsilon(x)$ is a strongly exact penalty function for problem (1) if $\exists\,\bar\epsilon$ s.t. $\forall\epsilon\in(0,\bar\epsilon]$, the set of global as well as the local minimizers of (1) and (8) are equal.

In the following result, we summarize some of the known results from [10].

Proposition 2.3: (Equivalence between the optimizers of (1) *and* (8)*):* If LICQ is satisfied $\forall x\in\mathbb{R}^n$,

(a) $f^\epsilon(x)$ is a weakly exact penalty function for (1).

(b) $(\bar x,\bar\lambda,\bar\mu)$ is a KKT point for (1) $\Rightarrow \forall\epsilon>0, \nabla f^\epsilon(\bar x) = 0$.

(c) Under the assumption that EMFCQ holds, the function $f^\epsilon(x)$ is an exact penalty function for problem (1). Also, $\exists\,\bar\epsilon$ s.t. $\forall\epsilon\in(0,\bar\epsilon], \nabla f^\epsilon(\bar x) = 0 \Rightarrow (\bar x,\lambda(\bar x),\mu(\bar x))$ is a KKT point for problem (1). Moreover, if the set of local optimal values of (1) is finite, then the function $f^\epsilon(x)$ is a strongly exact penalty function for problem (1).

III. Problem Statement

Consider a network with $n\in\mathbb{Z}_{\geq 1}$ agents connected by a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$. Each agent i has a decision variable $x_i\in\mathbb{R}$ and a twice continuously differentiable cost function $f_i:\mathbb{R}\to\mathbb{R}$. Consider the following optimization problem

$$\begin{aligned}\min_{x\in\mathcal{D}} \quad & f(x) = \sum_{i=1}^{n} f_i(x_i)\\ \text{s.t.} \quad & g(x)\leq 0,\ h(x) = 0\end{aligned} \tag{9}$$

with twice continuously differentiable vector valued functions $g:\mathbb{R}^n\to\mathbb{R}^m$ and $h:\mathbb{R}^n\to\mathbb{R}^p$ with $p\leq n$. Each of the functions g_j and h_k is local, i.e. they depend on an agent x_i and its neighbors $\{x_j\}_{j\in\mathcal{N}_i}$ only. Furthermore, all agents involved in a constraint can calculate the constraint and its gradients. Our aim is to develop a smooth distributed algorithm to find an optimizer to the constrained problem (9).

To solve (9), algorithms such as primal-dual dynamics, which optimize the Lagrangian in both the primal and dual variables, could be used. These algorithms are inherently distributed in nature if the objective function is separable and constraints are local but these suffer from slow convergence speeds and often need a good starting point. Also, in general, there are no robustness guarantees with this approach and the convergence to the optimizers is not monotonic. To improve convergence speed, instead of optimizing the Lagrangian, we could optimize the augmented Lagrangian but this might destroy the distributed nature of the dynamics.

Another approach to solve (9) in a distributed way consists of first reformulating the problem as an unconstrained optimization problem by adding non-differentiable penalty terms replacing the constraints to the objective function. After that, the problem could be solved by using subgradient based methods. These subgradient based methods are difficult to implement and analyzing their convergence to the optimizers requires tools from nonsmooth analysis. Also, they prohibit the use of accelerated second-order methods altogether.

Our approach consists of reformulating the problem as an unconstrained optimization problem based on the continuously differentiable penalty function methods described in Section II. We then show how the gradient of the penalty function could be computed in a distributed way. This enables us to use smooth gradient descent which has exponential rate of convergence and hence, overcome the drawbacks of the current approaches mentioned earlier. In Section IV, we show that the problem of distributed calculation of Lagrange multiplier functions and their gradients, which are involved in the calculation of the gradient of the

penalty function, could be reformulated as a linear algebraic equation with decomposable terms. We show how to solve this equation in a distributed manner in Section V.

IV. Distributed Formulation for Computing the Multiplier Functions and their Gradients

Many of the unconstrained minimization algorithms are gradient based. Hence, to find the optimizers of (8), it is desirable to calculate the gradient of the penalty function $f^\epsilon(x)$ in a distributed way, i.e. each agent should be able to calculate the partial derivative $\nabla_{x_i} f^\epsilon(x)$ locally. In this section, we identify the challenges associated with the distributed calculation of the gradient of the penalty function and show that they are equivalent to solving a linear algebraic equation. $\nabla f^\epsilon(x)$ is given by

$$\begin{aligned}\nabla f^\epsilon(x) &= \nabla f(x) + \nabla g(x)\lambda(x) + \nabla h(x)\mu(x) \\ &+ \nabla\lambda(x)(g(x) + Y^\epsilon(x)y^\epsilon(x)) + \nabla\mu(x)h(x) \\ &+ \frac{2}{\epsilon}\nabla g(x)(g(x) + Y^\epsilon(x)y^\epsilon(x)) + \frac{2}{\epsilon}\nabla h(x)h(x)\end{aligned} \tag{10}$$

The gradient of $f^\epsilon(x)$ w.r.t. x_i is given by

$$\begin{aligned}\nabla_{x_i} f^\epsilon(x) = \nabla_{x_i} f_i(x_i) &+ \sum_{j=1}^{m} \lambda_j(x)\nabla_{x_i} g_j(x) + \\ &\sum_{k=1}^{p} \mu_k(x)\nabla_{x_i} h_k(x) + \sum_{k=1}^{p} h_k(x)\nabla_{x_i}\mu_k(x) + \\ &\sum_{j=1}^{m} \big(g_j(x) + y_j^{\epsilon 2}(x)\big)\nabla_{x_i}\lambda_j(x) + \\ &\frac{2}{\epsilon}\sum_{j=1}^{m} \big(g_j(x) + y_j^{\epsilon 2}(x)\big)\nabla_{x_i} g_j(x) + \\ &\frac{2}{\epsilon}\sum_{k=1}^{p} h_k(x)\nabla_{x_i} h_k(x)\end{aligned} \tag{11}$$

From the above expression of $\nabla_{x_i} f^\epsilon(x)$, using the fact that each agent knows the value of all the constraints in which it is involved and their derivatives, it is clear that if every agent has information about $(\lambda(x), \mu(x))$ and $(\nabla\lambda(x), \nabla\mu(x))$, then each agent can compute all the terms locally except for $\sum_{j=1}^{m} \big(g_j(x) + y_j^{\epsilon 2}(x)\big)\nabla_{x_i}\lambda_j(x) + \sum_{k=1}^{p} h_k(x)\nabla_{x_i}\mu_k(x)$. Let us denote it by $\rho(x)$. Let us first see how we can formulate and solve the problem of calculating $(\lambda(x), \mu(x))$ and $(\nabla\lambda(x), \nabla\mu(x))$ in a distributed way. We then describe how to calculate $\rho(x)$ in a distributed way in Section VI.

For a given value of x, calculating $(\lambda(x), \mu(x))$ and $(\nabla\lambda(x), \nabla\mu(x))$ is equivalent to solving two linear algebraic equations (4) and (5). We could rewrite (4) and (5) as

$$N(x)\begin{bmatrix}\lambda(x)\\ \mu(x)\end{bmatrix} = -\begin{bmatrix}\nabla g(x)'\\ \nabla h(x)'\end{bmatrix}\nabla f(x) \tag{12a}$$

$$N(x)\begin{bmatrix}\nabla\lambda(x)'\\ \nabla\mu(x)'\end{bmatrix} = -\begin{bmatrix}R(x)\\ S(x)\end{bmatrix} \tag{12b}$$

The following result proves that we can actually decompose the matrix $N(x)$ and the right hand side of (12) as the summation of locally computable matrices.

Proposition 4.1: (Distribution of the known matrices in (12)*):* For any $x \in \mathbb{R}^n$, calculating $(\lambda(x), \mu(x))$ and $(\nabla\lambda(x), \nabla\mu(x))$ is equivalent to solving linear equation of the form

$$\sum_{i=1}^{n} \mathfrak{N}_i\mathfrak{v} = \sum_{i=1}^{n} \mathfrak{b}_i \tag{13}$$

where, $\mathfrak{N}_i \in \mathbb{R}^{q\times q}\ \forall i$, $\mathfrak{v} \in \mathbb{R}^q$ and $\mathfrak{b}_i \in \mathbb{R}^q\ \forall i$.

Remark 4.2: Solving linear algebraic equations distributively is an interesting problem of its own, cf. [17], [18]. Specifically, equations with same structure as (13) appear frequently [19] and have numerous applications, like distributed sensor fusion [20], or to find the maximum-likelihood estimate of an unknown parameter [21]. [20] exploits the positive definiteness of the matrices and [21] uses element wise average consensus to find the solutions of (13). [19] also exploits the positive definite property of the individual matrices and requires the agents to know the state as well as the matrices of the neighbors. In Section V, we propose a distributed algorithm that does not need the individual matrices to be positive definite and has exponential rate of convergence. •

V. Distributed Algorithm for Solving Linear Equations

In this section, we propose an exponentially fast distributed algorithm to find the solutions of the linear equation (13) developed in Section IV. Consider each agent having its own version of $\mathfrak{v}$ denoted by $\mathfrak{v}_i$. Then we can rewrite (13) as

$$\sum_{i=1}^{n} \mathfrak{N}_i\mathfrak{v}_i = \sum_{i=1}^{n} \mathfrak{b}_i \tag{14a}$$

$$(L \otimes I_q)\mathbf{V} = \mathbf{0} \tag{14b}$$

where, $\mathbf{V} := [\mathfrak{v}_1; \ldots; \mathfrak{v}_n]$. Equation (14b) ensures that all the $\mathfrak{v}_i$s are same. Although (14b) is distributed (meaning that agent i needs to only know the value of its $\mathfrak{v}_i$ and $\{\mathfrak{v}_j\}_{j\in\mathcal{N}_i}$ to check if the equation is satisfied), (14a) is not. To make it distributed, we introduce a new variable $y_i \in \mathbb{R}^q$. Let $Y = [y_1; \ldots; y_n]$ and introduce the following set of equations.

$$\begin{pmatrix}\eta & -L\otimes I_q\\ L\otimes I_q & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{V}\\ Y\end{pmatrix} = \begin{pmatrix}\mathfrak{B}\\ \mathbf{0}\end{pmatrix} \tag{15}$$

where, $\eta := \begin{bmatrix}\mathfrak{N}_1 & & \\ & \ddots & \\ & & \mathfrak{N}_n\end{bmatrix}$ and $\mathfrak{B} = \begin{bmatrix}\mathfrak{b}_1\\ \vdots\\ \mathfrak{b}_n\end{bmatrix}$. Note that the set of equations (15) is distributed. The following result characterizes the equivalence between (15) and (13).

Lemma 5.1: (Equivalence between (15) *and* (13)*):* Solving (15) finds the solution of (13).

We propose the following algorithm to find the solution of (15).

$$\dot{\mathbf{V}} = -\eta'[\eta\mathbf{V} - (L \otimes I_q)Y - \mathfrak{B}] - (L \otimes I_q)'(L \otimes I_q)\mathbf{V} \tag{16a}$$

$$\dot{Y} = (L \otimes I_q)'[\eta\mathbf{V} - (L \otimes I_q)Y - \mathfrak{B}] \tag{16b}$$

For the implementation of above algorithm, agent i needs to know its state and the state of its 2-hop neighbors. Hence, it is distributed. The following result characterizes the convergence properties of (16).

Proposition 5.2: (Exponential convergence to the desired solution): The dynamics in (16) find the solution of (15).

From the above discussion, it is clear that we can solve (13) in a distributed way. As a result, each agent would have a copy of the Lagrange multiplier functions $(\lambda(x), \mu(x))$ and their gradients $(\nabla\lambda(x), \nabla\mu(x))$. In the next section, we show that the framework proposed in this section could actually be used to compute other non-distributed terms in the gradient of penalty function and then, we propose our smooth distributed algorithm for finding an optimizer to (9).

VI. Distributed Optimization of the Penalty Function

In this section, we first complete our discussion of the distributed computation of the gradient of the penalty function and propose our distributed algorithm to find an optimizer of the original constrained problem (9).

A. Distributed computation of the gradient

As mentioned in Section IV, if each variable has a copy of the Lagrange multiplier functions $(\lambda(x), \mu(x))$ and their gradients $(\nabla\lambda(x), \nabla\mu(x))$, all that remains to calculate the gradient of the penalty function in a distributed way is the knowledge of $\rho(x)$. All the individual quantities in $\rho(x)$ are similar in nature in the sense they all need an agent to know the values of all the constraints, including the ones in which the agent is not involved. Let us look at how to calculate the first term. The discussion for the remaining two is same.

$$\sum_{j=1}^{m} g_j(x)\nabla_{x_{\bar{i}}}\lambda_j(x) = \sum_{i=1}^{n}\sum_{j=1}^{m}(\mathrm{g}_{ij}/n_j)\nabla_{x_{\bar{i}}}\lambda_j(x) \tag{17}$$

From the above equation, if agent i needs to calculate $\nabla_{x_i} f^\epsilon(x)$, it needs to estimate the above combined contribution from all the agents. To make the sure that the estimate is consistent, we would need all the agents to estimate this. This could again be proved equivalent to solving a linear equation of the form (15). The following result establishes this fact.

Lemma 6.1: (Formulation of estimation of (17) *as a linear equation):* For a given x, estimating the expression of (17) by all the agents is equivalent to solving a problem with the same structure as the linear algebraic equation (15).

Estimating (17) becomes similar to (15) with $q = n$ if we need to estimate it for all the agents, i.e. $\forall\bar{i} = \{1, \ldots, n\}$. Now that we know that (17) is similar to (15), we can use dynamics similar to the one in (16) to solve it. With the above discussion, it is clear that each agent i can calculate $\nabla_{x_i} f^\epsilon(x)$ locally.

B. Dynamics for the optimization of penalty function

Now that each agent i can calculate $\nabla_{x_i} f^\epsilon(x)$ locally, we propose to use the gradient dynamics to find an optimizer of (9). Let us denote by $\chi(x)$, the vector containing the versions of $\lambda(x), \mu(x), \nabla\lambda(x), \nabla\mu(x), \rho(x)$ for all the agents and the auxiliary variable $\mathbf{Y}(x)$ associated with them. Ideally, we would like to implement the following

$$\dot{x} = -\nabla f^\epsilon(x, \chi(x)) \tag{18}$$

But, in practice agents do not know $\chi(x)$ exactly and all the agents are estimating $\chi(x)$ using the dynamics

$$\dot{\hat{\chi}}(x) = -F(x, \hat{\chi}(x))$$

where, $F(x, \hat{\chi}(x))$ is given by an expression similar to (16). $\chi(x)$ would the equilibrium manifold for $\hat{\chi}(x)$. We want to run the dynamics for both x and $\hat{\chi}(x)$ simultaneously and do not want to initialize the dynamics of x at every iteration after $\hat{\chi}(x)$ has converged to somewhere near $\chi(x)$ as the latter would make the run-time of algorithm much higher. Hence, instead of (18), we propose to use the interconnected dynamics

$$\dot{x} = -\nabla f^\epsilon(x, \hat{\chi}(x)) \tag{19a}$$

$$\dot{\hat{\chi}}(x) = -F(x, \hat{\chi}(x)) \tag{19b}$$

A formal characterization of the convergence properties of this dynamics, and particularly of the fact that it finds an optimal solution of (9) if EMFCQ is satisfied, is the subject of our current research.

VII. Simulations

In this section, we illustrate the effectiveness of the proposed gradient based distributed algorithm for a nonlinear optimization problem. For simplicity and clarity of figures, we limit the number of variables to five. We assume the communication topology to be chain, i.e. all the agents are connected in series. The optimization problem that we consider is the following

$$\begin{aligned} \min_x \quad & f(x) = x_1^3 + 5x_2 + 10e^{x_3} + x_4^2 + 2x_5^2 \\ \text{s.t.} \quad & x_4 - x_3 \leq 0 \\ & x_1 - x_2^2 = 0 \end{aligned}$$

We first implement the centralized gradient descent for the penalty function. For implementation of the dynamics in MATLAB, we use first-order Euler discretization with a step-size of 0.002 and value of ϵ and γ as 0.1 and 1, respectively. Then, we implement the distributed gradient dynamics with a step-size of 0.002 for the dynamics of x and a higher step-size of 0.01 for the dynamics of $\hat{\chi}(x)$. Figure 1 shows the evolution of the original objective function $f(x)$ and the unconstrained objective function $f^\epsilon(x)$. Figure 2

displays the convergence of the agents' variables to their optimal values. We start with the initial conditions outside $\mathcal{F}$ and that explains why the value of penalty function is higher than the objective function in the starting. For both the centralized gradient descent as well as the proposed distributed algorithm, the values of the penalty function (and the objective function) and the optimizer after 4000 iterations are the same and given by $f^{\epsilon}(x^*) = 0.3962$ and $x^* = \begin{bmatrix}0.9297 & -0.9642 & -1.3266 & -1.3266 & 0.0000\end{bmatrix}'$, respectively.

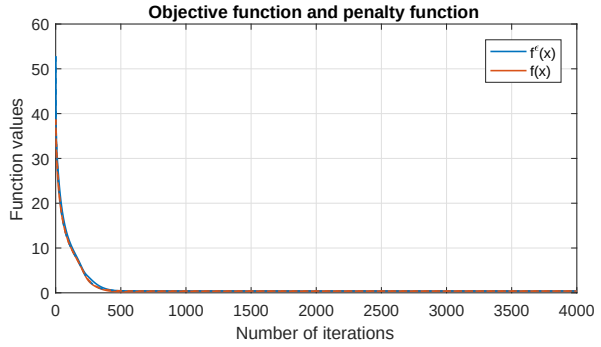

Fig. 1. Evolution of the objective function and penalty function for the centralized gradient descent and the proposed dynamics. Dashed lines represent the evolution using the centralized gradient descent and the solid ones represent the evoultion of functions using the proposed dynamics.

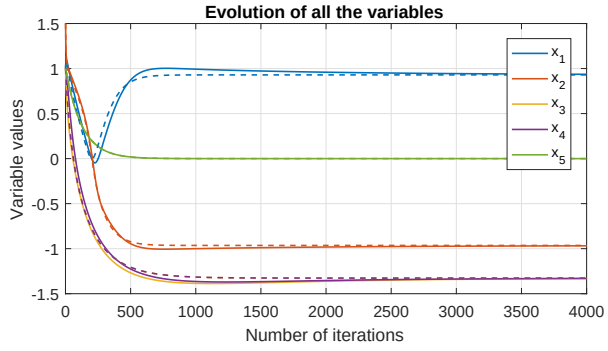

Fig. 2. Convergence of the variables of all the agents to the optimizers. Dashed lines represent the evolution using the centralized gradient descent and the solid ones represent the evoultion of functions using the proposed dynamics.

VIII. Conclusions

We have considered the problem of distributed optimization of nonlinear programs using gradient descent method based on the continuously differentiable exact penalty methods. We propose a framework to implement any gradient based algorithm in a distributed way. We also developed an algorithm to solve linear equations in a distributed way. Our proposed algorithms, both for the solution of optimization problem and the linear algebraic equation are based on gradient descent. Future work will be the formal characterization of the convergence and robustness properties of our algorithm. Also, we would consider subclasses of optimization problems whose structure allows us to simplify our approach and use accelerated methods to further improve the speed of convergence.

Acknowledgements

This work was supported by the ARPA-e Network Optimized Distributed Energy Systems (NODES) program, DE-AR0000695.

References

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

[2] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2012.

[3] J. Wang and N. Elia, "A control perspective for centralized and distributed convex optimization," in *IEEE Conf. on Decision and Control*, Orlando, Florida, 2011, pp. 3800–3805.

[4] A. Cherukuri and J. Cortés, "Initialization-free distributed coordination for economic dispatch under varying loads and generator commitment," *Automatica*, vol. 74, pp. 183–193, 2016.

[5] D. Richert and J. Cortés, "Robust distributed linear programming," *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2567–2582, 2015.

[6] D. Feijer and F. Paganini, "Stability of primal-dual gradient dynamics and applications to network optimization," *Automatica*, vol. 46, pp. 1974–1981, 2010.

[7] A. Cherukuri and J. Cortés, "Distributed generator coordination for initialization and anytime optimization in economic dispatch," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 3, pp. 226–237, 2015.

[8] D. P. Bertsekas, "Convergence of discretization procedures in dynamic programming," *IEEE Transactions on Automatic Control*, vol. 20, no. 6, pp. 415–419, 1975.

[9] ——, *Constrained Optimization and Lagrange Multiplier Methods*. Belmont, MA: Athena Scientific, 1982.

[10] G. Di Pillo and L. Grippo, "Exact penalty functions in constrained optimization," *SIAM Journal on Control and Optimization*, vol. 27, no. 6, pp. 1333–1360, 1989.

[11] G. Di Pillo, "Exact penalty methods," in *Algorithms for Continuous Optimization: The State of the Art*, E. Spedicato, Ed. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1994, pp. 209–253.

[12] T. Glad and E. Polak, "A multiplier method with automatic limitation of penalty growth," *Mathematical Programming*, vol. 17, no. 1, pp. 140–155, 1979.

[13] S. Lucidi, "New results on a continuously differentiable exact penalty function," *SIAM Journal on Optimization*, vol. 2, no. 4, pp. 558–574, 1992.

[14] F. Bullo, J. Cortés, and S. Martínez, "Distributed algorithms for robotic networks," in *Encyclopedia of Complexity and System Science*, R. A. Meyers, Ed. New York: Springer, 2009, entry 00168.

[15] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[16] R. T. Rockafellar, "Augmented Lagrange multiplier functions and duality in nonconvex programming," *SIAM Journal on Control*, vol. 12, no. 2, pp. 268–285, 1974.

[17] S. Mou, J. Liu, and A. S. Morse, "A distributed algorithm for solving a linear algebraic equation," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 2863–2878, Nov 2015.

[18] B. D. O. Anderson, S. Mou, A. S. Morse, and U. Helmke, "Decentralized gradient algorithm for solution of a linear equation," *Numerical Algebra, Control and Optimization*, vol. 6, no. 3, pp. 319–328, 2016.

[19] J. Lu and C. Y. Tang, "A distributed algorithm for solving positive definite linear equations over networks with membership dynamics," *IEEE Transactions on Control of Network Systems*, 2018, to appear.

[20] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Distributed sensor fusion using dynamic consensus," in *IFAC World Congress*, Prague, CZ, Jul. 2005, electronic proceedings.

[21] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Symposium on Information Processing of Sensor Networks*, Los Angeles, CA, Apr. 2005, pp. 63–70.