Distributed line search via dynamic convex combinations

Jorge Cortés   Sonia Martínez

Abstract— **This paper considers multi-agent systems seeking to optimize a convex aggregate function. We assume that the gradient of this function is distributed, meaning that each agent can compute its corresponding partial derivative with state information about its neighbors and itself only. In such scenarios, the discrete-time implementation of the gradient descent method poses the fundamental challenge of determining appropriate agent stepsizes that guarantee the monotonic evolution of the objective function. We provide a distributed algorithmic solution to this problem based on the aggregation of agent stepsizes via adaptive convex combinations. Simulations illustrate our results.**

I. INTRODUCTION

A popular approach to the coordination of multi-agent systems consists of designing a distributed algorithm that solves an optimization problem encoding the coordination task. This top-bottom method has been proven to be very useful in a variety of problems involving multi-vehicle coordination, network utility maximization, energy dispatch, and information processing by sensor networks. Due to a lack of centralized authority, the proposed algorithms are to be executed by employing local information only, which allows for greater scalability and robustness to agent failure. In this paper, we consider a particular class of convex optimization problems for which gradient-descent-type algorithms are naturally distributed. In other words, each agent can compute the partial derivative of the function to be optimized with information of its neighbors and itself. Despite their natural decentralization, the implementation of (steepest) gradient-descent-type algorithms requires global information in order for agents to find a common stepsize. This motivates the question addressed here of how agents can coordinate their stepsize computations.

Literature review. This manuscript contributes to the recent body of research on distributed optimization by a network of agents subject to intermittent interactions. In these works, the objective function can be expressed as a sum of convex functions and be subject to different inequality and equality constraints; see for example [8], [11], [15]. Building on consensus-based coordination rules [2], [10], [9], [6], the aforementioned efforts lead to discrete-time schemes employing function subgradients. Continuous-time approaches which are robust to errors due to communications and initialization include [12] on undirected networks and [5] on directed networks. With the goal of designing faster algorithms, [13], [14] focus on Newton schemes. Except for [14], which employs a decentralized backtracking line-search rule to implement the Armijo rule, the aforementioned approaches assume that agents have access to a common stepsize to implement the distributed algorithm. Another effort which focuses on the solution to linear programs includes [3] and references therein. However, these papers make use of alternative schemes to consensus such as leader election. Our work connects with the literature on algorithms for gradient-descent methods [1]. The classical steepest-descent method [4] for unconstrained minimization converges linearly and can show poor performance. However, the understanding of these algorithms is central for the theory and design of more sophisticated optimization algorithms [7]. It is within this simple context that we study how a network of agents can determine appropriate stepsizes in a distributed way. Our adaptive algorithm belongs to the class of distributed linear iterations which define agreement protocols [2], contributing further to this area of work.

Statement of contributions. We introduce a class of algorithms that allows a group of agents to descend a convex objective function by following an aggregated descent direction. Each agent employs a stepsize that results from a distributed stepsize computation subroutine. This strategy takes as inputs the stepsizes computed by each agent via a line-search procedure. By means of a proper initialization, and after only a finite number of rounds, the strategy outputs a vector of stepsizes that agents can readily implement to decrease the function. If let run indefinitely, the strategy converges to a convex combination of stepsizes that guarantees that the function decreases via the steepest descent direction or other alternative aggregated directions of descent. Most proofs are omitted for space reasons and will appear elsewhere.

Organization. Section II introduces basic notation, notions of graph theory, and line search. Section III states formally the problem of interest. Section IV introduces several stepsize aggregation models for distributed line search based on convex combinations and Section V presents a provable distributed linear iteration algorithm to compute them. This algorithm is analyzed both in continuous and discrete time, and its rate of convergence is characterized. Section VI presents simulations of the resulting algorithms. We gather our conclusions and ideas for future work in Section VII.

II. PRELIMINARIES

This section presents basic notions from graph theory, optimization via gradient descent, and line search.

The authors are with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92093, USA, {cortes,soniamd}@ucsd.edu

A. Notation

We employ $\mathbb{R}^n_{>0}$ (resp. $\mathbb{R}^n_{\geq 0}$) to denote the positive orthant (resp. the nonnegative orthant) of $\mathbb{R}^n$. We use the notation $\mathbf{1}_n \in \mathbb{R}^n_{>0}$ (resp. $\mathbf{1}_{n-1} \in \mathbb{R}^{n-1}_{\geq 0}$) for the vector $(1,\dots,1)^T$ (resp. $\mathbf{1}_{n-1} = (1,\dots,1)^T$). We denote the eigenvalues of a square matrix $M \in \mathbb{R}^{n\times n}$ as $\lambda_i(M)$, $i \in \{1,\dots,n\}$. We assume that the eigenvalues are indexed so that $\mathrm{Re}(\lambda_1(M)) \leq \mathrm{Re}(\lambda_2(M)) \leq \cdots \leq \mathrm{Re}(\lambda_n(M))$, where Re denotes the real part of a complex number. We denote by I_n the identity matrix of dimension $n \times n$. The spectral radius of M is $\rho(M) = \max_{i\in\{1,\dots,n\}} |\lambda_i(M)|$. The essential spectral radius of a matrix M with $\rho(M) = 1$ is $\rho_{\text{ess}}(M) = \max_{i\in\{1,\dots,n\}}\{|\lambda_i(M)| \mid \lambda_i(M) \neq 1\}$. The notation $M \geq 0$ means that M is positive semidefinite. In particular, $M_1 \geq M_2$ if and only if $M_1 - M_2 \geq 0$. A matrix $M \in \mathbb{R}^{n\times n}$ is Metzler if all its off-diagonal elements are nonnegative. A matrix $M \in \mathbb{R}^{n\times n}_{\geq 0}$ is irreducible if, for any nontrivial partition $J \cup K$ of the index set $\{1,\dots,n\}$, there exist $j \in J$ and $k \in K$ such that $m_{jk} \neq 0$. We let $\mathrm{span}\{w_1,\dots,w_l\}$ denote the vector space generated by the vectors $w_1,\dots,w_l \in \mathbb{R}^n$. Given $g : \mathbb{R} \to \mathbb{R}$ and $h : \mathbb{R} \to \mathbb{R}$, we denote $g(r) \in O(h(r))$ if and only if there is $C > 0$ and r_0 such that $|g(r)| \leq C|h(r)|$, for all $r \geq r_0$.

B. Graph-theoretic notions

We present some basic notions from algebraic graph theory following the exposition in [2]. An *undirected graph*, or simply *graph*, is a pair $G = (V, E)$, where V is a finite set called the vertex set and E is the edge set consisting of unordered pairs of vertices. For $i, j \in V$ and $i \neq j$, the set $\{i, j\}$ denotes an undirected edge, and i and j are *neighbors*. We let $\mathcal{N}_G(i)$ denote the set of neighbors of u_i in G. The graph G is *connected* if for any pair of nodes i, j there exists a sequence of edges $\{i, i_1\}, \{i_1, i_2\}, \dots, \{i_k, j\}$ connecting i with j. The *adjacency matrix* of a graph G is a non-negative symmetric matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}_{\geq 0}$ such that $a_{ij} \neq 0$ if and only if $\{i, j\}$ is an edge of the graph. Here, we consider $a_{ij} = 1$, when $\{i, j\} \in E$. Consider the diagonal matrix $D = \mathrm{diag}(A\mathbf{1}_n)$. The *Laplacian matrix* of G is defined as $L = D - A$, which is a symmetric and positive semi-definite matrix. Note that L has an eigenvalue at 0 and $\mathbf{1}_n$ is the corresponding eigenvector. A graph G is connected if and only if L is irreducible and 0 is a simple eigenvalue. Finally, a map $g : \mathbb{R}^n \to \mathbb{R}^n$ is *distributed over* G if, for all $j \in \{1,\dots,n\}$, the component g_j can be expressed as $g_j(x) = g_j(x_{i_1},\dots,x_{i_{n_j}})$, where $\mathcal{N}_G(j) = \{i_1,\dots,i_{n_j}\}$, for all $x \in \mathbb{R}^n$.

C. Directions of descent and line search

Given a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, we let $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ denote its gradient

$$\nabla f(x) = \Big(\frac{\partial f}{\partial x_1}(x), \dots, \frac{\partial f}{\partial x_n}(x)\Big).$$

Throughout the paper, we use the notation $\nabla_i f$ to refer to the ith component of ∇f. Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and $x \in \mathbb{R}^n$, $v \in \mathbb{R}^n$ is a direction of descent of f at x if there exists $T > 0$ such that

$$f(x + \delta v) < f(x), \quad \delta \in (0, T).$$

If f is continuously differentiable at x, this is equivalent to saying that $\nabla f(x)^T v < 0$. The procedure of calculating the actual step δ to be taken in the direction v is called *line search*. The choice of a stepsize and a direction that guarantees the reduction of the function at each iterate leads to various gradient algorithms. In particular, one could aim to find the best stepsize that optimizes the decrease in the value of f along a direction v, i.e.,

$$\epsilon_v = \mathrm{argmin}_{\delta\in[0,\infty)} f(x + \delta v). \tag{1}$$

Let $h_v(\delta) = f(x + \delta v)$. For a continuously differentiable function, it is not difficult to see that the stepsize (1) is characterized by the equation

$$h'_v(\epsilon_v) = \nabla f(x + \epsilon_v v)^T v = 0. \tag{2}$$

The choice $v = \nabla f$ (which corresponds to the direction that instantaneously descends f the most) leads to the steepest-descent method,

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k), \quad k \geq 0,$$

which locally converges to the set of minimizers of f.

III. Problem statement

Consider a network of n agents, indexed by $i \in \{1,\dots,n\}$, with interaction topology described by a graph G. The network state, denoted x, belongs to $\mathbb{R}^n$. Agent i is responsible for the ith component $x_i \in \mathbb{R}$. The results that follow can also be extended to scenarios where each agent supervises several components of the vector $x \in \mathbb{R}^n$, but here we keep the exposition simple. Consider a convex function $f : \mathbb{R}^n \to \mathbb{R}$ whose gradient $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ distributed over G. Thus, each agent $i \in \{1,\dots,n\}$ can compute

$$v_i(x) = (0, \dots, \nabla_i f(x), \dots, 0), \tag{3}$$

with information from its neighbors in G. The next result states that the line search procedure for f and each direction v_i can be carried out in a distributed way.

Lemma 3.1: (Individual agent stepsize computation): Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable and assume $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is distributed over G. Let $x \in \mathbb{R}^n$ and $i \in \{1,\dots,n\}$ with $v_i(x)$, as defined in (3), be nonvanishing. Then, the optimal stepsize $\epsilon_{v_i(x)}$ along $v_i(x)$ and the associated decrease $\Delta_{v_i(x)}$ in the value of f can be computed with knowledge only of $\{x_i\} \cup \{x_j \mid j \in \mathcal{N}_G(i)\}$.

Proof: For simplicity, we use the shorthand notation h_i, ϵ_i, and Δ_i to denote $h_{v_i(x)}$, $\epsilon_{v_i(x)}$, and $\Delta_{v_i(x)}$ respectively. Note that (2) in this case reduces to

$$h'_i(\epsilon_i) = \nabla_i f(x + \epsilon_i v_i(x))^T \nabla_i f(x) = 0. \tag{4}$$

The only difference between $x + \epsilon_i v_i(x)$ and x is in the ith component, which agent i is responsible for. Since the gradient of f is distributed over G, agent i has all the information required to solve equation (4). A similar argument holds for the associated decrease in the value of f,

$$\begin{aligned}\Delta_i &= f(x) - f(x + \epsilon_i v_i(x)) \\ &= h_i(0) - h_i(\epsilon_i) = -\int_0^{\epsilon_i} h_i'(\delta)\, d\delta \\ &= -\int_0^{\epsilon_i} \nabla_i f(x + \delta v_i(x))^T \nabla_i f(x)\, d\delta, \end{aligned} \tag{5}$$

which completes the result. ■

Note that the line search procedure performed by agent i assumes that all other agents remain fixed. The problem of interest to this paper is the following.

Distributed line-search computation problem. Let $x \in \mathbb{R}^n$. Given δ_i such that $f(x + \delta_i v_i(x)) < f(x)$, where $v_i(x)$ is given as in (3) for all $i \in \{1, \dots, n\}$, design a distributed algorithm that allows the group of agents to agree on stepsizes $(\epsilon_1, \dots, \epsilon_n) \in \mathbb{R}^n_{\geq 0}$ such that

$$f(x + \epsilon_1 v_1(x) + \dots + \epsilon_n v_n(x)) < f(x).$$

In particular, a solution such that $\epsilon_i = \epsilon$ for all $i \in \{1, \dots, n\}$, solves the *distributed steepest-descent line-search computation problem.*

We make the following considerations regarding the above problem. First, note that the choice $\epsilon_i = \delta_i$, $i \in \{1, \dots, n\}$, is not a solution in general. In principle, there are several ways to approach this problem. For instance, one can resort to parallel algorithms to identify those agents that maximize the function decrease and coordinate their changes in state accordingly via leader election. Instead, here we look for solutions that allow all agents to simultaneously contribute to the decrease of the function.

IV. Weighted network-aggregated stepsizes

The next result provides guidance as to how the problem stated can be solved. Lemma 4.1 determines how stepsizes based on a convex combination guarantee the decrease of the cost function.

Lemma 4.1: (Network-aggregated stepsize): Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex. For $x \in \mathbb{R}^n$, let $w_1, \dots, w_n \in \mathbb{R}^n$ be directions of descent of f from x. Let $\delta_i \in \mathbb{R}_{>0}$ be a stepsize such that $f(x+\delta_i w_i) < f(x)$, for each $i \in \{1, \dots, n\}$. Let $\mu_i \in [0, 1]$, for $i \in \{1, \dots, n\}$, such that $\mu_1 + \dots + \mu_n = 1$. Then $\mu_1\delta_1 w_1 + \dots + \mu_n \delta_n w_n$ is an aggregated direction of descent of f from x, and $f\left(x + \delta \sum_{i=1}^n \mu_i \delta_i w_i\right) < f(x)$.

Note that the aggregation procedure in Lemma 4.1 reduces the size of the agent stepsizes, i.e., $\mu_i \delta_i < \delta_i$, $i \in \{1, \dots, n\}$. This makes sense as the individual agent stepsizes have been computed with the overly optimistic assumption that nobody else would change its state. This reduction in stepsize is the price that the agents have to pay to make sure the aggregate function decreases. The following are particular cases of stepsizes that we consider in the sequel. With the notation of Lemma 4.1, let $w_i = v_i(x)$ be given by (3). The *common network-aggregated stepsize vector* is

$$\mu_i = \frac{\frac{1}{\delta_i}}{\frac{1}{\delta_1} + \dots + \frac{1}{\delta_n}}, \qquad i \in \{1, \dots, n\}. \tag{6}$$

By using this stepsize vector, agents decrease the function along $\nabla f(x)$. The *proportional-to-cost network-aggregated stepsize vector* is

$$\mu_i = \frac{\Delta_i}{\Delta_1 + \dots + \Delta_n}, \qquad i \in \{1, \dots, n\}, \tag{7}$$

where $\Delta_i = f_i(x) - f_i(x + \delta_i v_i(x))$, for $i \in \{1, \dots, n\}$. Finally, the *proportional-to-state network-aggregated stepsize vector* is

$$\mu_i = \frac{d_i}{d_1 + \dots + d_n}, \qquad i \in \{1, \dots, n\}, \tag{8}$$

where $d_i = \delta_i \|v_i(x)\|$, for $i \in \{1, \dots, n\}$. Note that the weights defined in (7) are larger for those agents who offer a larger decrease in the value of the objective function. Thus, they encode a type of "proportional fairness" in the way that each agent can decrease the cost function. A similar consideration applies to (8). We call the resulting direction of descent *proportional-to-cost* (resp. *proportional-to-state*) direction of descent.

Lemma 4.1 paves the way for performing line search in a distributed way. Using this result, the agents in the network can collectively fuse their stepsizes in order to guarantee that the resulting network state after updates by all agents decreases the value of the objective function. Remarkably, this is accomplished without the need to share the individual directions of motion of the agents. In particular, the aggregated stepsize models (6)-(8) take into account the current network state in the determination of the appropriate stepsizes. The challenge is then to perform these stepsize aggregations in a distributed way. We address this in the following section.

V. Adaptive algorithm for distributed stepsize computation

One can implement a number of distributed algorithms to compute the stepsizes (6)-(8) across the whole network. For instance, average consensus could be employed to compute the corresponding aggregate sums in the denominators of these expressions. This, together with knowledge of the size of the network, would allow each agent to compute the aggregated stepsize. However, the convergence of these algorithms is typically asymptotic, and so it may appear impractical to execute one at each state through the evolution of the network. Instead, we would like to find distributed algorithms that, for each $x \in \mathbb{R}^n$, even if not implementing exactly the models (6)-(8), (i) can guarantee that the function decreases and (ii) approach asymptotically the directions of descent and stepsizes provided in Lemma 4.1.

A. Distributed computation of convex combinations

We start by noting that the aggregated stepsize models (6)-(8) have a similar structure that can be captured as follows: given a vector $y \in \mathbb{R}^n_{>0}$, compute the aggregated vector

$$(y^T \mathbf{1}_n)^{-1} y.$$

Each model corresponds to a different choice of vector y. Here, we propose a continuous-time distributed algorithm that performs this computation.

Define the matrix $Q(y) \in \mathbb{R}^{n\times n}$ such that $Q_{ij}(y) = -y_i y_j$, for $(i,j) \in E$, and $Q_{ii}(y) = \sum_{j\in\mathcal{N}_G(i)} y_j^2$, $i \in \{1,\dots,n\}$, and consider the function $V : \mathbb{R}^n \to \mathbb{R}$, given by $V(\mu) = \frac{1}{2}\mu^T Q(y)\mu$. In the sequel, we consider that the network of agents interacts over a connected and undirected graph, G. Since G is undirected, it is easy to verify that $V(\mu) = \frac{1}{2}\sum_{i=1}^n \sum_{j\in\mathcal{N}_G(i)} (y_j\mu_i - y_i\mu_j)^2$. Three important properties of the matrix $Q(y)$ are that: (i) $Q(y) = Q(y)^T$, (ii) $-Q(y)$ is Metzler, and (iii) $Q(y)$ is irreducible (because G is connected).

Let us now define the quadratic program

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\mu^T Q(y)\mu, && \text{(9a)}\\ &\text{subject to} \quad \mathbf{1}_n^T \mu = 1. && \text{(9b)} \end{aligned}$$

Lemma 5.1: The unique solution to (9) is given by $\mu^\star = (y^T\mathbf{1}_n)^{-1} y$.

It is easy to see that $Q(y)\mu^\star = 0$. The previous lemma encodes key properties of $Q(y)$ and leads us to design the following distributed algorithm,

$$\dot{\mu} = -LQ(y)\mu, \qquad \mu(0) = \mu_0, \tag{10}$$

where $\mu_0 \in \mathbb{R}^n_{>0}$ satisfies $\mu_0^T \mathbf{1}_n = 1$. In coordinates, this can be rewritten as

$$\begin{aligned} \dot{\mu}_i &= -\sum_{j\in\mathcal{N}_G(i)} a_{ij}(\nabla_j V(\mu) - \nabla_i V(\mu)),\\ \mu_i(0) &= \mu_{0,i}, \end{aligned}$$

where $\nabla_i V(\mu) = 2\sum_{k\in\mathcal{N}_G(i)} (y_k^2\mu_i - y_i y_k \mu_k)$, leading to a distributed algorithm over G.

Note that the dynamical system (10) leaves $\mu(t)^T\mathbf{1}_n = 1$ invariant for all $t \in \mathbb{R}_{\geq 0}$. This can be verified by noting that $\dot{\mu}(t)^T \mathbf{1}_n = (\mu(t))^T Q L \mathbf{1}_n = 0$. The following result holds.

Lemma 5.2: For any $\mu_0 \in \mathbb{R}^n_{>0}$ such that $\mu_0^T\mathbf{1}_n = 1$, the solution of (10) converges asymptotically to $\mu^\star$, the solution to the quadratic program (9).

B. Discrete-time implementation and rate of convergence

Even though convex combinations are preserved by the dynamics (10), the algorithm does not leave $\mathbb{R}^n_{\geq 0}$ invariant because the matrix $-LQ$ is not positive. This means that the algorithm can not be stopped anytime to output an appropriate set of stepsizes. The algorithm also requires a continuous-time implementation, which prescribes communications to occur infinitely often. Because of these considerations, here we focus on its discrete-time implementation, and in particular, on the study of its rate of convergence.

Using a first-order Euler discretization, (10) becomes

$$\mu^{k+1} = (I_n - hLQ(y))\mu^k, \tag{11}$$

where $\mu^0 \in \mathbb{R}^n_{>0}$ satisfies $(\mu^0)^T\mathbf{1}_n = 1$. It can be seen that $(\mu^{k+1})^T\mathbf{1}_n = (\mu^k)^T(I_n - hQ(y)L)\mathbf{1}_n = (\mu^k)^T\mathbf{1}_n = 1$. The next result provides a sufficient condition on the stepsize h that guarantees convergence.

Lemma 5.3: For any $\mu^0 \in \mathbb{R}^n_{>0}$ such that $(\mu^0)^T\mathbf{1}_n = 1$, the solution of (11) converges asymptotically to $\mu^\star = (y^T\mathbf{1}_n)^{-1}y$ under the assumption that

$$h < \frac{2}{\lambda_n(L)\lambda_n(Q(y))}.$$

Moreover, the essential spectral radius of $I_n - hLQ(y)$ is upper bounded by $1 - h\lambda_2(L)\lambda_2(Q(y))$.

Using the bound on the essential spectral radius of $I_n - hLQ(y)$, we next determine a bound on the rate of convergence of the algorithm as follows.

Lemma 5.4: Let $r > 0$, and let $T_r > 0$ be the time it takes (11) to reach and remain in the ball of center $\mu^\star$ with radius r. Then

$$T_r \in O\left(\frac{1}{h\lambda_2(L)\lambda_2(Q(y))}\log\left(\frac{\|\mu^0 - F^\star\mu^0\|_2}{r}\right)\right),$$

where $F^\star = \frac{yz^T}{z^Ty}$ and z is the right eigenvector of $I_n - hLQ(y)$ with eigenvalue 1.

Remark 5.5 (Extension to $y \in \mathbb{R}^n_{\geq 0}$): In the previous two subsections we have assumed that $y_i > 0$ for all $i \in \{1,\dots,n\}$. The results can be extended for the case when $y_i = 0$, for some $i \in \{1,\dots,n\}$, by assuming that these nodes act as a relay between any of their neighbors in $G = (V,E)$. To see this, without loss of generality, suppose $y_1 = 0$ only for $i = 1$. In this case, the matrix $Q(y)$ will have an additional eigenvector, $e_1 = (1,0,\dots,0)^T \in \mathbb{R}^n$, with zero eigenvalue. Consider the graph $\overline{G} = (\overline{V},\overline{E})$ where $\overline{V} = \{2,\dots,n\}$, and $\{i,j\} \in \overline{E}$ if and only if $\{i,j\} \in E$ or $\{1,i\}, \{1,j\} \in E$. Let $\overline{L}$ be the associated graph Laplacian. Let $\overline{Q}(y)$ be the restriction of $Q(y)$ over $\mathbb{R}^n \setminus \operatorname{span}\{e_1\}$. System (11) can be replaced by

$$\begin{aligned} \mu_1^{k+1} &= \mu_1^k,\\ \overline{\mu}^{k+1} &= (I_n - h\overline{LQ}(y))\overline{\mu}^k, \end{aligned}$$

where a similar bound for h as in Lemma 5.3 can be taken, and $(\mu^0)^T\mathbf{1}_n = 1$. The analysis of the subsystem in $\overline{\mu}$ is similar to the one in μ in (11). First, it can be seen that $(\mu^k)^T\mathbf{1}_n = 1$ for all $k \geq 1$. More precisely, $(\mu^{k+1})^T\mathbf{1}_n = \mu_1^{k+1} + (\overline{\mu}^{k+1})^T\mathbf{1}_{n-1} = \mu_1^k + (\overline{\mu}^k)^T(I_n - h\overline{LQ}(y))^T\mathbf{1}_{n-1} = \mu_1^k + (\overline{\mu}^k)^T\mathbf{1}_{n-1} = 1$. In particular, we have that $(\overline{\mu}^0)^T\mathbf{1}_{n-1}$ is conserved. The analysis of the system, is similar to the previous discrete-time implementation, and it can be seen that it converges to the convex combination $\mu^\star = (\mu_1^0, \overline{\mu}^\star)$, where

$$\overline{\mu}_i^\star = \frac{(\overline{\mu}^0)^T \mathbf{1}_{n-1}}{y^T \mathbf{1}_n} y_i, \quad i \in \{2, \ldots, n\}. \qquad \bullet$$

C. Distributed line-search computation algorithm

Here, we describe a distributed algorithm that allows agents to adapt their step-sizes and solve approximately the distributed (steepest-descent) line-search computation problem. Agents start from an initial condition μ_0 such that $\mu_0^T \mathbf{1}_n = 1$ (e.g., $\mu_0 = \frac{1}{n}\mathbf{1}_n$). Note that the assumption that agents know n is necessary since it is equal to the dimension of $x \in \mathbb{R}^n$. Then, agents implement (11) for an agreed number of rounds N that guarantees $\mu_i^k \geq 0$, for all $i \in \{1, \ldots, n\}$. The algorithm is formally described in Algorithm 1.

Algorithm 1: DISTRIBUTED WEIGHTED STEPSIZE

Executed by: Each agent $i \in \{1, \ldots, n\}$
Data: the function f, the state x, the number of rounds $N \in \mathbb{N} \cup \{0\}$, and aggregated stepsize model $\mathcal{R}$

set $v_i(x) = -(0, \ldots, 0, \nabla_i f(x), 0, \ldots, 0)$
compute stepsize $\delta_i = \epsilon_i > 0$ satisfying $\nabla_i f(x + \epsilon_i v_i(x))^T \nabla_i f(x) = 0$
set y_i corresponding to aggregated stepsize model $\mathcal{R}$, send y_i to neighbors, receive $\{y_j \mid j \in \mathcal{N}_G(i)\}$, and compute $Q_{ij}(y)$ for $j \in \mathcal{N}_G(i)$
set $\mu_i^0 = \frac{1}{n}$
for $l \in \{1, \ldots, N\}$ **do**
  $\mu_i^l = ((I_n - hLQ(y))\mu^{l-1})_i$
  send μ_i^l to neighbors, receive $\{\mu_j^l \mid j \in \mathcal{N}_G(i)\}$
end
set $m_i^0 = \mu_i^N$
send m_i^0 to neighbors, receive $\{m_j^0 \mid j \in \mathcal{N}_G(i)\}$
for $l \in \{1, \ldots, N\}$ **do**
  $m_i^l = \min\{m_i^{l-1}, m_j^{l-1} \mid j \in \mathcal{N}_G(i)\}$
  send m_i^l to neighbors, receive $\{m_j^l \mid j \in \mathcal{N}_G(i)\}$
end
while $m_i^n < 0$ **do**
  reassign $\mu_i^N = ((I_n - hLQ(y))\mu^N)_i$
  reset $m_i^0 = \mu_i^N$
  send m_i^0 to neighbors, receive $\{m_j^0 \mid j \in \mathcal{N}_G(i)\}$
  for $l \in \{1, \ldots, N\}$ **do**
    $m_i^l = \min\{m_i^{l-1}, m_j^{l-1} \mid j \in \mathcal{N}_G(i)\}$
    send m_i^l to neighbors, receive $\{m_j^l \mid j \in \mathcal{N}_G(i)\}$
  end
end
change state from x_i to $x_i + \mu_i^N \delta_i \nabla_i f(x)$

The different aggregated stepsize models (6)-(8) are captured in the algorithm via $\mathcal{R}$. In this way, the choice $y_i = \frac{1}{\delta_i}$ leads to DISTRIBUTED WEIGHTED STEPSIZE (and results in the steepest descent direction). As N grows, this leads to the common network-aggregated stepsize vector (6). The choice $y_i = \Delta_i$, $i \in \{1, \ldots, n\}$ as in (7) leads to the DISTRIBUTED WEIGHTED STEPSIZE for the proportional-to-cost descent direction. As N grows, this leads to the proportional-to-cost network aggregated stepsize vector (7). Finally, the choice $y_i(f, x) = d_i$, $i \in \{1, \ldots, n\}$ as in (8) leads to the DISTRIBUTED WEIGHTED STEPSIZE for the proportional-to-state descent direction. As N grows, this leads to the proportional-to-state network aggregated stepsize vector (7).

The DISTRIBUTED WEIGHTED STEPSIZE algorithm can be informally described as follows. In order to implement a step of the gradient-descent algorithm, each agent outputs first a set of stepsizes μ_i^N, $i \in \{1, \ldots, n\}$. These stepsizes are obtained after applying (11) during N iterations. After this, if all of the μ_i^N are positive or zero, which happens when $m_i^n = \min_{j\{1,\ldots,n\}} \mu_j^N \geq 0$, for all $i \in \{1, \ldots, n\}$, then the gradient-descent procedure can be safely implemented. Otherwise, agents iterate (11) additional times until the property $\mu_i^N \geq 0$, $i \in \{1, \ldots, n\}$ holds. The algorithm assumes that $y_i > 0$, for all $i \in \{1, \ldots, n\}$. When $y_i = 0$, agent i should relay information from neighbors to other neighbors at any communication round.

We need a formal result that states properties of algorithm and its output.

VI. SIMULATIONS

In this section, we include some numerical experiments on a simple mathematical example to illustrate the results. We consider a network of 8 agents subject to a fixed topology corresponding to the graph G depicted in Figure 1. The

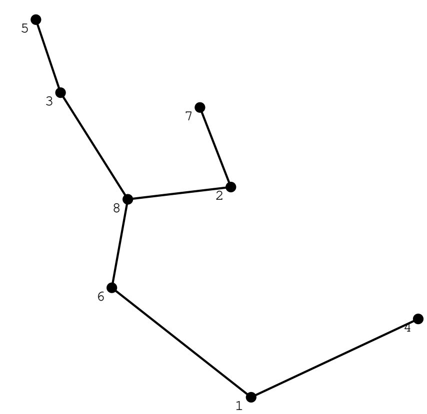

Fig. 1. Graph of 8 networked agents for the simulation example

function to be optimized $f : \mathbb{R}^8 \to \mathbb{R}^8$ is defined as $f(x) = x^T(I_n + L)x + q^T x$, where $q = (1, -1, 2, 1, 0, -1, 1, 0) \in \mathbb{R}^8$ and L is the graph Laplacian associated with G. It is easy to verify that f is convex and distributed over G.

We implement the algorithm in two main situations. First, we consider DISTRIBUTED WEIGHTED STEPSIZE for the steepest descent direction. Figure 2 compares how the function is decreased by the centralized steepest descent method (blue), the conservative steepest descent method if all agents had the information to compute the stepsize in (6) (red), and the algorithm DISTRIBUTED WEIGHTED STEPSIZE for the steepest descent direction with $N = 4$ (green) and an appropriate h. After $N = 4$, all weights μ_i^N have become positive. As it can be seen, both the conservative steepest

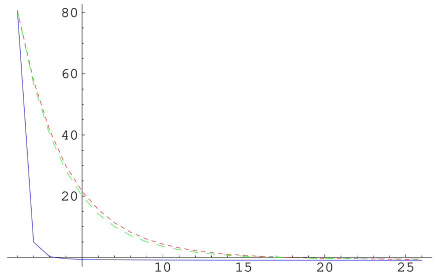

Fig. 2. Evolution of various algorithms to decrease f along the steepest descent direction

descent method and its decentralized version are very close even if the number of rounds ($N = 4$) used to compute the stepsizes in a distributed way is small. The differences between the centralized steepest descent method and the other two are to be expected, as the common network-aggregated stepsize vector (6) is more conservative in order to guarantee that the function is still decreased.

Second, we consider DISTRIBUTED WEIGHTED STEPSIZE for proportional-to-cost descent. Figure 3 compares the evolutions of the gradient algorithms following the steepest descent (blue), the decentralized proportional-to-cost descent method if all agents had the information to compute the stepsize as in (7) (red), and DISTRIBUTED WEIGHTED STEPSIZE for proportional-to-cost descent with $N = 4$ (green) and an appropriate h. After $N = 4$, all weights μ_i^N are already positive. Similarly as before, and as expected,

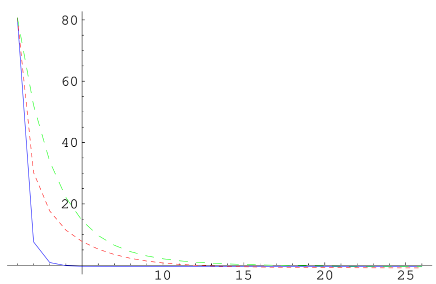

Fig. 3. Evolution of various algorithms to decrease f along the steepest descent direction and proportional-to-cost aggregated direction

the function is decreased less rapidly by means of the conservative proportional-to-cost descent direction method and its decentralized version via DISTRIBUTED WEIGHTED STEPSIZE when compared to the centralized steepest descent method. However the previous two are relatively close even though the number of rounds $N = 4$ used in DISTRIBUTED WEIGHTED STEPSIZE is low.

VII. CONCLUSIONS

We have considered networked scenarios where a group of agents seeks to optimize a convex aggregate function using gradient information. We have presented a novel distributed algorithm for the computation of aggregated stepsizes that guarantee the decrease of the objective function. We have analyzed the properties of this strategy when implemented both in continuous and discrete time, and characterized its rate of convergence. With a proper initialization, the algorithm gives rise to a convex combination after a finite number of rounds, and can therefore be implemented to fuse the stepsizes of individual agents. Simulations illustrate the results. Future work will be devoted to the analytical characterization of the performance of the proposed strategies, the consideration of scenarios with switching communication graphs, and the design of distributed line search strategies for higher-order (e.g., Newton) schemes.

ACKNOWLEDGMENTS

Both authors wish to thank Jon Nicolás and Alexandra Cortés-Martínez for constant inspiration and joy. This work was partially supported by grants NSF CMMI-1300272 (JC) and AFOSR-11RSL548 (SM).

REFERENCES

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

[2] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. American Mathematical Society. Princeton University Press, 2009. Available at http://www.coordinationbook.info.

[3] M. Burger, G. Notarstefano, and F. Allgower. Distributed robust optimization via cutting-plane consensus. In *IEEE Int. Conf. on Decision and Control*, pages 7457–7463, Maui, December 2012.

[4] A. Cauchy. Méthode générale pour la résolution des systems d'equations simultanées. *Comptes rendus de l'Académie des Sciences*, 25:46–89, 1847.

[5] B. Gharesifard and J. Cortés. Distributed continuous-time convex optimization on weight-balanced digraphs. *IEEE Transactions on Automatic Control*, 59(3), 2014. To appear.

[6] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[7] D.G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, Reading, Mass., 1973.

[8] A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[9] R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in multi-agent networked systems. *Proceedings of the IEEE*, 2006.

[10] W. Ren and R. Beard. *Distributed coordination of multi-vehicle cooperative control—Theory and applications*. Communications and Control Engineering Series. Springer-Verlag, 2008.

[11] P. Wan and M. D. Lemmon. Event-triggered distributed optimization in sensor networks. In *Symposium on Information Processing of Sensor Networks*, pages 49–60, San Francisco, CA, 2009.

[12] J. Wang and N. Elia. A control perspective for centralized and distributed convex optimization. In *IEEE Int. Conf. on Decision and Control*, pages 3800–3805, Orlando, Florida, 2011.

[13] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato. Newton-Raphson consensus for distributed convex optimization. *IEEE Transactions on Automatic Control*, 2013. Submitted.

[14] M. Zargham, A. Ribeiro, and A. Jadbabaie. A distributed line search for network optimization. In *American Control Conference*, pages 472–477, Montreal, Canada, 2012.

[15] M. Zhu and S. Martínez. On distributed convex optimization under inequality and equality constraints via primal-dual subgradient methods. *IEEE Transactions on Automatic Control*, 57:151–164, 2012.