A Quasi-Newton Prediction-Correction Method for Decentralized Dynamic Convex Optimization

Andrea Simonetto*, Alec Koppel†, Aryan Mokhtari†, Geert Leus*, and Alejandro Ribeiro†

***Abstract*—We introduce DeNT, a decentralized Newton-based tracking algorithm that solves and track the solution trajectory of continuously varying networked convex optimization problems. DeNT is derived from the prediction-correction methodology, by which the time-varying optimization problem is sampled at discrete time instances and then a sequence is generated via alternatively executing predictions on how the optimizers are changing and corrections on how they actually have changed. Prediction is based on the sample dynamics of the optimality conditions, while correction is based on a Newton method. After presenting DeNT, we show how it can be implemented in a decentralized way and analyze its convergence. We extend it to cases in which the knowledge on how the optimization programs are changing in time is only approximate, proposing DeANT. We then present an application to a resource allocation problem in a wireless network, demonstrating the proposed method outperforms existing methods by orders of magnitude, and exhibits a trade-off between convergence accuracy, sampling interval, and network communications.**

I. Introduction

We consider unconstrained convex optimization problems whose objective function $F(\boldsymbol{y};t)$ changes continuously in time (indicated as t) and its components are available at different nodes of a network. We assume that the objective function can be decomposed into two parts: the first part is a sum of locally available functions at the nodes and the second part is shared between neighboring nodes. This probem formulation has been studied in multiuser network optimization [1], control [2], and robotics [3], [4].

To solve time-varying optimization problems, one may sample them at discrete time instances t_k, $k = 0, 1, 2, \ldots$, and then solve each time-invariant instance of the problem, with objective $F(\boldsymbol{y};t_k)$. If the sampling period $h := t_{k+1} - t_k$ is chosen arbitrarily small, then one could find the solution trajectory $\boldsymbol{y}^*(t)$ with arbitrary accuracy. However solving such problems for each sampling time is not a viable option in most application domains, since the computation time for each optimizer would exceed the rate at which the solution trajectory changes, unless $\boldsymbol{y}^*(t)$ is approximately stationary. Moreover, in the decentralized case, the communications requirement for solving each instance of the time-varying problem would also yield an unacceptable latency. In short, the majority of distributed algorithms for convex problems with static objective function may not be easily extended to handle time-varying objectives, with the exception of the case in which the changes in the objective occur sufficiently slow [1], [5].

The work in this paper is supported in part by STW under the D2S2 project from the ASSYS program (project 10561) and in part by NSF CAREER CCF-0952867, and ONR N00014-12-1-0997.

*Andrea Simonetto and Geert Leus are with the Department of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2826 CD Delft, The Netherlands. Email: {a.simonetto, g.j.t.leus}@tudelft.nl.

†Aryan Mokhtari, Alec Koppel, and Alejandro Ribeiro are with the Department of Electrical and Systems Engineering, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104, USA. Email: {akoppel, aryanm, aribeiro}@seas.upenn.edu.

Decentralized running algorithms, which run at the same time-scale of the optimization problem and dynamically react to changes in the objective function, have been shown to converge to a neighborhood of the true optimizer $\boldsymbol{y}^*(t_k)$, despite the fact that only one round of communication is allowed per discrete time step [6]–[13]. The aforementioned works tackle a wide array of optimization problem classes (but mainly strong convex objective functions with no constraints). Notably [10] and [11] describe a running dual decomposition and a running alternating direction method of multiplier (ADMM) algorithm. It is important to note that, as we will show, all these methods are bounded to converge up to a bounded error of size $O(h)$, since they do not leverage information about how the cost function changes with time and only react to the variation of the objective function.

Prediction-correction algorithms [14], making use of tools in *non-stationary* optimization [15]–[18], are developed to iteratively solve convex programs which continuously vary in time. These methods operate by predicting the optimal solution at the discrete time instance t_{k+1} via an approximation of the variation of the objective function $F(\boldsymbol{y};t)$ from t_k to t_{k+1}, and then correcting the predicted solution by executing projected gradient or Newton descent. However, these methods are not applicable to decentralized optimization problems since each iteration requires access to Hessian inverse of the objective function $F(\boldsymbol{y};t)$ which is a global computation.

In [19], we have extended the methods in [14] to handle the decentralized computation of the Hessian inverse, but we have only considered gradient-based correction steps. In this paper,

i) We present a decentralized prediction-correction algorithm that has a Newton correction step (DeNT), thereby enabling better convergence and tracking performance;

ii) Under suitable conditions we show that the asymptotical error bound can be made arbitrarily close to $O(h^4)$, which is the centralized bound. The error bound depends specifically on the number of communication rounds each node has to perform per time step, and we display trade-offs between accuracy and communication burden.

iii) Finally, we consider cases in which the prediction of how the cost function changes in time is made via the estimated knowledge of its derivative in time, which is necessary in most practical settings.

Interesting related work in distributed time-varying optimization, where however the authors employ a *continuous-time* algorithmic strategy, can be found, e.g., in [20]–[22].

II. Problem Formulation

We consider a connected undirected network $\mathcal{G} = (V, E)$, with vertex set V containing n nodes associated with which is a decision variable $\boldsymbol{y}^i \in \mathbb{R}^p$. Moreover E denotes the edge set E of the network. Further define $\boldsymbol{y} = [\boldsymbol{y}^1; \ldots; \boldsymbol{y}^n] \in \mathbb{R}^{np}$ as the concatenation of the decision variables. We focus on problems where nodes aim at cooperatively minimizing the global *smooth strongly convex* cost function $F : \mathbb{R}^{np} \times \mathbb{R}_+ \to \mathbb{R}$, which can be written as the sum of a locally available function $f : \mathbb{R}^{np} \times \mathbb{R}_+ \to \mathbb{R}$, and a function $g : \mathbb{R}^{np} \times \mathbb{R}_+ \to \mathbb{R}$ induced by the network structure $\mathcal{G}$. In particular, the objective function $f(\boldsymbol{y}; t)$ is the sum of locally available functions as

$$f(\boldsymbol{y}; t) := \sum_{i \in V} f^i(\boldsymbol{y}^i; t) \ , \tag{1}$$

while the network related objective function $g(\boldsymbol{y}; t)$ has the form

$$g(\boldsymbol{y}; t) := \sum_{i \in V} g^{i,i}(\boldsymbol{y}^i; t) + \sum_{(i,j) \in E} g^{i,j}(\boldsymbol{y}^i, \boldsymbol{y}^j; t) \ . \tag{2}$$

We study the problem

$$\boldsymbol{y}^*(t) := \underset{\boldsymbol{y} \in \mathbb{R}^{np}}{\operatorname{argmin}} \ F(\boldsymbol{y}; t) := f(\boldsymbol{y}; t) + g(\boldsymbol{y}; t), \quad \text{for } t \geqslant 0. \tag{3}$$

Our goal is to enable the nodes to determine their own component of the solution $\boldsymbol{y}^*(t)$ of (3) for each time t in a decentralized fashion, i.e. a protocol such that each node only requires communication with neighboring nodes. Notice that nodes can minimize the objective function $f(\boldsymbol{y}; t)$ independently, while minimization of the objective function $g(\boldsymbol{y}; t)$ requires coordination and exchanging information between neighboring nodes. Examples of the formulation (3) are approximate distributed optimization, resource allocation problems, and estimation of spatially distributed processes, as reported in [19], [23] [1].

III. Algorithm Development

In order to solve the time-varying optimization problem in (3), the first step is sampling the continuously varying objective function $F(\boldsymbol{y}; t)$ at time instants t_k with $k = 0, 1, 2, \ldots$. This leads to a sequence of time-invariant problems

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

Having access to unlimited computational capabilities and allowing the nodes to exchange an arbitrarily number of messages between consequent time instances t_k and t_{k+1}, one could solve the problems (4) with high accuracy at each time instance. However, if the sampling period $h := t_{k+1} - t_k$ is small, this is usually very challenging even for moderate size networks. Therefore, we propose a tracking algorithm that generates a sequence of approximate optimizers for (4), i.e., $\{\boldsymbol{y}_k\}$ which asymptotically converge to the true optimizer of (4), $\boldsymbol{y}^*(t_k)$, up to a bounded tracking error. Formally, one would like to generate a sequence $\{\boldsymbol{y}_k\}$ for which

$$\limsup_{k \to \infty} \|\boldsymbol{y}_k - \boldsymbol{y}^*(t_k)\| = \varphi(h, \kappa) \ , \tag{5}$$

where φ is a function of the sampling period h and the average number of exchanged messages per node per time instance κ.

In order to generate such a sequence, we make use of prediction-correction methods [14], which we subsequently describe. Starting from an *arbitrary* initial condition $\boldsymbol{y}_0$, for each time $k \geqslant 0$, these methods operate by predicting a new approximate optimizer as

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

and then correct this predicted vector as,

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_{k+1|k} - \gamma \, \boldsymbol{c}_{k+1} \ , \tag{7}$$

for a certain correction direction $\boldsymbol{c}_{k+1} \in \mathbb{R}^{np}$ and nonnegative constant stepsize $\gamma > 0$.

The prediction direction is generated based on the information available at time t_k only, and it is based on the idea to keep the suboptimality of the predicted $\boldsymbol{y}_{k+1|k}$ w.r.t. $\boldsymbol{y}^*(t_{k+1})$ as close as possible to the suboptimality of $\boldsymbol{y}_k$ w.r.t. $\boldsymbol{y}^*(t_k)$. The correction direction is then selected as an approximate Newton step, and it is based on the information available at time t_{k+1}.

Since we consider cases where distinct nodes of a network are charged with the task of collaboratively minimizing the global objective F with only local information, we turn to extending the updates in (6) - (7) to allow for distributed implementation. We turn to approximating the Hessian inverse computation in (6) - (7) in the following subsections.

A. Hessian inverse approximation

Even in the case that the sparsity pattern of the Hessian has the same sparsity pattern as the graph, its inverse $[\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k; t_k)]^{-1}$ is not sparse in general and hence its computation requires global coordination. To derive a decentralized protocol in [19] we build upon a recently proposed technique algorithm to approximate Hessian inverses in a decentralized way which may be made arbitrarily accurate [24], [25]. The approximation is obtained by truncating the

[1]**Notation.** Vectors are written as $\boldsymbol{y} \in \mathbb{R}^n$ and matrices as $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. We use $\|\cdot\|$ to denote the Euclidean norm, both in the case of vectors, matrices, and tensors. The gradient of the function $F(\boldsymbol{y}; t)$ with respect to $\boldsymbol{y}$ at the point $(\boldsymbol{y}, t)$ is indicated as $\nabla_{\boldsymbol{y}} F(\boldsymbol{y}; t) \in \mathbb{R}^n$, while the partial derivative of the same function w.r.t. t at $(\boldsymbol{y}, t)$ is written as $\nabla_t F(\boldsymbol{y}; t) \in \mathbb{R}$. Similarly, the notation $\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}; t) \in \mathbb{R}^{n \times n}$ denotes the Hessian of $F(\boldsymbol{y}; t)$ w.r.t. $\boldsymbol{y}$ at $(\boldsymbol{y}, t)$, whereas $\nabla_{t\boldsymbol{y}} F(\boldsymbol{y}; t) \in \mathbb{R}^n$ denotes the partial derivative of the gradient of $F(\boldsymbol{x}; t)$ w.r.t. the time t at $(\boldsymbol{y}, t)$, i.e. the mixed first-order partial derivative vector of the objective. This notation is then consistent for higher derivatives.

Taylor expansion of the Hessian inverse.[2] In particular, define $\text{diag}[\nabla_{\boldsymbol{yy}} g(\boldsymbol{y}_k;t_k)]$ as the matrix whose diagonal is the one of the block diagonally dominant matrix $\nabla_{\boldsymbol{yy}} g(\boldsymbol{y}_k;t_k)$ [cf. Assumption 2]. We can write the Hessian as

$$\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k) = \mathbf{D}_k - \mathbf{B}_k \ , \tag{8}$$

where the matrices $\mathbf{D}_k$ and $\mathbf{B}_k$ are defined as

$$\mathbf{D}_k := \nabla_{\boldsymbol{yy}} f(\boldsymbol{y}_k;t_k) + \text{diag}[\nabla_{\boldsymbol{yy}} g(\boldsymbol{y}_k;t_k)] \ , \tag{9a}$$

$$\mathbf{B}_k := \text{diag}[\nabla_{\boldsymbol{yy}} g(\boldsymbol{y}_k;t_k)] - \nabla_{\boldsymbol{yy}} g(\boldsymbol{y}_k;t_k) \ . \tag{9b}$$

When the function f is strongly convex, the matrix $\mathbf{D}_k$ is a positive definite block diagonal matrix and encodes local network relationships. Moreover, the matrix $\mathbf{B}_k$ has the same structure of the graph. Furthermore, given that $\mathbf{D}_k$ is a positive definite block diagonal matrix, the objective function Hessian $\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k)$ can be alternatively written as

$$\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k) = \mathbf{D}_k^{1/2}(\mathbf{I} - \mathbf{D}_k^{-1/2}\mathbf{B}_k\mathbf{D}_k^{-1/2})\mathbf{D}_k^{1/2}. \tag{10}$$

Consider the Taylor series expansion of the second term inside the parentheses on the right-hand side of (10)

$$(\mathbf{I} - \mathbf{X})^{-1} = \sum_{\tau=0}^{\infty} \mathbf{X}^\tau, \quad \text{for } \mathbf{X} = \mathbf{D}_k^{-1/2}\mathbf{B}_k\mathbf{D}_k^{-1/2} \ . \tag{11}$$

Pre-multiplying and post-multiplying the expression in (11) by $\mathbf{D}_k^{-1/2}$ yields the infinite series representation of the Hessian inverse

$$[\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k)]^{-1} = \mathbf{D}_k^{-1/2} \sum_{\tau=0}^{\infty} \left(\mathbf{D}_k^{-1/2}\mathbf{B}_k\mathbf{D}_k^{-1/2}\right)^\tau \mathbf{D}_k^{-1/2}. \tag{12}$$

We introduce the decentralized prediction approach as a decentralized algorithm that approximates the Hessian inverse $[\nabla_{\boldsymbol{yy}} F(\boldsymbol{y}_k;t_k)]^{-1}$ in (6) by truncating the series in (12). The approximate Hessian inverse $\mathbf{H}_{k,(K)}^{-1}$ with K level of approximation is defined by the first K+1 terms in (12) as

$$\mathbf{H}_{k,(K)}^{-1} = \mathbf{D}_k^{-1/2} \sum_{\tau=0}^{K} \left(\mathbf{D}_k^{-1/2}\mathbf{B}_k\mathbf{D}_k^{-1/2}\right)^\tau \mathbf{D}_k^{-1/2}. \tag{13}$$

From the Hessian inverse approximation in (13), it follows that the prediction step can be written as

$$\boldsymbol{y}_{k+1|k} = \boldsymbol{y}_k - h\,\mathbf{H}_{k,(K)}^{-1}\nabla_{t\boldsymbol{y}} F(\boldsymbol{y}_k;t_k) =: \boldsymbol{y}_k - h\,\boldsymbol{d}_{k,(K)}, \tag{14}$$

where

$$\boldsymbol{d}_{k,(K)} := \mathbf{H}_{k,(K)}^{-1}\nabla_{t\boldsymbol{y}} F(\boldsymbol{y}_k;t_k) \tag{15}$$

is defined as the prediction direction for K level of approximation.

In the following proposition, we show that the direction $\boldsymbol{d}_{k,(K)}$ can be obtained in a decentralized way via $K+1$ rounds of communication among neighboring nodes.

[2]Note that in [19], we have introduced a different decomposition. Here we change the decomposition to obtain better performance, see [23] for a comprehensive discussion.

Proposition 1: Node i may compute its K order approximate prediction direction $\boldsymbol{d}_{k,(K)}^i$ by initializing it as

$$\boldsymbol{d}_{k,(0)}^i = [\mathbf{H}_{k,(0)}^{-1}]^{ii}\nabla_{t\boldsymbol{y}} F^i(\boldsymbol{y}_k;t_k) = (\mathbf{D}_k^{ii})^{-1}\nabla_{t\boldsymbol{y}} F^i(\boldsymbol{y}_k;t_k) \ , \tag{16}$$

and making use of the prediction directions $\boldsymbol{d}_{k,(\tau-1)}^i$ of itself and its neighbors, and recursing over $\tau = 0, \dots, K-1$

$$\boldsymbol{d}_{k,(\tau+1)}^i = (\mathbf{D}_k^{ii})^{-1}\Big(\sum_{j\in N^i\cup\{i\}} \mathbf{B}_k^{ij}\boldsymbol{d}_{k,(\tau)}^j + \nabla_{t\boldsymbol{y}} F^i(\boldsymbol{y}_k;t_k)\Big) \ , \tag{17}$$

where matrices $\nabla_{t\boldsymbol{y}} F^i(\boldsymbol{y}_k;t_k)$, $\mathbf{D}_k^{ii}$, and $\mathbf{B}_k^{ij}$ are defined as

$$\nabla_{t\boldsymbol{y}} F^i(\boldsymbol{y}_k;t_k) = \nabla_{t\boldsymbol{y}^i} f^i(\boldsymbol{y}_k^i;t_k) + \nabla_{t\boldsymbol{y}^i} g^{i,i}(\boldsymbol{y}_k^i;t_k) + \sum_{j\in N^i} \nabla_{t\boldsymbol{y}^i} g^{i,j}(\boldsymbol{y}_k^i,\boldsymbol{y}_k^j;t_k), \tag{18}$$

$$\mathbf{D}_k^{ii} := \nabla_{\boldsymbol{y}^i\boldsymbol{y}^i} f^i(\boldsymbol{y}_k^i;t_k) + \nabla_{\boldsymbol{y}^i} g^{i,i}(\boldsymbol{y}_k^i;t_k) \ , \tag{19}$$

$$\mathbf{B}_k^{ij} := -\nabla_{\boldsymbol{y}^i\boldsymbol{y}^j} g^{i,j}(\boldsymbol{y}_k^i,\boldsymbol{y}_k^j;t_k), \quad \text{for } j \in N^i. \tag{20}$$

Proof: By direct computation. ■

Making use of the result in Proposition 1, at time t_k, node i executes the local update

$$\boldsymbol{y}_{k+1|k}^i = \boldsymbol{y}_k^i - h\,\boldsymbol{d}_{k,(K)}^i \tag{21}$$

where $\boldsymbol{d}_{k,(K)}^i$ is computed as in (17), and only requires K communication exchanges between neighboring nodes. We next discuss how to alleviate the need for exact information about how the objective is varying in time.

B. Time derivative approximation

In most practical settings, knowledge of how the function F changes in time is unavailable. In such situations, to still be able to use our algorithms, we may estimate the term $\nabla_{t\boldsymbol{y}} F(\boldsymbol{y};t)$ via a first-order backward scheme. In particular, let $\tilde{\nabla}_{t\boldsymbol{y}} F_k$ be an approximate version of $\nabla_{t\boldsymbol{y}} F(\boldsymbol{y}_k;t_k)$. We compute $\tilde{\nabla}_{t\boldsymbol{y}} F_k$ as a first-order backward finite difference, as

$$\tilde{\nabla}_{t\boldsymbol{y}} F_k = \frac{1}{h}(\nabla_{\boldsymbol{y}} F(\boldsymbol{y}_k;t_k) - \nabla_{\boldsymbol{y}} F(\boldsymbol{y}_k;t_{k-1})). \tag{22}$$

The approximation $\tilde{\nabla}_{t\boldsymbol{y}} F_k$ requires only information of the first previous step. With this, we approximate the prediction direction as

$$\tilde{\boldsymbol{d}}_{k,(K)} := \mathbf{H}_{k,(K)}^{-1}\tilde{\nabla}_{t\boldsymbol{y}} F_k, \tag{23}$$

which can also be computed in a decentralized way with $K+1$ rounds of communication, in a similar fashion as shown in the proof of Proposition 1.

C. Decentralized correction step

The predicted variable $\boldsymbol{y}_{k+1|k}$, obtained with only local information is then corrected via (7). The correction direction can be computed as,

$$\boldsymbol{c}_{k+1} = -\boldsymbol{A}_{k+1}^{-1}\nabla_{\boldsymbol{y}} F(\boldsymbol{y}_{k+1|k};t_{k+1}), \tag{24}$$

where $\boldsymbol{A}_{k+1} \in \mathbb{R}^{np\times np}$ is a nonsingular "preconditioning" correction matrix, while $\nabla_{\boldsymbol{y}} F(\boldsymbol{y}_{k+1|k};t_{k+1})$ is the gradient of the cost function computed at the time t_{k+1} for the predicted

Algorithm 1 DeNT: Decentralized Newton Tracking at node i

```
Require: Initial variable $\boldsymbol{y}_0$, objective $F(\boldsymbol{y};t_0)$, $h$, $K$, $K'$.
 1: for $k = 0, 1, 2, \ldots$ do
 2:     Compute local prediction direction $\boldsymbol{d}^i_{k,(K)}$ by initializing as (16).
 3:     for $\tau = 0, 1, 2, \ldots, K-1$ do
 4:         Exchange predicted direction $\boldsymbol{d}^i_{k,(\tau+1)}$ with neighboring nodes $j \in N^i$ and execute the recursion (17)
 5:     end for
 6:     Predict the trajectory $\boldsymbol{y}^i_{k+1|k} = \boldsymbol{y}^i_k - h\,\boldsymbol{d}^i_{k,(K)}$
 7:     Acquire the updated function $F(\boldsymbol{y};t_{k+1})$
 8:     Compute the correction direction $\boldsymbol{c}_{k+1,(K')}$ by initializing as in a decentralized fashion as in
            $\boldsymbol{c}^i_{k+1,(0)} = [\mathbf{H}^{-1}_{k+1|k,(0)}]^{ii}\nabla_{\boldsymbol{yy}}F^i(\boldsymbol{y}_{k+1|k};t_{k+1})$
 9:     for $\tau = 0, 1, 2, \ldots, K'-1$ do
10:         Exchange correction step $\boldsymbol{c}^i_{k,(\tau+1)}$ with neighboring nodes $j \in N^i$ and execute the recursion $\boldsymbol{c}^i_{k+1,(\tau+1)} = (\mathbf{D}^{ii}_{k+1})^{-1}\Big(\sum_{j\in N^i\cup\{i\}} \mathbf{B}^{ij}_{k+1}\boldsymbol{c}^j_{k+1,(\tau)} + \nabla_{\boldsymbol{y}}F^i(\boldsymbol{y}_{k+1|k};t_{k+1})\Big)$
11:     end for
12:     Correct the trajectory prediction $\boldsymbol{y}^i_{k+1} = \boldsymbol{y}^i_{k+1|k} - \boldsymbol{c}^i_{k+1,(K')}$
13: end for
```

vector $\boldsymbol{y}_{k+1|k}$. Different choices of correction matrix give rise to different correction steps. The case $\boldsymbol{A}_{k+1} = \mathbf{I}$ corresponds to DeGT (decentralized gradient tracking) and is studied in [19].

To accelerate this method, we may select the preconditioning matrix to incorporate to second-order information of the objective as $\boldsymbol{A}^{-1}_{k+1} = \mathbf{H}^{-1}_{k+1|k,(K')}$, where $\mathbf{H}^{-1}_{k+1|k,(K')}$ is the K' level approximation of the inverse of the Hessian $\nabla_{\boldsymbol{yy}}F(\boldsymbol{y}_{k+1|k}, t_{k+1})$ evaluated at the time t_{k+1} for the predicted vector $\boldsymbol{y}_{k+1|k}$. With this choice and a unitary stepsize $\gamma = 1$ we obtain

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_{k+1|k} - \mathbf{H}^{-1}_{k+1|k,(K')}\nabla_{\boldsymbol{y}}F(\boldsymbol{y}_{k+1|k};t_{k+1}), \tag{25}$$

which is a K' level approximate Newton step. This step is also computable in a decentralized fashion. In practice, one can use the same algorithm for the prediction direction to compute the term $\mathbf{H}^{-1}_{k+1|k,(K')}\nabla_{\boldsymbol{y}}F(\boldsymbol{y}_{k+1|k};t_{k+1})$, where now the gradient takes the place of the time derivative. A similar analysis shows that (25) can be computed via $K'+1$ rounds of communication among neighbors.

Making use of this fact, we may derive a decentralized protocol for node i to execute its correction step, which is stated as

$$\boldsymbol{y}^i_{k+1} = \boldsymbol{y}^i_{k+1|k} - \gamma\,\boldsymbol{c}^i_{k+1,(K')}\ , \tag{26}$$

and requires only K' information exchanges with neighboring nodes $j \in N_i$. The updates in (21) and (26) taken together we call decentralized Newton tracking (DeNT), and are summarized in Algorithm 1. When the time-derivative estimate in (22) is used instead of its exact counterpart to compute the prediction step $\tilde{\boldsymbol{d}}^i_{k,(K)}$, the approximate version of (21) is referred to as decentralized approximate Newton Tracking (DeANT). We study the convergence properties of these methods in the following section.

IV. Convergence analysis

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

$$m\mathbf{I} \ \preceq\ \nabla_{\boldsymbol{yy}}f(\boldsymbol{y};t)\ \preceq\ M\mathbf{I}. \tag{27}$$

Assumption 2: The functions $g^{i,i}(\boldsymbol{y}^i;t)$ and $g^{i,j}(\boldsymbol{y}^i,\boldsymbol{y}^j;t)$ are twice differentiable. The aggregate function Hessian $\nabla_{\boldsymbol{yy}}g(\boldsymbol{y};t)$ is block diagonally dominant and the largest (smallest) eigenvalue of its block diagonal is upper bounded (lower bounded) by $L/2 < \infty$ ($\ell/2 > 0$).

Assumption 3: The derivatives of the global cost $F(\boldsymbol{y};t)$ defined in (3) are bounded for all $\boldsymbol{y} \in \mathbb{R}^{np}$, $\forall t$ as

$$\|\nabla_{\boldsymbol{t}\boldsymbol{y}}F(\boldsymbol{y};t)\| \leqslant C_0,\ \|\nabla_{\boldsymbol{yyy}}F(\boldsymbol{y};t)\| \leqslant C_1,\ \|\nabla_{\boldsymbol{yty}}F(\boldsymbol{y};t)\| \leqslant C_2. \tag{28}$$

From the bounds on the eigenvalues of Hessians $\nabla_{\boldsymbol{yy}}f(\boldsymbol{y};t)$ and $\nabla_{\boldsymbol{yy}}g(\boldsymbol{y};t)$ in Assumptions 1 and 2, respectively, it follows that the eigenvalues of the global cost Hessian $\nabla_{\boldsymbol{yy}}F(\boldsymbol{y};t)$ are uniformly bounded as

$$m\,\mathbf{I}\ \preceq\ \nabla_{\boldsymbol{yy}}F(\boldsymbol{y};t)\ \preceq\ (L+M)\,\mathbf{I}. \tag{29}$$

The smoothness and regularity conditions we require are standard in the analysis of time-varying optimization methods [10], [11], [18]. Besides guaranteeing that Problem (3) is strongly convex and has a *unique* solution for each time instance, these assumptions imply that the Hessian $\nabla_{\boldsymbol{yy}}F(\boldsymbol{y};t)$ is invertible[3].

Theorem 1: Consider the DeNT and DeANT algorithms. Let Assumptions 1-3 hold and fix K and K' as the Hessian inverse approximation levels for the prediction and correction steps, respectively. Let the function $\Gamma : \mathbb{N} \to \mathbb{R}$ be defined as

$$\Gamma(q) = \frac{C_0}{m}\left(\frac{L/2}{m+L/2}\right)^{q+1}. \tag{30}$$

There exist bounds $\bar{K}$, $\bar{h}$, and $\bar{R}$, such that if the sampling period h is chosen as $h \leqslant \bar{h}$, K and K' are chosen as $K, K' \geqslant \bar{K}$, and the initial optimality gap satisfies $\|\boldsymbol{y}_0 - \boldsymbol{y}^*(t_0)\| \leqslant \bar{R}$, then $\{\boldsymbol{y}_k\}$ converges exponentially to the solution trajectory $\boldsymbol{y}^*(t_k)$ up to a bounded error as

$$\begin{aligned}\limsup_{k\to\infty}\|\boldsymbol{y}_k - \boldsymbol{y}^*(t_k)\| \leqslant{}& O(h\Gamma(K)\Gamma(K'))+ \\ & O(h^2(\Gamma(K') + \Gamma(K)^2)) + O(h^3\Gamma(K)^2) + O(h^4).\end{aligned} \tag{31}$$

Proof: See [23]. ■

Theorem 1 says that DeNT and DeANT converge to a bounded tracking error defined in (31) once the algorithm reaches an attractor region (a detailed characterization of such an attraction region and its local nature is deferred to [23]). The error bound in (31) depends, as expected, on the sampling period h and the approximation levels K and K'. In the worst case, the asymptotic error floor will be of the

[3]Future research will be focused on weakening some of the assumptions in the context of non-smooth optimization, as preliminary tackled in [26].

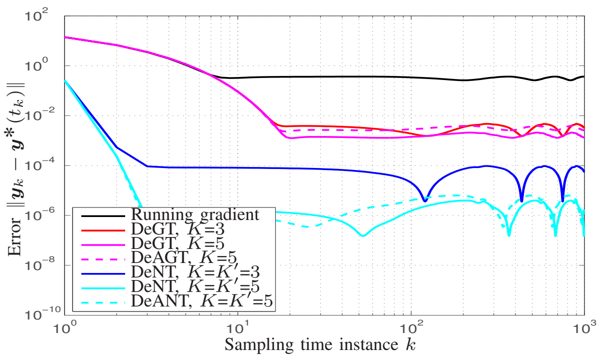

Fig. 1. Error with respect to the sampling time instance k for different algorithms applied to a continuous-time sensor network resource allocation problem (33), with $h = 0.1$.

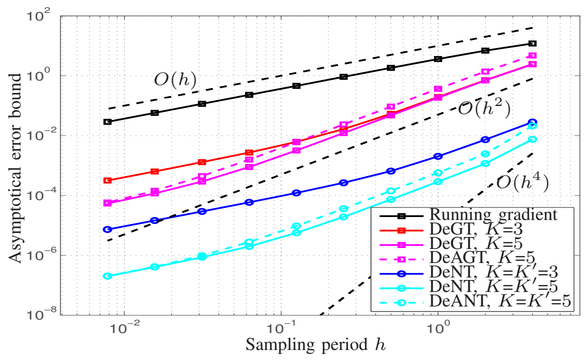

Fig. 2. Comparison of asymptotic error bound $\max_{k>\bar{k}} \{\|\boldsymbol{y}_k - \boldsymbol{y}^*(t_k)\|\}$ with sampling period h. Black dotted lines represent sampling period error bounds $O(h^r)$ for $r = 1, 2, 4$.

order $O(h)$. However, in some cases we may achieve tighter tracking guarantees. In particular, when the terms containing the function Γ are negligible, that is when K and K' are big enough, then the asymptotic error bound can achieve the $O(h^4)$ dependence, similarly to the centralized methods for this problem class [14]. This implies a trade-off between accuracy and communication effort.

V. Numerical evaluation

To evaluate the empirical performance of the algorithms derived in Section III, we consider a wireless sensor network aiming to collaboratively solve a time-varying resource allocation problem. This problem has been investigated by a number of authors in the time-invariant case – for instance, see [27], [28]. However, due e.g. to varying battery levels, or importance of each sensors when estimating spatially distributed time-varying fields, it makes sense to consider time-varying elements. Let $f^i : \mathbb{R}^p \times \mathbb{R}_+$ be a time-varying strongly convex smooth loss function associated with node i in a network of interconnected nodes $\mathcal{G}$. The function f^i encodes the quality of the transmission of node i, whose decision variable is denoted as $\boldsymbol{y}^i \in \mathbb{R}^p$. The resource allocation problem associated with this sensor network may be formulated in its time-varying version as

$$\underset{\boldsymbol{y}^1\in\mathbb{R}^p,\ldots,\boldsymbol{y}^n\in\mathbb{R}^p}{\text{minimize}} \sum_{i\in V} f^i(\boldsymbol{y}^i;t) \tag{32a}$$
$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{y} = \mathbf{b}(t), \tag{32b}$$

where $\boldsymbol{y} \in \mathbb{R}^{np}$ is the stacked version of all the local decision variables $\boldsymbol{y}_i$, while the matrix $\boldsymbol{A} \in \mathbb{R}^{l\times np}$ and the time-varying vector $\mathbf{b}(t) \in \mathbb{R}^l$ describe the resource allocation constraints, as in network flow formulations. In (32), $\boldsymbol{A} \in \mathbb{R}^{lp\times np}$ denotes the augmented graph edge incidence matrix. The matrix $\boldsymbol{A}$ is formed by $l \times n$ square blocks of dimension p. If the edge $e = (j,k)$ with $j < k$ links node j to node k the block (e,j) is $[\boldsymbol{A}]_{ej} = \mathbf{I}_p$ and the block $[\boldsymbol{A}]_{ek} = -\mathbf{I}_p$, where $\mathbf{I}_p$ denotes the identity matrix of dimension p. All other blocks are identically null. Moreover, the time-varying vectors $\mathbf{b}(t) \in \mathbb{R}^{lp}$ are induced by channel capacity and rate transmission constraints.

In cases where the computational capability of the individual sensors is limited, it is advantageous to solve (32) in an approximate and decentralized way. One way to do so is to make use of an approximate augmented Lagrangian approach as in [29], and solve the unconstrained problem

$$\underset{\boldsymbol{y}^1\in\mathbb{R}^p,\ldots,\boldsymbol{y}^n\in\mathbb{R}^p}{\text{minimize}} \sum_{i\in V} f^i(\boldsymbol{y}^i;t) + \frac{1}{\beta^2}\|\boldsymbol{A}\boldsymbol{y} - \mathbf{b}(t)\|^2 . \tag{33}$$

The parameter $\beta > 0$ tunes the approximation level. Given the structure of $\boldsymbol{A}$, problem (33) is an instance of (3).

As in [27], Section 5, we select the local functions $f^i(y^i;t)$ associated with each node i as

$$f^i(y^i;t) = \frac{1}{2}a^i(y^i - c^i(t))^2 + \log\left[1 + \exp\left(b^i(y^i - d^i(t))\right)\right], \tag{34}$$

where $y^i \in \mathbb{R}$, and $a^i, b^i, c^i(t), d^i(t)$ are (time-varying) scalar parameters. The second order derivative of f^i in y^i is bounded above by $a^i + (b^i)^2/4$ and below by a^i. We set each a^i and b^i to be draw from a uniform random distribution in $[1,2]$ and $[-2,2]$, respectively. We choose $c^i(t)$ and $d^i(t)$ as the time-varying functions

$$c^i(t) = 10\cos(\theta_c^i + \omega\, t), \quad \theta_c^i \sim \mathcal{U}[0,2\pi), \tag{35a}$$
$$d^i(t) = 10\cos(\theta_d^i + \omega\, t), \quad \theta_d^i \sim \mathcal{U}[0,2\pi), \tag{35b}$$

where we set $\omega = 0.1$. The sensors in the $n = 50$ node wireless network can communicate if they are closer than a certain distance, and in particular we set the maximum communication range as $r = 2.5\sqrt{2}/\sqrt{n}$. This generates a network of l links. We set $\mathbf{b} = \mathbf{0}$, while the approximation level $\beta = \sqrt{20}$. We can compute the bounds in the Assumptions 1-3 as

$$m = 1,\ M = 2,\ L = 1.4,\ C_0 = 2,\ C_1 = 0.8,\ C_2 = 0.8. \tag{36}$$

We analyze the behavior of DeNT and DeANT w.r.t. the decentralized running gradient method of [12], and the decentralized DeGT and DeAGT of [19]. In Figure 1, we depict how the different algorithms reach convergence as time passes for sampling interval $h = 0.1$. Observe that the time-approximation in DeAGT and DeANT does not degrade significantly the asymptotical error, while the number of communication K and K' play a more dominant role (especially in the case of DeNT).

We also observe this trend in Figure 2, where we analyze the behavior varying the sampling period h. We approximate the asymptotical error bound as

$$\max_{k > \bar{k}} \{\|\boldsymbol{y}_k - \boldsymbol{y}^*(t_k)\|\}, \tag{37}$$

for a given $\bar{k}$ (in the simulation results $\bar{k}$ is either 800 for $h \geqslant 1/16$ or 2000 for $h < 1/16$). We observe that, as expected by the analysis in Section IV, the running gradient has an asymptotical error that goes as $O(h)$, DeGT has one that varies between $O(h)$ and $O(h^2)$ depending on the approximation level K and h, and DeNT has one that varies between $O(h)$ and $O(h^4)$.

VI. Conclusion

We considered continuously varying convex programs whose objectives may be decomposed into two parts: a sum of locally available functions at the nodes and the a part that is shared between neighboring nodes. To solve this problem and track the solution trajectory, we proposed a distributed iterative procedure which samples the problem at discrete times. Each node predicts where the solution trajectory will be at the next time via an approximation procedure in which it communicates with its neighbors, and then corrects this prediction by incorporating information about how the local objective is varying, again via a decentralized local approximation. We developed an extension of this tool which allows for the case that the dynamical behavior of the objective must be estimated.

We established that this decentralized approximate second-order procedure converges to an asymptotic error bound which depends on the length of the sampling interval and the amount communications of the network. Moreover, we established that this convergence result also applies to the case where time derivatives must be approximated.

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] P. Ögren, E. Fiorelli, and N. Leonard, "Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment," *IEEE Transactions on Automatic Control*, vol. 49, no. 8, pp. 1292 – 1302, 2004.

[3] F. Arrichiello, *Coordination Control of Multiple Mobile Robots*. PhD thesis, Università degli studi di Cassino, 2006.

[4] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*. Applied Mathematics Series, Princeton University Press, 2008.

[5] 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.

[6] P. Braca, S. Marano, V. Matta, and P. Willett, "Asymptotic Optimality of Running Consensus in Testing Binary Hypotheses," *IEEE Transactions on Signal Processing*, vol. 58, no. 2, pp. 814 – 825, 2010.

[7] F. S. Cattivelli and A. H. Sayed, "Diffusion Strategies for Distributed Kalman Filtering and Smoothing," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2069 – 2084, 2010.

[8] 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.

[9] M. M. Zavlanos, A. Ribeiro, and G. J. Pappas, "Network Integrity in Mobile Robotic Networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 3 – 18, 2013.

[10] 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.

[11] 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.

[12] A. Simonetto and G. Leus, "Distributed Asynchronous Time-Varying Constrained Optimization," in *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, (Pacific Grove, USA), November 2014.

[13] A. Simonetto and G. Leus, "Double Smoothing for Time-Varying Distributed Multi-user Optimization," in *Proceedings of the IEEE Global Conference on Signal and Information Processing*, (Atlanta, US), December 2014.

[14] A. Simonetto, A. Mokhtari, A. Koppel, G. Leus, and A. Ribeiro, "A Class of Prediction-Correction Methods for Time-Varying Convex Optimization," *Submitted http://arxiv.org/abs/1509.05196*, 2015.

[15] B. T. Polyak, *Introduction to Optimization*. Optimization Software, Inc., 1987.

[16] 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.

[17] A. L. Dontchev and R. T. Rockafellar, *Implicit Functions and Solution Mappings*. Springer, 2009.

[18] 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.

[19] A. Simonetto, A. Mokhtari, A. Koppel, G. Leus, and A. Ribeiro, "A Decentralized Prediction-Correction Method for Networked Time-Varying Convex Optimization," in *Proceedings of the 6th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, (Cancun, Mexico), December 2015.

[20] S. Rahili and W. Ren, "Distributed convex optimization for continuous-time dynamics with time-varying cost functions," *arXiv preprint arXiv:1507.04878*, 2015.

[21] M. Fazlyab, S. Paternain, V. M. Preciado, and A. Ribeiro, "Interior point method for dynamic constrained optimization in continuous time," *arXiv preprint arXiv:1510.01396*, 2015.

[22] M. Ye and G. Hu, "Distributed Optimization for Systems with Time-Varying Quadratic Objective Functions," in *Proceedings of the 54th IEEE Conference on Decision and Control*, (Osaka, Japan), December 2015.

[23] A. Simonetto, A. Koppel, A. Mokhtari, G. Leus, and A. Ribeiro, "Decentralized Prediction-Correction Methods for Networked Time-Varying Convex Optimization," *Submitted http://arxiv.org/abs/1602.01716*, 2016.

[24] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton-Part I: Algorithm and Convergence," *arXiv preprint arXiv:1504.06017*, 2015.

[25] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton-Part II: Convergence Rate and Implementation," *arXiv preprint arXiv:1504.06020*, 2015.

[26] A. Simonetto and G. Leus, "On Non-Differentiable Time-Varying Optimization," in *Proceedings of the 6th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, (Cancun, Mexico), December 2015.

[27] L. Xiao and S. Boyd, "Optimal Scaling of a Gradient Method for Distributed Resource Allocation," *Journal of Optimization Theory and Applications*, vol. 129, no. 3, pp. 469 – 488, 2006.

[28] E. Ghadimi, I. Shames, and M. Johansson, "Multi-Step Gradient Methods for Networked Optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 21, pp. 5417 – 5429, 2013.

[29] P. Wan and M. D. Lemmon, "Event-Triggered Distributed Optimization in Sensor Networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, (San Francisco, US), pp. 49 – 60, April 2009.