A Decentralized Prediction-Correction Method for Networked Time-Varying Convex Optimization

Andrea Simonetto[§], Aryan Mokhtari[⋆], Alec Koppel[⋆], Geert Leus[§], and Alejandro Ribeiro[⋆]

[§]Dept. of EEMCS, Delft University of Technology, 2826 CD Delft, The Netherlands
{a.simonetto, g.j.t.leus}@tudelft.nl
[⋆]Dept. of ESE, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104, USA
{aryanm, akoppel, aribeiro}@seas.upenn.edu

***Abstract*—We study unconstrained time-varying convex optimization problems where the objective function is comprised of components that are revealed to distinct nodes of a network. We propose a distributed algorithm to find and track the solution trajectory based which samples the problem at discrete time steps. To do so, we develop a method based on considering prediction and gradient-based correction steps (DePCoT), while sampling the problem data at a constant rate of $1/h$. We establish that the asymptotic error bound behaves as $O(h^2)$, which outperforms the state of the art existing error bound of $O(h)$ for correction-only methods. Our main technical contributions are the prediction step and a decentralized method to approximate the Hessian inverse of the objective, which yields quantifiable trade-offs between communication and accuracy.**

I. Introduction

In this paper, we consider convex optimization problems where the objective function changes continuously in time and its components are available at distinct nodes of a network. The objective function can be decomposed into two parts: the first part is locally available at each node and the second part is shared between neighboring nodes. To be more precise, consider a connected undirected network containing n nodes where $\boldsymbol{y}^i \in \mathbb{R}^p$ is the decision variable of node i. Define $\boldsymbol{y} = [\boldsymbol{y}^1; \ldots; \boldsymbol{y}^n] \in \mathbb{R}^{np}$ as the concatenation of the decision variables. Nodes aim at cooperatively minimizing the global cost function $F : \mathbb{R}^{np} \times \mathbb{R}_+ \to \mathbb{R}$, which can be written as the sum of a locally available function $\Phi : \mathbb{R}^{np} \times \mathbb{R}_+ \to \mathbb{R}$ and a network related function $G : \mathbb{R}^{np} \times \mathbb{R}_+ \to \mathbb{R}$. Therefore, nodes minimize

$$\underset{\boldsymbol{y}\in\mathbb{R}^{np}}{\operatorname{argmin}}\; F(\boldsymbol{y};t) \;:=\; \underset{\boldsymbol{y}\in\mathbb{R}^{np}}{\operatorname{argmin}}\; \Phi(\boldsymbol{y};t) + G(\boldsymbol{y};t)\ . \tag{1}$$

Notice that nodes can minimize the objective function $\Phi(\boldsymbol{y};t)$ independently, while minimization of the objective function $G(\boldsymbol{y};t)$ requires coordination and information exchange between nodes. Problems of form (1) arise in time-varying versions of multiuser network optimization and resource allocation, see [1], [2] for time-invariant distributed algorithms for these problems.

We consider using the tools of *non-stationary* optimization [3]–[5] to solve problems of the form (1) by prediction-correction algorithms. We begin by reformulating (1) to a manner better suited to decentralized optimization and define a discretized version of (1) (Section II) by sampling it at a constant rate $1/h$. Then, we discuss the Gradient Trajectory Tracking (GTT) algorithm which uses a prediction-correction scheme for minimizing dynamic optimization problems in centralized settings (Section II-A). GTT predicts the optimal solution at the discrete time instance t_{k+1} by approximating variation of the objective function F from t_k to t_{k+1} and corrects the predicted solution by executing a single step of projected gradient descent. GTT may not be applied to decentralized optimization problems since the prediction step requires inverting a global Hessian of the objective F which is not amenable to decentralized computation. We propose a Decentralized Prediction-Correction Tracking (DePCoT) algorithm which approximates the prediction direction of GTT by truncating the Taylor series of the objective function Hessian inverse (Section II-B). We show a trade-off in the implementation of DePCoT between approximation accuracy and communication cost. We then analyze convergence properties of DePCoT (Section III) and establish that under some specific conditions algorithm converges linearly to a tracking error of $O(h^2)$ (Theorem 1). This result improves the error of state-of-the-art decentralized correction-only methods (so-called running methods) [6]–[8] which is in the order of $O(h)$. Finally, we present numerical simulations of an estimation problem of a spatially distributed process. The proofs of the results are available in [9].

This research is supported by STW under the D2S2 project from the ASSYS program, NSF CAREER CCF-0952867, and ONR N00014-12-1-0997.

II. Problem formulation and algorithm definition

Begin by considering a connected undirected network with n nodes and define $\mathcal{N}^i$ as the neighborhood of node i, i.e., the nodes it can exchange information with. Associate with node i a vector $\boldsymbol{y}^i \in \mathbb{R}^p$ as its decision variable and a time-varying local function $\phi^i : \mathbb{R}^p \times \mathbb{R}_+ \to \mathbb{R}$ with input arguments are the local variable $\boldsymbol{y}^i$ and time t. Further, define $g^{i,j} : \mathbb{R}^p \times \mathbb{R}^p \times \mathbb{R}_+ \to \mathbb{R}$ as a common objective function between nodes i and j which takes $\boldsymbol{y}^i$, $\boldsymbol{y}^j$, and t as inputs. The shared functions are symmetric $g^{i,j}(\boldsymbol{y}^i, \boldsymbol{y}^j; t) = g^{j,i}(\boldsymbol{y}^j, \boldsymbol{y}^i; t)$. Nodes aim at minimizing their local objective function ϕ^i while they collaborate with their neighbors to minimize the shared functions $g^{i,j}$: i.e., nodes aim at cooperatively solving the problem

$$\min_{\boldsymbol{y}^1,\ldots,\boldsymbol{y}^n} \sum_{i=1}^{n} \Big(\phi^i(\boldsymbol{y}^i;t) + g^{i,i}(\boldsymbol{y}^i,\boldsymbol{y}^i;t) + \sum_{j\in\mathcal{N}^i} g^{i,j}(\boldsymbol{y}^i,\boldsymbol{y}^j;t)\Big). \tag{2}$$

We stack the local decision variables into $\boldsymbol{y} = [\boldsymbol{y}^1; \ldots; \boldsymbol{y}^n] \in \mathbb{R}^{np}$ as in (1) and define the time-varying objective functions

$$\Phi(\boldsymbol{y};t) := \sum_{i=1}^{n} \phi^i(\boldsymbol{y}^i;t)\ ,\quad G(\boldsymbol{y};t) := \sum_{i=1}^{n} \sum_{j=i,j\in\mathcal{N}^i} g^{i,j}(\boldsymbol{y}^i,\boldsymbol{y}^j;t)\ , \tag{3}$$

in order to write the problem in (2) as (1).

To solve the dynamic optimization problem in (2) or its equivalent (1), the first step is sampling the objective function F at time instants t_k with $k = 0, 1, 2, \ldots$ which leads to the problem of finding the discrete-time globally optimal sequence

$$\boldsymbol{y}^*(t_k) \;:=\; \underset{\boldsymbol{y}\in\mathbb{R}^{np}}{\operatorname{argmin}}\; F(\boldsymbol{y};t_k) \qquad k \geqslant 0\ . \tag{4}$$

We aim to generate a discrete sequence $\{\boldsymbol{y}_k\}$ which remains close to the optimal trajectory $\boldsymbol{y}^*(t_k)$. In the following section we study the Gradient Trajectory Tracking (GTT) algorithm as a centralized method for solving the sequence of optimization problems in (4).

A. Gradient Trajectory Tracking

The GTT method executes a prediction-correction scheme to first estimate the change of optimal arguments from t_{k-1} to t_k and then correct the predicted solution by running a step of gradient descent [10]. The prediction step is built on modeling evolution of the iso-residual trajectory $\boldsymbol{y}(t)$. For each $\boldsymbol{y}(t)$, we can write $\nabla_{\boldsymbol{y}} F(\boldsymbol{y};t) = \boldsymbol{r}(t)$, where $\boldsymbol{r}(t)$ is a residual vector that is null at optimality. By perturbing this gradient condition for variations δt and $\delta \boldsymbol{y}$, we arrive at the dynamic system

$$\dot{\boldsymbol{y}} = -[\nabla_{\boldsymbol{yy}} F(\boldsymbol{y};t)]^{-1} \nabla_{t\boldsymbol{y}} F(\boldsymbol{y};t), \tag{5}$$

where $\nabla_{t\boldsymbol{y}} F(\boldsymbol{y};t) \in \mathbb{R}^{np+1}$ and $\nabla_{\boldsymbol{yy}} F(\boldsymbol{y};t) \in \mathbb{R}^{np\times np}$ are the mixed partial derivative and Hessian of the objective function F, respectively. By sampling at sampling times t_k, for $k = 0, 1, 2, \ldots$ using a first-order forward Euler scheme for the relation in (5), the predicted variable $\boldsymbol{y}_{k+1|k}$ is given by

$$\boldsymbol{y}_{k+1|k} = \boldsymbol{y}_k - h\,[\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k)]^{-1} \nabla_{t\boldsymbol{y}} F(\boldsymbol{y}_k;t_k). \tag{6}$$

The predicted variable $\boldsymbol{y}_{k+1|k}$ computed as in (6) is corrected by a step of projected gradient descent with stepsize $\gamma > 0$

$$\boldsymbol{y}_{k+1} = P_Y\left[\boldsymbol{y}_{k+1|k} - \gamma \nabla_{\boldsymbol{y}} F(\boldsymbol{y}_{k+1|k}; t_{k+1})\right]. \tag{7}$$

Based on the definition of the objective function F in (1), the Hessian $\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k)$ can be written as

$$\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k) := \nabla_{\boldsymbol{yy}} \Phi(\boldsymbol{y}_k;t_k) + \nabla_{\boldsymbol{yy}} G(\boldsymbol{y}_k;t_k), \tag{8}$$

where $\nabla_{\boldsymbol{yy}} \Phi(\boldsymbol{y}_k;t_k) \in \mathbb{R}^{np\times np}$ is a block diagonal matrix formed by the Hessian of local functions ϕ^i. In other words, the i-th diagonal block of $\nabla_{\boldsymbol{yy}} \Phi(\boldsymbol{y}_k;t_k)$ is given by $\nabla_{\boldsymbol{y}^i\boldsymbol{y}^i} \phi^i(\boldsymbol{y}_k^i;t_k)$. Further, $\nabla_{\boldsymbol{yy}} G(\boldsymbol{y}_k;t_k) \in \mathbb{R}^{np\times np}$ has the sparsity pattern of the graph, since its ij-th block $[\nabla_{\boldsymbol{yy}} G(\boldsymbol{y}_k;t_k)]^{ij} \in \mathbb{R}^{p\times p}$ is not null if and only if $j = i$ or $j \in \mathcal{N}^i$. Combining these observations we obtain that the Hessian $\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k)$ has the sparsity pattern of the graph, therefore, it can be computed in a decentralized manner. Although, the objective function Hessian $\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k)$ is graph sparse, the inverse $\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k)$ which is required for the prediction step in (6) is not necessarily graph sparse and computable in a decentralized manner. In the following section we introduce a new algorithm that approximates the time varying Hessian inverse $[\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k)]^{-1}$ by a graph sparse matrix.

B. Decentralized Prediction-Correction tracking

To implement the prediction step in (6), the Hessian inverse $[\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k)]^{-1}$ is required, however, it is not necessarily graph sparse. This means that it cannot be computed only by 1-hop communication. To overcome this difficulty, we generalize a recently proposed distributed algorithm to approximate Hessian inverses up to an arbitrary accuracy [11], [12]. The approximations are obtained by truncating the Taylor expansion of the Hessian inverse. To be more precise, let $\mathbf{H}_k$ be the objective function Hessian $\nabla_{\boldsymbol{yy}} F(\boldsymbol{y};t)$ computed at time t_k for $\boldsymbol{y}_k$, i.e., $\mathbf{H}_k = \nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k)$. Further, define L as the largest eigenvalue of the positive semi-definite matrix $\nabla_{\boldsymbol{yy}} G(\boldsymbol{y}_k;t_k)$ [cfr. Assumption 3]. We write the Hessian as $\mathbf{H}_k := \mathbf{D}_k - \mathbf{B}_k$, where matrices $\mathbf{D}_k$ and $\mathbf{B}_k$ are defined as

$$\mathbf{D}_k := \nabla_{\boldsymbol{yy}} \Phi(\boldsymbol{y}_k;t_k) + L\,\mathbf{I}, \quad \mathbf{B}_k := L\,\mathbf{I} - \nabla_{\boldsymbol{yy}} G(\boldsymbol{y}_k;t_k). \tag{9}$$

By assuming strong convexity of the function Φ [cfr. Assumption 2], the matrix $\mathbf{D}_k$ is a positive definite block diagonal matrix and encodes the local effects; the matrix $\mathbf{B}_k$ is positive semidefinite by construction and has the same structure of the graph. By definition $\mathbf{H}_k = \mathbf{D}_k - \mathbf{B}_k$, given that $\mathbf{D}_k$ is a positive definite block diagonal matrix, the objective function Hessian $\mathbf{H}_k$ can be written as $\mathbf{H}_k = \mathbf{D}_k^{1/2}(\mathbf{I} - \mathbf{D}_k^{-1/2}\mathbf{B}_k\mathbf{D}_k^{-1/2})\mathbf{D}_k^{1/2}$. To compute the Hessian inverse $\mathbf{H}_k^{-1}$ we can use the Taylor series $(\mathbf{I} - \mathbf{X})^{-1} = \sum_{\tau=0}^{\infty} \mathbf{X}^\tau$ for $\mathbf{X} = \mathbf{D}_k^{-1/2}\mathbf{B}_k\mathbf{D}_k^{-1/2}$ to obtain

$$\mathbf{H}_k^{-1} = \mathbf{D}_k^{-1/2} \sum_{\tau=0}^{\infty} \left(\mathbf{D}_k^{-1/2}\mathbf{B}_k\mathbf{D}_k^{-1/2}\right)^\tau \mathbf{D}_k^{-1/2}. \tag{10}$$

The expansion in (10), since $\|\mathbf{D}_k^{-1/2}\mathbf{B}_k\mathbf{D}_k^{-1/2}\| < 1$ – see [9]. We introduce the Decentralized Prediction-Correction Tracking (DePCoT) as a decentralized algorithm that approximates the Hessian inverse $\mathbf{H}_k^{-1}$ in (6) by truncating the series in (10). The approximate Hessian inverse $\hat{\mathbf{H}}_{k,(K)}^{-1}$ for DePCoT with K level of approximation is defined by the first K+1 terms in (10) as

$$\hat{\mathbf{H}}_{k,(K)}^{-1} = \mathbf{D}_k^{-1/2} \sum_{\tau=0}^{K} \left(\mathbf{D}_k^{-1/2}\mathbf{B}_k\mathbf{D}_k^{-1/2}\right)^\tau \mathbf{D}_k^{-1/2}. \tag{11}$$

The Hessian inverse approximation in (11) follows that the prediction step of DePCoT can be written as

$$\boldsymbol{y}_{k+1|k} = \boldsymbol{y}_k - h\,\hat{\mathbf{H}}_{k,(K)}^{-1} \nabla_{t\boldsymbol{y}} F(\boldsymbol{y}_k;t_k) := \boldsymbol{y}_k - h\,\boldsymbol{d}_{k,(K)}, \tag{12}$$

where $\boldsymbol{d}_{k,(K)} := \hat{\mathbf{H}}_{k,(K)}^{-1} \nabla_{t\boldsymbol{y}} F(\boldsymbol{y}_k;t_k)$ is defined as the prediction direction of DePCoT for K level of approximation.

The predicted variable $\boldsymbol{y}_{k+1|k}$ of DePCoT at step $k+1$ is corrected by descending through the negative objective function gradient $\nabla_{\boldsymbol{y}} F(\boldsymbol{y}_{k+1|k};t_{k+1}) \in \mathbb{R}^{np}$. Therefore, the *correction step* of DePCoT is identical to (7) and given by

$$\boldsymbol{y}_{k+1} = P_Y\left[\boldsymbol{y}_{k+1|k} - \gamma \nabla_{\boldsymbol{y}} F(\boldsymbol{y}_{k+1|k}; t_{k+1})\right], \tag{13}$$

where $\gamma > 0$ is the stepsize and P_Y is the projection operator to the convex set $Y = Y^1 \times \cdots \times Y^n$ [cfr. Assumption 1].

The prediction and correction steps of DePCoT in (7) and (13), respectively, are implementable in a decentralized manner. To study this statement define the components $\boldsymbol{d}_{k,(K)}^i \in \mathbb{R}^p$ of DePCoT's prediction direction $\boldsymbol{d}_{k,(K)} = [\boldsymbol{d}_{k,(K)}^1; \ldots; \boldsymbol{d}_{k,(K)}^n] \in \mathbb{R}^{np}$. Observe that node i can compute its prediction direction $\boldsymbol{d}_{k,(K)}^i$ by exchanging information with its neighbors. To be more precise, the sequence of DePCoT prediction directions satisfies

$$\boldsymbol{d}_{k,(\tau)} = \mathbf{D}_k^{-1}\left(\mathbf{B}_k \boldsymbol{d}_{k,(\tau-1)} + \nabla_{t\boldsymbol{y}} F(\boldsymbol{y}_k;t_k)\right). \tag{14}$$

Considering the graph sparse structure of $\mathbf{B}_k$ and block diagonally of $\mathbf{D}_k$, the local components of the prediction directions are related to each other as

$$\boldsymbol{d}_{k,(\tau)}^i = (\mathbf{D}_k^{ii})^{-1}\Big(\sum_{j\in\mathcal{N}^i, j=i} \mathbf{B}_k^{ij} \boldsymbol{d}_{k,(\tau-1)}^j + \nabla_{t\boldsymbol{y}} F(\boldsymbol{y}_k;t_k)^i\Big). \tag{15}$$

Therefore, node i can compute its prediction direction $\boldsymbol{d}_{k,(\tau)}^i$ by having access to the prediction directions $\boldsymbol{d}_{k,(\tau-1)}^i$ of itself and its neighbors. By initializing the prediction directions at step k as $\boldsymbol{d}_{k,(0)}^i = [\hat{\mathbf{H}}_{k,(0)}^{-1}]^{ii} \nabla_{t\boldsymbol{y}} F(\boldsymbol{y}_k;t_k)^i = (\mathbf{D}_k^{ii})^{-1} \nabla_{t\boldsymbol{y}} F(\boldsymbol{y}_k;t_k)^i$, nodes can compute their K level prediction direction $\boldsymbol{d}_{k,(0)}$ by K times recursively computing (15). Notice that according to (2), the local

Algorithm 1 Approximate prediction direction for node i

```
Input: Gradient $\nabla_{t\boldsymbol{y}}F(\boldsymbol{y}_k;t_k)^i$, matrices $\mathbf{D}_k^i$ and $\mathbf{B}_k^{ij}$ for $j \in \mathcal{N}^i, j = i$
    0: Compute the initial prediction direction $\boldsymbol{d}_{k,(0)}^i = (\mathbf{D}_k^{ii})^{-1}\nabla_{t\boldsymbol{y}}F(\boldsymbol{y}_k;t_k)^i$
    for $\tau = 0, 1, \ldots, K-1$
        1: Exchange the prediction direction $\boldsymbol{d}_{k,(\tau)}^i$ with neighbors $j \in \mathcal{N}^i$
        2: Compute the updated prediction direction $\boldsymbol{d}_{k,(\tau+1)}^i$ as in (15)
    end
Output: Return the approximate prediction direction $\boldsymbol{d}_{k,(K)}^i$
```

Algorithm 2 DePCoT at node i

```
Require: Initial variable $\boldsymbol{y}_0^i \in \mathbb{R}^p$
for $k = 0, 1, 2, \ldots$
    1: Compute the block: $\mathbf{D}_k^{ii} = \nabla_{\boldsymbol{yy}}\Phi(\boldsymbol{y}_k;t_k) + L\,\mathbf{I}$
    2: Exchange the decision variable $\boldsymbol{y}_k^i$ with neighbors $j \in \mathcal{N}^i$
    3: Compute the blocks $\mathbf{B}_k^{ii}$ and $\mathbf{B}_k^{ij}$ as in (18) and (19)
    4: Compute the mixed derivative $\nabla_{t\boldsymbol{y}}F(\boldsymbol{y}_k;t_k)^i$ as in (16)
    5: Compute the prediction direction $\boldsymbol{d}_{k,(K)}^i$ by Algorithm 1
    6: Execute the prediction step: $\boldsymbol{y}_{k+1|k}^i = \boldsymbol{y}_k^i - h\,\boldsymbol{d}_{k,(K)}^i$.
    7: Exchange the predicted variable $\boldsymbol{y}_{k+1|k}^i$ with neighbors $j \in \mathcal{N}^i$
    8: Compute the gradient $\nabla_{\boldsymbol{y}}F(\boldsymbol{y}_{k+1|k};t_{k+1})^i$ as in (21)
    9: Execute the correction step: $\boldsymbol{y}_{k+1}^i = P_{Y^i}[\boldsymbol{y}_{k+1|k}^i - \gamma\nabla_{\boldsymbol{y}}F(\boldsymbol{y}_{k+1|k};t_{k+1})^i]$
end
```

component of mixed derivative $\nabla_{t\boldsymbol{y}}F(\boldsymbol{y}_k;t_k)^i = \nabla_{t\boldsymbol{y}}\Phi(\boldsymbol{y}_k;t_k)^i + \nabla_{t\boldsymbol{y}}G(\boldsymbol{y}_k;t_k)^i$ can be computed as

$$\nabla_{t\boldsymbol{y}}F(\boldsymbol{y}_k;t_k)^i = \nabla_{t\boldsymbol{y}^i}\phi^i(\boldsymbol{y}_k^i;t_k) + \nabla_{t\boldsymbol{y}^i}g^{i,i}(\boldsymbol{y}_k^i,\boldsymbol{y}_k^i;t_k) + 2\sum_{j\in\mathcal{N}^i}\nabla_{t\boldsymbol{y}^j}g^{i,j}(\boldsymbol{y}_k^i,\boldsymbol{y}_k^j;t_k), \tag{16}$$

which requires access to the decision variables $\boldsymbol{y}_k^i$ of the neighboring nodes $j \in \mathcal{N}^i$. Moreover, the local blocks of matrices $\mathbf{D}_k$, which is locally available, can be computed as

$$\mathbf{D}_k^{ii} := \nabla_{\boldsymbol{y}^i\boldsymbol{y}^i}\phi^i(\boldsymbol{y}_k^i;t_k) + L\,\mathbf{I}, \tag{17}$$

where $\mathbf{I} \in \mathbb{R}^{p\times p}$ is the identity matrix. By exchanging information with neighbors the diagonal block $\mathbf{B}_k^{ii}$ is computable as

$$\mathbf{B}_k^{ii} := L\,\mathbf{I} - \nabla_{\boldsymbol{y}^i\boldsymbol{y}^i}g^{i,i}(\boldsymbol{y}_k^i,\boldsymbol{y}_k^i;t_k) - \sum_{j\in\mathcal{N}^i}\nabla_{\boldsymbol{y}^i\boldsymbol{y}^i}g^{i,j}(\boldsymbol{y}_k^i,\boldsymbol{y}_k^j;t_k), \tag{18}$$

and the non-zero off-diagonal blocks are given by

$$\mathbf{B}_k^{ij} := L\,\mathbf{I} - 2\nabla_{\boldsymbol{y}^i\boldsymbol{y}^j}g^{i,j}(\boldsymbol{y}_k^i,\boldsymbol{y}_k^j;t_k), \quad \text{for } j \in \mathcal{N}^i. \tag{19}$$

Observe that nodes can compute (17), (18), and (19) only by access to the local $\boldsymbol{y}_k^i$ and neighboring $\boldsymbol{y}_k^j$ variables $j \in \mathcal{N}^i$.

The correction step (13) is also decentralized given the assumption on Y [cfr. Assumption 1]. In particular, By defining components $\nabla_{\boldsymbol{y}}F(\boldsymbol{y};t)^i \in \mathbb{R}^p$ of the objective function gradient $\nabla_{\boldsymbol{y}}F(\boldsymbol{y};t) = [\nabla_{\boldsymbol{y}}F(\boldsymbol{y};t)^1;\ldots;\nabla_{\boldsymbol{y}}F(\boldsymbol{y};t)^n] \in \mathbb{R}^{np}$, the update in (13) can be decomposed into local components as

$$\boldsymbol{y}_{k+1}^i = P_{Y^i}\left[\boldsymbol{y}_{k+1|k}^i - \gamma\,\nabla_{\boldsymbol{y}}F(\boldsymbol{y}_{k+1|k};t_{k+1})^i\right]. \tag{20}$$

Based on the relations in (1) and (3), the local component $\nabla_{\boldsymbol{y}}F(\boldsymbol{y}_{k+1|k};t_{k+1})^i$ of gradient is given by

$$\nabla_{\boldsymbol{y}}F(\boldsymbol{y}_{k+1|k};t_{k+1})^i = \nabla_{\boldsymbol{y}^i}\phi^i(\boldsymbol{y}_{k+1|k}^i;t_{k+1}) + 2\sum_{j=i,j\in\mathcal{N}^i}\nabla_{\boldsymbol{y}^j}g^{i,j}(\boldsymbol{y}_{k+1|k}^i,\boldsymbol{y}_{k+1|k}^j;t_{k+1}). \tag{21}$$

The DePCoT method is summarized in Algorithm 2, while in Algorithm 1 we have summarized the approximate prediction direction computation. As for Algorithm 2, steps 1-4 as well as 7-8 are preliminary communication and computation steps in order to compute the prediction and correction steps. Steps 5-6 contain the approximate prediction step, while step 9 implements the correction step. As for Algorithm 1, steps 0-1 are preliminary computation and communication steps, while step 2 computes the approximate prediction direction per node.

III. Convergence analysis

In this section, we study the convergence properties of DePCoT. We show that as time passes the sequence of variable $\boldsymbol{y}_k$ approaches a neighborhood of the optimal argument $\boldsymbol{y}^*(t_k)$. In proving the results we make the following assumptions.

Assumption 1: There exists a set $Y = Y^1 \times \cdots \times Y^n \subseteq \mathbb{R}^{np}$ whose interior contains the optimal argument trajectory $\boldsymbol{y}^*(t)$ of (1) for each t, i.e., $\boldsymbol{y}^*(t) \in \mathrm{int}(Y)$, for $t \geqslant 0$.

Assumption 2: The local functions ϕ^i are twice differentiable and the eigenvalues of the Hessians $\nabla_{\boldsymbol{y}^i\boldsymbol{y}^i}\phi^i(\boldsymbol{y}^i;t)$ are bounded by constants $0 < m$ and $M < \infty$. Therefore, the eigenvalues of the function $\Phi(\boldsymbol{y};t) := \sum_{i=1}^n \phi^i(\boldsymbol{y}^i;t)$ are bounded uniformly as

$$m\mathbf{I} \preceq \nabla_{\boldsymbol{yy}}\Phi(\boldsymbol{y};t) \preceq M\mathbf{I}. \tag{22}$$

Assumption 3: The functions $g^{i,j}(\boldsymbol{y}^i,\boldsymbol{y}^j;t)$ are twice differentiable and the eigenvalues of the aggregate function Hessian $\nabla_{\boldsymbol{yy}}G(\boldsymbol{y};t)$ are bounded by constants 0 and $L < \infty$,

$$\mathbf{0} \preceq \nabla_{\boldsymbol{yy}}G(\boldsymbol{y};t) \preceq L\mathbf{I}. \tag{23}$$

Assumption 4: The derivatives of the global cost $F(\boldsymbol{y};t)$ defined in (1) are bounded for all $\boldsymbol{y} \in Y$, $\forall t$ as

$$\|\nabla_{t\boldsymbol{y}}F(\boldsymbol{y};t)\| \leqslant C_0,\ \|\nabla_{\boldsymbol{yyy}}F(\boldsymbol{y};t)\| \leqslant C_1,\ \|\nabla_{\boldsymbol{y}t\boldsymbol{y}}F(\boldsymbol{y};t)\| \leqslant C_2. \tag{24}$$

Assumption 1 is a weak assumption and for the case that the set Y is $\mathbb{R}^{np}$, we only assume the existence of a solution for (1) at each time t. However, it is very useful in practice, when we know a priori that the solution trajectory has to be, for instance, positive. The bounds on the eigenvalues of Hessians $\nabla_{\boldsymbol{yy}}\Phi(\boldsymbol{y};t)$ and $\nabla_{\boldsymbol{yy}}G(\boldsymbol{y};t)$ in Assumptions 2 and 3, respectively, follow that the eigenvalues of the global cost Hessian $\nabla_{\boldsymbol{yy}}F(\boldsymbol{y};t)$ are uniformly bounded as $m\,\mathbf{I} \preceq \nabla_{\boldsymbol{yy}}F(\boldsymbol{y};t) \preceq (L+M)\,\mathbf{I}$. This bound besides guaranteeing that Problem (1) is strongly convex and has a *unique* solution for each time instance, implies that the Hessian $\nabla_{\boldsymbol{yy}}F(\boldsymbol{y};t)$ is invertible. Conditions imposed on the higher order derivatives of F in Assumption 4 are often required in time-varying optimization to prove convergence [5], [7], [8]. In the following theorem we show linear convergence of the sequence of variables $\boldsymbol{y}_k$ to a neighborhood of $\boldsymbol{y}^*(t_k)$.

Theorem 1: Consider the DePCoT algorithm defined in (9)-(21). Let Assumptions 1-4 hold and define ρ and σ as

$$\rho := \left(1 + \gamma^2(L+M)^2 - \gamma m\right)^{1/2},\ \sigma := 1 + h\left[\frac{C_0C_1}{m^2} + \frac{C_2}{m}\right]. \tag{25}$$

If the sampling increment h and the stepsize $\gamma > 0$ are chosen properly to satisfy the condition $\rho\sigma < 1$, then the sequence $\boldsymbol{y}_k$ converges Q-linear to $\boldsymbol{y}^*(t_k)$ up to a bounded error as

$$\|\boldsymbol{y}_k - \boldsymbol{y}^*(t_k)\| \leqslant (\rho\sigma)^k\|\boldsymbol{y}_0 - \boldsymbol{y}^*(t_0)\| + \rho\frac{h\Gamma(\varrho) + O(h^2)}{1-\rho}, \tag{26}$$

where the function Γ and the parameter ϱ are defined as

$$\Gamma(\varrho) = \frac{C_0 \varrho^{K+1}}{m(m+L)(1-\varrho)}, \qquad \varrho = \frac{L}{m+L}. \tag{27}$$

Theorem 1 states that the sequence of variables $\boldsymbol{y}_k$ generated by DePCoT converges linearly to a neighborhood of $\boldsymbol{y}(t_k)$ where the error bound is proportional to $\Gamma(\varrho)h + O(h^2)$. Hence, for any level K of Hessian inverse approximation in DePCoT the error bound of order $O(h)$ is achievable. In addition, by choosing large enough approximation level K we can decrease $\Gamma(\varrho)$ in (27) and push towards the order of h to get the error bound of order $O(h^2)$. In particular, if K is chosen as $K \geqslant \lceil \log h / \log \varrho - 1 \rceil$, we obtain $O(h^2)$ asymptotical error bound, which is smaller relative to the $O(h)$ error bound of running algorithms.

IV. Numerical evaluation

We consider a wireless sensor network estimating the intensity of a two dimensional spatial circular wave. The location of the source is $\boldsymbol{\xi}^0 = [1.2; 1.2]$, while its space-time intensity at any location and at any time is

$$c(\boldsymbol{\xi}; t) = \cos(2\pi\omega(t - \|\boldsymbol{\xi} - \boldsymbol{\xi}^0\|/v))/(4\pi\|\boldsymbol{\xi} - \boldsymbol{\xi}^0\|), \tag{28}$$

where $\boldsymbol{\xi} \in \mathbb{R}^2$ is the space location, while ω and v are the frequency and velocity of the wave, respectively. Sensors are located in the positions $\boldsymbol{\xi}^i$ and estimate the intensity as,

$$\hat{c}(\boldsymbol{\xi}^i; t) = c(\boldsymbol{\xi}^i; t) + \eta^i, \quad \eta^i \sim \mathcal{N}(0, q), \tag{29}$$

with q is a given noise covariance. We formulate the estimation as a least-squares problem with spatial regularizer,

$$\min_{y^1,\dots,y^n} \sum_{i=1}^{n} \Big(\frac{1}{2q}\|y^i - \hat{c}(\boldsymbol{\xi}^i; t)\|_2^2 + \frac{\beta}{q}\sum_{j\in\mathcal{N}^i} \frac{w^{ij}}{2}\|y^i - y^j\|_2^2 \Big) \tag{30}$$

for which, $\beta > 0$ is a tuning parameter, and the regularization term serves to push closely located sensors to estimate similar intensities. For this purpose the weight w^{ij} is chosen as $w^{ij} = \exp(-\alpha\|\boldsymbol{\xi}^i - \boldsymbol{\xi}^j\|)/\delta$, where the parameter δ is the maximum degree of the nodes of the network, and $\alpha = -\log(0.5)/d$, where d is the maximum communication range distance.

In this numerical example, we consider $n = 500$ sensor nodes located in the square $[-1, 1]^2$. We set $d = 2\sqrt{2}/\sqrt{n}$. The other parameters are: $\omega = 0.1$, $v = 0.05$, $\sqrt{q} = h^3$, and $\delta = 13$. To see (30) as an instance of (2), it is sufficient to equate

$$f^i(y^i, t) = \frac{1}{2}(1 - \beta w^{ii})\|y^i\|_2^2 - \hat{c}(\boldsymbol{\xi}^i; t)y^i, \quad w^{ii} = \sum_{j\in\mathcal{N}_i} w^{ij},$$

$$g^{i,i}(y^i; t) = \beta\frac{w^{ii}}{2}\|y^i\|_2^2, \; g^{i,j}(y^i, y^j; t) = \beta\frac{w^{ij}}{2}\|y^i - y^j\|_2^2, \tag{31}$$

and it can be seen that Assumptions 1-4 hold. In the simulations, we use $\beta = 0.25$, and with this $m \geqslant 0.75, L \leqslant 1.5, C_0 = 0.17, C_1 = 0, C_2 = 0, C_3 = 0.11$. We compare the performance of different algorithms for solving (30). In particular, *(i)* A running gradient method (meaning Algorithm 2 without the prediction step); *(ii)* Our Algorithm 2 with K communication steps for the computation of the approximate Hessian inverse; *(iii-iv)* The running dual decomposition method and the running ADMM algorithm of [7] and [8], respectively, adapted to our networked scenario, where we perform dual decomposition/ADMM instead of gradient descent in the correction step (no prediction).

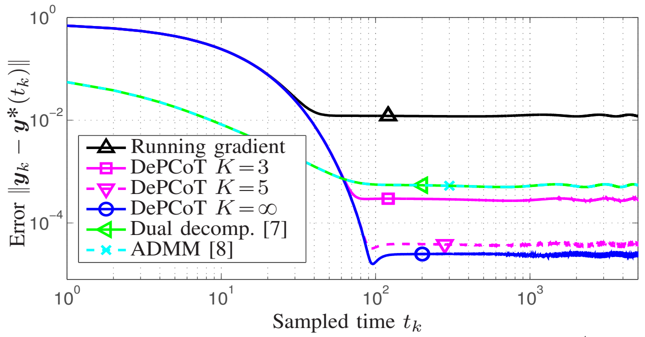

Fig. 1. Error with respect to the sampled time t_k for $h = 1/320$.

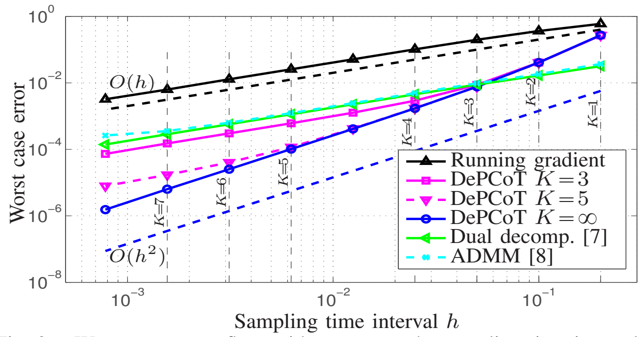

Fig. 2. Worst case error floor with respect to the sampling time interval h.

Figure 1 displays the error w.r.t. the time instance for the different methods. Observe that the tracking methods have significant better performance relative to the running methods. Figure 2 depicts the worst case error floor size, defined as $\max_{k\geqslant\bar{k}} \|\boldsymbol{y}_k - \boldsymbol{y}^*(t_k)\|$ with $\bar{k} = 2000$. In particular, we retrieve the theoretical results of Theorem 1. We have also plotted vertical lines to indicate which K would ensure an $O(h^2)$ error bound.

References

[1] J. Koshal, A. Nedić, and U. Y. Shanbhag, "Multiuser Optimization: Distributed Algorithms and Error Analysis," *SIAM Journal on Optimization*, vol. 21, no. 3, pp. 1046 – 1081, 2011.
[2] A. Beck, A. Nedić, A. Ozdaglar, and M. Teboulle, "An $O(1/k)$ Gradient Method for Network Resource Allocation Problems," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pp. 64 – 73, 2014.
[3] B. T. Polyak, *Introduction to Optimization*. Optimization Software, Inc., 1987.
[4] V. M. Zavala and M. Anitescu, "Real-Time Nonlinear Optimization as a Generalized Equation," *SIAM Journal of Control and Optimization*, vol. 48, no. 8, pp. 5444 – 5467, 2010.
[5] A. L. Dontchev, M. I. Krastanov, R. T. Rockafellar, and V. M. Veliov, "An Euler-Newton Continuation method for Tracking Solution Trajectories of Parametric Variational Inequalities," *SIAM Journal of Control and Optimization*, vol. 51, no. 51, pp. 1823 – 1840, 2013.
[6] S.-Y. Tu and A. H. Sayed, "Mobile Adaptive Networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 649 – 664, 2011.
[7] F. Y. Jakubiec and A. Ribeiro, "D-MAP: Distributed Maximum a Posteriori Probability Estimation of Dynamic Systems," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 450 – 466, 2013.
[8] Q. Ling and A. Ribeiro, "Decentralized Dynamic Optimization Through the Alternating Direction Method of Multipliers," *IEEE Transactions on Signal Processing*, vol. 62, no. 5, pp. 1185 – 1197, 2014.
[9] A. Simonetto, A. Mokhtari, A. Koppel, G. Leus, and A. Ribeiro, "Prediction-Correction Methods for Distributed Time-Varying Convex Optimization," *In preparation*, 2015, available at http://ens.ewi.tudelft.nl/~asimonetto/TimeVarying_part2.pdf.
[10] A. Simonetto, A. Koppel, A. Mokhtari, G. Leus, and A. Ribeiro, "Prediction-Correction Methods for Time-Varying Convex Optimization," in *Proceedings of the Asilomar Conference on Signals, Systems, and Computers (submitted)*, October 2015.
[11] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton-Part I: Algorithm and Convergence," *arXiv preprint arXiv:1504.06017*, 2015.
[12] ——, "Network Newton-Part II: Convergence Rate and Implementation," *arXiv preprint arXiv:1504.06020*, 2015.